БИБЛИОТЕКА СПЕЦИАЛИСТА ПО ВНЕДРЕНИЮ

А. А. Асатрян, А. Б. Голиков,
А. Н. Морозов, Д. Ю. Соломатин,
Ю. А. Федоров

МЕТОДИЧЕСКОЕ ПОСОБИЕ ПО ЭКСПЛУАТАЦИИ КРУПНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ НА ПЛАТФОРМЕ «1С:ПРЕДПРИЯТИЕ 8»

Издание 2

1С ПАБЛИШИНГ

**А. А. Асатрян, А. Б. Голиков, А. Н. Морозов,
Д. Ю. Соломатин, Ю. А. Федоров**

Методическое пособие по эксплуатации крупных информационных систем на платформе «1С:Предприятие 8». Издание 2

Электронная книга в формате pdf; ISBN 978-5-9677-2672-9.

Электронный аналог издания "Методическое пособие по эксплуатации крупных информационных систем на платформе «1С:Предприятие 8». Издание 2" (ISBN 978-5-9677-2735-1, М.: ООО "1С-Паблишинг", 2017; артикул печатной книги по прайс-листу фирмы "1С": 4601546133908; по вопросам приобретения печатных изданий издательства "1С-Паблишинг" обращайтесь к партнеру "1С", обслуживающему вашу организацию, или к другим партнерам фирмы "1С".)

Данная книга адресована специалистам, занимающимся эксплуатацией крупных информационных систем, построенных с использованием технологической платформы «1С:Предприятие», и представляет собой набор методик и подходов по обеспечению технологического качества информационных систем. При подготовке материалов использован практический опыт работы команд эксплуатации крупных информационных систем на проектах Центров компетенции по технологическим вопросам фирмы «1С» (ЦКТП).
В книге содержатся теоретические сведения об организации эксплуатации и описываются практические методы, приводится разбор настроек различных компонентов информационной системы. Кроме того, описываются базовые методики локализации и расследования проблем, приводятся примеры решения конкретных прикладных задач эксплуатации. Авторы постарались учесть отзывы читателей первого издания книги, дополнить и переработать материал для формирования более целостной картины процесса эксплуатации. В данном издании также добавлены примеры практик эксплуатации, используемых на реальных крупных внедрениях.
Рекомендуется в качестве методического материала при подготовке к аттестации «1С:Эксплуататор крупных информационных систем».

Оглавление

Введение

В настоящее время наблюдается существенный рост числа информационных систем, построенных с использованием технологической платформы «1С:Предприятие», с большим числом одновременно работающих пользователей. Также постепенно появляются новые и развиваются существующие технологии построения крупных информационных систем. Наиболее показательным является, например, активное развитие облачных решений, построенных по технологии 1cFresh.

При эксплуатации таких крупных и архитектурно сложных систем на первый план выходят вопросы обеспечения технологического качества информационной системы.

Высокие требования по доступности диктуют необходимость быстро реагировать на возникающие проблемы и обладать компетенциями в смежных областях. Необходимо уметь быстро применять знания и понимать основные принципы работы сложных информационных систем.

Необходимость оперативного обновления без потери качества, а также необходимость прозрачно для конечного пользователя решать вопросы масштабирования крупных информационных систем, в свою очередь, предъявляют особые требования к организации процесса эксплуатации, построения архитектуры и инфраструктуры, а также к квалификации специалистов, занимающихся эксплуатацией.

Для того чтобы максимально эффективно решать вопросы по эксплуатации крупных систем, построенных с использованием технологической платформы «1С:Предприятие», качественно настроить инфраструктуру, мониторинг, программные продукты, специалисту необходимо понимать основные аспекты организации, устройства систем, построенных на технологиях «1С:Предприятия», и принципы их функционирования, а также владеть основными навыками по администрированию и эксплуатации таких информационных систем. Специалист такого уровня должен обладать знаниями о ключевых настройках продуктов, а также о возможностях расследования проблем, которые в них могут возникать. Эти знания должны стать базовыми для решения повседневных задач по эксплуатации информационных систем, значительно повысить технологическое качество их работы – в особенности систем, построенных по технологии 1cFresh.

В данном учебном пособии изложены ключевые моменты, связанные с настройкой программного обеспечения для эффективной работы системы на базе платформы «1С:Предприятие», изложены базовые методики для расследования проблем доступности и производительности, а также вопросы организации эксплуатации.

Данное пособие предназначено для слушателей курса «1С:Эксплуатация крупных информационных систем», а также специалистов, занимающихся эксплуатацией информационных систем, построенных с использованием технологической платформы «1С:Предприятие». Рекомендуется в качестве методического материала при подготовке к аттестации «1С:Эксплуататор крупных информационных систем».

Скрипты, опубликованные в книге, можно скачать по ссылке:
http://v8.1c.ru/expert/materials/materials.zip

ГЛАВА 1

Технология 1cFresh с точки зрения эксплуатации

Использование технологии 1cFresh

Наиболее полным и сложным технологическим примером в части задач эксплуатации является эксплуатация информационных систем, построенных по технологии 1cFresh, которую разработала фирма «1С». Эта технология позволяет организовывать работу с прикладными решениями «1С:Предприятия» через Интернет в модели сервиса.

Прежде всего это технология, которая позволяет реализовать принцип multitenancy на платформе «1С:Предприятие». **Multitenancy** – подход, при котором один экземпляр программного обеспечения обслуживает большое количество арендаторов.

Работа в модели сервиса позволяет реализовать для прикладных решений «1С:Предприятия» бизнес-модель продажи и использования программного обеспечения, известную как SaaS (software as a service – «программное обеспечение как услуга»).

Пользователи могут работать с теми прикладными решениями, которые подключены их абоненту. Причем у каждого из них может быть индивидуальный набор приложений, доступ к которым им предоставил владелец абонента (см. рис. 1).

Существует два основных варианта использования технологии 1cFresh: «для других» и «для себя».

Вариант «для других» подразумевает, что поставщик сервиса предоставляет услуги посторонним для него людям.

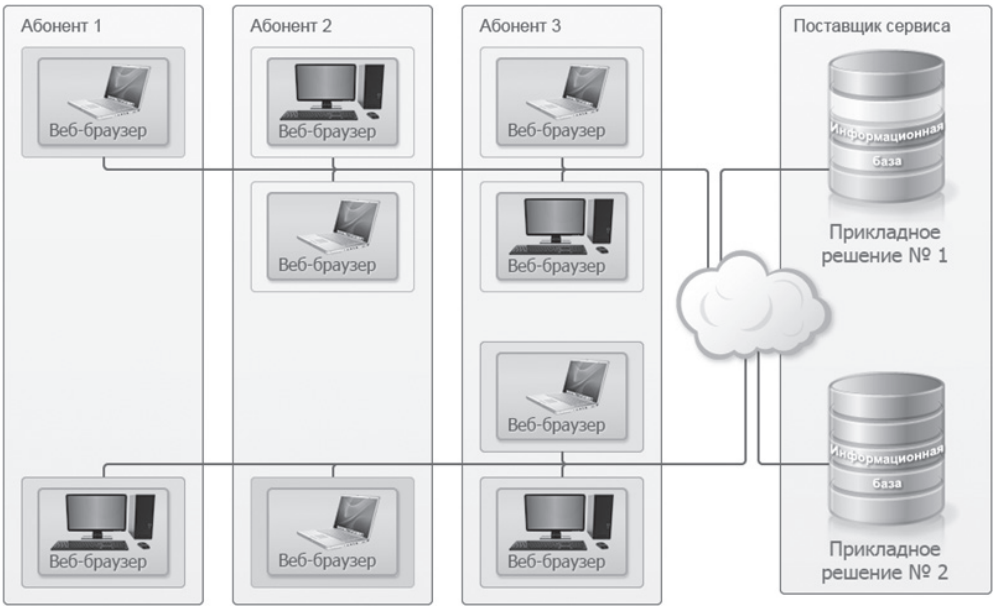


Рисунок 1. Схема работы пользователей в модели сервиса

Например, технологию 1cFresh может использовать организация – поставщик сервиса, чтобы предоставить малым и средним предприятиям доступ через Интернет к прикладным решениям на платформе «1С:Предприятие». При этом каждый сотрудник (независимо от принадлежности к организации) является пользователем сервиса.

Вариант «для себя» подразумевает, что поставщик сервиса предоставляет услуги своим собственным сотрудникам.

Например, технологию 1cFresh могут использовать холдинг или государственное учреждение, включающие много подразделений, для автоматизации и учета своей деятельности. При этом каждый сотрудник холдинга (независимо от принадлежности к подразделению) является пользователем сервиса.

Компоненты технологии

Технология 1cFresh состоит из нескольких основных компонентов, которые представлены на следующей схеме (рисунок 2).

Пользователи взаимодействуют с сервисом с помощью двух Java-приложений: сайта и форума.

- Сайт имеет простой и удобный интерфейс. Он позволяет пользователям запускать приложения, подключать новые приложения, регистрироваться в сервисе, просматривать новости и справочную информацию, и т. д.

- Форум предоставляет возможность пользователям сервиса общаться друг с другом и с представителями поставщика сервиса по вопросам функционирования сервиса и прикладных решений.

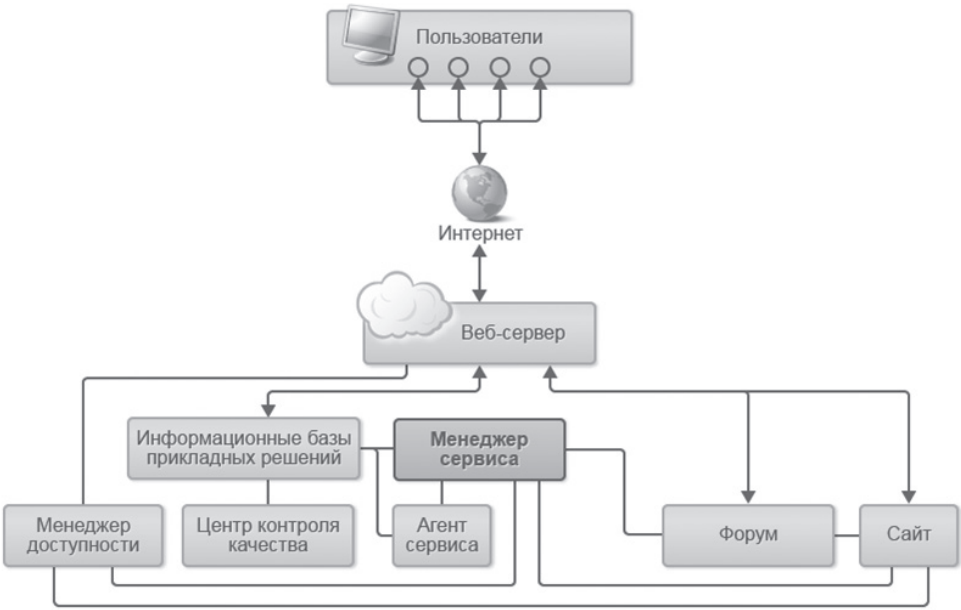


Рисунок 2. Компоненты технологии 1cFresh

Информационные базы прикладных решений разворачиваются в кластере серверов «1С:Предприятия» и публикуются на веб-сервере. Это главный прикладной компонент технологии, непосредственно с которым работают пользователи.

- **Менеджер сервиса** – это центральный и единственный обязательный компонент сервиса. Это прикладное решение на платформе «1С:Предприятие». Менеджер сервиса хранит в себе всю информацию о том, какие прикладные решения зарегистрированы в сервисе, какие области данных и какими абонентами используются, какие пользователи и с какими правами существуют в сервисе. Также менеджер сервиса хранит и предоставляет в случае необходимости прикладным решениям единую нормативно-справочную информацию, которая может централизованно обновляться. Имеется возможность редактировать необходимую информацию в соответствии с правами доступа пользователя менеджера сервиса. Менеджер сервиса координирует взаимодействие всех компонентов сервиса. Возможности:
  - регистрация нового абонента;
  - добавление, изменение, удаление пользователей;
  - добавление приложения из списка приложений в приложения абонента и удаление приложения;
  - назначение доступа пользователей абонента к приложениям абонента;

- создание резервных копий приложений, в т. ч. самостоятельная настройка автоматического резервного копирования;
- выгрузка данных приложения;
- запуск приложения.

- **Агент сервиса** – еще один компонент технологии. Это прикладное решение на платформе «1С:Предприятие». С помощью агента сервиса выполняются административные действия, для которых необходимо непосредственное соединение с кластером серверов. Агент сервиса обновляет прикладные решения, выполняет конвертацию данных прикладных решений из одной версии в другую, а также другие действия.
- **Менеджер доступности** – это также прикладное решение на платформе «1С:Предприятие». Менеджер доступности хранит информацию о недоступности ресурсов сервиса и позволяет выводить сообщения о недоступности пользователям сайта и форума даже в том случае, если все другие компоненты сервиса не функционируют.

Совместно с компонентами технологии может использоваться отдельное прикладное решение «Центр контроля качества». Оно помогает выполнять мониторинг системы и в случае обнаружения проблем оповещать ответственных по почте и SMS.

На рисунке 3 приведена упрощенная схема компонентов технологии на примере сервиса 1cFresh.com.

На самом деле связка «Интернет – веб-сервер – информационные базы» выглядит сложнее. В публичных высоконагруженных сервисах основная нагрузка приходится на информационные базы. Поэтому технология 1cFresh обеспечивает горизонтальное масштабирование путем добавления шлюзов приложений и дополнительных единиц масштабирования (см. рисунок 4).

Сервис позволяет использовать несколько информационных баз (и, как следствие, баз данных на сервере СУБД) для работы с одним и тем же прикладным решением. В этом случае задействуется еще один компонент сервиса – **шлюз приложений**. Он отвечает за горизонтальное масштабирование сервиса, обеспечивая единый внешний адрес конкретного прикладного решения (URL) независимо от того, к какой именно информационной базе в результате подключается абонент (перенаправляет пользователей сервиса на предназначенные им серверы с прикладными решениями). Шлюз написан на Java. Все шлюзы приложений регистрируются в менеджере сервиса, а он, в свою очередь, обладает информацией обо всех прикладных решениях и о том, какие области данных какими абонентами используются.

Добавление в сервис новых информационных баз выполняется в составе **единиц масштабирования (нод)**. Это позволяет упростить развертывание и администрирование. Единица масштабирования развертывается как единый модуль и содержит компоненты, необходимые для обработки обращений к информационным базам. В состав каждой единицы масштабирования входят кластер серверов «1С:Предприятия», сервер СУБД, на котором хранятся данные информационных баз,

а также один или два (для обеспечения отказоустойчивости) веб-сервера, обрабатывающих HTTP-обращения к информационным базам единицы масштабирования (см. рис. 5).

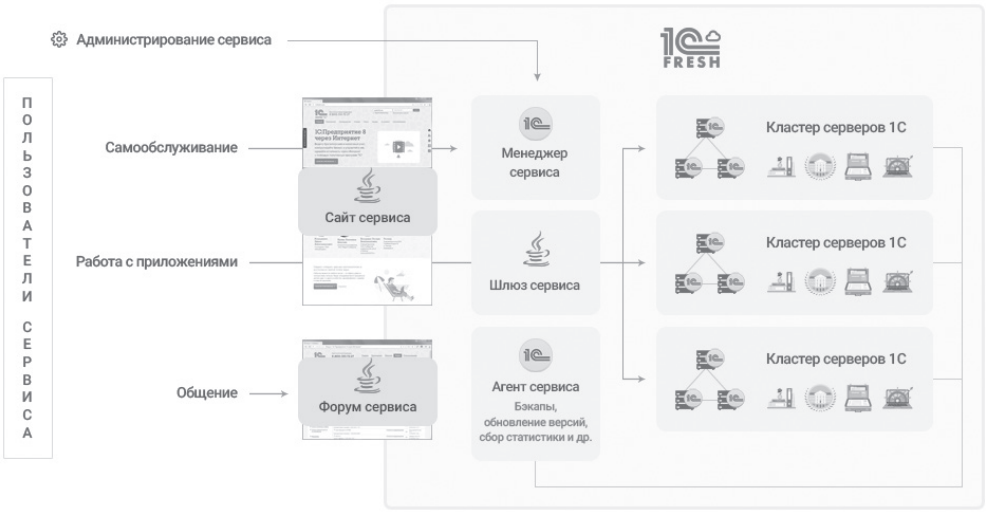


Рисунок 3. Упрощенная схема сервиса 1cFresh
(представлены не все компоненты сервиса)

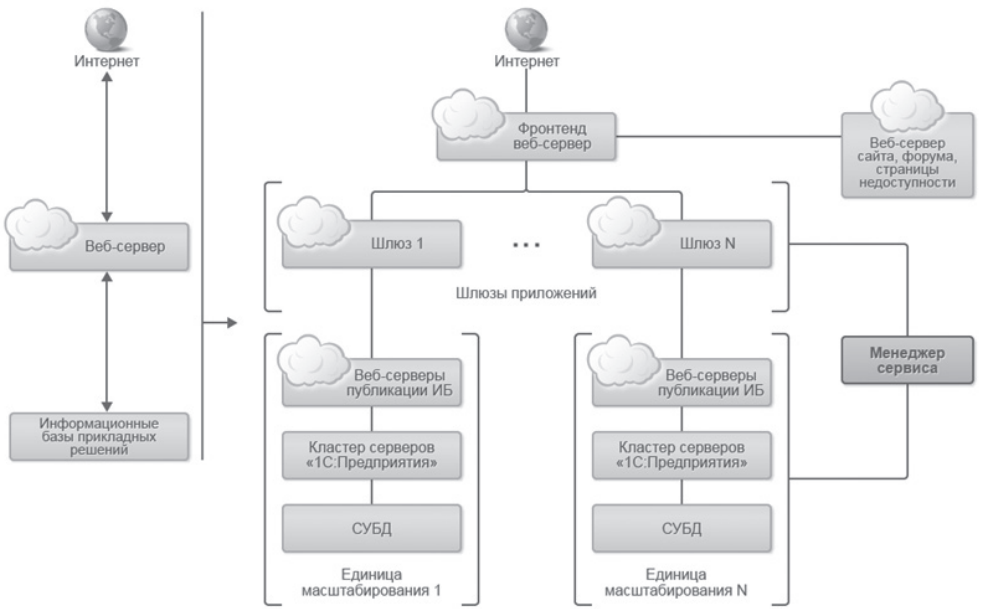


Рисунок 4. Шлюзы приложений и единицы масштабирования

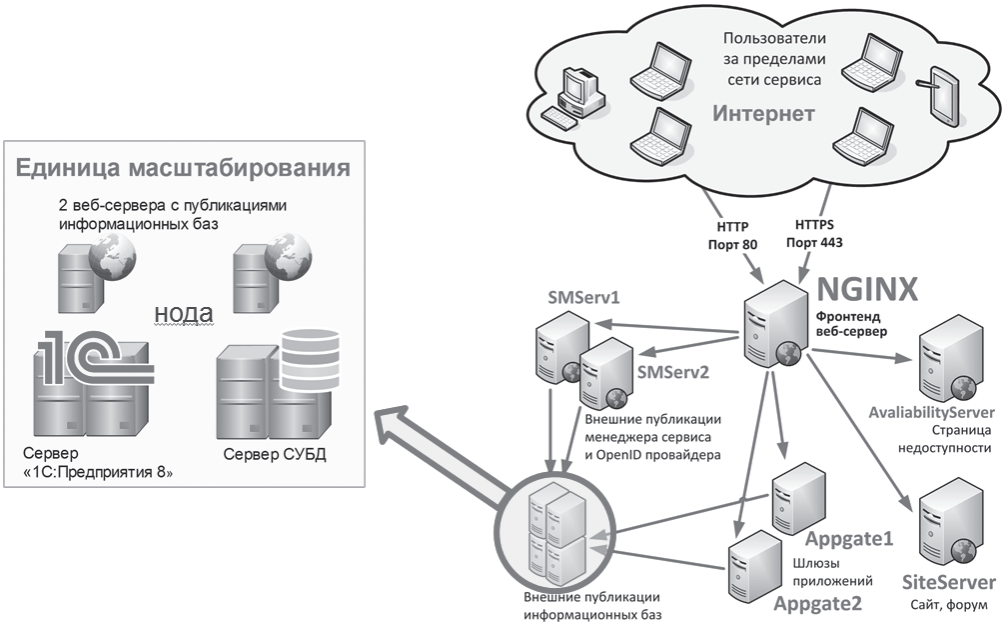


Рисунок 5. Механизм публикаций и единицы масштабирования

От общего к частному

Применительно к вопросам эксплуатации нужно помнить, что система, построенная по технологии 1cFresh, – это в первую очередь крупная информационная система, поэтому:

- подходы при эксплуатации такие же, как для крупных корпоративных внедрений;
- более того, корпоративные внедрения также могут выполняться на основе этой технологии;
- сервис разворачивается для предоставления услуг большому количеству пользователей;
- как следствие, требования к качеству исключительно высокие.

Далее по тексту данного методического пособия технология 1cFresh упоминается в качестве наиболее полного технологического примера для демонстрации подходов и методик к решению задач эксплуатации крупных информационных систем. При этом описанные далее методики в равной степени применимы к программным и аппаратным компонентам любой информационной системы, построенной с использованием технологической платформы «1С:Предприятие».

ГЛАВА 2

Организация эксплуатации крупной информационной системы

Общие вопросы

Эксплуатация крупной информационной системы

Эксплуатация крупной информационной системы – это направление, целью которого является минимизация вероятности возникновения проблем с качеством работы информационной системы на платформе «1С:Предприятие».

Это отдельный вид деятельности, который невозможно не учитывать и который гарантированно будет показывать обратную связь:

- если ресурсов выделяется достаточно, то система будет работать качественно;
- если ресурсов выделяется недостаточно, то будут периодические сбои и «пожарные» работы по восстановлению.

Рассмотрим построение эксплуатации крупной информационной системы. В качестве примера будем рассматривать некоторую систему, в которой одновременно работают от 200 (в ночное время) до 1 000 пользователей (днем) онлайн. При этом число реально зарегистрированных пользователей может оказаться значительно больше.

Основной задачей специалистов будет обеспечение непрерывной технологически качественной работы информационной системы с учетом необходимого развития системы в целях соответствия бизнес-требованиям пользователей.

Технологическое качество

Под технологическим качеством подразумеваются:

- Доступность системы:
  - определяется на внедрении (например, более 99 % процентов времени работы системы);
  - доступность может рассчитываться как для системы в течение всего времени работы, так и только в рабочее время;
  - доступность может рассчитываться для компонентов системы в отдельности.
- Производительность:
  - неухудшение производительности со временем в процессе эксплуатации;
  - производительность, комфортная для работы пользователей.
- Работоспособность системы и отсутствие ошибок:
  - операции у пользователей должны выполняться без технологических ошибок;
  - действия системы должны быть адекватными с точки зрения пользователя;
  - стабильный адекватный результат при выполнении каких-либо операций в системе.
- Неухудшение технологических показателей работы системы (например, загруженности оборудования) в результате обновлений на новые версии.

Здесь не рассматривается функциональное наполнение или соответствие реальным бизнес-процессам пользователей. Однако функциональное наполнение является неотъемлемой частью и одной из основных причин необходимости выполнения периодических обновлений программных продуктов системы.

Задачи

Какие задачи встают перед специалистами, которые хотят обеспечить технологическое качество системы с учетом планов по развитию:

- Планирование задач.
- Администрирование:
  - рабочей зоны;
  - подготовительной зоны;
  - зоны разработки.
- Эксплуатация рабочей зоны.
- Разработка:
  - необходимого инструментария;
  - доработка существующих механизмов;
  - автоматизация повторяющихся задач.

В задачи эксплуатации входят:

- Мониторинг и обслуживание информационной системы:
  - мониторинг всей рабочей зоны, развитие мониторинга;
  - решение проблем, выявленных по данным мониторинга;
  - управление системой, устранение возможных узких мест по данным мониторинга.
- Выполнение плановых задач по обслуживанию информационной системы:
  - введение новых единиц и их поддержка с началом работы пользователей в системе.
- Дежурство и «пожарные» работы:
  - внесение критичных исправлений для восстановления работы.
- Организация первой линии технической поддержки:
  - разбор и классификация обращений;
  - максимально оперативная реакция на обращения пользователей;
  - взаимодействие с пользователями;
  - трансфер собранной и структурированной информации о проблеме второй линии техподдержки.
- Организация второй линии технической поддержки:
  - разбор обращений и решение технологических проблем;
  - классификация проблем.
- Тестирование и обновление на новые версии решений без ухудшения качества.
- Аудит дополнительных отчетов и обработок.
- Автоматизация повторяющихся задач.

В задачи администрирования входят:

- Развертывание, конфигурирование и обслуживание оборудования.
- Конфигурирование среды виртуализации.
- Конфигурирование сети.
- Конфигурирование кластера серверов «1С».
- Конфигурирование СУБД.
- Регулярное создание бэкапов, хранение и развертывание бэкапов.
- Обеспечение работы с лицензиями, ключами, сертификатами.
- Обновление программных продуктов.
- Развертывание новых единиц масштабирования.
- Автоматизация повторяющихся задач.
- Настройка мониторинга при внесении изменений в компоненты системы.

В задачи разработки входят:

- Прием в эксплуатацию новых информационных баз и областей:
  - доработка конфигураций до возможности перехода в сервис.

- Разработка аналитических отчетов.
- Расширение статистики.
- Разработка механизмов для развития сервиса.
- Разработка механизмов интеграции.

Планирование

Для планирования задач могут использоваться следующие инструменты:

- менеджер сервиса;
- сервис-деск;
- «Центр контроля качества»;
- «Документооборот»;
- Jira;
- Confluence;
- Skype;
- Telegram;
- MS Project;
- другие.

Автоматизация

В случае использования множества серверов в рабочей и подготовительной зонах имеет смысл заранее подготовить шаблоны виртуальных машин, из которых будут «разворачиваться» и конфигурироваться типовые единицы масштабирования.

Такие шаблоны в минимальном виде – это, например:

- шаблон рабочего сервера;
- шаблон сервера СУБД;
- шаблон веб-сервера.

Но только шаблонов недостаточно для решения задачи быстрого развертывания единиц масштабирования информационной системы. Существенной частью этой задачи является автоматизация.

Особое внимание можно обратить на подготовку следующих механизмов:

- автоматическая донастройка виртуальной машины до заданных параметров;
- автоматизация развертывания информационной базы;
- настройка и применение групповых политик;
- автоматизация запуска, остановки служб, сбора служебной информации;
- автоматизация установки платформы, СУБД;
- автоматизация развертывания виртуальных машин из шаблонов в заданной конфигурации.

Организация эксплуатации

Регламенты, необходимые для организации процесса эксплуатации:

- Дежурство:
  - расписание, которое все видят и знают;
  - в каждый момент времени есть один ответственный за максимальную реакцию на входящие критичные заявки;
  - реакция на мониторинг.
- Разбор входящих проблем.
- Разбор задач по автоматизации и доработкам.
- Организация доступов.
- Изменения в рабочей зоне:
  - план обновлений;
  - check-листы:
    - по тестированию в тестовой зоне;
    - обновлению и проверке результатов обновления в подготовительной зоне;
    - обновлению в рабочей зоне;
  - журнал всех изменений.
- Организация приемки новых баз и областей.
- Организация приемки дополнительных отчетов и обработок.
- SLA:
  - с ЦОД;
  - с провайдерами;
  - для группы эксплуатации.

Важно учитывать, что построение эксплуатации крупной информационной системы невозможно без выращивания команды специалистов, которые думают в одном направлении. Члены команды должны одинаково понимать, что требуется получить от информационной системы в будущем.

Основная ценность – команда, решающая задачи эксплуатации, администрирования и разработки.

Важно придерживаться следующих принципов:

- Команда смотрит в одном направлении и в целом понимает:
  - что является наиболее критичным для сервиса;
  - что является наиболее срочным для сервиса;
  - что является наиболее важным для сервиса.
- Нет таких сведений о сервисе, которые известны только одному участнику и неизвестны никому более. Такая проблема решается:
  - наличием базы знаний;

- журналами изменений;
- check-листами;
- отработкой задач на подготовительном стенде;
- групповым чатом изменений в системе.

- Конфигурация нод, серверов, программных продуктов – «типовая», т. е. одинаковая, документированная для данной информационной системы:
  - конфигурируются и описываются конкретные единицы масштабирования;
  - все изменения отражаются во всех существующих единицах масштабирования;
  - исключений либо нет, либо их число постоянно сознательно и целенаправленно минимизируется;
  - горизонтально сервис масштабируется только типовыми нодами;
  - вертикальное масштабирование возможно только в рамках одной единицы масштабирования до момента, пока не будет достигнут некоторый заранее определенный лимит ресурсов для единицы масштабирования, затем создается новая единица масштабирования.
- Каждый участник команды эксплуатации должен иметь опыт работы на различных задачах:
  - обновление и тестирование;
  - дежурство;
  - автоматизация;
  - аудит.
- По каждой проблеме обязательно должен быть разбор, в котором решается:
  - что было сделано неправильно;
  - как нужно сделать, чтобы проблема не повторилась в будущем.
- Все участники эксплуатации должны быть в курсе текущих критичных проблем. Это решается:
  - журналом изменений и еженедельной отчетностью;
  - наличием общего чата, в котором отмечаются «пожарные» решения проблем, обновления или изменения в рабочей зоне.

Под *проблемами качества работы системы* подразумеваются проблемы работоспособности, устойчивости и производительности системы, ухудшение технологических показателей системы.

Под *изменениями рабочей системы* подразумеваются любые изменения программного, аппаратного, организационного или инфраструктурного уровней системы, включая следующие (но не ограничиваясь ими):

- Любые изменения кода конфигурации системы (в том числе связанные с доработками функционала или исправлением ошибок).
- Изменения версии «1С:Предприятия», настроек кластера «1С:Предприятия», настроек рабочих серверов, шлюзов, настроек информационной базы.

- Изменения состава, технических характеристик, настроек оборудования, настроек сети, настроек виртуальных машин, настроек операционной системы.
- Изменения типа СУБД, версии СУБД, настроек СУБД, планов обслуживания.
- Расширение состава операций, выполняемых пользователями системы.
- Значительное (например, более 10 %) увеличение числа пользователей системы.
- Добавление новых единиц масштабирования, информационных баз.

Для учета изменений состояния системы вводится понятие «версия системы». Версией системы называется ее состояние в определенный момент времени. Публикацией версии называется ее запуск в рабочую эксплуатацию.

Версия системы имеет уникальный номер и описание: дату публикации и перечень новых значений всех параметров системы, изменившихся по сравнению с предыдущей версией. Версии системы нумеруются по возрастанию – более поздние версии имеют большие номера.

Версии публикуются в соответствии с календарным планом. Внесение изменений в рабочую систему вне процедуры публикации версии не допускается. Иначе говоря, любые изменения могут быть внесены только в одну из следующих запланированных версий.

Версия, по сути, описывает совокупное состояние системы, которое имеет смысл сравнивать с другим состоянием, используемым при следующем обновлении.

Основные этапы процедуры публикации новой версии информационной системы:

- Согласование перечня изменений и срока выпуска.
- Разработка новой версии.
- Тестирование новой версии на тестовых стендах и тестовых данных.
- Контроль качества тестовой системы.
- Выпуск.
- Обновление в подготовительной зоне информационной системы.
- Тестирование новой версии в подготовительной зоне на реальных данных.
- Контроль качества системы в подготовительной зоне, контроль качества обновления, времени и ресурсов обновления.
- Обновление рабочей системы.
- Контроль качества рабочей системы.

В целях обеспечения качества новых версий информационной системы создается **подготовительный стенд**. Подготовительный стенд должен соответствовать рабочей системе по следующим параметрам:

- Код конфигураций информационных баз.
- Версия технологической платформы «1С:Предприятие», конфигурация кластера серверов.
- Тип и версия СУБД, конфигурация СУБД.
- Тип и версия ОС на всех серверах, конфигурация виртуальной и аппаратной среды.

- Актуальные пользовательские данные.
- Настройки приложений.
- Настройки серверов (СУБД, приложений, веб-серверов).
- Скрипты и планы обслуживания.
- Доступы.

Подготовительный стенд должен содержать полную копию рабочих информационных баз на момент начала подготовки к переводу на новую версию.

Сотрудники группы эксплуатации проводят на тестовой версии ряд тестов, включая функциональные и нагрузочные (многопользовательские). Выполняются все тесты, которые выполнялись перед публикацией предыдущей версии. Тесты могут быть доработаны (расширены) по решению группы эксплуатации на основании анализа состава планируемых изменений, а также проблем, возникших в рабочей зоне системы за прошедшее время.

Зоны системы

Разделение на зоны

При организации эксплуатации крупной информационной системы выделяются:

- Рабочая зона информационной системы.
- Подготовительная зона информационной системы.
- Зона тестирования продукта.
- Зона разработки продукта.

Рисунок 6. Зоны системы

При этом изменения в продукционную зону могут попадать только следующими путями:

- серверы разработки продукта;
- серверы тестирования продукта;
- подготовительные серверы системы;
- продукционные серверы системы.

Отличия зоны тестирования продукта от подготовительной зоны информационной системы:

- В зоне тестирования продукта он тестируется перед выпуском на тестовых данных на основных сценариях работы с этим продуктом.
- В подготовительной зоне тестируется информационная система в сборе в том виде, в котором она будет работать в рабочей системе.
- Подготовительная зона информационной системы и рабочая зона информационной системы не являются зонами тестирования одного или нескольких продуктов – эти зоны нужны для обеспечения технологического качества работы конкретной информационной системы.
- Зону тестирования продукт проходит до выпуска, в том время как в подготовительную зону попадают только выпущенные продукты, имеющие уникальный номер версии, а также дату и время выпуска.

Важно отметить, что от того, какие затраты клиент готов нести на обслуживание системы, зависит то, насколько указанная выше схема «ложится» на реальную систему клиента.

Например, могут рассматриваться варианты совмещения зон.

Вариант 1:

- рабочая зона;
- подготовительная и тестовая зона;
- зона разработки продукта.

Здесь объединяются подготовительная и тестовая зоны, при этом качество тестирования непосредственно перед обновлением в продуктиве напрямую зависит от качества соответствия подготовительной зоны рабочей зоне.

При этом зона разработки продукта, например, может вообще отсутствовать, если продукт не изменяется и не разрабатывается специалистами клиента. То есть процесс разработки продукта полностью отделен от рабочей информационной системы.

Вариант 2:

- рабочая зона;
- подготовительная зона;
- зона разработки и тестирования.

Такой вариант может применяться в случае, когда доработка продукта носит незначительный характер и практически не изменяет поведения и характеристик продукта. Тем не менее разработка не происходит в информационной системе в рабочих или подготовительных зонах.

Чего не следует делать

Совершенно точно не следует совмещать рабочую зону с подготовительной зоной, так как, по сути, новая версия системы будет «собираться» именно в продуктиве. Такой подход дает повышенный риск сбоев в результате обновлений.

Не стоит совмещать подготовительную зону и зону разработки. В этом случае появляется риск утечки данных, так как доступ к данным пользователей появляется у разработчиков.

Не следует совсем отказываться от использования тестовой и подготовительной зон, так как в этом случае тестирование, отладка, расследование проблем будут происходить на пользователях.

В этом случае страдает бизнес клиента, а от пользователей постоянно поступает негатив. Также стоит отметить, что после каждого такого обновления, нужного бизнесу, без тестирования в тестовой и подготовительной зонах следует период нестабильной работы системы, который по сути и является периодом опытно-промышленной эксплуатации, когда тестирование происходит на пользователях.

Что имеет смысл сделать

Если рабочая зона имеет типовую организацию и несколько единиц масштабирования, например по технологии 1cFresh, имеет смысл разбить рабочую зону на несколько и обновлять ее по частям.

Сначала стоит обновить самую маленькую часть. Затем, если система после обновления в продуктиве работает без сбоев, имеет смысл приступить к обновлению всей рабочей зоны.

Подготовительную зону имеет смысл развертывать скриптами, которые применяются для целей автоматизации, из бэкапов с рабочей зоны. Таким образом будет гарантироваться, что подготовительная зона максимально приближена к рабочей зоне.

При этом вы будете уверены:

- что у вас всегда есть актуальная копия системы (базы данных, настроек) на нужный вам момент времени и вы совершенно точно в состоянии ее развернуть;
- скрипты автоматизации работают без сбоев;
- бэкапы создаются без сбоев и их действительно можно использовать;
- вы периодически тренируете навык восстановления системы «с нуля»;
- известно достаточно точное время развертывания и создания единицы масштабирования с нуля.

Стоит продумать возможность перевода оборудования из подготовительной зоны в рабочую на случай непредвиденных аварий.

Имеет смысл также организовать и иметь полный удаленный доступ в рабочую и подготовительную зоны. При этом такой доступ:

- должен быть только у ограниченного, известного круга лиц;
- максимально защищен.

Удаленного доступа может не быть к зоне разработки и тестирования продукта. Продукт может представлять особенную ценность для организации, в том числе с точки зрения обеспечения безопасности, а открытие доступа не только создает риск утечки исходного кода, но и открывает дополнительные возможности злоумышленникам в тех системах, где продукт используется. К тому же, как правило, нет задачи поддерживать, например, доступность серверов разработки на уровне 99 % и выше удаленно: разработчики обычно сами в состоянии это сделать со своих рабочих мест. В некоторых случаях (например, географической отдаленности) существует задача обеспечения доступности системы сборки и тестирования – тогда такой доступ может быть организован.

Организация подготовительного стенда информационной системы

Положение подготовительного стенда информационной системы в эксплуатации рабочей системы:

- Рабочие серверы информационной системы:
  - запрещена отладка;
  - пользовательские данные;
  - обновление только с использованием check-листов по заранее согласованному плану;
  - настроен мониторинг;
  - доступ только у группы эксплуатации.
- Подготовительные серверы информационной системы:
  - разрешена отладка;
  - пользовательские данные;
  - все изменения фиксируются;
  - контроль обновления и работы на новых версиях продуктов, контроль потребления ресурсов на новых версиях;
  - настроен мониторинг;
  - доступ только у группы эксплуатации.
- Серверы тестирования продукта:
  - разрешена отладка;
  - тестовые данные;

- изменения вносятся постоянно;
- настроен мониторинг;
- доступ у разработчиков.

- Серверы разработки продукта:
  - разрешена отладка;
  - тестовые данные;
  - изменения вносятся постоянно;
  - мониторинг может отсутствовать;
  - доступ у разработчиков.

Можно выделить следующие этапы организации подготовительного стенда:

- Анализ схемы доступа к внутренним серверам продукционной площадки.
- Выбор схем резервирования инженерных систем.
- Выбор способа синхронизации с рабочей площадкой.
- Анализ требований к оборудованию подготовительной площадки.
- Организация адресации.
- Получение копий виртуальных машин либо конфигурирование машин из шаблонов с кастомизацией, конфигурационных файлов, баз данных с пользовательскими данными.
- Запуск синхронизации с рабочей площадкой (например, готовность автоматического развертывания актуальных бэкапов по команде).
- Организация внесения изменений на подготовительной и рабочей площадках.

Организация автоматического развертывания бэкапов с рабочей системы также позволяет быть уверенными, что бэкапы действительно есть, они собираются, а также могут быть развернуты за определенное время. Кроме того, важно, что специалисты группы эксплуатации периодически (по необходимости) восстанавливают из бэкапов подготовительный стенд и таким образом тренируются, что окажется существенным с точки зрения экономии времени в момент возможных «пожарных» работ в рабочей системе.

Синхронизация с рабочей площадкой должна проходить в автоматическом режиме и не должна отличаться от механики восстановления данных на продукционной системе в случае аварии.

Анализ схемы доступа к внутренним серверам продукционной площадки

Первым обязательным этапом организации подготовительного стенда является анализ схемы доступа к внутренним серверам продукционной площадки. Результатом анализа является документ, включающий в себя:

- описание сетей рабочей площадки, адресацию;
- имена серверов рабочей площадки, их адреса;

- роли серверов;
- программные продукты, используемые для доступа к серверам;
- список лиц, имеющих доступ к серверам.

Важным шагом этого этапа является составление описания сетей рабочей площадки, так как в подготовительной площадке должна полностью сохраниться адресация. Т. е. сами виртуальные машины не должны знать, где именно они находятся: в рабочей зоне или в подготовительной. Все виртуальные машины должны быть настроены по аналогии с рабочей зоной (может, конечно, меняться фон рабочего стола – например, для рабочей зоны фон рабочего стола красный с именем ноды и информацией из bginfo, а для подготовительной зоны – зеленый с именем ноды и информацией bginfo).

Например, в информационной системе можно обеспечить сегментацию на виртуальные сети:

- управления;
- 1С;
- SQL;

  и две зоны:

  - рабочая зона;
  - подготовительная зона.

Так как настройки адресации на подготовительном стенде остаются полностью такими же, как и в рабочей зоне, отличием от продукционного стенда может служить отсутствие доступа в Интернет на подготовительном стенде.

Также доступ в Интернет с подготовительного стенда следует закрывать, чтобы не было случайных рассылок реальным пользователям о возможных недоступностях, регламентных работах или о нововведениях в подготовительной зоне сервиса.

Организация рабочего стенда информационной системы

При организации рабочей зоны очень важно не получить систему с сотнями тайных связей. Очень важно, чтобы конфигурация рабочей системы была максимально типовой, строилась из очень простых единиц масштабирования с минимизацией связей с другими единицами. В процессе эксплуатации может возникнуть желание доработать и развить систему. Очень важно в процессе таких доработок сознательно стремиться к упрощению существующей конфигурации.

Здесь можно выделить следующие ключевые особенности:

- быстрое вхождение новых специалистов;
- больше одного человека понимают, как действительно работает тот или иной механизм в системе;
- шаблоны конфигурации типовые, поэтому отклонения в поведении узлов могут быть выявлены гораздо быстрее;
- упрощение автоматизации;

- высокая скорость встраивания новых механизмов;
- минимизация требований по ресурсам к подготовительному стенду (ввиду типизации конфигурации);
- уменьшенный объем тестирования с возможностью более качественного тестирования меньшего множества конфигураций;
- возможность быстрого расширения.

Например, организация рабочей зоны по технологии 1cFresh предполагает нодовую структуру развертывания информационной системы.

Регламенты

Подробно о регламентах, применяемых при эксплуатации крупных информационных систем с участием специалистов фирмы «1С», поговорим в главе 12. В данном разделе озвучим только базовые принципы, без соблюдения которых нельзя говорить о выстроенном процессе эксплуатации.

Организация внесения изменений на подготовительной и рабочей площадках

Вводится правило: *все изменения в рабочую систему вносятся через подготовительную площадку* информационной системы.

Сначала изменения попадают на подготовительный стенд, затем на рабочий стенд.

Существует только одно исключение из этого правила: *изменения в рабочую систему могут вноситься, минуя подготовительный стенд, только в экстренных случаях*.

По окончании аварийных работ в ночное время стенд должен быть синхронизирован с рабочей зоной, чтобы устранить разницу во внесенных изменениях.

Состояние подготовительного стенда может изменяться следующими способами:

- развертывание бэкапов;
- развертывание копий машин;
- с подготовкой check-листов;
- небольшие изменения без подготовки check-листов, но с обязательной записью в журнал регистрации проведенных работ.

Состояние рабочего стенда может изменяться следующими способами:

- по check-листам;
- в случае аварийных работ при недоступности рабочей системы;
- перенос небольших изменений с подготовительного стенда с записью в журнал регистрации проведенных работ.

Check-лист готовится заранее при выполнении изменений на подготовительном стенде.

Перед применением изменений на рабочей площадке check-листы распечатываются и передаются каждому участнику плановых работ.

Синхронизация участников выполняется оперативно по телефону или чату.

Журнал регистрации проведенных работ – специальный документ, который ведут все участники эксплуатации рабочей и подготовительной систем. В «бортовой журнал» попадают записи, отражающие все изменения, в очень сжатой форме. Цель «бортового журнала» – в случае аварии быстро определить, кто и когда вносил изменения.

Пример записи в журнале регистрации проведенных работ (**«бортовом журнале»**):

```
2020-01-01 Администратор1 <admin_petr@1c.ru> +79871234567
        * server1c-1
                – поправил com connector, выполнив команду из rdp сеанса C:\1C\current\regsvr32 comcntr.dll
----------------------------------------------------------------------------------------------------------------------------------------------------------
```

Контроль качества работы подготовительного стенда

Во время тестирования разработчики контролируют показатели качества работы системы и сравнивают их с аналогичными показателями, полученными при тестировании предыдущей версии. Если показатели новой версии ухудшились по сравнению с предыдущей, то принимается согласованное решение о дальнейших действиях. Ответственные за эксплуатацию информационной системы договариваются либо отложить выпуск до исправления проблем, либо опубликовать версию, несмотря на наличие проблем.

После исправления проблемы необходимо провести повторное контрольное тестирование.

Публикация в рабочей зоне

Версия, успешно прошедшая тестирование в подготовительной зоне, публикуется в рабочей системе в соответствии с планом публикации версий и check-листами, которые были подготовлены на подготовительном стенде.

Время выполнения обновления, а также особенности обновления фиксируются и сравниваются с результатами, полученными в подготовительной среде.

Отличия в поведении систем в подготовительной и рабочей зонах фиксируются и расследуются с целью устранения.

Версия, не прошедшая процедуру тестирования, не может быть опубликована в рабочей зоне.

В случае внесения исправлений в версию номер версии поднимается.

Контроль качества рабочей системы

Группа эксплуатации ведет постоянный контроль качества рабочей системы. Если какие-либо показатели качества работы текущей версии оказываются хуже, чем у предыдущей версии, то принимается согласованное решение об устранении проблемы. Стороны договариваются либо об отказе от использования текущей версии и откате на предыдущую, либо о продолжении работы, несмотря на существующие проблемы.

Допустимым временем простоя системы при откате на предыдущую версию считается *10 минут*.

При выявлении проблем в рабочей системе может быть принято решение о доработке тестов таким образом, чтобы в следующий раз подобная ошибка была выявлена на стадии тестирования.

Для качественного контроля за рабочей системой необходимо настраивать мониторинг.

Методику настройки мониторинга можно найти в главе 3 данного пособия «Мониторинг на продукционных серверах».

Также инструкция по настройке мониторинга с помощью **«Центра контроля качества»** имеется на ИТС («Разработка и администрирование» – «Корпоративный инструментальный пакет» – глава 5 «Центр контроля качества»).

ГЛАВА 3

Мониторинг на продукционных серверах

Настройка мониторинга в информационной системе необходима:

- для понимания технологического качества функционирования информационной системы;
- быстрой реакции на возникшие проблемы и предупреждения возможных проблем;
- определения возможностей информационной системы по масштабированию;
- расследования проблем, возникающих как между компонентами системы, так и в самих компонентах, которые работают в новых условиях.

При настройке мониторинга информационных систем могут использоваться продукты:

- ЦКК из «Корпоративного инструментального пакета»;
- Zabbix;
- другие.

Настройка Zabbix в современных информационных системах используется для мониторинга следующего оборудования:

- сетевые устройства;
- электрооборудование;
- физические серверы;
- дисковые полки.

Обычно такие настройки очень специфичны и требуют отдельных специальных инструкций для каждого экземпляра оборудования определенного производителя. Такие методики настройки не будут рассматриваться в этом пособии.

В целом в реальной системе должна собираться и контролироваться информация в следующих разрезах:

- Физическое оборудование:
  - сетевые устройства;
  - электрооборудование;
  - физические серверы;
  - дисковые полки.
- Гипервизор.
- Операционные системы:
  - Linux;
  - Windows.
- Приложения (с разделением по типам виртуальных машин):
  - Linux 1C:
    - ragent;
    - rmngr;
    - rphost;
    - ras;
    - rac;
    - 1cv8c.
  - Linux web:
    - Apache.
  - Linux db:
    - PostgreSQL.
  - Windows 1C:
    - ragent;
    - rmngr;
    - rphost;
    - ras;
    - rac;
    - 1cv8c.
  - Windows web:
    - Apache;
    - IIS.
  - Windows db:
    - PostgreSQL;
    - MS SQL Server.
  - Windows management.
  - Linux management.

- Archive.
- Nginx.
- Tomcat.
- Gate.

- Прикладные программы:
  - менеджер сервиса;
  - агент сервиса;
  - БП, УНФ, ЗУП, КА и т. д.
- Кластеры серверов «1С»:
  - ИБ;
  - рабочие серверы.

Настройка ЦКК для Windows-серверов

Общие сведения

«Центр контроля качества» (ЦКК) – типовая конфигурация, входящая в **Корпоративный инструментальный пакет**. ЦКК умеет собирать, агрегировать информацию и вовремя оповещать обо всех важных событиях в работе вашей системы.

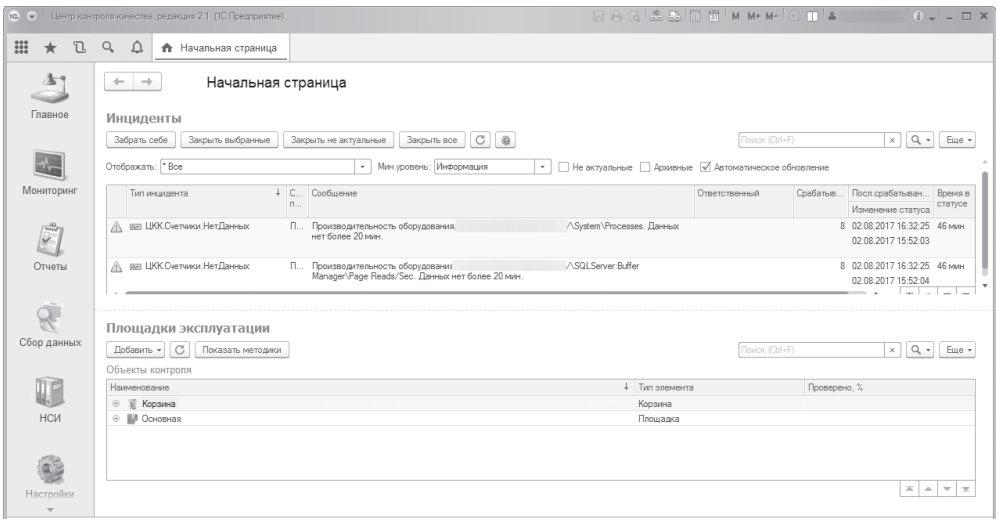


Рисунок 7. ЦКК. Начальная страница

В подсистеме «Мониторинг» вы сможете:

- просмотреть значения различных показателей по контрольным процедурам, которые уже были настроены;

- поделиться настроенной статистикой с коллегой (быстро показать, на что вы смотрите).

В подсистеме «Оповещения» вы сможете:

- создать новые оповещения (условия, при которых ЦКК должен будет отправить вам sms или e-mail);
- просмотреть журнал оповещений.

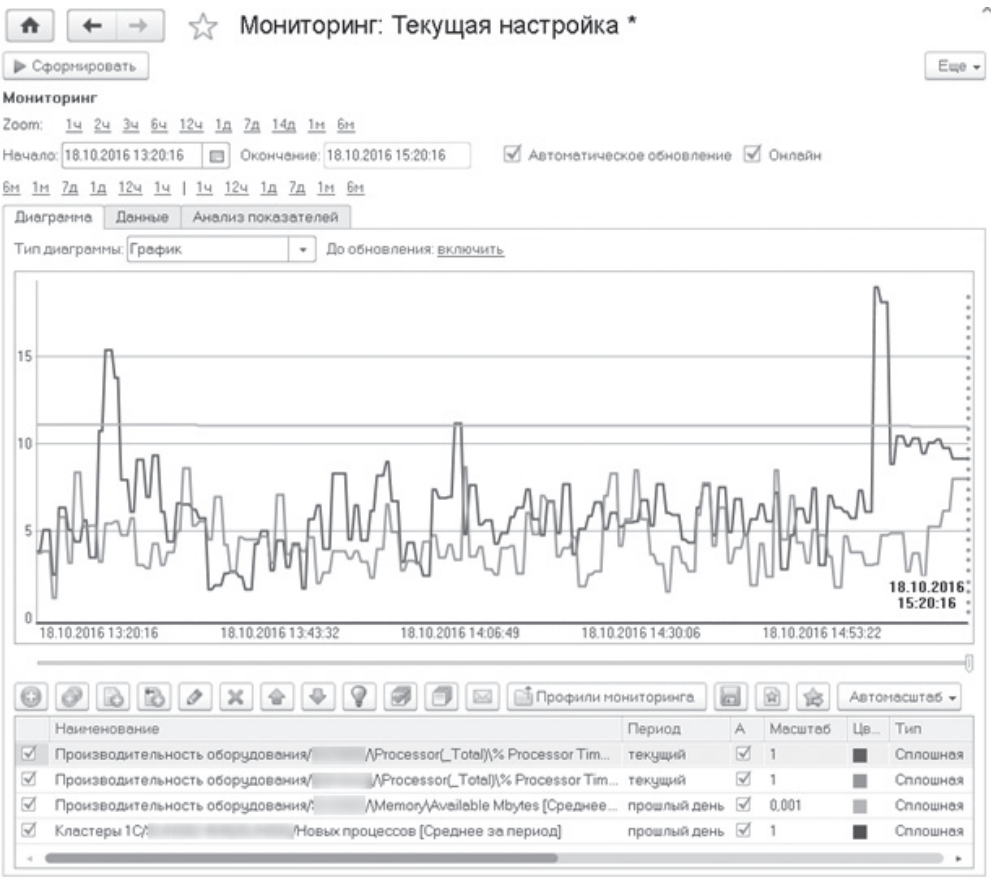


Рисунок 8. ЦКК. Мониторинг

Основным интерфейсом работы в ЦКК с версии 2.1.0 является интерфейс Площадки эксплуатации.

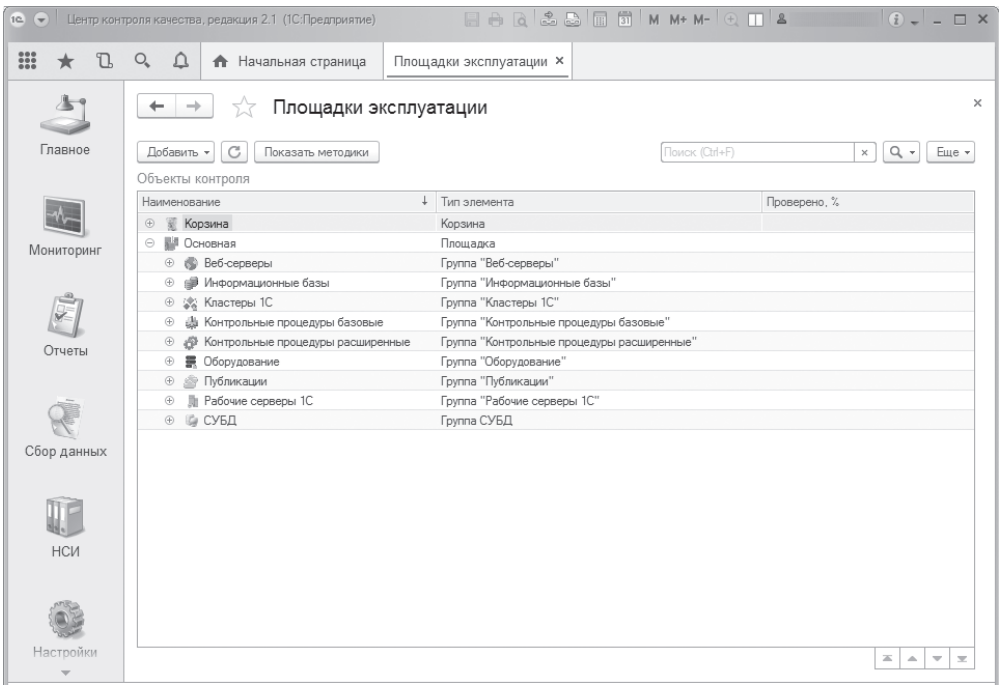


Рисунок 9. ЦКК. Площадки эксплуатации

В рамках данного интерфейса происходит добавление новых объектов контроля: единиц оборудования, информационных баз, публикаций. По мере добавления каждой единицы контроля появляется возможность настройки присущих ей контрольных процедур и дополнительных параметров мониторинга.

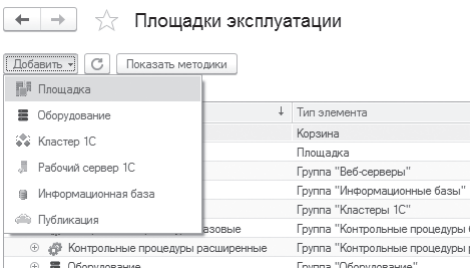


Рисунок 10. ЦКК. Добавление объектов контроля

Для минимальной настройки ЦКК рекомендуется внести в базу структуру площадки (оборудование, кластеры «1С», информационные базы, публикации), а также настроить сбор счетчиков производительности и основные контрольные процедуры:

- контроль подключений;

- контроль производительности;
- контроль потребления памяти;
- контроль устойчивости системы.

Настройка каждой из указанных процедур занимает не более 5–10 минут.

Настройка всех контрольных процедур в ЦКК желательна, т. к. в этом случае вы сможете в полной мере использовать существующую функциональность ЦКК.

Ниже приводится описание минимального объема механизмов ЦКК, которые могут использоваться при эксплуатации внедрения.

Зоны ответственности

В ЦКК для каждой контрольной процедуры задается **зона ответственности**, в которую входят пользователи, ответственные за эту контрольную процедуру.

Для вывода справочника зон ответственности следует выбрать команду Зоны ответственности в разделе меню Сервис.

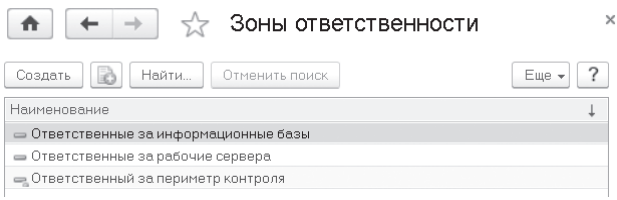


Рисунок 11. Зоны ответственности

Для добавления новой зоны ответственности, удаления зоны ответственности, изменения свойств зоны ответственности используются стандартные средства.

При добавлении зоны ответственности или изменении свойств зоны ответственности выводится окно свойств зоны ответственности.

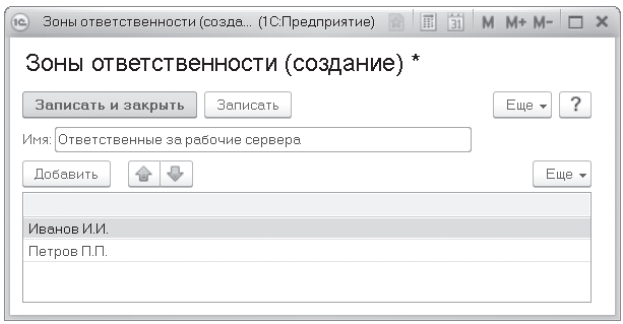


Рисунок 12. Создание зоны ответственности

Контроль подключений к информационной базе

Рекомендуем начать с настройки контроля подключений.

Контрольная процедура позволяет оперативно обнаруживать проблемы с подключением к информационной базе, а также собирает статистику доступности информационной базы.

При работе контрольной процедуры ЦКК каждые 10 секунд пытается установить подключение к информационной базе «1С:Предприятия 8» через опубликованный веб-сервис или при помощи COM-коннектора.

Способы подключения к информационной базе

При проверке соединения с информационной базой с помощью веб-сервиса требуется, чтобы в контролируемой информационной базе был опубликован веб-сервис, удовлетворяющий следующим требованиям:

- имя сервиса: RemoteControl;
- URI пространства имен: http://www.1c.ru/SSL/RemoteControl_1_0_0_1;
- сервис имеет единственную операцию с именем GetCurrentState;
- обработчик операции не имеет входящих параметров и возвращает значение типа boolean (http://www.w3.org/2001/XMLSchema).

Этот веб-сервис присутствует в конфигурации ЦКК.

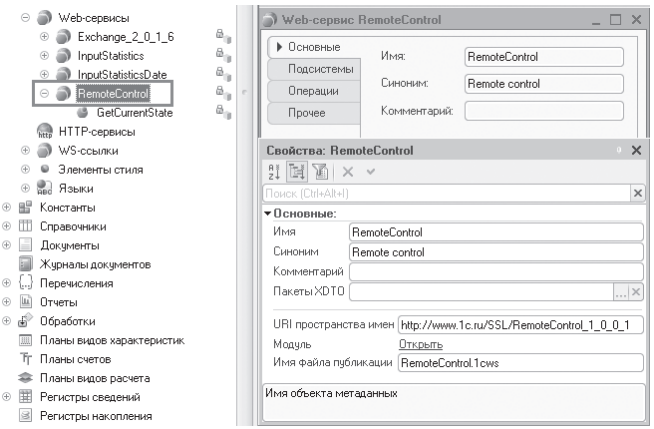


Рисунок 13. Веб-сервис RemoteControl

При проверке соединения с информационной базой с помощью COM-коннектора в настройках контрольной процедуры необходимо указать строку соединения с информационной базой. Эту строку можно увидеть в окне Запуск 1С:Предприятия под списком информационных баз.

Для настройки контрольной процедуры следует:

1. Открыть интерфейс Площадки эксплуатации на начальной странице ЦКК. Для настройки контрольной процедуры Контроль подключений в площадку должна быть добавлена информационная база, подключение к которой будет контролироваться.
2. В иерархии соответствующей площадки раскрыть (при необходимости) ветку Контрольные процедуры базовые – Контроль подключений и в ней ветку, соответствующую нужной информационной базе.
3. Щелкнуть двойным щелчком мыши по названию контрольной процедуры Контроль подключений для <имя базы>.
4. Будет выведено окно настройки процедуры Контроль подключений.

Рекомендуется использовать **подключение через Web-сервис** (см. рисунок 14), так как использование COM-коннектора предполагает наличие одинаковой версии платформы в ЦКК и в контролируемой ИБ.

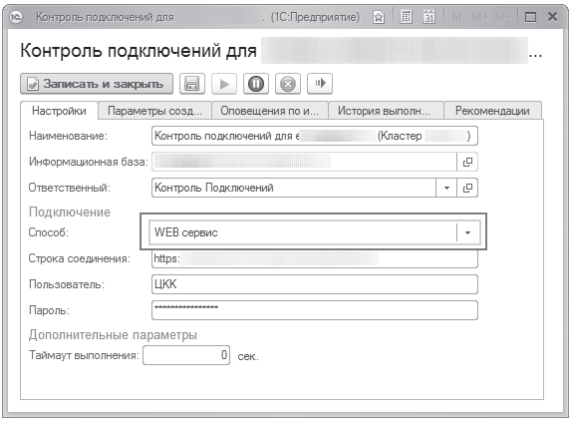


Рисунок 14. Окно настройки контрольной процедуры «Контроль подключений»

Контроль доступности

Для оценки состояния информационной системы ЦКК может проверять доступность не только информационных баз «1С:Предприятия» (контрольная процедура Контроль подключений), но также следующих ресурсов:

Вид ресурса	Способ проверки доступности
Веб-публикация	Опрос веб-публикации с помощью HTTP-запроса
Произвольный сетевой ресурс	Опрос командой ping
Сервер	Прием HTTP-запросов от агента ЦКК – специального приложения, устанавливаемого на контролируемый сервер

Проверка доступности веб-публикаций

Для проверки доступности веб-публикаций необходимо добавить публикацию в список доступности ресурсов ЦКК. Это можно сделать двумя способами:

1. Использовать интерфейс Площадки эксплуатации:
   - Выбрать меню ЦКК Главное – Площадки эксплуатации.
   - Добавить новую публикацию в соответствующую площадку эксплуатации: нажать кнопку Добавить и выбрать в выпадающем меню команду Публикацию.
2. Использовать интерфейс Доступность ресурсов:
   - Выбрать команду меню ЦКК Мониторинг – Доступность ресурсов.
   - Нажать кнопку Добавить и выбрать в выпадающем меню команду Публикацию.

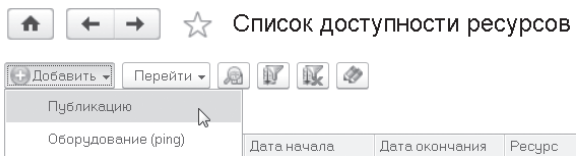


Рисунок 15. Добавление контроля доступности публикации

   - Будет выведено окно Настройка публикаций (создание):

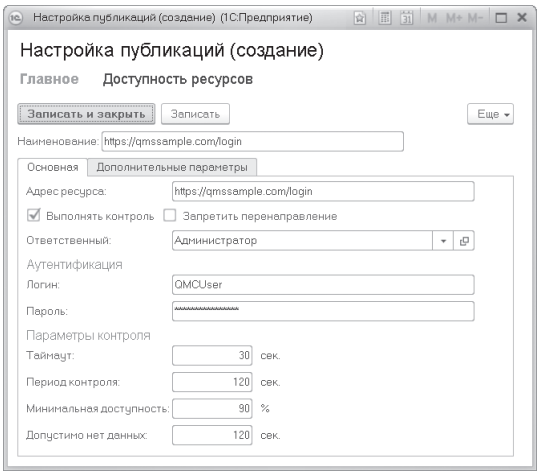


Рисунок 16. Окно настройки публикации

3. Задать в окне Настройка публикаций (создание) следующие параметры:
   - в поле Наименование – условное наименование публикации (оно может задаваться произвольно);

- на вкладке Основная – параметры доступа к публикации;
- на вкладке Дополнительные параметры можно задать стандартный ответ, с которым ЦКК будет сравнивать ответ, полученный от публикации.

4. Нажать в окне Настройка публикаций (создание) кнопку Записать и закрыть.

Проверка доступности ресурсов с помощью ping-запроса

Для настройки проверки доступности устройства с помощью команды ping необходимо добавить контролируемое устройство (хост) в список оборудования, контролируемого ЦКК:

1. Выбрать команду меню ЦКК Мониторинг.
2. Выбрать команду Доступность ресурсов.
3. Нажать кнопку Добавить и выбрать в выведенном контекстном меню команду Оборудование (ping).

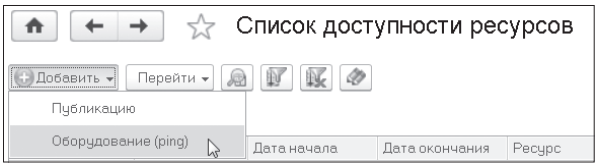


Рисунок 17. Добавление контроля доступности оборудования

4. В выведенном окне Настройка анализа данных по оборудованию (создание) задать следующие параметры:

Параметр	Описание
Наименование	Условное наименование контролируемого устройства (хоста). Может задаваться произвольно
Имя компьютера	Доменное имя или IP-адрес контролируемого устройства (хоста)
Роль	Если для контролируемого устройства не требуется собирать счетчики производительности, следует выбрать роль Внешний ресурс

5. На вкладке Дополнительно следует:
   - установить флажок Проверять доступность;
   - задать параметры проверки доступности:
     - Период контроля – интервал времени, в течение которого проверяется доступность с помощью ping-запроса;
     - Минимальная доступность – минимально допустимый процент времени доступности ресурса;
     - Допустимо нет данных – максимально допустимый интервал недоступности ресурса в секундах;

- флажки Использовать и Регистрировать дампы под надписью Внешний агент следует снять, если на контролируемое устройство не устанавливается внешний агент ЦКК.

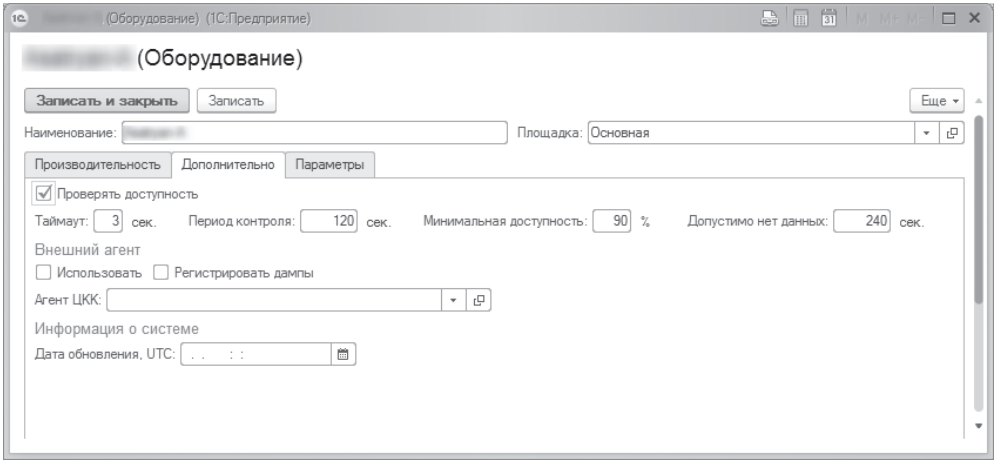


Рисунок 18. Настройка анализа данных по оборудованию

6. Нажать кнопку Записать и закрыть.

После этого регламентное задание Контроль доступности оборудования будет периодически отправлять команду ping к контролируемому устройству.

Проверка доступности серверов с помощью внешнего агента ЦКК

ЦКК может проверять доступность серверов также с помощью специальной программы – агента ЦКК. **Агент ЦКК** – это Java-приложение, которое устанавливается на контролируемый сервер и взаимодействует с ЦКК.

Проверка доступности с помощью агента ЦКК имеет следующие преимущества:

- она надежнее, чем проверка доступности с помощью ping-запросов, поскольку постоянные ping-запросы могут быть заблокированы сетевой инфраструктурой;
- агент ЦКК может не только выдавать ЦКК запросы для проверки доступности, но и регистрировать в ЦКК дампы сервера, на котором установлен агент ЦКК;
- агент может проверять доступность информационных баз;
- агент позволяет контролировать параметры оборудования, на котором он установлен.

Агент ЦКК не требует лицензии «1С:Предприятия» и не занимает лицензию «1С:Предприятия». Подробно работа с агентом ЦКК описана ниже в разделе «Настройка и использование агента ЦКК».

Контроль производительности

Контрольная процедура Контроль производительности является крайне важной для поддержания необходимой производительности и масштабируемости информационной системы. Отказ от выполнения данной контрольной процедуры может привести к серьезным проблемам с производительностью информационной базы.

Для работы контрольной процедуры в контролируемую информационную базу «1С:Предприятия 8» должна быть внедрена подсистема оценки производительности «1С:Библиотеки стандартных подсистем» (БСП), эта система должна быть включена и должен быть настроен экспорт результатов замеров производительности в каталог в виде XML-файлов.

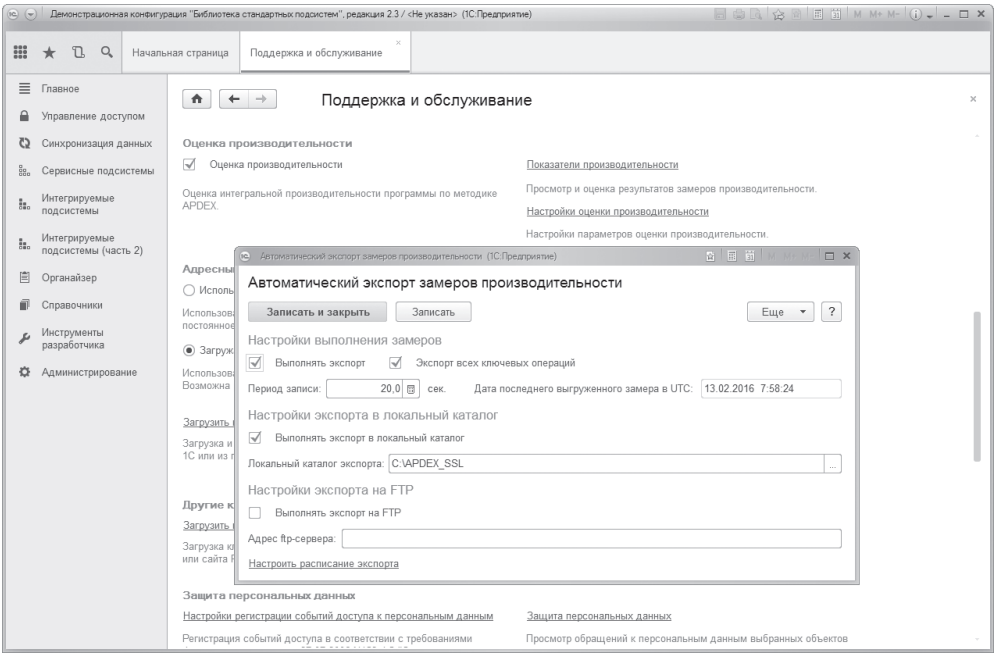


Рисунок 19. БСП. Настройка экспорта замеров производительности

При работе контрольной процедуры ЦКК читает и анализирует данные, сформированные подсистемой оценки производительности. По результатам выполнения контрольной процедуры ЦКК может формировать отчеты о производительности.

Для настройки контрольной процедуры следует:

1. Выбрать команду Главное – Площадки эксплуатации.
2. В дереве объектов контроля соответствующей площадки раскрыть (при необходимости) ветку Контрольные процедуры базовые – Контроль производительности и в ней ветку, соответствующую нужной информационной базе.

3. Щелкнуть двойным щелчком мыши по названию контрольной процедуры Контроль производительности с названием информационной базы, к которой относится контрольная процедура.
4. Будет выведено окно настройки процедуры Контроль производительности:

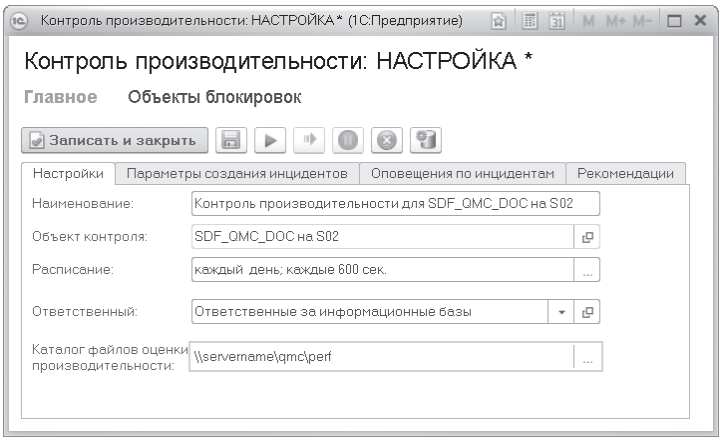


Рисунок 20. Окно настройки контрольной процедуры «Контроль производительности»

В информационной системе выполняется большое количество разнообразных операций. Необходимо отобрать из них только ключевые операции. Именно по ним будет оцениваться производительность системы в целом.

Рекомендуется считать операцию ключевой при выполнении одного из следующих условий:

- Операция является критичной для бизнес-процессов заказчика. При недостаточной производительности системы на этой операции могут происходить потери для бизнеса заказчика.
- Операция выполняется одновременно значительным количеством пользователей (более 10).
- Имеются жалобы пользователей на производительность на этой операции.

Список операций должен составляться при непосредственном участии специалистов заказчика.

Для каждой операции необходимо определить ее приоритет – уникальное целое число. Чем выше приоритет операции, тем важнее ее производительность для бизнеса заказчика. Правильно расставленные приоритеты позволят в дальнейшем оценить серьезность проблем с производительностью в системе и правильно определить приоритеты работ по оптимизации.

Приоритеты операций должны определяться при непосредственном участии специалистов заказчика.

Для каждой операции необходимо определить целевое время – T.

Целевое время – это требование заказчика к скорости выполнения данной ключевой операции. Обратите внимание на то, что это значение не вычисляется и не подбирается опытным путем. Оно назначается исходя из требований бизнес-процессов заказчика, соображений удобства пользователей и т. п. Критерий, применяющийся при назначении времени T: если данная операция будет всегда выполняться за время, меньшее или равное T, то это целиком и полностью устроит заказчика.

Целевое время T для каждой ключевой операции должно определяться при непосредственном участии специалистов заказчика.

Иногда заказчик может затрудняться с определением времени T для некоторых ключевых операций. В таких случаях можно попробовать решить задачу «от обратного»: зная Apdex, получить значение целевого времени T.

Методика вычисления коэффициента производительности описана ниже.

Коэффициент производительности вычисляется отдельно по каждой ключевой операции.

Значение коэффициента производительности по данной операции на данный момент времени вычисляется по 100 последним замерам времени выполнения этой операции, предшествующим моменту оценки.

Коэффициент производительности вычисляется по формуле $(NS + NF/2)/N$, где:

- N – общее количество выполненных операций;
- NS – количество операций, выполненных за время, меньшее или равное T;
- NF – количество операций, выполненных за время, меньшее или равное 4T.

Значения коэффициента производительности оцениваются по следующей шкале:

Диапазон		Оценка
от	до	
0,94	1,00	Отлично
0,85	0,93	Хорошо
0,70	0,84	удовлетворительно
0,50	0,69	Плохо
0,00	0,49	неприемлемо

Как правило, в реальных проектах целью работ является достижение коэффициента производительности не ниже 0,85 по всем ключевым операциям.

Загружаемые в ЦКК данные с помощью контрольной процедуры Контроль производительности загружаются по ключевым операциям. При этом в таких отчетах, как «Оценка производительности», данные по производительности каждой операции будут представлены не только в виде количества, среднего времени, но и в терминах Apdex.

Контроль потребления памяти

Контроль потребления памяти – это контрольная процедура ЦКК, благодаря которой можно оперативно обнаруживать случаи превышения заданного порогового значения оперативной памяти, занятой рабочими процессами сервера. Дополнительным плюсом работы контрольной процедуры является сохранение в ЦКК информации о работе сеансов в контролируемом кластере серверов. По характеру изменения числа сеансов можно очень точно судить о нарушениях в работе системы (например, резкое падение числа сеансов в тот момент, когда не запланировано никаких регламентных работ). Таким образом, число сеансов является важным индикатором работы системы.

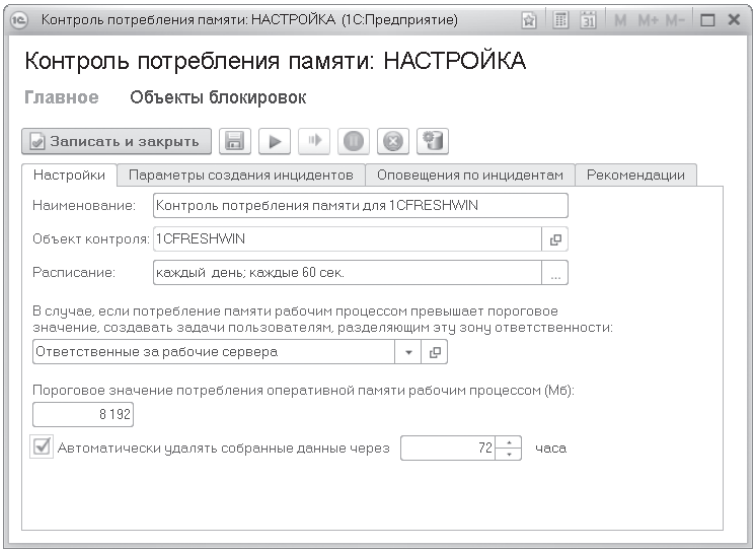


Рисунок 21. Окно настройки контрольной процедуры «Контроль потребления памяти»

Для настройки контрольной процедуры необходимо:

1. Выбрать команду Главное – Площадки эксплуатации.
2. В списке объектов контроля раскрыть (при необходимости) ветку Кластеры «1С» и в ней ветку, соответствующую нужному кластеру.
3. Щелкнуть двойным щелчком мыши по названию контрольной процедуры Контроль потребления памяти с названием кластера серверов «1С:Предприятия», к которому относится контрольная процедура.
4. Будет выведено окно настройки процедуры Контроль потребления памяти.

Контроль устойчивости системы

Контрольная процедура Контроль устойчивости системы предназначена:

- для настройки сбора дампов процессов в случае их аварийных завершений;
- автоматического разбора, сохранения статистики, архивации и перемещения дампов процессов с продукционных серверов;
- настройки сбора минимального необходимого технологического журнала для последующего анализа аварийных завершений процессов кластера;
- оповещения группы ответственных об аварийных завершениях процессов с образованием дампа памяти и без него.

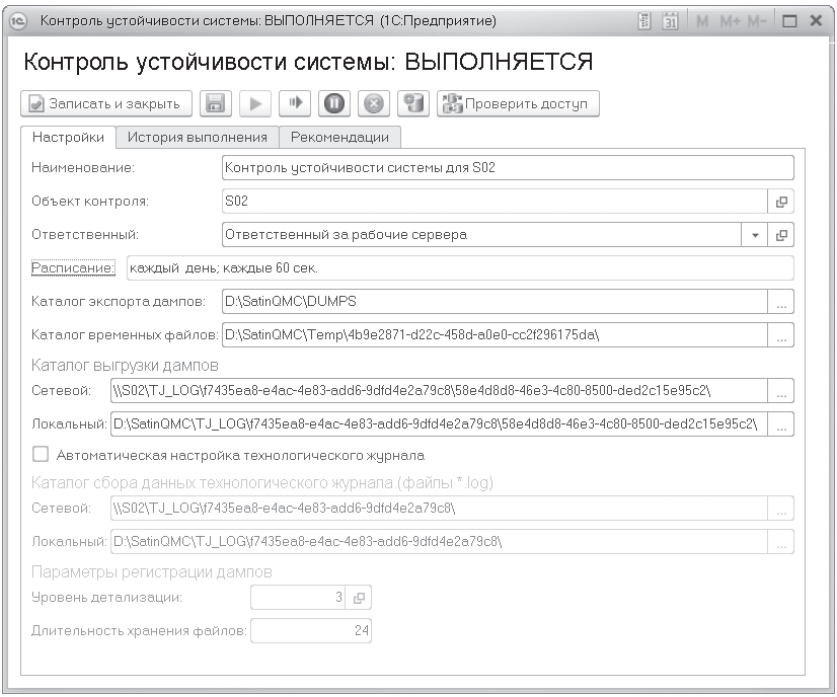


Рисунок 22. Окно настройки контрольной процедуры «Контроль устойчивости системы»

Для настройки контрольной процедуры необходимо:

1. Выбрать команду Главное – Площадки эксплуатации.
2. В списке объектов контроля раскрыть (при необходимости) ветку Контрольные процедуры базовые и в ней ветку Контроль устойчивости.
3. Щелкнуть двойным щелчком мыши по названию контрольной процедуры Контроль устойчивости системы с названием рабочего сервера, к которому относится контрольная процедура.
4. Будет выведено окно настройки процедуры Контроль устойчивости системы.

По умолчанию экспортируются все уникальные (имеющие одинаковые имя, версию и смещение) дампы, однако в настройках каждого дампа можно включить свойство Сохранять все файлы дампов, в этом случае экспортироваться будут все файлы дампов.

Использование ЦКК для агрегации данных

Сбор счетчиков

Типовая проблема эксплуатации крупных информационных систем – сбор счетчиков производительности. Основные сложности:

1. Необходимость собирать однотипные наборы счетчиков производительности с различного оборудования.
2. При вводе в эксплуатацию «не забыть» включить необходимый набор счетчиков производительности.
3. Настройка не должна быть долгой.
4. Наличие счетчиков, не предусмотренных разработчиками решения.
5. Необходимость хранить много данных длительное время.

В ЦКК проблема однотипных показателей с множества различного оборудования решена посредством ввода такой сущности, как «Роль оборудования». Состав счетчиков заполняется однократно для каждой роли, и далее эта роль просто указывается в соответствующей табличной части при настройке сбора счетчиков для единицы оборудования.

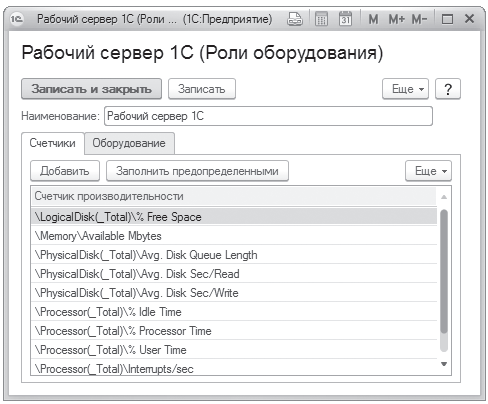


Рисунок 23. Настройка роли для сбора счетчиков

В случае изменения состава счетчиков для роли можно на закладке «Оборудование» выполнить синхронизацию счетчиков для всех или части единиц оборудования, где выбрана эта роль.

При этом есть возможность управлять составом счетчиков из формы элемента единицы оборудования (см. рисунок 24):

- Заполнять счетчики из роли нажатием на кнопку «Синхронизировать счетчики».
- Фиксировать счетчики, специфичные для данной единицы, чтобы не перезаполнять их при синхронизации с ролью (двойной клик в колонке «Фикс.»).
- Включать и отключать формирование инцидентов в случае отсутствия счетчиков в течение определенного времени (соответствующий флаг в колонке «Контроль»).

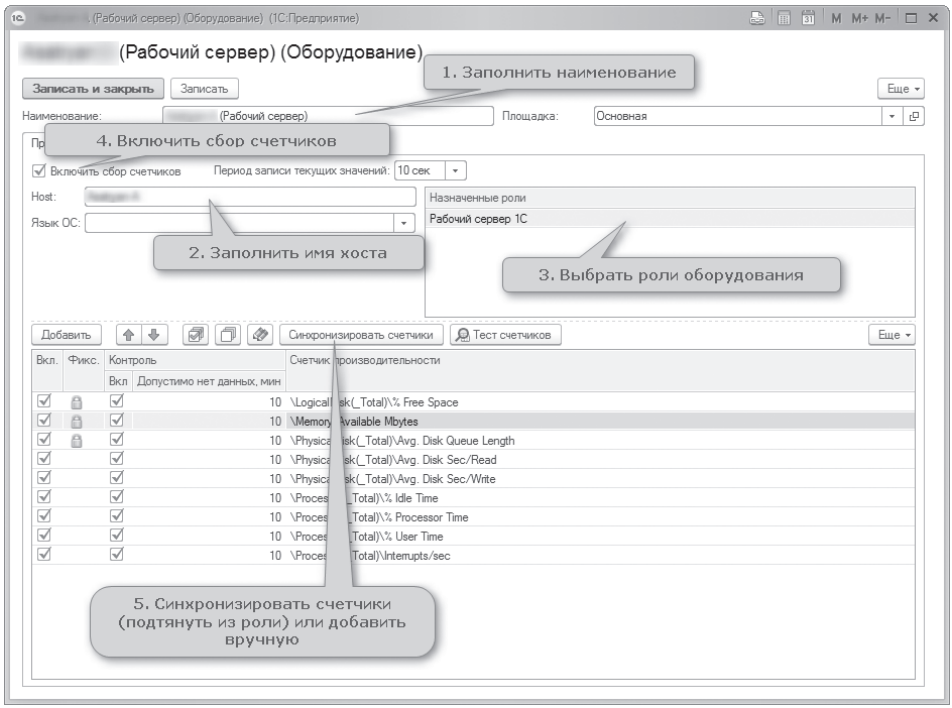


Рисунок 24. Быстрая настройка сбора счетчиков производительности при вводе оборудования в эксплуатацию

Агрегация собственных данных

ЦКК имеет встроенный веб-сервис InputStatistics, который предназначен для того, чтобы принимать данные от внешних источников, уникальных для вашей системы (для которых может и не существовать пока специализированной контрольной процедуры).

В конфигурацию ЦКК включена тестовая обработка TestInputStatistics, предназначенная для того, чтобы показать, как можно использовать реализованный веб-сервис.

На вход принимается строка в формате:

- Группа1.ВложеннаяГруппа2.Значение.Число.

Например, могут быть следующие входные данные:

- Группа.123.
- Группа.Подгруппа.234.
- Группа.Подгруппа2.345.
- Организация.Подразделение.ПлощадкаСерверов.СерверСУБД.AvgDiskQueueLength.7.

Уровни вложенности иерархии разделяются точкой. Значение (число) всегда идет после последней точки в строке. ЦКК автоматически сгруппирует значения в соответствии с заданной на входе иерархией. Значения могут храниться в разрезе узлов деревьев и листов.

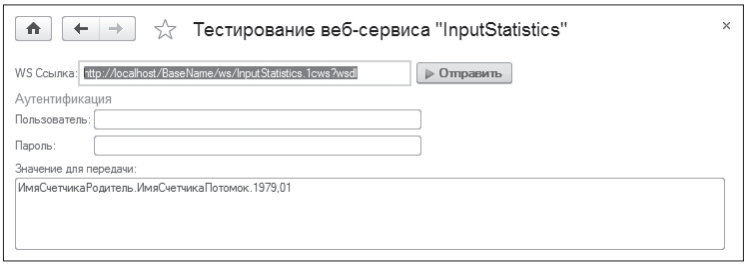


Рисунок 25. Тестирование веб-сервиса InputStatistics

Посмотреть полученную статистику можно в разделе Мониторинг, выбрав показатель Счетчики.

Так как данные на графиках в разделе Мониторинг отображаются за выбранный период, каждая точка на графике представляет собой некоторый агрегат. Для корректного отображения данных необходимо выбрать, как будут сворачиваться данные за выбранный период. Можно:

- просуммировать;
- посчитать среднее;
- показать количество вызовов веб-сервиса в случае с переданной ему иерархией счетчиков.

По выбранным счетчикам также можно настроить гибкие оповещения и получать e-mail и/или sms в случае отклонения выбранного счетчика (либо группы счетчиков) от заданной величины.

Пример настройки сбора данных по загруженности оборудования с помощью PowerShell (3.0 или 4.0) для Windows-серверов с агрегацией данных в ЦКК

Нужно выполнить следующие действия:

1. Скопировать файл скрипта srv_perf.ps1 и файл настройки srv_perf.xml на сервер, с которого (!) будет осуществляться контроль.
2. Добавить на все контролируемые (!) серверы пользователя, от имени которого будем получать необходимые данные. Пусть пользователь будет QMC_WS.
3. Добавить пользователя QMC_WS в группу Performance Monitor Users на всех контролируемых серверах.
4. Открыть с контролирующего сервера файл настройки srv_perf.xml.
5. Если контролируемые серверы не являются серверами СУБД с установленным и используемым MS SQL Server (т.е. являются рабочими серверами «1С»), тогда нужно удалить все счетчики, которые находятся между

```
<!-- For MS SQL Server only -->
```

и

```
<!-- END For MS SQL Server only -->
```

Все указанные счетчики должны нормально собираться с помощью Performance Monitor. Если какие-то счетчики называются по-другому, требуется заменить их.

Обратите внимание! В файле настройки указаны счетчики для логических дисков C, D и E. Если таких дисков у вас нет, требуется заменить эти строки либо удалить выбранные счетчики.

1. В разделе:

```
<Parameters> <Parameters>
```

перечислить все серверы, как указано в скрипте в примере для сервера, например:

```
<Servers>
        <Server>Server1C-1</Server>
        <Server>Server1C-2</Server>
</Servers>
```

2. В строке:

```
<WSURI>http://ServerQMC/qmc/ws/InputStatistics.1cws?wsdl</WSURI>
```

указать путь до опубликованного веб-сервиса в ЦКК. Пусть требуется аутентификация для подключения к веб-сервису. Для этого пусть пользователь будет QMC, а его пароль – QMC123.

3. Сохранить отредактированный файл srv_perf.xml.
4. Запустить один раз скрипт из PowerShell:

```
.\ srv_perf.ps1
```

Указать один раз логин и пароль (например: "", "123").

Убедиться, что рядом со скриптом появился файл wsInputStatistics.xml с сохраненными данными аутентификации, привязанными к серверу.

5. Открыть srv_perf.ps1.
6. Найти строки:

```
## need one time to create password file wsInputStatistics.xml. If changed user profile
                                                    or file deleted we need uncomment line below & run script interactly
$cred = Get-Credential | EXPORT-CLIXML $cred_xml_path
```

Поставить комментарий на вторую строку. Должно получиться так:

```
##$cred = Get-Credential | EXPORT-CLIXML $cred_xml_path
```

7. Сохранить файл.
8. Создать задачу в Task Scheduler от имени созданного пользователя QMC_WS.
9. В задаче запустить указанный скрипт по расписанию, например раз в минуту.

На контролирующем сервере пользователь QMC_WS должен обладать соответствующими правами (на запуск скриптов PowerShell).

Если в скрипте указано:

```
$debug = $true
```

то всегда будет создаваться журнал, путь к которому указан в файле srv_perf.xml.

Если указать:

```
$debug = $
```

то при безошибочной работе журнал создаваться не должен.

10. Убедиться, что в ЦКК в течение 5 минут появились нужные данные. Это можно проверить, если зайти в раздел Мониторинг, добавить счетчик «Счетчики» и указать нужный счетчик.

Настройка и использование агента ЦКК

ЦКК может работать как самостоятельно, так и в связке с агентом ЦКК. Агент ЦКК – это java-приложение, устанавливаемое на контролируемый сервер. Использование агента ЦКК позволяет перераспределить нагрузку с сервера приложений ЦКК на агентов и добавляет дополнительную функциональность, такую как:

- Контроль параметров оборудования:
  - версия и разрядность операционной системы;
  - установленные обновления ОС (для ОС Windows);
  - объем оперативной памяти;
  - процессоры;
  - диски;
  - сетевые адаптеры.
- Контроль основных настроек оборудования:
  - ключи реестра (для ОС Windows);
  - настроенные диапазоны портов (для ОС Windows);
  - переменные окружения.

Штатный сценарий работы – один агент ЦКК, связанный с одной базой ЦКК, на одну машину.

Подготовка к использованию агента ЦКК

1. Для использования внешнего агента необходимо выполнение следующих условий:
2. В информационной базе ЦКК должен быть опубликован http-сервис AgentQMC.
3. В информационную базу ЦКК должен быть добавлен пользователь с одной из ролей:
   - **АгентПолныеПрава** – для автоматического добавления контролируемых хостов;
   - **Агент** – если необходим только контроль соединения.
4. На контролируемый компьютер (хост) необходимо установить последнюю версию виртуальной машины Java.
5. Если мониторинг машин в ЦКК уже настроен ранее и вы собираетесь перейти на мониторинг с использованием агента ЦКК, то необходимо убедиться, что имя хоста (поле Host) в единице оборудования ЦКК соответствует имени компьютера (как в свойствах компьютера). Агент при автоматической регистрации в ЦКК ищет соответствие по имени хоста. Если не найдется подходящая по имени хоста единица оборудования, агент ЦКК создаст в ЦКК новую.

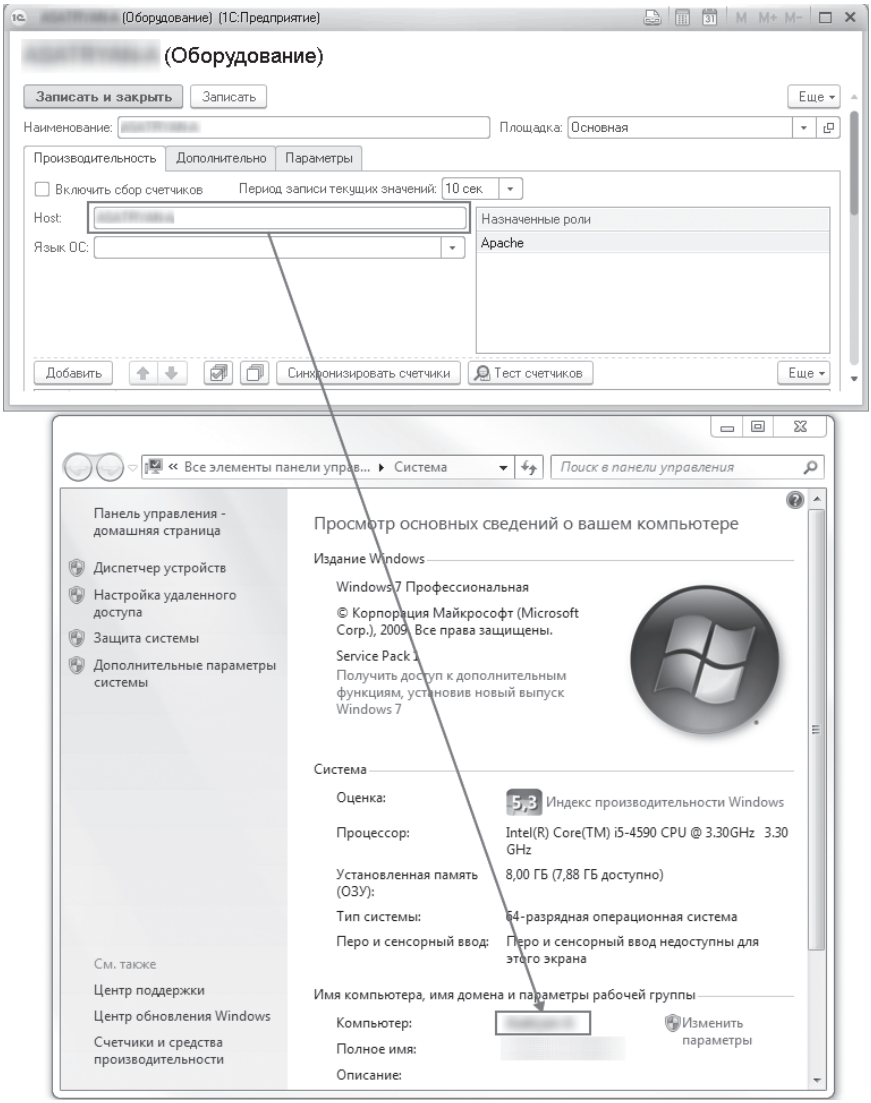


Рисунок 26. Идентификация агента ЦКК по имени хоста

Установка агента ЦКК под Windows

Агент устанавливается как обычное Windows-приложение из файла установки setup.exe. Также можно выполнить установку на несколько машин, используя команду msiexec с ключом /i.

На одном из шагов установки есть возможность указать параметры работы агента. Вид окна установки с параметрами агента представлен на рисунке 27.

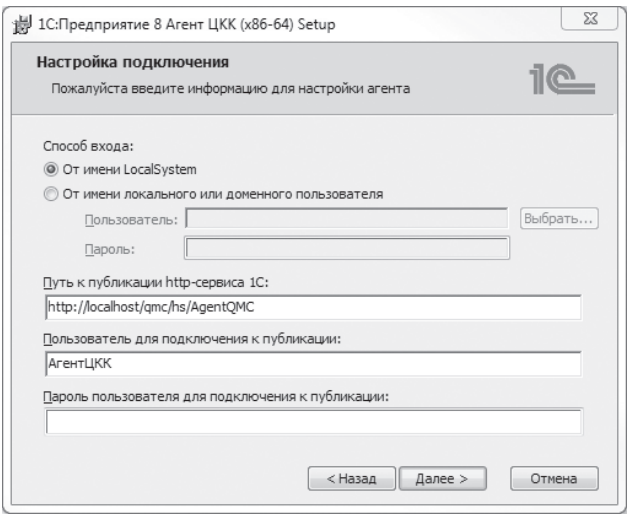


Рисунок 27. Окно установки агента ЦКК на этапе настройки параметров

Перечислим параметры, которые можно настроить при установке агента ЦКК в окне установки:

- **Способ входа** – параметр позволяет выбрать пользователя, под которым будет запускаться служба агента (AgentQMC). У этого пользователя должен быть доступ к каталогам установки агента ЦКК, каталогу сбора дампов, а также данный пользователь должен быть прописан в локальной политике безопасности «Вход в качестве службы».
- **Путь к публикации http-сервиса** – сетевой путь к публикации http-сервиса AgentQMC в ЦКК.
- **Пользователь для подключения к публикации** – пользователь ЦКК, под которым будет работать агент ЦКК (тот пользователь, которому на этапе подготовки к использованию агента давали роль **АгентПолныеПрава** или **Агент**).
- **Пароль пользователя для подключении к публикации** – пароль пользователя ЦКК, под которым будет работать агент.

Если в процессе установки указали некорректные параметры, их всегда можно изменить в файле настроек агента ЦКК settings.xml, который лежит в каталоге установленной версии агента.

Структура файлов в каталоге установки агента ЦКК

В каталоге установки агента ЦКК (AgentQMC) под каждую версию создается отдельная папка. Также в этом каталоге лежит папка conf. Она хранит в себе конфигурационный файл conf.cfg, в котором содержится уникальный идентификатор агента. Идентификатор формируется при первой установке агента и не меняется при обновлении агента на новые версии. По данному идентификатору агент регистрируется в ЦКК.

Пример содержимого файла conf.cfg:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<settings>
   <Id>d3be60a9-3e1f-4135-bb5a-73c62730e0f5</Id>
</settings>
```

Содержимое каталога установленной версии агента ЦКК представлено на рисунке:

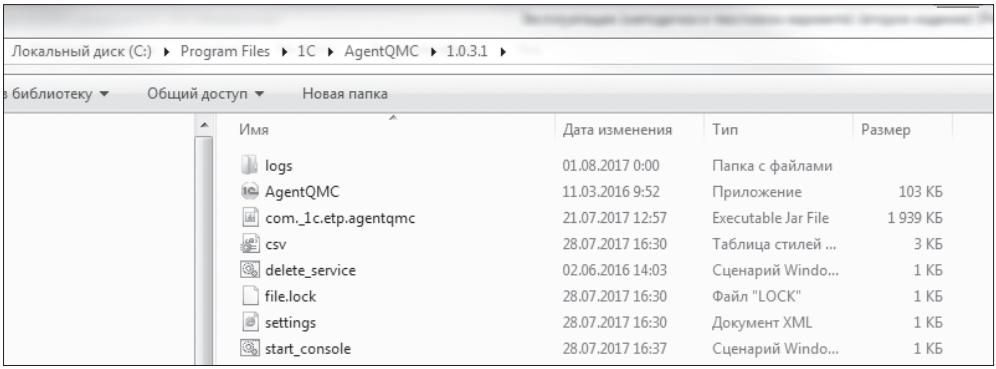


Рисунок 28. Структура файлов в каталоге установки агента ЦКК

Перечислим основные значимые файлы и каталоги:

- Папка logs – хранит файлы логов агента ЦКК в формате *.log в разрезе суток. В логах отражаются события связи агента ЦКК с ЦКК, ошибки при выполнении операций. Периодические операции работы агента (сбор дампов, проверка доступности и т.д.), если они завершились штатно, не логируются.
- AgentQMC – приложение, позволяющее запустить агента ЦКК как службу.
- com._1c.etp.agentqmc – jar-файл агента ЦКК (ведь агент – это java-приложение).
- delete_service.bat – сценарий Windows для удаления службы агента ЦКК.
- settings.xml – файл настроек агента ЦКК, в котором хранятся параметры подключения к ЦКК, включаются и отключаются функции самого агента.

Пример содержимого файла settings.xml:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Settings>
   <ConfPath>C:\Program Files\1C\AgentQMC\conf\</ConfPath>
   <Connector>
      <Type>HTTP</Type>
      <URL_Service>http://localhost/qmc/hs/AgentQMC</URL_Service>
      <Port>8095</Port>
      <Login>АгентЦКК</Login>
      <Password></Password>
   </Connector>
   <Dumps>
      <Enable>false</Enable>
      <Source>C:\Dumps\</Source>
      <Storage>C:\Dumps\Export\</Storage>
   </Dumps>
</Settings>
```

В соответствующих блоках файла settings.xml:

- путь к каталогу конфигурационного файла – <ConfPath>;
- параметры подключения к ЦКК – <Connector>;
- параметры сбора дампов – <Dumps>.

Взаимодействие ЦКК и агента ЦКК

После установки и запуска службы агента ЦКК агент пытается «познакомиться» с ЦКК – зарегистрироваться в ЦКК, прописав свой идентификатор в справочнике Агенты ЦКК, и прописаться в единице оборудования, на которой он установлен. В момент запуска службы агент устанавливает соединение с базой ЦКК посредством вызова метода HandShake http-сервиса AgentQMC. Признаком успешной установки связи агента с базой ЦКК является появление в справочнике Агенты ЦКК соответствующей строчки с именем хоста.

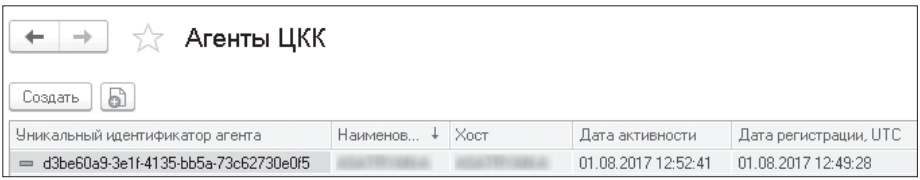


Рисунок 29. Регистрация идентификатора агента в ЦКК

В логе агента ЦКК в случае успешной процедуры регистрации появится такой текст:

```
01.08.2017 15:21:49.204 [Thread-3] INFO  com._1c.etp.agentqmc.services.ServiceRemoteControl - Remote control is stopped.
01.08.2017 15:21:49.204 [Thread-3] INFO  java.lang.Class - STOP AgentQMC version 1.0.3.1, pid 21844.
01.08.2017 15:21:49.304 [main] INFO  com._1c.etp.agentqmc.services.ServiceRemoteControl - Remote control is stopped.
01.08.2017 15:21:50.300 [Thread-2] INFO  com._1c.etp.agentqmc.ShutDownHook - ShutDownHook run.
01.08.2017 15:21:50.300 [Thread-2] INFO  com._1c.etp.agentqmc.services.ServiceRemoteControl - Remote control is stopped.
01.08.2017 15:21:50.300 [Thread-2] INFO  java.lang.Class - STOP AgentQMC version 1.0.3.1, pid 21844.
01.08.2017 15:49:24.516 [main] INFO  com._1c.etp.agentqmc.MainService - START AgentQMC version 1.0.3.1, pid 19164.
01.08.2017 15:49:24.517 [main] INFO  com._1c.etp.agentqmc.settings.SettingsXML - read file settings
                                     'C:\Program Files\1C\AgentQMC\1.0.3.1\settings.xml'.
01.08.2017 15:49:24.517 [main] INFO  com._1c.etp.agentqmc.settings.SettingsXML - Registration dumps OFF. Source=
                                     'C:\Dumps\', Storage='C:\Dumps\Export\'.
01.08.2017 15:49:24.517 [main] INFO  com._1c.etp.agentqmc.conf.ConfXML - read file conf
                                     'C:\Program Files\1C\AgentQMC\conf\conf.cfg'.
01.08.2017 15:49:24.517 [main] INFO  com._1c.etp.agentqmc.MainService - AgentQMC ID=d3be60a9-3e1f-4135-bb5a-73c62730e0f5
01.08.2017 15:49:26.763 [main] INFO  com._1c.etp.agentqmc.services.ServiceComputerInfo - host=HOST_NAME;os name=
                                       Microsoft Windows 7 Professional ;osArch=64-bit;os version=6.1.7601;user language=ru.
01.08.2017 15:49:29.160 [main] INFO  com._1c.etp.agentqmc.connector.ConnectorHttp - POST to
                                     http://localhost/qmc/hs/AgentQMC/Handshake';response code=200;
01.08.2017 15:49:30.545 [main] INFO  com._1c.etp.agentqmc.connector.ConnectorHttp - Http server is listening on port 8095
```

Если агент ЦКК не зарегистрировался в базе, а в логах агента пишутся сообщения о кодах возврата, отличных от 200, значит, при настройке допущены ошибки и агенту не удается корректно связаться с ЦКК. Для детального расследования проблемы, если из стандартных логов агента причина непонятна, можно включить подробное логирование в режиме trace.

Включение режима trace для логов

Для включения подробного логирования в режиме trace необходимо выполнить следующие манипуляции:

В каталоге установки агента ЦКК (по умолчанию C:\Program Files\1C\AgentQMC\<номер версии>) найти файл com._1c.etp.agentqmc.jar.

Открыть этот файл любым архиватором.

В перечне файлов архива найти файл log4j2.xml и сохранить его из архива в каталог установки агента ЦКК (по умолчанию C:\Program Files\1C\AgentQMC\<номер версии>).

В сохраненном файле log4j2.xml заменить строку *Root level="info"* на *Root level="trace"* и сохранить изменения.

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="INFO">

  <Appenders>
    <Console name="Console" target="SYSTEM_OUT">
      <PatternLayout pattern="%d{dd.MM.yyyy HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n"/>
    </Console>
    <RollingFile name="RollingFile" fileName="logs/app.log" filePattern="logs/%d{yyyy-MM-dd}-%i.log">
      <PatternLayout pattern="%d{dd.MM.yyyy HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n" charset="UTF-8"/>
      <Policies>
        <TimeBasedTriggeringPolicy/>
        <SizeBasedTriggeringPolicy size="10 MB"/>
      </Policies>
      <DefaultRolloverStrategy max="100"/>
    </RollingFile>
  </Appenders>

  <Loggers>
    <Root level="info">
      <AppenderRef ref="RollingFile"/>
    </Root>
  </Loggers>
</Configuration>
```

Перезапустить службу агента ЦКК.

Теперь в папку logs пишутся все события для более детального расследования проблемы. Не рекомендуется оставлять данный режим включенным постоянно, так как логи могут занимать значительное место на диске, а при штатной работе агента такой детализации логов не требуется. Для возврата к стандартному режиму логирования (info) достаточно удалить ранее сохраненный файл log4j2.xml из каталога установки агента ЦКК и перезапустить службу.

Настройка технологического журнала

Имеет смысл всегда настраивать хотя бы минимальный технологический журнал для отслеживания качества работы системы. При этом хорошей практикой будет настройка минимального журнала на всех серверах (тестовых, подготовительных, продукционных и серверах разработки). Сбор минимального технологического журнала является практически бесплатной возможностью всегда отвечать на вопрос: «А что же именно произошло?»

Настройка сбора технологического журнала производится в файле logcfg.xml.

Расположение файла logcfg.xml, а также его структура подробно описаны в документации: «Приложение 3. Описание и расположение служебных файлов» – «3.14. logcfg.xml».

При любой настройке технологического журнала следует всегда контролировать рост объема вашего журнала в течение некоторого времени. Контроль можно выполнять с помощью счетчиков (% свободного места на диске).

Серверный технологический журнал

Настройка минимального технологического журнала может выглядеть следующим образом:

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="http://v8.1c.ru/v8/tech-log">
<dump create="true" location="C:\DUMPS" type="0" prntscrn="false"/>

<log location="C:\LOGS\All" history="28">
  <event>
     <eq property="Name" value="excp"/>
  </event>
   <property name="all"/>
</log>

</config>
```

Такой технологический журнал будет:

- Включать запись дампов от аварийных завершений всех процессов кластера. Объем дампов будет минимальным (type="0") и не будет содержать дополнительных сегментов данных и памяти процесса.
- Содержать все исключения, которые были сгенерированы процессами технологической платформы (*не из кода на встроенном языке!*). Не все такие исключения являются ошибками или явными проблемами. Однако их наличие должно обратить внимание администратора (эксплуататора) площадки серверов на то, что, возможно, наблюдается какая-либо проблема.

Зачем нужен такой журнал:

- Вы можете оценить стабильность работы системы – число аварийных завершений процессов. При этом если в имени файла дампа совпадают имя процесса, версия и смещение (rphost_8.3.4.437_01234567), то, скорее всего, такие аварийные завершения были вызваны одной и той же причиной.

- Причину аварийных завершений процессов вы сможете узнать из технологического журнала. Имя файла дампа, сгенерированное технологической платформой, всегда *заканчивается на PID* процесса. Проще всего найти нужный файл технологического журнала так:
  - найти файл с именем процесса и PID (например, rphost_1234);
  - затем открыть последний (в хронологическом порядке) файл;
  - в конце этого файла обязательно будет событие EXCP, в котором может быть указан стек кода на встроенном языке, при выполнении которого и произошло аварийное завершение работы процесса.
- Можно оценить общее число исключений и *сгруппировать их по типам* (описанию в свойстве Descr), для того чтобы оценить, на что стоит обратить наибольшее внимание.
- Можно оценить общее число ошибок параллельной работы (ошибок блокировок и взаимоблокировок). Все такие ошибки обязательно записываются в технологический журнал. Тексты ошибок блокировок, которые возникли на уровне СУБД, могут различаться в зависимости от используемой СУБД, ее локализации и версии. В любом случае ВСЕ ошибки, которые произошли на уровне СУБД, будут «подписаны» соответствующим образом (например, «Microsoft OLE DB Provider for SQL Server»).

Ошибки блокировок на уровне технологической платформы всегда будут записаны в технологический журнал и будут иметь вид (текст в поле Descr будет включать в себя):

- «Превышено максимальное время ожидания предоставления блокировки» – для таймаутов на управляемых блокировках;
- «Неустранимый конфликт блокировок» – для взаимоблокировок на управляемых блокировках.

Такого (минимального) технологического журнала недостаточно, чтобы расследовать возникшие ошибки. Для расследования ошибок необходимо использовать ЦУП или более подробный технологический журнал.

В предложенной настройке технологический журнал будет собираться в директорию C:\LOGS\All.

Следует помнить, что НЕ должно быть никаких посторонних файлов в директории ALL и в директориях журналов процессов, которые создаст технологическая платформа. В противном случае технологические журналы записываться не будут. Будьте особенно осторожны при настройке технологических журналов, т. к. технологическая платформа никак не сообщит о том, что настройка выполнена некорректно. В худшем случае не будут записываться никакие технологические журналы на этом сервере. По этой причине рекомендуется всегда после настройки технологического журнала подождать одну минуту (за это время все процессы кластера серверов гарантированно перечитают конфигурационный файл технологического журнала) и убедиться, что технологический журнал собран и ведется в соответствии с вашими ожиданиями.

На всех продукционных площадках предлагается по умолчанию всегда настраивать следующий технологический журнал.

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="http://v8.1c.ru/v8/tech-log">
<dump create="true" location="C:\DUMPS" type="3" prntscrn="false"/>

<log location="C:\LOGS\All" history="28">
 <event>
    <eq property="Name" value="EXCP"/>
 </event>
<event>
    <eq property="Name" value="EXCPCNTX"/>
 </event>
 <event>
    <eq property="Name" value="CONN"/>
 </event>
 <event>
    <eq property="Name" value="PROC"/>
 </event>
 <event>
    <eq property="Name" value="ADMIN"/>
 </event>
 <event>
    <eq property="Name" value="SESN"/>
 </event>
 <event>
    <eq property="Name" value="CLSTR"/>
 </event>
  <property name="all"/>
</log>

</config>
```

Такая настройка позволяет собирать минимальный объем информации, на основании которого можно расследовать до 80 % проблем, возникающих при работе технологической платформы. В случае крайне нестабильной работы системы будет занимать много места директория с дампами процессов (C:\DUMPS), так как в данном случае будут записываться полные снимки процессов в момент их аварийного завершения. В случае нестабильной работы рекомендуется настроить автоматический разбор дампов из такой директории – например, с помощью контрольной процедуры Контроль устойчивости системы в ЦКК.

Директория с технологическим журналом (C:\LOGS\All) при нормальной работе не должна превышать пары сотен мегабайтов за сутки (конечно же, все зависит от нагрузки и конфигурации системы; нужно учитывать, что оценка может оказаться очень грубой и некорректной). Так как по истечении заданного времени (history="28") «старый» технологический журнал будет автоматически удаляться технологической платформой (для того чтобы избежать забивания журналами дискового пространства), рекомендуется настроить автоматическое копирование (например, раз в сутки) всех журналов со сжатием (например, в zip) на другой ресурс с помощью планировщика операционной системы. Таким образом вы будете иметь журналы за любой момент работы системы.

Полезным может оказаться технологический журнал, в который будут попадать все длительные события. Настройка такого технологического журнала может выглядеть так:

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="http://v8.1c.ru/v8/tech-log">

<log location="C:\LOGS\LongEvents" history="28">
  <event>
     <ne property="Name" value=""/>
     <ge property="Duration" value="20000"/>
  </event>
   <property name="all"/>
</log>

</config>
```

Файл настройки приведен для версии технологической платформы 8.3, в фильтре по полю Duration указано время 20 секунд (в сотнях микросекунд; для настройки длительности в микросекундах используется поле Durationus). Все события, которые попадают в такой журнал (при нормальной работе системы такой журнал должен быть скромных размеров), должны быть предметом рассмотрения – в первую очередь события SDBL со свойством Func=CommitTransaction или Func=RollbackTransaction. Такие события будут иметь длительность внешней транзакции (вложенные транзакции технологической платформой не поддерживаются). Если транзакция длится более 20 секунд (скорее всего, в рамках транзакции будут установлены транзакционные блокировки на какие-либо ресурсы), она может стать «виновником» ошибок блокировок. Также внимание стоит уделить длительным запросам.

Для расследования ошибок на управляемых блокировках необходим технологический журнал вида:

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="http://v8.1c.ru/v8/tech-log">

<log location="C:\LOGS\TLOCKS" history="4">
  <event>
     <eq property="Name" value="TLOCK"/>
  </event>
  <event>
     <eq property="Name" value="TTIMEOUT"/>
  </event>
  <event>
     <eq property="Name" value="TDEADLOCK"/>
  </event>
   <property name="all"/>
</log>

</config>
```

Следует обратить внимание, что журнал может быть большого объема, а длительность его ротации в примере указана равной 4 часам. Ниже приведен пример того, как можно провести расследование ошибок управляемых блокировок (на примере управляемых взаимоблокировок).

Для того чтобы воспроизвести и расследовать простейшую взаимоблокировку, необходимо выполнить следующие шаги (просьба учитывать, что это «учебный» пример, и не писать так в своих конфигурациях):

1. Создать конфигурацию с регистром сведений РегистрСведений1. Регистр независимый, непериодический.
2. Сделать обработку, в которой есть две команды:

```
&НаКлиенте
Процедура ПервыйУчастник(Команда)
  ПервыйУчастникНаСервере();
КонецПроцедуры

&НаСервереБезКонтекста
Процедура ПервыйУчастникНаСервере()
  НачатьТранзакцию();
  НаборЗаписейРегистрСведений1 = РегистрыСведений.РегистрСведений1.СоздатьНаборЗаписей();
  НаборЗаписейРегистрСведений1.Отбор.Измерение1.Установить("Test1");
  НаборЗаписейРегистрСведений1.Отбор.Измерение2.Установить("Test2");
  //TLOCK shared
  НаборЗаписейРегистрСведений1.Прочитать();

  //5 секунд паузы
  СделатьПаузу(5000);


  //TLOCK exclusive
  НаборЗаписейРегистрСведений1.Записать();

  ЗафиксироватьТранзакцию();
КонецПроцедуры

&НаКлиенте
Процедура ВторойУчастник(Команда)
  ВторойУчастникНаСервере();
КонецПроцедуры

&НаСервереБезКонтекста
Процедура ВторойУчастникНаСервере()
  НачатьТранзакцию();
  НаборЗаписейРегистрСведений1 = РегистрыСведений.РегистрСведений1.СоздатьНаборЗаписей();
  НаборЗаписейРегистрСведений1.Отбор.Измерение1.Установить("Test1");
  НаборЗаписейРегистрСведений1.Отбор.Измерение2.Установить("Test2");
  //TLOCK shared
  НаборЗаписейРегистрСведений1.Прочитать();

  //5 секунд паузы
  СделатьПаузу(5000);

  //TLOCK exclusive
  НаборЗаписейРегистрСведений1.Записать();

  ЗафиксироватьТранзакцию();
КонецПроцедуры
```

СделатьПаузу() – метод, реализованный в типовой конфигурации КИП:ТестЦентр, который предназначен для паузы; принимает параметр в мс.

3. Настроить технологический журнал.

4. Запустить два клиента: первый выполняет первую команду, второй – вторую.

   Получаем ошибку взаимоблокировки на управляемых блокировках. Смотрим в собранный технологический журнал:

```
10:58.515038-3,TLOCK,4,process=rphost,p:processName=test_lock3,t:clientID=41,t:applicationName=1CV8C,t:computerName=
TEST-AN,t:connectID=16,SessionID=8,AppID=1CV8C,Regions=InfoRg10.DIMS,Locks='InfoRg10.DIMS Shared Fld11=
"Test1" Fld12="Test2"',WaitConnections=,Context='Форма.Вызов : ОбщаяФорма.РабочийСтол.Модуль.ПервыйУчастникНаСервере
ОбщаяФорма.РабочийСтол.Форма : 19 : НаборЗаписейРегистрСведений1.Прочитать();'
11:00.496028-3,TLOCK,4,process=rphost,p:processName=test_lock3,t:clientID=45,t:applicationName=1CV8,t:computerName=
TEST-AN,t:connectID=17,SessionID=10,Usr=DefUser,Regions=InfoRg10.DIMS,Locks='InfoRg10.DIMS Shared Fld11=
"Test1" Fld12="Test2"',WaitConnections=,Context='Форма.Вызов : ОбщаяФорма.РабочийСтол.Модуль.ВторойУчастникНаСервере
ОбщаяФорма.РабочийСтол.Форма : 50 : НаборЗаписейРегистрСведений1.Прочитать();'
11:05.519001-0,TDEADLOCK,5,process=rphost,p:processName=test_lock3,t:clientID=45,t:applicationName=1CV8,
t:computerName=TEST-AN,t:connectID=17,SessionID=10,Usr=DefUser,DeadlockConnectionIntersections='17 16InfoRg10.
DIMS Exclusive Fld11="Test1" Fld12="Test2",16 17 InfoRg10.DIMS Exclusive Fld11="Test1" Fld12="Test2"',Context='Форма.Вызов :
ОбщаяФорма.РабочийСтол.Модуль.ВторойУчастникНаСервере
ОбщаяФорма.РабочийСтол.Форма : 58 : НаборЗаписейРегистрСведений1.Записать();'
11:05.519003-16002,TLOCK,4,process=rphost,p:processName=test_lock3,t:clientID=45,t:applicationName=1CV8,t:computerName=
TEST-AN,t:connectID=17,SessionID=10,Usr=DefUser,Regions=InfoRg10.DIMS,Locks='InfoRg10.DIMS Exclusive Fld11=
"Test1" Fld12="Test2"',WaitConnections=16,Context='Форма.Вызов :
ОбщаяФорма.РабочийСтол.Модуль.ВторойУчастникНаСервере
ОбщаяФорма.РабочийСтол.Форма : 58 : НаборЗаписейРегистрСведений1.Записать();'
11:05.519024-1997023,TLOCK,4,process=rphost,p:processName=test_lock3,t:clientID=41,t:applicationName=1CV8C,
t:computerName=TEST-AN,t:connectID=16,SessionID=8,AppID=1CV8C,Regions=InfoRg10.DIMS,Locks=
'InfoRg10.DIMS Exclusive Fld11="Test1" Fld12="Test2"',WaitConnections=17,Context='Форма.Вызов :
 ОбщаяФорма.РабочийСтол.Модуль.ПервыйУчастникНаСервере
ОбщаяФорма.РабочийСтол.Форма : 26 : НаборЗаписейРегистрСведений1.Записать();'
```

5. Найти событие TDEADLOCK и выписать свойства поля:

```
DeadlockConnectionIntersections='
17 16 InfoRg10.DIMS Exclusive Fld11="Test1" Fld12="Test2",
16 17 InfoRg10.DIMS Exclusive Fld11="Test1" Fld12="Test2"'
```

   Видим, что ошибка произошла на регистре сведений InfoRg10 при попытке установить исключительную управляемую блокировку на поля:

```
Fld11="Test1" Fld12="Test2"
```

   16 и 17 – это номера

```
t:connectID
```

   участников взаимоблокировки. До события

```
TDEADLOCK
```

   указаны события

```
TLOCK
```

   участников 16 и 17. Например:

```
10:58.515038-3,TLOCK,4,process=rphost,p:processName=test_lock3,t:clientID=41,t:applicationName=1CV8C,t:computerName=
TEST-AN,t:connectID=16,SessionID=8,AppID=1CV8C,Regions=InfoRg10.DIMS,Locks='InfoRg10.DIMS Shared Fld11=
"Test1" Fld12="Test2"',WaitConnections=,Context='Форма.Вызов : ОбщаяФорма.РабочийСтол.Модуль.ПервыйУчастникНаСервере
ОбщаяФорма.РабочийСтол.Форма : 19 : НаборЗаписейРегистрСведений1.Прочитать();'
```

Видим, что

```
t:connectID=16
```

установил разделяемую управляемую блокировку на поля регистра сведений:

```
Locks='InfoRg10.DIMS Shared Fld11="Test1" Fld12="Test2"'
```

Эту операцию первый участник выполнил из:

```
Форма.Вызов : ОбщаяФорма.РабочийСтол.Модуль.ПервыйУчастникНаСервере
ОбщаяФорма.РабочийСтол.Форма : 19 : НаборЗаписейРегистрСведений1.Прочитать();
```

Управляемая блокировка (разделяемая или исключительная) снимается только в конце внешней транзакции.

При чтении набора записей технологическая платформа сама устанавливает разделяемую управляемую блокировку.

При записи набора записей технологическая платформа сама устанавливает исключительную управляемую блокировку.

Собственно блокировка при записи:

```
11:05.519024-1997023,TLOCK,4,process=rphost,p:processName=test_lock3,t:clientID=41,t:applicationName=
1CV8C,t:computerName=TEST-AN,t:connectID=16,SessionID=8,AppID=1CV8C,Regions=InfoRg10.DIMS,Locks='InfoRg10.
DIMS Exclusive Fld11="Test1" Fld12="Test2"',WaitConnections=17,Context='Форма.Вызов : ОбщаяФорма.РабочийСтол.Модуль.
ПервыйУчастникНаСервере
ОбщаяФорма.РабочийСтол.Форма : 26 : НаборЗаписейРегистрСведений1.Записать();'
```

Мы видим, что установить он ее не смог и ждал в течение

```
1997023
```

микросекунд участника

```
WaitConnections=17
```

то есть с номером

```
t:connectID=17
```

Это же симметрично сделал второй участник взаимоблокировки.

Таким образом мы по технологическому журналу полностью выяснили, что именно и как именно произошло.

6. В статье https://kb.1c.ru/articleView.jsp?id=46 указан этот сценарий в разделе «Повышение уровня блокировки ресурса в рамках одной транзакции».

   Для того чтобы исправить эту ошибку в коде конфигурации, нужно, чтобы *блокировка в транзакции изначально осуществлялась с максимальным необходимым уровнем изоляции*. То есть необходимо установить управляемую исключительную блокировку перед чтением набора записей.

Для расследования причин неоптимальной работы запросов может оказаться полезным следующий технологический журнал (пример сделан для сервера СУБД MS SQL Server):

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="http://v8.1c.ru/v8/tech-log">

<log location="C:\LOGS\DBMSSQL" history="4">
  <event>
     <eq property="Name" value="DBMSSQL"/>
     <like property="p:processName" value="%MyInfoBase%"/>
     <eq property="User" value="Василий"/>
  </event>
   <property name="all"/>
</log>
<plansql/ >
</config>
```

В примере настроены планы запросов (<plansql/>). Несмотря на то что сбор планов включается на СУБД для всех баз, планы будут записываться только для тех событий, которые удовлетворяют указанным фильтрам. Рекомендуем всегда настраивать фильтрацию как можно более тонко. В этом случае ваш журнал не будет иметь значительного объема, а его разбор будет удобным. В примере приведена настройка фильтрации по информационной базе (MyInfoBase) и по пользователю («Василий»).

Типичные причины неоптимальной работы запросов и методы оптимизации можно найти в статье https://kb.1c.ru/articleView.jsp?id=44.

Клиентский технологический журнал

Наиболее распространенной задачей, в которой требуется настройка клиентского технологического журнала, является расследование длительного входа в систему. Для того чтобы расследовать, что именно пытается делать клиентское приложение (например, какую форму открывает), нужен клиентский технологический журнал. Пример настройки полного клиентского технологического журнала:

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="http://v8.1c.ru/v8/tech-log">

<log location="C:\LOGS\1cv8" history="4">
  <event>
     <ne property="Name" value=""/>
  </event>
   <property name="all"/>
</log>
</config>
```

Рекомендуется в общем случае настраивать полный технологический журнал для анализа работы клиентских приложений. Такой журнал не будет иметь большого объема (ввиду того что клиентское приложение одно и выполняет действий значительно меньше, чем сервер).

ВНИМАНИЕ!

Не следует настраивать такой журнал на рабочем сервере.

В общем случае не рекомендуется запускать клиентские приложения на продукционных серверах.

Разбор технологического журнала

Ниже приводится пример, показывающий, как можно разбирать технологический журнал. В этом примере подготовлен такой учебный шаблон, который позволит с минимальными трудозатратами получить необходимый результат. Возможно, этот учебный шаблон не будет самым оптимальным (ввиду своей общности).

В случае если разбирается технологический журнал на Windows-сервере, возможно использовать специальное ПО, например Cygwin.

Не забываем установить пакеты perl. Далее приведены примеры для сценария, при котором мы уже находимся в директории, где лежат директории с журналами процессов кластера.

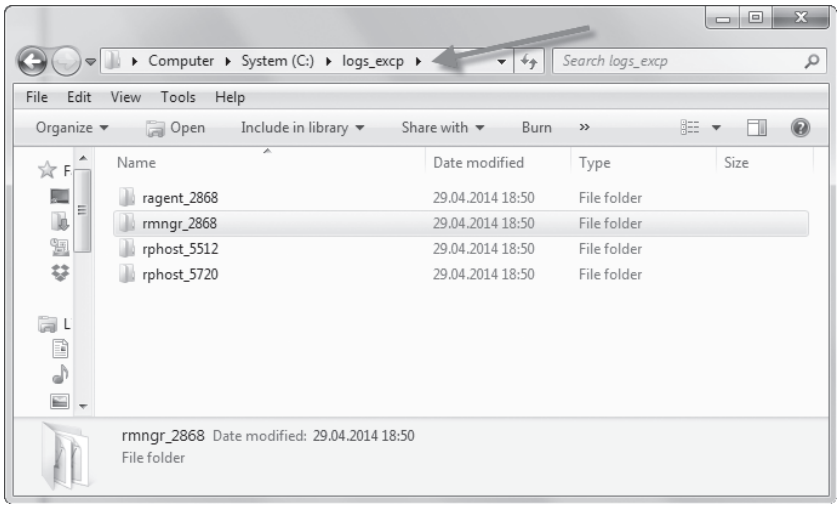


Рисунок 30. Директория с технологическими журналами процессов кластера

Например, мы хотим получить *группировку по полю Descr всех событий EXCP* только по процессам rphost.

Вывод в виде таблицы с колонками:

- Количество.
- Descr.

Выполняем:

```
cat rphost*/*.log | perl descr.pl | sort | uniq -c | sort -rn >> result.txt
```

Содержимое descr.pl

```
#!/usr/bin/perl
use strict;

my $event;
my %actions = (
   'EXCP' => [
            {
               'action' => sub {
                        my ($event) =@_;
                        my ($garbage, $context) = split /Descr='/, $event;
                        my ($descr, $garbage) = split /'/, $context;
                        $descr =~ s/|'//g;
                        $descr =~ s/\s+/ /g;

                        print "EXCP Descr $descr\n" if $descr;
               },
            },
   ],
);

while (<>) {
   $event=process_event($event) if /^\d\d:\d\d\.\d+/;
   $event .= $_;
}

sub process_event($) {
   my ($event) = @_;
   return unless $event;
   foreach my $event_type ( keys %actions ) {
            next unless $event =~ /^[^,]+,$event_type,/;
            foreach my $issue ( @{ $actions{$event_type} }) {
               &{$issue->{action}}($event);
            }
   }
}
```

Можно немного усложнить этот пример, добавив фильтрацию – например, уникальных идентификаторов и т. п. Результат можно вывести в файл res1.txt для дальнейшей обработки.

```
cat */*.log | perl ~/descr.pl | sed -r 's/.{8}-.{4}-.{4}-.{4}-.{12}/{GUID}/g' | sed -r 's/\:[0-9]{4,5}/{PORT}/g' | sed -r 's/
[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+/{ADDR}/g' | sed -r 's/сеанс.*приложение/сеанс {INFO} приложение/g' | sed -r 's/key value is ([^)]+)/
key value is {VALUE}/g' | sort | uniq -c | sort -rnb > res1.txt
```

Если журналы находятся в архивах в какой-либо директории, то можно не извлекать их предварительно.

Тогда извлечение из архива и анализ могут выглядеть следующим образом:

```
$ find //server/techlogs/2016-07-19/* -name '*.zip' -print0 | xargs -0 -i -n 1 unzip -p {} '*.log' 2>/dev/null | perl ~/descr.pl | sed -r
's/.{8}-.{4}-.{4}-.{4}-.{12}/{GUID}/g' | sed -r 's/\:[0-9]{4,5}/{PORT}/g' | sed -r 's/[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+/{ADDR}/g' | sed -r 's/сеанс
.*приложение/сеанс {INFO} приложение/g' | sed -r 's/key value is ([^)]+)/key value is {VALUE}/g' | sort | uniq -c | sort -rnb > res1.txt
```

В целом можно ровно для своей задачи сделать достаточно простой скрипт с обходом директорий:

```
#!/bin/sh

for dirs in $1;
do
   echo "Directory="$dirs;
   for archives in `find "$dirs"/* -name '*.zip' -print0`
   do
      echo "archives="$archives;
      for FilesInArchive in `unzip -l $archives | grep 'log' | awk '{print $4}'`;
         do
            echo "FilesInArchive="$FilesInArchive;
            unzip -p "$archives" "$FilesInArchive" 2>/dev/null | perl ~/descr.pl | sed -r 's/.{8}-.{4}-.{4}-.{4}-.{12}/{GUID}/g'
            | sed -r 's/\:[0-9]{4,5}/{PORT}/g' | sed -r 's/[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+/{ADDR}/g' | sed -r 's/сеанс.*приложение/сеанс {INFO}
            приложение/g' | sed -r 's/key value is ([^)]+)/key value is {VALUE}/g' | sort | uniq -c | sort -rnb | head -n 10;
         done
   done
done
```

Суть скрипта descr.pl, по сути, сводится к обработке журнала по событиям и только потом фильтрации по нужным соответствиям. При этом получить журнал по событиям можно, выполнив:

```
perl -n -e 'if (/^\d\d:\d\d\.\d+/) {$event =~ s/\n/<line>/g; print $event."\n"; $event = "";} $event .= $_; END{print $event."\n"};'
```

Смысл этого выражения сводится к тому, что мы утверждаем: все события в технологическом журнале начинаются с ^\d\d:\d\d\.\d+/, т. е. с минут, секунд и микросекунд, указанных через двоеточие и точку соответственно. В этом случае нужные фильтрации можно применить после приведения события к виду строки.

Например, мы хотим получить суммарную длительность событий DBMSSQL и SDBL с группировкой по последней строке стека на встроенном языке.

Вывод в виде таблицы с колонками:

- СуммарнаяДлительность (в микросекундах).
- Количество.
- Контекст.

```
cat rphost*/*.log | perl query.pl | awk -F'-' '{mas[$2] += $1; count[$2] += 1;} END {for (i in mas) {print mas[i] "
                                                                                    " count[i] " " i};}' | sort -rn >> result.txt
```

Содержимое query.pl

```
#!/usr/bin/perl
use strict;

my $event;
my %actions = (
   'DBMSSQL' => [
            {
               'action' => sub {
                        my ($event) =@_;
                        my ($date, $garbage) = split /,DBMSSQL/, $event;
                        $date =~ s/\d\d:\d\d\.\d+-//g;

                        my ($garbage, $context) = split /Context='/, $event;
                        if ($context =~ /\n/) {
                           my @mlc = split /\n/, $context;
                           $context = $mlc[$#mlc];
                        }
                        $context =~ s/^\s+|'$//g;
                        $context =~ s/;'/;/g;
                        print "$date-$context\n" if $context;
               },
            },
   ],
            'SDBL' => [
            {
               'action' => sub {
                        my ($event) =@_;
                        my ($date, $garbage) = split /,SDBL/, $event;
                        $date =~ s/\d\d:\d\d\.\d+-//g;

                        my ($garbage, $context) = split /Context='/, $event;
                        if ($context =~ /\n/) {
                           my @mlc = split /\n/, $context;
                           $context = $mlc[$#mlc];
                        }
                        $context =~ s/^\s+|'$//g;
                        $context =~ s/;'/;/g;
                        print "$date-$context\n" if $context;
               },
            },
   ],
);

print "\n";
while (<>) {
   $event=process_event($event) if /^\d\d:\d\d\.\d+/;
   $event .= $_;
}

sub process_event($) {
   my ($event) = @_;
   return unless $event;
   foreach my $event_type ( keys %actions ) {
            next unless $event =~ /^[^,]+,$event_type,/;
            foreach my $issue ( @{ $actions{$event_type} }) {
               &{$issue->{action}}($event);
            }
   }
}
```

Приведенный скрипт в качестве контекста указывает последнюю строку из стека вызова на встроенном языке. В случае, когда возникает необходимость получить группировку по первой строке стека на встроенном языке, достаточно строки:

```
$context = $mlc[$#mlc];
```

заменить на

```
$context = $mlc[1];
```

Предположим, что мы хотим получить суммарную длительность ожиданий на управляемых блокировках. В этом случае query.pl изменится на tlock.pl.

Содержимое tlock.pl:

```
#!/usr/bin/perl
use strict;

my $event;
my %actions = (
   'TLOCK' => [
            {
               'action' => sub {
                        my ($event) =@_;

                        my ($date, $garbage) = split /,TLOCK/, $event;
                        $date =~ s/\d\d:\d\d\.\d+-//g;

                        my ($garbage, $region) = split /Regions=/, $event;
                        my ($region, $garbage) = split /,Locks='/, $region;

                        print "$date-$region\n" if $region;
               },
            },
   ],
);

while (<>) {
   $event=process_event($event) if /^\d\d:\d\d\.\d+/;
   $event .= $_;
}

sub process_event($) {
   my ($event) = @_;
   return unless $event;
   foreach my $event_type ( keys %actions ) {
            next unless $event =~ /^[^,]+,$event_type,/;
            foreach my $issue ( @{ $actions{$event_type} }) {
               &{$issue->{action}}($event);
            }
   }
}
```

Итак, скрипт сделан таким образом, что он очень легко модифицируется под конкретную задачу. Если идея показалась вам интересной, предлагается ознакомиться с книгой «Регулярные выражения» (автор – Джеффри Фридл).

Описанные приемы хорошо подходят не только для анализа технологических журналов. Например, может возникнуть необходимость отследить системные вызовы процессов кластера серверов, чтобы понять, насколько часто процесс обращается к каким-либо ресурсам системы. Для этих целей можно воспользоваться командой strace. *Strace* – это диагностическая утилита (в ОС Linux, для Windows аналогичную функциональность предоставляет ProcMon от SysInternals), отслеживающая системные вызовы, которые представляют собой механизм трансляции, обеспечивающий интерфейс между процессом и операционной системой. Вывод этой утилиты представляется в виде:

```
open(".", O_RDONLY|O_NONBLOCK|O_LARGEFILE|O_DIRECTORY|O_CLOEXEC) = 3
fstat64(3, {st_mode=S_IFDIR|0755, st_size=4096, ...}) = 0
fcntl64(3, F_GETFD)                     = 0x1 (flags FD_CLOEXEC)
getdents64(3, /* 18 entries */, 4096)   = 496
getdents64(3, /* 0 entries */, 4096)    = 0
close(3)                                = 0
fstat64(1, {st_mode=S_IFIFO|0600, st_size=0, ...}) = 0
```

Если исследуемый процесс сейчас находится к продуктивной системе и выполняет тысячи операций, то и таких вызовов будут тысячи или десятки тысяч.

Например, может возникнуть желание понять, что делает процесс rphost, особенно в следующей ситуации:

- когда мы видим, что вызовы сеансов находятся в исполнении этим процессом сейчас;
- у нас нет под рукой (и вообще нет) исходного кода платформы «1С»;
- у нас не включен режим отладки (ключ –debug);
- процессов rphost одновременно может работать несколько с такими же симптомами;
- вывод может занимать десятки тысяч строк, а нам нужно быстро сориентироваться в ситуации.

Для того чтобы выполнить какую-либо команду по всем работающим сейчас процессам rphost, достаточно воспользоваться конструкцией:

```
for i in `pgrep rphost`; do echo "rphost =" $i; done
```

Здесь мы выведем все PID работающих процессов rphost.

Раз мы можем получить PID, мы можем и подключиться с помощью strace, и вывести по 10 строк вывода strace каждого процесса:

```
for i in `pgrep rphost`; do echo "rphost =" $i; strace -f -p $i  | head; done
```

При желании ошибки можно «выкинуть», а вывод ограничить определенными вызовами, например: открытие, чтение и запись файлов.

```
for i in `pgrep rphost`; do echo "rphost =" $i; strace -f -p $i -e open,read,write 2>&1 | head; done
```

Осталось только немного «причесать» вывод, чтобы увидеть более ясно top операций, но теперь не из 10, а из 5000, например, вызовов.

```
for i in `pgrep rphost`; do echo "rphost =" $i; strace -f -p $i -e open,read,write 2>&1 | awk -F'=' '{print $1}' | awk -F']' '{print $2}'
| sed -r 's/v8.*tmp/v8_{NAME}.tmp/g' | sed -r 's/v8.*xml/v8_{NAME}.xml/g' | sed -r 's/v8.*wsdl/v8_{NAME}_wsdl/g' | head -n 5000
| sort -nb | uniq -c | sort -rnb | head -n 20; done
```

Обратите внимание, что программа построена по абсолютно тем же принципам, что и предыдущие однострочные программы: получение данных, преобразование к нужной форме, подсчет, сортировка, вывод.

В дальнейшем под свои задачи программу можно усложнить – например, расширив список анализируемых выводов.

```
for i in `pgrep rphost`; do echo "rphost =" $i; time strace -r -f -p $i -e open,read,write,futex,stat,close,poll,lseek,openat,flock,
recvfrom 2>&1 | head -n 100000 | sed -r 's/<\.\.\. //g' | sed -r 's/ resumed/_resumed/g' | sed -r 's/,? \".*\".*, /,{STR},/g' | sed -r 's/lseek\
([0-9]+, [0-9]+, /lseek({NUM},{NUM},/g' | sed -r 's/, /,/g' | awk '{sum[$4]+=$3; count[$4]+=1;} END {for (i in sum) {if(sum[i]>0.0001)
{print sum[i] " " count[i] " " i}}}' | sort -rnb | head -n 20; done
```

Настройка Performance Monitor для Windows-серверов

Настройка сбора данных Performance Monitor необходима для оценки загруженности оборудования серверов приложений. Следует обратить внимание, что должен быть настроен сбор данных со всех серверов продукционной площадки.

Необходимо убедиться, что собираются данные по всем следующим счетчикам:

```
"\Memory\Available Mbytes"
"\Process("1cv8*")\% Processor Time"
"\Process("1cv8*")\Private Bytes"
"\Process("1cv8*")\Virtual Bytes"
"\Process("ragent*")\% Processor Time"
"\Process("ragent*")\Private Bytes"
"\Process("ragent*")\Virtual Bytes"
"\Process("rphost*")\% Processor Time"
"\Process("rphost*")\Private Bytes"
"\Process("rphost*")\Virtual Bytes"
"\Process("rmngr*")\% Processor Time"
"\Process("rmngr*")\Private Bytes"
"\Process("rmngr*")\Virtual Bytes"
"\LogicalDisk(_Total)\Free Megabytes"
"\Processor(_Total)\% Processor Time"
"\Memory\Pages/sec"
"\System\Processor Queue Length"
"\PhysicalDisk(_Total)\Avg. Disk Queue Length"
"\PhysicalDisk(*)\Avg. Disk Queue Length"
"\PhysicalDisk(*)\Avg. Disk Bytes/Read"
"\PhysicalDisk(*)\Avg. Disk Bytes/Write"
"\Network Interface(*)\Bytes Total/sec"
```

Добавить (настроить) такой набор счетчиков Performance Monitor можно командой:

```
logman create counter 1C_counter -f bincirc -c "\Memory\Available Mbytes" "\Process("1cv8*")\%% Processor Time" "\Process
("1cv8*")\Private Bytes" "\Process("1cv8*")\Virtual Bytes" "\Process("ragent*")\%% Processor Time" "\Process("ragent*")\Private Bytes"
"\Process("ragent*")\Virtual Bytes" "\Process("rphost*")\%% Processor Time" "\Process("rphost*")\Private Bytes" "\Process("rphost*")
\Virtual Bytes" "\Process("rmngr*")\%% Processor Time" "\Process("rmngr*")\Private Bytes" "\Process("rmngr*")\Virtual Bytes"
"\LogicalDisk(_Total)\Free Megabytes" "\Processor(_Total)\%% Processor Time" "\Memory(_Total)\Pages/sec" "\System(_Total)
\Processor Queue Length" "\PhysicalDisk(_Total)\Avg. Disk Queue Length" "\PhysicalDisk(*)\Avg. Disk Queue Length" "\PhysicalDisk(*)
\Avg. Disk Bytes/Read" "\PhysicalDisk(*)\Avg. Disk Bytes/Write" "\Network Interface(*)\Bytes Total/sec" -si 5 -v mmddhhmm
```

ВНИМАНИЕ!

Имена счетчиков могут незначительно различаться в зависимости от версии операционной системы.

Данные будут собираться каждые 5 секунд.

Рекомендуется также не забыть настроить планировщик задач на автозапуск выбранного счетчика (например, каждый час, если сбор данных еще не запущен). Это нужно для тех случаев, когда возникает необходимость перезапускать продукционные серверы. Обычно в такие моменты о включении сбора данных Performance Monitor вспоминают в последнюю очередь.

ГЛАВА 4

Администрирование серверов с развернутой технологической платформой «1С:Предприятие»

Настройка рабочих серверов с развернутой технологической платформой «1С:Предприятие»

Ниже приводится инструкция по настройке рабочих серверов с технологической платформой «1С:Предприятие».

Перед изучением данного материала рекомендуется предварительно прочитать раздел документации по технологической платформе «1С:Предприятие 8» (на ИТС: «Разработка и администрирование» – «1С:Предприятие 8.3.x. Документация» – «2.1. Устройство кластера серверов»).

Рекомендуется при настройке рабочего сервера пройти указанный ниже check-лист и продумать, нужна ли указанная ниже настройка в вашем конкретном случае.

Если такая настройка нужна, то необходимо выполнить ее. Важно в каждом пункте сознательно принимать решение о том, как именно вы хотите настроить рабочий сервер.

1. Определить, сколько информационных баз будет использоваться в кластере для работы пользователей.

Существует несколько вариантов развертывания:

- в продуктивной среде и подготовительной зоне;
- в тестовой зоне;
- в зоне разработки.

Наибольшие требования с точки зрения доступности информационной системы будут при развертывании в продуктивной и подготовительной зонах. В этих случаях желательно все нерабочие информационные базы вынести в отдельный кластер на отдельные рабочие серверы.

Возможно, возникнет желание сделать копию рабочей информационной базы и развернуть ее в том же кластере в продуктивной среде – например, для того чтобы восстановить определенные данные за прошлые сутки. Стоит проследить:

- чтобы к копии базы не было доступа у пользователей;
- в копии базы были выключены регламентные задания;
- копия базы не участвовала в обменах.

Не следует восстанавливать базу за предыдущий период в рабочей системе, нужно получать необходимые данные и выполнять работы в подготовительной зоне информационной системы.

Рекомендуется в продуктивной зоне настраивать кластер с минимальным числом необходимых баз, чтобы снизить возможное влияние тестовых баз на работу пользователей. В тестовой зоне и зоне разработки ограничений по числу информационных баз в кластере условно нет.

2. Определить, сколько пользователей будет работать одновременно.

   Число одновременно работающих пользователей информационной базы является одним из основных параметров, определяющих нагрузку на информационную систему.

   Этот параметр также необходим для корректного расчета конфигурации оборудования (подробности на ИТС: «Разработка и администрирование» – «Методическая поддержка для разработчиков и администраторов 1С:Предприятия 8» – «Разработчикам» – «Технологические вопросы крупных внедрений» – «Методики» – «Расчет параметров серверного оборудования»), который выполняется:

   - исходя из конфигурации системы;
   - сценария работы пользователей;
   - числа одновременно работающих пользователей;
   - используемых версий программных продуктов.

3. Настроить профили пользователей ОС, от имени которых будут запускаться процессы кластера.

   Необходимо определиться, будут ли процессы кластера серверов работать от имени различных пользователей информационной системы.

   Это может быть необходимо для того, чтобы код, который выполняется в rphost, точно не мог обратиться к каким-либо определенным данным на рабочем сервере или выполнить операцию с административными правами.

Для этого нужно:

- Подготовить файл swpuser.ini (на ИТС: «Разработка и администрирование» – «1С:Предприятие 8.3.x. Документация» – «Руководство администратора» – «Приложение 3. Описание и расположение служебных файлов» – «3.25. swpuser.ini»).
- Создать пользователей операционной системы, от имени которых будут запускаться процессы кластера.
- Создать профили соответствующих пользователей.
- Настроить ограничение прав доступа пользователей к каталогам на уровне ОС, к которым процессы не должны иметь доступа.

Для того чтобы создать профили пользователей ОС, достаточно один раз войти от их имени в ОС Windows.

4. Настроить технологические журналы и дампы.

   Для этого необходимо настроить:

   - Технологический журнал (http://kb.1c.ru/articleView.jsp?id=74#Настройка технологического журнала).
   - Сбор дампов процессов кластера средствами платформы (указанием в logcfg.xml секции dump) либо *Windows Error Reporting Services* (http://kb.1c.ru/articleView.jsp?id=96).

5. Проверить настройки операционной системы.

   5.1. Настроить рабочие серверы.

   Настройка производится в соответствии с check-листом по настройке рабочих серверов в продукционной зоне (приложение 6 данного пособия, http://kb.1c.ru/articleView.jsp?id=88).

   Если при выполнении настроек вы сознательно пропускаете какой-либо пункт, необходимо заранее иметь хорошую техническое обоснование того, почему этого делать не стоит.

   5.2. Настроить параметры сетевого стека для обеспечения возможности обработки большого числа подключений.

   Необходимо настроить рабочий сервер в соответствии с инструкцией, которая позволяет избежать ошибки «An operation on a socket could not be performed because the system lacked sufficient buffer space or because a queue was full» (http://kb.1c.ru/articleView.jsp?id=87).

   При открытии большого количества TCP-соединений есть вероятность того, что в операционной системе будут выбраны все доступные динамические порты. По умолчанию в Windows в качестве динамических доступны порты в диапазоне 1024–5000.

   Решение

   Для устранения проблемы необходимо увеличить количество доступных динамических портов на сервере, на котором расположен кластер серверов «1С:Предприятия».

Windows 2003

В ключе реестра HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters нужно добавить следующий параметр:

- MaxUserPort = dword: 64510.

После выполнения настройки необходимо произвести перезапуск сервера.

Windows 2008 и старше

1. В ключе реестра HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters нужно добавить следующие параметры:
   - MaxUserPort = dword:65534
   - MaxFreeTcbs = dword:100000
   - TcpTimedWaitDelay = dword:30
   - StrictTimeWaitSeqCheck = dword:1
2. В ключе реестра HKLM\System\CurrentControlSet\Services\AFD\Parameters указать:
   - EnableDynamicBacklog= 1
   - MinimumDynamicBacklog= 20
   - MaximumDynamicBacklog= 20000
   - DynamicBacklogGrowthDelta= 10
3. Выполнить настройку стека TCP/IP (расширенный диапазон портов) с помощью интерпретатора командной строки (cmd):

```
netsh int ipv4 set dynamicport tcp start=1025 num=64510
netsh int ipv4 set dynamicport udp start=1025 num=64510
netsh int ipv6 set dynamicport tcp start=1025 num=64510
netsh int ipv6 set dynamicport udp start=1025 num=64510
```

Мы не рекомендуем игнорировать именно этот пункт инструкции, так как, по нашему опыту, он стал одним из основных, которые приводят к проблемам доступности и стабильности. Особенность недоступностей по этой настройке также связана с тем, что проблема не проявляется без нагрузки. То есть недоступность может возникнуть именно в пиковую нагрузку, и существенное время может быть потеряно, так как проблема не решится простым перезапуском процессов кластера, веб-серверов и т. д.

Linux

Требуется установить настройку:

```
net.ipv4.netfilter.ip_conntrack_max = 1048576
```

т. е. максимальное количество соединений для работы механизма connection tracking (используется, например, iptables).

При слишком маленьких значениях ядро начинает отвергать входящие подключения с соответствующей записью в системном логе. Для установки настройки можно выполнить:

```
sysctl -w net.ipv4.netfilter.ip_conntrack_max=1048576
```

5.3. Убедиться, что брандмауэр операционной системы настроен таким образом, что не запрещает процессам кластера взаимодействовать корректно.

Информация по клиент-серверному варианту работы приведена на ИТС в главе 2 «Клиент-серверный вариант работы». Обратите внимание на используемые порты, которые указываются в параметрах центрального сервера:

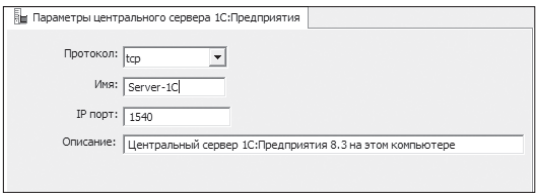


Рисунок 31. Параметры центрального сервера кластера серверов «1С:Предприятия»

В свойствах кластера серверов:

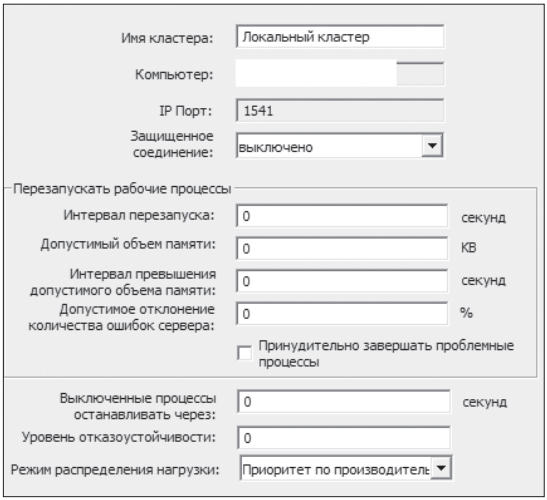


Рисунок 32. Свойства кластера серверов «1С:Предприятия»

И рабочих серверов кластера:

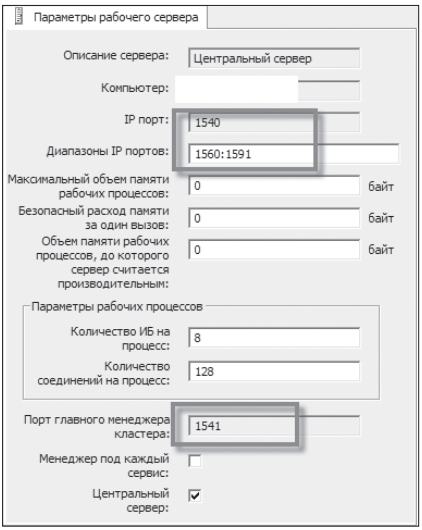


Рисунок 33. Параметры рабочего сервера

Также необходимо открыть порт для указания сетевого порта для ras – в случае использования. Инструкцию можно найти на ИТС: глава 5 «Администрирование» – «5.3.3.2. Запуск сервера администрирования».

```
ras cluster (--daemon) --port=<port> <host[:port]>
```

--port или -p указывает сетевой порт, по которому утилита администрирования будет взаимодействовать с сервером администрирования. Значение по умолчанию равно 1545.

```
(--daemon) – указывается для ОС Linux.
```

5.4. Убедиться, что на рабочих серверах кластера одновременно не используются IPv4 и IPv6 (см. рис. 34).

Возможные проблемы не будут связаны с одновременным их использованием, но будут связаны с несоответствием попыток использования протокола и неготовности сети к его поддержке. Следует указывать именно тот протокол, который совершенно точно используется в вашей организации.

Отключить использование второго протокола можно через «Панель управления» Windows или (что более правильно) через реестр, как описано здесь: https://support.microsoft.com/en-us/help/929852/how-to-disable-ipv6-or-its-components-in-windows.

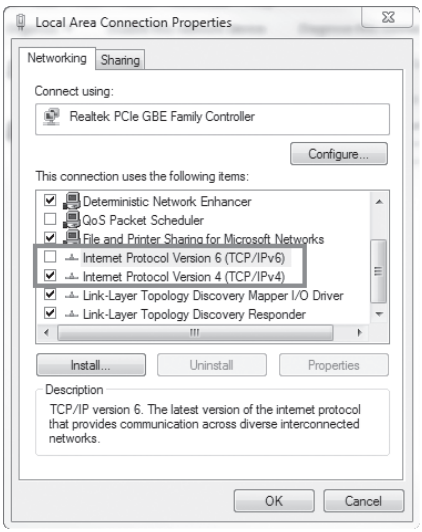


Рисунок 34. Настройка использования IPv4 и IPv6

5.5. Убедиться, что схема управления питанием – «Высокая производительность».

В зависимости от конфигурации системы следует проверить:

- Настройки на уровне операционной системы.
- Настройки на уровне среды виртуализации.
- Настройки в BIOS.

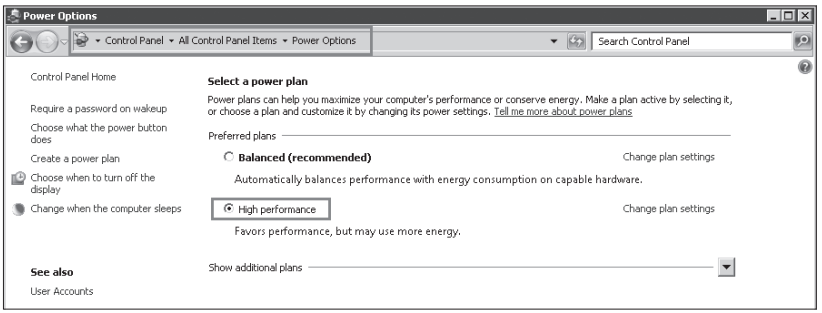


Рисунок 35. Схема электропитания

Текущая схема электропитания хранится в реестре HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Power\User\PowerSchemes в параметре ActivePowerScheme, который может принимать значения:

- 381b4222-f694-41f0-9685-ff5bb260df2e – «Сбалансированный план»;
- a1841308-3541-4fab-bc81-f71556f20b4a – «Экономия энергии»;
- 8c5e7fda-e8bf-4a96-9a85-a6e23a8c635c – «Высокая производительность»;
- Иной идентификатор – план электропитания, созданный пользователем.

Для массовой проверки параметра электропитания по большому количеству серверов можно смотреть именно на значение ключа реестра.

5.6. Убедиться, что установлены компоненты Microsoft Data Access Components.

Этот пункт нужен для настройки с СУБД MS SQL Server.

В противном случае будете получать ошибку вида: «Компоненты OLE DB провайдера не найдены». Скачать MDAC можно с официальных ресурсов компании Microsoft.

6. Настроить кластер серверов.

6.1. Необходимо добавить рабочие серверы в кластер.

Информация по работе со списком серверов кластера находится на ИТС «Клиент-серверный вариант. Руководство администратора» – глава 5 «Администрирование» – раздел «5.2.5. Работа со списком рабочих серверов кластера».

6.2. Настроить условия перезапуска.

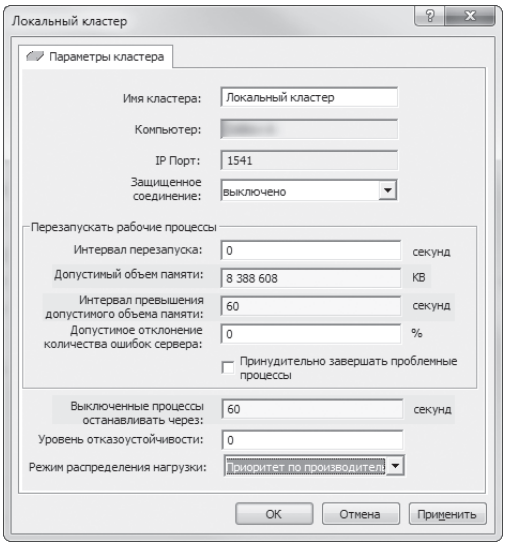


Рисунок 36. Условия перезапуска рабочих процессов по превышению памяти

Рекомендуется всегда настраивать следующие параметры:

- Допустимый объем памяти,
- Интервал превышения допустимого объема памяти,
- Выключенные процессы останавливать через/

Общее правило расчета «Допустимого объема памяти»:

*Д.О.П. = ОбщийОбъемОЗУ - ОбъемСистемы - ОбъемRPHOST * Коэффициент*,

где *Коэффициент* = 1,2,3… – кол-во рабочих процессов при штатной работе.

Допустимый объем памяти стоит устанавливать из расчета, что в случае срабатывания условия превышения показателя будет запущен еще один процесс rphost того же объема, как и при нормальной работе кластера серверов в этой информационной системе.

Например, на рабочем сервере имеем 12 Гб ОЗУ. Допустим, для конкретной информационной системы характерен размер rphost около 3 Гб. В этом случае порог превышения памяти следует рассчитывать следующим образом:

- *Допустимый объем памяти = 12 ГБ - 2 Гб (объем, занимаемый процессами системы) - 3 Гб * 1 rphost (объем всех процессов rphost) = 7 Гб.* То есть процесс rphost при худшем сценарии может вырасти до 7 Гб.

Для случая, когда при штатной работе используются два процесса rphost:

- *Допустимый объем памяти = 12 ГБ - 2 Гб (объем, занимаемый процессами системы) - 3 Гб * 2 rphost (объем всех процессов rphost) = 4 Гб.* То есть процесс rphost при худшем сценарии может вырасти до 4 Гб.

Такая рекомендация исходит из особенностей поведения в момент перезапуска процессов кластера. Это происходит следующим образом:

- процесс rphost превышает Допустимый объем памяти в течение «Интервала превышения допустимого объема памяти», срабатывает условие перезапуска процессов кластера;
- запускается «новый» процесс rphost;
- «старый» процесс rphost выключается, но не завершается;
- соединения назначаются на «новый» процесс rphost, который сразу полноценно включается в работу;
- «старый» процесс будет исполнять вызовы (которые еще существуют) еще максимум в течение времени, заданного параметром Выключенные процессы останавливать через, но не более того;
- через время, заданное параметром Выключенные процессы останавливать через, «старый» процесс rphost завершается;
- новый процесс полноценно работает.

То есть в течение периода, указанного в параметре Выключенные процессы останавливать через, будут одновременно работать как минимум два процесса rphost: «старый» и «новый».

Не следует указывать Допустимый объем памяти меньше нормального рабочего объема памяти процесса rphost для вашей системы, так как противном случае у вас постоянно будут перезапускаться процессы кластера серверов.

Не следует одновременно указывать слишком малое время в «Интервале перезапуска» и малое время в параметре Выключенные процессы останавливать через, так как это приведет к частым перезапускам процессов с минимальным ожиданием завершения исполняемых клиент-серверных вызовов. В этом случае возможно частое возникновение ситуаций, в которых момент перезапуска попадает на исполнение клиент-серверного вызова. В таких случаях штатным поведением будет возникновение на клиентских приложениях ошибки вида: «На сервере "1С:Предприятия" произошла неисправимая ошибка. Приложение будет закрыто».

Настройки Интервал превышения допустимого объема памяти, Выключенные процессы останавливать через следует стараться указывать как можно меньше, исходя из характера нагрузки на информационную систему – например, по 60 секунд, если мы рассчитываем, что все операции (или большая их часть) должны выполниться быстрее 60 секунд.

Чем больше значения указанных параметров, тем менее эффективен может оказаться механизм перезапуска процессов, но зато это позволит «успешно выполнить» большее число вызовов.

В этой же группе настроек находится параметр Принудительно завершать проблемные процессы.

Начиная с версии технологической платформы 8.3.10, рекомендуется включать этот параметр. После включения параметра кластер станет аварийно завершать проблемные процессы. Это поведение особенно важно, если произойдет «зависание» процессов кластера по какой-либо причине. «Зависание» должно быть не только успешно обнаружено, но процесс будет завершен автоматически (с образованием дампа памяти процесса), запущен новый процесс. Кластер будет самостоятельно восстанавливать работоспособность при обнаружении зависаний. Все действия относительно обнаружения зависаний и завершения процессов будут фиксироваться в технологических журналах кластера в событиях ATTN. При отсутствии этой настройки кластер будет только детектировать зависания, о чем будет сообщать в событиях ATTN, но никаких действий по самостоятельному завершению процессов предпринимать не будет.

6.3. Настроить расположение каталога кластера.

Необходимо убедиться:

- что на дисках достаточно места;
- сеансовые данные расположены на быстрых дисках.

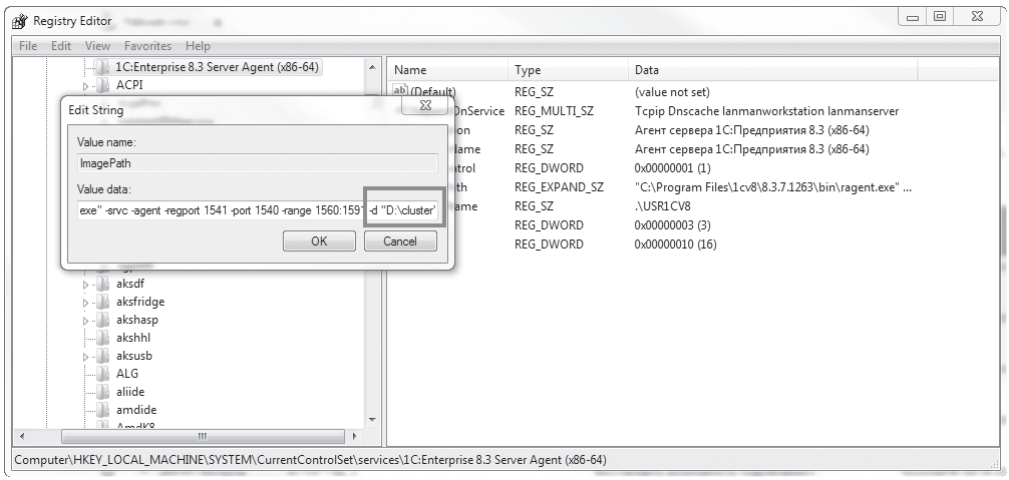


Рисунок 37. Настройка расположения каталога кластера

6.4. Настроить число соединений и информационных баз на процесс.

Настройку необходимо выполнить с учетом конфигурации системы (http://kb.1c.ru/articleView.jsp?id=89).

Постарайтесь выполнять настройку таким образом, чтобы она не приводила к запуску 100 процессов rphost, т. к. значительное число процессов rphost приводит к неэффективному использованию памяти процессами кластера.

Не стоит просто так уменьшать параметры Количество соединений на процесс, Количество ИБ на процесс. Нужно понимать, что с увеличением числа рабочих процессов кластера менее эффективно будут работать различные системы кеширования в процессах кластера.

Если у вас нет технического обоснования, почему именно так будет лучше, рекомендуем оставить значения по умолчанию.

Стоит рассчитывать, что каждый из процессов rphost может параллельно исполнять до 512 вызовов, не ограничиваясь числом установленных соединений. Естественно, что из даже 128 установленных соединений не во всех соединениях будут выполнены вызовы одновременно, и даже в течение одной минуты.

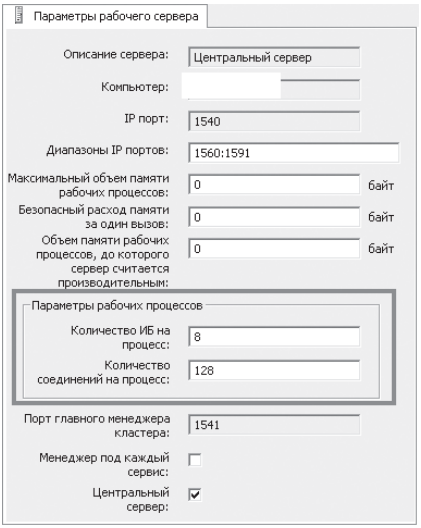


Рисунок 38. Настройка числа соединений и ИБ на процесс

6.5. Выполнить настройки для случая нескольких рабочих серверов в кластере.

Необходимо настроить **требования назначения функциональности** (подробности на ИТС: «5.2.5.4. Требования назначения функциональности», глава 5 «Администрирование»):

- Обязательно должно быть явно указано расположение:
  - сервиса журнала регистрации;
  - сервиса полнотекстового поиска данных;

- ✧ сервиса работы с внешними источниками данных.

○ Обязательно нужно продумать:

- ✧ расположение клиентских и серверных лицензий и сервисов лицензирования;
- ✧ расположение сеансовых данных.

6.6. Ограничения на расход памяти на вызов (версия КОРП).

В версии КОРП можно настроить ограничения на расход памяти в момент клиент-серверного вызова.

Самые важные параметры:

○ Максимальный объем памяти рабочих процессов;

○ Безопасный расход памяти за один вызов.

Значения по умолчанию:

○ Максимальный объем памяти рабочих процессов:

- ✧ «0» – 80 % объема оперативной памяти сервера;
- ✧ «-1» – без ограничения.

○ Безопасный расход памяти за один вызов:

- ✧ «0» – 5 % от макс. объема памяти;
- ✧ «-1» – вызов считается опасным, если за время вызова достигнут максимальный объем памяти рабочего процесса.

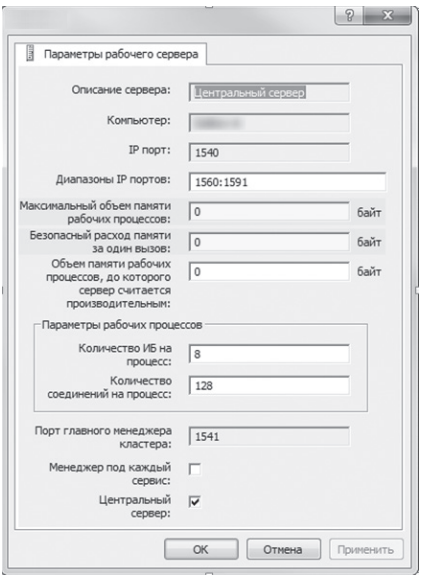


Рисунок 39. Ограничения на расход памяти на вызов

7. Первый запуск.

   На этом этапе следует выполнить следующие шаги:

   - Запустить кластер серверов, создать информационную базу.
   - Зарегистрировать программные лицензии (на ИТС: глава 9 «Защита от несанкционированного использования: особенности и настройка» – «9.3. Система программного лицензирования»).
   - Убедиться, что пользователь может войти в информационную базу без ошибок.

8. Отказоустойчивость.

   В случае необходимости настройки отказоустойчивого кластера необходимо выполнить следующие шаги.

   8.1. Проверить лицензии.

   Нужно убедиться, что на рабочих серверах, которые должны выполнять роль «Центральных серверов», достаточно лицензий для работы всех пользователей в информационной системе при отсутствии одного из «Центральных серверов» в случае возможного (теоретически) отказа.

   8.2. Установить флаг «Центральный сервер».

   Установить флаг, как на рисунке ниже.

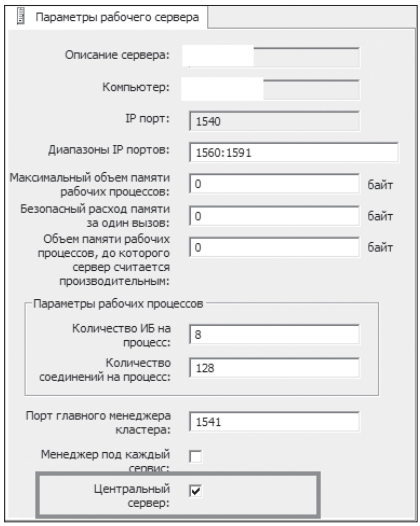


Рисунок 40. Флаг «Центральный сервер»

Если в состав кластера входят несколько центральных серверов, то реестр кластера ведется каждым из главных менеджеров кластера. Для того чтобы данные каждой копии реестра кластера были актуальными, внутри кластера серверов выполняется постоянная синхронизация реестра кластера между главными менеджерами кластера центральных серверов.

ВАЖНО!

Без необходимости не стоит устанавливать флаг Центральный сервер более чем у 1 сервера кластера. Это избыточные затраты на синхронизацию данных.

8.3. Установить «Уровень отказоустойчивости».

Установить параметр, как на рисунке ниже (по умолчанию равен 0).

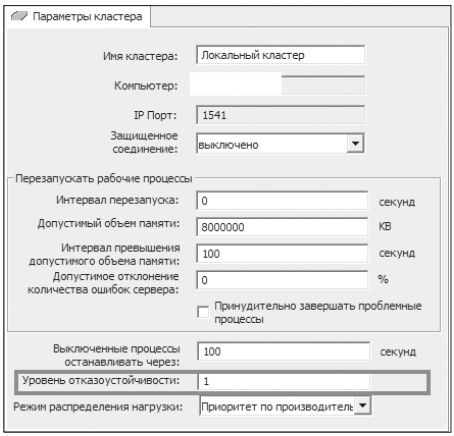


Рисунок 41. Уровень отказоустойчивости

Уровень отказоустойчивости определяет максимальное количество рабочих серверов, входящих в состав кластера, одновременный выход которых из строя не приведет к аварийному завершению сеансов подключенных пользователей. Надо понимать, что под «выходом из строя» подразумеваются ситуации, подобные следующим: отключение питания компьютера, разрыв сетевого кабеля, проблемы с операционной системой, не позволяющие запустить процесс, и т. д.

Подробно про уровень отказоустойчивости можно прочитать в документации: глава 2 «Клиент-серверный вариант работы» – «2.1.5. Отказоустойчивый кластер».

Обратите внимание, что не нужно просто так, без необходимости, указывать уровень отказоустойчивости, отличный от 0 (максимально возможный уровень отказоустойчивости), так как это приведет к избыточным накладным расходам.

8.4. Особенности конфигурирования отказоустойчивого кластера.

- Два центральных сервера, уровень отказоустойчивости 1:
  - оба сервера должны быть достаточными для эффективной работы всех пользователей требуется следить за загрузкой.
- Лицензии:
  - на каждом из серверов необходимы серверные лицензии (ничем не отличается от схемы с одним рабочим сервером);

- ✧ на каждом из серверов необходим ПОЛНЫЙ комплект клиентских лицензий (полный комплект необходим для работы всех пользователей).

○ Два центральных сервера, уровень отказоустойчивости 1 + 2 сервера лицензирования:

- ✧ Один сервер лицензирования нет смысла выделять, т. к. он становится единой точкой отказа:
  - вы решаете вынести серверные лицензии на серверы лицензирования;
  - на каждом из серверов необходим ПОЛНЫЙ комплект клиентских лицензий;
  - на каждом сервере необходим ПОЛНЫЙ комплект серверных лицензий:
    - иначе неоткуда брать лицензии в том случае, если рабочий сервер с назначенным сервисом лицензирования не будет доступен;
    - при отсутствии серверной лицензии часть пользователей может получать ошибки.

○ Порядок получения клиентских лицензий:

- ✧ локальные лицензии для клиентского приложения;
- ✧ получение лицензий с сервера;
- ✧ локальная лицензия для сервера, где расположено хранилище сеансов с сеансом;
- ✧ запрос лицензии у сервера лицензирования.

○ Получение лицензий в случае наличия программных и аппаратных лицензий:

- ✧ берется в первую очередь последняя используемая лицензия того же типа;
- ✧ историю получения лицензий можно увидеть в информации о системе.

9. Замечания.

9.1. Не следует настраивать exec backup (или аналогичные утилиты) на директории кластера серверов.

Причина в том, что в этих директориях могут располагаться хранилища с сеансовыми данными и/или данные полнотекстового поиска, выполнять бэкап которых не нужно, а его создание будет приводить к избыточным накладным расходам. Такие служебные данные будут получены в процессе работы системы при необходимости. В общем случае неправильно эти данные назвать кешами, но относиться к ним нужно аналогичным образом.

9.2. Не следует настраивать сжатие данных дисков с директорией кластера.

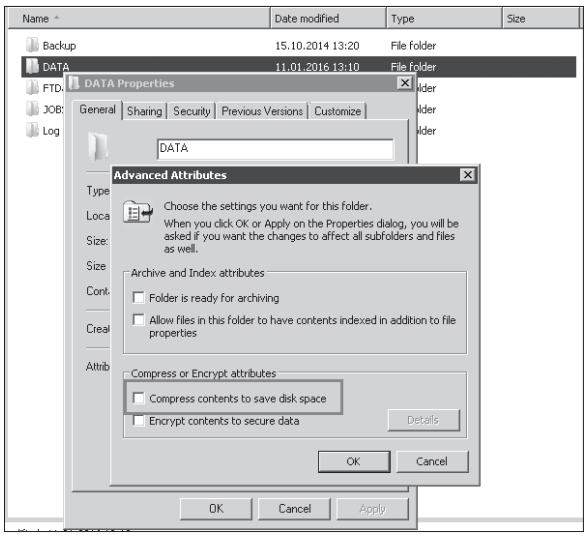


Рисунок 42. Настройка сжатия данных диска с директорией кластера

9.3. Не следует забывать про периодическое выполнение дефрагментации дисков ОС Windows.

Постарайтесь отследить, чтобы такие операции не выполнялись в период наибольшей нагрузки на систему.

9.4. Настроить защиту от вирусов.

В случае расположения рабочих серверов кластера в зоне, доступ к которой строго ограничен, не имеет смысла настраивать антивирусные решения на рабочих серверах.

Настройка антивирусных решений на таких серверах будет оказывать существенное влияние на производительность при практическом отсутствии выигрыша с точки зрения защиты.

При этом стоит обеспечить защиту антивирусными решениями тех пользовательских компьютеров, с которых выполняется доступ к рабочим серверам кластера и сетевым директориям.

Сервер администрирования кластера серверов

Информация по серверу администрирования размещена в документации по технологической платформе «1С:Предприятие», опубликованной на https://its.1c.ru/db/v8310doc#bookmark:cs:TI000000189.

Помещение в книгу данного и некоторых других материалов обусловлено двумя факторами:

- не у всех есть доступ к официальной документации, а в «альтернативных» источниках инструмент освещен на уровне легенд и слухов о том, «как оно работает»;

- у тех, кто имеет доступ к документации, в 80 % случаев встречается либо неподдельное удивление при упоминании о существовании данного инструмента, либо непонимание, «как оно на самом деле работает».

Поэтому попробуем и здесь совместить теорию с реальной жизнью и вывести данный инструмент из тени.

Общая информация

Для администрирования кластера серверов можно использовать специальный сервер администрирования кластера. В состав сервера входят собственно сервер (ras) и утилита командной строки (rac), позволяющая управлять кластером серверов. Утилита позволяет выполнить весь объем работ, необходимый для администрирования кластера серверов и может запускаться на удаленном сервере, что позволяет использовать связку rac/ras в качестве инструмента группового удаленного администрирования кластеров серверов «1С:Предприятия».

Сервер администрирования (ras) может выполняться как в режиме приложения, так и в режиме службы ОС Windows или демона ОС Linux. Общая схема работы выглядит следующим образом:

- Запускается сервер администрирования (как приложение или служба/демон).
- Утилита командной строки соединяется с сервером администрирования для выполнения необходимых действий.
- На время выполнения операций сервер администрирования выполняет подключение к кластеру серверов и после выполнения операций отключается от кластера.

Сервер администрирования и утилита администрирования входят в состав компонент сервера «1С:Предприятия» при установке системы и располагаются по умолчанию в каталоге установки платформы, в подпапке bin.

Для взаимодействия сервера администрирования и утилиты администрирования используется сетевой порт 1545, который может быть переопределен с помощью параметра --port командной строки запуска сервера администрирования (ras).

Результат работы утилиты представляет собой описание одного или нескольких объектов данных (например, перечень зарегистрированных в кластере серверов информационных баз) и представлен в виде таблицы:

```
<Имя параметра> : <Значение параметра>
```

Более подробную информацию о параметрах сервера администрирования (ras) или утилиты администрирования (rac) можно получить в командной строке, запустив соответствующий исполняемый файл с параметром help:

```
ras help
rac help
```

На диске ИТС также поставляется пакет Java-архивов, который позволяет взаимодействовать с сервером администрирования из программы на языке Java, без помощи консольной утилиты администрирования (http://its.1c.ru/db/metod8dev#content:4985:hdoc).

Запуск сервера администрирования (ras) для ОС Windows

В режиме приложения

Запуск сервера администрирования в режиме приложения выполняется с помощью командной строки следующего вида:

```
ras cluster --port=<port> <host[:port]>
```

В команде запуска могут использоваться следующие ключи:

cluster

Запуск сервера администрирования в режиме администрирования кластера серверов.

--port или -p

Указывает сетевой порт, по которому утилита администрирования будет взаимодействовать с сервером администрирования. Значение по умолчанию равно 1545.

<host[:port]>

Указывается адрес агента сервиса того кластера серверов, для администрирования которого запускается сервер администрирования.

Если адрес агента кластера не задан явным образом, то по умолчанию используется адрес localhost:1540.

В режиме сервиса

Для запуска сервера администрирования в режиме сервиса необходимо зарегистрировать сервер администрирования в качестве службы. Данная операция может быть выполнена с помощью утилиты sc. Для выполнения регистрации необходимы права администратора.

В качестве примера рассмотрим командный файл, выполняющий регистрацию службы сервера.

Файл register-ras.bat:

```
@echo off
rem %1  – полный номер версии «1С:Предприятия»
set SrvUserName=<имя пользователя>
set SrvUserPwd=<пароль пользователя>
set CtrlPort=1540
set AgentName=localhost
set RASPort=1545
set SrvcName="1C:Enterprise 8.3 Remote Server"
set BinPath="\"C:\Program Files\1cv8\%1\bin\ras.exe\" cluster --service --port=%RASPort% %AgentName%:%CtrlPort%"
```

```
set Desctiption="Сервер администрирования 1С:Предприятия 8.3"
sc stop %SrvcName%
sc delete %SrvcName%
sc create %SrvcName% binPath= %BinPath% start= auto obj= %SrvUserName% password=
                                                    %SrvUserPwd% displayname= %Desctiption%
```

Перед применением данного командного файла необходимо указать в нем данные реального пользователя (имя и пароль), от имени которого будет работать служба сервера администрирования (строки set SrvUserName= и set SrvUserPwd=). Данный командный файл выполняет регистрацию сервера администрирования со следующими параметрами:

- Имя службы: 1C:Enterprise 8.3 Remote Server.
- Отображаемое имя: Сервер администрирования 1С:Предприятия 8.3.
- Порт сервера администрирования: 1545.
- Адрес кластера серверов «1С:Предприятия»: localhost:1540.
- Режим запуска службы: Автоматический.

Пример использования:

```
register-ras 8.3.3.100
```

Запуск сервера администрирования (ras) для ОС Linux

В режиме приложения

Запуск сервера администрирования в режиме приложения выполняется с помощью командной строки следующего вида:

```
./ras cluster --port=<port> <host[:port]>
```

В команде запуска могут использоваться следующие ключи:

cluster

Запуск сервера администрирования в режиме администрирования кластера серверов.

--port или -p

Указывает сетевой порт, по которому утилита администрирования будет взаимодействовать с сервером администрирования. Значение по умолчанию равно 1545.

<host[:port]>

Указывается адрес агента сервиса того кластера серверов, для администрирования которого запускается сервер администрирования.

Если адрес агента кластера не задан явным образом, то по умолчанию используется адрес localhost:1540. Но рекомендуется всегда указывать имя (hostname) компьютера вместо localhost.

В режиме демона

Для запуска сервера администрирования (ras) в режиме демона необходимо запустить сервер администрирования с использованием специального ключа командной строки:

```
./ras cluster --daemon --port=<port> <host[:port]>
```

Ключи командной строки запуска сервера администрирования (ras) в ОС Windows и ОС Linux идентичны.

Утилита администрирования платформы «1С:Предприятие» (rac)

Использование:

```
rac [mode] [command] [options] [arguments]
```

Поддерживаемые режимы (mode):

help - отображение справочной информации для указанного режима;

agent - режим администрирования агента кластера серверов;

cluster - режим администрирования кластера серверов;

manager - режим администрирования менеджера кластера серверов;

server - режим администрирования рабочего сервера;

process - режим администрирования рабочего процесса;

service - режим администрирования сервиса менеджера кластера;

infobase - режим администрирования информационной базы;

connection - режим администрирования соединений;

session - режим администрирования сеансов информационных баз;

lock - режим администрирования блокировок;

rule - режим управления требованиями назначения;

profile - режим управления профилями безопасности кластера.

Общие параметры:

--version | -v

– получение версии утилиты;

--help | -? | -h

– отображение краткой информации об утилите.

Общие аргументы:

<host>[:<port>]

– адрес сервера администрирования (по умолчанию: localhost:1545).

Режим agent

Использование:

```
rac agent [command] [options] [arguments]
```

– режим администрирования агента кластера серверов.

Общие параметры:

--version | -v

– получение версии утилиты;

--help | -? | -h

– отображение краткой информации об утилите.

Общие аргументы:

<host>[:<port>]

– адрес сервера администрирования (по умолчанию: localhost:1545).

Параметры:

--agent-user=<name>

– имя администратора агента кластера;

--agent-pwd=<pwd>

– пароль администратора агента кластера.

Команды:

admin

– управление администраторами агента кластера.

Дополнительные команды:

list

– получение списка администраторов агента кластера;

register

– добавление нового администратора агента кластера:

--name=<name>

– обязательный, имя администратора;

--pwd=<name>

– пароль администратора в случае аутентификации паролем;

--descr=<descr>

– описание администратора;

--auth=pwd[,os]

– доступные способы аутентификации:

- pwd – при помощи имени пользователя и пароля;
- os – аутентификация средствами ОС;

--os-user=<name>

– имя пользователя операционной системы;

remove

– удаление администратора агента кластера:

--name=<name>

– обязательный, имя администратора агента кластера.

Режим cluster

Использование:

```
rac cluster [command] [options] [arguments]
```

– режим администрирования кластера серверов.

Команды:

admin

– управление администраторами кластера.

Дополнительные команды:

list

– получение списка администраторов кластера;

register

– добавление нового администратора кластера.

Параметры:

--name=<name>

– обязательный, имя администратора;

--pwd=<name>

– пароль администратора в случае аутентификации паролем (другой вариант – аутентификация ОС) ;

--descr=<descr>

– описание администратора;

--auth=pwd[,os]

– доступные способы аутентификации:

- pwd – при помощи имени пользователя и пароля;
- os – аутентификация средствами ОС;

--os-user=<name>

– имя пользователя операционной системы;

--agent-user=<name>

– имя администратора агента кластера;

--agent-pwd=<pwd>

– пароль администратора агента кластера;

remove

– удаление администратора кластера:

--name=<name>

– обязательный, имя администратора кластера

--cluster=<uuid>

– обязательный, идентификатор кластера серверов;

--cluster-user=<name>

– имя администратора кластера;

--cluster-pwd=<pwd>

– пароль администратора кластера;

info

– получение информации о кластере:

--cluster=<uuid>

– обязательный, идентификатор кластера серверов;

list

– получение списка информации о кластерах;

insert

– регистрация нового кластера:

--host=<host>

– обязательный, имя (или IP-адрес) компьютера, на котором расположен реестр кластера и процесс главного менеджера кластера;

--port=<port>

– обязательный, основной порт основного менеджера;

--name=<name>

– имя (представление) кластера;

--expiration-timeout=<seconds>

– период принудительного завершения (в секундах);

--lifetime-limit=<seconds>

– период перезапуска рабочих процессов кластера (в секундах);

--max-memory-size=<Kb>

– максимальный объем виртуального адресного пространства (в килобайтах), занятого рабочим процессом;

--max-memory-time-limit=<seconds>

– максимальный период превышения критического объема памяти (в секундах);

--security-level=<level>

– уровень безопасности соединений;

--session-fault-tolerance-level=<level>

– уровень отказоустойчивости;

--load-balancing-mode=performance|memory

– режим распределения нагрузки:

- performance – с приоритетом по доступной производительности;
- memory – с приоритетом по доступной оперативной памяти;

--errors-count-threshold=<percentage>

– допустимое отклонение количества ошибок сервера (в процентах);

--kill-problem-processes=<yes/no>

– принудительно завершать проблемные процессы;

--agent-user=<name>

– имя администратора агента кластера;

--agent-pwd=<pwd>

– пароль администратора агента кластера;

update

– обновление параметров кластера:

--cluster=<uuid>

– обязательный, идентификатор кластера серверов;

--name=<name>

– имя (представление) кластера;

--expiration-timeout=<seconds>

– период принудительного завершения (в секундах);

--lifetime-limit=<seconds>

– период перезапуска рабочих процессов кластера (в секундах);

--max-memory-size=<Kb>

– максимальный объем виртуального адресного пространства (в килобайтах), занятого рабочим процессом;

--max-memory-time-limit=<seconds>

– максимальный период превышения критического объема памяти (в секундах);

--security-level=<level>

– уровень безопасности соединений;

--session-fault-tolerance-level=<level>

– уровень отказоустойчивости;

--load-balancing-mode=performance|memory

– режим распределения нагрузки:

- performance – с приоритетом по доступной производительности;
- memory – с приоритетом по доступной оперативной памяти;

--errors-count-threshold=<percentage>

– допустимое отклонение количества ошибок сервера (в процентах);

--kill-problem-processes=<yes/no>

– принудительно завершать проблемные процессы;

--agent-user=<name>

– имя администратора агента кластера;

--agent-pwd=<pwd>

– пароль администратора агента кластера;

remove

– удаление кластера:

--cluster=<uuid>

– обязательный, идентификатор кластера серверов;

--cluster-user=<name>

– имя администратора кластера;

--cluster-pwd=<pwd>

– пароль администратора кластера.

Режим server

Использование:

```
rac server [command] [options] [arguments]
```

– режим администрирования рабочего сервера.

Параметры:

--cluster=<uuid>

– обязательный, идентификатор кластера серверов;

--cluster-user=<name>

– имя администратора кластера;

--cluster-pwd=<pwd>

– пароль администратора кластера.

Команды:

info

– получение информации о рабочем сервере:

--server=<uuid>

– обязательный, идентификатор рабочего сервера кластера серверов;

list

– получение списка информации о рабочих серверах;

insert

– регистрация рабочего сервера:

--agent-host=<host>

– обязательный, имя хоста или IP-адрес агента сервера;

--agent-port=<port>

– обязательный, основной порт агента сервера;

--port-range=<min>:<max>

– обязательный, диапазон IP-портов для динамического распределения, возможно указание нескольких диапазонов;

--name=<name>

– наименование рабочего сервера;

--using=main|normal

– вариант использования рабочего сервера:

- main – в качестве центрального сервера;
- normal – в качестве обычного сервера;

--infobases-limit=<count>

– максимальное количество информационных баз на рабочий процесс;

--memory-limit=<Kb>

– предел использования памяти рабочими процессами;

--connections-limit=<count>

– максимальное количество соединений на рабочий процесс;

--cluster-port=<port>

– номер порта главного менеджера кластера;

--dedicate-managers=all|none

– вариант размещения менеджеров сервисов:

- all – размещать все сервисы в отдельных менеджерах;
- none – размещать все сервисы в одном менеджере;

--safe-working-processes-memory-limit=<bytes>

– максимальный объем памяти рабочих процессов на сервере;

--safe-call-memory-limit=<bytes>

– безопасный расход памяти за один вызов (в байтах);

update

– изменение параметров рабочего сервера:

--server=<uuid>

– обязательный, идентификатор рабочего сервера кластера серверов;

--port-range=<min>:<max>

– диапазон IP-портов для динамического распределения, возможно указание нескольких диапазонов;

--using=main|normal

– вариант использования рабочего сервера:

- main – в качестве центрального сервера;
- normal – в качестве обычного сервера;

--infobases-limit=<count>

– максимальное количество информационных баз на рабочий процесс;

--memory-limit=<Kb>

– предел использования памяти рабочими процессами;

--connections-limit=<count>

– максимальное количество соединений на рабочий процесс;

--dedicate-managers=all|none

– вариант размещения менеджеров сервисов:

- all – размещать все сервисы в отдельных менеджерах;
- none – размещать все сервисы в одном менеджере;

--safe-working-processes-memory-limit=<bytes>

– максимальный объем памяти рабочих процессов на сервере;

--safe-call-memory-limit=<bytes>

– безопасный расход памяти за один вызов (в байтах);

remove

– удаление рабочего сервера:

--server=<uuid>

– обязательный, идентификатор рабочего сервера кластера серверов.

Режим infobase

Использование:

```
rac infobase [command] [options] [arguments]
```

– режим администрирования информационной базы.

Параметры:

--cluster=<uuid>

– обязательный, идентификатор кластера серверов;

--cluster-user=<name>

– имя администратора кластера;

--cluster-pwd=<pwd>

– пароль администратора кластера.

Команды:

info

– получение информации об информационной базе:

--infobase=<uuid>

– обязательный, идентификатор информационной базы;

--infobase-user=<name>

– имя администратора информационной базы;

--infobase-pwd=<pwd>

– пароль администратора информационной базы;

summary

– управление краткой информацией об информационных базах.

Дополнительные команды:

info

– получение краткой информации об указанной информационной базе:

--infobase=<uuid>

– обязательный, идентификатор информационной базы;

list

– получение списка краткой информации об информационных базах;

update

– обновление краткой информации об информационной базе:

--infobase=<uuid>

– обязательный, идентификатор информационной базы;

--descr=<descr>

– описание информационной базы;

create

– создание информационной базы;

--create-database

– при создании информационной базы создать базу данных;

--name=<name>

– обязательный, имя информационной базы;

--dbms=MSSQLServer|PostgreSQL|IBMDB2|OracleDatabase

– обязательный, тип СУБД, в которой размещается информационная база:

- MSSQLServer – MS SQL Server;
- PostgreSQL – PostgreSQL;
- IBMDB2 – IBM DB2;
- OracleDatabase – Oracle Database;

--db-server=<host>

– обязательный, имя сервера баз данных;

--db-name=<name>

– обязательный, имя базы данных;

--locale=<locale>

– обязательный, идентификатор национальных настроек информационной базы;

--db-user=<name>

– имя администратора базы данных;

--db-pwd=<pwd>

– пароль администратора базы данных;

--descr=<descr>

– описание информационной базы;

--date-offset=<offset>

– смещение дат в информационной базе;

--security-level=<level>

– уровень безопасности установки соединений с информационной базой;

--scheduled-jobs-deny=on|off

– управление блокировкой выполнения регламентных заданий:

- on – выполнение регламентных заданий запрещено;
- off – выполнение регламентных заданий разрешено;

--license-distribution=deny|allow

– управление выдачей лицензий сервером «1С:Предприятия»:

- deny – выдача лицензий запрещена;
- allow – выдача лицензий разрешена;

update

– обновление информации об информационной базе:

--infobase=<uuid>

– обязательный, идентификатор информационной базы;

--infobase-user=<name>

– имя администратора информационной базы;

--infobase-pwd=<pwd>

– пароль администратора информационной базы;

--dbms=MSSQLServer|PostgreSQL|IBMDB2|OracleDatabase

– тип СУБД, в которой размещается информационная база:

- MSSQLServer – MS SQL Server;
- PostgreSQL – PostgreSQL;
- IBMDB2 – IBM DB2;
- OracleDatabase – Oracle Database;

--db-server=<host>

– имя сервера баз данных;

--db-name=<name>

– имя базы данных;

--db-user=<name>

– имя администратора базы данных;

--db-pwd=<pwd>

– пароль администратора базы данных;

--descr=<descr>

– описание информационной базы;

--denied-from=<date>

– начало интервала времени, в течение которого действует режим блокировки сеансов;

--denied-message=<msg>

– сообщение, выдаваемое при попытке нарушения блокировки сеансов;

--denied-parameter=<string>

– параметр блокировки сеансов;

--denied-to=<date>

– конец интервала времени, в течение которого действует режим блокировки сеансов;

--permission-code=<string>

– код разрешения, разрешающий начало сеанса вопреки действующей блокировке сеансов;

--sessions-deny=on|off

– управление режимом блокировки сеансов:

- on – режим блокировки начала сеансов включен;
- off – режим блокировки начала сеансов выключен;

--scheduled-jobs-deny=on|off

– управление блокировкой выполнения регламентных заданий:

- on – выполнение регламентных заданий запрещено;
- off – выполнение регламентных заданий разрешено;

--license-distribution=deny|allow

– управление выдачей лицензий сервером «1С:Предприятия»:

- deny – выдача лицензий запрещена;
- allow – выдача лицензий разрешена;

--external-session-manager-connection-string=<connect-string>

– параметры внешнего управления сеансами;

--external-session-manager-required=yes|no

– обязательное использование внешнего управления сеансами:

- yes – использование внешнего управления сеансами обязательно;
- no – использование внешнего управления сеансами не обязательно;

--security-profile-name=<name>

– профиль безопасности информационной базы;

--safe-mode-security-profile-name=<name>

– профиль безопасности внешнего кода;

drop

– режим удаления информационной базы:

--infobase=<uuid>

– обязательный, идентификатор информационной базы;

--infobase-user=<name>

– имя администратора информационной базы;

--infobase-pwd=<pwd>

– пароль администратора информационной базы;

--drop-database

– при удалении информационной базы удалить базу данных;

--clear-database

– при удалении информационной базы очистить базу данных.

Режим session

Использование:

```
rac session [command] [options] [arguments]
```

– режим администрирования сеансов информационных баз.

Параметры:

--cluster=<uuid>

– обязательный, идентификатор кластера серверов;

--cluster-user=<name>

– имя администратора кластера;

--cluster-pwd=<pwd>

– пароль администратора кластера.

Команды:

info

– получение информации о сеансе:

--session=<uuid>

– обязательный, идентификатор сеанса информационной базы;

--licenses

– вывод информации о лицензиях, полученных сеансом;

list

– получение списка информации о сеансах:

--infobase=<uuid>

– идентификатор информационной базы;

--licenses

– вывод информации о лицензиях, полученных сеансом;

terminate

– принудительное завершение сеанса:

--session=<uuid>

– обязательный, идентификатор сеанса информационной базы.

Примеры использования

А теперь от «абстрактных» имен команд и ключей перейдем к конструированию из них команд реальных, применяемых на практике для администрирования кластеров серверов «1С:Предприятия» на крупных внедрениях.

Отметим, что для rac/ras действуют все те же правила, что и для любых других утилит, запускаемых из командной строки. То есть во всех примерах выше предполагается, что команда выполняется из того же каталога, где лежат ras и rac. В противном случае указывается полный путь. Например, запуск сервера администрирования кластера (ras) в таком случае будет выглядеть так:

```
"C:\Program Files\1cv8\8.3.10.2561\bin\ras.exe" cluster --port=1545 myCluster:2540
```

Мы запустили север администрирования кластера на порту 1545 для кластера myCluster:2540.

Теперь выполним простую манипуляцию – получим список информации о кластерах. Ниже приведена команда и ее вывод:

```
"C:\Program Files\1cv8\8.3.10.2561\bin\rac.exe" myCluster:1545 cluster list
=============================================================
cluster                    : eac6cd72-1873-495b-84d3-e2552c80befb
host                       : myCluster
port                       : 2541
name                       : "Локальный кластер"
expiration-timeout         : 0
lifetime-limit             : 0
max-memory-size            : 0
max-memory-time-limit      : 0
security-level             : 0
session-fault-tolerance-level : 1
load-balancing-mode        : performance
errors-count-threshold     : 0
kill-problem-processes     : 0
```

Заметьте, что здесь мы уже работаем, используя запущенный ранее сервер администрирования – через порт 1545.

Давайте получим список серверов в кластере, используя идентификатор кластера. Этот идентификатор мы получили на прошлом шаге через rac, можно его также посмотреть в каталоге кластера в файле C:\Program Files\1cv8\srvinfo\1cv8wsrv.lst (расположение каталога кластера может отличаться от значения по умолчанию).

```
"C:\Program Files\1cv8\8.3.10.2561\bin\rac.exe" myCluster:1545 server --cluster="eac6cd72-1873-495b-84d3-e2552c80befb" list
=============================================================
server                     : 981bd418-7ea6-43ed-ba40-fabc2cd4adf2
agent-host                 : s01
agent-port                 : 2540
port-range                 : 2560:2591
name                       : "Центральный сервер"
using                      : main
dedicate-managers          : none
infobases-limit            : 8
memory-limit               : 0
connections-limit          : 128
safe-working-processes-memory-limit : 0
safe-call-memory-limit     : 0
cluster-port               : 2541

server                     : be5224b4-952f-4fda-9bf6-65b918cd83f3
agent-host                 : s02
agent-port                 : 7540
port-range                 : 7560:7591
name                       : "s02"
using                      : main
dedicate-managers          : none
infobases-limit            : 8
memory-limit               : 0
connections-limit          : 128
safe-working-processes-memory-limit : 0
safe-call-memory-limit     : 0
cluster-port               : 2541
```

На следующем примере поработаем со списком сеансов. Для начала получим список:

```
"C:\Program Files\1cv8\8.3.10.2561\bin\rac.exe" myCluster:1545 server session --cluster="eac6cd72-1873-495b-84d3-e2552c80befb" list
=============================================================
session                          : 0834823c-9f21-4fbe-b7b4-d7231611df05
session-id                       : 1
infobase                         : 5ef40807-a2cb-4f2b-bc85-fb2f4b9e54b3
connection                       : 00000000-0000-0000-0000-000000000000
process                          : 00000000-0000-0000-0000-000000000000
user-name                        : DefUser
host                             :
app-id                           : 1CV8C
locale                           : ru
started-at                       : 2017-08-09T15:08:34
last-active-at                   : 2017-08-09T15:08:45
hibernate                        : no
passive-session-hibernate-time   : 1200
hibernate-session-terminate-time : 86400
blocked-by-dbms                  : 0
blocked-by-ls                    : 0
bytes-all                        : 21789
bytes-last-5min                  : 21789
calls-all                        : 31
calls-last-5min                  : 31
dbms-bytes-all                   : 273119
dbms-bytes-last-5min             : 273119
db-proc-info                     :
db-proc-took                     : 0
db-proc-took-at                  :
duration-all                     : 5973
duration-all-dbms                : 1950
duration-current                 : 0
duration-current-dbms            : 0
duration-last-5min               : 5973
duration-last-5min-dbms          : 1950

session                          : 5e897d2c-c377-4f8a-9c22-4b3addf37433
session-id                       : 1
infobase                         : f367b666-0ffa-4ee7-abe3-41dc6e6fa2af
connection                       : bd458135-5296-4004-be05-6dd9b3bbf4e2
process                          : e7f0058b-0abd-4ee8-a06d-070ab620866d
user-name                        : DefUser
host                             : HOST_NAME
app-id                           : BackgroundJob
locale                           : ru_RU
started-at                       : 2017-08-09T15:08:56
last-active-at                   : 2017-08-09T15:08:56
hibernate                        : no
passive-session-hibernate-time   : 0
hibernate-session-terminate-time : 0
blocked-by-dbms                  : 0
blocked-by-ls                    : 0
bytes-all                        : 0
bytes-last-5min                  : 0
calls-all                        : 0
calls-last-5min                  : 0
dbms-bytes-all                   : 131179
dbms-bytes-last-5min             : 131179
db-proc-info                     : "69"
db-proc-took                     : 0
db-proc-took-at                  : 2017-08-09T15:08:57
```

```
duration-all                    : 0
duration-all-dbms               : 531
duration-current                : 12558
duration-current-dbms           : 0
duration-last-5min              : 0
duration-last-5min-dbms         : 531

session                         : be7e50bf-45bb-456d-ab73-ffb5d158405f
session-id                      : 2
infobase                        : 5ef40807-a2cb-4f2b-bc85-fb2f4b9e54b3
connection                      : 00000000-0000-0000-0000-000000000000
process                         : 00000000-0000-0000-0000-000000000000
user-name                       : DefUser
host                            :
app-id                          : 1CV8C
locale                          : ru
started-at                      : 2017-08-09T15:08:52
last-active-at                  : 2017-08-09T15:08:52
hibernate                       : no
passive-session-hibernate-time  : 1200
hibernate-session-terminate-time : 86400
blocked-by-dbms                 : 0
blocked-by-ls                   : 0
bytes-all                       : 5772
bytes-last-5min                 : 5772
calls-all                       : 6
calls-last-5min                 : 6
dbms-bytes-all                  : 15241
dbms-bytes-last-5min            : 15241
db-proc-info                    :
db-proc-took                    : 0
db-proc-took-at                 :
duration-all                    : 438
duration-all-dbms               : 172
duration-current                : 0
duration-current-dbms           : 0
duration-last-5min              : 438
duration-last-5min-dbms         : 172
```

Теперь выполним любимое действие администратора – завершим сеанс принудительно:

```
"C:\Program Files\1cv8\8.3.10.2561\bin\rac.exe" myCluster:1545 session --cluster="eac6cd72-1873-495b-84d3-e2552c80befb"
terminate --session="be7e50bf-45bb-456d-ab73-ffb5d158405f"
```

Вывод в случае успеха у команды пустой, но «красное окно» с текстом «Сеанс отсутствует или удален» в клиентском приложении демонстрирует, что все у нас получилось.

Надеемся, что приведенные простые примеры использования сервера администрирования кластера и утилиты администрирования помогли вам увидеть, насколько прост и эффективен данный инструмент. Особенно он незаменим для удаленного мониторинга и администрирования кластеров серверов для большого распределенного контура. Используя развернутые на каждом кластере серверы администрирования (ras), администратор при помощи исполняемых файлов и скриптов может проводить с ними удаленные манипуляции (во многих операциях ЦКК в том числе работает через сервер администрирования кластера).

Оценка состояния сервера при использовании ОС Linux

Как быстро понять, загружен ли сервер и чем именно

При эксплуатации реальной информационной системы могут возникнуть ситуации, в которых вам придется зайти на реальный рабочий сервер. Скорее всего вы зайдете на сервер не из любопытства, а так как что-то пошло не так. И как понять, в какую сторону смотреть, чтобы сэкономить время?

Рекомендуем рассмотреть использование следующих утилит в той последовательности, которая приведена в книге. Естественно, рекомендуем потренироваться на машине «в стороне» до реальных применений на промышленной системе.

- uptime
  - По сути, вам нужен для того, чтобы увидеть load average.
- dmesg | tail
  - Нам нужно понять, были ли ошибки ядра.
- vmstat 1
  - Можно увидеть общую статистику во времени каждую секунду, посмотреть по-крупному, что происходит.
- mpstat -P ALL 1
  - Попробовать увидеть загрузку CPU по ядрам.
- pidstat 1
  - Узнаем использование процессами ресурсов.
- iostat -xz 1
  - Сможем увидеть disk I/O.
- free -m
  - Сможем увидеть использование памяти.
- sar -n DEV 1
  - Сможем увидеть network I/O.
- sar -n TCP,ETCP 1
  - Увидим статистику по TCP.
- top
  - По-крупному увидим, какие процессы какой объем ресурсов потребляют.

Не стоит считать, что выше указан полный список команд и приложений. Их много больше. Применение указанных команд занимает несколько десятков секунд и может использоваться именно как первая оценка, ответ на вопрос: «Куда смотреть?»

По большому счету большая часть указанных выше приложений пересекается по данным с приложениями:

- atop;
- htop.

К тому же atop позволяет записывать данные в файлы и просматривать их за предыдущие периоды.

Рекомендуется все из перечисленных в этом разделе приложений установить на промышленные серверы на случае необходимости проведения расследований.

ГЛАВА 5

Администрирование Microsoft SQL Server при работе с «1С:Предприятием»

Установка Microsoft SQL Server

1. Необходимо выбрать редакцию Microsoft SQL Server исходя из потребностей и параметров «железа» (https://www.microsoft.com/ru-ru/server-cloud/products/sql-server-editions/overview.aspx).
2. Выбрать необходимые компоненты (см. рис. 43).

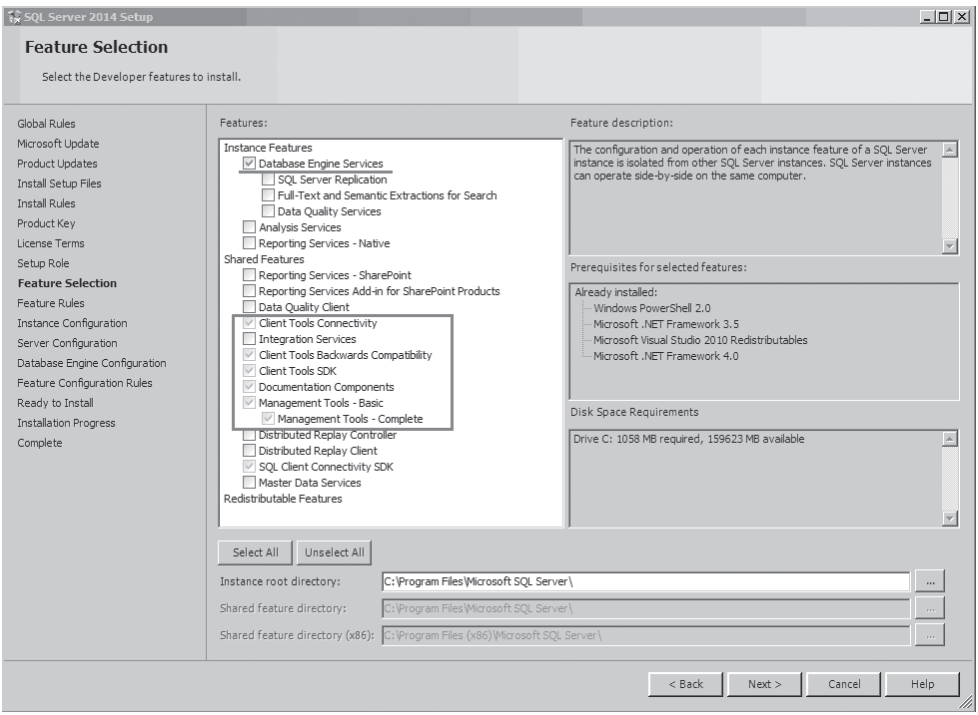


Рисунок 43. Выбор компонентов для установки MS SQL Server

3. Задать имя «экземпляра» или установить «экземпляр» по умолчанию (предпочтительно):

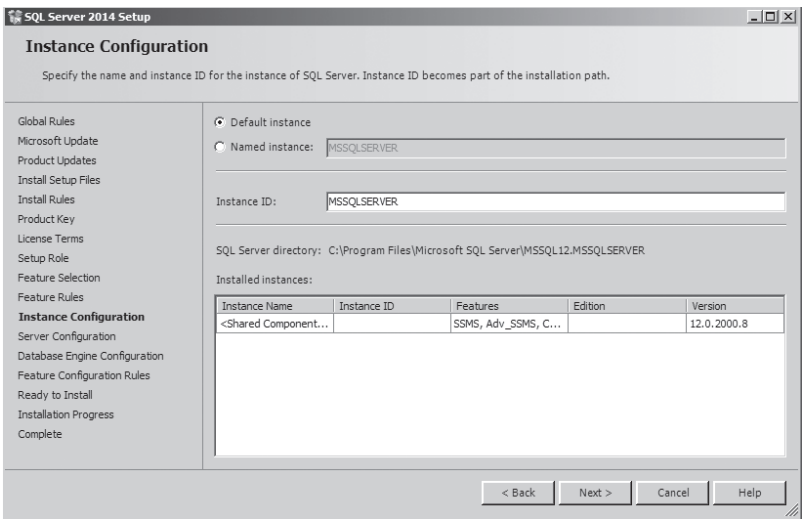


Рисунок 44. Имя «экземпляра»

4. Указать пользователей, от имени которых будут запускаться службы сервера БД:

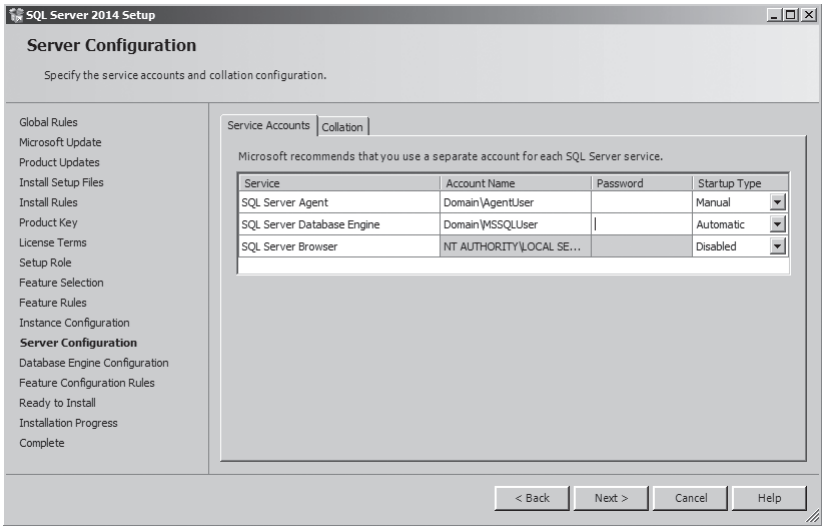


Рисунок 45. Пользователи служб

5. Параметр Collation установить равным Cyrillic_General_CI_AS:

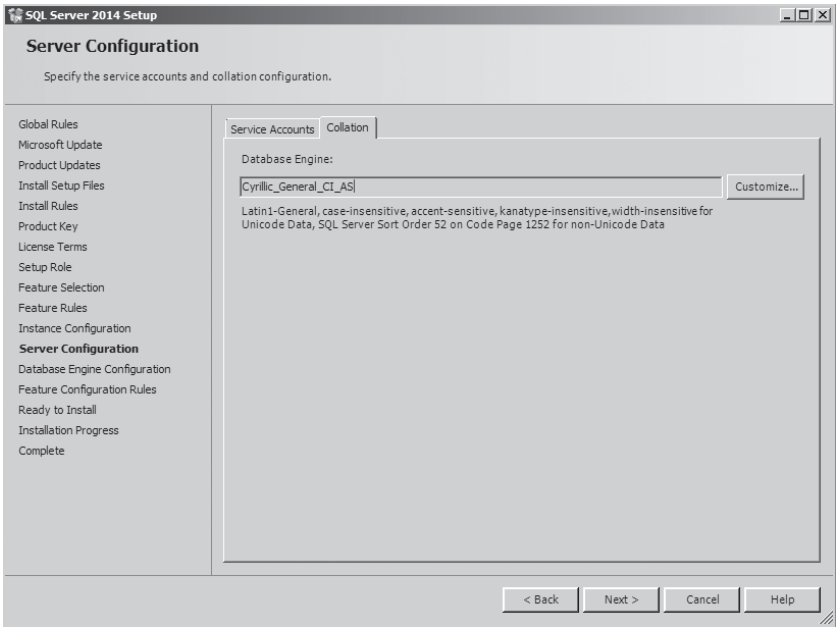


Рисунок 46. Установка параметра Collation

6. Установить смешанный режим аутентификации:

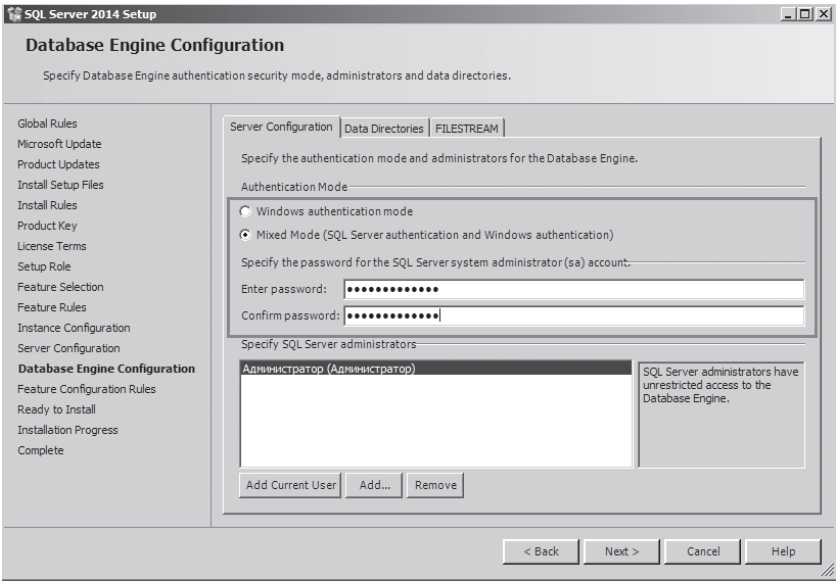


Рисунок 47. Смешанный режим аутентификации

7. Указать каталоги баз по умолчанию:

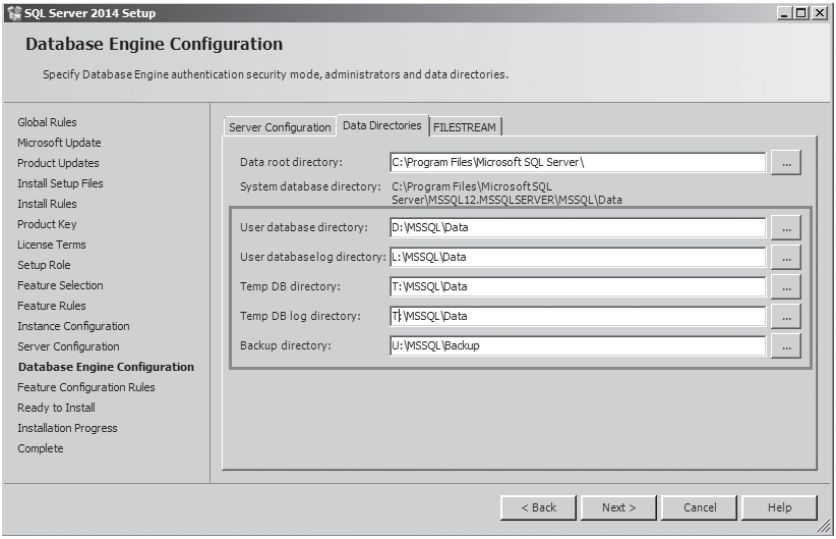


Рисунок 48. Каталоги баз по умолчанию

Подготовка дисков

Выровнять сектора дисков по границе 1024 Кб и отформатировать с размером блока 64 Кб (если нет иных рекомендаций от производителя).

Подробности: https://technet.microsoft.com/en-us/library/dd758814.aspx.

Пользователи служб Microsoft SQL Server

Если необходим доступ по сети, то нужно указывать пользователя, которому разрешен доступ к сетевым ресурсам.

Требование по смене пароля пользователя лучше убрать.

Операционная система

Включить возможность Database instant file initialization

Включить для пользователя, от имени которого запущена служба Microsoft SQL Server: https://msdn.microsoft.com/en-us/library/ms175935.aspx.

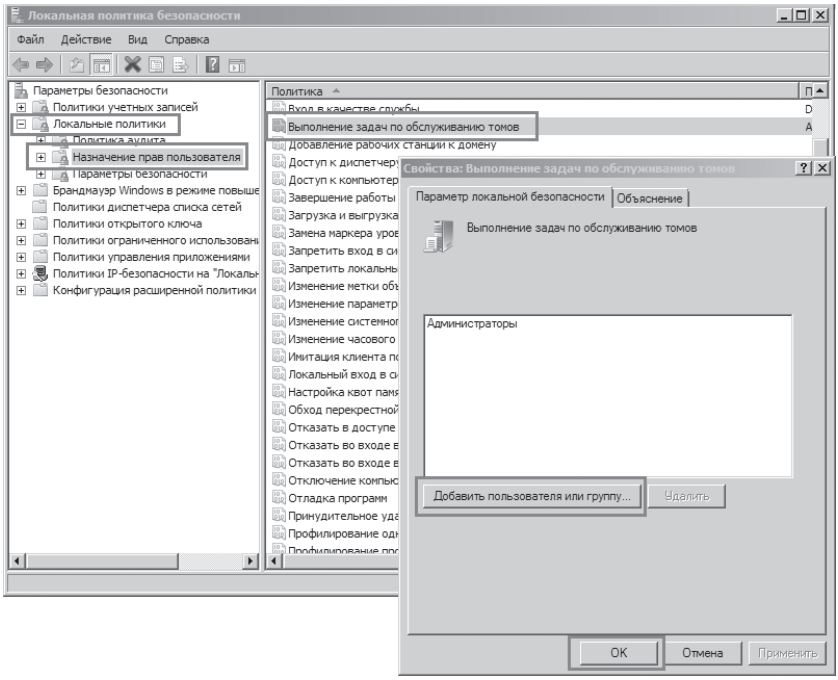


Рисунок 49. Включение возможности Database instant file initialization

Проверить работу Database instant file initialization

Создать новую базу с размером файла данных 5 Гб, журнал транзакций – 1 Мб. Если база создалась моментально, то все работает корректно. Созданную базу удалить.

Установить разрешение на Lock pages in memory

Блокировка страниц в памяти для пользователя, от имени которого запущена служба Microsoft SQL Server: https://msdn.microsoft.com/ru-ru/library/ms190730(v=sql.120).aspx

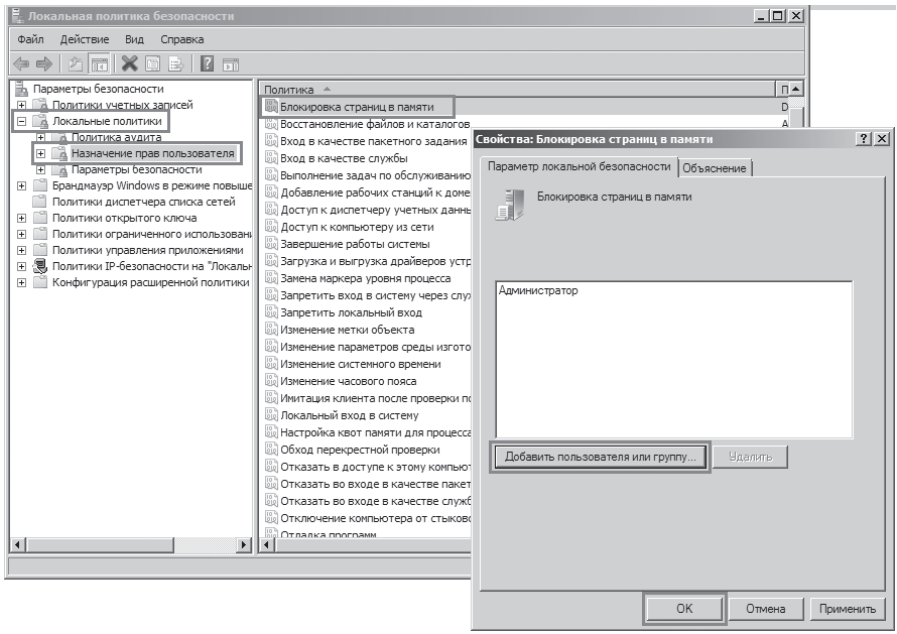


Рисунок 50. Разрешение на блокировку страниц в памяти

Если сервер «1С:Предприятия» установлен вместе с Microsoft SQL Server, то данную настройку производить не нужно.

Схема управления питанием – «Высокая производительность»

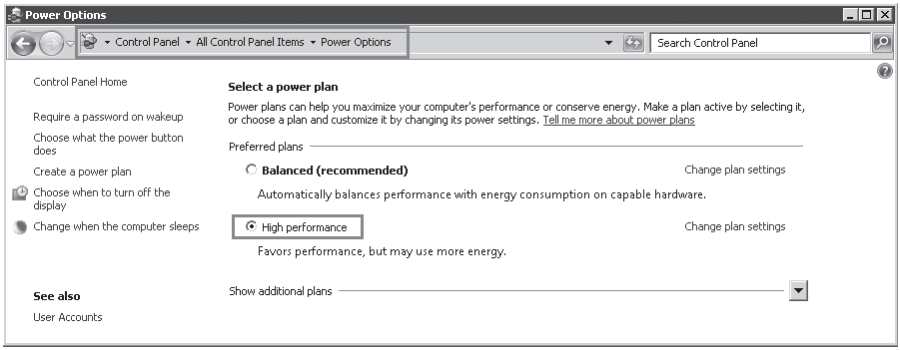


Рисунок 51. Схема управления питанием

Проверить отсутствие сжатия файлов данных и файлов журналов

Такую проверку имеет смысл выполнить также для системных баз (например, tempdb).

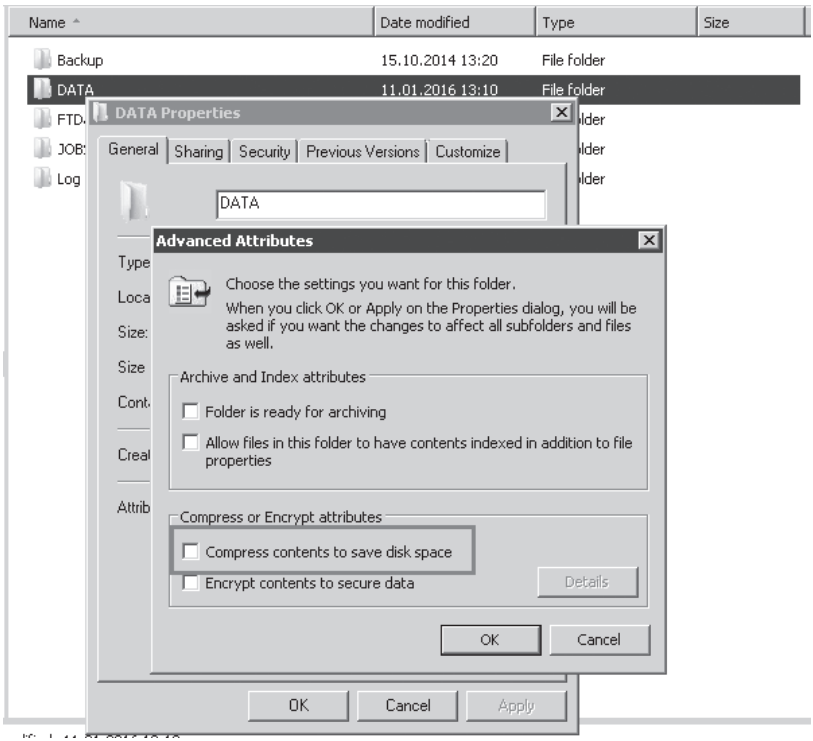


Рисунок 52. Настройка сжатия каталогов с данными и файлами журналов

Добавить файлы данных и журнала транзакций в исключения системы автоматического резервного копирования.

Системы автоматического резервного копирования (например, Symantec Backup Exec) не должны копировать файлы базы данных и журнала транзакций.

Желательно не настраивать антивирусные решения на серверах СУБД.

Настройки сервера

Версия

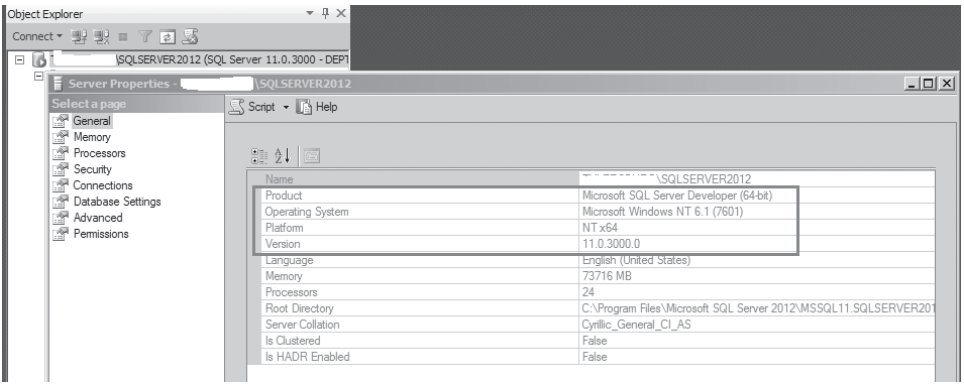


Рисунок 53. Проверка версии

```
SELECT @@VERSION
SELECT
        SERVERPROPERTY('productversion') AS productversion,
        SERVERPROPERTY('productlevel') AS productlevel,
        SERVERPROPERTY('edition') AS edition,
        SERVERPROPERTY('Collation') AS Collation,
        SERVERPROPERTY('ResourceLastUpdateDateTime') AS ResourceLastUpdateDateTime
```

Где взять обновление и узнать версию по номеру: https://support.microsoft.com/en-us/kb/321185.

Использование памяти

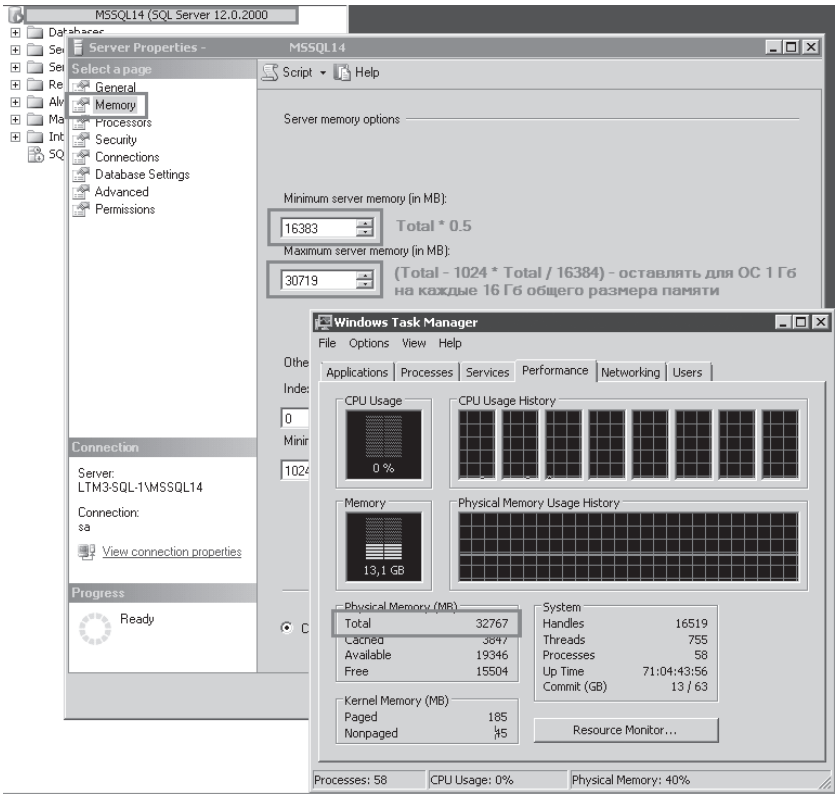


Рисунок 54. Использование памяти

Если сервер «1С:Предприятия» установлен вместе с Microsoft SQL Server, то верхний порог памяти необходимо уменьшить на величину, достаточную для работы сервера «1С».

Установить флаг Boost SQL Server priority

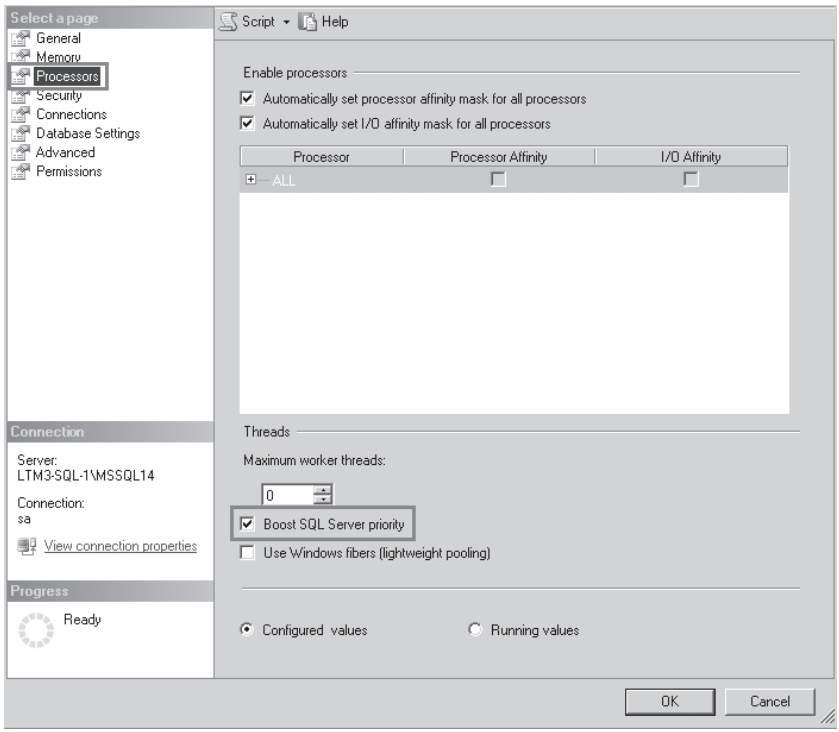


Рисунок 55. Флаг Boost SQL Server priority

Если сервер «1С:Предприятия» установлен вместе с Microsoft SQL Server, то данную настройку производить не нужно.

Задать расположение файлов базы данных по умолчанию

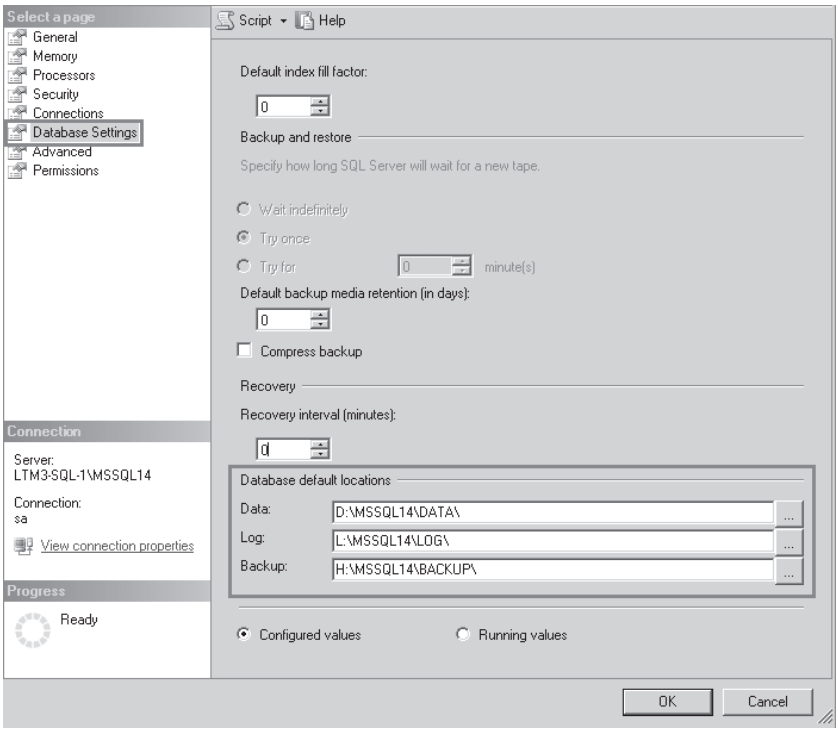


Рисунок 56. Расположение файлов БД по умолчанию

Index fill factor может быть полезен при массовых INSERT, но лучше его установить для конкретного индекса.

Файлы данных и файлы журналов транзакций желательно размещать на разных дисковых массивах. При этом требование к скорости дисковой подсистемы файла журнала транзакций выше, чем у файла данных. Согласно рекомендации от Microsoft, время отклика «диска» с файлами базы данных должно составлять 10–20 миллисекунд, а «диска» с файлами журнала транзакций 1–5 мс.

Max degree of parallelism

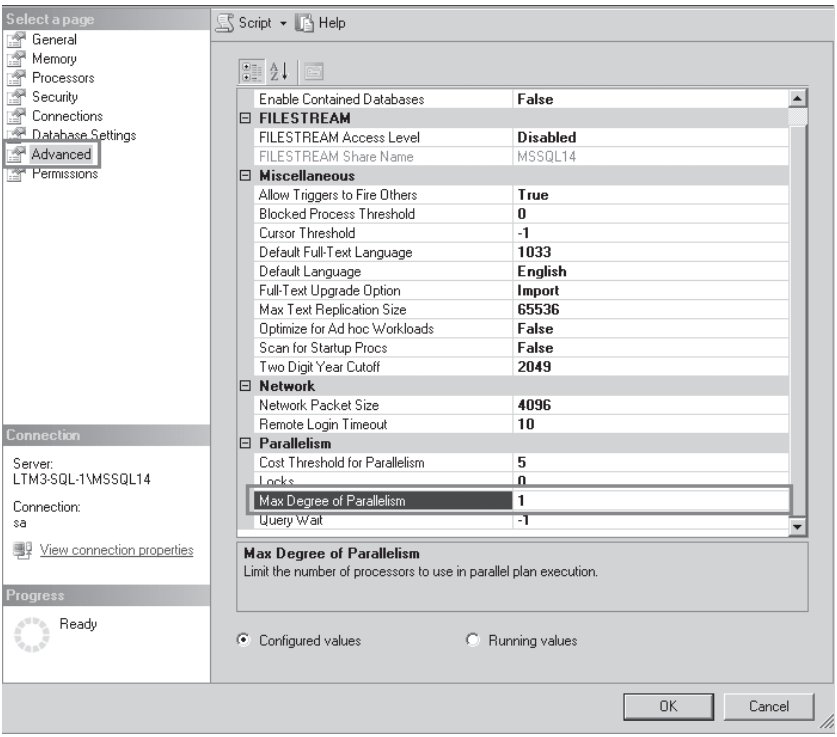


Рисунок 57. Параметр Max degree of parallelism

Параметры Max Degree of Parallelism и Cost Threshold for Parallelism необходимо выбрать исходя из вашего профиля нагрузки. Если профиль нагрузки еще не определен, то установить MAXDOP=1.

Включить аутентификацию SQL Server

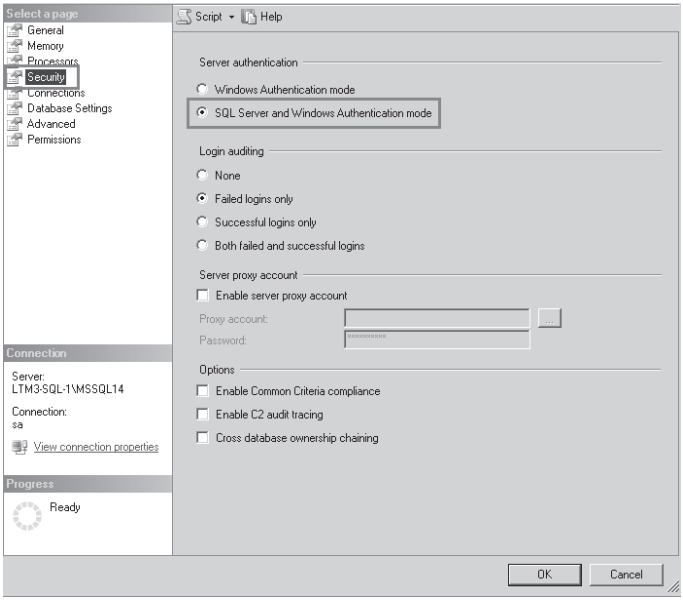


Рисунок 58. Аутентификация SQL Server

Может быть подключение с разных точек и от разных пользователей Windows или Linux.

Создать логины для каждой рабочей базы и назначить логинам роли: public, dbcreator

Рисунок 59. Настройка прав

Включить возможность административного подключения

https://msdn.microsoft.com/ru-ru/library/ms189595(v=sql.120).aspx

```
EXEC sp_configure 'remote admin connections', 1
GO
RECONFIGURE
GO
```

Полезные флаги трассировки

- 4199 – для Microsoft SQL Server 2014 необходимо включить исправление ошибок оптимизатора (https://support.microsoft.com/en-us/kb/974006).
- 1117 – одновременный рост всех файлов данных (недокументированный флаг).
- 1118 – не использовать смешанные экстенты (когда страницы разных объектов располагаются в одном экстенте). Подробнее: https://support.microsoft.com/en-us/kb/2154845 https://msdn.microsoft.com/en-us/library/ms188396.aspx. Для Microsoft SQL Server 2016 данная настройка включена по умолчанию.
- 1211 – отключает эскалацию во всех случаях – как по превышению памяти, так и по количеству блокировок.
- 1224 – отключает эскалацию по количеству блокировок.

Настройка сетевых протоколов

Включить протокол TCP/IP. Если сервер «1С:Предприятия» расположен вместе с Microsoft SQL Server – включить протокол Shared Memory.

Протокол Named pipes необходимо отключить.

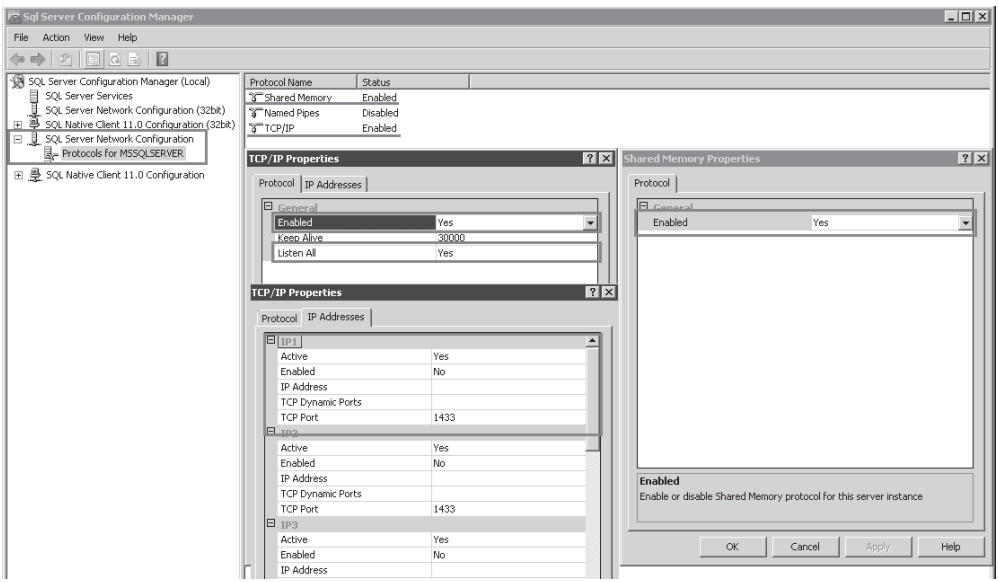


Рисунок 60. Настройка протоколов

База данных model

Новая база данных создается из копии базы *model*. Все настройки, указанные в *model*, будут в новой базе данных.

Начальный размер файла данных – от 1 Гб до 10 Гб.

Начальный размер журнала транзакций – от 1 Гб до 2 Гб.

Прирост файлов – 512 Мб.

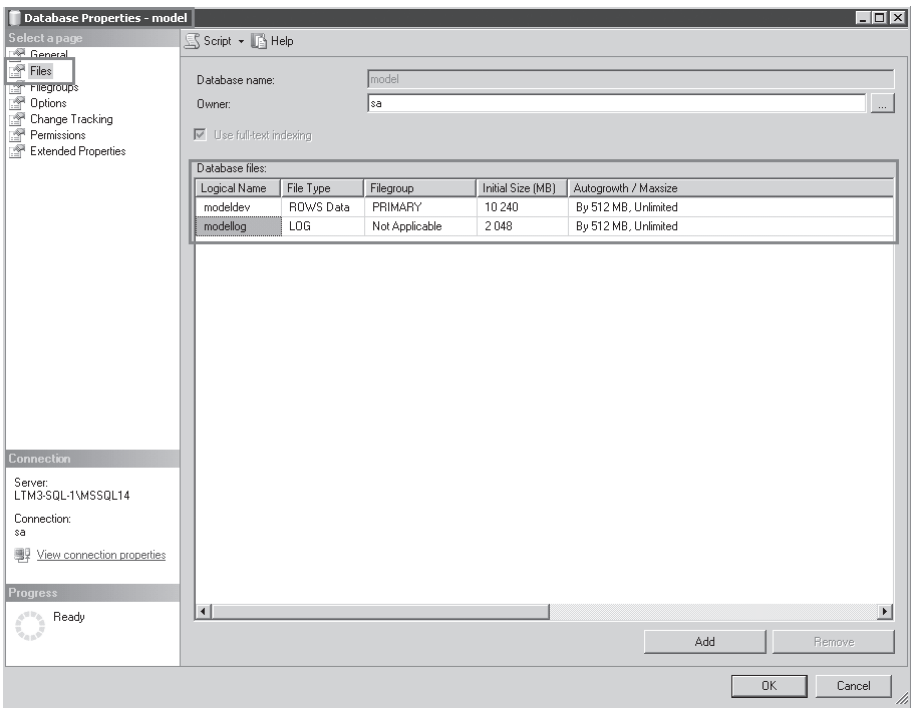


Рисунок 61. База данных model

Установить модель восстановления в зависимости от политики резервного копирования.

Auto Update Statistics Asynchronously

Параметр Auto Update Statistics Asynchronously необходимо устанавливать в зависимости от характера нагрузки.

Если планируется многократное изменение таблиц данных без скачкообразного изменения количества записей, то Auto Update Statistics Asynchronously = True. Если планируется, что будет сильное изменение количества записей в таблицах (например, загрузки данных из внешних систем), тогда Auto Update Statistics Asynchronously = False.

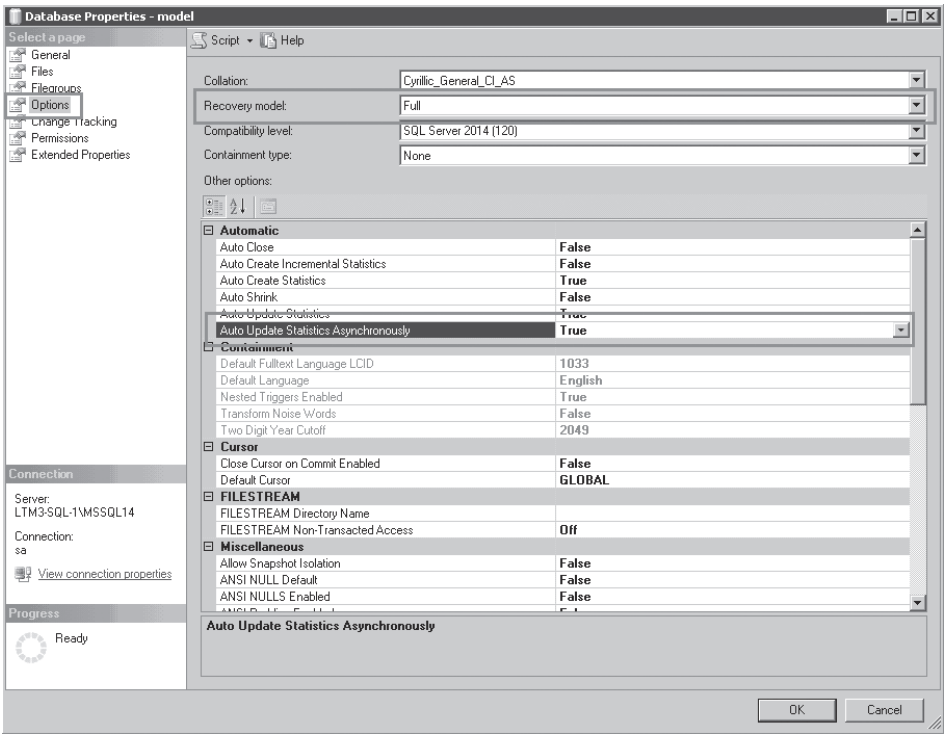


Рисунок 62. Параметр Auto Update Statistics Asynchronously

База tempdb

Возьмем для разбора следующий случай: очень нагруженная база; «1С» постоянно создает временные таблицы; помимо «1С» в нее пишет данные и сам сервер БД.

Желательно размещать на максимально быстрых дисках.

Если под tempdb выделен отдельный диск (массив), то необходимо установить размер файлов таким образом, чтобы был занят весь диск. Затем запретить автоматическое приращение файлов (auto grow).

Если tempdb расположена вместе с рабочими базами данных, то начальный размер установить от 10 Гб и прирост файлов по 512 Мб.

Количество файлов данных рекомендуется указывать до суммарного количества ядер сервера. Однако тесты показывают, что после 8 файлов данных рост производительности будет несущественным.

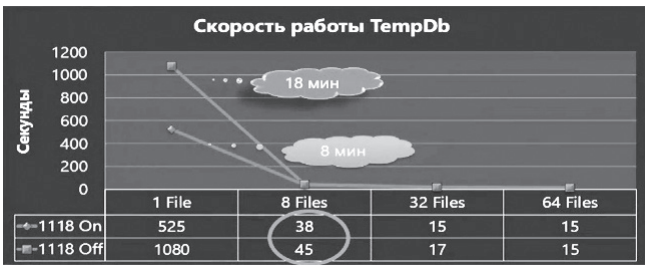


Рисунок 63. Тест скорости работы tempdb

Файл журнала транзакций должен быть один.

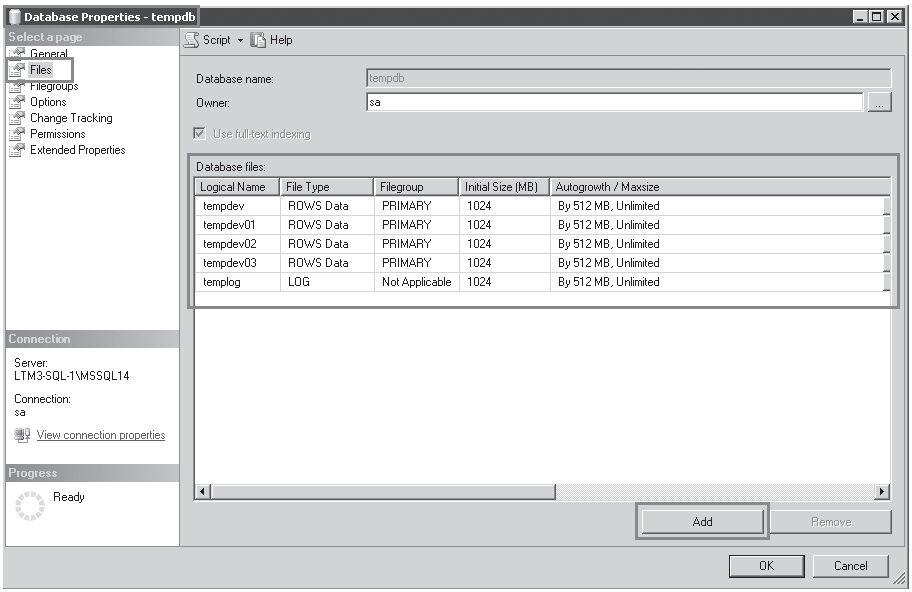


Рисунок 64. Файл журнала транзакций

Рабочая база

Параметры рабочей базы аналогичны параметрам базы model, за исключением начального размера файлов (Initial Size).

Начальный размер файла данных стоит указывать равным ожидаемому размеру базы за длительный период эксплуатации.

Размер файла журнала транзакций следует указывать такой, чтобы исключить его расширение (auto grow). То есть указанного размера файла журнала должно хватать на весь период работы между операциями BACKUP LOG.

Резервное копирование

Модели восстановления

Существуют 3 модели восстановления:

- **Full** (режим полного протоколирования) – в этом режиме максимальное количество операций записывается в журнал транзакций. Журнал транзакций автоматически не обрезается. Этот режим обеспечивает максимальные возможности восстановления (за счет небольшого снижения производительности).
- **Bulk-logged** (режим неполного протоколирования) – это компромисс между требованиями производительности и возможностями восстановления. При использовании этого режима запись в журнал практически отключается для операций следующих типов:
  - массовая вставка (команды BULK INSERT, SELECT INTO, загрузка средствами bcp и т. п.);
  - вставка/изменение больших двоичных данных (text, ntext, image);
  - операции по созданию, перестроению и удалению индексов.

  Автоматическая перезапись журналов транзакций при этом не производится, работа с транзакциями, не включающими в себя перечисленные операции, производится как обычно. При работе в этом режиме вы лишаетесь возможности использовать журнал транзакций для восстановления (при утрате файлов данных, на момент времени или на метку транзакции), если в нем была хотя бы одна запись о перечисленных ранее операциях. Microsoft рекомендует не использовать этот режим восстановления на постоянной основе, а переключаться в него из режима Full на время выполнения больших операций массовой вставки и потом возвращаться обратно.
- **Simple** (простая модель восстановления) – максимальный выигрыш в производительности и удобстве работы за счет возможностей восстановления. Минимально протоколируются те же операции, что и в режиме **Bulk-logged**, кроме того, журнал транзакций автоматически очищается (блоками, размер которых изначально равен 256 Кб, но при необходимости он может быть автоматически увеличен). В результате получаем максимальную производительность и возможность не думать о потенциальной нехватке места в журнале транзакций (не всегда). Но в этом режиме использовать журнал транзакций для восстановления уже не удастся.

Виды резервного копирования

Полное резервное копирование (*full backup* или *base backup*). В резервную копию записываются все данные, которые есть в базе. Пустые страницы при этом не копируются, поэтому если, например, файлы вашей базы данных имеют размер 1 Гб, а реально данные в них занимают всего лишь 200 Мб, то получится резервная копия размером 200 Мб.

Конечно, полное резервное копирование, как и все другие типы резервного копирования, производится в оперативном режиме (*online*), без отключения пользователей.

Есть момент, с которым часто возникает путаница. Предположим, что полное резервное копирование базы данных началось в 7 часов, а закончилось в 8. Данные по состоянию на какой момент оказались помещены в резервную копию: на 7 или 8 часов?

Правильный ответ – на 8 часов. Действительно, в момент начала резервного копирования база данных стабилизируется (т. е. никакие изменения в ее файлы не вносятся для обеспечения целостности резервной копии). Однако после того, как перенос самой базы данных завершен, к резервной копии дописывается информация о всех изменениях, которые были внесены в базу данных во время резервного копирования, то есть с 7 до 8 часов.

Разностное резервное копирование (*differential backup*). В этом случае в резервную копию записываются все изменения, которые были произведены с момента **полного** резервного копирования. Именно с момента последнего **полного** резервного копирования, а не другого разностного! Разностное резервное копирование можно использовать только в дополнение к полному.

Резервное копирование журналов транзакций (*transaction log backup*). Работает только в том случае, если для БД установлен режим восстановления **Full** или **Bulk-logged**. Если не производить резервное копирование журналов транзакций, то не будет производиться и их очистка. В результате место в файлах журналов транзакций может закончиться (а если для них установлен неограниченный размер, то закончится и место на диске). Это так называемое инкрементальное резервное копирование. В резервной копии сохраняются только измененные данные с момента последнего резервного копирования. Для того чтобы восстановить базу данных полностью, необходимо не потерять все резервные копии журналов транзакций начиная с последнего полного (или разностного) резервного копирования.

В качестве дополнительного типа резервного копирования можно рассматривать **резервное копирование файлов** (*file backup*) и **файловых групп** (*filegroup backup*). Применяется оно достаточно редко и обычно только в двух ситуациях:

- когда резервная копия всей базы данных не помещается на носитель. В этом случае размер файлов или файловых групп подбирается под размер носителя;
- когда в базе данных таблицы можно условно поделить на две группы: таблицы-справочники, в которые изменения практически не вносятся, и пользовательские таблицы, которые изменяются активно. В этом случае каждая группа таблиц помещается в свою файловую группу, а затем проводится резервное копирование каждой файловой группы со своей частотой.

Возможные сценарии резервного копирования

В 9 из 10 случаев на отечественных предприятиях используется самое простое расписание резервного копирования: каждую ночь производится полное резервное копирование всей базы данных, и сразу после этого – резервное копирование журналов транзакций.

Корпорация Microsoft считает идеальным вариантом другое расписание:

- раз в неделю – полное резервное копирование;
- раз в сутки (каждую ночь) – разностное резервное копирование;
- несколько раз в день – резервное копирование журналов транзакций.

Этот вариант наилучшим образом подходит для больших баз данных (сотни гигабайт), которые активно изменяются.

Восстановление базы данных

При возникновении любой проблемы с БД, требующей восстановления этой БД, **обязательно** (если это возможно) нужно сделать резервную копию журнала транзакций без его обрезки (это позволит впоследствии восстановить базу данных до состояния на момент сбоя):

```
BACKUP LOG <Имя базы данных> TO DISK='...' WITH NO_TRUNCATE
```

Затем необходимо запретить пользователям доступ к базе данных, подлежащей восстановлению. Это можно сделать разными способами:

- Установить для параметра Restrict Access в свойствах базы данных значение Restricted. Если же пользователи базы данных могут подключаться с правами dbo, то для этого параметра можно установить значение Single.
- Если на сервере имеется только одна рабочая база данных, то лучше просто на время восстановления отключить сетевой доступ к SQL Server. Для этого можно, например, на время восстановления отключить протокол TCP/IP в контейнере SQL Server Network Configuration в SQL Server Configuration Manager.

Может случиться так, что база данных повреждена настолько сильно, что изменить ее свойства не удается. Она при этом может находиться в состоянии suspect (подозрительное) или в автономном режиме offline. Если база данных находится в автономном режиме, то запустить ее восстановление вам не удастся. В этой ситуации самый простой выход – отсоединить (detach) поврежденную базу данных и произвести восстановление с резервной копии так, как будто эта база данных отсутствует на сервере вообще. Для того чтобы отсоединить базу данных, помеченную как подозрительная (suspect), ее необходимо вначале перевести в состояние «экстренной необходимости» (emergency): ALTER DATABASE db1 SET emergency.

Если база данных находится в рабочем режиме, то лучше подстраховаться, создав еще одну, самую свежую, резервную копию этой базы.

Чтобы восстановить базу данных за секунду до сбоя, необходимо:

1. Сделать резервную копию журнала транзакций.
2. Восстановить самый поздний полный бэкап, оставив базу в режиме NORECOVERY.
3. Восстановить самый поздний разностный бэкап, оставив базу в режиме NORECOVERY.
4. Последовательно восстановить бэкапы журналов транзакций до последнего, в котором содержатся данные на момент сбоя, оставив базу в режиме NORECOVERY.
5. При восстановлении последнего бэкапа журнала транзакций указать время, до которого его необходимо восстановить, и выбрать, что база должна перейти в режим RECOVERY.

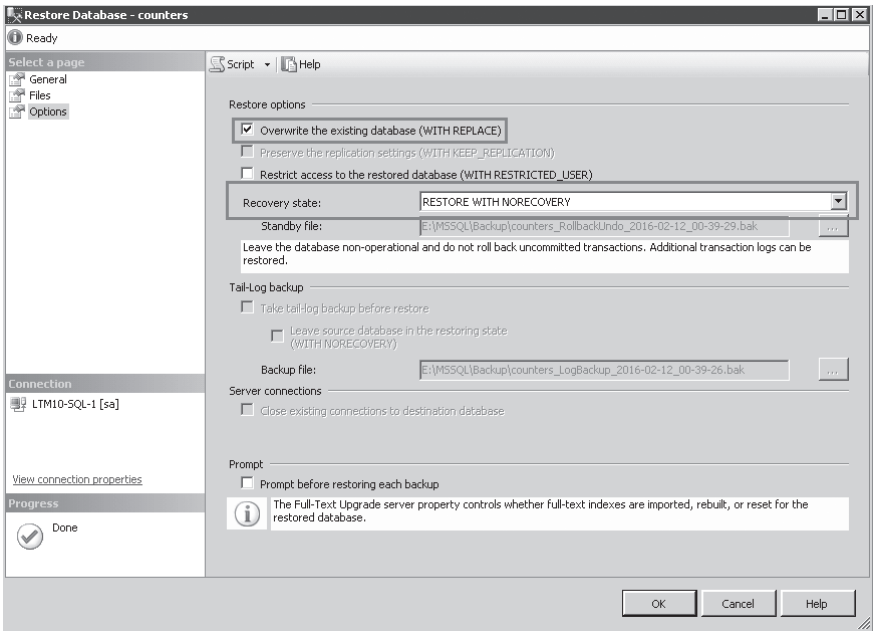


Рисунок 65. Настройки восстановления БД

Сценарий восстановления базы данных

Администратор должен иметь план восстановления базы данных на случай сбоя. План должен быть напечатан на бумаге и находиться в доступном месте.

Необходимо хотя бы один раз пройти все пункты данного плана, «проиграв» ситуацию с восстановлением на резервном сервере.

Мониторинг

Базовые инструменты

Error log

Находится примерно здесь:

```
C:\Program Files\Microsoft SQL Server\MSSQL11.MSSQLSERVER\MSSQL\Log
```

Содержит информацию о запуске/остановке сервера, неудачных попытках подключения, внутренних ошибках и т. д.

Database mail account

Для отправки писем об ошибках: https://msdn.microsoft.com/ru-ru/library/hh245116(v=sql.120).aspx.

Операторы

Необходимо создать операторов, которым будут отправляться сообщения об ошибках.

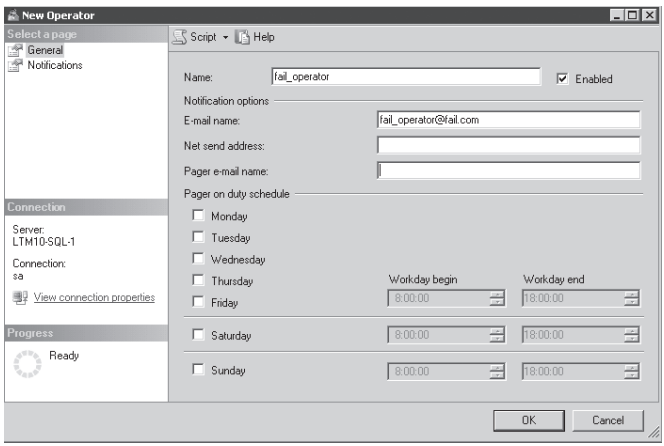


Рисунок 66. Добавление операторов

В «планах обслуживания» при ошибках отправлять оповещение оператору.

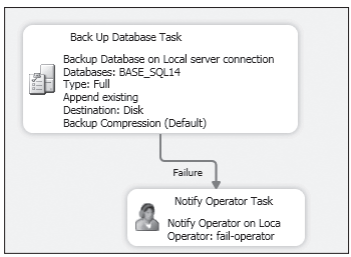


Рисунок 67. «План обслуживания»

Трассировка через Extended events

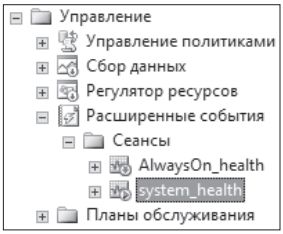


Рисунок 68. Настроенная трассировка через Extended events

```
CREATE EVENT SESSION [dbmssql] ON SERVER
ADD EVENT sqlserver.rpc_completed(
    ACTION(sqlserver.database_id,sqlserver.session_id,sqlserver.sql_text)
    WHERE ([sqlserver].[database_name]=N'ИМЯ БАЗЫ ДАННЫХ')),
ADD EVENT sqlserver.sql_batch_completed(
    ACTION(sqlserver.database_id,sqlserver.session_id,sqlserver.sql_text)
    WHERE ([sqlserver].[database_name]=N'ИМЯ БАЗЫ ДАННЫХ'))
ADD TARGET package0.event_file(SET filename=N'E:\MSSQL\Trace\counters.xel')
WITH (MAX_MEMORY=4096 KB,EVENT_RETENTION_MODE=ALLOW_SINGLE_EVENT_LOSS,MAX_DISPATCH_LATENCY=30
SECONDS,MAX_EVENT_SIZE=0 KB,MEMORY_PARTITION_MODE=NONE,TRACK_CAUSALITY=OFF,STARTUP_STATE=OFF)

-- запуск трассировки запросов
ALTER EVENT SESSION [dbmssql] ON SERVER STATE = START

-- остановка трассировки запросов
ALTER EVENT SESSION [dbmssql] ON SERVER STATE = STOP

-- удаление трассировки запросов
DROP EVENT SESSION [dbmssql] ON SERVER

-- чтение файлов трассировки
SELECT
        event_data.value('(event/action[@name="sql_text"])[1]', 'nvarchar(max)') AS TextData,
        event_data.value('(event/action[@name="database_id"])[1]', 'int') AS DatabaseID,
        event_data.value('(event/action[@name="session_id"])[1]', 'int') AS SPID,
        event_data.value('(event/data[@name="duration"])[1]', 'bigint') AS Duration,
        event_data.value('(event/@timestamp)[1]', 'datetime') AS EndTime,
        event_data.value('(event/data[@name="physical_reads"]/value)[1]', 'bigint') AS Reads,
        event_data.value('(event/data[@name="writes"])[1]', 'bigint') AS Writes,
        event_data.value('(event/data[@name="cpu_time"])[1]', 'bigint') AS CPU
FROM (
        SELECT CAST(event_data AS XML) AS event_data
        FROM sys.fn_xe_file_target_read_file('E:\MSSQL\Trace\counters*.xel', null, null, null)
) xel
```

Dynamic Management Views (DMV)

Динамические административные представления и функции возвращают данные о состоянии сервера, которые могут использоваться для контроля исправности экземпляра сервера, диагностики проблем и настройки производительности: https://msdn.microsoft.com/ru-ru/library/ms188754(v=sql.120).aspx

Пример получения ожиданий, используя DMV

```
USE master
GO

/*
DBCC SQLPERF ('sys.dm_os_wait_stats', CLEAR);
GO
*/

WITH ByWaitTypes([Тип ожидания], [ожидания сигнала %], [ожидания ресурса %], [ожидания ms]) AS
(
SELECT TOP 20 wait_type
  , cast(100.0 * sum(signal_wait_time_ms)/sum(wait_time_ms) AS NUMERIC (20,2))
  , cast(100.0 * sum(wait_time_ms  – signal_wait_time_ms)/sum(wait_time_ms) AS NUMERIC(20,2))
  , sum(wait_time_ms)
FROM sys.dm_os_wait_stats
WHERE wait_time_ms <> 0 AND wait_type NOT LIKE '%SLEEP%'
GROUP BY wait_type
ORDER BY sum(wait_time_ms) DESC
)
SELECT TOP 1 'Тип ожидания' = N'ВСЕ!'
  , 'ожидания сигнала %' = (SELECT cast(100.0 * sum(signal_wait_time_ms)/
   sum (wait_time_ms) AS NUMERIC (20,2)) FROM sys.dm_os_wait_stats)
  , 'ожидания ресурса %' =(SELECT cast(100.0 * sum(wait_time_ms  – signal_wait_time_ms)/
   sum(wait_time_ms) AS NUMERIC(20,2)) FROM sys.dm_os_wait_stats)
  , 'ожидания ms' =(SELECT sum(wait_time_ms) FROM sys.dm_os_wait_stats)
FROM sys.dm_os_wait_stats
UNION
SELECT [Тип ожидания], [ожидания сигнала %], [ожидания ресурса %], [ожидания ms]
FROM ByWaitTypes
ORDER BY [ожидания ms] DESC
```

Полезные материалы в «Технологической базе знаний» kb.1c.ru

Высокая загрузка дисковой подсистемы на сервере СУБД MS SQL Server:

http://kb.1c.ru/articleView.jsp?id=77

Высокая загрузка CPU на сервере СУБД MS SQL Server. Часть 1:

http://kb.1c.ru/articleView.jsp?id=76

Высокая загрузка CPU на сервере СУБД MS SQL Server. Часть 2:

http://kb.1c.ru/articleView.jsp?id=105

Полезные и популярные запросы к DMV

См. в приложении 4.

Клиентские компоненты MS SQL на компьютере администратора БД

Необходимо заранее установить на локальные машины администраторов клиентскую часть Microsoft SQL Server: SQL Server Management Studio или OSQL.EXE (https://www.microsoft.com/ru-RU/download/details.aspx?id=36433).

Счетчики производительности

Набор рекомендуемых счетчиков:

```
\Memory\Available Mbytes
\Memory\Pages/sec

\LogicalDisk(_Total)\Free Megabytes
\LogicalDisk(*)\% Disk Time
\LogicalDisk(*)\% Disk Idle Time
\LogicalDisk(*)\% Disk Write Time
\LogicalDisk(*)\% Disk Read Time

\Processor(_Total)\% Processor Time

\System(_Total)\Processor Queue Length

\PhysicalDisk(_Total)\Avg. Disk Queue Length
\PhysicalDisk(*)\Avg. Disk Queue Length

\Network Interface(*)\Bytes Total/sec

\PhysicalDisk(_Total)\Avg. Disk sec/Read
\PhysicalDisk(_Total)\Avg. Disk sec/Write
\PhysicalDisk(*)\% Disk Time
\PhysicalDisk(*)\% Disk Idle Time
\PhysicalDisk(*)\% Disk Write Time
\PhysicalDisk(*)\% Disk Read Time

\Paging File(_Total)\*

\SQLServer:Access Methods\Full Scans/sec
\SQLServer:Buffer Manager\Buffer cache hit ratio
\SQLServer:Buffer Manager\Free list stalls/sec
\SQLServer:Buffer Manager\Lazy writes/sec
\SQLServer:Buffer Manager\Page life expectancy
\SQLServer:Buffer Node\Page life expectancy
\SQLServer:Databases\Active Transactions
\SQLServer:Databases(_Total)\Transactions/sec
\SQLServer:General Statistics\Active Temp Tables
\SQLServer:General Statistics\Transactions
\SQLServer:Locks(_Total)\Average Wait Time (ms)
\SQLServer:Locks(_Total)\Lock Requests/sec
\SQLServer:Locks(_Total)\Lock Timeouts (timeout &gt; 0)/sec
\SQLServer:Locks(_Total)\Lock Timeouts/sec
\SQLServer:Locks(_Total)\Lock Wait Time (ms)
\SQLServer:Locks(_Total)\Lock Waits/sec
\SQLServer:Locks(_Total)\Number of Deadlocks/sec
\SQLServer:Memory Manager\Memory Grants Pending
\SQLServer:Wait Statistics\Lock waits
\SQLServer:Wait Statistics(*)\Log buffer waits
\SQLServer:Wait Statistics(*)\Log write waits
\SQLServer:Wait Statistics(*)\Network IO waits
\SQLServer:Wait Statistics(*)\Non-Page latch waits
\SQLServer:Wait Statistics(*)\Page IO latch waits
\SQLServer:Wait Statistics(*)\Page latch waits
```

Обслуживание базы данных

Периодические операции обслуживания, которые рекомендуется проводить с базой данных MS SQL Server:

- обновление статистик;
- очистка процедурного кеша;
- дефрагментация индексов;
- реиндексация таблиц базы данных.

На назначении каждой операции останавливаться не будем, так как это стандартные операции администрирования СУБД, а повторять материал, многократно изложенный в многочисленных источниках, не хочется. Подробнее о порядке проведения регламентных процедур и их настройке написано здесь: http://kb.1c.ru/articleView.jsp?id=13.

Опишем специфические вещи, которые позволяют повысить качество эксплуатации информационной системы.

Например, мы говорим, что нужно проводить дефрагментацию и периодическое перестроение индексов. Но с точки зрения задач эксплуатации все операции обслуживания должны укладываться в технологическое окно, а полная дефрагментация или перестроение индексов – операции довольно длительные на крупных базах. Чтобы максимально эффективно проводить регламентные процедуры на СУБД, не пересекая при этом границы технологического окна, можно использовать скрипт, позволяющий проводить эти регламентные операции избирательно, при этом давая максимальный полезный эффект в пределах временных рамок технологического окна:

```
USE tempdb;
SET NOCOUNT ON;
SET QUOTED_IDENTIFIER ON;
GO

IF OBJECT_ID('handdleFragmentationIndexes','P') IS NOT NULL
BEGIN
DROP PROCEDURE  handdleFragmentationIndexes
PRINT 'handdleFragmentationIndexes already exist and was DROPED'
END
ELSE PRINT 'handdleFragmentationIndexes is not exist and will CREATED'
GO

CREATE PROCEDURE handdleFragmentationIndexes (@debugMode bit = 0, @mode varchar(50)
                                                                    = 'DETAILED', @databaseName varchar(50) = 'ALL')
AS
BEGIN

-- @mode =
-- LIMITED (default): This mode is the fastest and scans the smallest number of pages. It scans all pages for a heap,
but only scans the parent-level pages, which means, the pages above the leaf-level, for an index.
-- SAMPLED This mode returns statistics base on a one percent sample of the entire page in the index or heap.
If the index or heap has fewer than 10 000 pages, DETAILD mode is used instead of SAMPLED.
-- DETAILED Detailed mode scans all pages and returns all statistics. Be careful, from LIMITED to SAMPLED to DETAILED,
the mode are progressively slower, because more work is performed in each. In my script I am using this one
```

```
SET NOCOUNT ON;

DECLARE @objectid int;
DECLARE @indexid int;
DECLARE @partitioncount bigint;
DECLARE @schemaname sysname;
DECLARE @objectname sysname;
DECLARE @indexname sysname;
DECLARE @partitionnum bigint;
DECLARE @partitions bigint;
DECLARE @frag float;
DECLARE @spaceUsed float;
DECLARE @command varchar(MAX);
DECLARE @command2 varchar(MAX);
DECLARE @commandStatistics varchar(8000);
DECLARE @database_id int;
DECLARE @t1 DATETIME;
DECLARE @t2 DATETIME;
DECLARE @time_diff int;

DECLARE @db_id int;
DECLARE @db_name varchar(50);
DECLARE @allow_page_locks int;

SET @t1 = GETDATE();

CREATE TABLE #med_info_index(object_id int ,index_id int ,partition_number int ,avg_fragmentation_in_percent
float,[db_name] varchar(150),[avg_page_Space_Used_in_Percent] float)
CREATE TABLE #med_info_index_detail ([objectid] int, [indexid] int, [partitionnum] int, [frag] float,[objectname] varchar(150),
[schemaname] varchar(150), [db_name] varchar(150),[indexname] varchar(150),[partitioncount] int, [allow_page_locks] int,
[avg_page_Space_Used_in_Percent] float)

-- STEP 1
PRINT 'Step 1 starting... ' + CAST (GETDATE() AS nvarchar(max))
PRINT '- Get all index with a fragmation over 20% for all database'
-----------------------------------------------

IF @databaseName = 'ALL'
   SET @database_id = NULL;
ELSE
   SELECT @database_id = ISNULL(DB_ID(@databaseName),0);

INSERT #med_info_index(object_id, index_id, partition_number, avg_fragmentation_in_percent,[db_name],
[avg_page_Space_Used_in_Percent])
   SELECT object_id, index_id, partition_number, avg_fragmentation_in_percent,db_name(database_id) as [db_name],
   ISNULL(avg_page_Space_Used_in_Percent,100)
   FROM sys.dm_db_index_physical_stats(@database_id, NULL, NULL, NULL , @mode)
   WHERE avg_fragmentation_in_percent >= 20
   AND database_id IN (SELECT database_id FROM sys.databases
               WHERE [state] = 0 /*ONLINE*/ AND is_read_only = 0 AND database_id > 4 /*SKIP SYSTEM DB*/
            -- Documentation : http://msdn.microsoft.com/en-us/library/ms178534.aspx
               )

-- STEP 2
PRINT 'Step 2 starting... ' + CAST (GETDATE() AS nvarchar(max))
PRINT '- Get all objects details about the index with a fragmentation over 20% for all database'
-----------------------------------------------
DECLARE dbList CURSOR FOR
   SELECT object_id, index_id, partition_number, avg_fragmentation_in_percent,[db_name],[avg_page_Space_Used_in_Percent]
   FROM #med_info_index
 FOR READ ONLY

OPEN dbList
```

```
FETCH NEXT FROM dbList  INTO @objectid, @indexid, @partitionnum, @frag,@db_name,@spaceUsed;
  WHILE @@FETCH_STATUS = 0
    BEGIN

       SET @command2 = ' DECLARE @objectname varchar(50), @indexname varchar(50),@schemaname varchar(50),
                                                                                @partitioncount int, @allow_page_locks int; '
       +  ' SELECT @objectname = o.name, @schemaname = s.name '
       +  ' FROM ['+@db_name+'].sys.objects AS o '
       +  ' JOIN sys.schemas as s ON s.schema_id = o.schema_id '
       +  ' WHERE o.object_id = ' + CAST(@objectid as varchar(50))
       +  ' '
       +  ' SELECT @indexname = name, @allow_page_locks=allow_page_locks '
       +  ' FROM ['+@db_name+'].sys.indexes'
       +  ' WHERE  object_id = '+CAST(@objectid as varchar(50))+' AND index_id = '+CAST(@indexid as varchar(50))
       +  ' '
       +  ' SELECT @partitioncount = count (*) '
       +  ' FROM ['+@db_name+'].sys.partitions '
       +  ' WHERE object_id = '+CAST(@objectid as varchar(50))+' AND index_id = '+CAST(@indexid as varchar(50))
       +  ' '
       +  ' INSERT INTO #med_info_index_detail '
       +  ' ([objectid], [indexid], [partitionnum], [frag],[objectname], [schemaname],[indexname],[partitioncount],
[db_name],[allow_page_locks],[avg_page_Space_Used_in_Percent])'
       +  ' VALUES ('''+CAST(@objectid as varchar(50))+''', '''+CAST(@indexid as varchar(50))+''', '''+CAST(@partitionnum as
varchar(50))+''', '''+CAST(@frag as varchar(50))+''',@objectname, @schemaname,@indexname, @partitioncount, '''
+@db_name+''', @allow_page_locks, '''+CAST(@spaceUsed as varchar(50))+''');'
       +  ' ';

       EXEC (@command2);

    FETCH NEXT FROM dbList  INTO @objectid, @indexid, @partitionnum, @frag,@db_name,@spaceUsed;
    END
CLOSE dbList;
DEALLOCATE dbList;

-- STEP 3
PRINT 'Step 3 starting... ' + CAST (GETDATE() AS nvarchar(50))
PRINT '- reorganizing and rebuilding all index in the previous list '
-------------------------------------------------
DECLARE DefragList CURSOR FOR
    SELECT [objectid], [indexid], [partitionnum], [frag],[objectname], [schemaname],[indexname],[partitioncount],
[db_name],[allow_page_locks],[avg_page_Space_Used_in_Percent]
    FROM  #med_info_index_detail
    ORDER BY [frag] DESC
    FOR READ ONLY

OPEN DefragList

FETCH NEXT FROM DefragList
INTO @objectid, @indexid, @partitionnum, @frag,@objectname, @schemaname,@indexname,@partitioncount,
@db_name,@allow_page_locks,@spaceUsed;

WHILE @@FETCH_STATUS = 0
BEGIN

IF (@indexid) IS NOT NULL
    BEGIN
       -- 30 is an arbitrary decision point at which to switch between reorganizing and rebuilding

       IF @objectname IS NOT NULL AND @indexname IS NOT NULL AND @partitioncount IS NOT NULL
       BEGIN
       IF ((@frag >= 20.0 AND @frag < 30.0) OR (@spaceUsed < 75.0 AND @spaceUsed > 60.0)) AND @allow_page_locks= 1
          BEGIN
```

```
            SELECT @command = 'USE ['+@db_name+']; ALTER INDEX [' + @indexname + '] ON ' + @schemaname + '.
[' + @objectname + '] REORGANIZE';
            IF @partitioncount > 1
               SELECT @command = @command + ' PARTITION=' + CONVERT (CHAR, @partitionnum);

            SELECT @commandStatistics = 'USE ['+@db_name+'];UPDATE STATISTICS ' + @schemaname + '.
[' + @objectname + '] ( '+ @indexname + ') WITH FULLSCAN;'

            IF @debugMode = 1 PRINT GETDATE() + ' ' + @commandStatistics
            ELSE
               BEGIN
                  PRINT CAST (GETDATE() AS nvarchar(50)) + ' ' + @commandStatistics
                  EXEC (@commandStatistics)
               END
            IF @debugMode = 1 PRINT @command
            ELSE
               BEGIN
                  PRINT CAST (GETDATE() AS nvarchar(50)) + ' ' + @command
                  EXEC (@command)
               END

         END

      IF (@frag >= 30.0) OR (@spaceUsed < 60.0)
         BEGIN
            SELECT @command = 'USE ['+@db_name+']; ALTER INDEX [' + @indexname + '] ON ' + @schemaname + '.
[' + @objectname + '] REBUILD';

            IF @partitioncount > 1
            BEGIN
               SELECT @command = @command + ' PARTITION=' + CONVERT (CHAR, @partitionnum);

               -- if it's a partition we update statistics manually
               SELECT @commandStatistics = 'USE ['+@db_name+']; UPDATE STATISTICS ' + @schemaname + '.
[' + @objectname + '] ( '+ @indexname + ') WITH FULLSCAN;'

               IF @debugMode = 1 PRINT @commandStatistics
               ELSE
                  BEGIN
                     PRINT CAST (GETDATE() AS nvarchar(50))+ ' ' + @commandStatistics
                     EXEC (@commandStatistics)
                  END
            END

            IF @debugMode = 1 PRINT @command
            ELSE
               BEGIN
                  PRINT CAST (GETDATE() AS nvarchar(50)) + ' ' + @command
                  EXEC (@command)
               END
         END
      END
   END

FETCH NEXT FROM DefragList INTO @objectid, @indexid, @partitionnum, @frag,@objectname, @schemaname,
@indexname,@partitioncount, @db_name,@allow_page_locks,@spaceUsed;

END

CLOSE DefragList;
DEALLOCATE DefragList;
PRINT 'Step 3 end'
```

```
-- STEP 4
PRINT 'Step 4 starting...'
PRINT '- Remove temp table'
-----------------------------------------------------
DROP TABLE #med_info_index;
DROP TABLE #med_info_index_detail;
PRINT 'Step 4 end'

SET @t2 = GETDATE();
SELECT @time_diff = DATEDIFF(SECOND,@t1,@t2)

PRINT 'Execute time is ' + CAST(@time_diff / 60 / 60 / 24 % 7 AS nvarchar(50)) + ' days ' + CAST(@time_diff / 60 / 60 % 24
AS nvarchar(50)) + ' hours '
   + CAST(@time_diff / 60 % 60 AS nvarchar(50)) + ' minutes and ' + CAST(@time_diff % 60 AS nvarchar(50)) + ' seconds '
END

GO

EXECUTE handdleFragmentationIndexes

GO
DBCC FREEPROCCACHE
DROP PROCEDURE handdleFragmentationIndexes
```

Несмотря на довольно длинный текст, суть скрипта довольно очевидна:

- Для обработки берем только индексы, фрагментированные на 20 % и более.
- При фрагментации 20–30 % или в случае, когда использование выделенного места – 60–75 %, выполняем реорганизацию (REORGANIZE).
- При фрагментации 30 % и более или в том случае, если процент использования выделенного под индекс места менее 60, производим перестроение (REBUILD) индекса.

Такая выборочная работа с индексами позволяет не тратить время на обработку слабо фрагментированных индексов, реорганизация или перестроение которых повлияют на производительность системы незначительно, а также наиболее эффективно обработать те индексы, которые все-таки нуждаются в этом.

Отказоустойчивость

Отказоустойчивые кластеры (failover cluster)

Отказоустойчивые кластеры SQL Server *не могут применяться для повышения производительности*! Единственный выигрыш, который дает кластер SQL Server, – это выигрыш в отказоустойчивости. Выигрыш в производительности при этом обеспечивает именно добавление рабочих серверов в кластере платформы «1С:Предприятие». При этом следует понимать, что в отказоустойчивом кластере (при отсутствии запретов в требованиях назначения функциональности) работают все рабочие серверы кластера. Таким образом, добавление N серверов в кластер приводит к использованию всех, в т. ч. N добавленных, серверов, что и приводит к снижению удельной нагрузки на сервер.

Кластеры – это вовсе не лекарство от всех болезней! Кластеры не защитят:

- от ошибок пользователей;
- проблем с подсистемой хранения данных (RAID, SAN и т. п.);
- некорректных действий разработчиков приложения;
- ошибок в программном обеспечении самого SQL Server.

Однако кластеры решают проблемы с отказом некоторых аппаратных подсистем сервера, обеспечивают балансировку нагрузки и значительно лучшую масштабируемость системы с увеличением нагрузки и числа пользователей.

При использовании кластеров восстановление работоспособности приложения в случае аппаратного сбоя будет производиться намного быстрее и в автоматическом режиме.

Физически кластер – это два или более компьютера (желательно – близких по параметрам оборудования), которые подключены к внешнему устройству хранения (RAID, SAN и т. п.). Это внешнее устройство хранения обязательно должно обеспечивать подключение нескольких серверов одновременно. Также на серверах, которые входят в кластер, желательно установить по дополнительному сетевому адаптеру, который будет использоваться только для обмена информацией между элементами кластера. У каждого сервера есть свое имя, но пользователи при обращении к службе, работающей в кластере (например, SQL Server), видят не какое-то из этих имен, а имя третьего компьютера – виртуального сервера, работу которого и защищает кластер.

Кластер может работать в двух вариантах конфигурации: **активный/пассивный** и **активный/активный**.

Конфигурация **активный/пассивный** (*active*/*passive*) используется в большинстве случаев. В этом случае в обычном режиме нагрузку со стороны пользователей обслуживает только основной узел. Вторичный узел выполняет роль запасного и не несет какой-то нагрузки. Такая конфигурация является наиболее надежной и удобной, но при этом вторичный сервер будет фактически простаивать.

Чтобы он не стоял без дела, можно настроить конфигурацию **активный/активный** (*active*/*active*). В этой конфигурации два узла, входящие в кластер, обслуживают два виртуальных сервера – каждый со своей задачей. При этом для одной задачи первый сервер выполняет роль основного, а второй сервер – вторичного. А для второй задачи второй сервер выполняет роль основного, а первый – вторичного. При выходе из строя любого из серверов, входящих в кластер, всю нагрузку принимает на себя оставшийся сервер.

Зеркалирование (mirroring)

При использовании этого средства изменения, которые вносятся в базу данных на основном сервере (он называется **сервером-принципалом**), мгновенно (или почти мгновенно, в зависимости от выбранных параметров) отображаются в копии базы данных на другом сервере SQL Server.

Преимущества перед использованием кластера следующие:

- зеркальное отображение баз данных не требует применения специального оборудования;
- серверы, которые участвуют в зеркальном отображении баз данных, не обязательно должны находиться друг рядом с другом;
- в кластере серверы работают с одной физической базой данных, которая находится на внешнем устройстве хранения. Таким образом, выход из строя этого устройства хранения приведет к отказу всего кластера. В зеркальном отображении используются две отдельные копии базы данных, что повышает надежность работы.

По сравнению с доставкой журналов преимущества следующие:

- переключение ролей в случае отказа основного сервера может производиться автоматически (при наличии следящего сервера – **witness server**);
- при применении зеркального отображения баз данных в некоторых ситуациях не потребуется вносить какие-либо изменения в сетевую инфраструктуру или в настройки клиентов. Клиенты при необходимости автоматически переключаются на использование зеркальной копии (это относится только к приложениям, которые используют SQL Native Client или .NET SQL Provider).

Для зеркального отображения можно использовать два режима:

- **Синхронный режим (**synchronous mode**)** – при использовании этого режима транзакция не будет завершена, если она не прошла на обоих серверах. При использовании синхронного режима идентичность данных на двух серверах гарантируется. Однако скорость транзакций при этом может существенно замедлиться.
- **Асинхронный режим (**asynchronous mode, другое название – high-performance mode – «режим высокой производительности») – в этом случае транзакция вначале завершается на первом сервере, а затем информация о ней немедленно передается на второй сервер. Задержек при работе транзакций уже не будет, но данные между серверами могут синхронизироваться с небольшим отставанием.

AlwaysOn availability groups

Технология **AlwaysOn** в MS SQL Server предназначена для повышения доступности баз данных. В AlwaysOn присутствуют элементы сразу нескольких технологий высокой доступности сервера MS SQL Server:

- Кластеризация. AlwaysOn, как и кластер MS SQL Server, имеет общее имя и IP-адрес для экземпляров MS SQL Server, также технология AlwaysOn работает как кластерная служба Windows. Для подключения клиентов к базе данных необходимо указать только один IP-адрес.
- Зеркалирование. В AlwaysOn за основу взята технология зеркалирования MS SQL Server. Присутствует первичная реплика и до 4 (3 в синхронном режиме) вторичных реплик (в MS SQL Server 2014 – до 8 реплик). Возможен асинхронный или синхронный режим работы реплик.

На вторичные реплики при инициализации база данных доставляется, как при log shipping –e, то есть автоматически делается полная резервная копия и журнал транзакций переносятся на общий ресурс и автоматически восстанавливается на вторичной реплике. Вторичные реплики доступны для чтения данных.

Преимущества технологии AlwaysOn:

- поддерживается автоматический переход на вторичные реплики;
- вторичные реплики возможно использовать для чтения данных;
- вторичные реплики возможно использовать для снятия резервных копий без создания нагрузки на первичную реплику;
- поддерживается автоматическое восстановление страниц.

ГЛАВА 6

Администрирование PostgreSQL при работе с «1С:Предприятием»

Основы

Дистрибутив

Дистрибутивы на сайте «1С» (https://releases.1c.ru/project/AddCompPostgre):

- Windows;
- RPM;
- DEB;
- патч для «1С» (если собирать Postgres из исходных текстов).

Структура хранения

В PostgreSQL кластер – это не классический кластер серверов (см. рис. 69).

С точки зрения ОС это каталог со всеми базами и управляющими файлами. Каждая таблица – это файл. При превышении размера 1 Гб создается новый файл.

global – содержит общие для всего кластера таблицы (например, pg_database).

pg_clog – информация о статусах транзакций. На каждую транзакцию выделяется 2 бита (стартовала, завершена, отменена, завершена).

pg_log – логи postgres (по умолчанию).

pg_stat_tmp – временные файлы подсистемы статистики.select * from pg_stat*.

pg_tblspc – символьные ссылки на табличные пространства (tablespaces).

pg_xlog – WAL-файлы.

base – содержит папки с базами данных.

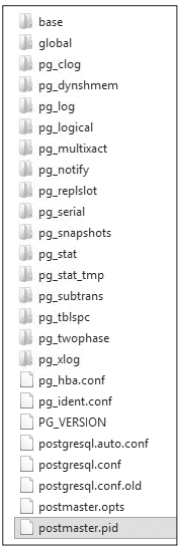


Рисунок 69. Структура хранения данных в кластере

Имя папки – oid из таблицы pg_database.

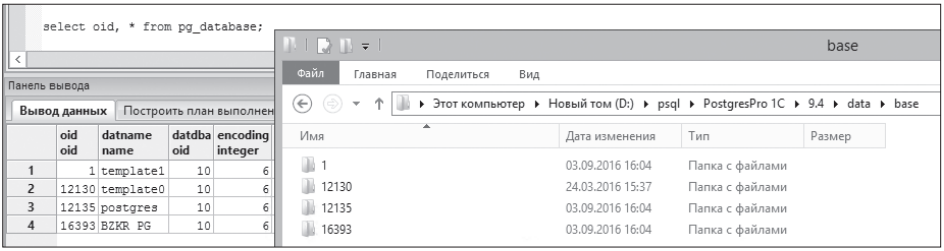


Рисунок 70. Хранение БД

В каждой папке – файлы таблиц. Узнать, в каком файле какая таблица, можно с помощью запроса:

```
SELECT pg_relation_filepath('имя таблицы');
```

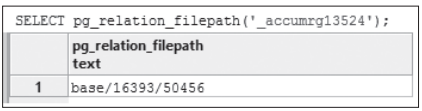


Рисунок 71. Пример получения адреса хранения таблицы

Имя файла меняется после truncate или vacuum.

Файлы *_vm содержат информацию о страницах, строки которых видны всем транзакциям (используется при autovacuum и сканировании индексов).

Файлы *_fm – карта свободного места в таблице. 1 бит на страницу. Показывает, есть ли на странице возможность записать строку (1 – есть, 0 – нет).

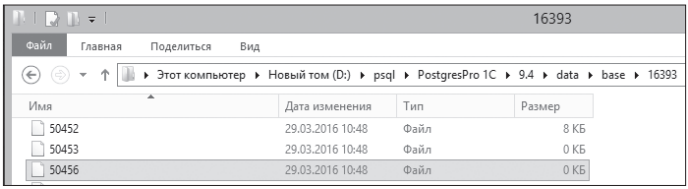


Рисунок 72. Хранение таблиц

Изменить расположение файлов можно через параметры конфигурационного файла postgresql.conf:

- pg_log – параметр log_directory;
- pg_stat_tmp – параметр stats_temp_directory. При высокой нагрузке рекомендуется перенести каталог на RAM-диск;
- pg_xlog – остановить сервер, перенести каталог pg_xlog на другой диск, в папке data создать символическую ссылку pg_xlog на новое расположение;
- temp_tablespaces – параметр позволяет перенести временные таблицы на отдельный диск (лучше SSD). Необходимо создать табличное пространство (см. CREATE TABLESPACE);
- создать табличные пространства v81c_index и v81c_data, это даст возможность расположить индексы и данные на разных дисках.

Распределение каталогов кластера по разным дискам позволяет снизить нагрузку на дисковую подсистему и распределить занимаемое место по разным носителям.

Версии строк

При изменении строки в таблице создается ее новая версия. Версии строк и их видимость транзакциями определяются полями xmin и xmax:

- xmin – номер транзакции, создавшей строку;
- xmax – номер транзакции, удалившей строку.

xmin	xmax	xmin committed	xmin aborted	xmax committed	xmax aborted	heap hot upd	heap only tuple	ctid	данные
100	0							(0,1)	42,AAA

Рисунок 73. Версии строк

Транзакции

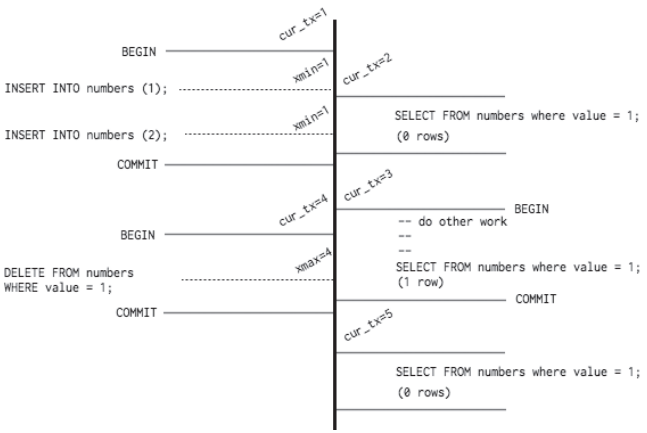


Рисунок 74. Логика работы транзакций в PostgreSQL

- Счетчик транзакций (232).
- Пространство номеров транзакций закольцовано.
- Младшей считается транзакция, которая отстоит от другой меньше чем на «полкруга».
- В ходе очистки (VACUUM) таблиц выполняется так называемая «заморозка».
- Строки, чей xmin гораздо меньше самой старой из запущенных транзакций и «возраст» превышает vacuum_freeze_min_age, «замораживаются».

Рисунок 75. Счетчик транзакций

VACUUM

- Фоновый процесс.
- Удаляет строки старых версий.
- VACUUM – не требует монопольного доступа, но не освобождает место на диске.
- VACUUM ANALYZE – после удаления строк обновляет статистику.

- VACUUM FULL – переносит живые данные в новый файл, а старый удаляет. Требует монопольного доступа.
- Рекомендуется выполнять минимум 1 раз в сутки.

Основные процессы PostgreSQL

postmaster (сейчас **postgres**) –

главный процесс сервера PostgreSQL. Он является родительским для всех остальных процессов.

checkpointer –

checkpointer process сбрасывает на диск «грязные» страницы с данными.

bgwriter –

background writer (bgwriter). Этот процесс срабатывает каждые bgwriter_delay секунд, ищет в shared buffer pool измененные страницы, извлекает bgwriter_lru_maxpages и записывает их на диск.

walwriter –

wal writer process записывает на диск и синхронизирует WriteAheadLog (WAL) после фиксации (COMMIT) транзакции.

autovacuum –

autovacuum launcher process автоматизирует запуск команд VACUUM и ANALYZE. Он проверяет таблицы и индексы на большое количество вставленных или удаленных tuples.

pgstat –

stats collector process собирает статистику по кластеру. Собранная статистика доступна в таблицах и представлениях, начинающихся с pg_stat*. В том числе она помогает планировщику выбирать наиболее эффективный план запроса.

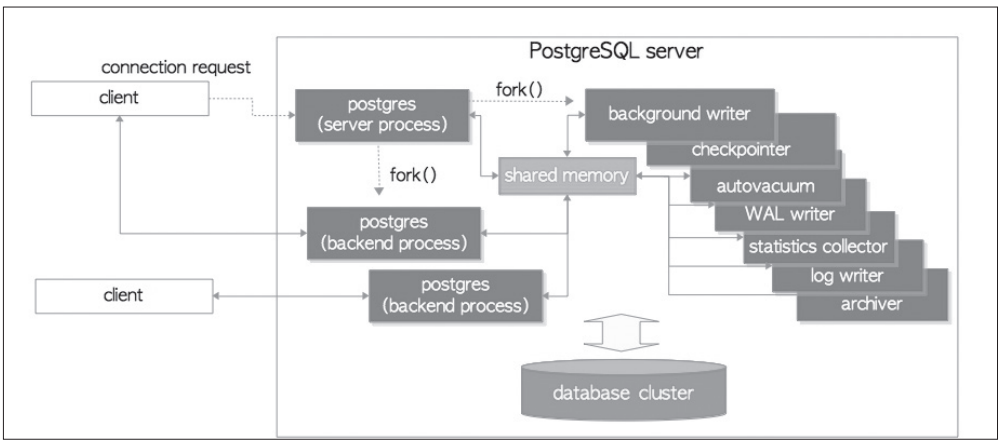


Рисунок 76. Структура процессов PostgreSQL

ИД п...	Имя	Командная строка
14124	postgres.exe	"D:\psql\PostgresPro 1C\9.4\bin\postgres.exe" -D "D:\psql\PostgresPro 1C\9.4\data"
7080	postgres.exe	"D:/psql/PostgresPro 1C/9.4/bin/postgres.exe" "--forklog" "8388" "8392"
12948	postgres.exe	"D:/psql/PostgresPro 1C/9.4/bin/postgres.exe" "--forkboot" "8264" "-x4"
11852	postgres.exe	"D:/psql/PostgresPro 1C/9.4/bin/postgres.exe" "--forkboot" "8276" "-x3"
12864	postgres.exe	"D:/psql/PostgresPro 1C/9.4/bin/postgres.exe" "--forkboot" "8280" "-x5"
10056	postgres.exe	"D:/psql/PostgresPro 1C/9.4/bin/postgres.exe" "--forkavlauncher" "8252"
6696	postgres.exe	"D:/psql/PostgresPro 1C/9.4/bin/postgres.exe" "--forkcol" "8244"
5948	postgres.exe	"D:/psql/PostgresPro 1C/9.4/bin/postgres.exe" "--forkbackend" "8208"
4944	postgres.exe	"D:/psql/PostgresPro 1C/9.4/bin/postgres.exe" "--forkbackend" "8228"
14340	postgres.exe	"D:/psql/PostgresPro 1C/9.4/bin/postgres.exe" "--forkbackend" "8232"
10828	postgres.exe	"D:/psql/PostgresPro 1C/9.4/bin/postgres.exe" "--forkbackend" "7568"

Рисунок 77. Процессы PostgreSQL во время выполнения

Параметры подключения по умолчанию определяют переменные окружения:

- PGDATA – расположение каталога data;
- PGDATABASE – база данных по умолчанию;
- PGLOCALEDIR – язык сообщений;
- PGPORT – порт;
- PGUSER – пользователь.

Подробности здесь: https://www.postgresql.org/docs/9.3/static/libpq-envars.html

Расширения

- Находятся в каталоге ../share/extension.
- Установить расширение:
  CREATE EXTENSION [IF NOT EXISTS] extension_name.
- Можно написать свое расширение.
- pageinspect – низкоуровневый доступ к страницам базы данных.
- pgrowlocks – информация о блокировках.

Редактор SQL | Графический конструктор запросов

Предыдущие запросы

```
CREATE EXTENSION pg_buffercache;
SELECT * FROM  pg_buffercache;
```

Панель вывода

Вывод данных | Построить план выполнения | Сообщения | История

	bufferid integer	relfilenode oid	reltablespace oid	reldatabase oid	relforknumber smallint	relblocknumber bigint	isdirty boolean	usagecount smallint
1	1	12131	1664	0	0	0	f	5
2	2	11877	1664	0	0	0	f	5
3	3	11896	1663	16393	0	0	f	5
4	4	11896	1663	16393	0	1	f	5
5	5	11896	1663	16393	0	2	f	5
6	6	11896	1663	16393	0	3	f	5
7	7	11896	1663	16393	0	4	f	5
8	8	11896	1663	16393	0	5	f	5

Рисунок 78. Пример работы с расширением

Логирование

Настройки логов находятся в postgresql.conf:

- log_directory – каталог логов;
- log_filename – формат имен файлов;
- log_min_duration_statement – минимальная длительность запросов;
- log_checkpoints – логировать checkpoint (on/off);
- log_duration – логировать запросы, время которых превышает log_min_duration_statement (on/off);
- log_lock_waits – логировать блокировки, время которых больше deadlock_timeout (on/off);
- log_statement – виды логируемых запросов (ddl, mod, all);
- log_temp_files – логировать работу с временными файлами >= N Кб.

Удобным инструментом анализа логов является **pgbadger**:

- Скачать: wget https://github.com/downloads/dalibo/pgbadger/pgbadger-2.1.tar.gz
- Распаковать:

```
tar -xzvf pgbadger-2.1.tar.gz
```

- Настроить формат логов:

```
log_line_prefix = '%t [%p]: [%l-1]'
```

- Настроить события.
- Применить настройки:

```
sudo service postgresql reload
```

- Запустить:

```
perl pgbadger -a 1 /var/log/postgresql/postgresql-main.log
```

Необходимо добавить настройки в pg_conf для работы **pgbadger**:

```
log_min_duration_statement = 0
log_line_prefix = '%t [%p]: [%l-1] ' или '%t [%p]: [%l-1] user=%u,db=%d,client=%h '
log_checkpoints = on
log_connections = on
log_disconnections = on
log_lock_waits = on
log_temp_files = 0
log_autovacuum_min_duration = 0
lc_messages='C'
log_duration = on
log_statement = all
log_destination = stderr
```

Настройки приведены согласно https://github.com/dalibo/pgbadger/blob/master/README

ВАЖНО!

При большой нагрузке **pgbadger** может влиять на производительность по причине большого потока записи на диск. Лучше вынести на отдельный шпиндель.

Собранная статистика может быть выведена для анализа в виде html-страницы out.html.

Что можно проанализировать:

- Общую статистику:
  - по SQL-трафику;
  - по SELECT;
  - по INSERT/UPDATE/DELETE;
  - длительности запросов;
  - подготовительных запросов (INSERT INTO PRODUCT (name, price) VALUES (?, ?));
  - общей активности.
- Информацию:
  - по соединениям;
  - сессиям;
  - checkpoint;
  - использованию временных файлов;
  - длительным запросам;
  - наиболее частым запросам;
  - удаленным данным;
  - ожиданиям на блокировках;
  - и т. д.

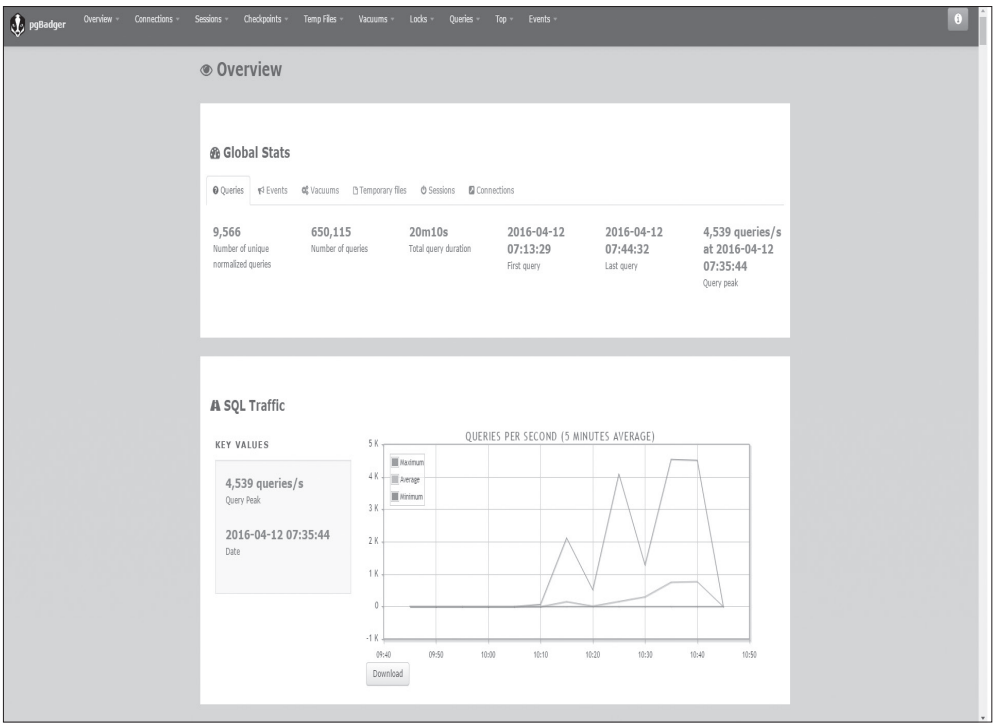


Рисунок 79. Общий вид out.html

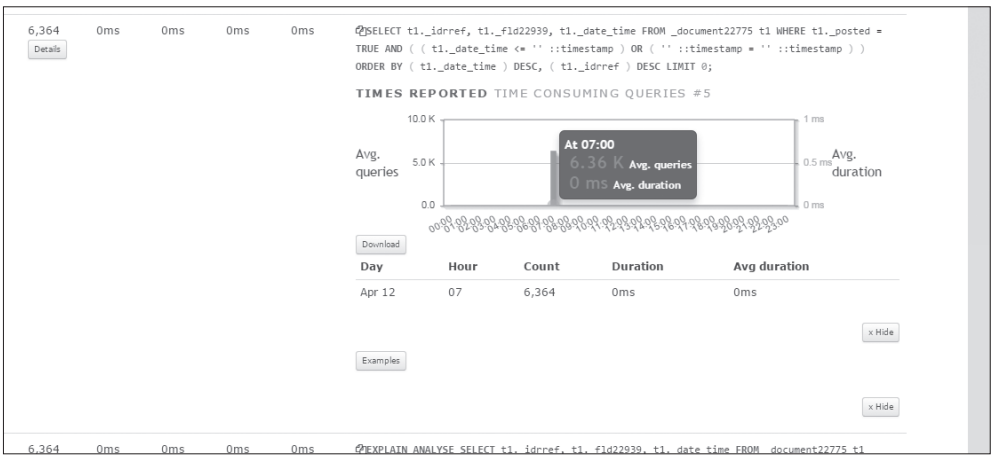


Рисунок 80. Информация по длительным запросам

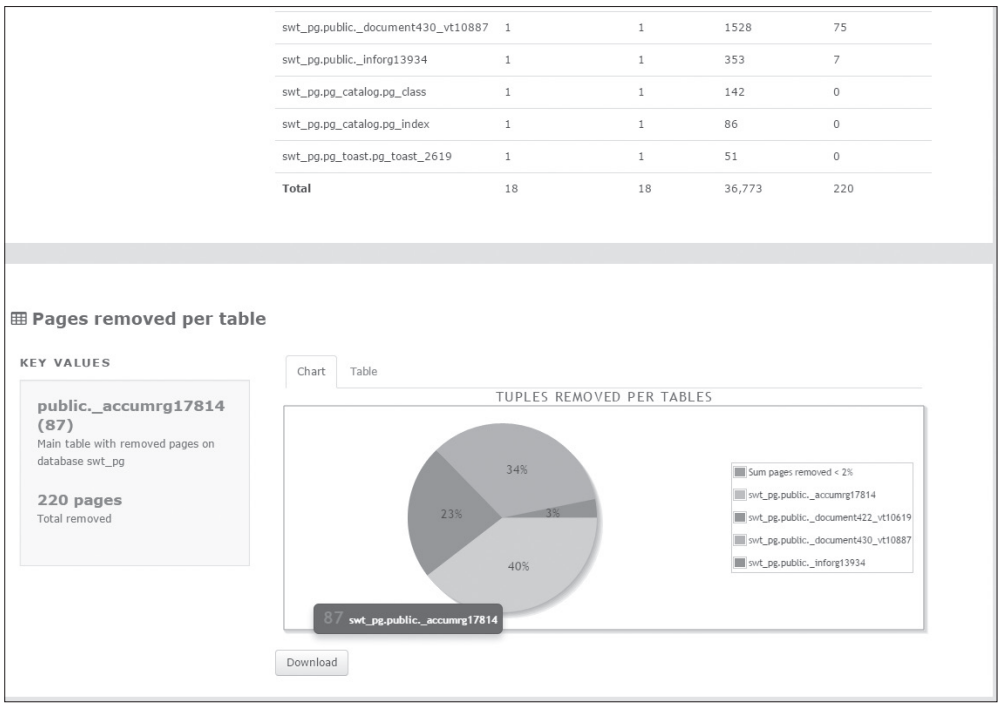


Рисунок 81. Информация по удаленным tumple

Настройки PostgreSQL для работы с «1С:Предприятием»

Основные параметры postgresql.conf

- shared_buffers – размер буферного пула;
- temp_buffers – количество страниц в памяти для временных таблиц;
- effective_cache_size – позволяет оценить, в памяти данные или на диске;
- work_mem – рекомендованный лимит памяти для одного запроса;
- autovacuum on/off – надо on;
- fsync – ждать от контроллера подтверждения записи;
- max_connections – количество одновременных соединений;
- synchronous_commit – ждать ответа от диска при commit;
- random_page_cost – стоимость случайного доступа к диску;
- max_files_per_process – максимальное число открытых файлов.

Общие положения

Далее описывается настройка PostgreSQL версий 9.2–9.4 на максимальную производительность для платформы «1С:Предприятие». Предполагается, что сервер, используемый для PostgreSQL, является достаточно производительным и имеет приблизительно:

- 4 – 512 Gb RAM;
- 2 – 256 CPU cores;
- RAID 0-1 или SSD.

Данный документ подразумевает хотя бы поверхностное знакомство с архитектурой PgSQL. Приведенные в документе параметры являются приблизительными – стартовыми для тонкой настройки.

Настройки сервера для PostgreSQL

- Рекомендуется отключать HyperThreading. Для программ типа систем управления базами данных от него скорее вред, чем польза.
- Также рекомендуется отключать Energy Saving, поскольку в противном случае могут непредсказуемо возрастать задержки ответов БД.
- Запретить своппинг разделяемой памяти SYSV/posix (FreeBSD: kern.ipc. shm_use_phys=1).

Обозначения

- RAM – объем оперативной памяти сервера. Если сервер используется не только для PostgreSQL, то надо уменьшить эту величину на объем занятой памяти;
- NCores – суммарное число ядер на всех CPU-серверах;
- max_connections – максимальное число коннектов (или сессий) к PgSQL. Задается в конфигурационном файле;
- WAL (Write Ahead Log) – опережающий лог действий с таблицами и индексами. Основная задача – целостность и отказоустойчивость базы данных при одновременном росте производительности;
- checkpoint – точка восстановления база данных. Все WAL-данные, записанные до checkpoint, становятся не нужны;
- X..Y – диапазон значений от X до Y включительно.

Параметры производительности

```
shared_buffers = RAM/4
```

Количество памяти, выделенной PgSQL для совместного кеша страниц. Эта память разделяется между всеми процессами PgSQL.

```
temp_buffers = 256MB
```

Максимальное количество страниц для временных таблиц. То есть это верхний лимит размера временных таблиц в каждой сессии.

```
work_mem = RAM/32..64 или 32MB..128MB
```

Лимит памяти для обработки одного запроса. Эта память индивидуальна для каждой сессии. Теоретически максимально потребная память равна max_connections * work_mem, но на практике такого не встречается, потому что большая часть сессий почти всегда висит в ожидании. Это рекомендательное значение используется оптимайзером: он пытается предугадать размер необходимой памяти для запроса и, если это значение больше work_mem, указывает экзекьютору сразу создать временную таблицу. Work_mem не является в полном смысле лимитом: оптимайзер может и промахнуться, тогда запрос займет больше памяти (возможно, в разы). Это значение можно уменьшать, следя за количеством создаваемых временных файлов.

```
Select sum(temp_files) as temp_files, pg_size_pretty(sum(temp_bytes)) as temp_size from pg_stat_database;
maintenance_work_mem = RAM/16..32 или work_mem * 4 или 256MB.4GB
```

Лимит памяти для обслуживающих задач (например, вакуум, автовакуум или создание индексов).

```
effective_cache_size = RAM – shared_buffers
```

Оценка размера кеша файловой системы. Увеличение параметра увеличивает склонность системы выбирать IndexScan планы. И это хорошо.

```
effective_io_concurrency = 2
```

Оценочное значение одновременных запросов к дисковой системе, которые она может обслужить единовременно. Для одиночного диска = 1, для RAID – 2 или больше.

```
random_page_cost = 1.5-2.0 для RAID, 1.1-1.3 для SSD
```

Стоимость чтения рандомной страницы (по умолчанию 4). Чем меньше seek time дисковой системы, тем меньше должен быть этот параметр (но > 1.0). Излишне большое значение параметра увеличивает склонность PgSQL к выбору планов со сканированием всей таблицы (PgSQL считает, что дешевле последовательно читать всю таблицу, чем рандомно индекс). И это плохо.

```
autovacuum = on
```

Включение автовакуума. **Не следует выключать его!**

```
autovacuum_max_workers = NCores/4..2 но не меньше 4
```

Количество процессов автовакуума. Общее правило: чем больше write-запросов, тем больше процессов. На read-only базе данных достаточно одного процесса.

```
autovacuum_naptime = 20s
```

Время сна процесса автовакуума. Слишком большая величина будет приводить к тому, что таблицы не будут успевать вакуумиться, и как следствие вырастут bloat и размеры таблиц и индексов. Малая величина приведет к бесполезному нагреванию.

```
bgwriter_delay = 20ms
```

Время сна между циклами записи на диск фонового процесса записи. Данный процесс ответственен за синхронизацию страниц, расположенных в shared_buffers с диском. Слишком большое значение этого параметра приведет к возрастанию нагрузки на checkpoint-процесс и процессы, обслуживающие сессии (backend). Малое значение приведет к полной загрузке одного из ядер.

```
bgwriter_lru_multiplier = 4.0
bgwriter_lru_maxpages = 400
```

Параметры, управляющие интенсивностью фонового процесса записи. За один цикл bgwriter записывает количество данных не большее, чем было записано в прошлый цикл, умноженное на bgwriter_lru_multiplier, но не превышающее при этом bgwriter_lru_maxpages.

```
synchronous_commit = off
```

Выключение синхронизации с диском в момент коммита. Создает риск потери последних нескольких транзакций (в течение 0,5–1 секунды), но гарантирует целостность базы данных, в цепочке коммитов гарантированно отсутствуют пропуски. Но значительно увеличивает производительность.

```
checkpoint_segments = 32..256   < 9.5
```

Максимальное количество сегментов WAL между checkpoint. Слишком частые checkpoint приводят к значительной нагрузке на дисковую подсистему по записи. Каждый сегмент имеет размер 16 Мб.

```
checkpoint_completion_target = 0.5..0.9
```

Степень «размазывания» checkpoint. Скорость записи во время checkpoint регулируется так, чтобы время checkpoint было равно времени, прошедшему с прошлого checkpoint, умноженному на checkpoint_completion_target.

```
min_wal_size = 512MB .. 4G        > =9.5
max_wal_size = 2 * min_wal_size    > =9.5
```

Минимальный и максимальный объем WAL-файлов. Аналогично checkpoint_segments.

```
ssl = off
```

Выключение шифрования. Для защищенных ЦОД шифрование бессмысленно, но приводит к увеличению загрузки CPU.

```
fsync = on
```

Выключение параметра приводит к росту производительности, но появляется значительный риск потери всех данных при внезапном выключении питания. Внимание: если RAID имеет кеш и находится в режиме write-back, нужно проверить наличие и функциональность батарейки кеша RAID-контроллера! Иначе данные, записанные в кеш RAID, могут быть потеряны при выключении питания, и как следствие PgSQL не гарантирует целостность данных.

```
commit_delay = 1000
commit_siblings = 5
```

Групповой коммит нескольких транзакций. Имеет смысл включать, если темп транзакций превосходит 1000 TPS. Иначе эффекта не имеет.

```
temp_tablespaces = 'NAME_OF_TABLESPACE'
```

Дисковое пространство для временных таблиц/индексов. Помещение временных таблиц/индексов на отдельные диски может увеличить производительность. Предварительно надо создать tablespace командой CREATE TABLESPACE. Если характеристики дисков отличаются от основных дисков, то следует в команде указать соответствующий random_page_cost.

Подробнее см. статью https://www.postgresql.org/docs/9.4/static/sql-createtablespace.html

```
row_security = off            >= 9.5
```

Отключение контроля разрешения уровня записи.

```
pg_stat_temp
```

Необходимо изменять значение по умолчанию пути к директории pg_stat_temp. Причина в интенсивном изменении файлов в этой директории, что создает значительную нагрузку на дисковую подсистему, что, в свою очередь, может значительно замедлить выполнение операций. Директорию рекомендуется размещать в RAM-диске (для Windows) или tmpfs (для linux).

```
max_files_per_process = 1000 (default)
```

Максимальное количество открытых файлов на один процесс PostreSQL. Один файл – это как минимум либо индекс, либо таблица, но таблица может состоять из нескольких файлов. Если PostgreSQL упирается в этот лимит, он начинает открывать/закрывать файлы, что может сказываться на производительности. Диагностировать проблему под Linux можно с помощью команды lsof.

Параметры для платформы «1С:Предприятие»

```
standard_conforming_strings = off
```

Разрешить использовать символ «\» для экранирования.

```
escape_string_warning = off
```

Не выдавать предупреждение об использовании символа «\» для экранирования.

```
max_locks_per_transaction = 256
```

Максимальное число блокировок индексов/таблиц в одной транзакции.

```
max_connections = 500..1000
```

Количество одновременных коннектов/сессий.

Online_analyse

Online_analyse – это дополнительная библиотека, предназначенная для автоматического пересчета статистики об изменениях таблиц. Автоматический анализ статистики включен по умолчанию только для временных таблиц. Использование online_analyse увеличивает нагрузку на CPU.

```
online_analyze.enable         = on    # включить
online_analyze.scale_factor   = 0.1   # коэффициент кол-ва измененных строк для старта on-line analyze
online_analyze.threshold      = 50    # количество измененных строк для старта on-line analyze
online_analyze.min_interval   = 10000 # минимальный интервал (милисекунд) для старта on-line analyze
online_analyze.table_type     = "all" # типы таблиц для анализа: all, persistent, temporary, none
online_analyze.exclude_tables = ""    # не анализируемые таблицы
online_analyze.include_tables = ""    # анализировать только таблицы из списка (замещает exclude_tables)
```

Рисунок 82. Настройка online_analyse в postgresql.conf

Подробности – на https://kb.1c.ru/articleView.jsp?id=91.

Расследование проблем

Лог медленных запросов

Настроить журнал:

- log_duration = on – включает логирование запросов;
- log_min_duration_statement = 1 – устанавливает минимальное время выполнения запроса (в миллисекундах), который попадает в лог.

Перезапустить сервер:

```
sudo service postgresql restart
```

Смотреть выполняющиеся запросы:

```
tail -f /var/log/postgresql/postgresql-9.log
```

Рекомендуется настраивать на рабочих серверах.

Выполняющиеся сейчас запросы

```
SELECT
st.waiting,
st.backend_xid,
st.backend_xmin,
st.state,
st.query,
st.*
FROM
pg_stat_activity as st
WHERE st.state = ‘active’
```

Поиск медленных запросов

```
SELECT
st.waiting,
st.pid,
st.state,
st.query
FROM
pg_stat_activity as st
WHERE st.state = ‘active’
```

pid integer	query text	waiting boolean	state text
9040		f	active
3496	SELECT Creation,Modified,Attributes,DataSize,BinaryData FROM Config WHERE FileName = $1 ORDER BY PartNo	f	active

Рисунок 83. Результат поиска медленных запросов

Если не указывать условие state = 'active', то будут выведены последние запросы по всем соединениям.

Другой вариант:

```
SELECT
    s.usename
    ,s.pid AS process_id
    ,s.waiting
    ,age(now(), s.query_start)
    ,query
    ,s.state
  FROM pg_stat_activity s
  WHERE 1=1AND s.xact_start IS NOT NULL
```

Ожидание на блокировках

```
SELECT
blocked_locks.pid,
blocking_activity.backend_xid,
blocked_activity.query,
blocked_activity.waiting,
blocking_activity.query as executing_query,
blocking_activity.*
FROM  pg_catalog.pg_locks         blocked_locks
JOIN pg_catalog.pg_stat_activity blocked_activity  ON blocked_activity.pid = blocked_locks.pid
JOIN pg_catalog.pg_locks         blocking_locks
ON blocking_locks.locktype = blocked_locks.locktype
AND blocking_locks.DATABASE IS NOT DISTINCT FROM blocked_locks.DATABASE
AND blocking_locks.relation IS NOT DISTINCT FROM blocked_locks.relation
AND blocking_locks.page IS NOT DISTINCT FROM blocked_locks.page
AND blocking_locks.tuple IS NOT DISTINCT FROM blocked_locks.tuple
AND blocking_locks.virtualxid IS NOT DISTINCT FROM blocked_locks.virtualxid
AND blocking_locks.transactionid IS NOT DISTINCT FROM blocked_locks.transactionid
AND blocking_locks.classid IS NOT DISTINCT FROM blocked_locks.classid
AND blocking_locks.objid IS NOT DISTINCT FROM blocked_locks.objid
AND blocking_locks.objsubid IS NOT DISTINCT FROM blocked_locks.objsubid
AND blocking_locks.pid != blocked_locks.pid
JOIN pg_catalog.pg_stat_activity blocking_activity ON blocking_activity.pid = blocking_locks.pid
WHERE NOT blocked_locks.GRANTED;

SELECT
  COALESCE(l1.relation::regclass::text,l1.locktype) as locked_item,
  w.waiting, w.query as waiting_query,
  l1.mode as waiting_mode,
  (select now()  – xact_start as waiting_xact_duration from pg_stat_activity where pid = w.pid),
  (select now()  – query_start as waiting_query_duration from pg_stat_activity where pid = w.pid),
  w.pid as waiting_pid, w.usename as waiting_user, w.state as waiting_state,
  l.waiting, l.query as locking_query, l2.mode as locking_mode,
  (select now()  – xact_start as locking_xact_duration from pg_stat_activity where pid = l.pid),
  (select now()  – query_start as locking_query_duration from pg_stat_activity where pid = l.pid),
  l.pid as locking_pid, l.usename as locking_user, l.state as locking_state
FROM pg_stat_activity w
JOIN pg_locks l1 ON w.pid = l1.pid AND NOT l1.granted
JOIN pg_locks l2 ON (l1.transactionid = l2.transactionid AND l1.pid != l2.pid)
    OR (l1.database = l2.database AND l1.relation = l2.relation and l1.pid != l2.pid)
JOIN pg_stat_activity l ON l2.pid = l.pid
WHERE w.waiting
ORDER BY l.query_start,w.query_start;

select
   locks.*
from
   pg_class as class
   join pg_locks as locks
   on class.oid = locks.relation
where
   class.relname = ‘<имя таблицы>’;
```

Резервное копирование и восстановление

Возможные подходы:

- Дамп SQL:
  - копия отдельного объекта или базы;
  - восстановление на другой версии и архитектуре;
  - медленно.
- Резервное копирование на уровне файловой системы:
  - быстро;
  - на любой момент времени;
  - только весь кластер (отдельную базу нельзя).
- Непрерывное архивирование.

Дамп SQL

Создание дампа:

```
pg_dump имя_БД > файл_дампа
```

Восстановление из дампа:

```
psql имя_БД < файл_дампа
```

Можно даже восстановиться из базы другого сервера:

```
pg_dump -h сервер1 имя_БД | psql -h сервер2 имя_БД
```

Или сжимать дампы:

```
pg_dump имя_БД | gzip > имя_файла.gz
```

И разжимать:

```
gunzip -c имя_файла.gz | psql имя_БД
```

pg_dump – логический бэкап:

- Допустимые форматы:
  - psql – в виде текста sql (восстановление через psql);
  - custom – вместе с данными пишет заголовки объектов (восстановление через pg_restore);
  - directory – аналогичен custom, но выгружает не в один файл, а в каталог.
- 10 Тб (очень «грустный» процесс).
- Активность пользователей допустима, но нежелательна.
- Не выгружает статистику, надо запускать ANALIZE.
- Может выгружать в несколько потоков (при формате directory).
- Делается snapshot на момент начала бэкапа, и база выгружается в консистентном состоянии.

- Если бэкап шел неделю, то будет получена копия недельной давности.

pg_dumpall – полный бэкап кластера:

- Пользователи.
- Базы данных.
- Табличные пространства.
- Работает только в SQL-формате.

pg_restore – восстановление:

- Можно восстановить как одну таблицу, так и базу на момент начала pg_backup.
- -1 флаг выполняет весь restore в одной транзакции.
- Можно выключить fsync:
  - не забыть включить обратно;
  - pg_dumpall -g – global objects only;
  - будут включены users, groups, tablespaces;
  - в обычный бэкап не включены.

Физические бэкапы

Особенности:

- Сильно быстрее.
- Привязаны к архитектуре, компиляции, флагам компиляции и т. д.
- Привязаны к той же версии PG.
- Поддерживается бэкап всего кластера.

  Существуют следующие способы:

  Offline backup:
  - Stop postgres;
  - Backup files;
  - Start postgres;
  - работает, если нет необходимости в высоком uptime.

  System snapshot:
  - все табличные пространства;
  - не забыть pg_xlog и pg_clog.

  Online backups:
- **pg_basebackup** – физический бэкап на уровне файловой системы:
  - Нужен транзакционный лог.
  - Настроить уровень WAL:
    - wal_keep_segments – нужно сделать больше, так как, пока копируются файлы данных, можно потерять сегменты журналов;

- wal_level = acrhive (или выше). Потребует рестарта, поэтому надо озаботиться заранее;
- max_wal_senders >= 1.

- Запускать от суперпользователя с ролью REPLICATION.
- В pg_hba.conf дать разрешение на базу данных replication.
- Используется один backend:
  - 100 % одно ядро.
- Профиль I/O сильно меняется в сторону сканов на дисковой подсистеме, поэтому лучше не делать backup на ту же подсистему, на которой находятся база.
- Лучше увеличивать сжатие:
  - обычно имеются дополнительные ресурсы по CPU и нет дополнительных ресурсов по I/O.

- **pg_start_backup**:
  - Убедиться, что архивирование WAL включено и работает.
  - Подключиться к базе данных как суперпользователь и выполнить команду SELECT pg_start_backup('метка'), где метка – это любая строка, которую вы хотите использовать как уникальный идентификатор данной операции резервного копирования (хорошей практикой является использование полного пути, куда планируется разместить файл дампа с резервной копией). Pg_start_backup создает в каталоге кластера файл с меткой резервного копирования, называемый backup_label, содержащий информацию о вашей резервной копии, включая время начала и строку с меткой. Для выполнения этой команды неважно, к какой базе данных в кластере вы подключаетесь. Можно игнорировать результат, возвращаемый данной функцией, но если он говорит об ошибке, то нужно разобраться с ней, перед тем как продолжить. По умолчанию выполнение pg_start_backup может занять длительное время. Так происходит, потому что функция выполняет контрольную точку (checkpoint) и операции ввода/вывода, требуемые для выполнения этого действия, будут занимать значительное временя – по умолчанию половину вашего интервала между контрольными точками (см. конфигурационный параметр checkpoint_completion_target).
  - Выполнить резервное копирование, используя подходящий инструмент резервного копирования для файловой системы, такой как tar или cpio (не pg_dump или pg_dumpall).
- **pg_stop_backup**:
  - Снова подключиться к базе данных как суперпользователь и выполнить команду SELECT pg_stop_backup(). Она завершит работу режима резервного копирования и выполнит автоматическое переключение на следующий WAL-сегмент. Причина переключения состоит в выравнивании последнего файла с WAL-сегментом, работавшим во время выполнения резервного копирования, с целью подготовки к архивации.
  - Как только заархивируются файлы с WAL-сегментами, появившиеся во время резервного копирования, процесс завершен.

- Файл, идентифицируемый функцией pg_stop_backup, является последним сегментом, который требуется для полного списка файлов резервной копии.
- Если включена переменная archive_mode, функция pg_stop_backup не завершится, пока последний сегмент не будет заархивирован.
- Архивирование этих файлов происходит автоматически, поскольку уже настроена команда archive_command.

```
# - Archiving -

#archive_mode = off     # allows archiving to be done
                # (change requires restart)
#archive_command = ''        # command to use to archive a logfile segment
                # placeholders: %p = path of file to archive
                #               %f = file name only
                # e.g. 'test ! -f /mnt/server/archivedir/%f && cp %p /mnt/server/archivedir/%f'
#archive_timeout = 0        # force a logfile segment switch after this
                # number of seconds; 0 disables
```

Рисунок 84. Настройки архивирования

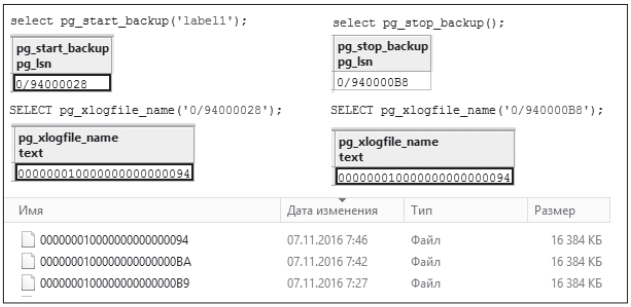


Рисунок 85. Физические бэкапы

Непрерывная архивация

Архивация

- Идет постоянное копирование WAL-файлов.
- Можно восстановиться на любой момент времени от полного бэкапа.
- Запускается специальный процесс archiver.
- Необходимы настройки:
  - archive_mode = on;
  - archive_command – текст shell;
  - archive_timeout – время автоматического создания нового сегмента wal.

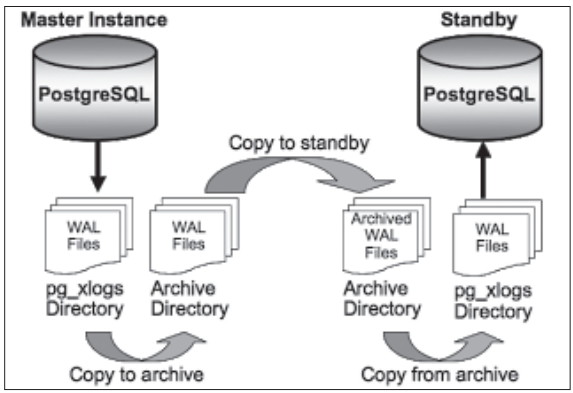


Рисунок 86. Непрерывная архивация

Восстановление

- Восстановить копию кластера.
- Удалить все файлы из pg_xlog, которые оказались в копии.
- Скопировать все WAL, которые создались между pg_start_backup и pg_stop_backup.
- В каталоге кластера создать файл recovery.conf (пример в ../share/recovery.conf.sample).
- В файле recovery.conf указать standby_mode = on и задать restore_command.
- Сервер будет постоянно получать новые WAL-файлы и восстанавливать их.
- При необходимости выполняется команда pg_ctl promote на резервном сервере или «подкладывается» файл, указанный в параметре trigger_file.

Дополнительные источники информации

- http://explain.depesz.com/ – удобный просмотр планов запросов;
- https://kb.1c.ru/articleView.jsp?id=91;
- https://kb.1c.ru/articleView.jsp?id=23.

ГЛАВА 7

Особенности настройки веб-серверов

Сравнение

На рисунке 87 приведен сравнительный анализ популярных веб-серверов. Актуальную статистику можно увидеть на https://w3techs.com/technologies/cross/web_server/ranking.

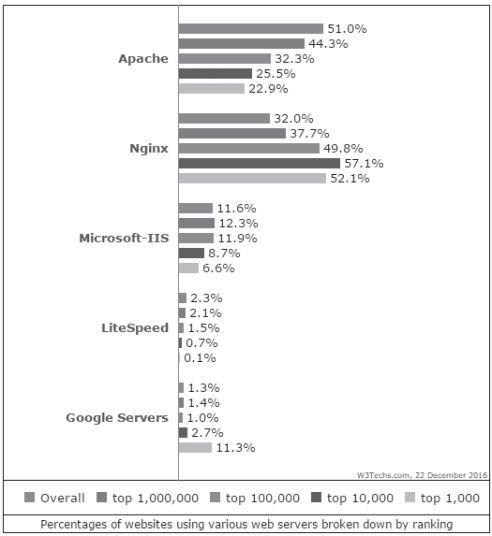


Рисунок 87. Статистика использования веб-серверов

При этом платформа «1С:Предприятие» поддерживает возможность размещения веб-публикаций информационных баз только для веб-серверов Apache и Microsoft IIS.

Особенности настройки Nginx

Веб-сервер Nginx в технологии 1cFresh используется обычно в качестве front-сервера. Внутренняя архитектура позволяет Nginx не проседать даже под очень высокими нагрузками, выдерживая свыше 10 000 одновременных соединений.

Для настройки NGINX используются конфигурационные файлы, которые помещаются в каталоги /etc/nginx и /etc/nginx/conf.d сервера при настройке front-сервера по технологии 1cFresh. Эти настройки будут обеспечивать возможность использования в сервисе:

- шлюза приложений;
- протокола HTTPS и OpenID-аутентификации;
- сайта и форума, с использованием единой точки входа;
- страницы недоступности.

Формат конфигурационных файлов Nginx описан в документации по Nginx (http://nginx.org/en/docs/, http://nginx.org/ru/docs/), также можно использовать книги:

- Nedelcu C. Nginx HTTP Server / 2-nd Edition, Packt Publishing, 2013.
- Aivaliotis D. Mastering Nginx / Packt Publishing, 2013.

Ниже описаны основные конфигурационные файлы Nginx и приведены примеры их настройки для использования в технологии 1cFresh.

ВАЖНО!

> Полное описание конфигурационных файлов и пояснения их настроек приведены в «Технологии публикации решений 1cFresh».

Особенности настройки IIS

Веб-сервер IIS чаще всего используется в технологии 1cFresh в качестве сервера для публикации информационных баз. При этом есть опыт его использования и в качестве front-сервера.

При использовании IIS в качестве **сервера для публикации информационных баз** в технологии 1cFresh стоит учитывать ряд особенностей настройки.

Добавление пула приложений

Для публикаций информационных баз «1С:Предприятия 8» целесообразно создать в IIS отдельный пул приложений.

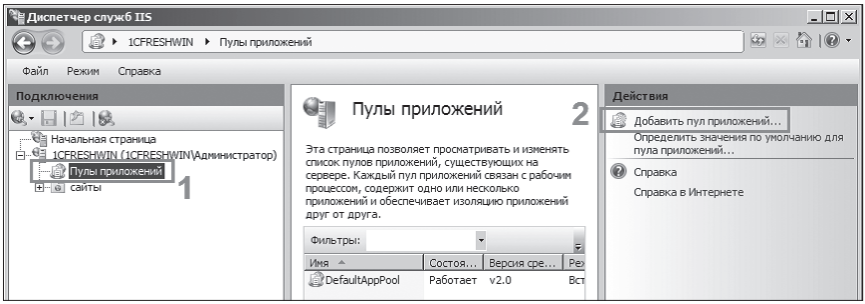


Рисунок 88. Добавление пула приложений IIS

В окне Добавление пула приложений задать параметры пула:

- Версии среды .NET Framework – Без управляемого кода.
- Режим управляемого конвейера – Встроенный.

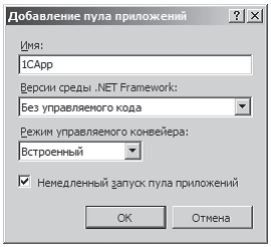


Рисунок 89. Параметры добавляемого пула приложений IIS

Добавление веб-сайтов

Для публикаций информационных баз «1С:Предприятия 8» создать в IIS два веб-сайта:

- 1cFresh_ext – для внешних публикаций информационных баз «1С:Предприятия 8». Физический путь для файлов этого веб-сайта пусть будет C:\inetpub\wwwroot\1cFresh_ext.
- 1cFresh_int – для внутренних публикаций информационных баз «1С:Предприятия 8». Физический путь для файлов этого веб-сайта пусть будет C:\inetpub\wwwroot\1cFresh_int.

Сайтам 1cFresh_ext и 1cFresh_int назначить пул приложений 1cApp. Будем публиковать внешние публикации информационных баз «1С:Предприятия 8» на порту 80, а внутренние – на порту 8888. Чтобы не возникало конфликта сайта 1cFresh_ext с веб-сайтом по умолчанию, изменим свойства веб-сайта по умолчанию, назначив ему порт 8000 (вместо 80).

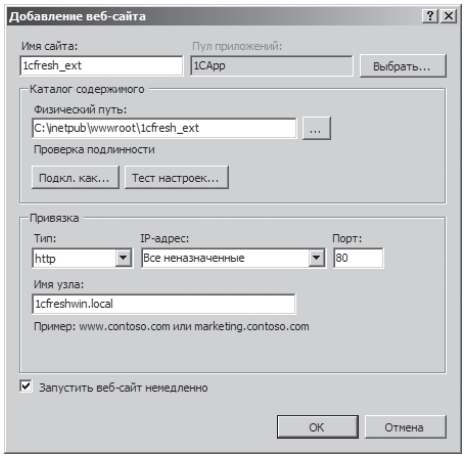


Рисунок 90. Добавление веб-сайта 1cFresh

Задание обработчиков для сайтов внешних и внутренних публикаций

Для корректной работы необходимо указать, что публикации на сайтах 1cFresh_ext, 1cFresh_int должны обрабатываться с помощью модуля расширения веб-сервера платформы «1С:Предприятие 8».

1. В левой части окна «Диспетчера служб IIS» раскрыть основную ветку сервера (в нашем примере это «1CFRESHWIN (1CFRESHWIN\Администратор)»), открыть ветку Сайты.
2. Выбрать сайт внешней публикации (1cFresh_ext).
3. Щелкнуть двойным щелчком мыши по значку Сопоставления обработчиков.
4. Выбрать в правой части окна «Диспетчера служб IIS» пункт Добавить сопоставление сценария.
5. В выведенном окне Добавление сопоставления сценария задать параметры:
   - Путь запроса – * ;
   - Исполняемый файл – C:\Program Files\1cv8\current\bin\wsisapi.dll;
   - Имя – 1C_Enterprise.

   После нажатия на кнопку ОК и появления вопроса «Разрешить данное расширение ISAPI?» ответить Да.

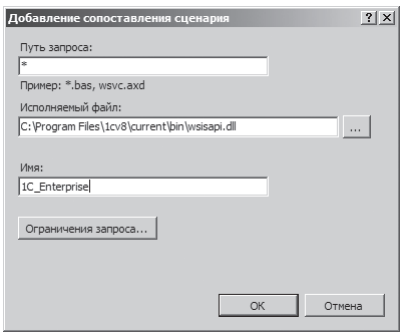


Рисунок 91. Добавление модуля веб-расширения «1С:Предприятия 8» для публикаций веб-сайта

6. В левой части окна «Диспетчера служб IIS» выбрать сайт 1cFresh_int и таким же способом назначить ему модуль веб-расширения «1С:Предприятия 8».

Задание режима обработки ошибок

Для всех сайтов веб-сервера IIS установить режим вывода подробной информации об ошибках:

1. В левой части окна «Диспетчера служб IIS» выбрать весь узел сервера (в нашем примере – пункт 1CFRESHWIN (1CFRESHWIN\Администратор)).
2. Щелкнуть двойным щелчком мыши по значку Страницы ошибок.
3. В правой части окна «Диспетчера служб IIS» выбрать пункт Изменить параметры.
4. В выведенном окне Изменение параметров страниц ошибок выбрать режим Подробные сообщения об ошибках.

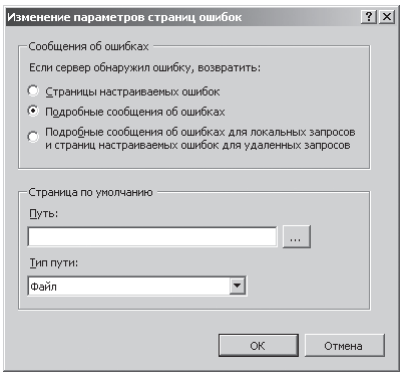


Рисунок 92. Задание режима вывода сообщений об ошибках

Для использования IIS **в качестве front-сервера** актуальна иная настройка режима вывода сообщений об ошибках: «Страницы настраиваемых ошибок». Данная опция применяется для настройки собственной (нетиповой) страницы недоступности.

ВАЖНО!

При использовании IIS в качестве front-сервера невозможна настройка полноценной страницы недоступности, так как IIS не позволяет переопределять через «Страницы настраиваемых ошибок» ошибки с кодами 400, 403.9, 411, 414, 500, 500.11, 500.14, 500.15, 501, 503 и 505 (информация актуальна для IIS 7 https://technet.microsoft.com/en-us/library/cc731570%28v=ws.10%29.aspx).

Особенности настройки Apache

Сервер Apache используется в модели сервиса для публикации информационных баз.

Установка Apache

Для установки веб-сервера Apache нужно войти в консоль сервера 1cFresh.local от имени пользователя root и ввести команду:

```
apt-get -y install apache2
```

Запуск Apache

Чтобы проверить, включен ли автозапуск Apache, нужно ввести команду:

```
update-rc.d apache2 defaults
```

Получить ответ:

```
System start/stop links for /etc/init.d/apache2 already exist,
```

означающий, что автозапуск Apache включен.

Включить службу веб-сервера Apache командой:

```
service apache2 start
```

Чтобы проверить, что служба Apache запущена, ввести команду:

```
service apache2 status
```

Получить ответ следующего вида:

```
Apache2 is running (pid 4887).
```

Это значит, что служба Apache запущена.

Каталоги Apache

Веб-сервер Apache использует следующие каталоги на сервере:

- /etc/apache2, /etc/apache2/conf.d – настройки;
- /usr/lib/apache2 – исполняемые файлы;
- /var/log/apache2 – логи;
- /var/www/ – сайт по умолчанию.

Настройка Apache для нужд сервиса

Публикации для внешних и внутренних баз размещаются в разных каталогах и на разных портах. С этим связаны особенности настройки.

Публикации информационных баз сервиса в описании технологии 1cFresh предлагается делать доступными на следующих портах веб-сайта: внутренние – 8888, внешние – 8889 (через 80-й работает front-сервер).

Для разнесения внешних и внутренних публикаций по разным каталогам и настройки портов требуется произвести следующие манипуляции:

1. Создать подкаталоги 1cFresh_a и 1cFresh_int в каталоге /etc/apache2 сервера 1cFreshL.local:

```
mkdir -p /etc/apache2/1cFresh_a
mkdir -p /etc/apache2/1cFresh_int
```

2. Вызвать на редактирование файл /etc/apache2/ports.conf сервера 1cFreshL.local:

   mcedit /etc/apache2/ports.conf

3. Удалить прежнее содержимое этого файла и вместо него ввести следующие строки:

```
ServerName <имя сайта>
Listen 8889
Listen 8888
Include 1cFresh_a/*.conf
Include 1cFresh_int/*.conf
```

4. Сохранить файл и выйти из редактора.
5. Содержимое каталога /etc/apache2/sites-enabled можно удалить, так как предлагаемый по умолчанию сайт Apache нам не нужен.
6. Для применения настроек перезапустить службу:

```
service apache2 restart
```

Мультипроцессные модули Apache

Apache – очень гибкий http-сервер, способный работать на различных платформах, что накладывает дополнительные требования в части его возможностей выполнять свои функции. Такая кроссплатформенность обеспечивается модульной архитектурой Apache и возможностью расширять необходимую функциональность с помощью подключаемых или компилируемых модулей.

С версии Apache 2.0 сервер поставляется с набором мультипроцессных модулей (МП-модулей). Они позволяют более эффективно настроить сервер для нужд конкретной системы и оптимизировать его работу на определенной платформе. Например, версия Apache для Windows работает более эффективно, благодаря тому что МП-модуль mpm_winnt использует собственные сетевые функции ОС Windows. В сборках Apache для различных операционных систем уже подключен МП-модуль по умолчанию, а именно:

- Netware – mpm_netware;
- OS/2 – mpmt_os2;
- Windows – mpm_winnt;
- Unix – prefork, worker или event.

Так как под Unix здесь понимаются все Unix-подобные операционные системы (Linux, Solaris, Mac OS X и т. д.), то нужно дать комментарий, в каком случае делается выбор соответствующего МП-модуля. Согласно документации (https://httpd.apache.org/docs/2.4/en/mpm.html), выбор модуля по умолчанию зависит от двух факторов:

Поддержка потоков.

Поддержка потокобезопасной работы с соединениями.

Так как большинство современных ОС поддерживают обе эти возможности, по умолчанию выбирается event. Если обе возможности на уровне ОС не поддерживаются, по умолчанию выбирается prefork. Если поддерживается только разделение на потоки, то выбирается worker.

Для корректной работы систем на платформе «1С:Предприятие 8» МП-модулей по умолчанию обычно достаточно и «тонкий тюнинг» не требуется.

При компилировании Apache с другим МП-модулем требуется проводить тестирование стабильности и быстродействия такой сборки http-сервера на вашей площадке.

ГЛАВА 8

Практический пример развертывания внедрения по технологии 1cFresh

В этой главе на практическом примере рассмотрены некоторые типичные задачи, которые могут встать перед администратором информационной системы, построенной на основе технологии 1cFresh.

Описание тестовой среды

Одной из особенностей технологии 1cFresh является то, что в ее основе лежит большое количество различных компонентов, взаимодействующих между собой. С одной стороны, это позволяет обеспечить необходимую гибкость технологии, давая возможность реализовать большое количество различных вариантов настройки информационной системы. С другой стороны, корректная настройка каждого из компонентов в отдельности и их взаимодействия в целом является непростой задачей для администратора.

Для облегчения процесса знакомства с технологией в состав продукта «Технология публикации решений 1cFresh» включены пошаговые руководства, описывающие наиболее типичные варианты развертывания компонентов сервиса. В руководствах описаны варианты для серверов, работающих под управлением ОС Windows, Linux, а также для смешанной инфраструктуры, когда одна часть серверов работает на Windows, а другая на Linux. В процессе первичного знакомства с технологией настоятельно рекомендуется самостоятельно выполнить процесс развертывания хотя бы в одном из перечисленных вариантов – это гарантирует, что будет получено представление о назначении и о способе настройки каждого из компонентов.

В некоторых случаях специалистам по разработке или администрированию требуется тестовая инфраструктура с развернутыми компонентами технологии сервиса. В этих

случаях процесс самостоятельного развертывания может послужить причиной неоправданных затрат времени. Поэтому в состав продукта «Технология публикации решений 1cFresh» включен готовый шаблон виртуальной машины, включающий в себя все компоненты технологии сервиса в готовом к использованию виде (ОС Ubuntu Linux), а также инструкции по подготовке таких шаблонов (для ОС Windows и Linux). В тестовом примере, рассматриваемом в данном пособии, используется именно такой шаблон (вариант для ОС Linux, версия технологии 1.0.18.1).

Документация по технологии, примеры развертывания и шаблоны виртуальных машин доступны по следующему адресу (после приобретения): https://releases.1c.ru/project/FreshPublic.

Обратите внимание, что шаблон, включенный в комплект поставки «Технологии публикации решений 1cFresh», предназначен только для целей обучения, разработки и тестирования. Не следует использовать данный шаблон для реальной работы пользователей.

Настройка машин

1. Настройка компьютера администратора.

   Рекомендуется установить следующие приложения:

   - Текстовый редактор, например: Notepad++, EmEditor или Sublime Text (или любой другой на ваш вкус).

     В процессе работы часто возникает необходимость редактирования текстовых файлов различных форматов, например файлов настройки технологического журнала logcfg.xml. Ключевым требованием к редактору должна стать подсветка синтаксиса, что значительно облегчает процесс редактирования и выявления ошибок.

   - Cygwin (стандартные компоненты, а также openssh и mc).

     Это UNIX-подобная среда и интерфейс командной строки для Microsoft Windows. Вместе с cygwin устанавливается большое количество различных утилит (например, таких как cat, awk, sed, grep), которые способны значительно облегчить процесс анализа журналов (в частности, технологических журналов «1С:Предприятия»).

     Также cygwin будет полезен при администрировании Linux-серверов – как в плане обеспечения удаленного доступа через ssh, так и для решения различных административных задач. Например, с его помощью легко может быть решена задача удаленного выполнения команд на группе серверов.

   - Total Commander.

     Это приложение удобно использовать при администрировании как Windows-, так и Linux-серверов для копирования различных файлов (например, дистрибутивов платформы, технологических журналов и т. п.).

     При администрировании Linux-серверов дополнительно необходимо установить расширение SFTP (wfx_sftp_1_4_63_8); при этом если используется аутентификация по ключу, то для настройки подключения

необходимо сконвертировать ключ PEM в формат PPK с помощью утилиты puttygen (входит в состав Putty).

2. Настройка сервера.

   В шаблоне развертывания технологии используется ОС Ubuntu 14.04.

   Проверить используемый дистрибутив Linux, а также его версию можно с помощью следующей команды:

   ```
   cat /etc/*release*
   ```

Установка компонентов сервера «1С»

- Платформа «1С:Предприятие».

  В развернутом шаблоне уже установлен сервер «1С:Предприятия». Тем не менее может потребоваться заменить версию на другую. Это можно быстро сделать с помощью следующих команд:

  - Удаление:

    ```
    dpkg -l | grep enter | awk -F' ' '{print $2}' | xargs dpkg –r
    ```

    или

    ```
    dpkg -l | grep enter | cut -d “ “ -f 3 | xargs dpkg –r
    ```

  - Установка:

    ```
    dpkg –i *
    ```

    Команду нужно выполнять в каталоге, в который скопированы deb-пакеты дистрибутива технологической платформы. При этом в каталоге не должно быть других файлов.

    Для корректной работы платформы в среде Linux также должны быть установлены следующие компоненты (в шаблоне уже установлены):

  - Шрифты Microsoft TrueType:

```
apt-get install ttf-mscorefonts-installer
```

  - Пакет ImageMagic:

```
apt-get install imagemagick
```

  - HASP License Manager, если используются аппаратные ключи.

    В шаблоне используется менеджер лицензий от Etersoft.

  При необходимости установить его можно следующим образом:

  - Скачать дистрибутив с сайта Etersoft:

    http://updates.etersoft.ru/pub/Etersoft/HASP/7.40/Ubuntu

    Например:

```
wget http://updates.etersoft.ru/pub/Etersoft/HASP/stable/Ubuntu/12.04/haspd_7.40-eter10ubuntu_i386.deb
Установить deb-пакет:
dpkg -i haspd_7.40-eter10ubuntu_i386.deb
```

- Веб-сервер Apache 2.4.

  Просмотреть версии приложения, доступные для установки, можно с помощью следующей команды:

  ```
  apt-cache policy apache2
  ```

  А установить нужную версию можно так:

  ```
  apt-get install apache2=<version>
  ```

- Сервер СУБД PostgreSQL.

  Для поддержки работы с технологической платформой в PostgreSQL необходимо внести ряд изменений (наложить патчи).

  Патчи, а также собранные дистрибутивы PostgreSQL со всеми необходимыми изменениями доступны для скачивания по следующей ссылке: https://releases.1c.ru/project/AddCompPostgre

- Java 1.7.

  Используется для работы сайта и шлюза приложений. Установить можно с помощью следующей команды:

  ```
  apt-get install openjdk-7-jre
  ```

  Проверка установленной версии:

  ```
  java –version
  ```

Основные компоненты сервиса

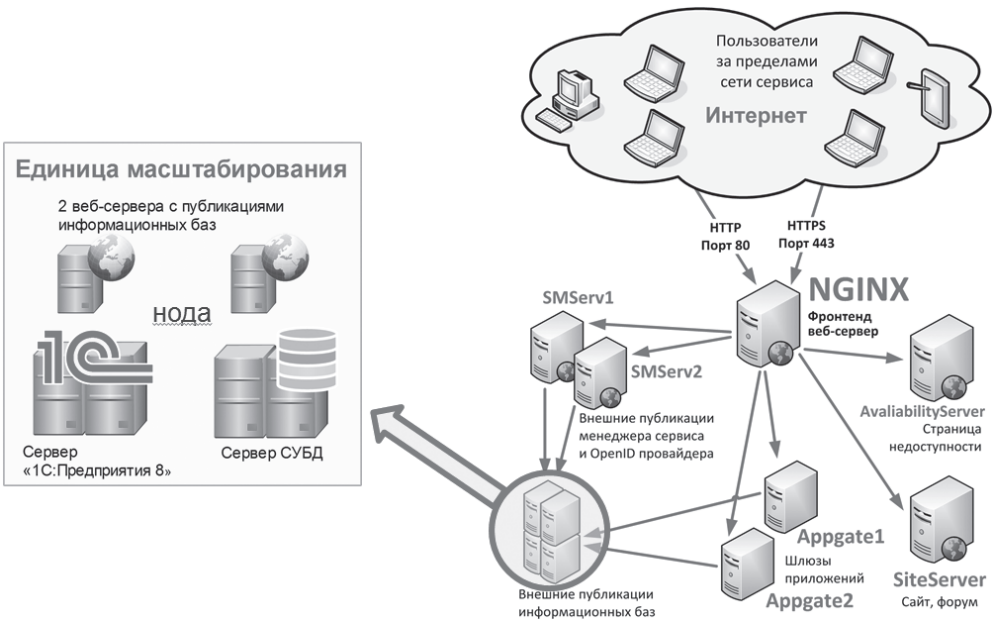


Рисунок 93. Основные компоненты сервиса

Установка и настройка Nginx

1. Почему необходим Nginx?

   Обеспечивает единую точку входа:

   - трафик перенаправляем в нужном направлении – в зависимости от URL запроса.

     HTTPS:

   - Nginx обеспечивает шифрование трафика по протоколу https. Все взаимодействие внешних пользователей с сервисом выполняется только в зашифрованном виде.

     Терминирование защищенного трафика:

   - Nginx является конечной точкой для защищенного канала. Компоненты сервиса взаимодействуют друг с другом без использования шифрования.

     Повышение производительности:

   - возможно использование нескольких веб-серверов, что позволяет выполнять горизонтальное масштабирование на этом участке, а также повышает надежность;
   - кеширование и сжатие информации;
   - быстрая отдача статического контента.
   - Альтернатива – IIS.

2. Установка Nginx:

```
apt-get -y install nginx
```

   - Альтернативный вариант:
     - http://nginx.org/
     - Download
     - Linux packages for mainline version
     - Найти свой дистрибутив и выполнить описанные действия.

3. Настройка Nginx.

   /etc/nginx/nginx.conf – **основной конфигурационный файл**:

   - user nginx – имя пользователя операционной системы, от имени которого запускаются рабочие процессы Nginx.
   - worker_processes 2 – число рабочих процессов. Обычно выставляется по числу ядер на машине, если сервер выделенный. Также в последних версиях Nginx стало доступно значение параметра auto.
   - worker_rlimit_nofile 10240 – число файлов, которые может открыть рабочий процесс.
   - error_log /var/log/nginx/error.log warn – расположение лога для регистрации ошибок, а также уровень важности регистрируемых событий.
   - pid /var/run/nginx.pid – задает расположение файла, в котором хранится идентификатор (PID) главного процесса.

Раздел events – предназначен для директив, влияющих на обработку соединений:

- worker_connections 1024 – максимальное число соединений, которые может открыть рабочий процесс (включая как клиентские соединения, так и соединения с проксируемыми серверами);
- use epoll – метод, используемый для обработки соединений. Epoll – наиболее эффективный метод при работе в Linux с ядром 2.6 и старше.

Раздел http – предназначен для конфигурирования Nginx в качестве балансировщика нагрузки:

- Include /etc/nginx/1c_upstream.conf – в этот файл вынесено описание групп серверов, между которыми распределяется нагрузка. Каждая такая группа называется upstream. В шаблоне определено 3 upstream:
  - backend83 – публикации информационных баз на веб-сервере (в данном случае веб-серверы с публикациями информационных баз «1С:Предприятия» выступают в качестве backend-серверов, а Nginx – в качестве frontend-сервера). Используется для тех баз, которые не требуется делить на несколько (SD, ЦКК, менеджер сервиса);
  - gate – шлюзы приложений, через которые производится работа с публикациями прикладных информационных баз (БП, УНФ, ЗУП…);
  - upstream_availability – upstream, на который происходит перенаправление в случае возникновения 400-х и 500-х ошибок. В шаблоне задано перенаправление на сайт: server <имя сервера>:8000;
  - директивы, заданные в настройках upstream: server – адрес и порт сервиса, входящих в группу (директив этого вида может быть больше одной); ip_hash – задает метод балансировки нагрузки, при котором запросы от одного клиента отправляются на один и тот же сервер из числа входящих в группу; keepalive 16 – задает размер пула соединений с upstream-сервером, который может удерживать каждый из рабочих процессов.
- Параметры proxy:
  - proxy_buffering on; – разрешает буферизацию ответов проксируемого сервера, считывая ответ от сервера максимально быстро, а затем отдавая его клиенту;
  - proxy_buffer_size 32k; – размер буфера для чтения первой части ответа от проксируемого сервера (как правило, заголовки);
  - proxy_buffers 20 512k; – число и размер буферов для одного соединения, в которые будут читаться ответы, получаемые от проксируемого сервера;
  - proxy_connect_timeout 5; – таймаут установки соединения с проксируемым сервером;
  - proxy_max_temp_file_size 0; – в случае если ответ от проксируемого сервера не поместился в буфер, часть ответа может быть записана во временный файл. Значение 0 отключает эту возможность.

- keepalive_timeout 300 300; – время, в течение которого keep-alive соединение будет удерживаться со стороны сервера. Для динамических сайтов, к которым относятся веб-клиенты «1С:Предприятия», рекомендуется задавать достаточно большое значение. Это позволяет минимизировать задержки, связанные с установкой соединения.

- client_max_body_size 4096m; – максимально допустимый размер тела запроса, полученного от клиента.
- client_body_buffer_size 256k; – размер буфера для чтения тела запроса, полученного от клиента. Если тело запроса больше заданного буфера, то используется временный файл на диске.
- Настройки логирования:
  - Задают формат и расположение лог-файлов. Рекомендуется использовать формат с вертикальной чертой в качестве разделителя полей, т. к. это значительно облегчит процесс анализа таких логов (например, с помощью утилиты awk). Возможный вариант настройки:

```
log_format fresh '$time_local|$status|$remote_addr|$upstream_addr|
$upstream_status|$request_time|
$upstream_response_time|
$upstream_http_x_fresh_backend
($upstream_http_x_destination_id)$err_log_backend|
$body_bytes_sent|$host|$request|$http_referer|
$http_user_agent|$http_cookie|
$upstream_http_set_cookie';
```

```
access_log /var/log/nginx/access.log fresh.
```

  - include /etc/nginx/conf.d/*.conf; – подключает конфигурационные файлы из каталога /etc/nginx/conf.d/

local.conf

В файле описана конфигурация виртуального сервера, работающего на порту 80 (http). Конфигурация задана таким образом, что любой запрос получает код возврата http 301 «ресурс навсегда перемещен» с перенаправлением на тот же адрес, но по протоколу https:

```
listen      80;
server_name <имя сервера>;
location / {
    return 301 https://$server_name$request_uri;
}
```

local-ssl.conf

В файле описана конфигурация виртуального сервера, работающего по протоколу https. Наиболее важные директивы:

- listen 443; – сервер слушает порт 443;
- proxy_intercept_errors on; – включает перехват ошибок (если включен, то используется директива error_page);

- группа директив ssl_ – задает расположение сертификатов, используемые протоколы шифрования и т. д.;
- location @start – именованный location. Location используются для задания той или иной конфигурации в зависимости от адреса, а именованные – для перенаправления. В данном случае применяется инструкция rewrite, которая заменяет адрес на другой – в этом примере по регулярному выражению:

```
rewrite ^(/a/[a-zA-Z0-9_]+/([0-9]+/)?).*$ $1 last;
```

- поскольку регулярные выражения не являются базовыми знаниями для «1Сника», то для понимания можно воспользоваться одним из сервисов, объясняющих регулярные выражения. Такие сервисы можно найти с помощью поисковой системы по ключевым словам explain regex, например: https://regex101.com/#python
- location /availability/ – описывает правило обработки запросов, пришедших на адрес /availability (страница недоступности). Здесь используется директива proxy_path, которая предназначена для перенаправления вызова на какой-либо backend из числа описанных в файле /etc/nginx/1c_upstream.conf. В данном случае выполняется перенаправление на upstream_availability, за которым скрывается адрес веб-сервера Java-приложений Apache Tomcat, который обслуживает сайт (см. файл /etc/nginx/1c_upstream.conf):

```
location /availability/ {
        include         1c_common.conf;
        recursive_error_pages on;
        error_page      400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415
416 417 500 501 502 503 504 505  /i/index.html;
        proxy_pass      http://upstream_availability;
    }
```

Здесь также фигурируют следующие директивы:

- □ recursive_error_pages on; – разрешает делать несколько перенаправлений через директиву error_page;
- □ error_page <коды ошибок> /i/index.html; – задает адрес, который будет показываться для указанных ошибок.

- location /i/ – указывает адрес статической страницы недоступности (/var/www/failover);
- location ^~ /a/openid – правило обработки запросов к OpenID. Здесь также используется директива proxy_path, но перенаправление выполняется на backend83, т. е. на веб-серверы, на которых, в свою очередь, выполнена веб-публикация менеджера сервиса в режиме провайдера OpenID:

```
proxy_pass http://backend83;
```

- ^~ /a/adm/e1cib/start – эта страница используется в процессе openID-аутентификации к менеджеру сервиса. Также выполняется перенаправление на backend83, но с особенностью: если в процессе openID-аутентификации возникают ошибки, то выполняется перенаправление на адрес приложения. Это сделано, для того чтобы спрятать служебные окна, непонятные пользователю;

- ^~ /a/wcib/hs и ^~ /a/wcibprivate/hs – веб-сервисы внешнего управления сеансами, которые также опубликованы на backend веб-сервере. Для того чтобы соединение не удерживалось (с целью повышения производительности при большом количестве подключений), тут указана следующая директива: keepalive_timeout 0;
- ^~ /a/sd – описывает правила обработки запросов к информационной базе сервис-деска;
- публикации прикладных баз, веб-сервисов, их странички входа и openid:

```
~* /a/\w+/\d+/e1cib/start
~* /a/\w+/\d+/\w+/e1cib/oid2rp
~* /a/\w+/\d+/ws
~* /a/\w+/\d+
```

Особенности:

- перенаправление выполняется не на backend83, а на gate, т. е. шлюз приложений;
- не выполняется перехват ошибок при работе с веб-сервисом (proxy_intercept_errors off). Это сделано, поскольку результат обращения к веб-сервису обычно не показывается пользователю и перехватывать техническое описание ошибки в данном случае было бы бессмысленно.

- location / – обращения к корню сайта перенаправляются на TomCat;
- /resources/images/content – обращения к статическим ресурсам сайта перенаправляются напрямую к файлам на диске.

Файл 1c_common.conf

- Включен в описании отдельных location в файле /etc/nginx/local-ssl.conf.
- Описывает таймауты:
  - proxy_connect_timeout – время, в течение которого Nginx ожидает подключения к серверу upstream;
  - 5 секунд, затем переходим к следующему серверу в группе;
  - proxy_read_timeout – таймаут между операциями чтения. По сути, ограничение на длительность одного клиент-серверного вызова: выполняется ожидание ответа от сервера в течение 75 секунд; если за это время ответ не был получен, то вызов завершается с ошибкой;
  - proxy_next_upstream error – признак переадресации запроса следующему серверу в группе, если произошла ошибка соединения. Альтернативные варианты: timeout, off, http_500 и т. д. (можно перечислять их в директиве через пробел).

Файл 1c_app.conf

- Конфигурационный файл для обработки запросов к прикладным базам (указан в описании соответствующих location в файле local-ssl.conf).
- Отличается добавлением признака X-Forwarded-Port 443.

Файл 1c_error.conf

- Описывает правила обработки ошибок при обращении к базам «1С:Предприятия».
- if ($http_user_agent = 1CV8C) { – указывает, что если подключение из тонкого клиента, то описание ошибок нужно передать простым текстом, так как в противном случае в текст ошибки, выводимой пользователю, будут включены теги html.
- error_page – указывает, на какие страницы перенаправлять при возникновении ошибок.

Файл 1c_error_openid.conf

- Описаны правила обработки ошибок при обращении к публикации /a/openid.
- Наполнение аналогично файлу 1c_error.conf.

Файл 1c_error_site.conf

В этом файле описаны правила обработки ошибок при обращении к сайту.

Файл 1c_keepalive.conf

- Директива proxy_set_header Connection ""; указывает, что в запрос не следует помещать заголовок «Connection: close». Это позволяет сохранять http-соединение активным.
- Используется в local-ssl.conf.

Для того чтобы проверить корректность конфигурации Nginx, нужно выполнить следующую команду:

```
service nginx configtest
```

Дополнительную документацию о структуре конфигурационных файлов Nginx и их параметрах можно найти по следующим адресам:

- http://nginx.org/ru/docs/beginners_guide.html
- http://nginx.org/ru/docs/ngx_core_module.html
- http://nginx.org/ru/docs/http/ngx_http_core_module.html
- http://nginx.org/en/docs/http/load_balancing.html
- http://nginx.org/en/docs/http/ngx_http_upstream_module.html
- http://nginx.org/ru/docs/http/ngx_http_proxy_module.html

Конфигурация веб-публикаций

1. Расположение конфигурационных файлов: /etc/apache2/
2. Особенности настройки Apache:
   - Основной конфигурационный файл в этой версии apache – apache2.conf.

- Все специфичные настройки описаны в конфигурационных файлах, которые подключены с помощью директив Include: Include ports.conf

Файл /etc/apache2/ports.conf:

```
LoadModule _1cws_module "/opt/1C/v8.3/x86_64/wsap22.so"
...
Listen 8888
Listen 8889
Listen 8890
Include 1cFresh_a/*.conf
Include 1cFresh_int/*.conf
```

- Директива LoadModule предназначена для загрузки модуля расширения «1С:Предприятия».
- Директивы Listen описывают порты, которые слушает веб-сервер.
- Директивы Include предназначены для подключения конфигурационных файлов, описывающих публикации информационных баз «1С:Предприятия»:
  - 1cFresh_a – каталог с внешними публикациями (предназначенными для доступа внешних пользователей);
  - 1cFresh_int – каталог с внутренними публикациями (предназначенными для внутреннего взаимодействия компонентов сервиса);
  - Пример файла настройки публикации информационной базы:

```
Alias /a/ea /var/www/1cFresh/a/ea
<Directory /var/www/1cFresh/a/ea/>
   AllowOverride All
   Options None
   Order allow,deny
   Allow from all
   SetHandler 1c-application
   ManagedApplicationDescriptor /var/www/1cFresh/a/ea/default.vrd
</Directory>
```

В каждом файле настройки публикации присутствует параметр ManagedApplicationDescriptor – это путь к файлу, в котором описаны дополнительные параметры, специфичные для веб-расширения технологической платформы «1С:Предприятие», такие как параметры подключения к информационной базе, режим использования разделения данных и т. п.

  - Пример файла default.vrd неразделенной информационной базы:

```
<?xml version="1.0" encoding="UTF-8"?>
<point
      base="/int/sm"
      ib="Srvr=localhost;Ref=sm;"
      xmlns="http://v8.1c.ru/8.2/virtual-resource-system"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <ws />
      <debug enable="true" url="tcp://localhost"/>
</point>
```

- Пример файла default.vrd разделенной информационной базы:

```
<?xml version="1.0" encoding="UTF-8"?>
<point
      base="/a/ea"
      ib="Srvr=1cFresh-lin-1.local;Ref=ea;"
      xmlns="http://v8.1c.ru/8.2/virtual-resource-system"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <zones>
            <zone specify="false" safe="true"/>
            <zone specify="true" safe="true"/>
      </zones>
   <openid>
      <rely url="https://1cFresh-lin-1.local/a/openid/e1cib/oid2op" />
   </openid>
   <debug enable="true" url="tcp://localhost"/>
</point>
```

Разделенная публикация отличается наличием тегов <zones>, которые описывают использование разделителей данных для данной публикации, а также тегов <openid>, которые включают использование OpenID-аутентификации.

Конфигурация менеджера сервиса

1. В шаблоне виртуальной машины уже присутствует настроенный менеджер сервиса. Однако в процессе иногда может возникнуть необходимость развертывания новой информационной базы менеджера сервиса. Для этого необходимо выполнить следующие действия:
   - Создание и публикация ИБ менеджера сервиса:
     - Создать базу.
     - Войти первый раз – при этом выполнится начальное заполнение базы. После завершения начального заполнения необходимо сразу же завершить клиентское приложение.
     - В конфигураторе задать пароль для всех пользователей, кроме «Аноним», и снять флаг Показывать в списке выбора.
     - Создать файл внешней публикации /etc/apache2/1cFresh_a/adm.conf.
     - Создать файл описания виртуального ресурса /var/www/1cFresh/a/adm/default.vrd.
     - Создать файл внутренней публикации /etc/apache2/1cFresh_int/sm.conf.
     - Создать файл описания виртуального ресурса /var/www/1cFresh/int/default.vrd.
     - Выполнить перезапуск веб-сервера:

```
sudo service apache2 restart.
```

     - Проверить работоспособность публикации (https://1cFresh-lin-1.local/a/adm) .

- Настройка ИБ менеджера сервиса:
  - Рабочий стол – Сервис – Настройки конфигурации – Настройки доступа.
  - Заполнить следующие поля:
    - внешний адрес этого приложения – через https;
    - внутренний адрес этого приложения – через http;
    - пользователь удаленного доступа – RemoteAccess – пользователь, который выполняет подключение к веб-сервисам внутренних публикаций;
    - остальные поля заполняются после регистрации этой базы в списке информационных баз и развертывания сайта.
  - Регистрация сегмента сервиса:
    - Рабочий стол – Сегменты сервиса;
    - создать сегмент «Локальная сеть»;
    - рабочие серверы;
    - создать рабочий сервер с именем, каким он называется в консоли кластера;
    - при необходимости задать учетные записи администрирования (если заданы администраторы кластера);
    - создать кластер;
    - если кластер работает под Linux, то необходимо использовать способ администрирования ras.
  - Регистрация конфигурации менеджера сервиса:
    - Администрирование – Конфигурации;
    - создать новый элемент;
    - заполнить параметры;
    - имя должно точно совпадать с именем, заданным в конфигураторе;
    - тип – «управляющая».
  - Регистрация версии конфигурации менеджера сервиса:
    - В элементе справочника Конфигурации Перейти – Версии конфигураций;
    - Создать;
    - Файл конфигурации – для менеджера сервиса указывать не нужно;
    - заполнить поля Версия, Тип версии, Версия платформы;
    - в документации указано, что версия должна совпадать с реальной. Тем не менее несовпадение версий обычно не приводит ни к каким негативным последствиям (справедливо только для менеджера сервиса!).
  - Регистрация информационной базы менеджера сервиса:
    - Администрирование – Информационные базы;
    - создать элемент;
    - заполнить поля;

- ✧ веб-адрес нужно указать (это внешний адрес), а веб-адрес управления нужно оставить пустым (адрес используется для подключения к базе из менеджера сервиса);
- ✧ если в менеджере сервиса уже зарегистрирована какая-нибудь информационная база с пустым веб-адресом управления, то возникнет сообщение об ошибке. В этом случае в соответствующем поле можно написать произвольный адрес.

○ Регистрация менеджера сервиса в менеджере сервиса:

- ✧ Администрирование – Менеджеры сервиса;
- ✧ заполнить поля;
- ✧ записать;
- ✧ выбрать менеджер сервиса на вкладке Настройки доступа: Рабочий стол – Настройки конфигурации – Настройки доступа – Этот менеджер сервиса.

○ Регистрация версий платформы:

- ✧ Администрирование – Версии платформы 1С:Предприятие;
- ✧ при первом запуске текущая версия платформы регистрируется автоматически.

2. Регистрация прикладной информационной базы.

□ Создание ИБ «Бухгалтерии предприятия»:

○ Создать базу.

○ Войти первый раз, при этом будет выполнено начальное заполнение базы. После того как процесс начального заполнения завершится, необходимо завершить клиентское приложение.

○ Если пропустить этот шаг, то при проверке подключения будет возникать ошибка: «Выполняется обновление на новую версию».

○ Войти конфигуратором и создать пользователей:

- ✧ Администратор – Администратор системы + Полные права;
- ✧ RemoteAccess – роль Выполнение синхронизации данных и все роли Удаленный доступ…;
- ✧ Сразу же снять флаг Показывать в списке выбора.

○ Настроить журнал регистрации и установить признак проверки сложности паролей пользователей.

○ Войти в базу и установить значения констант:

- ✧ Детализировать обновление ИБ в журнале регистрации – включить;
- ✧ Использовать автономную работу в модели сервиса – включить;
- ✧ Использовать синхронизацию данных – включить;
- ✧ Использовать синхронизацию данных в локальном режиме – выключить;
- ✧ Использовать синхронизацию данных в модели сервиса – включить;
- ✧ Использовать синхронизацию данных в модели сервиса с приложением в Интернете – включить;

- ✧ Использовать синхронизацию данных в модели сервиса с локальной программой – выключить.

- □ Публикация ИБ «Бухгалтерии предприятия»:
  - ○ Создать файл внешней публикации /etc/apache2/1cFresh_a/ea.conf.
  - ○ Создать файл описания виртуального ресурса /var/www/1cFresh/a/ea/default.vrd:
    - ✧ Обратите внимание на тег <zones>.
    - ✧ Это описание разделителей в порядке их следования в конфигурации.
    - ✧ При использовании OpenID-аутентификации необходимо добавить тег <openid>.
  - ○ Создать файл внутренней публикации /etc/apache2/1cFresh_int/ea.conf.
  - ○ Создать файл описания виртуального ресурса /var/www/1cFresh/int/ea/default.vrd.
  - ○ Выполнить перезапуск веб-сервера:

```
sudo service apache2 restart
```

  - ○ Проверить доступность внутренней публикации без использования Nginx: http://1cFresh-lin-1.local:8888/int/ea
  - ○ Проверить доступность внешней публикации без использования Nginx: http://1cFresh-lin-1.local:8888/a/ea/0
  - ○ Убедиться, что войти без указания области нельзя: http://1cFresh-lin-1.local:8888/a/ea
  - ○ Регистрация конфигурации БП в менеджере сервиса:
    - ✧ Администрирование – Конфигурации;
    - ✧ создать новый элемент;
    - ✧ заполнить параметры;
    - ✧ имя должно точно совпадать с именем, заданным в конфигураторе;
    - ✧ тип – «прикладная».
  - ○ Регистрация версии конфигурации БП:
    - ✧ Перейти – Версии конфигураций;
    - ✧ Создать;
    - ✧ Файл конфигурации – для менеджера сервиса указывать не нужно;
    - ✧ заполнить поля Версия, Тип версии, Версия платформы. Здесь версия должна совпадать с реальной, иначе не будет работать загрузка областей в базу;
    - ✧ файл конфигурации – нужен для работы агента;
    - ✧ файл начальных данных – нужен для создания новых областей. Если файл начального заполнения не задан, то функциональность не пострадает – просто будут отрабатывать обработчики начального заполнения области. Это приведет к тому, что замедлится создание новых областей, а также вырастет нагрузка на оборудование, на котором расположены прикладные информационные базы, поэтому при реальной

эксплуатации рекомендуется всегда указывать файлы начального заполнения;

- ✧ для создания файлов начального заполнения необходимо создать новую файловую базу, загрузить в нее прикладную конфигурацию (БП, УНФ, ЗУП…) соответствующей версии и запустить клиентское приложение. После того как обработчики начального заполнения завершат работу, следует перейти в раздел Администрирование – Сервис – Выгрузить данные для перехода в сервис (верно для БП; для других конфигураций нужный пункт меню может отличаться). Откроется мастер выгрузки данных, после его выполнения будет сохранен файл, который можно использовать в качестве файла начального заполнения;
- ✧ аналогичным образом формируется файл начального заполнения для демонстрационного режима, но в этом случае для выгрузки можно использовать демобазу.

○ Регистрация информационной базы:

- ✧ Администрирование – Информационные базы;
- ✧ создать элемент;
- ✧ заполнить поля;
- ✧ указать веб-адрес (адрес внешней публикации, без номера области и слеша в конце);
- ✧ указать веб-адрес управления (адрес внутренней публикации без слеша в конце – используется для подключения к базе из менеджера сервиса);
- ✧ пользователь управления – RemoteAccess;
- ✧ перед проверкой подключения необходимо обязательно записать элемент;
- ✧ проверить правильность настроек кнопкой Проверить подключение;
- ✧ если используется агент сервиса, то указать учетную запись администрирования (Перейти – Учетные записи администрирования).

3. Переключить новую базу на использование старого формата журнала регистрации:

□ Это можно сделать после того, как все базы развернуты.

□ Остановить службу кластера:

```
sudo service srv1cv83 stop
```

□ Перейти в каталог /home/usr1cv8/.1cv8/1C/1cv8/reg_1541/<GUID>/1Cv8Log/.

□ Удалить или переместить старые файлы журнала.

□ Создать файл 1Cv8.lgf.

□ Назначить права доступа на созданный файл 1Cv8.lgf.

□ Запустить службу кластера:

```
sudo service srv1cv83 start
```

4. Создание областей в прикладной информационной базе:
   - Администрирование – Информационные базы – Бухгалтерия предприятия;
   - перейти во вкладку Дополнительные;
   - установить флаг Добавлять новые области;
   - указать количество свободных областей;
   - если подготовлены данные начального заполнения, то установить флаг Заполнять области начальными данными;
   - сохранить изменения.
5. Просмотр обмена сообщениями:
   - Администрирование – Обмен сообщениями;
   - Настройка:
     - все точки должны быть зеленого цвета, а даты актуальными;
     - если имеются ошибки (записи красного цвета), то для просмотра их описания необходимо дважды щелкнуть по соответствующей записи.
6. Просмотр списка областей данных:
   - Рабочий стол – Приложения;
   - флаг Только используемые.
7. Регистрация пользователей – владельцев абонента:
   - Зайти в менеджер сервиса под именем «Аноним»:
     - зарегистрироваться;
     - заполнить параметры: почта – любая, телефон – не обязателен;
     - просмотреть почту. В шаблоне виртуальной машины установлено приложение mutt, с помощью которого можно просмотреть почту, отправляемую из менеджера сервиса. Для просмотра сообщений введите в консоли: mutt
     - активировать учетную запись.
   - Если нужно подтвердить регистрацию вручную (почта не дошла):
     - войти в МС от имени Администратор;
     - Обслуживание – Пользователи сервиса – Запросы на регистрацию;
     - найти запрос.
   - Если включены тарифы, то нужно убедиться, что пользователь подписан на тариф:
     - Если не заполнить, то добавление приложений будет недоступно.
     - Как заполнять:
       - Обслуживание – Абоненты;
       - Изменить;
       - Подписки на тарифы;
       - Создать;
       - заполнить параметры.

8. Добавление приложения для начала ведения учета:
   - Войти в менеджер сервиса от имени свежесозданного пользователя.
   - Добавить – Для начала ведения учета.
   - Заполнить параметры.
   - Дождаться завершения загрузки.
9. Переход в сервис из локальной версии:
   - Войти в МС от имени свежесозданного пользователя.
   - Добавить – Загрузить данные из файла.
   - Выбрать файл.
   - Заполнить параметры.
   - Дождаться завершения загрузки.

Настройка OpenID-аутентификации

1. Зачем нужна OpenID-аутентификация?

   При включенной OpenID-аутентификации после входа в область информационной базы или на сайт информация о выполненном входе сохраняется на компьютере пользователя. После этого при входе в другую область сохраненная информация используется для аутентификации в другой области, в результате чего пользователю не требуется каждый раз вводить пароль.
2. Настройка OpenID-аутентификации:
   - Обязательно требуется наличие SSH – для этого используется Nginx.
   - Опубликовать OpenID-провайдер – используется менеджер сервиса:
     - /etc/apache2/1cFresh_a/openid.conf
     - /var/www/1cFresh/a/openid/default.vrd
   - Прописать OpenID в публикациях: rely
     - Например, /var/www/1cFresh/a/ea/default.vrd

Настройка шлюза приложений

1. Зачем нужен шлюз приложений?
   - Горизонтальная масштабируемость сервиса. По сути, определяется возможностью шлюза направить вызов в определенную единицу масштабирования, вычисленную по номеру области в заголовке запроса, которая точно должна быть в этой единице масштабирования.
   - Обеспечивается с помощью Java-приложения.
2. Как настроить:
   - Все настройки на сервере лежат в каталоге /opt/1C/1cFresh/appgate/settings.
   - Файл /opt/1C/1cFresh/appgate/settings/appgatesettings.xml:

- listenPort 8080 – порт, на который приходят запросы от клиентских приложений;
- managePort 9090 – порт, на который приходят управляющие команды от менеджера сервиса;
- appIdDepth 3 – расположение номера области в адресе (/a/ea/4 – 3-й элемент);
- таймауты:
  - readTimeout – максимальное время ожидания (в миллисекундах) поступления данных от удаленной стороны. Значение по умолчанию (отсутствующий элемент) – время ожидания не ограничено;
  - writeTimeout – максимальное время ожидания (в миллисекундах) отправки данных удаленной стороне. Значение по умолчанию (отсутствующий элемент) – время ожидания не ограничено;
- invalidApplicationStatus – код ответа при обращении к неизвестному приложению/номеру области.

□ Пользователь в appgate:

- для задания логина и пароля необходимо выполнить следующую команду:

```
/opt/1C/1cFresh/appgate/setAuth.sh appgate пароль;
```

- от имени этого пользователя менеджер сервиса будет выполнять подключение к appgate.

□ Пользователь appgate в менеджере сервиса:

- Обслуживание – Пользователи сервиса – Пользователи сервиса;
- Доступ к информационной базе разрешен;
- задать пароль пользователя;
- включить роль Удаленный доступ (шлюз приложений);
- свойства пользователя сервиса – Блокировать добавление.

□ Настройка веб-сервисов менеджера сервиса:

- Администрирование – Настройки конфигурации – Шлюзы приложения;
- Использовать шлюзы приложений;
- Веб-сервис – внутренняя публикация веб-сервисов менеджера сервиса (порты Apache);
- задать имя пользователя и пароль, которые задали в appgate.

□ Регистрация шлюза в менеджере:

- Администрирование – <Без раздела> – Шлюзы приложений;
- добавить запись;
- прописать настройки, заданные в конфигурационных файлах шлюза:
  - порт для запросов от клиентов;
  - порт управления;

- ✧ имя пользователя;
- ✧ пароль;

○ кнопка Обновить настройки подключения тестирует правильность настроек.

□ Изменение настроек веб-серверов.

Шлюз приложений полезен, если имеется более одной информационной базы. В этом случае можно использовать определенный адрес как общую точку входа для всех пользователей определенной конфигурации. Это дает свободу администраторам в плане размещения информационных баз на разных серверах, а также позволяет реализовать более сложные схемы с разделением одной информационной базы на две и более (технология «сплита» базы).

Для ознакомления с работой шлюза приложения необходимо иметь две разные публикации с одинаковым именем. Это можно легко сделать, когда имеется несколько физических или виртуальных серверов – в таком случае нет никаких ограничений в плане создания публикаций с одинаковым именем. Однако в том случае, если в распоряжении имеется только один тестовый виртуальный сервер, единственной возможностью для решения этой задачи является создание виртуальных веб-серверов, разнесенных по разным портам. Для настройки таких серверов необходимо:

○ Настроить файл /etc/apache2/1cFresh_a/ea.conf – все содержимое файла поместить внутрь следующей конструкции:

```
<VirtualHost *:8889>
   ServerName 1cFresh-ext
…
</VirtualHost>
```

○ Обратите внимание на ServerName – если его не задать, то будет возникать ошибка: «Этот сайт не может обеспечить безопасное соединение. Сайт отправил недействительный ответ».

○ Выполнить аналогичную настройку в файле публикации для второй информационной базы:

- ✧ /etc/apache2/1cFresh_a/ea_2.conf
- ✧ Для второй базы используем порт 8890.

○ Выполнить перезапуск веб-сервера:

```
service apache2 restart
```

□ Регистрация веб-серверов:

○ На эти серверы шлюз будет перенаправлять запросы.

○ Администрирование – <Без раздела> – Веб-серверы.

○ Прописать адрес и порт.

○ Наименование любое.

○ В нашем примере сервер Apache слушает 3 порта, 2 из которых будут прописаны в менеджере сервиса.

- Группы веб-серверов:
  - Одна база может быть опубликована сразу на нескольких веб-серверах.
  - Запрос может уйти на любой из них.
  - Для этого такие веб-серверы нужно включить в группу.
  - Администрирование – <Без раздела> – Группы веб-серверов.
  - Создаем элемент.
- Прописываем группу веб-серверов в свойствах информационной базы:
  - Администрирование – <Без раздела> – Информационные базы.
  - БП.
  - Веб-доступ – Группа веб-серверов.
  - То же самое в УНФ.
- Изменение настроек Nginx:
  - /etc/nginx/1c_upstream.conf
  - Upstream gate
  - Прописать вместо 8889 -> 8080 (порты гейта).
  - В шаблоне используется конфигурация с одним шлюзом приложений, но технически возможно использование нескольких шлюзов (в этом случае потребуется несколько физических или виртуальных серверов).
- Проверка:
  - service nginx restart
  - Войти в область данных.
  - Проверить результат по логам Nginx:
  - /var/log/appgate/info.log
  - sudo service appgate stop
    - ✧ В открытых приложениях в этот момент произойдет перемещение на страницу недоступности.

Переключение баз, в которые добавляются области

1. Запретить добавление областей в одной из информационных баз:
   - Администрирование – Информационные базы.
   - Выбрать базу, которую хотим отключить.
   - Нажать F2 или «карандашик».
   - Перейти на вкладку Дополнительные.
   - Снять флаг Добавлять новые области.
2. Удалить пустые области из базы, добавление областей в которую запрещаем:
   - Администрирование – Информационные базы.
   - Выбрать базу, которую хотим отключить.

- Нажать F2 или «карандашик».
- Перейти – Приложения.
- Снять флаг Только используемые.
- У пустых областей изменить состояние с Готово на К удалению, при этом необходимо задать любое наименование, иначе выполнить запись не удастся.

Проверка, к какой базе относится область

Для того чтобы проверить, к какой именно информационной базе относится определенная область, необходимо:

- На рабочем столе перейти по гиперссылке Приложения.
- В открывшемся списке найти нужную область по номеру.
- Открыть запись.
- Посмотреть значение поля Информационная база.

Подключение расширений, дополнительных отчетов и обработок

Общая информация

У партнеров имеется возможность подключения внешних отчетов и обработок. Это, по сути, обычные внешние отчеты и обработки, подготовленные специальным образом. При этом необходимо соблюдать ряд ограничений:

- Запрещено использование любого кода, который может угрожать безопасности сервиса:
  - Выполнить; Вычислить; ЗапуститьПриложение;
  - Использование привилегированного режима;
  - Внешние компоненты;
- Запрещено использование любого кода, который может угрожать производительности сервиса:
  - Неоптимальные запросы:
    - https://kb.1c.ru/articleView.jsp?id=44
  - Неоптимальный код:
    - Пауза в коде путем пустого цикла
    - Создание коллекций потенциально неограниченного объема в памяти сервера
  - Используется безопасный режим:
    - Нет возможности обращаться к любым файлам на сервере
    - Нет возможности обращаться к сетевым ресурсам
    - Нет возможности обращаться к внешним источникам данных

- Обработка должна быть кроссплатформенной:
  - Абсолютные имена файлов исключены
  - COM-объекты исключены
  - Обработка должна работать в любом поддерживаемом клиенте, либо выдавать корректное сообщение об ошибке
- Обработка должна корректно работать в клиент-серверном варианте:
  - Код клиента и сервера должен быть корректно разделен
  - Должны быть исключены длительные клиент-серверные вызовы
- Обработка должна корректно работать в режиме сервиса:
  - Не только через «Файл / открыть», но и при запуске из кода
- Обработка должна быть пригодной для аудита:
  - Обфускация кода и закрытые модули исключены
- Должна быть исключена возможность нарушения бизнес-логики пользователем
  - Исключены универсальные обработки
  - Исключена запись в режиме ОбменДанными.Загрузка = Истина;
- Рекомендуется, чтобы обработка соответствовала стандартам разработки (http://its.1c.ru/db/v8std). В частности:
  - Не должно быть закомментированного кода
  - Не должно быть неиспользуемого кода и реквизитов
- Должна присутствовать справочная информация с описанием обработки

Аналогичные требования предъявляются и к расширениям конфигурации.

Настройка в Менеджере сервиса

1. Настроить МС для поддержки дополнительных обработок и/или расширений:
   - Администрирование – Настройки конфигурации – Адаптация приложений.

   Дополнительные отчеты и обработки:
   - Разрешить загрузку дополнительных отчетов и обработок пользователями в приложения – включает режим добавления обработок непосредственно в область, без ограничений – снять.
   - Использовать каталог дополнительных отчетов и обработок – включает возможность добавления обработок с прохождением аудита – установить.
   - Разрешить выполнение дополнительных отчетов и обработок регламентными заданиями – решение принимается самостоятельно.
   - Разрешить загрузку дополнительных отчетов и обработок, не использующих безопасный режим – для публичного сервиса снять.
   - Использовать аудит загружаемых дополнительных отчетов и обработок – установить.
   - Настроить состав аудиторов – настроить.

- Редактировать критерии приема дополнительных отчетов и обработок в каталог – просто текст с правилами.
- Задать расписание уведомления исполнителей о новых задачах – позволяет задать расписание формирования уведомлений.

Расширения:

- Разрешить загрузку расширений пользователями в приложения – включает режим добавления расширений непосредственно в область, без ограничений – снять.
- Использовать каталог расширений – включает возможность добавления расширений с прохождением аудита – установить.
- Использовать аудит загружаемых расширений – установить.
- Настроить состав аудиторов – настроить.
- Редактировать критерии приема расширений в каталог – просто текст с правилами.
- Задать расписание уведомления исполнителей о новых задачах – позволяет задать расписание формирования уведомлений.

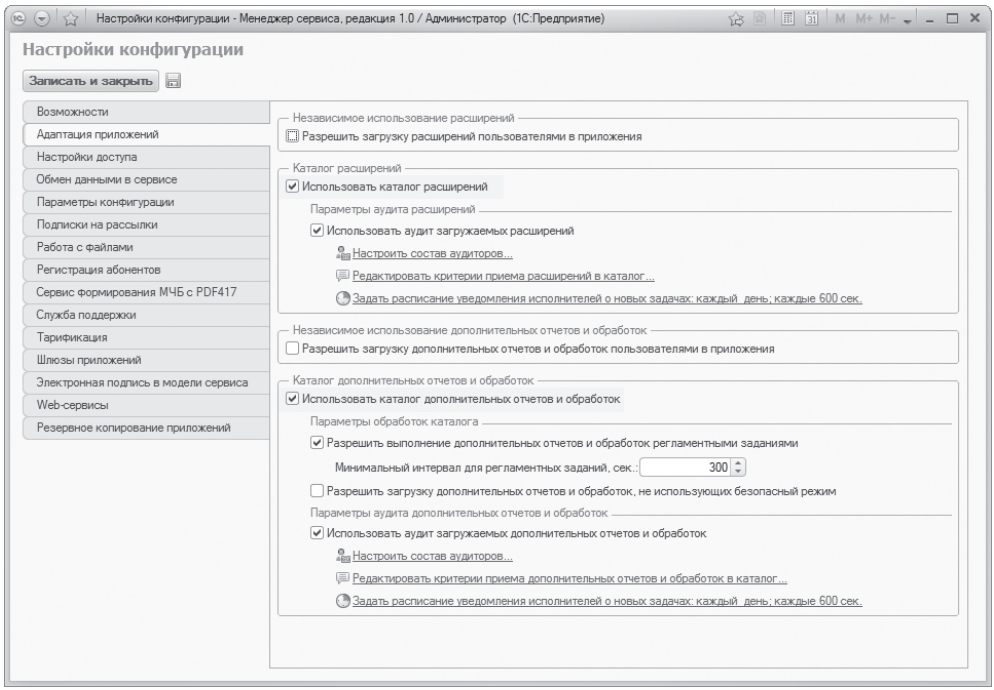


Рисунок 94. Настройка использования расширений и дополнительных отчетов/обработок в менеджере сервиса

2. Назначить права аудитору:

   Дополнительные отчеты и обработки:

   - Обслуживание – Пользователи сервиса – Пользователи сервиса.
   - Открыть пользователя Оператор.
   - Главное – Только выбранные роли – снять флаг.
   - Добавить роль Аудитор дополнительных отчетов и обработок.

   Расширения:

   - Обслуживание – Пользователи сервиса – Пользователи сервиса.
   - Открыть пользователя Оператор.
   - Главное – Только выбранные роли – снять флаг.
   - Добавить роль Аудитор расширений.

3. Включить пользователю возможность добавления обработок и/или расширений:

   Дополнительные отчеты и обработки:

   - Войти в менеджер сервиса под пользователем Оператор.
   - Обслуживание – Пользователи сервиса – Пользователи абонентов.
   - Найти пользователя, которому предоставляется право добавления внешних отчетов и обработок.
   - Открыть вкладку Разработка в модели сервиса.
   - Установить флаг Авторизовать для разработки дополнительных отчетов и обработок (с прохождением аудита администрацией сервиса).
   - Нажать Записать и закрыть.

   Расширения:

   - Войти в менеджер сервиса под пользователем Оператор.
   - Обслуживание – Пользователи сервиса – Пользователи абонентов.
   - Найти пользователя, которому предоставляется право добавления расширений.
   - Открыть вкладку Разработка в модели сервиса.
   - Установить флаг Авторизовать для разработки расширений (с прохождением аудита администрацией сервиса).
   - Нажать Записать и закрыть.

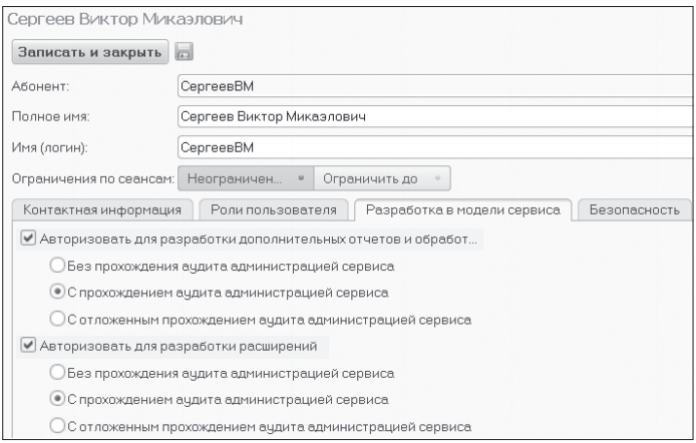


Рисунок 95. Включить пользователю возможность добавления обработок и/или расширений

- Пользователь Администратор может только проверять, предоставлено это право или нет:
  - Обслуживание – Пользователи сервиса – Пользователи абонентов;
  - Изменить;
  - Вкладка Доступ к дополнительным отчетам и обработкам – должен стоять флаг Разрешить создание новых дополнительных отчетов и обработок.
  - Вкладка Доступ к расширениям – должен стоять флаг Разрешить создание новых дополнительных отчетов и обработок.

4. Разработать и добавить обработку/расширение:
   - Открыть ИТС: БСП 2.3 – глава 3 «Настройка и использование подсистем при разработке конфигурации» – раздел «3.16. Дополнительные отчеты и обработки» – подраздел «Использование при разработке конфигурации» – «Создание нового отчета или обработки». Там описано по шагам, что нужно делать.
   - Разработка обработки:
     - Создать новую обработку.
     - В модуль объекта добавить обработчик Функция СведенияОВнешнейОбработке() Экспорт – см. «Пример реализации функции СведенияОВнешнейОбработке с ручным определением структуры параметров». Также можно использовать «Пример реализации функции СведенияОВнешнейОбработке с использованием программного интерфейса» – в этом случае нужно будет дополнительно задать значение параметра Информация.
     - Задать параметры регистрации (описаны в ИТС выше):
       - **Вид** – «ДополнительнаяОбработка»;
       - **Назначение** – только для обработок, предназначенных для определенных объектов Неопределено;

- ✧ **Наименование** – предназначен для идентификации обработки администратором. Нужно написать: «Тестовая дополнительная обработка (адм)»;
- ✧ **Версия** – для идентификации версии обработки. Строка. Нужно написать: «1.0.1.2»;
- ✧ **БезопасныйРежим** – в примере обработчика нет, но добавить нужно. Иначе не будет работать, если мы задали такое требование. Значение – Истина;
- ✧ **Информация** – текстовое описание того, что делает обработка и для чего она нужна. Нужно написать: «Тестовая дополнительная обработка (описание)»;
- ✧ **ВерсияБСП** – минимальная версия БСП, с которой будет работать обработка. Не изменяем.

○ Задать параметры команды:

- ✧ **Представление** – представление команды в пользовательском интерфейсе. Нужно написать: «Тестовая дополнительная обработка (команда в интерфейсе)»;
- ✧ **Идентификатор** – идентификатор команды – любая строка, уникальная в пределах данной обработки (отчета);
- ✧ **Использование** – тип команды из числа ДополнительныеОтчетыИОбработкиКлиентСервер.ТипКоманды…(). Там же описано, как их использовать. Нужно написать: ДополнительныеОтчетыИОбработкиКлиентСервер.ТипКомандыОткрытиеФормы();
- ✧ **ПоказыватьОповещение** – признак того, что нужно показывать оповещение при начале и при завершении работы обработки. Нужно написать Истина.

○ Создать форму обработки:

- ✧ назначить форму основной;
- ✧ создать параметр формы – ИдентификаторКоманды (тип Строка);
- ✧ добавить процедуру формы ПриСозданииНаСервере;
- ✧ добавить код, выводящий значение параметра ИдентификаторКоманды;
- ✧ добавить команду Команда1, вывести ее на форму и написать обработчик, выводящий произвольный текст.

□ Подготовить обработку/расширение к загрузке в сервис:

Дополнительные отчеты и обработки:

□ Открыть Менеджер сервиса от имени того пользователя, которому были предоставлены права на добавление дополнительных отчетов и обработок или расширений.

□ Еще – Дополнительные отчеты и обработки.

□ Создать/изменить.

□ Наименование.

- Создать версию.
- Скачать помощник подготовки.
- Открыть в БП, или УНФ, или в демо-базе БСП 2.3.
- Из файла на диске.
- Выбрать файл – посмотреть, как заполнились поля.
- Выбрать нужные разделы.
- Запуск по расписанию – пропустить пункт.
- Сохранить комплект поставки на диск…
- Готово.
- Загрузить обработку в сервис:
- Вернуться в менеджер сервиса.
- Выбрать файл ZIP.
- Выбрать конфигурацию «Бухгалтерия предприятия».

Расширения:

- Открыть Менеджер сервиса от имени того пользователя, которому были предоставлены права на добавление дополнительных отчетов и обработок или расширений.
- Еще – Расширения.

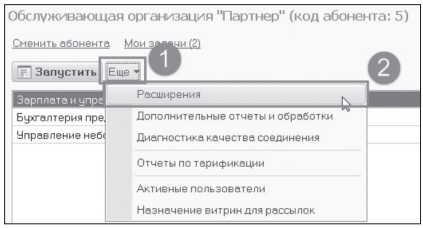


Рисунок 96. Добавление расширения

- Создать/изменить.
- Выбрать файл расширения.
- Указать информацию о расширении.
- Указать конфигурации и версии, с которыми совместимо расширение.
- В случае работы в небезопасном режиме указать опции работы (использование привелегированного режима, каталог временных файлов, доступные ресурсы).
- Готово.
- Обработка/расширение, отмеченное серым цветом, ждет аудита; кружок, если новая версия.

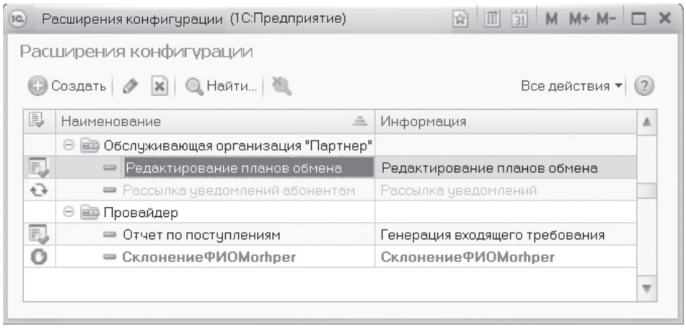


Рисунок 97. Список расширений в различных статусах

5. Провести аудит:
   - □ Войти в Менеджер сервиса под пользователем Оператор.
   - □ Найти раздел Мои задачи на рабочем столе.
   - □ Открыть задачу.
   - □ Одобрить или отклонить.
   - □ Завершить аудит.
   - □ Пользователю на почту будет отправлено оповещение.
6. Подключить обработку/расширение:
   - □ Открыть Менеджер сервиса от имени того пользователя, которому были предоставлены права на добавление дополнительных отчетов и обработок.
   - □ Открыть раздел Еще – Дополнительные отчеты и обработки (Еще – Расширения).
   - □ Нажать кнопку Изменить.
   - □ Нажать Права доступа – добавить нужным пользователям.
   - □ Установка/удаление:
     - ○ Добавить в нужные базы.
     - ○ Правая кнопка мыши – Установить.
     - ○ Выбрать разделы – Рабочий стол.
     - ○ Готово.
     - ○ Дождаться завершения установки внешней обработки в прикладную информационную базу.
   - □ В случае обновления на новую версию изменения должны попасть в прикладную базу автоматически (в течение нескольких минут). Проверить версию обработки в прикладной базе можно, открыв пункт меню Администрирование – Отчеты и обработки – Дополнительные отчеты и обработки – Версия.
7. Вывести обработку:
   - □ Войти в область прикладной информационной базы.
   - □ Администрирование – Отчеты и обработки – Дополнительные отчеты и обработки:

- Раздел Банк и касса.
- Щелкнуть правой кнопкой мыши на команде и выбрать пункт Настроить быстрый доступ…

□ Перейти в тот раздел, который был выбран на этапе настройки внешней обработки; Сервис – Дополнительные обработки:

- Найти команду и запустить ее. Должна открыться форма подготовленной внешней обработки.
- Посмотреть сообщения, выданные при открытии формы.
- Выполнить команду на форме.
- Посмотреть сообщения, выданные при выполнении команды.

ГЛАВА 9

Использование командной строки (bash) для анализа журналов

В процессе расследования проблем технологического качества при эксплуатации крупных информационных систем часто приходится анализировать большие объемы текстовой информации, различной по структуре и содержанию. Как правило, это различные журналы (access- и error-логи веб-серверов, журналы PostgreSQL, технологические журналы платформы «1С:Предприятие 8»). Прежде чем перейти к вопросу расследования проблем, изучим инструментарий.

Наиболее универсальным инструментом для анализа и структурирования большого объема текстовых данных показали себя на практике bash и его вариации. В данной главе рассматриваются примеры команд и подходы к их использованию при расследовании проблем эксплуатации.

Для пользователей Linux утилиты командной строки доступны «из коробки». Для пользователей Windows существуют варианты:

- с Windows 10 есть встроенный bash (пока в beta-версии) (https://msdn.microsoft.com/en-us/commandline/wsl/about);
- есть инструмент cygwin (https://www.cygwin.com/);
- есть git bash, который в своей поставке содержит все нужное (https://git-scm.com).

Авторы в OS Windows предпочитают использовать git bash, так как, по сути, его запуск и подготовка к использованию сводятся к клику правой кнопкой мыши в директории с журналами и выбору кнопки Git bash here.

Для знакомства с bash рекомендуется книга Скотт Граннемана «Linux. Необходимый код и команды. Карманный справочник».

Для знакомства с регулярными выражениями рекомендуем книгу Джеффри Фридла «Регулярные выражения».

Для удобства чтения

В данном материале будет применяться прием, используемый для удобства чтения.

В bash перенос строки обозначается символом \.

То есть, по сути, однострочная программа:

```
cat rphost*/*.log | sed 's/-/,/' | awk –F',' '{if($2 > 20000000) {print}}' | grep ',SDBL,.*Transaction' | tail
```

может выглядеть и выполняться так:

```
cat rphost*/*.log | \
sed 's/-/,/' | \
awk –F',' '{if($2 > 20000000) {print}}' | \
grep ',SDBL,.*Transaction' | \
tail
```

Для «сложных» однострочных программ такой подход будет удобен, чтобы программы менее походили на «страшные заклинания», однако вряд ли вы его будете применять на практике (авторы не применяют).

Простейшие операции

Копирование файла:

cp <что_копировать> <куда_копировать>

Копирование с удаленной машины server:

scp server:/var/dir/<что_копировать> /home/alex/<куда_копировать>

Переместить/переименовать файл:

mv <что_переместить> <куда_переместить>

Удалить файл:

rm <что_удалить>

Удалить все, включая поддиректории, без подтверждения:

rm –rf <что_удалить>

Список файлов:

ls

Список файлов в порядке «последние использованные – снизу» с информацией о правах и времени их модификации:

ls –lhatr

Только список в одну колонку:

ls -1

Только список в строку:

```
ls -1 | xargs
```

Получить список пользователей, а также что они выполняют:

```
w
```

Узнать первичную информацию о сервере

Время работы и нагрузка:

```
uptime
```

Физический или виртуальный сервер?

```
dmidecode -t system | grep “Product Name”
```

История команд:

```
history
```

Имя машины:

```
hostname
```

Информация о сервере:

```
dmidecode -t system
```

Оценка процессорных ресурсов

Сразу оценить среднюю нагрузку:

```
uptime
```

Топ процессов:

```
atop
```

Нажатие **p** покажет активность по программам.

По конкретному процессу с PID 1234:

```
pidstat -p 1234 1
```

Потребление памяти

Потребление памяти по программам:

```
atop
```

Нажатие **p** покажет активность по программам.

Информация по общесистемному использованию памяти:

cat /proc/meminfo

Текущее потребление памяти в системе:

sar -r 1 1

Использование дисков

Получить время утилизации дисковой подсистемы от времени работы:

iostat

Для указания технической информации о диске нужно выполнить:

smartctl -a /dev/sda1

Понять, что занимает основной объем:

df -h

du -s /mnt/c/* | sort -n | tail

Простой тест записи 10 Гб на диск:

time head -c 10000000000 /dev/urandom/ > /share/all/test.log

Информация о physical volumes:

pvs

Информация о logical volumes:

Lvs

Различные полезные команды в Linux

Получение дампов процессов в Linux без остановки процесса с сервера srv1c:

```
cd /var/cores
gcore -o /var/cores/rphost <PID>
scp srv1c:/var/cores/rphost.1234 rphost
```

Работа с atop при просмотре уже записанных журналов загруженности оборудования:

```
atop -r /var/log/atop
(m – память, v – процессы, u – пользователи, d – диск, t – следующий, T – предыдущий)
```

Получить список открытых файлов:

Lsof

Условия

Выполнить cmd2 только в том случае, если удалось успешно выполнить cmd1:

```
cmd1 && cmd2
```

Выполнить cmd2 только в том случае, если НЕ удалось успешно выполнить cmd1:

```
cmd1 || cmd2
```

Конвейер pipe

```
cmd1 | cmd2
```

Анализ журналов

Найти файлы по имени в текущей директории:

```
find . –name "16091011.log"
```

Найти файлы по размеру больше 10 Мб от корня файловой системы:

```
find / -depth -size +10M
```

Найти файлы старше 21:40 04.05.2016:

```
touch -t 05042140 ~/datastamp ; find . -newer ~/datastamp -exec ls -l {} \; rm ~/datastamp;
```

Просто вывести:

```
cat file
```

Вывести и пролистать:

```
more file
```

```
less file
```

Вывести первые 10:

```
head file
```

Вывести первые N:

```
head file –n 20
```

Вывести последние 10:

```
tail
```

Вывести последние N:

```
tail file –n 5
```

Следить за изменениями в файлах и выводить каждую секунду обновления:

```
tail –f <файл>
```

```
tail –f <множество_файлов*> (кроме «новых»)
```

Использование grep

Вывод строк, включающих соответствия выражению, из входного потока:

```
grep <выражение>
```

Поиск во всех подкаталогах:

```
grep -r <выражение>
```

Поиск в файлах:

```
grep <выражение> <файлы>
```

Вывод только совпадающего фрагмента:

```
grep –o <выражение>
```

Вывод только совпадающего фрагмента с использованием perl-конструкций:

```
grep –oP <выражение>
```

Вывести 5 строк до и после найденного выражения:

```
grep <выражение> -C 5
```

Вывести совпадения, но не выводить имена файлов и пути:

```
grep -h <выражение> <файлы>
```

```
grep -rh <выражение>
```

Вывести только определенное поле, например ClientID=<номер>, от строк, совпадающих с <выражением> (не выводя всю строку). Например, выражение – ",EXCP,":

```
grep -oPh ',EXCP,\K(ClientID=\d+)' <файлы>
```

Поиск по журналам rphost за определенный час (12 часов):

```
grep <выражение> rphost*/*12.log
```

Языки-утилиты

sed

Поиск и замена:

```
sed ‘s/найти/заменить/’
```

Вывести содержимое файла и заменить GUIDы, адреса и номера портов на короткие имена:

```
cat file | sed -r 's/.{8}-.{4}-.{4}-.{4}-.{12}/{GUID}/g' | sed -r 's/\:[0-9]{4,5}/{PORT}/g' | sed -r 's/[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+/{ADDR}/g'
```

Убрать все с начала строки до выражения:

```
sed 's/^.*<выражение>//'
```

Убрать все с выражения до конца строки:

```
sed 's/<выражение>.*$//'
```

Авторы считают более удобным в качестве альтернативы **sed** применять конструкцию с использованием **perl**:

```
perl -pe 's/найти/заменить/'
```

Но это вопрос предпочтений и очень незначительного выигрыша в производительности и удобстве.

Применить более одного раза (для всех вхождений):

```
perl -pe 's/найти/заменить/g'
```

awk

Программы awk:

```
awk –F<delimiter> 'BEGIN { программа анализа} END {программа}'
```

Например, получить сумму по 2-й колонке, если разделитель «;»:

```
awk –F';' '{sum+=$2;} END {print "Sum=" sum}'
```

Разделителем может быть даже слово:

```
awk –F'Memory=' '{sum+=$2;} END {print "Sum=" sum}'
```

Группировка по входным данным (группируемое поле – $1, суммируется – $2)

```
awk '{count[$1]+=$2;} END {for (i in count) {print i " " count[i]}}'
```

Простые реальные примеры

Вывести топ-10 совпадений с именами файлов, содержащий текст «Lock request timeout»:

```
grep -r 'Lock request timeout' | awk -F: '{print $1}' | sort | uniq -c | sort -r | head
```

Выводить все события-«исключения», возникающие сейчас в работающих процессах rphost:

```
tail –f rphost*/*.log | grep ',EXCP,'
```

Вывести последние 10 событий фиксации или отката транзакции в журналах всех процессов rphost длительностью более 20 секунд:

```
cat rphost*/*.log | sed 's/-/,/' | awk –F',' '{if($2 > 20000000) {print}}' | grep ',SDBL,.*Transaction' | tail
```

Оптимизация

Можно написать такую однострочную программу, которая будет выполняться недели... Ничего сложного в этом нет.

Для того чтобы программы выполнялись быстро, следует пользоваться следующими правилами:

- постараться как можно ближе к началу отобрать необходимый объем строк;
- не оперировать миллионами строк;
- помнить, что сортировки на сотнях тысяч строк могут быть дорогими.

Вы можете воспользоваться утилитой time для получения реального времени выполнения программы. Например:

```
time cat rphost*/*.log | sed 's/-/,/' | awk –F',' '{if($2 > 20000000) {print}}' | grep ',SDBL,.*Transaction' | tail
```

Когда вы только пишете свою программу, вы вряд ли захотите в процессе отладки каждый раз ждать, как прочитаются миллионы строк собранных журналов.

Поэтому в этом случае можно сразу после первого pipe | добавить head, тем самым отобрав только 10 строк.

Если программа на 10 строках работает ожидаемо, то, возможно, ожидаемое вами поведение сохранится и на большем числе строк (все зависит от вас).

Например:

```
cat rphost*/*.log | head | sed 's/-/,/' | awk –F',' '{if($2 > 20000000) {print}}' | grep ',SDBL,.*Transaction' | tail
```

Фильтрация событий технологического журнала платформы «1С:Предприятие 8»

В технологический журнал могут попадать события, которые занимают более одной строки.

Например, события DBMSSQL содержат текст запроса и могут содержать контекст. Нужен прием, который позволит фильтровать не строки, а именно события, включая контекст и другие многострочные поля.

Попробуем получить все события DBMSSQL по определенному сеансу SessionID=2049:

```
cat rphost_*/*.log | \
perl -n -e 'if (/^\d\d:\d\d\.\d+/) {$event =~ s/\n/<line>/g; print $event."\n"; $event = "";} $event .= $_; END{print $event."\n"};' | \
perl -pe 's/\xef\xbb\xbf//g' | \
grep "DBMSSQL.*SessionID=2049" | \
sed 's/<line>/\n/g' | \
less
```

Ключевые действия приведенного скрипта:

- преобразование события к одной строке

```
perl -n -e 'if (/^\d\d:\d\d\.\d+/) {$event =~ s/\n/<line>/g; print $event."\n"; $event = "";} $event .= $_; END{print $event."\n"};'
```

- избавляемся от символов BOM, которые могут принести проблемы при работе с различными кодировками, т. к. не будут учитываться

```
perl -pe 's/\xef\xbb\xbf//g' | \
```

- фильрация «утилитами» в привычном построчном режиме

```
grep "DBMSSQL.*SessionID=2049"
```

- преобразование «обратно» к многострочному варианту после необходимых фильтраций

```
sed 's/<line>/\n/g'
```

Обратное преобразование не всегда необходимо, поэтому можно, например, заменять <line> на пробелы:

```
sed 's/<line>/ /g'
```

Альтернативным способом преобразования события к однострочному и обратно к многострочному выводу является использование awk:

```
cat rphost*/*.log | awk -vORS= '{if(match($0, "^[0-9][0-9]\:[0-9][0-9]\.[0-9]+\-")) print "\n"$0; else print $0;}' | grep
'DBMSSQL.*SessionID=2049' | sed -e "s/\xef\xbb\xbf/ /g" | head
```

Естественно, есть и другие способы.

Применение теории

Приведенные примеры позволяют решить поставленные задачи в течение буквально пары минут, имея в руках только технологический журнал и «инструменты bash».

Поиск 5 наиболее длительных транзакций

Шаг 1. Получим только фиксации и откаты транзакций:

```
grep -P "SDBL.*Func=(Commit|Rollback)Transaction.*Context" rphost*/*.log | head
```

Шаг 2. Получим все свойства события полностью:

```
cat rphost*/*.log | perl -n -e 'if (/^\d\d:\d\d\.\d+/) {$event =~ s/.\n/<line>/g; print $event."\n"; $event = "";} $event .= $_; END{print $event};
' | perl -pe 's/\xef\xbb\xbf//g' | grep -P "SDBL.*Func=(Commit|Rollback)Transaction.*Context" | sed 's/<line>/\n/g' | head
```

Шаг 3. Оставляем события в одну строку (убираем «обратное» преобразование к многострочному варианту)

```
cat rphost*/*.log | perl -n -e 'if (/^\d\d:\d\d\.\d+/) {$event =~ s/\n/ /g; print $event."\n"; $event = "";} $event .= $_; END{print $event.
"\n";}' | perl -pe 's/\xef\xbb\xbf//g' | grep -P "SDBL.*Func=(Commit|Rollback)Transaction" | head
```

*Шаг 4***.** Оставим только нужные поля.

Например, длительность и первую строку контекста:

```
cat rphost*/*.log | perl -n -e 'if (/^\d\d:\d\d\.\d+/) {$event =~ s/.\n/<line>/g; print $event."\n"; $event = "";} $event .= $_; END{print $event};
' | perl -pe 's/\xef\xbb\xbf//g' | grep -P "SDBL.*Func=(Commit|Rollback)Transaction.*Context" | grep -oP "^\d+:\d+.\d+-\K(\d+),SDBL,.
*(,Context=[\'\"]<line>[^(<line>)]*)" | sed 's/<line>//g' | sed 's/,SDBL,.*Context/,Context/g' | head
```

или

```
cat rphost*/*.log | \
perl -n -e 'if (/^\d\d:\d\d\.\d+/) {$event =~ s/.\n/<line>/g; print $event."\n"; $event = "";} $event .= $_; END{print $event};' | \
perl -pe 's/\xef\xbb\xbf//g' | \
grep -P "SDBL.*Func=(Commit|Rollback)Transaction.*Context" | \
grep -oP "^\d+:\d+.\d+-\K(\d+),SDBL,.*(,Context=[\'\"]<line>[^(<line>)]*)" | \
sed 's/<line>//g' | sed 's/,SDBL,.*Context/,Context/g' | \
head
```

Например, длительность и последнюю строку контекста:

```
cat rphost*/*.log | perl -n -e 'if (/^\d\d:\d\d\.\d+/) {$event =~ s/.\n/<line>/g; print $event."\n"; $event = "";} $event .= $_; END{print $event};
' | perl -pe 's/\xef\xbb\xbf//g' | grep -P "SDBL.*Func=(Commit|Rollback)Transaction.*Context" | perl -pe 's/^\d+:\d+.\d+-//g'
| perl -pe 's/,SDBL,.*Context.*<line>[ \t]+/,Context=/g' | perl -pe 's/,SDBL,.*Context=/,Context=/g' | sed 's/<line>//g' | head
```

или

```
cat rphost*/*.log | \
perl -n -e 'if (/^\d\d:\d\d\.\d+/) {$event =~ s/.\n/<line>/g; print $event."\n"; $event = "";} $event .= $_; END{print $event};' | \
perl -pe 's/\xef\xbb\xbf//g' | \
grep -P "SDBL.*Func=(Commit|Rollback)Transaction.*Context" | \
perl -pe 's/^\d+:\d+.\d+-//g' | \
perl -pe 's/,SDBL,.*Context.*<line>[ \t]+/,Context=/g' | \
perl -pe 's/,SDBL,.*Context=/,Context=/g' | \
sed 's/<line>//g' | \
head
```

В конструкции 's/^\d+:\d+.\d+-//g' мы избавились от полей до длительности события, чтобы в дальнейшем можно было удобно группировать.

Шаг 5. Пробуем применить группировку с помощью awk:

```
awk -F, '{sum[$2]+=$1; count[$2]+=1;} END {for(i in sum) {print s[i] " " sum[i]/count[i] " " count[i] " " i}}'
```

Результат:

```
cat rphost*/*.log | perl -n -e 'if (/^\d\d:\d\d\.\d+/) {$event =~ s/.\n/<line>/g; print $event."\n"; $event = "";} $event .= $_;
END{print $event};' | perl -pe 's/\xef\xbb\xbf//g' | grep -P "SDBL.*Func=(Commit|Rollback)Transaction.*Context" | perl -pe 's/^\d+
:\d+.\d+-//g' | perl -pe 's/,SDBL,.*Context.*<line>[ \t]+/,Context=/g' | perl -pe 's/,SDBL,.*Context=/,Context=/g' | sed 's/<line>//g' | awk -F',
Context=' '{sum[$2]+=$1; count[$2]+=1;} END {for(i in sum) {print sum[i] " " sum[i]/count[i] " " count[i] " " i}}' | sort -rnb | head -n 5
```

или

```
cat rphost*/*.log | \
perl -n -e 'if (/^\d\d:\d\d\.\d+/) {$event =~ s/.\n/<line>/g; print $event."\n"; $event = "";} $event .= $_; END{print $event};' | \
perl -pe 's/\xef\xbb\xbf//g' | \
grep -P "SDBL.*Func=(Commit|Rollback)Transaction.*Context" | \
perl -pe 's/^\d+:\d+.\d+-//g' | \
perl -pe 's/,SDBL,.*Context.*<line>[ \t]+/,Context=/g' | \
perl -pe 's/,SDBL,.*Context=/,Context=/g' | \
sed 's/<line>//g' | \
awk -F',Context=' '{sum[$2]+=$1; count[$2]+=1;} END {for(i in sum) {print sum[i] " " sum[i]/count[i] " " count[i] " " i}}' | \
sort -rnb | \
head -n 5
```

Поиск 5 наиболее длительных запросов к СУБД MS SQL Server

На первый взгляд кажется, что можно применить уже известный прием:

```
cat rphost*/*.log | \
perl -n -e 'if (/^\d\d:\d\d\.\d+/) {$event =~ s/.\n/<line>/g; print $event."\n"; $event = "";} $event .= $_; END{print $event};' | \
perl -pe 's/\xef\xbb\xbf//g' | \
grep -P "DBMSSQL.*Context" | \
perl -pe 's/^\d+:\d+.\d+-//g' | \
perl -pe 's/,DBMSSQL,.*Context.*<line>[ \t]+/,Context=/g' | \
perl -pe 's/,DBMSSQL,.*Context=/,Context=/g' | \
sed 's/<line>//g' | \
awk -F',Context=' '{sum[$2]+=$1; count[$2]+=1;} END {for(i in sum) {print sum[i] " " sum[i]/count[i] " " count[i] " " i}}' | \
sort -rnb | \
head -n 5
```

Но проблема в том, что не все события DBMSSQL имеют контексты. Поэтому группировка по первой или последней строке контекста не подойдет. Нужно научиться группировать тексты запросов. Однако в них могут быть числовые параметры, различные имена временных таблиц (#tt17, #tt56), когда структура таких таблиц по сути одинаковая.

Таким образом, нам нужно научиться сворачивать тексты запросов так, чтобы «уникальные» имена пропали. Для тренировки явно добавим опцию

```
grep -v 'Context'
```

исключения Context в нашу программу, хотя, понятно, что *самые длительные запросы будут только среди запросов без контекста из-за использования* grep -v 'Context'. Но так проще для тренировки.

Шаг 1. Получаем запросы без контекстов.

```
cat rphost*/*.log | \
perl -n -e 'if (/^\d\d:\d\d\.\d+/) {$event =~ s/.\n/<line>/g; print $event."\n"; $event = "";} $event .= $_; END{print $event};' | \
perl -pe 's/\xef\xbb\xbf//g' | \
grep -P "DBMSSQL.*Sql" | \
grep -v 'Context' | \
perl -pe 's/<line>//' | \
perl -pe 's/^\d+:\d+.\d+-//g' | \
perl -pe 's/,DBMSSQL,.*Sql=/,Sql=/g' | \
head
```

Шаг 2. Можно убрать GUID и имена временных таблиц, номера строк и цифры.

```
cat rphost*/*.log | \
perl -n -e 'if (/^\d\d:\d\d\.\d+/) {$event =~ s/.\n/<line>/g; print $event."\n"; $event = "";} $event .= $_; END{print $event};' | \
perl -pe 's/\xef\xbb\xbf//g' | \
grep -P "DBMSSQL.*Sql" | \
grep -v 'Context' | \
perl -pe 's/<line>//' | perl -pe 's/^\d+:\d+.\d+-//g' | \
perl -pe 's/,DBMSSQL,.*Sql=/,Sql=/g' | \
perl -pe 's/\w+-\w+-\w+-\w+-\w+/{GUID}/g' | \
perl -pe 's/\(\d+\)/({NUM})/g' | \
perl -pe 's/tt\d+/{TempTable}/g' | \
head
```

Результат:

```
cat rphost*/*.log | perl -n -e 'if (/^\d\d:\d\d\.\d+/) {$event =~ s/.\n/<line>/g; print $event."\n"; $event = "";} $event .= $_; END{print $event};
' | perl -pe 's/\xef\xbb\xbf//g' | grep -P "DBMSSQL.*Sql" | grep -v 'Context' | perl -pe 's/<line>//' | perl -pe 's/^\d+:\d+.\d+-//g'
| perl -pe 's/,DBMSSQL,.*Sql=/,Sql=/g' | perl -pe 's/\w+-\w+-\w+-\w+-\w+/{GUID}/g' | perl -pe 's/\(\d+\)/({NUM})/g' | perl -pe 's/tt\d+
/{TempTable}/g' | awk -F',Sql=' '{sum[$2]+=$1; count[$2]+=1;} END {for(i in sum) {print sum[i] " " sum[i]/count[i] " " count[i] " " i}}'
| sort -rnb | head -n 5
```

или

```
cat rphost*/*.log | \
perl -n -e 'if (/^\d\d:\d\d\.\d+/) {$event =~ s/.\n/<line>/g; print $event."\n"; $event = "";} $event .= $_; END{print $event};' | \
perl -pe 's/\xef\xbb\xbf//g' | \
grep -P "DBMSSQL.*Sql" | \
grep -v 'Context' | \
perl -pe 's/<line>//' | \
perl -pe 's/^\d+:\d+.\d+-//g' | \
perl -pe 's/,DBMSSQL,.*Sql=/,Sql=/g' | \
perl -pe 's/\w+-\w+-\w+-\w+-\w+/{GUID}/g' | \
perl -pe 's/\(\d+\)/({NUM})/g' | \
perl -pe 's/tt\d+/{TempTable}/g' | \
awk -F',Sql=' '{sum[$2]+=$1; count[$2]+=1;} END {for(i in sum) {print sum[i] " " sum[i]/count[i] " " count[i] " " i}}' | \
sort -rnb | \
head -n 5
```

Поиск 5 наиболее длительных вызовов

В целом задача не отличается от предыдущих. Однако в предыдущих задачах мы можем заметить, что вывод awk нас не всегда устраивает, так как представлен в «удобном для чтения» формате. На самом деле он немного мешает.

Решим задачу и изменим немного вывод awk:

```
awk -F',Context=' '{sum[$2]+=$1; count[$2]+=1;} END {for(i in sum) {printf "%d %d %d %s\n",sum[i],sum[i]/count[i], count[i],i}}'
```

Результат:

```
cat rphost*/*.log | perl -n -e 'if (/^\d\d:\d\d\.\d+/) {$event =~ s/.\n/<line>/g; print $event."\n"; $event = "";} $event .= $_;
END{print $event};' | perl -pe 's/\xef\xbb\xbf//g' | grep -P ",CALL,.*,Context=" | perl -pe 's/<line>//' | perl -pe 's/^\d+:\d+.\d+-//g'
| perl -pe 's/,CALL,.*Context/,Context/g' | perl -pe 's/,Interface=.*OutBytes=\d+//g' | awk -F',Context=' '{sum[$2]+=$1; count[$2]+=1;}
END {for(i in sum) {printf "%d %d %d %s\n",sum[i],sum[i]/count[i], count[i],i}}' | sort -rnb | head -n 5
```

или

```
cat rphost*/*.log | \
perl -n -e 'if (/^\d\d:\d\d\.\d+/) {$event =~ s/.\n/<line>/g; print $event."\n"; $event = "";} $event .= $_; END{print $event};' | \
perl -pe 's/\xef\xbb\xbf//g' | \
grep -P ",CALL,.*,Context=" | \
perl -pe 's/<line>//' | \
perl -pe 's/^\d+:\d+.\d+-//g' | \
perl -pe 's/,CALL,.*Context/,Context/g' | \
perl -pe 's/,Interface=.*OutBytes=\d+//g' | \
awk -F',Context=' '{sum[$2]+=$1; count[$2]+=1;} END {for(i in sum) {printf "%d %d %d %s\n",sum[i],sum[i]/count[i], count[i],i}}' | \
sort -rnb | \
head -n 5
```

Поиск 5 пространств, на которых больше других возникали ожидания на управляемых блокировках

Используем тот же прием. Однако теперь нас интересует поле Regions. Также обращаем внимание, что нас интересуют не все события TLOCK, а только те, в которых было зафиксировано ожидание при установке управляемой блокировки. Помним, что в этом случае поле WaitConnections= не пустое.

Результат:

```
cat rphost*/*.log | perl -n -e 'if (/^\d\d:\d\d\.\d+/) {$event =~ s/.\n/<line>/g; print $event."\n"; $event = "";} $event .= $_; END{print $event};
' | perl -pe 's/\xef\xbb\xbf//g' | grep -P ",TLOCK,.*WaitConnections=\d+,Context" | perl -pe 's/^\d+:\d+.\d+-//g'
| grep -P 'WaitConnections=\d+' | perl -pe 's/,Locks=.*$//g' | perl -pe 's/,TLOCK,.*Regions=/,Regions=/g' | sed 's/<line>//g'
| awk -F',Regions=' '{sum[$2]+=$1; count[$2]+=1;} END {for(i in sum) {printf "%d %d %d %s\n", sum[i], sum[i]/count[i],count[i],i}}'
| sort -rnb | head -n 5
```

или

```
cat rphost*/*.log | \
perl -n -e 'if (/^\d\d:\d\d\.\d+/) {$event =~ s/.\n/<line>/g; print $event."\n"; $event = "";} $event .= $_; END{print $event};' | \
perl -pe 's/\xef\xbb\xbf//g' | \
grep -P ",TLOCK,.*WaitConnections=\d+,Context" | \
perl -pe 's/^\d+:\d+.\d+-//g' | grep -P 'WaitConnections=\d+' | \
perl -pe 's/,Locks=.*$//g' | \
perl -pe 's/,TLOCK,.*Regions=/,Regions=/g' | \
sed 's/<line>//g' | \
awk -F',Regions=' '{sum[$2]+=$1; count[$2]+=1;} END {for(i in sum) {printf "%d %d %d %s\n", sum[i], sum[i]/count[i],count[i],i}}' | \
sort -rnb | \
head -n 5
```

Аналогичным способом разобранную конструкцию можно самостоятельно применять и для других задачи поиска топ-5 <чего-то>.

Поиск по другим журналам

Если посмотреть, например, на access логи nginx, то мы видим пригодные для разбора журналы.

Например, (в зависимости от того, как выглядят журналы у вас) получение оценки, для какой единицы масштабирования запросы выполняются дольше других, может быть таким:

```
grep -P '24/Mar/2017:12'  /var/log/nginx/access.log | awk -F'|' '{if($6>40) {sum[$8]+=$6; count[$8]+=1; countall+=1;}}
END {for (i in sum) {print count[i]*100/countall "% " sum[i] " " sum[i]/count[i] " " count[i] " " i}}' | sort -nb | head -n 50
```

Когда серверов много…

Может возникнуть необходимость выполнить одну и ту же команду, например по ssh, на множестве различных серверов. Допустим, вы знаете, что в вашей сети серверы имеют имена вида *server1, server2, ... server100*. Тогда мы можем вывести имена всех серверов примерно так:

```
for i in `seq 1 100`; do echo server$i; done
```

Но тогда, например, поиск EXCP по множеству серверов будет выглядеть примерно так:

```
for i in `seq 1 100`; do ssh server$i 'grep ,EXCP, /var/log/srv1cv8/logs/FULL/rphost*/*.log'; done
```

Так как журналы могут быть большого объема, такая конструкция может привести к тому, что на множестве серверов будет обрабатываться большой объем данных.

Есть базовые рекомендации по применению однострочных команд (скриптов) для работы с журналами на распределенном контуре:

- Не следует выполнять очень тяжелые операции по большому числу серверов однострочными программами. Легко может быть допущена ошибка, поэтому всегда нужно иметь понимание, как при этом не навредить системе, так как при таком выполнении могут быть задействованы существенные ресурсы.
- Команды, выполняемые на серверах рабочего контура, обычно предназначены для того, чтобы узнать, есть ли проблемы сейчас.
- Для разбора и анализа статистической информации по журналам за большой период лучше использовать журналы, уже перемещенные в специальное сетевое хранилище.

По этой причине в нашем примере лучше искать более точечно. Например, среди последних 1000 строк:

```
for i in `seq 1 100`; do ssh server$i 'tail -n 1000 /var/log/srv1cv8/logs/FULL/rphost*/*.log' | grep ",EXCP,"; done
```

Естественно, через pipe | вы можете применять уже знакомые вам конструкции.

Архивы

Технологические журналы рекомендуется складывать в zip-архивы и не удалять какое-то время, т.к. они могут понадобиться для расследования тех проблем, которые были. Естественно, хранить такие архивы лучше не рабочих серверах, а в отведенных для этого хранилищах. Но разархивировать журналы долго, хочется сразу начать искать по архивам.

Для работы с заархивированными журналами можно воспользоваться конструкцией вида:

```
find * -name '*.zip' -print0 | xargs -0 -i -n 1 unzip -p {} '*.log' 2>/dev/null | <команда обработки данных журнала>
```

Таким образом, поиск в определенной директории zip-архива на удаленном сервере для поиска событий EXCP может выглядеть так:

```
find //server/disk/2017-05-04/logs -name '*.zip' -print0 | xargs -0 -i -n 1 unzip -p {} '*/FULL/rphost*/*.log' '*/FULL/rmngr*
/*.log' 2>/dev/null | grep ",EXCP," | head
```

В случае работы с архивами с типом, отличным от zip, команда извлечения из архива и синтаксис будут отличаться. Ниже приведены примеры команд извлечения из архивов, созданных различными архиваторами:

```
unzip file.zip
gunzip file.gz
unrar x file.rar
bunzip2 file.bz2
uncompress file.Z
7z x file.7z
```

ГЛАВА 10

Методика расследования проблем при эксплуатации крупных систем

Базовые инструменты

Полезные ресурсы:

- Методика анализа источников высокой нагрузки на СУБД по данным DMV: http://kb.1c.ru/articleView.jsp?id=76.
- Методика анализа источников высокой нагрузки на СУБД по контекстам конфигурации: http://kb.1c.ru/articleView.jsp?id=105.
- Скотт Граннеман. «Linux. Карманный справочник».
- Джеффри Фридл. «Регулярные выражения».

Инструменты для анализа технологических журналов:

- Cygwin: https://cygwin.com/install.html.
- ActivePerl: http://www.activestate.com/activeperl.

Группировка событий CALL по контекстам по значению свойства Memory:

```
cat rphost*/*.log | perl -n -e 'if (/^\d\d:\d\d:\.\d+/) {$event =~ s/\n/<line>/g; print $event."\n"; $event = "";} $event .
= $_; END {print $event."\n"};' | grep ',Context' | sed -r 's/.*Context=([^,]+),.*,Memory=([-0-9]+),.*/\2;\1/g' | awk -F';' '{mas[$2]+=$1}
END {for (i in mas) {print mas[i] " = " i}}' | sort -rn
```

Топ запросов, создающих нагрузку на CPU на сервере СУБД за последний час:

```
SELECT
SUM(qs.max_elapsed_time) as elapsed_time,
SUM(qs.total_worker_time) as worker_time
into T1 FROM (
    select top 10000
    *
```

```
    from
    sys.dm_exec_query_stats qs
    where qs.last_execution_time > (CURRENT_TIMESTAMP  – '01:00:00.000')
    order by qs.last_execution_time desc
) as qs
;
select top 10000
(qs.max_elapsed_time) as elapsed_time,
(qs.total_worker_time) as worker_time,
qp.query_plan,
st.text,
dtb.name,
qs.*,
st.dbid
INTO T2
FROM
sys.dm_exec_query_stats qs
CROSS APPLY sys.dm_exec_query_plan(qs.plan_handle) qp
CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) st
left outer join sys.databases as dtb on st.dbid = dtb.database_id
where qs.last_execution_time > (CURRENT_TIMESTAMP  – '01:00:00.000')
order by qs.last_execution_time desc
;
select top 100
(T2.elapsed_time*100/T1.elapsed_time) as percent_elapsed_time,
(T2.worker_time*100/T1.worker_time) as percent_worker_time,
T2.*
from
T2 as T2
INNER JOIN T1 as T1
ON 1=1
order by T2.worker_time desc
;
drop table T2
;
drop table T1
;
```

Группировка событий EXCP по полю Descr:

```
cat rphost*/*.log | perl descr.pl | sort | uniq -c | sort -rn >> result.txt
```

Листинг descr.pl:

```
#!/usr/bin/perl
use strict;

my $event;
my %actions = (
   'EXCP' => [
        {
           'action' => sub {
                my ($event) =@_;
                my ($garbage, $context) = split /Descr=['|"]/, $event;
                my ($descr, $garbage) = split /'/, $context;
                $descr =~ s/|'//g;
                $descr =~ s/\s+/ /g;

                print "EXCP Descr $descr\n" if $descr;
           },
```

```
        },
    ],
);
while (<>) {
    $event=process_event($event) if /^\d\d:\d\d\.\d+/;
    $event .= $_;
}
process_event($event) if ($event =~ m/^\d\d:\d\d\.\d+/);

sub process_event($) {
    my ($event) = @_;
    return unless $event;
    foreach my $event_type ( keys %actions ) {
        next unless $event =~ /^[^,]+,$event_type,/;
        foreach my $issue ( @{ $actions{$event_type} }) {
            &{$issue->{action}}($event);
        }
    }
}
```

Группировка событий CALL по длительности по последней строке контекста:

```
cat rphost*/*.log | perl parse_call.pl | awk -F'-' '{dur[$2]+=$1;} END {for (i in dur) {print dur[i] "  – " i;}}' | sort -rn >> result.txt
```

Листинг parse_call.pl:

```
#!/usr/bin/perl

use strict;

my $event;
my %actions = (
    'CALL => [
        {
            'action' => sub {
                my ($event) =@_;
                my ($date, $garbage) = split /,CALL/, $event;
                $date =~ s/\d\d:\d\d\.\d+-//g;

                my ($garbage, $context) = split /Context="/, $event;
                if (!$context) {($garbage, $context) = split /Context='/, $event;}
                if (!$context) {($garbage, $context) = split /Context=/, $event;}

                if ($context =~ /\n/) {
                    my @mlc = split /\n/, $context;
                    $context = $mlc[$#mlc];
                }
                $context =~ s/^\s+|'$//g;
                $context =~ s/;'/;/g;

                if ($context) {print "$date-$context\n";}
                else {print "$date-<no context>\n";}
            },
        },
    ],
);

print "\n";
while (<>) {
    $event=process_event($event) if /^\d\d:\d\d\.\d+/;
```

```
    $event .= $_;
}
process_event($event) if ($event =~ m/^\d\d:\d\d\.\d+/);

sub process_event($) {
    my ($event) = @_;
    return unless $event;
    foreach my $event_type ( keys %actions ) {
        next unless $event =~ /^[^,]+,$event_type,/;
        foreach my $issue ( @{ $actions{$event_type} }) {
            &{$issue->{action}}($event);
        }
    }
}
```

Для анализа событий других типов нужно заменить в тексте скрипта CALL на название необходимого события.

Группировка событий CALL по длительности по первой строке контекста:

```
cat rphost*/*.log | perl parse_call.pl | awk -F'-' '{dur[$2]+=$1;} END {for (i in dur) {print dur[i] "  – " i;}}' | sort -rn >> result.txt
```

Листинг parse_call.pl:

```
#!/usr/bin/perl

use strict;

my $event;
my %actions = (
    'CALL' => [
        {
            'action' => sub {
                my ($event) =@_;
                my ($date, $garbage) = split /,CALL/, $event;
                $date =~ s/\d\d:\d\d\.\d+-//g;

                my ($garbage, $context) = split /Context="/, $event;
                if (!$context) {($garbage, $context) = split /Context='/, $event;}
                if (!$context) {($garbage, $context) = split /Context=/, $event;}

                if ($context =~ /\n/) {
                    my @mlc = split /\n/, $context;
                    $context = $mlc[0];
                        if (!$context) {
                                $context = $mlc[1];
                        }
                }
                $context =~ s/^\s+|'$//g;
                $context =~ s/;'/;/g;

                if ($context) {print "$date-$context\n";}
                else {print "$date-<no context>\n";}
            },
        },
    ],
);

print "\n";
while (<>) {
```

```
    $event=process_event($event) if /^\d\d:\d\d\.\d+/;
    $event .= $_;
}
process_event($event) if ($event =~ m/^\d\d:\d\d\.\d+/);

sub process_event($) {
    my ($event) = @_;
    return unless $event;
    foreach my $event_type ( keys %actions ) {
        next unless $event =~ /^[^,]+,$event_type,/;
        foreach my $issue ( @{ $actions{$event_type} }) {
            &{$issue->{action}}($event);
        }
    }
}
```

Для анализа событий других типов нужно заменить в тексте скрипта CALL на название необходимого события.

Локализация проблемы

В случае возникновения проблем с доступностью и производительностью первым шагом в расследовании является локализация проблемы. Необходимо выяснить, у всех ли пользователей возникает проблема, во всех ли базах/областях, при выполнении какого функционала.

Также необходима локализация проблемы в части узла системы, в котором она происходит. Требуется получать оценку со всех уровней за одно и то же воспроизведение (время).

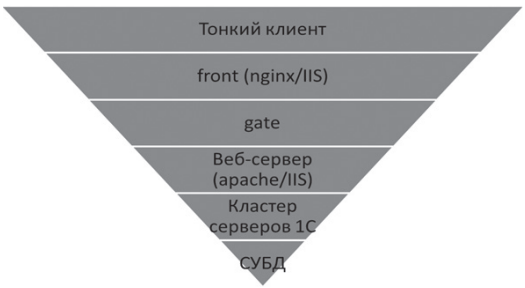


Рисунок 98. Узлы системы для анализа при локализации проблемы (на примере контура 1cFresh)

Проблемы производительности

Категории проблем

На первом шаге анализа необходимо понять, с проблемой какого рода вы встретились:

- Общая медленная работа сервиса у всех пользователей.

- Общая медленная работа сервиса у определенных пользователей.
- Медленная работа определенного функционала у всех пользователей либо у значительной части пользователей.
- Медленная работа определенного функционала у определенных пользователей.

Такую классификацию необходимо выполнить, так как причины проблем разного вида существенно различаются, так же как и методики их расследования.

Методы классификации

В процессе анализа необходимо учитывать совокупность показателей, среди которых:

- время выполнения операций, рассчитанное по различным методикам (например, Apdex или среднее время наихудших замеров);
- наличие жалоб от других пользователей;
- счетчики производительности (загруженности) оборудования;
- счетчики, встроенные в подсистему «ЦентрМониторинга»;
- воспроизводится ли проблема в тестовой базе (или в другой ноде) или в тестовой области в рабочей базе.

Для расследования проблем производительности в сервисе критически важно организовать централизованный сбор данных замеров производительности и агрегацию показателей Apdex. Для этого следует использовать *возможности ЦКК по импорту замеров производительности*.

Необходимость использования ЦКК для этой цели связана, во-первых, с возможностью горизонтального масштабирования сервиса, что приводит к тому, что один и тот же пользователь сервиса может работать параллельно в нескольких информационных базах. Как следствие, данные о замерах также хранятся во множестве информационных баз. Во-вторых, доступ к данным замеров в разделенной информационной базе требует наличия у специалиста, занимающегося вопросами производительности, административного доступа к данным, который в общем случае отсутствует. В-третьих, в ЦКК встроены намного более развитые средства для анализа замеров, чем в любые другие конфигурации.

Рассмотрим особенности каждого класса проблем подробно.

Общая медленная работа сервиса у всех пользователей

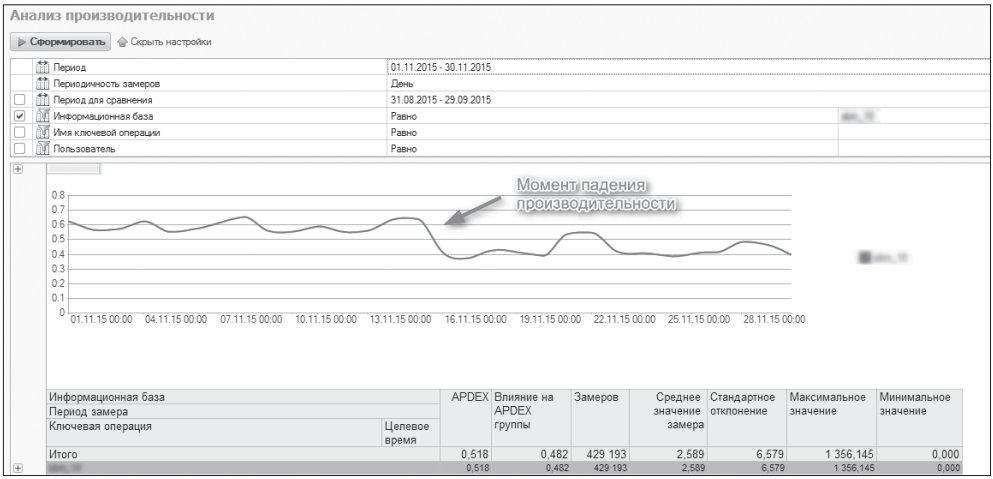
Характерным признаком этого класса является то, что пользователь не может указать на конкретную операцию, которая замедлилась, либо пользователь будет указывать на те операции, которые он выполняет чаще всего.

Для выявления того факта, что проблема воспроизводится у большого количества пользователей (массовая проблема), необходимо понять:

- *Есть ли аналогичные обращения от других пользователей*. Как правило, массовые проблемы с производительностью сервиса приводят к большому

количеству обращений, поэтому перед началом анализа имеет смысл выяснить, единично поступившее обращение или нет.

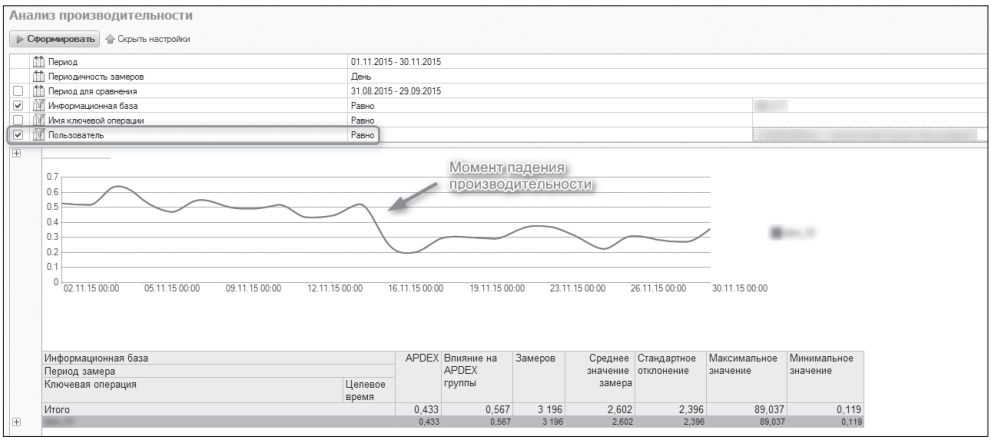
- *Как изменялся Apdex за последнее время*. Для этого необходимо проанализировать динамику Apdex и по системе в целом, и по обратившемуся пользователю. Рекомендуется использовать для этой цели отчет Анализ производительности ЦКК, сформировав его с отбором по отдельному пользователю, а затем без отбора. Например, факт резкого изменения производительности всего сервиса указывает на то, что в системе произошли какие-то изменения:



Информационная база / Период замера / Ключевая операция	Целевое время	APDEX	Влияние на APDEX группы	Замеров	Среднее значение замера	Стандартное отклонение	Максимальное значение	Минимальное значение
Итого		0,518	0,482	429 193	2,589	6,579	1 356,145	0,000
[redacted]		0,518	0,482	429 193	2,589	6,579	1 356,145	0,000

Рисунок 99. График Apdex за период по сервису в целом

Аналогичный график по конкретному пользователю с высокой долей вероятности указывает на то, что проблема этого пользователя заключается именно в проблемах сервиса в целом:



Информационная база / Период замера / Ключевая операция	Целевое время	APDEX	Влияние на APDEX группы	Замеров	Среднее значение замера	Стандартное отклонение	Максимальное значение	Минимальное значение
Итого		0,433	0,567	3 196	2,602	2,396	89,037	0,119
[redacted]		0,433	0,567	3 196	2,602	2,396	89,037	0,119

Рисунок 100. График Apdex за период по определенному пользователю

Помимо резкого изменения скорости работы сервиса может наблюдаться и постепенная деградация производительности. Для выявления таких проблем целесообразно анализировать более длительный период времени.

- Дополнительно имеет смысл мониторить среднее время выполнения определенного класса быстрых операций (т. е. выполняющихся в клиентском приложении без использования механизма длительных операций). В качестве такого класса операций может выступать, в частности, время проведения всех видов документов. Необходимость такого анализа связана с тем, что в некоторых случаях данные Apdex могут не показывать проблему. Например, если в системе постоянно выполняется большое количество ключевых операций с заведомо большим целевым временем, то в этом случае будет регистрироваться большое количество замеров с высоким значением Apdex. В результате проблема может быть замаскирована среди таких операций. Для выполнения анализа среднего времени выполнения операций можно воспользоваться формой мониторинга ЦКК:

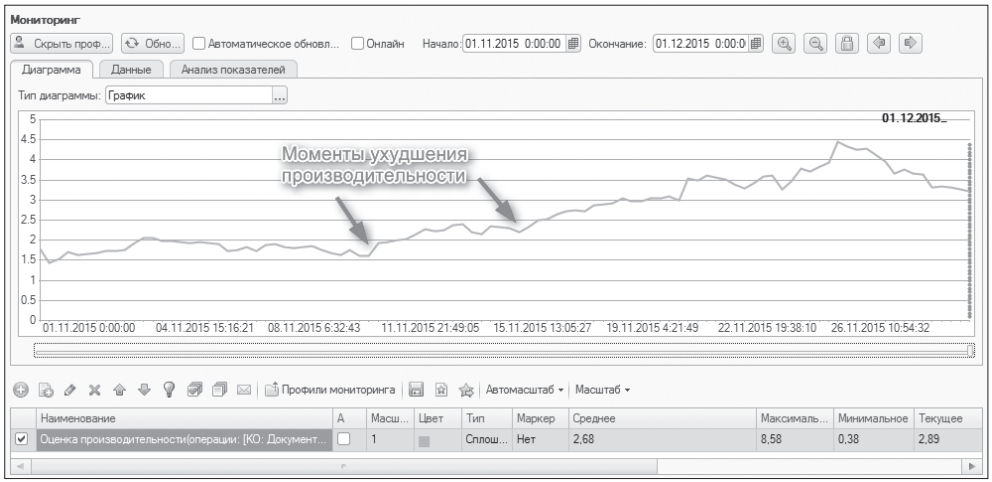


Рисунок 101. График среднего времени выполнения операций за период

Общая медленная работа сервиса у определенных пользователей

Симптомы проблем такого рода, на которые указывают пользователи, обычно полностью совпадают с симптомами предыдущей категории. Однако, в отличие от нее, данные Apdex по пользователю заметно отличаются от показателей сервиса в худшую сторону. Также для проблем этого вида не удается выявить факты деградации общей производительности сервиса, коррелирующие с моментом возникновения проблем у пользователя.

Медленная работа определенного функционала у всех пользователей либо у значительной части пользователей

Для этой категории проблем характерно указание конкретных сценариев при описании проблемы пользователем. Этот класс проблем диагностируется так же, как проблемы с общим замедлением сервиса, с тем лишь отличием, что при формировании отчетов требуется дополнительно установить отбор по конкретной ключевой операции.

Медленная работа определенного функционала у определенных пользователей

В случае если пользователь указывает на конкретный сценарий, в котором воспроизводится проблема, и по результатам анализа установлено, что она не является массовой, то проблему следует отнести к этой категории.

В этом случае требуется провести проверки и попытаться максимально точно воспроизвести медленную работу определенного функционала. Если проблема действительно в конкретной реализации функционала, то она должна воспроизвестись.

Типичные причины проблем производительности

Общая медленная работа сервиса у всех пользователей

Возможные причины:

- Резкая деградация производительности:
  - изменения в настройке оборудования;
  - изменения в конфигурации информационной базы;
  - изменения настроек программных продуктов.
- Плавная деградация производительности:
  - постепенно нарастающий дефицит вычислительных мощностей;
  - постепенно нарастающий дефицит пропускной способности дискового хранилища;
  - постепенно нарастающий дефицит пропускной способности сети;
  - деградация производительности неоптимальных запросов вследствие роста объема базы;
  - деградация производительности вследствие постепенного накопления изменений конфигурации информационной базы, негативно влияющих на производительность.

Если в результате анализа выявлено резкое падение производительности, то важно установить точный момент, когда произошла деградация. Опираясь на эту информацию, необходимо проанализировать изменения, произошедшие в системе. Например, среди таких изменений может быть обновление конфигурации информационной базы, включение нового регламентного задания, перемещение виртуального сервера на другое оборудование и т. д.

Если же деградация происходила постепенно, то необходимо понять, с чем связаны наблюдающиеся проблемы.

Для установления факта нехватки производительности оборудования следует организовать сбор сведений о загрузке оборудования. Для этих целей можно использовать как системные средства мониторинга (например, perfmon + logman при использовании ОС Windows), так и *возможности ЦКК по сбору сведений о производительности оборудования*. Подробнее о настройке мониторинга на рабочих серверах говорится в главе 3 данного пособия.

Диагностировать длительные запросы можно либо путем анализа технологических журналов по событиям DBMSSQL (или других, в зависимости от используемой СУБД) и SDBL, либо используя ЦУП. При использовании MSSQL также имеет смысл проанализировать, какие виды ожиданий чаще всего возникают на сервере (подробнее на https://kb.1c.ru/articleView.jsp?id=92).

Диагностировать длительные вызовы можно путем анализа технологических журналов по событиям CALL, SCALL, VRSREQUEST, VRSRESPONSE.

Также проблемы производительности могут постепенно накапливаться по мере перехода на новые версии конфигураций. Проверить факт такого изменения можно, проведя нагрузочное тестирование и сравнив производительность двух версий: актуальной и старой, во время работы на которой проблем производительности не наблюдалось. При этом важно проведение нагрузочных тестов в максимально стабильных условиях. Для определения возможности сравнения достаточно провести несколько тестов подряд и оценить разброс показателей производительности на текущем оборудовании и текущей конфигурации системы. Увеличение разброса свидетельствует об очередях на различных уровнях работы информационной системы.

Общая медленная работа сервиса у определенных пользователей

Есть две основных причины проблем такого рода:

- Нетипичное наполнение области:
  - большой объем данных;
  - нестандартные настройки.
- Проблемы подключения:
  - медленный компьютер;
  - низкая скорость интернет-канала;
  - нестабильный интернет-канал.

Для расследования таких проблем необходимо убедиться, что они не связаны с конкретными сценариями работы. Зачастую под общей медленной работой сервиса пользователи на самом деле имеют в виду медленную работу тех операций, которые они выполняют чаще всего. Поэтому необходимо подробно расспросить пользователя о тех сценариях, на которых воспроизводится проблема, и проверить воспроизведение на копии области.

Если пользователь затрудняется сказать, какие операции работают медленно, то нужно проверить те операции, которые пользователь чаще всего выполняет (по данным отчета «Анализ производительности» ЦКК).

Если проблему удалось воспроизвести, то для дальнейшего расследования следует воспользоваться методикой https://kb.1c.ru/articleView.jsp?id=73#медленное выполнение операции.

Если проблему воспроизвести не удалось, то, вероятно, она связана с особенностями подключения к сервису конкретного пользователя. Для диагностики проблем такого рода можно воспользоваться обработкой Диагностика качества подключения, встроенной в менеджер сервиса.

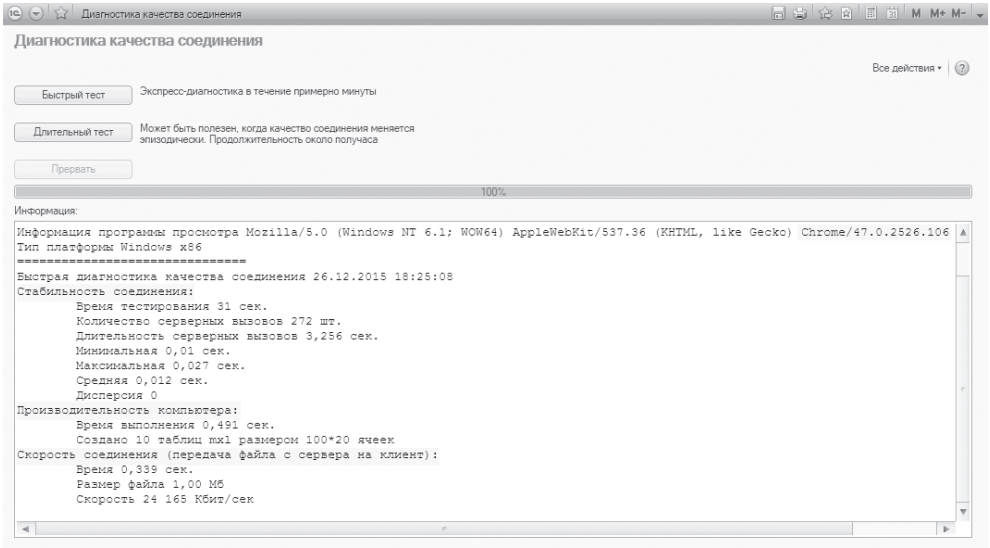


Рисунок 102. Обработка «Диагностика качества соединения»

Для обеспечения высокого качества работы сервиса необходимо:

- чтобы соединение обладало высокой стабильностью (до версии технологической платформы 8.3.6 включительно факт неполучения ответа от сервера приводит к зависанию тонкого клиента);
- соединение имело хорошую скорость (рекомендуется не менее 2 Мбит);
- компьютер обладал достаточной производительностью. Конкретное значение определяется эмпирически и зависит от типа браузера, использовавшегося для выполнения теста; значение на рис. 43 (0,491) означает хорошую производительность компьютера, в случае если оно получено при использовании Google Chrome. Значения более 1–1,5 с (в случае использования Google Chrome) обычно указывают на недостаточную производительность клиентского компьютера.

Если пользователь использует медленный компьютер, то для решения проблем производительности рекомендуется использовать тонкий клиент. Если для пользователя принципиально использование браузера, то для достижения наилучшей производительности рекомендуется использование Google Chrome.

Медленная работа определенного функционала у всех пользователей либо у значительной части пользователей

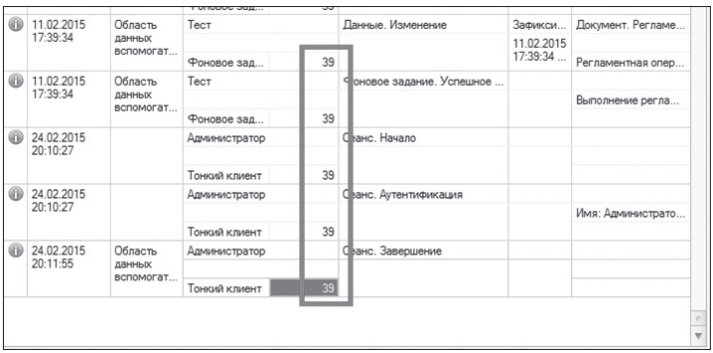
В случае если проблема воспроизводится у всех пользователей в определенном функционале, то для расследования необходимо попробовать воспроизвести ее на тестовой области данных. Если проблему на имеющихся данных воспроизвести не удается, то необходимо получить данные области одного из тех пользователей, у которых проблема точно воспроизводится, и проверить воспроизведение на этих данных.

Обычно массовые проблемы достаточно легко воспроизводятся, после чего можно приступить к расследованию. Методика расследования: https://kb.1c.ru/articleView.jsp?id=73#медленное выполнение операции.

Медленная работа определенного функционала у определенных пользователей

Проблемы такого рода относятся к числу наиболее трудных для расследования. Рекомендуемая последовательность анализа следующая:

- Получить максимально подробное описание проблемы от пользователя.
- Попробовать воспроизвести проблему на копии области пользователя. Если воспроизвести удалось, то выполнить анализ с помощью методики https://kb.1c.ru/articleView.jsp?id=73#медленное выполнение операции.
- Если воспроизвести не удалось, то настроить технологический журнал с событиями DBMSSQL (или другими, в зависимости от используемой СУБД), CALL и SCALL.
- Попросить пользователя воспроизвести операцию и засечь время начала выполнения и длительность операции по секундомеру. Также это можно выполнить самостоятельно на компьютере пользователя, если проблема воспроизводится «прямо сейчас» и пользователь согласен на удаленное подключение к его компьютеру.
- Если операция относится к числу ключевых, то уточнить точное время начала и окончания операции по данным замеров производительности.
- Выяснить номер сеанса по данным журнала регистрации.



11.02.2015 17:39:34	Область данных вспомогат...	Тест		Данные. Изменение	Зафикси... 11.02.2015 17:39:34 ...	Документ. Регламе...
		Фоновое зад...	39			Регламентная опер...
11.02.2015 17:39:34	Область данных вспомогат...	Тест		Фоновое задание. Успешное ...		Выполнение регла...
		Фоновое зад...	39			
24.02.2015 20:10:27		Администратор		Сеанс. Начало		
		Тонкий клиент	39			
24.02.2015 20:10:27		Администратор		Сеанс. Аутентификация		Имя: Администрато...
		Тонкий клиент	39			
24.02.2015 20:11:55	Область данных вспомогат...	Администратор		Сеанс. Завершение		
		Тонкий клиент	39			

Рисунок 103. Уточнение номера сеанса по журналу регистрации

- Отфильтровать собранный технологический журнал по номеру сеанса и типу события, используя следующую команду:

```
cat <ПутьКЖурналам>/*/* | perl -n -e 'if (/^\d\d:\d\d\.\d+/) {$event =~ s/\n/<line>/g; print $event."\n"; $event = "";} $event .= $_;
END{print $event."\n"};' | grep "SessionID=<НомерСеанса>" | grep ",<ТипСобытия>," | sed 's/<line>/\n/g' > <ТипСобытия>.txt
```

  где:

  - <ПутьКЖурналам> – путь к каталогу с собранными технологическими журналами;
  - <НомерСеанса> – номер сеанса, в котором выполнялась анализируемая операция;
  - <ТипСобытия> – тип события (CALL, DBMSSQL или SDBL).
  - Предварительно на компьютере, на котором выполняется анализ, должен быть установлен cygwin и perl. Скрипт должен выполняться из-под командной оболочки cygwin. Отфильтрованный журнал будет сохранен в файл <ТипСобытия>.txt.

- Посчитать суммарную длительность серверных вызовов (события CALL) и суммарное время выполнения запросов (DBMSSQL, SDBL), используя следующую команду:

```
cat <ТипСобытия.txt> | awk -F'-' '{mas[$2] += $1; count[$2] += 1;} END {for (i in mas) {print mas[i] " " count[i] " " i};}' | sort -rn >
<ТипСобытия_Итого>.txtгде <ТипСобытия>  – тип события (CALL, DBMSSQL или SDBL)
```

- На основе собранной информации попытаться понять, в чем заключается проблема. Например, если очевидно, что проблема связана с длительным временем выполнения запросов, то проанализировать тексты и, при необходимости, планы выполнявшихся запросов (возможно, потребуется собрать дополнительно).

ГЛАВА 11

Методики разработки высоконагруженных систем на платформе «1С:Предприятие 8»

Когда речь заходит о неухудшении качества эксплуатируемой крупной системы, одну из центральных ролей начинает играть качество программного кода используемых прикладных решений. В данной главе попробуем выделить основные подходы к разработке прикладных решений в контексте их эксплуатации и обеспечения технологического качества, расставим акценты и выделим особенности, критичные для облачных решений. Также будут приведены основные стандарты разработки из документации к платформе «1С:Предприятие», которые мы используем для разработки высокопроизводительных решений.

Особенности разработки в облаке

Когда речь заходит о разработке конфигураций, поддерживающих разделение данных, действуют все те же правила и рекомендации в части оптимизации производительности, что и для «обычных» систем. Но есть ряд особенностей, которые нужно учитывать.

Давайте представим сначала, что такое разделенная база. По сути это N маленьких баз, объединенных в одну (число N может достигать нескольких тысяч, оно ограничено только регламентами эксплуатации на конкретном внедрении и временем обновления), на одном оборудовании. При этом предполагается, что объем занимаемых аппаратных ресурсов для облачной базы не должен быть эквивалентен тем ресурсам, на которых работали все тысячи локальных неразделенных баз, которые со временем переехали в облако.

И в данном случае эффективное использование этих общих аппаратных ресурсов напрямую влияет на качество работы системы. Продемонстрируем это на простых примерах.

Пример 1. В конфигурации есть «тяжелый» отчет. При формировании отчета rphost забирает ~500 Mb памяти.

В обычной клиент-серверной базе это не страшно, так как отчет формирует только главный бухгалтер (он на предприятии один) два-три раза в день и можно потерпеть.

В облачной базе 4000 областей, одновременно работают 500 пользователей, все эти пользователи – главные бухгалтеры. 500 * 0.5 Gb = ~250 Gb требуемой памяти… из имеющихся 32 Gb, которых в обычных условиях хватает с избытком.

Что будет происходить в этом случае с сервером, мы помним из предыдущих глав. В корректно настроенном кластере серверов «1С:Предприятия 8» это вызовет перезапуск rphost по превышению допустимого объема памяти. Но, учитывая заданное условие, что пользователи работают одновременно, они израсходуют всю доступную оперативную память быстрее, чем механизм кластера выполнить штатный перезапуск.

Вывод: «тяжелые» операции в разделенной базе требуют гораздо большего внимания в части оптимизации по сравнению с неразделенной.

Пример 2. В конфигурации есть справочник, который содержит классификатор банков. Совсем маленький – примерно 150 Mb. Этот классификатор сделали разделенным просто потому, что так показалось удобнее разработчику.

В обычной клиент-серверной базе таблица займет 150 Mb. В облачной базе 4000 областей. Таблица займет 4000 * 150 Mb = 600 Gb, из которых полезной нагрузкой будет 1/4000 и еще ~600 Gb мусора, который будет занимать место на сервере СУБД и в сетевом хранилище, куда мы помещаем сохраненные бэкапы.

Вывод: использование разделения данных должно быть обоснованным, только для тех объектов, где для каждой области данные действительно должны быть различны и изолированны.

Пример 3. В конфигурации создано регламентное задание, которое сделали разделенным.

Запуск регламентного задания выполняется со значениями разделителей, которые заданы для этого задания в свойстве РазделениеДанных объекта РегламентноеЗадание. Если регламентное задание отмечено как предопределенное, при первом входе с новой комбинацией разделителей будет создано регламентное задание (в информационной базе) с текущей комбинацией разделителей. Расписание регламентного задания настроено в метаданных. Созданным в информационной базе регламентным заданиям устанавливается единое расписание. Что произойдет с производительностью?

Сервер: 12 ядер, 32 Gb памяти. Обслуживает информационную базу с 4000 областей. Длительность выполнения задания – 1 секунда – очень быстро. В один момент времени выполняется запуск фоновых заданий, порожденных регламентными заданиями с определенной комбинацией разделителей. Мы знаем, что работа одного потока (фонового задания) занимает одно ядро процессора. То есть для нашей ситуации получим время выполнения:

*4000 * 1 сек. /12 ядер =~ 300 секунд.*

А если вспомнить, что на том же сервере могут работать еще и сеансы пользователей, будет очевидно: наличие такого разделенного регламентного задания создаст предпосылки для возникновения очередей при обращении к процессору и как следствие – замедление как при выполнении самого задания, так и при работе пользователей.

Вывод:

Для разделенных прикладных решений в целом:

- Не рекомендуется создавать разделенные регламентные задания, которые выполняются в большом количестве областей данных.
- Рекомендуется создавать неразделенное регламентное задание, которое будет само обрабатывать разные значения разделителей.

Для прикладных решений, ориентированных на работу в режиме сервиса по технологии 1cFresh:

- Не должно быть определено регламентных заданий, которые включены в состав любого из разделителей.
- Необходимо использовать подсистему БСП «Очередь заданий» либо разработать аналогичный механизм очереди заданий самостоятельно.

Аудит дополнительных отчетов, обработок, расширений в 1cFresh

При переходе на работу в облако клиент/пользователь получает ряд неоспоримых преимуществ, в первую очередь в части расходов на закупку и модернизацию собственных аппаратных мощностей и программного обеспечения, обслуживание и администрирование, на покупку лицензий. При этом клиенту приходится соблюдать определенные правила, вызванные особенностями «проживания» в облаке, одно из которых – единая конфигурация прикладного решения без возможности нетиповой доработки под свои нужды.

Но в технологии 1cFresh для целей расширения функциональности конкретной области предусмотрено подключение дополнительных отчетов, обработок и расширений.

С учетом высоких требований к качеству кода таких дополнительных объектов сформировался ряд рекомендаций, на основании которых проводится их аудит на пригодность к встраиванию в облако.

Основные правила для дополнительных отчетов и обработок

Использование безопасного режима

Все расширения конфигурации, дополнительные отчеты/обработки выполняются в сервисе в безопасном режиме. Запрещенные операции:

- Выполнение методов Выполнить() и Вычислить().
- Привилегированный режим.
- Работа с внешними компонентами.
- Работа с файловой системой (кроме временных файлов).
- COM-объекты (серверный код может выполняться на Linux, мы не можем гарантировать наличие COM-объектов на сервере).
- Доступ к сети Интернет.

Рекомендации по использованию безопасного режима:

- Учтите, что небезопасные операции запрещены только на сервере!
- Используйте механизм расширения безопасного режима БСП и сценарным выполнением дополнительных отчетов и обработок.
- Опишите используемые расширения безопасного режима в свойствах расширения, параметрах регистрации дополнительного отчета/обработки.
- При отладке дополнительного отчета или обработки открывайте их не через главное меню: Файл – Открыть, а через интерфейс подсистемы дополнительных отчетов и обработок БСП. Открывайте и проверяйте обработку не с правами администратора, а с набором прав тех пользователей, которые будут использовать обработку.

Вопросы производительности

Расширения конфигурации, дополнительные отчеты и обработки не должны вызывать деградацию производительности сервиса. Поэтому используемые запросы на встроенном языке должны быть оптимально построены и соответствовать стандартам. Типичные причины неоптимальной работы запросов и методы оптимизации запросов рассмотрены в статье 1С:ИТС (http://its.1c.ru/db/metod8dev#content:4050:hdoc) и далее в данной книге.

Особое внимание следует уделить оптимизации:

- соединений с виртуальными таблицами, подзапросами (они могут привести к значительному замедлению запроса);
- отборов с «ИЛИ»;
- запросов с получением данных через точку от полей составного ссылочного типа (при выполнении такого запроса будет выполняться соединение со всеми таблицами объектов, входящими в составной тип).

При написании расширения конфигурации, дополнительного отчета и обработки необходимо избегать кода, выполнение которого может привести к неоправданно высокому потреблению процессорного времени. В первую очередь это касается «пустых» циклов.

Универсальные обработки и расширения

Хорошей практикой является разработка универсальных обработок и расширений, выполняющих общую, стандартную для различных прикладных конфигураций бизнес-функциональность.

При этом следует избегать универсальных расширений конфигурации и дополнительных обработок, не учитывающих бизнес-логику конкретного прикладного решения, особенно если такие обработки или расширения могут изменять данные пользователей.

Универсальные расширения и обработки такого плана опасны по следующим причинам:

- для их использования может быть необходим высокий уровень квалификации пользователей, а его гарантировать невозможно;
- пользователь может изменить реквизиты служебных объектов или объектов, которые он не собирался менять;
- возможно нарушение бизнес-логики прикладного решения.

Примером «опасной» универсальной обработки, которую не стоит давать рядовым пользователям, могут являться обработки по массовому перепроведению документов, групповому изменению реквизитов и т. д.

Использование временных файлов

При доступе к временным файлам необходимо следовать требованиям стандартов, описанным по ссылке. В частности, неправильно писать:

```
ТемпФайл = "/tmp/temx.txt"
```

Правильно:

```
ТемпФайл = ПолучитьИмяВременногоФайла("txt")
```

При этом гарантируется уникальность имени временного файла и выполняется автоматическая очистка при рестарте рабочего процесса. Также необходимо обеспечить удаление временных файлов после использования.

Прочие вопросы

При разработке расширений конфигурации, дополнительных отчетов и обработок необходимо учитывать следующие требования:

- Длительные операции необходимо выполнять в фоновом режиме (подробнее см. по ссылке).

- Если пользовательские данные покидают сервис, то пользователь должен дать на это согласие.
- Не должна нарушаться штатная логика работы прикладных решений (конфигураций). Не следует отключать штатные механизмы и проверки, например с помощью конструкций вида:

```
ОбменДанными.Загрузка = Истина
```

- Не следует скрывать от пользователя системную информацию об ошибках, полученную в операторе Попытка – Исключение. Без этой информации службе поддержки сервиса будет значительно труднее разбираться в причинах возникновения ошибки.

Оптимизация использования оперативной памяти

Оперативная память – ценный ограниченный ресурс. В многопользовательских, а особенно разделенных (облачных), системах неэффективное использование оперативной памяти может стать для работоспособности системы фатальным. Для корректной работы с памятью важно понимать, как именно с ней работает платформа, поэтому уделим оперативной памяти повышенное внимание.

Представление данных в памяти

Платформа «1С:Предприятие 8» в памяти состоит из множества различных объектов. Часть этих объектов доступны из встроенного языка «1С:Предприятия 8», остальные нужны для функционирования внутренних механизмов платформы.

Важно запомнить одну простую вещь: все, что доступно из языка конфигурации, является объектами (даже значения примитивных типов, например, числа). Соответственно, базовая логика работы с объектами в системе подчиняется правилам работы с объектами объектно-ориентированных языков программирования.

Объекты могут ссылаться друг на друга. Например, в массиве могут содержаться ссылки на другие объекты. Хранение ссылки на объект с точки зрения занимаемой памяти гораздо экономичнее, чем создание второго такого же объекта в памяти.

Продемонстрируем это утверждение на примере. Напишем две процедуры, выполняющие с точки зрения прикладной логики одно и то же действие – создающих два одинаковых массива, содержащих числа:

```
&НаСервере
Функция СоздатьМассивЧисел(Размер = 10000)
        Результат = Новый Массив(Размер);
        Для Индекс = 1 По Размер цикл
                Результат[Индекс-1] = Индекс;
        КонецЦикла;
        Возврат Результат;
КонецФункции

&НаСервере
Процедура СоздатьМассивЧиселДваРаза()
```

```
        Массив1  = СоздатьМассивЧисел();
        Массив2 = СоздатьМассивЧисел();
КонецПроцедуры

&НаСервере
Процедура КлонироватьМассивЧисел()
        Исходный = СоздатьМассивЧисел();

        Клон = Новый Массив(Исходный.Количество());
        Для Индекс = 0 по Исходный.Количество()-1 цикл
                Клон[Индекс] = Исходный[Индекс];
        КонецЦикла;
КонецПроцедуры

&НаКлиенте
Процедура Тест1(Команда)
        СоздатьМассивЧиселДваРаза();
КонецПроцедуры

&НаКлиенте
Процедура Тест2(Команда)
        КлонироватьМассивЧисел();
КонецПроцедуры
```

Команда Тест1(), в процессе выполнения которой массив создавался дважды, перед окончанием серверного вызова показала расход памяти *~2,5 Мб*.

Инф. база	Номер сеанса	Память (текущая)
aasatryan_bgu	4	2 650 246

Рисунок 104. Результат выполнения команды «Тест1()» в консоли кластера

Здесь мы фактически два раза создали по 10 000 объектов типа Число (на самом деле чуть меньше, так как объекты чисел 0–9 кешируются) и поместили их в два массива.

Команда Тест2(), в процессе выполнения которой второй массив создавался путем добавления ссылок на элементы первого массива, перед окончанием серверного вызова показала расход памяти *~1,5 Мб*.

Инф. база	Номер сеанса	Память (текущая)
aasatryan_bgu	4	1 663 396

Рисунок 105. Результат выполнения команды «Тест1()» в консоли кластера

Даже на таком простом примере видна логика использования памяти на уровне объектов, и можно сделать выводы о подходах к разработке и оптимизации прикладных решений, которые позволят расходовать память более эффективно.

Пойдем дальше. Поговорим о **мутабельных** и **немутабельных** объектах. За этими терминами скрываются изменяемые и неизменяемые объекты соответственно. Все значения примитивных типов (Строка, Число, Булево, Тип) являются немутабельными. Все остальные типы, как правило, мутабельные (объекты этих типов могут менять свое состояние).

С самой природой **немутабельных** объектов связана особенность их использования в части потребления оперативной памяти. Немутабельный объект не может быть изменен, причем не только на уровне прикладной конфигурации, но и на уровне платформы.

Преимущества такого поведения:

- меньшая подверженность кода ошибкам;
- безопасность для многопоточного использования;
- уменьшение потребления памяти.

Недостатком такой неизменяемости является избыточное создание временных копий объектов в некоторых сценариях. Характерный пример такой избыточности, приводящей к неоптимальному использованию памяти, а в некоторых сценариях к замедлениям выполнения кода прикладного решения, – сложение строк (конкатенация):

```
Функция Тест(Строка1, Строка2, Строка3)
        Результат = Строка1 + Строка2 + Строка3;
        Возврат Результат;
КонецФункции
```

Давайте внимательно посмотрим на этот простой кусок кода и, используя наши знания об особенностях немутабельных объектов, попробуем оценить, как будет использоваться оперативная память.

Мы помним, что результирующая переменная Результат немутабельного типа Строка. То есть нельзя записать в переменную Результат значение Строка1+Строка2, а потом дополнить его значением Строка3. А значит, нам не обойтись без промежуточных значений в памяти, необходимых для хранения промежуточных результатов сложения. Из этих рассуждений получается следующее:

- область памяти 1 занята объектом Строка1;
- область памяти 2 занята объектом Строка2;
- область памяти 3 занята объектом Строка3;
- область памяти 4 будет занята объектом, созданным при сложении копий объектов Строка1 и Строка2 (еще не помещая в Результат);
- область памяти 5 будет занята объектом Результат, полученным от сложения копий объекта, полученного на предыдущем шаге, и Строка3.

Вот такая нетривиальная логика для простого сложения строк: операции копирования памяти и временная строка. Очевидно, что чем больше слагаемых, тем больше временных строк будет создаваться и памяти расходоваться.

Особенно явный негативный эффект такого поведения заметен при конкатенации больших строк – например, при формировании по частям текста длинных запросов в коде прикладных решений. Кроме большого объема занимаемой памяти возможны замедления, связанные с выделением (аллоцированием) областей памяти для помещения копий объектов. При интенсивном ее использовании память сильно фрагментирована, и на выделение памяти под большой объект может тратиться достаточно ощутимое время.

В актуальных версиях платформы для сложения строк рекомендуется использовать метод СтрСоединить(), принимающий в качестве входного параметра массив складываемых строк. Данный метод оптимизирован в части использования памяти. Но при его использовании нужно учитывать, что для маленьких строк накладные расходы на создание исходного массива могут превысить эффект от самого оптимизированного сложения строк.

Объем памяти, занимаемой объектами

Как подсчитать объем памяти, занимаемый объектом? Точно – никак, так как размер объекта зависит от множества факторов. Но, например, для примитивных типов можно указать примерные условия. Для составных типов можно ориентироваться на количество хранимых ссылок на другие объекты. Для лучшего понимания использования памяти в объектной модели приведем несколько примеров того, как выделяется память под наиболее часто используемые объекты.

Булево

- Размер – 20 байт для x86 и 40 байт для x64.
- Есть всего два объекта на процесс ОС – Истина и Ложь.

Число

- Размер – 52 байта для x86 и 72 байта для x64.
- Плюс дополнительная память для чисел, не помещающихся в число двойной точности по стандарту IEEE 754.
- Есть кеш для целых чисел от 0 до 9.

Строка

- Размер – 40 байт для x86 и 64 байта для x64.
- Может хранить короткие строки внутри себя (до 9 символов).
- Для длинных строк используется дополнительная память размером 2*N символов.

Дата

- Размер – 24 байта для x86 и 40 байт для x64.

Массив

- Размер – 42 байта для x86 и 104 байта для x64.
- Дополнительно – количество элементов, умноженное на размер ссылки (8 байт для x86 и 16 байт для x64).
- Если в массив добавляли элементы с помощью метода Добавить(), то дополнительное количество памяти умножаем на 1,5.

Соответствие

- Размер – 68 байт для x86 и 136 байт для x64.
- Дополнительно – количество элементов, умноженное на размер ссылки (16 байт для x86 и 32 байта для x64).

Управление временем жизни объектов

Для правильного управления жизнью объектов платформы необходимо знать, когда их можно безопасно удалить из памяти. Для этого используется алгоритм подсчета ссылок на объекты. Такой же, например, используется в языке Swift.

Это сборщик мусора или нет? В ранних работах по computer science подсчет ссылок считали одним из вариантов сборщика мусора. В современной интерпретации под сборщиком мусора в основном подразумевают трассировочные алгоритмы, такие как в Java или .NET.

Работа алгоритма подсчета ссылок на объекты:

- У каждого объекта есть целочисленный счетчик, в котором хранится количество ссылок на него.
- Когда где-то сохраняется ссылка на объект, счетчик увеличивается на единицу, когда очищается ссылка – уменьшается на единицу.
- Когда значение счетчика становится равным 0, объект автоматически удаляется.
- Ссылки на объекты могут храниться как в коде конфигурации (локальные переменные на стеке), так и в других объектах.
- Также есть более долговременные хранилища объектов. Например, кеш параметров сеанса, повторно используемые возвращаемые значения, временное хранилище.

Недостаток алгоритма подсчета ссылок: невозможно автоматически разорвать циклическую ссылку на объект (объекты).

Циклические ссылки

Чем вредна циклическая ссылка?

- Возникает утечка памяти – объект (объекты) невозможно удалить.
- Также все объекты, на которые явно или неявно ссылается зацикленный объект, будут находиться в памяти до окончания работы процесса операционной системы.

Утечки памяти – это не только дополнительно занимаемая память в процессе:

- Например, на клиенте есть утечка управляемой формы.
- Данные формы на сервере не будут освобождены.
- Не снимется блокировка объекта базы данных, если его редактировали.

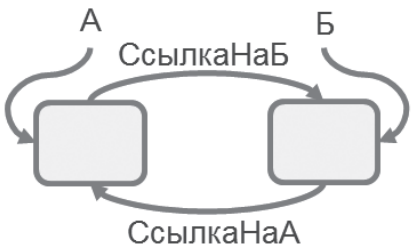


Рисунок 106. Утечка памяти

Как определять утечки памяти?

- По системным счетчикам потребляемой процессом памяти:
  - Хорошо подходит счетчик working set (рабочий набор).
  - Можно отслеживать изменение счетчика при многократном повторении одной и той же операции (например, открытие формы).
- По технологическому журналу, с помощью событий LEAKS и MEM:
  - Но учтите, что объекты могут оказаться в различных кешах.
  - Например, в кеше повторного использования возвращаемых значений.
  - Такие ситуации не являются утечками памяти.

Как находить циклические ссылки?

- По технологическому журналу, с помощью событий LEAKS.
- В 8.3.10 – новый инструмент диагностики.
- Простой lifehack, но работает далеко не во всех случаях:
  - Попробуйте ЗначениеВСтрокуВнутр().
  - Все взорвется, если попробуете подсунуть зацикленный объект.
  - Использовать только для отладки кода!

Модель данных в памяти

Данные платформы и конфигурации в памяти можно классифицировать по следующим признакам:

- время жизни данных;
- область видимости данных;
- предельный объем занимаемой памяти;
- способ инициализации данных.

Все перечисленные выше признаки в итоге влияют на общий объем потребляемой памяти.

Время жизни данных

- Постоянное:
  - Пока существует хоть один сеанс, связанный с информационной базой.
  - Такие данные нельзя просто взять и удалить из памяти.
  - Пример – метаданные конфигурации.
- Переменное (кеши платформы):
  - Пока есть свободная память или не прошло определенное время с момента инициализации данных.
  - Можно безопасно удалить из памяти в любой момент времени.
  - Пример – повторное использование возвращаемых значений.

- Локальное:
  - Живут во время вызова кода конфигурации.
  - Пример – временные объекты, созданные в коде конфигурации.

Область видимости данных

- Информационная база:
  - Данные доступны всем сеансам.
  - Пример – метаданные конфигурации.
- Сеанс:
  - В каждом сеансе свои копии данных.
  - Пример – объект базы данных (справочник, документ и т. п.).
- Прочие области (не столь важные для рассмотрения):
  - Область данных.
  - Пользователь.
  - Транзакция.

Предельный объем занимаемой памяти

- Ограниченный:
  - Может занимать много, но всегда есть некоторый предел.
  - В основном это те данные, которые зависят от метаданных конфигурации.
- Неограниченный:
  - Объекты, создаваемые в коде конфигурации.
  - Утечки памяти.

Способ инициализации данных

- Инициализация всех данных в памяти:
  - Метаданные конфигурации на сервере без режима отладки.
  - Отложенная инициализация данных в памяти:
  - Метаданные конфигурации в клиенте файловой базы.
  - Повторное использование возвращаемых значений.

В общем потребление памяти зависит от:

- Количества загруженных информационных баз:
  - На каждую требуется память под метаданные и т. п.
- Количества сеансов:
  - Данные открытых управляемых форм.
  - Данные в кешах платформы.
- Варианта развертывания:
  - Разные способы инициализации данных.

- Используемой функциональности платформы:
  - Изменение элементов/реквизитов форм.
  - Чтение объектов в циклах.
  - Анализ метаданных из кода конфигурации.

Очевидное – невероятное

- Описание формы:
  - Находится в кеше информационной базы (доступно для всех сеансов).
  - При изменении формы описание попадает в сеансовый кеш.
  - С базой работают 1000 человек – потребление памяти на описание формы выросло в 1000 раз.
- Чтение объектов в цикле:
  - При чтении объекта он попадает в сеансовый кеш.
  - Пока работает код конфигурации, объект будет находиться в памяти.
  - При массовой обработке все объекты будут в памяти, даже если вы уже не ссылаетесь на них из кода.
- Работа с метаданными:
  - Клиент файловой базы использует отложенную загрузку метаданных:
    - чтобы быстрее запускалась конфигурация;
    - и потребляла меньше памяти.
  - Потому что у клиентов компьютеры слабенькие и памяти мало.
  - Перебирая все метаданные из кода конфигурации, особенно при старте, мы убиваем оптимизацию в платформе.
- Работа с запросами (уже рассказывали, вкратце повторимся):
  - Платформа кеширует промежуточное представление для некоторых запросов (динамический список, отчеты).
  - Генерация уникального запроса из кода конфигурации, по сути, отключает кеширование и ведет к постоянному росту памяти.

Правила эффективного использования памяти при разработке прикладных решений

Не следует разрабатывать решения исходя из неограниченного объема оперативной памяти. Для высоконагруженных многопользовательских, а особенно облачных, систем любое неэффективное использование памяти может катастрофически сказаться на работоспособности.

Следует избегать формирования больших структур данных в памяти. Если объем данных, с которыми работает бизнес-логика, сам по себе ничем не ограничен, его нужно ограничивать искусственно, обрабатывая данные порциями и сохраняя промежуточные результаты в базу или файлы.

При потенциально не ограниченных выборках данных из ИБ следует получать данные из базы порциями фиксированного размера.

Например, неправильно:

```
Запрос = Новый Запрос;
Запрос.Текст =
 "ВЫБРАТЬ
 | Номенклатура.Ссылка,
 | Номенклатура.Наименование,
 | Номенклатура.ВидНоменклатуры
 |ИЗ
 | Справочник.Номенклатура КАК Номенклатура";
// Выгрузка всего справочника в таблицу значений
Номенклатура = Запрос.Выполнить().Выгрузить();
Для каждого ПозицияНоменклатуры Из Номенклатура Цикл
 // Обработка элемента справочника
 // ...
КонецЦикла;
```

поскольку весь результат запроса сразу помещается в память, в таблицу значений. Также *неправильно*:

```
Запрос = Новый Запрос;
Запрос.Текст =
 "ВЫБРАТЬ
 | Номенклатура.Ссылка,
 | Номенклатура.Наименование,
 | Номенклатура.ВидНоменклатуры
 |ИЗ
 | Справочник.Номенклатура КАК Номенклатура";
РезультатЗапроса = Запрос.Выполнить();
// Обход результата запроса
ВыборкаДетальныеЗаписи = РезультатЗапроса.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
 // Обработка элемента выборки
 // ...
КонецЦикла;
```

поскольку и в этом случае при выполнении запроса его результат будет сначала считан в память целиком. Если используется 32-битная версия платформы и размер результата запроса превосходит размер имеющейся памяти, то данные будут записаны на диск, а затем считаны оттуда в процессе вызовов Выборка.Следующий().

Правильно ограничивать результат запроса искусственно:

```
ВсеОбработано = Ложь;
Пока Истина Цикл
 Запрос = Новый Запрос;
 Запрос.Текст =
  "ВЫБРАТЬ ПЕРВЫЕ 1000
  | Номенклатура.Ссылка,
  | Номенклатура.Наименование,
  | Номенклатура.ВидНоменклатуры
  |ИЗ
  | Справочник.Номенклатура КАК Номенклатура
  |ГДЕ
  | <условие выборки необработанных записей>";
 РезультатЗапроса = Запрос.Выполнить();
```

```
ВсеОбработано = РезультатЗапроса.Пустой();
Если ВсеОбработано Тогда
 Прервать;
КонецЕсли;
// Обход порции результата запроса
ВыборкаДетальныеЗаписи = РезультатЗапроса.Выбрать();
Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
 // Обработка элемента выборки
 // ...
КонецЦикла;
КонецЦикла;
```

Также правильно:

```
Выборка = Справочники.Номенклатура.Выбрать(..., Отбор);
Пока Выборка.Следующий() Цикл
 // Обработка элемента выборки
 // ...
КонецЦикла;
```

поскольку в этом случае платформа «1С:Предприятие» выполняет получение данных из базы порциями фиксированного размера.

Также недопустимо работать с большими XML-документами с помощью объектов встроенного языка, предназначенных для обработки файлов целиком:

- текстовые документы в ТекстовыйДокумент;
- XML в ДокументDOM;
- HTML в ДокументHTML.

В таком случае весь файл загружается в оперативную память целиком. Исключения составляют отдельные случаи, когда необходим произвольный доступ к содержимому файла, к какой-то конкретной его части.

Следует использовать объекты для последовательной записи и последовательного чтения: ЧтениеXML, ЧтениеТекста, ЗаписьXML, ЗаписьТекста, с помощью которых можно прочитать файл порциями и расходовать память экономно.

Методики разработки в части разграничения прав доступа

Проверка прав доступа

В случае большого количества ролей в конфигурации (от нескольких десятков) не рекомендуется использовать ролевую настройку видимости в элементах форм (просмотр и редактирование реквизитов по ролям, пользовательскую видимость полей формы по ролям, использование команд по ролям). Это может существенно усложнить работу с элементами управления из кода, а также понизить устойчивость кода к изменению состава ролей и снизить скорость отображения формы. Также подобная модификация формы в зависимости от прав пользователя значительно усложняет процесс функционального тестирования и расследования проблем.

Вместо этого следует придерживаться следующих подходов:

- при сильных различиях внешнего вида и функциональности формы в зависимости от наличия тех или иных ролей у пользователя – разрабатывать отдельные формы, специализированные под конкретный набор прав пользователя;
- при незначительных отличиях – выполнять проверку прав в коде. При этом следует иметь в виду, что программное управление видимостью может снизить скорость открытия формы, что нужно учитывать при выборе между предлагаемыми подходами.

Для повышения устойчивости кода к составу ролей в конфигурации следует использовать в коде не проверку наличия роли (метод РольДоступна), а проверку наличия права доступа (метод ПравоДоступа).

В тех случаях, когда роль не дает никаких прав на объекты метаданных, а служит только для определения того или иного дополнительного права, допустимо использовать метод РольДоступна.

Использование привилегированного режима

Привилегированный режим позволяет:

- выполнить операции с данными от лица пользователей, которым данные недоступны;
- ускорить работу, так как в привилегированном режиме не накладываются ограничения на доступ к данным.

Привилегированный режим следует использовать:

- когда требуется с логической точки зрения отключить проверку прав;
- когда допустимо отключить проверку прав, чтобы ускорить работу, и при этом работа с данными от лица пользователя логически не нарушает установленные для него права доступа.

Так, если регистры или иные объекты не должны быть доступны пользователю для чтения (формирование отчетов), просмотра (в интерфейсе), редактирования, стоит на уровне прав ограничивать доступ пользователя к данным объектам. Если при этом объекты используются в прикладной логике алгоритмов, стоит производить такие операции в привилегированном режиме.

В то же время неоправданное использование привилегированного режима может привести к проблемам безопасности пользовательских данных.

Потенциально опасны любые экспортные процедуры и функции, которые выполняют на сервере какие-либо действия с предварительной безусловной установкой привилегированного режима, так как это отключает проверку прав доступа текущего пользователя.

Включать привилегированный режим следует точечно, чтобы остальной код не терял возможности проверки прав пользователя. Для этого нужно:

- установить привилегированный режим точно перед выполнением действия;

- выполнить действие без проверки прав;
- отключить привилегированный режим сразу же после выполнения действия;
- продолжить выполнение кода в непривилегированном режиме.

Подробнее о методиках разграничения прав доступа при разработке прикладных решений можно прочитать в документации: https://its.1c.ru/db/v8std#browse:13:-1:65.

Использование параметров сеанса

Общие сведения

Параметры сеанса предназначены для хранения значений определенных типов для каждого клиентского сеанса на время его работы.

Не рекомендуется:

- использовать параметры сеанса для хранения значений, используемых исключительно в клиентской логике т. к. в клиент-серверном варианте «1С:Предприятия 8» параметры сеанса хранятся на сервере, любое их считывание или изменение в процессе работы на клиенте потребует дополнительного серверного вызова и увеличит объем передаваемых данных с клиента на сервер и обратно.
- использовать параметры сеанса для кеширования вычисленных значений, которые многократно используются в серверной бизнес-логике. В таких случаях следует определять функцию в серверном общем модуле с повторным использованием возвращаемых значений.

Установка параметров сеанса «по требованию»

Не следует производить инициализацию параметров сеанса при запуске программы, так как:

- не все параметры сеанса запрашиваются из кода конфигурации при запуске программы;
- при работе программы возможно намеренное обнуление значений параметров сеанса из кода на встроенном языке.

Параметры сеанса должны быть инициализированы в обработчике УстановкаПараметровСеанса модуля сеанса только в тот момент, когда к ним происходит первое обращение как к неустановленным.

Подробнее о параметрах сеанса и стандартах работы с ними можно прочитать в документации: https://its.1c.ru/db/v8std#content:2149184062:hdoc.

Рекомендации по разработке оптимальных запросов

Общие требования

Прежде чем перейти к более продвинутым методам оптимизации запросов, необходимо убедиться, что сам запрос адекватен решаемой задаче.

Следует минимизировать объем выборки таким образом, чтобы выбирать ровно те данные, которые требуются для решения задачи.

При работе с запросами нужно получать только необходимые данные. Например, если нужно получить значения конкретных полей, не следует выбирать все поля «на всякий случай» с помощью конструкции ВЫБРАТЬ * ИЗ …

Кроме того, в большинстве случаев следует минимизировать общее количество запросов к СУБД. Чаще всего (не всегда!) получение всей необходимой информации одним запросом «дешевле», чем получение данных частями.

С другой стороны, не следует пытаться любой ценой перенести выполнение задачи в СУБД. Это связано со спецификой работы оптимизатора запросов. СУБД обычно оптимизирует и выполняет простые запросы более эффективно, чем сложные.

Следует рассмотреть альтернативные меры:

- по подготовке различных (более простых, частных) текстов запроса в зависимости от предусловий и значений параметров запроса – вместо отправки в СУБД одного большого универсального запроса;
- по более эффективной постобработке данных, выбранных запросом из СУБД, на стороне сервера «1С:Предприятия» средствами встроенного языка.

При разработке запросов нужно быть уверенным, что они использует эффективные планы выполнения запросов. Для сложных запросов велика вероятность выбора СУБД неоптимального плана выполнения запроса.

Поэтому не следует неоправданно усложнять запрос. В первую очередь:

- Не следует добавлять вложенные запросы только для повышения читаемости.
- Избегать сложных условий соединения в предложении ГДЕ, в особенности содержащих подзапросы и конструкции ВЫБОР.
- Использовать в запросе минимально необходимое число таблиц. В зависимости от структуры таблиц уже и 5–7 таблиц в одном запросе может быть много (время, затрачиваемое оптимизатором СУБД на анализ запроса, растет нелинейно, в итоге получается плохой план выполнения).

Для того чтобы узнать, какой план выполнения запроса выбран оптимизатором СУБД, можно воспользоваться консолью запросов, технологическим журналом или средствами СУБД. Как правило, запрос сложный и будет плохо выполняться, если в скомпилированном плане выполнения запроса есть timeout warning, который означает, что оптимизатору СУБД не хватило времени на поиск наилучшего плана запроса.

Несоответствие индексов и условий запроса

Необходимо убедиться в том, что для всех условий, использованных в запросе, имеются подходящие индексы.

Условия используются в следующих секциях запроса:

- ВЫБРАТЬ … ИЗ … ГДЕ <условие>
- СОЕДИНЕНИЕ … ПО <условие>
- ВЫБРАТЬ … ИЗ <ВиртуальнаяТаблица>(, <условие>)
- ИМЕЮЩИЕ <условие>

Для каждого условия должен существовать подходящий индекс. Подходящим является индекс, удовлетворяющий следующим требованиям:

- Индекс содержит все поля, перечисленные в условии.
- Эти поля находятся в самом начале индекса.
- Эти поля идут подряд, то есть между ними не «вклиниваются» поля, не участвующие в условии запроса.

Если в структуре базы данных отсутствует индекс, удовлетворяющий всем перечисленным условиям, то для получения результата СУБД будет вынуждена сканировать таблицу или один из ее индексов. Это приведет к увеличению времени выполнения запроса, а также к возможному снижению параллельности системы, поскольку возрастет количество установленных блокировок.

При создании объекта метаданных «1С:Предприятие» автоматически создает индексы, которые должны подходить для работы большинства запросов. Если вы пытаетесь оптимизировать работу прикладной конфигурации, добавляя нестандартные индексы напрямую в базу данных, вы не правы, и не только с точки зрения лицензионного соглашения (его можно найти в каталоге установки версии платформы в папке licenses). При следующей реструктуризации платформа восстановит стандартные для нее индексы таблиц.

В тех случаях, когда автоматически созданных индексов недостаточно, можно дополнительно проиндексировать реквизиты объекта метаданных.

При этом реквизиты справочников и документов рекомендуется индексировать с дополнительным упорядочиванием. Такой индекс будет учитывать упорядочивание по основному представлению объекта, тем самым он будет эффективно использоваться, например, когда в списке установлен отбор по данному реквизиту, а сам список упорядочен по полям основного представления объекта.

Не следует создавать индексы «на всякий случай» или заведомо избыточные индексы, т. к. любой индекс – это, напрмер, дополнительный объем, занимаемый базой на диске, а также увеличение времени реструктуризации. В частности:

- не следует дополнительно индексировать первое измерение регистра, поскольку для поиска по значению первого измерения подходит основной индекс таблицы итогов, который автоматически создаст платформа;

- не следует создавать индексы по низкоселективным полям. Например, индексировать реквизит типа Булево имеет смысл, только если незначительная часть записей всегда будет иметь одно значение и в запросах всегда выбираются записи по этому значению.

Примеры

В конфигурации описан регистр накопления ТоварыНаСкладах:

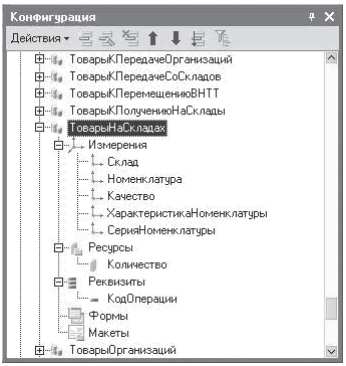


Рисунок 107. «ТоварыНаСкладах»

Платформа «1С:Предприятие» автоматически создаст для таблицы остатков данного регистра индекс по периоду и всем измерениям в том порядке, в котором они перечислены в конфигураторе.

Рассмотрим несколько примеров запросов и проанализируем, смогут ли они оптимально выполняться при такой структуре данных.

Запрос 1

```
Запрос.Текст = "ВЫБРАТЬ
| ТоварыНаСкладахОстатки.Склад,
| ТоварыНаСкладахОстатки.Номенклатура,
| ТоварыНаСкладахОстатки.Качество
|ИЗ
| РегистрНакопления.ТоварыНаСкладах.Остатки(, Номенклатура = &Номенклатура) КАК ТоварыНаСкладахОстатки";
```

В условии отсутствует отбор по первому полю индекса (Склад). Такой запрос не сможет выполниться оптимально.

Запрос 2

```
Запрос.Текст = "ВЫБРАТЬ
| ТоварыНаСкладахОстатки.Склад,
| ТоварыНаСкладахОстатки.Номенклатура,
| ТоварыНаСкладахОстатки.Качество
|ИЗ
| РегистрНакопления.ТоварыНаСкладах.Остатки(
| ,
| Качество = &Качество
| И Склад = &Склад) КАК ТоварыНаСкладахОстатки";
```

В данном случае между измерениями Склад и Качество в структуре регистра находится измерение Номенклатура, которое не задано в условии запроса. Этот запрос также не сможет выполняться оптимально.

Запрос 3

```
Запрос.Текст = "ВЫБРАТЬ
| ТоварыНаСкладахОстатки.Склад,
| ТоварыНаСкладахОстатки.Номенклатура,
| ТоварыНаСкладахОстатки.Качество,
| ТоварыНаСкладахОстатки.КоличествоОстаток
|ИЗ
| РегистрНакопления.ТоварыНаСкладах.Остатки(
| ,
| Номенклатура = &Номенклатура
| И Склад = &Склад) КАК ТоварыНаСкладахОстатки";
```

В этом случае требование соответствия индекса и запроса не нарушено. Данный запрос будет выполнен СУБД оптимальным способом.

Разыменование ссылочных полей составного типа в языке запросов

В языке запросов можно обращаться не только к полям исходных таблиц запроса, перечисленных в предложении ИЗ, но и к полям таблицы, на которую ссылается поле исходной таблицы запроса, если это поле имеет ссылочный тип. Имена полей при этом пишутся «через точку». Применение такой конструкции приводит к неявному соединению с дополнительными таблицами для получения значений полей «через точку».

Например, в запросе:

```
ВЫБРАТЬ
ТоварныеЗапасы.Товар КАК Товар,
ТоварныеЗапасы.Количество КАК Количество,
ТоварныеЗапасы.Товар.Артикул КАК Артикул
ИЗ
РегистрНакопления.ТоварныеЗапасы КАК ТоварныеЗапасы
...
```

кроме явно указанной в предложении ИЗ таблицы РегистрНакопления.ТоварныеЗапасы неявно участвует таблица Справочник.Товары для получения значения поля Артикул. А в случае использования ограничений доступа на уровне записей (RLS) к запросу добавляются еще и таблицы, участвующие в RLS к таблице Справочник.Товары.

Большое число исходных таблиц запроса приводит к его усложнению и может значительно увеличивать время его выполнения. Особенно это важно помнить в тех случаях, когда поле таблицы ссылочного типа имеет составной тип и может содержать ссылки на несколько таблиц. В таком случае получение полей других таблиц «через точку» от такого поля составного типа приведет к соединению со всеми таблицами, ссылки на которые могут оказаться в данном поле и в RLS к этим таблицам.

Получение данных «через точку» от ссылочных полей составного типа крайне нежелательно. Необходимо указывать ровно столько возможных типов для данного поля, сколько необходимо. Не следует без необходимости использовать типы «любая ссылка», «ссылка на любой документ» и т. п. Кроме усложнения запросов это также негативно скажется на времени реструктуризации.

Ограничения на соединения с вложенными запросами и виртуальными таблицами

При написании запросов не следует использовать соединения с вложенными запросами. Следует соединять друг с другом только объекты метаданных или временные таблицы. Если запрос использует соединения с вложенными запросами, то его следует переписать с использованием временных таблиц (неважно, с какой стороны соединения находится вложенный запрос).

Если запрос содержит соединения с вложенными запросами, то это может привести к следующим негативным последствиям:

- Крайне медленное выполнение запроса при слабой загрузке серверного оборудования. Замедление запроса может быть очень значительным (до нескольких порядков).
- Нестабильная работа запроса. При некоторых условиях запрос может работать достаточно быстро, при других – очень медленно.
- Значительная разница по времени выполнения запроса на разных СУБД.
- Повышенная чувствительность запроса к актуальности и полноте статистик. Сразу после полного обновления статистик запрос может работать быстро, но через некоторое время опять замедлиться.

Пример потенциально опасного запроса, использующего соединение с вложенным запросом:

```
ВЫБРАТЬ ...
ИЗ Документ.РеализацияТоваровУслуг
ЛЕВОЕ СОЕДИНЕНИЕ (
  ВЫБРАТЬ ИЗ РегистрСведений.Лимиты
  ГДЕ ...
  СГРУППИРОВАТЬ ПО ...
) ПО ...
```

Оптимизатор сервера СУБД (независимо от того, какую СУБД вы используете) не всегда может правильно оптимизировать подобный запрос. В данном случае проблемой для оптимизатора является выбор правильного способа соединения. Существуют несколько алгоритмов соединения двух выборок. Выбор того или иного алгоритма зависит от того, сколько записей будет содержаться в одной и в другой выборке. В том случае, если вы соединяете две физические таблицы, СУБД может легко определить объем обеих выборок на основании имеющейся статистики. Если же одна из соединяемых выборок представляет собой вложенный запрос, то понять, какое количество записей она вернет, становится очень сложно. В этом случае СУБД может ошибиться с выбором плана, что приведет к катастрофическому падению производительности запроса.

При этом необходимо проиндексировать созданную временную таблицу. В качестве индексных полей следует указать все поля, которые используются в условии соединения.

Для вышеприведенного примера получится следующий пакетный запрос:

```
// Создать менеджер временных таблиц
МенеджерВТ = Новый МенеджерВременныхТаблиц;
Запрос = Новый Запрос;
Запрос.МенеджерВременныхТаблиц = МенеджерВТ;
// Текст пакетного запроса
Запрос.Текст = "
  // Заполняем временную таблицу. Запрос к регистру лимитов.
  | ВЫБРАТЬ ...
  | ПОМЕСТИТЬ Лимиты
  | ИЗ РегистрСведений.Лимиты
  | ГДЕ ...
  | СГРУППИРОВАТЬ ПО ...
  | ИНДЕКСИРОВАТЬ ПО ...;

  // Выполняем основной запрос с использованием временной таблицы
  ВЫБРАТЬ ...
  ИЗ Документ.РеализацияТоваровУслуг
  ЛЕВОЕ СОЕДИНЕНИЕ Лимиты
  ПО ...;"
```

Переписывание запроса по приведенной выше методике имеет целью упростить работу оптимизатору СУБД. В переписанном запросе все выборки, участвующие в соединениях, будут представлять собой физические таблицы, и СУБД сможет легко определить размер каждой выборки. Это позволит СУБД гарантированно выбрать самый быстрый из всех возможных планов. Причем СУБД будет делать правильный выбор независимо ни от каких условий. Переписанный подобным образом запрос будет работать одинаково хорошо на любых СУБД, что особенно важно при разработке тиражных решений. Кроме того, переписанный подобным образом запрос лучше читается, проще для понимания и отладки.

Если в запросе используется соединение с виртуальной таблицей языка запросов «1С:Предприятия» (например, РегистрНакопления.Товары.Остатки()) и запрос работает с неудовлетворительной производительностью, то рекомендуется вынести обращение к виртуальной таблице в отдельный запрос с сохранением результатов во временной таблице.

Исключением из этого правила является случай, когда при левом соединении выборка по ведущей таблице дает мало записей, а вложенный запрос много. Тогда использование соединения с вложенным запросом (виртуальной таблицей) более оптимально, чем использование временных таблиц.

Однако риск неоптимальности запроса с временными таблицами в описанном случае заведомо меньше вероятности ошибки СУБД при построении неправильного плана для сложного запроса. Поэтому следует применять соединение с вложенными запросами и виртуальными таблицами только в том случае, если есть уверенность в их оптимальности. При отсутствии такой уверенности следует применять соединение с временными таблицами.

Также исключением из этого правила являются запросы в динамических списках, поскольку в них нет возможности создания временных таблиц.

Следует избегать неявных подзапросов, которые получаются при использовании вложенных соединений:

```
ВЫБРАТЬ ...
ИЗ Справочник.Номенклатура
   ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ТоварыНаСкладах
      ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ТоварыОрганизаций
      ПО ...
   ПО ...
Проблема в том, что, по сути, этот запрос аналогичен следующему:
ВЫБРАТЬ ...
ИЗ Справочник.Номенклатура     ЛЕВОЕ СОЕДИНЕНИЕ (
      ВЫБРАТЬ ...
      ИЗ РегистрНакопления.ТоварыНаСкладах
         ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ТоварыОрганизаций
         ПО ...)
   ПО ...
```

Вместо вложенных соединений, как показано выше, следует использовать последовательные соединения:

```
ВЫБРАТЬ ...
ИЗ Справочник.Номенклатура
   ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ТоварыНаСкладах
   ПО ...
   ЛЕВОЕ СОЕДИНЕНИЕ РегистрНакопления.ТоварыОрганизаций
   ПО ...
```

При этом следует понимать, что вложенные и последовательные соединения – это разные запросы, которые могут дать разный результат.

Если вложенное соединение использовано исходя из предположения, что оно аналогично последовательному соединению, то следует просто переписать его на последовательное соединение.

Если вложенное соединение делается осмысленно, то от него следует отказаться, так как оно может существенно снизить производительность, как и соединение с подзапросом. Как и в случае с подзапросом, такое соединение можно заменить на соединение с временной таблицей.

Ограничения на использование вложенных запросов в условии соединения

Не следует использовать вложенные запросы в условии соединения. Это может привести к значительному замедлению запроса и (в отдельных случаях) к его полной неработоспособности на некоторых СУБД. Пример запроса с использованием вложенного запроса в условии соединения:

```
Запрос.Текст = "ВЫБРАТЬ
   | ОстаткиТоваров.Номенклатура КАК Номенклатура,
   | Цены.Цена КАК ЦенаПрошлогоМесяца
   |ИЗ
   | РегистрНакопления.ТоварыНаСкладах.Остатки(...) КАК ОстаткиТоваров
   | ЛЕВОЕ СОЕДИНЕНИЕ РегистрСведений.Цена КАК Цены
   | ПО Цены.Номенклатура = ОстаткиТоваров.Номенклатура И
```

```
| Цены.Период В (
| ВЫБРАТЬ МАКСИМУМ(ЦеныПрошлогоМесяца.Период)
| ИЗ РегистрСведений.Цена КАК ЦеныПрошлогоМесяца
| ГДЕ ЦеныПрошлогоМесяца.Период < НАЧАЛОПЕРИОДА(ОстаткиТоваров.Период, МЕСЯЦ)
| И ЦеныПрошлогоМесяца.Номенклатура = ОстаткиТоваров.Номенклатура
| )
| ГДЕ ОстаткиТоваров.Склад = &Склад";
```

В данном случае вложенный запрос в условии соединения используется для получения как бы «среза последних» на конец предыдущего периода. Причем для каждой номенклатуры период может быть разным.

Подобный запрос рекомендуется переписать с использованием временных таблиц. Например, это можно сделать следующим образом:

```
Запрос.Текст = "
 // Максимальные даты установки цен в прошлом периоде для данных номенклатур
  |ВЫБРАТЬ
  | ОстаткиТоваров.Номенклатура КАК Номенклатура,
  | МАКСИМУМ(Цены.Период) КАК Период   |ПОМЕСТИТЬ ДатыПоНоменклатурам
  |ИЗ
  | РегистрНакопления.ТоварыНаСкладах.Остатки(...) КАК ОстаткиТоваров
  | ЛЕВОЕ СОЕДИНЕНИЕ РегистрСведений.Цена КАК Цены
  | ПО Цены.Номенклатура = ОстаткиТоваров.Номенклатура И
  | Цены.Период < НАЧАЛОПЕРИОДА(ОстаткиТоваров.Период, МЕСЯЦ)
  | СГРУППИРОВАТЬ ПО ОстаткиТоваров.Номенклатура
  | ГДЕ ОстаткиТоваров.Склад = &Склад;

 // Выбрать данные по цене за найденный период
  |ВЫБРАТЬ
  | ДатыПоНоменклатурам.Номенклатура КАК Номенклатура,
  | Цены.Цена КАК ЦенаПрошлогоМесяца
  |ИЗ ДатыПоНоменклатурам
  | ЛЕВОЕ СОЕДИНЕНИЕ РегистрСведений.Цена КАК Цены
  | ПО Цены.Номенклатура = ОстаткиТоваров.Номенклатура И
  | Цены.Период = ДатыПоНоменклатурам.Период";
```

Обращения к виртуальным таблицам

При использовании виртуальных таблиц в запросах следует передавать в параметры таблиц все условия, относящиеся к данной виртуальной таблице. Не рекомендуется обращаться к виртуальным таблицам при помощи условий в секции ГДЕ и т. п.

Такой запрос будет возвращать правильный (с точки зрения функциональности) результат, но СУБД будет намного сложнее выбрать оптимальный план для его выполнения. В некоторых случаях это может привести к ошибкам оптимизатора СУБД и значительному замедлению работы запроса.

Например, следующий запрос использует секцию ГДЕ запроса для выборки из виртуальной таблицы:

```
Запрос.Текст = "ВЫБРАТЬ
| Номенклатура
|ИЗ
| РегистрНакопления.ТоварыНаСкладах.Остатки()
|ГДЕ
| Склад = &Склад";
```

Возможно, что в результате выполнения этого запроса сначала будут выбраны все записи виртуальной таблицы, а затем из них будет отобрана часть, соответствующая заданному условию.

Рекомендуется ограничивать количество выбираемых записей на самом раннем этапе обработки запроса. Для этого следует передать условия в параметры виртуальной таблицы.

```
Запрос.Текст = "ВЫБРАТЬ
| Номенклатура
|ИЗ
| РегистрНакопления.ТоварыНаСкладах.Остатки(, Склад = &Склад)";
```

При обращении к виртуальной таблице следует передавать в условия наиболее простые конструкции, например: «Измерение = Значение». Не рекомендуется использовать подзапросы и соединения (*) в параметрах виртуальной таблицы, так как это приводит к медленной работе запроса.

При необходимости использовать подзапросы рекомендуется соблюдать следующие условия:

- в подзапросе только одна таблица, нет соединений с другими таблицами;
- если в подзапросе таблица табличной части, то не должно быть обращения к реквизитам таблицы-шапки;
- если в подзапросе таблица, у которой могут быть табличные части, то не должно быть обращений к табличным частям;
- если в подзапросе временная таблица, то не должно быть условий (раздела ГДЕ);
- • если в подзапросе постоянная таблица, то условие (раздел ГДЕ) допустимо, только если условие выполняется для 80 % (или более) случаев; отсутствие условия означает выполнение для 100 % случаев;
- если в подзапросе постоянная таблица, то в ограничениях доступа к данным (RLS) не должно содержаться подзапросов и соединений (допускаются только простые условия вида ГДЕ Реквизит = Значение, ГДЕ Истина). Например, при использовании стандартных шаблонов RLS, входящих в состав подсистемы «Управление доступом» «Библиотеки стандартных подсистем», к запросу неявно добавляется конструкция Exists с несколькими подзапросами и соединениями. В таких случаях следует переписать исходный запрос с использованием временной таблицы или привилегированного режима.

Эффективные условия запросов

Условия запросов должны быть написаны оптимально с точки зрения производительности, чтобы исключить существенное увеличение длительности выполнения запросов при увеличении объема данных в таблицах.

Поля основного условия в секциях ГДЕ, ПО и виртуальных таблицах должны быть проиндексированы. Основное условие может быть уточнено дополнительным условием, но объединять их следует только по И.

Основное условие – это то, что позволяет ограничить объем выборки больше других условий, и его составляющие объединены по И.

Дополнительное условие – это то, что объединено с основным условием по И, и его составляющие могут быть любой сложности (НЕ, <>, +, -, /, *, функции и т. п.).

Основное условие должно содержать только такие операции, которые позволяют выполнять поиск по индексу:

- для первого и всех используемых полей индекса, кроме последнего, только = и И;
- для последнего или единственного используемого поля индекса допустимо использовать =, >, <, >=, <=, ПОДОБНО, МЕЖДУ, В, ИЛИ (приводимое к В);
- нельзя использовать арифметические операции, функции, отрицания и неравенства.

Для условий в ГДЕ или в виртуальной таблице следует индексировать поля в основной таблице, из которой выполняется выборка. Для условий в ПО ЛЕВОГО соединения следует индексировать поля в правой таблице. Для условий в ПО ВНУТРЕННЕГО соединения следует индексировать поля в таблице с большим количеством записей.

Основное условие желательно строить таким образом, чтобы оно использовало индексы, которые автоматически создает платформа.

Не следует использовать ИЛИ в секции ГДЕ запроса. Это может привести к тому, что СУБД не сможет использовать индексы таблиц и будет выполнять сканирование, что увеличит время работы запроса и вероятность возникновения блокировок. Вместо этого следует разбить один запрос на несколько и объединить результаты.

Не рекомендуется использовать логическое ИЛИ в условиях соединения, то есть в секции ПО запроса. Это также может привести к выбору неоптимального плана и медленной работе запроса.

Если в конфигурации описано несколько ролей с разным ограничением доступа на уровне записей (RLS), то не следует назначать одному пользователю более одной такой роли. Если один пользователь будет включен, например, в две роли с RLS (бухгалтер и кадровик), то при выполнении всех его запросов к их условиям будут добавляться условия обоих RLS с использованием логического ИЛИ.

Подробнее о стандартах разработки в части работы с запросами можно прочесть здесь: https://kb.1c.ru/articleView.jsp?id=44.

Рекомендации по работе с блокировками

Общие сведения об избыточных блокировках

Проектные ошибки при выборе и проектировании структуры того или иного объекта метаданных для реализации прикладной функциональности могут привести к большому количеству избыточных блокировок и, как следствие, к серьезному падению общей производительности системы.

Ожидание на блокировке данных происходит в том случае, если две различные сессии «1С:Предприятия» пытаются захватить один и тот же ресурс. При работе с разными ресурсами ожидания на блокировке не происходит. В данном контексте термин «ресурс» используется в качестве обозначения неделимой совокупности данных, которая блокируется (или не блокируется) только вся целиком.

Таким образом, вопрос сводится к тому, какие именно ресурсы захватываются при выполнении того или иного действия с данными. Или, иначе говоря, насколько «мелко нарезаны» данные «1С:Предприятия».

При анализе структуры метаданных следует обратить особое внимание на следующие объекты:

- константы;
- последовательность;
- регистры бухгалтерии;
- регистры накопления.

Режим разделения итогов для регистров бухгалтерии

Если в системе осуществляется оперативная запись движений по бухгалтерскому регистру в многопользовательском режиме, то рекомендуется включить для данного регистра режим разделения итогов. При включенном режиме разделения итогов пользователи смогут параллельно обновлять таблицу остатков даже в том случае, если у них совпадают период, счет и значения измерений.

В противном случае таблица остатков регистра бухгалтерии может стать узким местом при конкурентной работе большого количества пользователей.

Предположим, что два пользователя одновременно проводят документы, которые осуществляют движение по данному регистру. Пользователи будут блокировать друг друга в том случае, если движения:

- относятся к одному и тому же периоду;
- относятся к одному и тому же счету;
- имеют одинаковые значения измерений, то есть организацию и валюту.

В реальной жизни одновременное выполнение перечисленных условий является весьма вероятным, поскольку большинство пользователей будут работать в одном периоде, с одним счетом и с одинаковыми значениями измерений (организация и валюта). Это может привести к возникновению ожиданий на блокировках и снижению общей производительности системы.

Для решения этой проблемы следует включить режим разделения итогов. После этого конкурирующие пользователи смогут параллельно записывать движения по регистру даже в том случае, если совпадают период, номер счета и значения всех измерений. Однако если при этом осуществляется контроль остатков по данному регистру, то эффекта от включения режима разделения не будет.

Подробнее о стандартах разработки в части работы с блокировками можно прочесть здесь: https://kb.1c.ru/articleView.jsp?id=45.

Перехват исключений в коде

Недопустимо делать проверки наличия у объекта реквизитов, методов, макетов и т. п., используя для этого исключения, так как это может привести к сложно диагностируемым ошибкам, а также затрудняет отладку в режиме «Останавливаться по ошибке». Вместо перехвата исключений в этом случае рекомендуется:

- использовать механизмы работы с метаданными, чтобы явным образом проверять наличие или отсутствие реквизита (макета и т. п.);
- если различия связаны с особенностями встраивания библиотек – описывать особенности явным образом в переопределяемых;
- пересмотреть логику работы методов, использующих перехват исключений. Например, можно предусмотреть параметры, которые определяются в вызывающем коде и указывают, нужно или нет обращаться к какому-либо методу или свойству объекта.

Например, *неправильно*:

```
Попытка
        КонтекстЭДОСервер.ПолучитьМакет("КомпонентаОбмена");
        ПутьВК = КонтекстЭДОСервер.ПутьКОбъекту + ".Макет.КомпонентаОбмена";
Исключение
КонецПопытки;
```

Правильно:

```
МакетКомпонентыОбмена = КонтекстЭДОСервер.Метаданные().Макеты.Найти("КомпонентаОбмена");
Если МакетКомпонентыОбмена <> Неопределено Тогда
        ПутьКМакету = КомпонентаОбмена.ПолноеИмя()
КонецЕсли;
```

При использовании транзакций следует придерживаться следующей схемы обработки исключений в коде на сервере:

```
// 1. Начало транзакции
НачатьТранзакцию();
Попытка
// 2. Вся логика блокировки и обработки данных размещается в блоке Попытка-Исключение
        Запрос = Новый Запрос("...");
        Выборка = Запрос.Выполнить().Выбрать();
        Пока Выборка.Следующий() Цикл
        ...
        КонецЦикла;

// 3. В самом конце обработки данных выполняется попытка зафиксировать транзакцию
        ЗафиксироватьТранзакцию();
        Исключение
// 4. В случае любых проблем с СУБД, транзакция сначала отменяется...
        ОтменитьТранзакцию();
// 5. ...затем проблема фиксируется в журнале регистрации...
        ЗаписьЖурналаРегистрации(НСтр("ru = 'Выполнение операции'"), УровеньЖурналаРегистрации.Ошибка,,,
                                ПодробноеПредставлениеОшибки(ИнформацияОбОшибке()));
// 6. ... после чего, проблема передается дальше вызывающему коду.
        ВызватьИсключение;
КонецПопытки;
```

Поскольку исключение не отменяет транзакцию сразу, но запрещает успешное завершение транзакции, то все вызовы НачатьТранзакцию, с одной стороны, и ЗафиксироватьТранзакцию или ОтменитьТранзакцию, с другой стороны, должны быть парными.

Отсутствие обработки исключительных ситуаций приводит к зависшим транзакциям и к сложно диагностируемым ошибкам вида «В этой транзакции уже происходили ошибки» в произвольных местах кода. При этом в случае вложенных операторов НачатьТранзакцию следует обеспечить вызов всех парных операторов ОтменитьТранзакцию. Для этого в конце блока Исключение необходимо пробросить исключение выше по стеку с помощью ВызватьИсключение (как в примере выше) и соответствующим образом обработать исключение на каждом уровне стека.

Неправильно использовать исключения для приведения значения к типу. Для таких операций необходимо использовать возможности объекта ОписаниеТипов.

Например, *неправильно*:

```
Попытка
 КоличествоДнейРазрешения = Число(Значение);
Исключение
 КоличествоДнейРазрешения = 0; // значение по умолчанию
КонецПопытки;
```

Правильно:

```
ОписаниеТипа = Новый ОписаниеТипов("Число");
КоличествоДнейРазрешения = ОписаниеТипа.ПривестиЗначение(Значение);
```

Клиент-серверное взаимодействие

Минимизация количества серверных вызовов

Разработку управляемого приложения необходимо вести с контролем количества вызовов серверных процедур и функций из клиентского кода (серверных вызовов), а в некоторых случаях – также объема передаваемых данных между клиентом и сервером (трафика).

Общее количество серверных вызовов складывается:

- из обращений на сервер, которые выполняет платформа «1С:Предприятие»,
- вызовов, которые выполняются из клиентского кода конфигурации – отклонение, привносимое конфигурацией по отношению к платформе.

В общем виде при проектировании клиент-серверного взаимодействия в конфигурации следует руководствоваться тем, что на каждое действие пользователя в клиентском коде конфигурации не должно выполняться дополнительных вызовов на сервер. Любые исключения из этого правила требуют дополнительного обоснования.

Отдельно для режима с низкой скоростью соединения следует оптимизировать не только количество вызовов, но и объем передаваемых данных между клиентом и сервером (трафик). Отладку клиент-серверного взаимодействия в этом режиме работы рекомендуется проводить в режиме имитации задержек серверных вызовов.

Ниже рассмотрены типовые действия пользователя и даны рекомендации по организации клиент-серверного взаимодействия.

Минимизация кода, выполняемого на клиенте

Необходимо минимизировать объем кода, который выполняется на стороне клиентского приложения. В частности, не следует выполнять на клиенте сложные алгоритмы, требующие значительных ресурсов компьютера. В таких случаях выполнение алгоритма на клиенте может занимать гораздо больше времени, чем передача управления с клиента на сервер, выполнение алгоритма на сервере и возврат результата обратно на клиент.

Следует размещать такие алгоритмы в серверном коде, выполняя к ним минимально необходимое число обращений с клиента.

Это требование продиктовано:

- тем, что, как правило, клиентский компьютер менее производительный, чем серверный компьютер;
- необходимостью приемлемого качества работы в веб-клиенте. Клиентский код выполняется интерпретатором встроенного языка, который в веб-клиенте работает заметно медленнее, чем в тонком или толстом клиенте.

Рекомендуется оставлять на клиенте такие алгоритмы, скорость работы которых заведомо выше, чем затраты, необходимые на вызов одной серверной функции. Например, перерасчет доступности элементов управления в форме при изменении пользователем данных выполняется на клиенте.

Исключение из этого правила составляют отдельные случаи, когда функциональная подсистема предназначена для работы с программным обеспечением, установленным на клиентском компьютере. Например, работа с торговым оборудованием, интеграция с клиент-банком, формирование печатных форм в офисные программы и т. п.

В тех случаях, когда функциональная подсистема предназначена для работы с клиентским программным обеспечением только в определенных режимах работы клиента, следует использовать директивы препроцессора. Например, для кода, недоступного в веб-клиенте:

```
#Если ВебКлиент Тогда
Предупреждение(НСтр("ru = 'Загрузка адресного классификатора не доступна в веб-клиенте.'"));
#Иначе
ОткрытьФорму("РегистрСведений.АдресныйКлассификатор.Форма.ФормаЗагрузкиАдресногоКлассификатора");
#КонецЕсли
```

ГЛАВА 12

Регламенты и практики эксплуатации крупных информационных систем на платформе «1С:Предприятие 8»

В предшествующих главах мы с вами освоили подходы к организации эксплуатации, рассмотрели с разных сторон как эксплуатацию в целом, так и конкретные организационные, методологические и технические приемы. Применяя на практике эти подходы, вы сможете качественно выстроить процесс эксплуатации своей информационной системы независимо от ее текущего состояния и размера.

В этой главе мы будем рассматривать регламенты и практические подходы к эксплуатации крупных информационных систем на реальных проектах, в которых участвуют специалисты фирмы «1С». Они качественно дополнят изложенный выше материал и дадут примеры повышения качества эксплуатации на реальных примерах.

Приемка и тестирование прикладных решений

Все знают, что тестировать – это хорошо и правильно, а не тестировать – «весело». Но эта книга про то, как делать правильно, поэтому рассмотрим процесс тестирования командой эксплуатации прикладных решений (типовых и доработанных), поступающих от поставщика. Под поставщиком здесь может подразумеваться как внешний подрядчик, который занимается разработкой/доработкой прикладного решения, так и отдел разработки ПО вашей компании, если речь идет о процессе внутренней разработки.

Приемка конфигураций

Чтобы понять, что очередной релиз конфигурации неработоспособен, не обязательно сразу же разворачивать полноценный процесс тестирования с развертыванием актуальных бэкапов баз на подготовительном контуре, обновлением на новый релиз, замерами производительности. Этап, сознательно выделенный из собственно тестирования, – приемка прикладного решения – может выявить потенциально неработоспособный функционал с минимальными затратами ресурсов.

На практике приемка прикладных решений чаще всего сводится к выполнению следующих шагов:

1. Проверить соответствия версии конфигурации:
   - □ версия конфигурации должна соответствовать заявленной версии нетипового решения;
   - □ версия конфигурации (в свойствах конфигурации, открытой в конфигураторе) должна соответствовать исходной версии типового решения (для случаев типового и доработанного типового решения);
   - □ версия конфигурации должна быть больше предыдущей.

   В противном случае могут некорректно отработать обработчики обновлений и иные механизмы БСП. При использовании полностью нетипового решения руководствуйтесь логикой версионирования данного решения и ориентированной на нее прикладной логикой.
2. Убедиться, что предоставленная конфигурация не снята с поддержки. Если на вашем проекте обновление на новые релизы ведется с сохранением поддержки, данное условие также нужно проконтролировать при приемке. При нарушении данного критерия необходимо получить новую поставку релиза.
3. Выполнить синтаксический контроль модулей конфигурации и проверку логической целостности. Проверку можно осуществить интерактивно в конфигураторе (Конфигурация – Проверка конфигурации).

   На рисунке 108 представлен минимальный вариант настройки. В зависимости от используемой в решении функциональности нужно задействовать соответствующие опции «Синтаксического контроля» (при работе решения через веб-клиент ставить флаг «Веб-клиент» и т. д.).

   Альтернативным способом проверки является запуск конфигуратора в пакетном режиме из командной строки. Ниже приведен пример такой команды.

```
chcp 1251 >nul
"C:\Program Files (x86)\1cv8\8.3.10.2561\bin\1cv8.exe" DESIGNER /S "server_name\db_name" /N Администратор /P ""
/CheckConfig -ConfigLogIntegrity -ThinClient -Server /DisableStartupDialogs /DisableStartupMessages /Out c:\output.txt
```

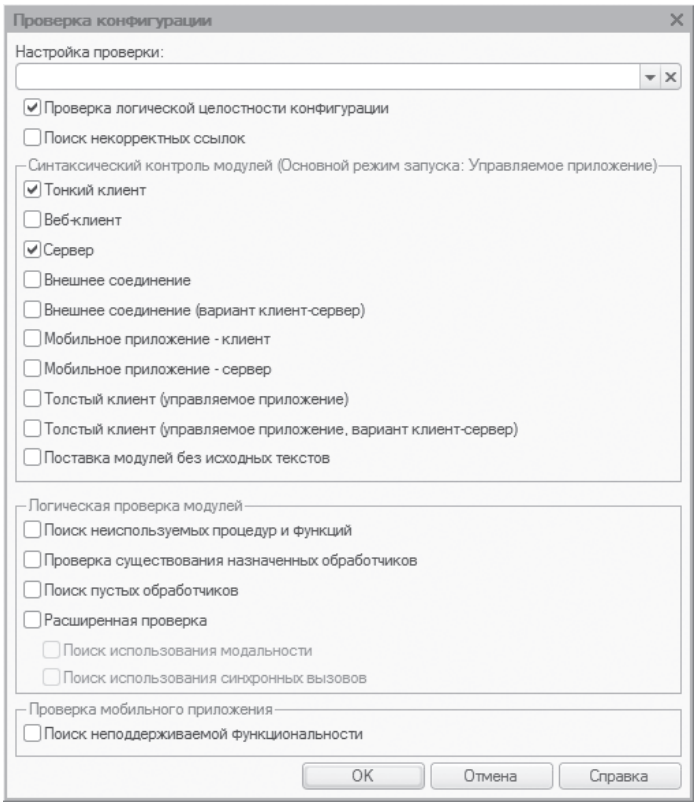


Рисунок 108. Окно проверки конфигурации

4. Провести сравнение, объединение предоставленной конфигурации со старой (текущей установленной) версией и проверить на наличие удаленных объектов метаданных. Если в новой конфигурации удалены объекты метаданных, реквизиты или табличные части, это повод перепроверить, не ошибочное ли это удаление. В случае ошибочного удаления есть вероятность при обновлении потерять данные, получить неработоспособную информационную базу.

   Сравнение можно осуществить как интерактивно через меню Конфигурация – Сравнить, объединить с конфигурацией из файла... в конфигураторе, так и в пакетном режиме из командной строки.

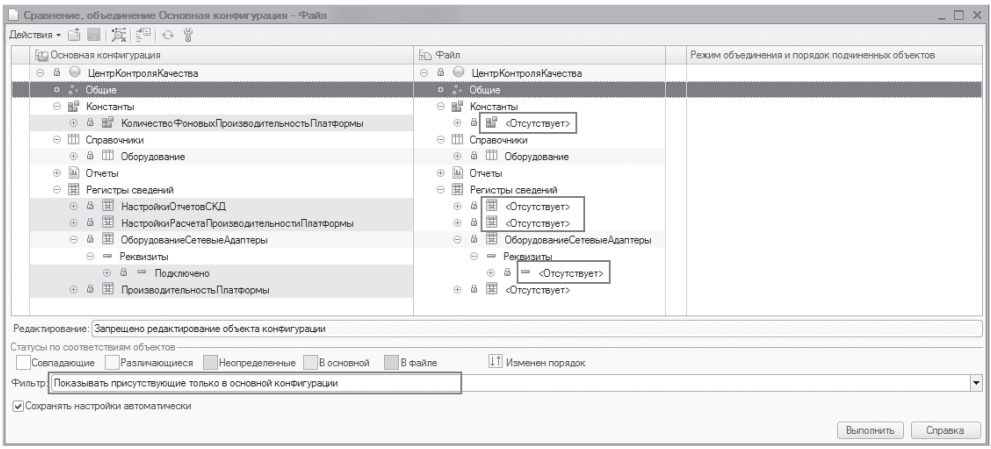


Рисунок 109. Окно «Сравнение, объединение» с фильтром по удаленным объектам

Пример команды:

```
chcp 1251 >nul
"C:\Program Files (x86)\1cv8\8.3.10.2561\bin\1cv8.exe" DESIGNER /S " server_name\db_name " /N Администратор
/P "" /CompareCfg -FirstConfigurationType MainConfiguration -SecondConfigurationType File -SecondConfigurationKey
c:\1cv8.cf -MappingRule ByObjectIDs -IncludeChangedObjects -IncludeDeletedObjects -IncludeAddedObjects
-ReportType Full -ReportFormat txt -ReportFile "C:\Report.txt" /DisableStartupDialogs /DisableStartupMessages /Out c:\output.txt
```

В данном скрипте учитываются все виды измененных объектов. Для нашей задачи достаточно –IncludeDeletedObjects.

В зависимости от принятой на конкретном внедрении практики эксплуатации результаты приемки могут как включаться в общий check-лист тестирования конфигурации, так и оформляться отдельным протоколом. Но важно, чтобы в том или ином виде результат был зафиксирован и сохранен. Пример протокола приемки конфигурации приведен в приложении 7.

Тестирование прикладных решений

После успешного прохождения этапа приемки наступает этап тестирования. Здесь мы, конечно, не будем описывать процесс тестирования программного продукта на этапе разработки. Предполагается, что этап разработки выстроен, а тестирование при разработке выполняется в необходимом объеме. Нам интересно, что именно должна тестировать команда эксплуатации после получения очередной поставки конфигурации.

Для чего вообще тестировать конфигурацию команде эксплуатации, если тестирование уже прошло на этапе разработки? Как минимум по двум причинам:

1. Команда эксплуатации *тестирует на реальных актуальных данных* своей информационной системы, к которым нет доступа у команды разработки. Конфигурация, идеально прошедшая все тесты на абстрактных данных команды

разработки, может показать абсолютно негативный результат на реальных данных в вашей базе на вашем оборудовании.

2. Разрабатывают люди, а им свойственно ошибаться. Если в процессе тестирования при разработке что-то не учли или недосмотрели, а пострадает ваша система, то это в первую очередь вина команды эксплуатации (пустили в продуктив непроверенный продукт) и только во вторую – вина разработчиков.

Хорошей практикой является получение от поставщика доработанной конфигурации вместе со следующим комплектом документов:

- описание функциональных изменений;
- «новое в релизе» – публикуемое описание функциональных изменений (то, что будет показано пользователям);
- перечень затрагиваемых этими изменениями объектов конфигурации (описание технических изменений);
- сценарий тестирования этих доработок;
- протокол внутреннего тестирования релиза поставщиком решения (в случае нетиповой или доработанной типовой конфигурации).

Проверка обновления

На практике при тестировании командой эксплуатации нового релиза конфигурации происходит проверка и контроль следующих параметров:

- время реструктуризации;
- время выполнения монопольных обработчиков обновления;
- время выполнения отложенных обработчиков обновления;
- время выполнения набора контрольных ключевых операций – согласованный набор действий в базе для проверки корректности результата и времени выполнения (проведение самых активно используемых документов, формирование отчетов);
- нагрузка на оперативную память, дисковую подсистему, процессор серверов кластера серверов «1С:Предприятия 8» при выполнении каждого из перечисленных выше этапов.

Чаще всего вывод о применимости обновления делается не на основании абсолютных значений приведенных показателей (например, реструктуризация за 6 часов – это много или в пределах нормы?). Обновление применимо, если производительность ключевых операций не снизилась необоснованно (обоснованно – это когда, например, документ проводился по одному регистру, а стал проводиться по двадцати, да еще с контролем остатков), время реструктуризации и выполнения обработчиков укладываются в технологическое окно.

Сценарное тестирование

Объем такого тестирования зависит от конфигурации, которую вы используете:

- типовая;
- доработанная типовая;
- нетиповая (доработанная типовая, снятая с поддержки = нетиповая).

В случае типовой конфигурации сценарное тестирование может не проводиться – либо проводиться для выбранного набора бизнес-критичных сценариев. В доработанной типовой конфигурации тестируется все, что тестируется в типовой, плюс нетиповые доработки данного релиза.

Нетиповая конфигурация тестируется по тем же правилам, что и доработанная типовая, но сам объем тестирования может быть намного большим за счет значительного количества доработок, а вариативность проверяемых сценариев шире.

Кроме набора основных бизнес-сценариев в минимальном варианте проверяется корректность интеграции решения с внешними системами.

В случае работы в модели сервиса также проверяются:

- взаимодействие с менеджером сервиса;
- выгрузка области в локальную копию;
- загрузка данных из выгрузки локальной копии в область.

По результатам тестирования формируется протокол для фиксации итогов. Пример протокола можно увидеть в приложении 7.

Порядок обновления прикладной конфигурации

Теперь опишем алгоритм обновления прикладной конфигурации. Данный алгоритм актуален как при выполнении обновления в подготовительной зоне для тестирования прикладного решения, так и для обновления в продуктиве.

Стандартной практикой по оформлению протокола обновления является оформление в виде check-листа. Образец check-листа на примере обновления версии платформы представлен в приложении 1. Для обновления версии конфигурации логика составления аналогичная, оформление – в удобном для команды виде.

Итак, полный алгоритм обновления прикладной конфигурации:

1. Выполнить проверки валидности входных данных:
   a. Соответствие новой и текущей версий конфигурации описанным в параметрах.
   b. Наличие файла конфигурации/обновления (cf/cfu) в нужном каталоге.
3. Выполнить подготовку ИБ к выполнению обновления:
   a. Создать полный (full) бэкап базы.
   b. Запрет выполнения фоновых заданий и новых подключений (при наличии автоматизации для массового обновления – используя rac).

c. Завершить фоновые задания.

d. Завершить пользовательские сеансы.

e. Создать разностный (diff) бэкап базы и/или бэкап транзакционных логов.

6. Выполнить загрузку новой конфигурации (при наличии автоматизации для массового обновления – с использованием пакетного запуска конфигуратора с ключом /LoadCfg).
7. Выполнить обновление конфигурации БД (при наличии автоматизации для массового обновления – с использованием пакетного запуска конфигуратора с ключом /UpdateDBCfg).
8. Выполнить первый вход в базу и контроль работоспособности базовых функций. Войти в тестовую область (или области) с выполнением обработчиков обновления (выполнятся автоматически при входе).
9. Выполнить контроль возможности выполнения тестового сценария клиентом в тестовой области.
10. Проанализировать наличие ошибок в журнале регистрации и технологическом журнале, которые могли быть продемонстрированы пользователю.
11. *Принять решение об успешности обновления или откате. После этой точки не откатываемся*. В случае ошибки сделать разностный бэкап базы и/или бэкап транзакционных логов для администраторов для расследования.
12. Разблокировать выполнение фоновых заданий.
13. Снять блокировку входа пользователей.
14. Дождаться окончания обновления областей (обновляются штатными фоновыми, нужно обработкой контролировать, что они завершились во всех областях).
15. Аналогично дождаться завершения отложенных обработчиков.
16. Выполнить контроль по технологическим журналам и журналу регистрации ошибок от обработчиков обновления (по методу в CALL и по RetExcp).
17. Сохранить/передать протокол обновления.
18. Сохранить/передать журнал регистрации от обновления для анализа.

В случае неуспеха на одном из этапов 3–8 процесс следует остановить и выполнить восстановление базы из ранее сохраненного бэкапа.

Теперь расставим акценты и дадим пояснения по некоторым действиям, включенным в данный алгоритм.

Для чего делать бэкап перед обновлением, понятно – чтобы в случае неуспеха выполнить откат. Но зачем делать полный бэкап, а затем разностный? Что мешает выгнать пользователей и сделать один полный бэкап? Таким образом мы сокращаем время плановой недоступности. Пока делается полный бэкап, а процесс этот может быть очень долгим при большом объеме базы, пользователи продолжают работать в базе. Недоступность начинается только тогда, когда полный бэкап подготовлен

и нам нужно только сделать дополнительно разностный бэкап или бэкап журнала транзакций, в который попадут транзакции, не отраженные в полном бэкапе. А эта операция по длительности очень короткая.

Для чего смотреть на технологический журнал и журнал регистрации после обновления? С учетом того, что обновление проверено на актуальной копии базы в подготовительном контуре, казалось бы, все уже проверено. На самом деле, конечно, это не так. Какой бы актуальной ни была копия базы, на которой проводилась проверка обновления, и насколько бы подготовительный контур ни соответствовал рабочему, проверка того, что при обновлении в рабочей системе не проявились ошибки, не обнаруженные в подготовительной зоне, должна выполняться в обязательном порядке.

Реакция на инциденты

Мы ранее говорили, что в грамотно организованном процессе эксплуатации крупной системы всегда должен быть дежурный администратор – человек, который отвечает за обеспечение качества работы системы в данный момент времени, обеспечивая своевременную реакцию на инциденты. Вся работа дежурного обязательно должна быть регламентирована – описаны действия дежурного по реакции на инциденты различного характера, зафиксированы шаги по устранению или обходу известных технологических проблем.

Рассмотрим пример алгоритма действий дежурного при возникновении недоступности информационной базы.

Если система мониторинга фиксирует недоступность системы либо инцидент пришел от пользователя или по иным каналам, порядок действий дежурного администратора должен быть следующим:

- Убедиться в наличии недоступности:
  - недоступность оборудования: ping, проверка возможности удаленного входа;
  - недоступность базы/публикации: проверка входа через внешнюю и внутреннюю веб-публикации (при наличии), входа по прямому подключению с указанием сервера и имени БД.
- Оповестить о недоступности в чате эксплуатации, зафиксировав:
  - недоступную единицу оборудования (ноду, базу, публикацию);
  - симптомы.
- В случае выполнения каких-либо действий по восстановлению эта информация также отражается в чате.
- После восстановления доступности факт восстановления отражается в чате.

Пример:

```
[15:55:35] Иванов И: недоступна нода 5, база sbm_5
[15:55:37] Иванов И: висит на входе через внешнюю, внутреннюю публикации и по прямому подключению
[15:55:50] Иванов И: по ProcExp вижу зависание поторков в rphost
[15:55:55] Иванов И: снимаю дампы процессов кластера
[15:56:15] Иванов И: перезапускаю кластер
[15:56:34] Иванов И: доступность восстановлена. Успешный вход через внешнюю, внутреннюю публикации и напрямую
```

Резервное копирование и хранение данных

Одной из составляющих процесса эксплуатации информационной системы является грамотная организация создания и хранения данных:

- резервные копии баз данных;
- версии прикладного ПО (платформа, прикладные конфигурации и т. д.);
- данные мониторинга (журналы, счетчики производительности и т. д.);
- файлы настроек;
- скрипты автоматизации.

Резервное копирование баз данных

Способы создания резервных копий (бэкапов) баз данных и их виды подробно рассмотрены в главах 7 и 8 данной книги. В данном разделе опишем, из чего следует исходить при выборе политики резервного копирования.

С точки зрения задач эксплуатации на выбор политики создания и восстановления бэкапов влияют **два основных фактора**:

- допустимое время простоя системы в случае аварии;
- допустимая глубина потери данных в случае аварии.

Здесь под аварией понимаем такую нештатную ситуацию, когда база данных в продуктиве пришла в полностью неработоспособное состояние и нужно восстанавливаться из резервной копии. Также предполагаем, что здесь рассматривается именно сценарий, когда не настроено зеркалирование или AlwaysOn. Есть только база и ее резервные копии.

Есть также ряд второстепенных факторов, которые могут накладывать дополнительные условия и ограничения на процесс резервного копирования и восстановления:

- размер базы данных;
- количество свободного дискового пространства в сетевом хранилище;
- наличие резервных мощностей для развертывания актуальной копии базы данных.

Есть ряд простых правил, которые в рамках описанных выше факторов являются базовыми:

- Хранить бэкапы на тех же дисках, что и базу данных – плохое решение. В случае «потери» дисковой подсистемы теряете все. Лучше использовать отдельные мощности под файловое хранилище с резервированием.
- В большинстве случаев нет необходимости создавать и хранить только полные бэкапы: это избыточно как с точки зрения занимаемого в файловом хранилище места, так и с точки зрения времени восстановления из резервной копии.
- Если время восстановления базы из резервной копии «с нуля» превышает допустимое время простоя системы, вам необходимы дополнительные мощности, на которых вы автоматически будете разворачивать актуальную копию базы из бэкапа. В случае аварии восстановление будет сводиться к приведению этой копии к актуальному состоянию путем последовательного наката резервных копий журнала транзакций.
- Частота, с которой вы будете делать резервные копии журнала транзакций, равна требованиям по допустимой глубине потери данных в случае аварии.

  Приведем пример. Требование клиента: в случае аварии допустимая потеря данных – 15 минут. Частота резервного копирования журнала транзакций – 15 минут.
- Делать бэкапы недостаточно для уверенности в том, что ваша система застрахована от аварий. Необходимо с определенной периодичностью проводить проверочное восстановление из бэкапов с проверкой их работоспособности. Например, после создания очередной дневной резервной копии автоматически скриптом производить восстановление из этого бэкапа в подготовительной зоне с проверкой корректности восстановления и работоспособности информационной базы (произвести вход).
- Если вы придерживаетесь правил регулярной проверки восстановления, то чаще всего нет необходимости хранить все бэкапы за все периоды, независимо от политики резервного копирования. Стандартной является следующая практика хранения бэкапов:
  - один полный бэкап на конец года (чаще всего на конец финансового года, когда сдана готовая отчетность);
  - по одному полному бэкапу за каждый месяц текущего года;
  - по одному бэкапу за каждую неделю месяца;
  - все бэкапы (полные, разностные, журналов транзакций) за последние семь дней.

Сценарии и политики создания и хранения резервных копий могут варьироваться в зависимости от специфики конкретной информационной системы.

Хранение версий прикладного ПО

Хорошей практикой в процессе эксплуатации информационных систем является хранение всех версий ПО и прикладных конфигураций, которые когда-либо (даже «эпизодически») появлялись в системе. Эти версии также должны размещаться на мощностях файлового хранилища.

Вместе с ними должны храниться и протоколы тестирования соответствующих версий. Это позволит при необходимости в любой момент обратиться к данной информации для сравнительного анализа конфигураций, сравнения параметров нагрузки, создаваемых текущей и более ранней версией прикладного решения, и т. п.

Хранение данных мониторинга

Для хранения данных мониторинга есть два базовых правила (не считая того, которое гласит: «Мониторинг должен быть!»):

- Данные мониторинга не должны постоянно храниться на тех машинах, с которых собираются (место кончится).
- Период хранения данных мониторинга определяется их необходимостью для расследования проблем (какая польза, например, от технологических журналов пятилетней давности?).

Чтобы данные мониторинга не засоряли ваши продуктивные сервера, должна быть обеспечена их автоматическая миграция на сервер сетевого хранилища. Пример того, как такое перемещение может быть реализовано для технологических журналов платформы «1С:Предприятие 8»:

```
#получим начальные параметры
$now_date = Get-Date -Format "yyyyMMddhh"
$now_date_for_folder  = Get-Date $now_date -Format "yyyyMMdd"
$server = ($env:COMPUTERNAME)

$log_path = "C:\LOGS"
$repo_path = "\\MyFileServer\logs\" + $server + "\" + $now_date_for_folder
if (!(Test-Path ($repo_path)))
{
        New-Item -Path $repo_path -ItemType directory
}

#получим список файлов ТЖ
$files = Get-ChildItem $log_path -Include *.log -Recurse

foreach ($file in $files)
{
        if ($file -ne $null)
        {
                #определим дату файла по имени
                $file_date = [datetime]::ParseExact(($file.Name).Replace(".log",""), "yyMMddHH", $null)
                #переместим все не из текущего часа
                if ($file_date -lt $now_date)
                {
                        $file_repo_path = $file.DirectoryName.Replace($log_path, $repo_path)
                        if (!(Test-Path ($;)))
```

```
                {
                    New-Item -Path $file_repo_path -ItemType directory
                }
                Move-Item -Path $file.FullName -Destination $file_repo_path -Force
            }
        }
}
```

Запустив подобные скрипты на автоматическое периодическое выполнение, вы обеспечите перенос и структурированное хранение данных мониторинга. Далее в сетевом хранилище данные мониторинга должны быть помещены в архив для экономии занимаемого места.

Период хранения данных мониторинга может варьироваться в зависимости от регламентов и задач эксплуатации на конкретном проекте. Наиболее частая практика – хранить данные мониторинга ровно за тот период, за который проведен анализ этих данных и есть зарегистрированные проблемы.

Хранение настроек и скриптов

В крупной распределенной системе все процессы эксплуатации, все скрипты автоматизации и все настройки должны быть стандартизованы. Мы об этом писали ранее, и это очевидно. Если в системе, состоящей из двух, пяти, десяти серверов, еще можно попытаться обосновать специфические настройки и их «абсолютную значимость» ровно для этой машины, то когда счет машин в контуре переваливает за несколько десятков, такая система становится просто неконтролируемой, неуправляемой и применить понятие «эксплуатация» к такому хаосу можно очень условно.

Очевидно, что стандартизованные настройки и скрипты должны, в том числе, храниться и распространяться централизованно. Но для обеспечения качества процессов эксплуатации за счет повышения контролируемости процесса изменения настроек и, как следствие, быстрого обнаружения некорректной настройки мы рекомендуем настройки и скрипты хранить в разрезе версий.

Для версионирования скриптов и настроек можно использовать любую привычную систему управления версиями, которые получили большое распространение среди программистов как средство управления версиями программных модулей разрабатываемого ими ПО. Наиболее распространенным системами управления версиями на текущий момент являются **Git** и **SVN (Subversion, официальное название – Apache Subversion)**.

Основные полезные эффекты использования систем управления версиями для задач эксплуатации:

- возможность использовать команды автоматизации для централизованного развертывания из репозитория и применение единой версии файлов ко всем серверам;
- возможность сравнить текущую версию настроек с любой предыдущей (фактически – любую версию с любой);

- возможность оперативно увидеть, кто, когда и какие изменения внес;
- возможность оперативно вернуться к более ранней версии в случае выявления проблемы в новой версии.

Правила именования баз, серверов и нод

Когда количество узлов системы (баз, серверов, нод и т. п.) становится велико, то качество эксплуатации системы начинает сильно зависеть от того, насколько структура и наименования компонент позволяют администратору быстро среагировать на проблему.

Например, серверы в вашем контуре именуются по порядку: *server1, server2, server3,..., serverN*. Давайте соберем из них ноды из двух веб-серверов, одного сервера «1С:Предприятия 8» и одного сервера СУБД:

Node1:

Server1
Server2
Server3
Server4

Node2:

Server5
Server6
Server7
Server8

...

Уже видно, какой недостаток в этой схеме: глядя на ноду и ее серверы, невозможно определить предназначение того или иного сервера, а глядя просто на список серверов, невозможно определить, к какой ноде он относится. Где-то под рукой нужно иметь список соответствий «сервер – описание», чтобы понять, что это за машина. Не очень удобно, когда нужно восстанавливать доступность и каждая секунда на счету.

Давайте представим, как нужно именовать машины в крупной системе, чтобы уже из названия машины была понятна ее роль. Итак, чего нам не хватает в имени сервера?

Номер ноды: получая сигнал о недоступности, нам в первую очередь нужно понимать, какая нода и, соответственно, какие базы недоступны.

Роль сервера в ноде (веб-сервер, сервер «1С:Предприятия», сервер СУБД и т. д.): эта информация позволит быстрее локализовать проблему в случае недоступности. Например, быстро вывести нерабочий сервер из кластера серверов «1С:Предприятия».

Номер сервера данной роли в ноде.

С учетом этого наш пример будет выглядеть так:

Node1:

Web-1-1

Web-1-2

App-1-1

DB-1-1

Node2:

Web-2-1

Web-2-2

App-2-1

DB-2-1

...

Такая практика именования серверов используется нами на крупных внедрениях. В зависимости от специфики конкретного проекта могут вводиться дополнительные правила именования для быстрого доступа к информации о специфике ноды или сервера. Например, в проектах, где используются разные операционные системы, в имя ноды или сервера часто добавляют признак той или иной установленной ОС (nix, win).

Также в случае использования нескольких фиксированных конфигураций ноды этот признак отражается в названии ноды. Чаще всего на практике используются конфигурации нод, различаемых по «размеру» или «мощности» (это условная характеристика, интерпретируемая по-своему в зависимости от специфики проекта). Часто применяется такая градация:

- микронода (S, small);
- средняя нода (M, medium);
- большая нода (L, large).

Такая градация наиболее проста для восприятия, и именно ее мы будем использовать далее в примерах. В реальной практике очень часто происходит немного иначе:

- «стандартные» ноды или набор стандартных нод (например, S, M, L) – наиболее распространенные на проекте, так как ровно они являются стандартом и общим критерием по возможной нагрузке, создаваемой базами на ноду;
- вынужденные отклонения от «стандарта» (например, при существовании в облачном контуре крупных необлачных баз, наличии демонстрационного контура с незначительной нагрузкой в рамках рабочего и т. д.).

Отклонение от «стандарта» в большую сторону (увеличение мощности ноды) – вынужденное, из-за наличия выбивающейся из стандарта базы (баз). Отклонение от «стандарта» в меньшую сторону (уменьшение мощности ноды) обычно обусловлено наличием невысоконагруженных баз, необлачных баз со специфической конфигурацией и т. д., использование для работы которых «стандартной»

ноды было бы избыточным с точки зрения использования аппаратных мощностей. Чем меньше таких отклонений от «стандарта», тем лучше! Желательно – не более 1–2 вариаций.

С учетом сказанного выше наш пример мог бы выглядеть так:

Node-S-1:

Web-1-1

Web-1-2

App-1-1

DB-1-1

Node-L-2:

Web-2-1

Web-2-2

App-2-1

DB-2-1

...

Или, в случае разделения по разным ОС:

Node-ubuntu-S-1:

Web-1-1

Web-1-2

App-1-1

DB-1-1

Node-win-L-2:

Web-2-1

Web-2-2

App-2-1

DB-2-1

...

Теперь мы при первом же взгляде понимаем конфигурацию ноды из перечня типовых (предполагается, что аппаратные характеристики типовых нод ответственный за эксплуатацию знает на уровне рефлексов) и роль каждого сервера в ней.

Именовать базы в кластере также предлагается с привязкой к ноде – из соображений более быстрой реакции на проблему и ее локализации. Вариант именования баз:

<ТипБазыВНоде> – *<НомерНоды>* – *<ПорядковыйНомерБазыВНоде>*

где

<ТипБазыВНоде> – тип прикладной конфигурации ea, sbm, erp и т. п.

Пример:

ea_1_1 – база БП № 1 на первой ноде;

ea_2_1 – база БП № 1 на второй ноде;

sbm_1_1 – база УНФ № 1 на первой ноде.

Скрипты автоматизации стандартных действий администратора

В данном разделе приведены примеры некоторых скриптов, позволяющих автоматизировать стандартные действия администратора. Очевидно, что использовать скрипты «as is» не рекомендуется. Любой скрипт, взятый из книги или любого другого источника, требует отладки, проверки корректности его работы именно в вашей системе в подготовительной зоне. Если это утверждение вам кажется странным и наталкивает на мысль вроде: «Если нужно отлаживать и проверять, значит, скрипт неуниверсален» – предлагаем освежить в памяти материал главы 2 данной книги – «Организация эксплуатации крупной информационной системы».

Немедленно завершить приложения

```
taskkill /F /IM ragent.exe
taskkill /F /IM rmngr.exe
taskkill /F /IM rphost.exe
taskkill /F /IM httpd.exe
taskkill /F /IM w3wp.exe
taskkill /F /IM 1cv8c.exe
taskkill /F /IM 1cv8.exe
```

Получить дампы процессов

Этот скрипт должен выполняться из той же директории, в которой находится утилита procdump.exe. Обратите внимание, что здесь используется ключ –ma. По этому ключу будут сняты полные дампы процессов, что может занять длительное время (зависит от количества оперативной памяти, занятой процессом), в течение которого система будет недоступна. В случае использования в продуктиве рекомендуется снимать мини-дампы (убрать из скрипта ключи –ma). В большинстве случаев таких дампов достаточно для расследования проблем.

```
@ echo off
echo Start dumping processes on this server...
CD %~dp0
for /f "usebackq tokens=2" %%a in (`tasklist /FO list /FI "IMAGENAME eq rmngr.exe" ^| find /i "PID:"`) do (
start /b procdump.exe -ma %%a rmngr_%%a
)
for /f "usebackq tokens=2" %%a in (`tasklist /FO list /FI "IMAGENAME eq rphost.exe" ^| find /i "PID:"`) do (
start /b procdump.exe -ma %%a rphost_%%a
)
for /f "usebackq tokens=2" %%a in (`tasklist /FO list /FI "IMAGENAME eq ragent.exe" ^| find /i "PID:"`) do (
start /b procdump.exe -ma %%a ragent_%%a
)
for /f "usebackq tokens=2" %%a in (`tasklist /FO list /FI "IMAGENAME eq ragent.exe" ^| find /i "PID:"`) do (
```

```
start /b procdump.exe -ma %%a ragent_%%a
)
for /f "usebackq tokens=2" %%a in (`tasklist /FO list /FI "IMAGENAME eq httpd.exe" ^| find /i "PID:"`) do (
start /b procdump.exe -ma %%a httpd_%%a
)
for /f "usebackq tokens=2" %%a in (`tasklist /FO list /FI "IMAGENAME eq w3wp.exe" ^| find /i "PID:"`) do (
start /b procdump.exe -ma %%a w3wp_%%a
)
for /f "usebackq tokens=2" %%a in (`tasklist /FO list /FI "IMAGENAME eq 1cv8c.exe" ^| find /i "PID:"`) do (
start /b procdump.exe -ma %%a 1cv8c_%%a
)
for /f "usebackq tokens=2" %%a in (`tasklist /FO list /FI "IMAGENAME eq 1cv8.exe" ^| find /i "PID:"`) do (
start /b procdump.exe -ma %%a 1cv8_%%a
)
```

Остановка службы «1С:Предприятие» с очисткой временных файлов

```
set LOG_FILE="scripts.log"
set SERVICE_1C_NAME="1C:Enterprise 8.3 Server Agent (x86-64)"
set SERVICE_RAS_NAME="1C:Enterprise 8.3 Remote Server"
set CNTX_PATH="C:\srvinfo\reg_1541"
set PFL_PATH="C:\ProgramData\1C\1cv8"
set TEMP_PATH="C:\Windows\Temp"
echo stop %DATE% %TIME% >> %TEMP_PATH%\%LOG_FILE%
sc stop %SERVICE_1C_NAME%
sc stop %SERVICE_RAS_NAME%
timeout 5
taskkill /f /im "rphost.exe"
taskkill /f /im "rmngr.exe"
taskkill /f /im "ragent.exe"
taskkill /f /im "ras.exe"
timeout 5
echo done stop %DATE% %TIME% >> %TEMP_PATH%\%LOG_FILE%
echo clean temp %DATE% %TIME% >> %TEMP_PATH%\%LOG_FILE%
DEL /Q /F /S %CNTX_PATH%\snccntx*
DEL /Q /F %PFL_PATH%\*.pfl
DEL /Q /F /S %TEMP_PATH%\*.*
echo done clean temp %DATE% %TIME% >> %TEMP_PATH%\%LOG_FILE%
```

Запуск службы «1С:Предприятие»

```
set LOG_FILE="scripts.log"
set SERVICE_1C_NAME="1C:Enterprise 8.3 Server Agent (x86-64)"
set SERVICE_RAS_NAME="1C:Enterprise 8.3 Remote Server"
echo start %DATE% %TIME% >> %TEMP_PATH%\%LOG_FILE%
sc start %SERVICE_1C_NAME%
sc start %SERVICE_ RAS _NAME%
echo done start %DATE% %TIME% >> %TEMP_PATH%\%LOG_FILE%
```

Рестарт с очисткой временных файлов

```
set LOG_FILE="scripts.log"
set SERVICE_1C_NAME="1C:Enterprise 8.3 Server Agent (x86-64)"
set SERVICE_RAS_NAME="1C:Enterprise 8.3 Remote Server"
set CNTX_PATH="C:\srvinfo\reg_1541"
set PFL_PATH="C:\ProgramData\1C\1cv8"
set TEMP_PATH="C:\Windows\Temp"
echo stop %DATE% %TIME% >> %TEMP_PATH%\%LOG_FILE%
sc stop %SERVICE_1C_NAME%
```

```
sc stop %SERVICE_RAS_NAME%
timeout 5
taskkill /f /im "rphost.exe"
taskkill /f /im "rmngr.exe"
taskkill /f /im "ragent.exe"
taskkill /f /im "ras.exe"
timeout 5
echo done stop %DATE% %TIME% >> %TEMP_PATH%\%LOG_FILE%
echo clean temp %DATE% %TIME% >> %TEMP_PATH%\%LOG_FILE%
DEL /Q /F /S %CNTX_PATH%\snccntx*
DEL /Q /F %PFL_PATH%\*.pfl
DEL /Q /F /S %TEMP_PATH%\*.*
echo done clean temp %DATE% %TIME% >> %TEMP_PATH%\%LOG_FILE%
echo start %DATE% %TIME% >> %TEMP_PATH%\%LOG_FILE%
sc start %SERVICE_1C_NAME%
sc start %SERVICE_ RAS _NAME%
echo Service %SERVICE_1C_NAME% restarted at %DATE% %TIME% >> %TEMP_PATH%\%LOG_FILE%
```

Аварийное завершение процесса rphost, который потребляет больше N Гб памяти

N сильно зависит от вашей системы, нагрузки, конфигурации и т. д. В этом примере N = 8 Гб.

```
@echo off
REM MemLimit in bytes!
REM MemLimit is 8 GB
set MemLimit=8796093022208
echo MemLimit is set %MemLimit% bytes
for /f "usebackq tokens=2" %%a in (`tasklist /FO list /FI "IMAGENAME eq rphost.exe" ^| find /i "PID:"`) do (
        for /f "usebackq tokens=1" %%c  in (`"wmic process where ProcessId=%%a get WorkingSetSize"`) do (
                SET "var="&for /f "delims=0123456789" %%i in ("%%c") do set var=%%i
                        if not defined var (
                                if /I %%c GTR %MemLimit% (
                                echo Killing process rphost_%%a with Mem Usage %%c for breaking limit %MemLimit%
                                taskkill /F /PID %%a
                        )
                )
        )
)
```

ПРИЛОЖЕНИЕ 1

Примеры check-листов обновления

Обновление версии технологической платформы «1С:Предприятия»

Обновление платформы 8.3.7.1182 на рабочем стенде (29.09.2016).

Участники:

- Администратор1 – веб-серверы (Linux).
- Администратор2 – серверы «1С:Предприятия» и СУБД (Windows).
- Администратор3 – прикладная часть (информационная база).
- Администратор3 – контроль и координация.

Информация:

- Версия платформы: 8.3.7.1182
- Дистрибутив тонкого клиента: _8_3_7_1182.zip
- Обновляемые серверы: {2,4,12,13}

Подготовительный стенд:

Linux

192.168.123.61 SERVERWEB2-1
192.168.123.62 SERVERWEB2-2
192.168.123.65 SERVERWEB4-1
192.168.123.66 SERVERWEB4-2
192.168.123.45 SERVERWEB12-1

192.168.123.46 SERVERWEB12-2
192.168.123.47 SERVERWEB13-1
192.168.123.48 SERVERWEB13-2
192.168.123.95 DSRS1

Windows

192.168.123.26 SERVER1C2-1
192.168.123.28 SERVER1C4-1
192.168.123.21 SERVER1C12-1
192.168.123.22 SERVER1C13-1

Порядок выполнения работ

	Операция	Ответственный	Готово
ПОДГОТОВКА			
	Установить новые клиентскую и серверную части платформы «1С:Предприятие» версии 8.3.7.1182 на серверах: server1c{2,4,12,13}[-1]	Администратор2	
	Полный бэкап баз mssql (за час до старта): Accounting Accounting_12 Accounting_15	Администратор2	
СТАРТ			
	Выключить сервисы «1С:Предприятия»: server1c{2,4,12,13}-1:1540 с помощью скрипта C:\1C\1c_stop.cmd	Администратор2	
	Дифференциальный бэкап баз mssql Accounting Accounting_12 Accounting_15	Администратор2	
	Установить новую серверную часть платформы «1С:Предприятие» версии 8.3.7.1182 и перезапустить Apache на серверах: serverweb{2,4,12,13}-{1,2}, DSRS1	Администратор1	
	Переключить серверную часть платформы «1С:Предприятие» на новую версию 8.3.7.1182 и запустить сервисы на серверах: server1c{2,4,12,13}[-1] с помощью скрипта C:\1C\1c_update_and_start.cmd	Администратор2	
	В менеджере сервиса установить версию платформы 8.3.7.1182 для соответствующих элементов справочника «Кластеры», кроме агента server1c2-1	Администратор3	
	Проверить работоспособность новой версии платформы «1С:Предприятие»: • Автообновление ТК. • Запуск по одному приложению, размещенному на server1c{2,4,12,13}[-1]: в тонком клиенте; в веб-клиенте. • Работу сценария сбора статистики агентом server1c1-1	Администратор3	
ФИНИШ			
	Оповестить участников о завершении	Администратор3	

ПРИЛОЖЕНИЕ 2

Пример еженедельного отчета по качеству работы информационной системы

Отчет по сервису «Сервис по технологии 1cFresh»

Автор: Иванов И.М.

Что можно отметить по итогам прошлой недели:

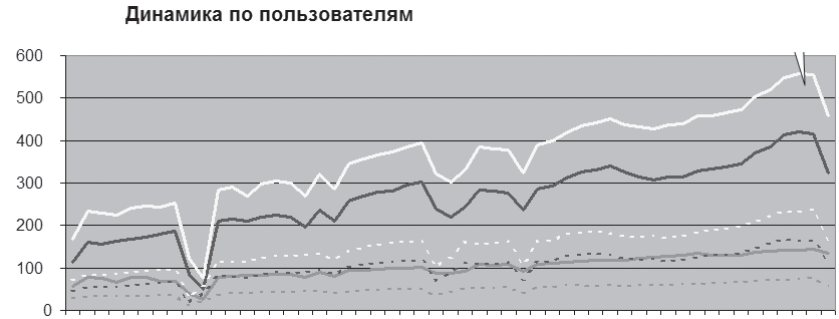
- Медленное выполнение операции корректировки налога в форме НДФЛ в документе Начисление зарплаты сотрудников организации.

  Для исправления проблемы было сделано следующее (собирается от двух до пяти таких наиболее значимых моментов):

  - подготовлена и выполнена обработка по изменению константы РассчитыватьДокументыПриРедактировании и отключению расчета при редактировании;
  - исправление лишних вызовов в конфигурации.

Динамика абонентов и пользователей

Динамика среднего количества активных пользователей, которые проработали в сервисе хотя бы минуту. На графике приведена информация по общему количеству пользователей, а также с детализацией по базам. Период динамики (одна точка на графике) – неделя.

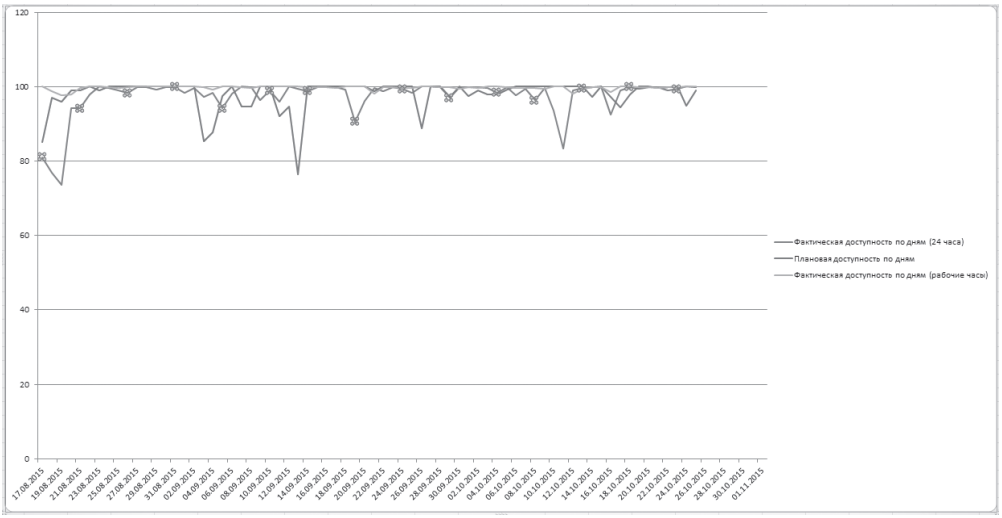


Доступность сервиса

Показатели	План	Факт
Доступность за неделю, %	**99,74**	**99,70**
Доступность с начала года, %	**99,69**	**99,57**

График доступности по всем информационным базам (на одном графике)

Рассчитывается по формуле: *ОбщаяДоступность = Средняя доступность всех за неделю*.



Список недоступностей; их причины; что сделано, чтобы не повторялось:

[13.11.2015 с 8:18 по 9:36] [13.11.2015 с 10:17 по 12:26] На сервере СУБД 4-й ноды на диске E закончилось место, все место было занято tempbd.

Решение или причина: вместо поиска виновного был многократно выполнен перезапуск СУБД, затем рабочих серверов кластера. Недоступность – только в результате неправильных действий по устранению проблемы.

Исполнители: Арнольд Недоступников, Мальвина Косячкина.

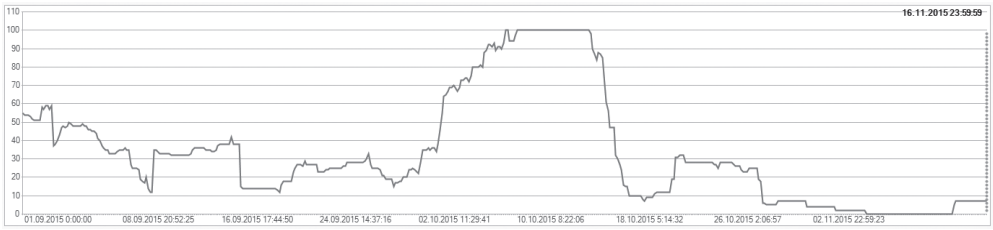
Устранил: Аскольд Правильнов.

[13.11.2015 с 13:49 по 13:55] Мальвина Косячкина: нода 13, платформа 1С 8.2. Не найдена лицензия.

Решение или причина: проблема воспроизвелась один раз, повторений не было, недоступности не было.

Расследование проблемы: Аскольд Правильнов.

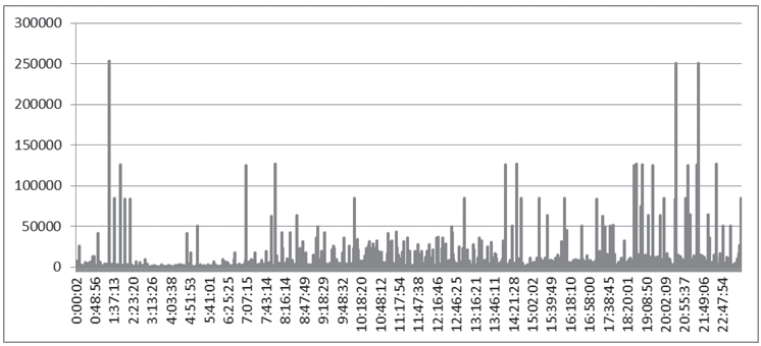
На графике показана динамика аварийных завершений процессов за последний месяц, показатель отображает динамику количества.



Топ–5 ошибок по данным технологических журналов

1. PRO-1234567 Ошибка «ТекстОшибки».
2. PRO-1234567 Ошибка «ТекстОшибки».
3. PRO-1234567 Ошибка «ТекстОшибки».
4. PRO-1234567 Ошибка «ТекстОшибки».
5. PRO-1234567 Ошибка «ТекстОшибки».

График длительности запросов по данным внешнего IIS за неделю



Топ–5 проблем производительности по всем конфигурациям, что делается для решения

1. PRO 1 Медленное выполнение…
2. PRO 2 Медленное выполнение…
3. PRO 3 Медленное выполнение…
4. PRO 4 Медленное выполнение…
5. PRO 5 Медленное выполнение…

Производительность информационных баз за неделю

Выгрузка из ЦКК по Apdex, среднее время, число операций в сравнении с предыдущей неделей. Если есть замедление, то указать причины.

Информационная база / Период замера / Ключевая операция	Целевое время	APDEX	Влияние на APDEX группы	Замеров	Среднее значение замера	Стандартное отклонение	Максимальное значение	Минимальное значение
Итого		0,613(-0,048)	0,387	96 925	18,346(-0,422)	64,569	13 381,142	0,001
emzkou_3 (Кластер app-5-1)		0,621(-0,046)	0,243	62 238	15,480(+1,403)	44,038	2 881,408	0,001
11.11.2015г.		0,625(-0,049)	0,062	10 311	15,487(+3,913)	37,768	1 615,597	0,001
12.11.2015г.		0,621(-0,195)	0,061	10 007	16,393(+2,475)	52,833	2 339,409	0,001
13.11.2015г.		0,602(-0,135)	0,064	10 050	15,630(+4,273)	46,061	1 724,301	0,001
14.11.2015г.		0,697(+0,044)	0,004	811	21,165(+6,748)	41,599	634,812	0,031
15.11.2015г.		0,732(+0,060)	0,005	1 242	17,322(+1,337)	28,026	268,429	0,016
16.11.2015г.		0,624(-0,051)	0,067	11 054	14,667(+0,395)	34,549	1 017,210	0,001
17.11.2015г.		0,620(-0,016)	0,070	11 375	16,238(-0,705)	55,664	2 881,408	0,001
18.11.2015г.		0,614(-0,011)	0,046	7 388	13,146(-2,598)	28,114	392,343	0,001
emzkgu_3 (Кластер app-4-1)		0,597(-0,049)	0,144	34 687	23,489(-3,721)	90,162	13 381,142	0,016
11.11.2015г.		0,607(-0,136)	0,062	5 490	26,328(-2,413)	185,333	13 381,142	0,031
12.11.2015г.		0,609(-0,083)	0,052	4 610	23,778(-0,212)	45,801	854,259	0,047
13.11.2015г.		0,576(-0,031)	0,056	4 598	25,558(-2,954)	42,556	552,590	0,031
14.11.2015г.		0,516(-0,142)	0,030	2 147	18,604(-8,486)	29,355	532,150	0,031
15.11.2015г.		0,705(+0,096)	0,004	426	23,298(+0,515)	34,315	307,936	0,109
16.11.2015г.		0,597(-0,048)	0,068	5 876	27,641(-1,973)	96,268	5 950,265	0,031
17.11.2015г.		0,582(-0,068)	0,086	7 141	20,810(-6,076)	41,289	975,266	0,031
18.11.2015г.		0,645(+0,035)	0,045	4 399	18,681(-7,682)	37,232	643,075	0,016

По сервису в проекте эксплуатации системы, проблем:

- В работе – 140:
  - Новых за последнюю неделю – ____.
  - Решено за последнюю неделю – ___.

Топ дампов

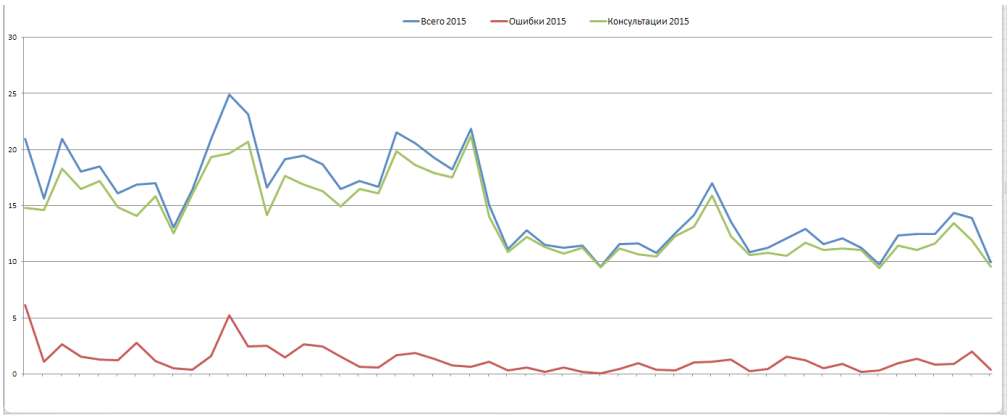
Вариант дампа	Количество
RMNGR_8.3.6.2264_FAF1C5F0	2
RMNGR_8.3.6.2359_F6C7C60A	2
RPHOST_8.3.6.2359_540245EA	2
RPHOST_8.3.6.2264_540245EA	1
Итого	**7**

Топ ошибок

- Проблемы из-за 8.2 – 49 шт.
- Проблем в работе – 11.
- Проблем, требующих перехода на релиз 8.3.6.2431, – 19.
- Проблем, требующих переноса в спецсборку, – 9.
- Проблем, требующих проверки перед закрытием, – 7.

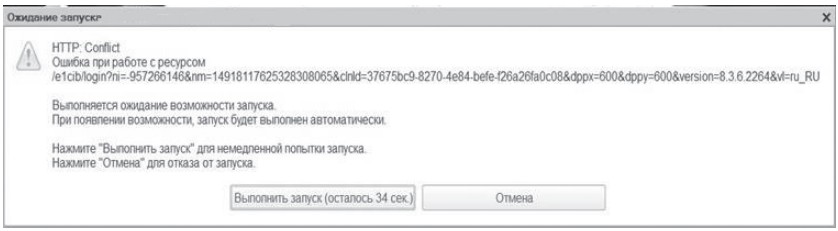
Количество инцидентов

Динамика количества инцидентов активных пользователей сервиса представлена на графике, в т. ч. по типам инцидентов, в сравнении с аналогичным периодом прошлого года.



Топ–10 жалоб пользователей за неделю; их причины; связь с задачами; что сделано, чтобы не повторялось

1. Ошибка «Bad gateway».
2. Ошибка «HTTP Conflict» (попытки обратиться не к тому клиенту).



3. Конфликты блокировок (вариации: «В БГУ постоянно ошибка блокировки таблиц», «Не могу работать в блоке з/п», «Отпуск без сохранения содержания» и «Увольнение! Появляется окно – конфликт блокировок» и т. д.) – проблема была всегда, особенно ярко проявляется во время расчета зарплаты.
4. Медленная работа программы (вариации по тексту заявок: «Ожидание формирования отчета ОСВ более 10 минут», «При открытии обработки "Обмен" программа зависает», «Очень медленная работа с документом начисления заработной платы» и т. д.).

Выполненные обновления конфигураций и платформы

- _____
- _____
- _____

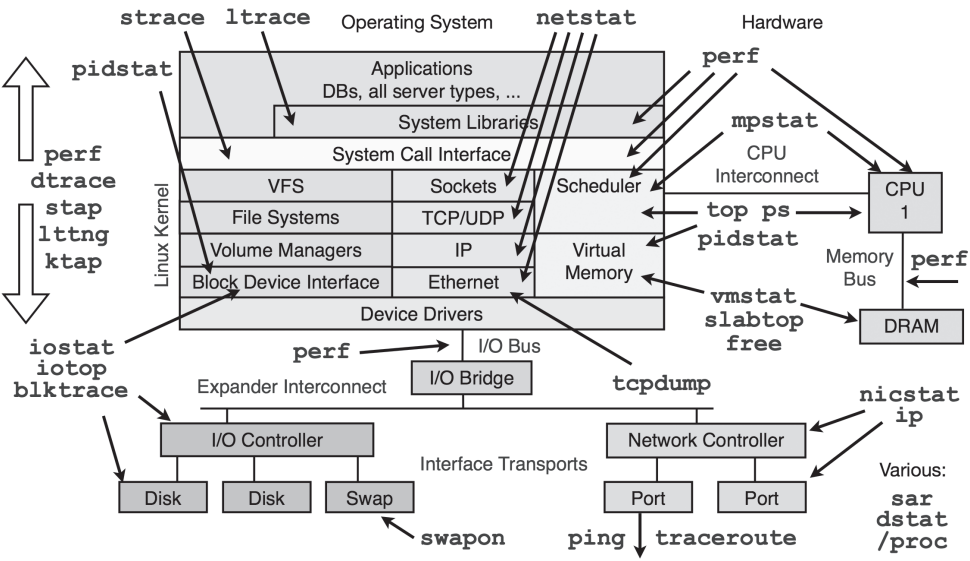
Планы обновлений конфигураций и платформы

- _____
- _____

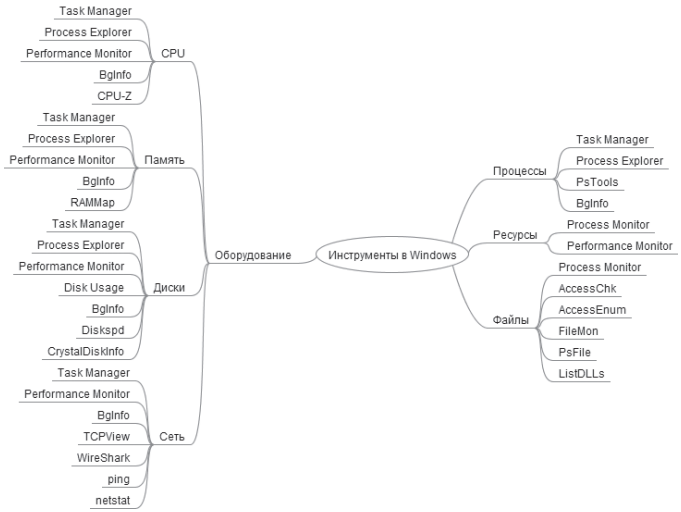
ПРИЛОЖЕНИЕ 3

«Шпаргалка» для работы с инструментами анализа производительности

Linux



Windows



Основные счетчики производительности perfmon:

- "\Memory\Available Mbytes",
- “\Process(“1cv8*”)\% Processor Time”,
- “\Process(“1cv8*”)\Private Bytes”,
- “\Process(“1cv8*”)\Virtual Bytes”,
- “\Process(“ragent*”)\% Processor Time”,
- “\Process(“ragent*”)\Private Bytes”,
- “\Process(“ragent*”)\Virtual Bytes”,
- “\Process(“rphost*”)\% Processor Time”,
- “\Process(“rphost*”)\Private Bytes”,
- “\Process(“rphost*”)\Virtual Bytes”,
- “\Process(“rmngr*”)\% Processor Time”,
- “\Process(“rmngr*”)\Private Bytes”,
- “\Process(“rmngr*”)\Virtual Bytes”,
- “\LogicalDisk(_Total)\Free Megabytes”,
- “\Processor(_Total)\% Processor Time”,
- “\Memory\Pages/sec”,
- “\System\Processor Queue Length”,
- “\PhysicalDisk(_Total)\Avg. Disk Queue Length”,
- “\PhysicalDisk(*)\Avg. Disk Queue Length”,
- “\PhysicalDisk(*)\Avg. Disk Bytes/Read”,
- “\PhysicalDisk(*)\Avg. Disk Bytes/Write”,
- “\Network Interface(*)\Bytes Total/sec”.

ПРИЛОЖЕНИЕ 4

Топ запросов к DMV MS SQL Server для расследования проблем производительности

Топ запросов, создающих нагрузку на CPU на сервере СУБД за последний час

```
SELECT
SUM(qs.max_elapsed_time) as elapsed_time,
SUM(qs.total_worker_time) as worker_time
into T1 FROM (
      select top 100000
      *
      from
      sys.dm_exec_query_stats qs
      where qs.last_execution_time > (CURRENT_TIMESTAMP  – '01:00:00.000')
      order by qs.last_execution_time desc
) as qs
;

select top 10000
(qs.max_elapsed_time) as elapsed_time,
(qs.total_worker_time) as worker_time,
qp.query_plan,
st.text,
dtb.name,
qs.*,
st.dbid
INTO T2
FROM
sys.dm_exec_query_stats qs
CROSS APPLY sys.dm_exec_query_plan(qs.plan_handle) qp
CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) st
```

```
left outer join sys.databases as dtb on st.dbid = dtb.database_id
where qs.last_execution_time > (CURRENT_TIMESTAMP  – '01:00:00.000')
order by qs.last_execution_time desc
;
select top 100
(T2.elapsed_time*100/T1.elapsed_time) as percent_elapsed_time,
(T2.worker_time*100/T1.worker_time) as percent_worker_time,
T2.*
from
T2 as T2
INNER JOIN T1 as T1
ON 1=1
order by T2.worker_time desc
;
drop table T2
;
drop table T1
;
```

Нагрузка на CPU по базам

```
WITH DB_CPU_Stats
AS
(SELECT DatabaseID, DB_Name(DatabaseID) AS [DatabaseName], SUM(total_worker_time) AS [CPU_Time_Ms]
FROM sys.dm_exec_query_stats AS qs
CROSS APPLY (SELECT CONVERT(int, value) AS [DatabaseID]
        FROM sys.dm_exec_plan_attributes(qs.plan_handle)
        WHERE attribute = N'dbid') AS F_DB
GROUP BY DatabaseID)
SELECT ROW_NUMBER() OVER(ORDER BY [CPU_Time_Ms] DESC) AS [row_num],
    DatabaseName, [CPU_Time_Ms],
    CAST([CPU_Time_Ms] * 1.0 / SUM([CPU_Time_Ms]) OVER() * 100.0 AS DECIMAL(5, 2)) AS [CPUPercent]
FROM DB_CPU_Stats
WHERE DatabaseID > 4 -- system databases
AND DatabaseID <> 32767 -- ResourceDB
ORDER BY row_num OPTION (RECOMPILE);
```

Наибольшая нагрузка на CPU

```
SELECT TOP 10
[Average CPU used] = total_worker_time / qs.execution_count
,[Total CPU used] = total_worker_time
,[Execution count] = qs.execution_count
,[Individual Query] = SUBSTRING (qt.text,qs.statement_start_offset/2,
(CASE WHEN qs.statement_end_offset = -1
THEN LEN(CONVERT(NVARCHAR(MAX), qt.text)) * 2
ELSE qs.statement_end_offset END  –
qs.statement_start_offset)/2)
,[Parent Query] = qt.text
,DatabaseName = DB_NAME(qt.dbid)
FROM sys.dm_exec_query_stats qs
CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) as qt
ORDER BY [Average CPU used] DESC;
```

Наиболее часто выполняемые запросы

```
SELECT TOP 10
[Execution count] = execution_count
,[Individual Query] = SUBSTRING (qt.text,qs.statement_start_offset/2,
      (CASE WHEN qs.statement_end_offset = -1
        THEN LEN(CONVERT(NVARCHAR(MAX), qt.text)) * 2
       ELSE qs.statement_end_offset END  – qs.statement_start_offset)/2)
,[Parent Query] = qt.text
,DatabaseName = DB_NAME(qt.dbid)
FROM sys.dm_exec_query_stats qs
CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) as qt
ORDER BY [Execution count] DESC;
```

Текущая статистика по задержкам

```
SELECT TOP 10
[Wait type] = wait_type,
[Wait time (s)] = wait_time_ms / 1000,
[% waiting] = CONVERT(DECIMAL(12,2), wait_time_ms * 100.0
          / SUM(wait_time_ms) OVER())
FROM sys.dm_os_wait_stats
WHERE wait_type NOT LIKE '%SLEEP%'
ORDER BY wait_time_ms DESC;
```

Индексы с высокими издержками при использовании

```
-- Create required table structure only.
-- Note: this SQL must be the same as in the Database loop given in the following step.
SELECT TOP 1
      [Maintenance cost]  = (user_updates + system_updates)
      ,[Retrieval usage] = (user_seeks + user_scans + user_lookups)
      ,DatabaseName = DB_NAME()
      ,TableName = OBJECT_NAME(s.[object_id])
      ,IndexName = i.name
INTO #TempMaintenanceCost
FROM   sys.dm_db_index_usage_stats s
INNER JOIN sys.indexes i ON  s.[object_id] = i.[object_id]
  AND s.index_id = i.index_id
WHERE s.database_id = DB_ID()
  AND OBJECTPROPERTY(s.[object_id], 'IsMsShipped') = 0
  AND (user_updates + system_updates) > 0 -- Only report on active rows.
  AND s.[object_id] = -999  -- Dummy value to get table structure.
;

-- Loop around all the databases on the server.
EXEC sp_MSForEachDB    'USE [?];
-- Table already exists.
INSERT INTO #TempMaintenanceCost
SELECT TOP 10
      [Maintenance cost]  = (user_updates + system_updates)
      ,[Retrieval usage] = (user_seeks + user_scans + user_lookups)
      ,DatabaseName = DB_NAME()
      ,TableName = OBJECT_NAME(s.[object_id])
      ,IndexName = i.name
FROM   sys.dm_db_index_usage_stats s
INNER JOIN sys.indexes i ON  s.[object_id] = i.[object_id]
  AND s.index_id = i.index_id
WHERE s.database_id = DB_ID()
```

```
  AND i.name IS NOT NULL    -- Ignore HEAP indexes.
  AND OBJECTPROPERTY(s.[object_id], ''IsMsShipped'') = 0
  AND (user_updates + system_updates) > 0 -- Only report on active rows.
ORDER BY [Maintenance cost]  DESC
;
'
-- Select records.
SELECT TOP 10 * FROM #TempMaintenanceCost
ORDER BY [Maintenance cost]  DESC
-- Tidy up.
DROP TABLE #TempMaintenanceCost
```

Запросы, создающие нагрузку на диск

```
SELECT
SUM(qs.total_physical_reads) as physical_reads,
SUM(qs.total_logical_reads) as logical_reads
into T1 FROM (
select top 100000
*
from
sys.dm_exec_query_stats qs
where qs.last_execution_time > (CURRENT_TIMESTAMP  – '01:00:00.000')
order by qs.total_physical_reads desc
) as qs
;
select top 100
(qs.total_physical_reads) as physical_reads,
(qs.total_logical_reads) as logical_reads,
qp.query_plan,
st.text,
dtb.name,
qs.*,
st.dbid
INTO T2
FROM
sys.dm_exec_query_stats qs
CROSS APPLY sys.dm_exec_query_plan(qs.plan_handle) qp
CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) st
left outer join sys.databases as dtb on st.dbid = dtb.database_id
where qs.last_execution_time > (CURRENT_TIMESTAMP  – '01:00:00.000')
order by qs.total_physical_reads desc
;
select
(T2.physical_reads*100/T1.physical_reads) as percent_physical_reads,
(T2.logical_reads*100/T1.logical_reads) as percent_logical_reads,
T2.*
from
T2 as T2
INNER JOIN T1 as T1
ON 1=1
order by T2.total_physical_reads desc
;
drop table T2
;
drop table T1
;
```

Базы, создающие нагрузку на диск

```
WITH DB_Disk_Reads_Stats
AS
(SELECT DatabaseID, DB_Name(DatabaseID) AS [DatabaseName], SUM(total_physical_reads) AS [physical_reads]
 FROM sys.dm_exec_query_stats AS qs
 CROSS APPLY (SELECT CONVERT(int, value) AS [DatabaseID]
              FROM sys.dm_exec_plan_attributes(qs.plan_handle)
              WHERE attribute = N'dbid') AS F_DB
 GROUP BY DatabaseID)
SELECT ROW_NUMBER() OVER(ORDER BY [physical_reads] DESC) AS [row_num],
       DatabaseName, [physical_reads],
       CAST([physical_reads] * 1.0 / SUM([physical_reads]) OVER() * 100.0 AS DECIMAL(5, 2)) AS [Physical_Reads_Percent]
FROM DB_Disk_Reads_Stats
WHERE DatabaseID > 4 -- system databases
AND DatabaseID <> 32767 -- ResourceDB
ORDER BY row_num OPTION (RECOMPILE);
```

Длительные транзакции

```
DECLARE @curr_date as DATETIME
SET @curr_date = GETDATE()
select --SESSION_TRAN.*,
SESSION_TRAN.session_id AS connectID, -- "Соединение с СУБД" в консоли кластера 1С
--TRAN_INFO.*,
TRAN_INFO.transaction_begin_time,
DateDiff(MINUTE, TRAN_INFO.transaction_begin_time, @curr_date) AS Duration, -- Длительность в минутах
TRAN_INFO.transaction_type, -- 1 = транзакция чтения-записи; 2 = транзакция только для чтения;
3 = системная транзакция; 4 = распределенная транзакция.
TRAN_INFO.transaction_state,
-- 0 = Транзакция еще не была полностью инициализирована;
-- 1 = Транзакция была инициализирована, но еще не началась;
-- 2 = Транзакция активна;
-- 3 = Транзакция закончилась. Используется для транзакций «только для чтения»;
-- 4 = Фиксирующий процесс был инициализирован на распределенной транзакции. Предназначено только
       для распределенных транзакций. Распределенная транзакция все еще активна, но дальнейшая обработка
       не может иметь место;
-- 5 = Транзакция находится в готовом состоянии и ожидает разрешения;
-- 6 = Транзакция зафиксирована;
-- 7 = Производится откат транзакции;
-- 8 = откат транзакции был выполнен.
--CONN_INFO.*,
CONN_INFO.connect_time,
CONN_INFO.num_reads,
CONN_INFO.num_writes,
CONN_INFO.last_read,
CONN_INFO.last_write,
CONN_INFO.client_net_address,
CONN_INFO.most_recent_sql_handle,
--SQL_TEXT.*,
SQL_TEXT.dbid,
db_name(SQL_TEXT.dbid) AS IB_NAME,
SQL_TEXT.text,
--QUERIES_INFO.*,
QUERIES_INFO.start_time,
QUERIES_INFO.status,
QUERIES_INFO.command,
QUERIES_INFO.wait_type,
QUERIES_INFO.wait_time,
PLAN_INFO.query_plan
```

```
FROM sys.dm_tran_session_transactions AS SESSION_TRAN
JOIN sys.dm_tran_active_transactions AS TRAN_INFO
ON SESSION_TRAN.transaction_id = TRAN_INFO.transaction_id
LEFT JOIN sys.dm_exec_connections AS CONN_INFO
ON SESSION_TRAN.session_id = CONN_INFO.session_id
CROSS APPLY sys.dm_exec_sql_text(CONN_INFO.most_recent_sql_handle) AS SQL_TEXT
LEFT JOIN sys.dm_exec_requests AS QUERIES_INFO
ON SESSION_TRAN.session_id = QUERIES_INFO.session_id
LEFT JOIN (
SELECT VL_SESSION_TRAN.session_id AS session_id,
VL_PLAN_INFO.query_plan AS query_plan
FROM sys.dm_tran_session_transactions AS VL_SESSION_TRAN
INNER JOIN sys.dm_exec_requests AS VL_QUERIES_INFO
ON VL_SESSION_TRAN.session_id = VL_QUERIES_INFO.session_id
CROSS APPLY sys.dm_exec_text_query_plan(VL_QUERIES_INFO.plan_handle, VL_QUERIES_INFO.statement_start_offset,
VL_QUERIES_INFO.statement_end_offset) AS VL_PLAN_INFO) AS PLAN_INFO
ON SESSION_TRAN.session_id = PLAN_INFO.session_id
ORDER BY transaction_begin_time ASC
```

Наиболее часто выполняемые запросы

```
SELECT TOP 100
[Execution count] = execution_count
,[Individual Query] = SUBSTRING (qt.text,qs.statement_start_offset/2,
(CASE WHEN qs.statement_end_offset = -1
THEN LEN(CONVERT(NVARCHAR(MAX), qt.text)) * 2
ELSE qs.statement_end_offset END  – qs.statement_start_offset)/2)
,[Parent Query] = qt.text
,DatabaseName = DB_NAME(qt.dbid)
FROM sys.dm_exec_query_stats qs
CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) as qt
ORDER BY [Execution count] DESC;
```

Запросы с высокими издержками на ввод/вывод

```
SELECT TOP 100
[Average IO] = (total_logical_reads + total_logical_writes) / qs.execution_count
,[Total IO] = (total_logical_reads + total_logical_writes)
,[Execution count] = qs.execution_count
,[Individual Query] = SUBSTRING (qt.text,qs.statement_start_offset/2,
(CASE WHEN qs.statement_end_offset = -1
THEN LEN(CONVERT(NVARCHAR(MAX), qt.text)) * 2
ELSE qs.statement_end_offset END  – qs.statement_start_offset)/2)
,[Parent Query] = qt.text
,DatabaseName = DB_NAME(qt.dbid)
FROM sys.dm_exec_query_stats qs
CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) as qt
ORDER BY [Average IO] DESC;
```

Использование кешей сервера СУБД

```
SELECT TOP(100) [type], SUM(single_pages_kb) AS [SPA Mem, Kb]
FROM sys.dm_os_memory_clerks
GROUP BY [type]
ORDER BY SUM(single_pages_kb) DESC;
```

Использование кешей по базам данных сервера СУБД

```
SELECT DB_NAME(database_id) AS DB,COUNT(row_count)*8.00/1024.00 AS MB, COUNT(row_count)*8.00/1024.00/1024.00 AS GB
FROM sys.dm_os_buffer_descriptors
GROUP BY database_id
ORDER BY MB DESC
```

Свободно в tempdb

```
SELECT SUM(unallocated_extent_page_count) AS [free pages],
(SUM(unallocated_extent_page_count)*1.0/128) AS [free space in MB]
FROM sys.dm_db_file_space_usage;
```

Список длительных транзакций

```
SELECT transaction_id, *
FROM sys.dm_tran_active_snapshot_database_transactions
ORDER BY elapsed_time_seconds DESC;
```

ПРИЛОЖЕНИЕ 5

Топ запросов к PostgreSQL для расследования проблем производительности

Выполняющиеся запросы с их продолжительностью

```
SELECT
  st.state, st.*
  FROM
  pg_stat_activity as st
  where st.state = 'active'
```

Список текущих блокировок

```
SELECT
  l.mode,
  d.datname,
  c.relname,
  l.granted,
  l.transactionid
  FROM
  pg_locks AS l
  LEFT JOIN pg_database AS d ON l.database= d.oid
  LEFT JOIN pg_class AS c ON l.relation = c.oid;
```

Неустановленные блокировки

```
SELECT relation::regclass, * FROM pg_locks WHERE NOT GRANTED;
```

Транзакции, ожидающие на блокировках

```
SELECT blocked_locks.pid     AS blocked_pid,
       blocked_activity.usename  AS blocked_user,
       blocking_locks.pid     AS blocking_pid,
       blocking_activity.usename AS blocking_user,
       blocked_activity.query    AS blocked_statement,
       blocking_activity.query   AS current_statement_in_blocking_process,
       blocked_activity.application_name AS blocked_application,
       blocking_activity.application_name AS blocking_application
  FROM  pg_catalog.pg_locks         blocked_locks
   JOIN pg_catalog.pg_stat_activity blocked_activity  ON blocked_activity.pid = blocked_locks.pid
   JOIN pg_catalog.pg_locks         blocking_locks
      ON blocking_locks.locktype = blocked_locks.locktype
      AND blocking_locks.DATABASE IS NOT DISTINCT FROM blocked_locks.DATABASE
      AND blocking_locks.relation IS NOT DISTINCT FROM blocked_locks.relation
      AND blocking_locks.page IS NOT DISTINCT FROM blocked_locks.page
      AND blocking_locks.tuple IS NOT DISTINCT FROM blocked_locks.tuple
      AND blocking_locks.virtualxid IS NOT DISTINCT FROM blocked_locks.virtualxid
      AND blocking_locks.transactionid IS NOT DISTINCT FROM blocked_locks.transactionid
      AND blocking_locks.classid IS NOT DISTINCT FROM blocked_locks.classid
      AND blocking_locks.objid IS NOT DISTINCT FROM blocked_locks.objid
      AND blocking_locks.objsubid IS NOT DISTINCT FROM blocked_locks.objsubid
      AND blocking_locks.pid != blocked_locks.pid
   JOIN pg_catalog.pg_stat_activity blocking_activity ON blocking_activity.pid = blocking_locks.pid
  WHERE NOT blocked_locks.GRANTED;
```

Объем таблиц

```
SELECT
   table_name,
   pg_size_pretty(table_size) AS table_size,
   pg_size_pretty(indexes_size) AS indexes_size,
   pg_size_pretty(total_size) AS total_size
FROM (
   SELECT
      table_name,
      pg_table_size(table_name) AS table_size,
      pg_indexes_size(table_name) AS indexes_size,
      pg_total_relation_size(table_name) AS total_size
   FROM (
      SELECT ('"' || table_schema || '"."' || table_name || '"') AS table_name
      FROM information_schema.tables
   ) AS all_tables
   ORDER BY total_size DESC
) AS pretty_sizes
```

Для сбора статистики по запросам в postgres необходимо:

- Открыть файл postgresql.conf.
- Добавить в список загружаемых библиотек (shared_preload_libraries) библиотеку pg_stat_statements.
- Настроить параметры pg_stat_statements.
- Перезапустить postgres.

Параметры pg_stat_statements:

- pg_stat_statements.max – определяет максимальное число запросов, отлеживаемых модулем статистики. Если число запросов превышает значение параметра, то самые редко используемые удаляются из статистики;
- pg_stat_statements.track – определяет уровень отслеживаемых запросов: top – только пользовательские, all – все, none – не отслеживать;
- pg_stat_statements.save – on/off сохранять статистику при перезапуске postgres.

Для большинства задач достаточно настройки:

```
# postgresql.conf
shared_preload_libraries = 'pg_stat_statements'

pg_stat_statements.max = 10000
pg_stat_statements.track = all
```

По умолчанию данные статистики по запросам недоступны. Их необходимо подключить с помощью команды:

```
CREATE EXTENSION pg_stat_statements
```

Размер базы данных

```
select pg_size_pretty(pg_database_size(current_database()));
```

Размер таблицы

```
select pg_size_pretty(pg_relation_size('table name'));
```

Процент попадания в кеш

```
SELECT
  round(100 * sum(blks_hit) / sum(blks_hit + blks_read), 3) as cache_hit_ratio
FROM pg_stat_database;
```

Получить идентификатор процесса

```
SELECT pid, * FROM pg_stat_activity
```

Перезагрузка конфигурации без рестарта сервера

```
select pg_reload_conf();
```

Размер таблиц и индексов

```
SELECT
   t.tablename,
   indexname,
   c.reltuples::integer AS num_rows,
   pg_size_pretty(pg_relation_size(quote_ident(t.tablename)::text)) AS table_size,
   pg_size_pretty(pg_relation_size(quote_ident(indexrelname)::text)) AS index_size,
   CASE WHEN x.is_unique = 1  THEN 'Y'
      ELSE 'N'
   END AS UNIQUE,
   idx_scan AS number_of_scans,
   idx_tup_read AS tuples_read,
   idx_tup_fetch AS tuples_fetched
FROM pg_tables t
LEFT OUTER JOIN pg_class c ON t.tablename=c.relname
LEFT OUTER JOIN
      (SELECT indrelid,
         max(CAST(indisunique AS integer)) AS is_unique
      FROM pg_index
      GROUP BY indrelid) x
      ON c.oid = x.indrelid
LEFT OUTER JOIN
   ( SELECT c.relname AS ctablename, ipg.relname AS indexname, x.indnatts AS number_of_columns, idx_scan,
                                                idx_tup_read, idx_tup_fetch,indexrelname FROM pg_index x
         JOIN pg_class c ON c.oid = x.indrelid
         JOIN pg_class ipg ON ipg.oid = x.indexrelid
         JOIN pg_stat_all_indexes psai ON x.indexrelid = psai.indexrelid )
   AS foo
   ON t.tablename = foo.ctablename
WHERE t.schemaname='public'
ORDER BY pg_relation_size(quote_ident(indexrelname)::text) desc;
```

Таблицы с максимальным пустым местом

```
-- установка расширения
create extension pg_freespacemap;

select
   result.table_name,
   result.freespace as fororder,
   pg_size_pretty(result.size) as size,
   pg_size_pretty(result.freespace) as freespace,
   result.freespace * 100 / result.size as percent_free
from
   (select
      t.table_name,
      pg_total_relation_size(t.table_name) as size,
      sum(s.avail) as freespace
   from
      (select table_name from information_schema.tables where table_schema = 'public') as t,
      LATERAL pg_freespace(t.table_name) as s
   group by
      t.table_name) as result
order by
    fororder desc
limit 20;
```

Размер временных таблиц «1С»

```
select relnamespace, sum(relpages::bigint*8192), pg_size_pretty(sum(relpages::bigint*8192))
                                from pg_class where relname like 'tt%' group by 1 order by 2 desc;

select sum(relpages::bigint*8192), pg_size_pretty(sum(relpages::bigint*8192)) from pg_class where relname like 'tt%';
```

Базы данных с максимальным IO

```
SELECT
  pg_stat_database.datname AS DatabaseName,
  MAX(pg_stat_database.tup_returned) AS TotalReads,
  MAX(pg_stat_database.tup_inserted + pg_stat_database.tup_updated + pg_stat_database.tup_deleted) AS TotalWrites,
  MAX(pg_stat_database.tup_returned + pg_stat_database.tup_inserted + pg_stat_database.tup_updated
                                                                    + pg_stat_database.tup_deleted) AS IO,
  SUM(pg_stat_statements.calls) AS ExecutionCount

FROM
  pg_catalog.pg_stat_database AS pg_stat_database
  LEFT OUTER JOIN pg_stat_statements AS pg_stat_statements
  ON pg_stat_statements.dbid = pg_stat_database.datid

GROUP BY
  pg_stat_database.datname

ORDER BY
  IO Desc
```

Отсутствие индексов

```
SELECT
  relname,
  seq_scan  – coalesce(idx_scan, 0) AS too_much_seq,
  case when seq_scan  – coalesce(idx_scan, 0) > 0 THEN 'Нет индекса?' ELSE 'OK' END AS Message,
  pg_relation_size(relname::regclass) AS rel_size,
  seq_scan,
  coalesce(idx_scan, 0) AS idx_scan
 FROM
   pg_stat_all_tables
 WHERE
   schemaname='public'
   AND pg_relation_size(relname::regclass)>10000 -- ограничение на  размер анализируемых таблиц
 ORDER BY
   too_much_seq DESC;
```

Использование индексов

```
SELECT
    t.tablename,
    indexname,
    c.reltuples AS num_rows,
    pg_size_pretty(pg_relation_size(quote_ident(t.tablename)::text)) AS table_size,
    pg_size_pretty(pg_relation_size(quote_ident(indexrelname)::text)) AS index_size,
    CASE WHEN indisunique THEN 'Y' ELSE 'N' END AS UNIQUE,
    idx_scan AS number_of_scans, idx_tup_read AS tuples_read, idx_tup_fetch AS tuples_fetched
FROM pg_tables t
LEFT OUTER JOIN pg_class c ON t.tablename=c.relname
LEFT OUTER JOIN
```

```
   ( SELECT c.relname AS ctablename, ipg.relname AS indexname, x.indnatts AS number_of_columns,
                idx_scan, idx_tup_read, idx_tup_fetch, indexrelname, indisunique FROM pg_index x
         JOIN pg_class c ON c.oid = x.indrelid JOIN pg_class ipg ON ipg.oid = x.indexrelid
         JOIN pg_stat_all_indexes psai ON x.indexrelid = psai.indexrelid )
   AS foo
   ON t.tablename = foo.ctablename
WHERE t.schemaname='public'
ORDER BY 1,2;

--------------------------------------------------------------------
SELECT
        idstat.relname AS table_name,
        indexrelname AS index_name,
        idstat.idx_scan AS times_used,
        pg_size_pretty(pg_relation_size(tabstat.relid)) AS table_size,
        pg_size_pretty(pg_relation_size(indexrelid)) AS index_size,
        n_tup_upd + n_tup_ins + n_tup_del as num_writes,
        indexdef AS definition
FROM
        pg_stat_user_indexes AS idstat
        JOIN pg_indexes ON indexrelname = indexname
        JOIN pg_stat_user_tables AS tabstat ON idstat.relname = tabstat.relname
WHERE
        idstat.idx_scan < 200
        AND indexdef !~* 'unique'
ORDER BY
        idstat.relname,
        indexrelname;

--------------------------------------------------------------------
WITH table_scans as (
   SELECT relid,
      tables.idx_scan + tables.seq_scan as all_scans,
      ( tables.n_tup_ins + tables.n_tup_upd + tables.n_tup_del ) as writes,
            pg_relation_size(relid) as table_size
      FROM pg_stat_user_tables as tables
),
all_writes as (
   SELECT sum(writes) as total_writes
   FROM table_scans
),
indexes as (
   SELECT idx_stat.relid, idx_stat.indexrelid,
      idx_stat.schemaname, idx_stat.relname as tablename,
      idx_stat.indexrelname as indexname,
      idx_stat.idx_scan,
      pg_relation_size(idx_stat.indexrelid) as index_bytes,
      indexdef ~* 'USING btree' AS idx_is_btree
   FROM pg_stat_user_indexes as idx_stat
      JOIN pg_index
         USING (indexrelid)
      JOIN pg_indexes as indexes
         ON idx_stat.schemaname = indexes.schemaname
            AND idx_stat.relname = indexes.tablename
            AND idx_stat.indexrelname = indexes.indexname
   WHERE pg_index.indisunique = FALSE
),
index_ratios AS (
SELECT schemaname, tablename, indexname,
   idx_scan, all_scans,
   round(( CASE WHEN all_scans = 0 THEN 0.0::NUMERIC
      ELSE idx_scan::NUMERIC/all_scans * 100 END),2) as index_scan_pct,
   writes,
```

```
    round((CASE WHEN writes = 0 THEN idx_scan::NUMERIC ELSE idx_scan::NUMERIC/writes END),2)
        as scans_per_write,
    pg_size_pretty(index_bytes) as index_size,
    pg_size_pretty(table_size) as table_size,
    idx_is_btree, index_bytes
    FROM indexes
    JOIN table_scans
    USING (relid)
),
index_groups AS (
SELECT 'Never Used Indexes' as reason, *, 1 as grp
FROM index_ratios
WHERE
    idx_scan = 0
    and idx_is_btree
UNION ALL
SELECT 'Low Scans, High Writes' as reason, *, 2 as grp
FROM index_ratios
WHERE
    scans_per_write <= 1
    and index_scan_pct < 10
    and idx_scan > 0
    and writes > 100
    and idx_is_btree
UNION ALL
SELECT 'Seldom Used Large Indexes' as reason, *, 3 as grp
FROM index_ratios
WHERE
    index_scan_pct < 5
    and scans_per_write > 1
    and idx_scan > 0
    and idx_is_btree
    and index_bytes > 100000000
UNION ALL
SELECT 'High-Write Large Non-Btree' as reason, index_ratios.*, 4 as grp
FROM index_ratios, all_writes
WHERE
    ( writes::NUMERIC / total_writes ) > 0.02
    AND NOT idx_is_btree
    AND index_bytes > 100000000
ORDER BY grp, index_bytes DESC )
SELECT reason, schemaname, tablename, indexname,
    index_scan_pct, scans_per_write, index_size, table_size
FROM index_groups;
```

Нагрузка, создаваемая запросами

```
SELECT
  pg_database.datname AS Database,
  pg_stat_statements.query AS Query,
  pg_stat_statements.calls AS ExecutionCount,
  pg_stat_statements.total_time ExecutionTime,
  pg_stat_statements.shared_blks_read + pg_stat_statements.shared_blks_written AS Memory,
  pg_stat_statements.local_blks_read + pg_stat_statements.local_blks_written AS IO,
  pg_stat_statements.temp_blks_read + pg_stat_statements.temp_blks_written AS Temp

FROM
  pg_stat_statements AS pg_stat_statements
  INNER JOIN pg_database AS pg_database
  ON pg_database.oid = pg_stat_statements.dbid
```

```
ORDER BY
  ExecutionTime DESC



with s AS (SELECT
CASE WHEN sum(total_time) = 0 THEN 1 ELSE sum(total_time) END AS sum_time,
CASE WHEN sum(blk_read_time+blk_write_time) = 0 THEN 1 ELSE sum(blk_read_time+blk_write_time) END as sum_iotime,
CASE WHEN sum(total_time-blk_read_time-blk_write_time) = 0 THEN 1 ELSE sum(total_time-blk_read_time-blk_write_time)
                                                                                                    END as sum_cputime,
CASE WHEN sum(calls) = 0 THEN 1 ELSE sum(calls) END AS sum_calls,
CASE WHEN sum(rows) = 0 THEN 1 ELSE sum(rows) END  as sum_rows
FROM pg_stat_statements
)

SELECT
'all_users@all_databases' as "user@database",
100 AS time_percent,
100 AS iotime_percent,
100 AS cputime_percent,
(sum_time/1000)*'1 second'::interval as total_time,
(sum_cputime/sum_calls)::numeric(20, 2) AS avg_cpu_time_ms,
(sum_iotime/sum_calls)::numeric(20, 2) AS avg_io_time_ms,
sum_calls as calls,
100 AS calls_percent,
sum_rows as rows,
100 as row_percent,
'all_queries' as query
FROM s

UNION ALL

SELECT
(select rolname from pg_roles where oid = p.userid) || '@' || (select datname from pg_database where oid = p.dbid)
                                                                                                    as "user@database",
(100*total_time/(SELECT sum_time FROM s))::numeric(20, 2) AS time_percent,
(100*(blk_read_time+blk_write_time)/(SELECT sum_iotime FROM s))::numeric(20, 2) AS iotime_percent,
(100*(total_time-blk_read_time-blk_write_time)/(SELECT sum_cputime FROM s))::numeric(20, 2) AS cputime_percent,
(total_time/1000)*'1 second'::interval as total_time,
((total_time-blk_read_time-blk_write_time)/calls)::numeric(20, 2) AS avg_cpu_time_ms,
((blk_read_time+blk_write_time)/calls)::numeric(20, 2) AS avg_io_time_ms,
calls,
(100*calls/(SELECT sum_calls FROM s))::numeric(20, 2) AS calls_percent,
rows,
(100*rows/(SELECT sum_rows from s))::numeric(20, 2) AS row_percent,
query
FROM pg_stat_statements p
WHERE
(total_time-blk_read_time-blk_write_time)/(SELECT sum_cputime FROM s)>=0.005

UNION ALL

SELECT
'all_users@all_databases' as "user@database",
(100*sum(total_time)/(SELECT sum_time FROM s))::numeric(20, 2) AS time_percent,
(100*sum(blk_read_time+blk_write_time)/(SELECT sum_iotime FROM s))::numeric(20, 2) AS iotime_percent,
(100*sum(total_time-blk_read_time-blk_write_time)/(SELECT sum_cputime FROM s))::numeric(20, 2) AS cputime_percent,
(sum(total_time)/1000)*'1 second'::interval,
(sum(total_time-blk_read_time-blk_write_time)/sum(calls))::numeric(10, 3) AS avg_cpu_time_ms,
(sum(blk_read_time+blk_write_time)/sum(calls))::numeric(10, 3) AS avg_io_time_ms,
sum(calls),
(100*sum(calls)/(SELECT sum_calls FROM s))::numeric(20, 2) AS calls_percent,
```

```
sum(rows),
(100*sum(rows)/(SELECT sum_rows from s))::numeric(20, 2) AS row_percent,
'other' AS query
FROM pg_stat_statements p
WHERE
(total_time-blk_read_time-blk_write_time)/(SELECT sum_cputime FROM s)<0.005

ORDER BY 4 DESC;
```

Использование буферного кеша

```
SELECT
  'total', pg_size_pretty(count(*) * (SELECT current_setting('block_size')::int))
FROM pg_buffercache

UNION SELECT
  'dirty', pg_size_pretty(count(*) * (SELECT current_setting('block_size')::int))
FROM pg_buffercache
WHERE isdirty

UNION SELECT
  'clear', pg_size_pretty(count(*) * (SELECT current_setting('block_size')::int))
FROM pg_buffercache
WHERE NOT isdirty

UNION SELECT
  'used', pg_size_pretty(count(*) * (SELECT current_setting('block_size')::int))
FROM pg_buffercache
WHERE reldatabase IS NOT NULL

UNION SELECT
  'free',pg_size_pretty(count(*) * (SELECT current_setting('block_size')::int))
FROM pg_buffercache
WHERE reldatabase IS NULL;


SELECT
  d.datname,
  pg_size_pretty(pg_database_size(d.datname)) AS database_size,
  pg_size_pretty(count(b.bufferid) * (SELECT current_setting('block_size')::int)) AS size_in_shared_buffers,
  round((100 * count(b.bufferid) / (SELECT setting FROM pg_settings WHERE name = 'shared_buffers')::decimal),2)
                                                                                                    AS pct_of_shared_buffers
FROM pg_buffercache b
JOIN pg_database d ON b.reldatabase = d.oid
WHERE b.reldatabase IS NOT NULL
GROUP BY 1 ORDER BY 4 DESC LIMIT 10;
```

ПРИЛОЖЕНИЕ 6

Check-лист по настройке серверов

Настройку серверов в продукционной зоне рекомендуется выполнять в соответствии с приведенным ниже check-листом.

1. Настроен сбор технологических журналов на всех продукционных рабочих серверах с платформой «1С:Предприятие»:
   - секция ALL (события EXCP, PROC, CONN, SESN, CLSTR, SRVC, ADMIN);
   - секция CALLSCALL (события CALL, SCALL);
   - сбор дампов аварийных завершений процессов кластера.

   Настройка выполняется с помощью файла logcfg.xml.

   Пример настройки технологического журнала:

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="http://v8.1c.ru/v8/tech-log">
<dump create="true" location="C:\DUMPS" type="3" prntscrn="false" externaldump="1"/>
<log location="C:\LOGS\All" history="28">
        <event>
                <eq property="Name" value="EXCP"/>
        </event>
        <event>
                <eq property="Name" value="EXCPCNTX"/>
        </event>
        <event>
                <eq property="Name" value="PROC"/>
        </event>
        <event>
                <eq property="Name" value="ADMIN"/>
        </event>
        <event>
                <eq property="Name" value="CONN"/>
        </event>
        <event>
                <eq property="Name" value="SESN"/>
        </event>
```

```
        <event>
                <eq property="Name" value="CLSTR"/>
        </event>
        <property name="all"/>
</log>
<log location="C:\LOGS\CallScall" history="28">
        <event>
                <eq property="Name" value="CALL"/>
        </event>
        <event>
                <eq property="Name" value="SCALL"/>
        </event>
        <property name="Context">
    <event>
      <eq property="name" value=""/>
    </event>
        </property>
        <property name="all"/>
</log>
</config>
```

В данном примере события CALL и SCALL собираются без контекстов, что в значительной мере сокращает объем собираемых журналов. Однако в некоторых случаях контексты необходимы.

Тогда часть:

```
        <property name="Context">
    <event>
      <eq property="name" value=""/>
    </event>
        </property>
```

необходимо будет удалить.

Обратите внимание: history="28" указан равным 28 часам, что при достаточно интенсивной нагрузке может привести к сбору достаточно большого объема журналов.

Если места на дисках недостаточно, лучше отказаться от журнала с CALL, SCALL, а сбор «основного» журнала "C:\LOGS\All" сократить до 12 часов.

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="http://v8.1c.ru/v8/tech-log">
<dump create="true" location="C:\DUMPS" type="3" prntscrn="false" externaldump="1"/>
<log location="C:\LOGS\All" history="12">
        <event>
                <eq property="Name" value="EXCP"/>
        </event>
        <event>
                <eq property="Name" value="EXCPCNTX"/>
        </event>
        <event>
                <eq property="Name" value="PROC"/>
        </event>
        <event>
                <eq property="Name" value="ADMIN"/>
        </event>
        <event>
                <eq property="Name" value="CONN"/>
```

```
        </event>
        <event>
                <eq property="Name" value="SESN"/>
        </event>
        <event>
                <eq property="Name" value="CLSTR"/>
        </event>
        <property name="all"/>
</log>
</config>
```

Следует также настроить архивирование технологических журналов с последующим копированием на сетевой ресурс.

При настройке технологических журналов следует постараться не увеличивать число секций журналов до десятков, так как на обработку и анализ необходимости журналирования всех событий каждой секции (<log>…</log>) все же тратится процессорное время.

Настроен сбор счетчиков Performance Monitor на всех продукционных серверах. Запуск счетчика добавлен в Task Schedule.

Впрочем, при применении любых настроек или автоматизации следует убеждаться, что они будут работать после перезагрузки сервера. Самый простой способ проверить, что все сделано правильно, – перезапустить сервер, пока он еще не встроен в рабочую систему.

Настроен Windows Error Reporting Service на процессы ragent, rmngr, rphost:

http://kb.1c.ru/articleView.jsp?id=96

Дополнительную информацию о настройке WER можно получить здесь:

http://msdn.microsoft.com/en-us/library/windows/desktop/bb787181%28v=vs.85%29.aspx

http://msdn.microsoft.com/en-us/library/bb513638(VS.85).aspx

На рабочих серверах есть только продукционные кластеры с продукционными базами.

Другие (не рабочие) базы и кластеры должны отсутствовать.

«Исключением» из данного правила могут являться «обслуживающие» (хотя все же продукционные) информационные базы, такие как:

- «Центр контроля качества».
- «Центр управления производительностью».
- Менеджер сервиса.
- Агент сервиса и т. п.

Обратите внимание, что не должно быть:

- развернутых копий продукционных информационных баз;
- тестовых информационных баз;
- информационных баз разработки.

Выполнены настройки перезапуска процессов по памяти для всех серверов:

http://kb.1c.ru/articleView.jsp?id=79

Ограничения по безопасному расходу памяти на вызов – либо «По умолчанию», либо имеют такие значения, при которых они действительно могут сработать на конкретном оборудовании.

Не должно быть установлено таких значений, которые «отключают» работу механизмов контроля потребления памяти на вызов.

Наличие утилит:

- Process Explorer;
- RAMMap;
- ProcDump;
- ProcMon

в определенной директории, например C:\Utils.

Ярлык для этой директории должен быть на рабочем столе.

2. Наличие скриптов автоматизации в определенной директории, например C:\Scripts. Ярлык для этой директории должен быть на рабочем столе.
3. Настроена контрольная процедура «Контроль подключений» для всех продукционных баз в «Центре контроля качества».
4. Настроен контроль рабочих серверов с помощью агента ЦКК.
5. Настроена контрольная процедура «Контроль потребления памяти» для всех продукционных баз в «Центре контроля качества».
6. Настроен сбор данных по загруженности оборудования для всех серверов продукционной площадки в ЦКК из мастера настройки в ЦКК.

   У пользователя ОС, от имени которого запущен rphost, – вхождение в группу **Windows Performance Monitor Users**.
7. Настроены оповещения в ЦКК:
   - по месту на дисках для всех продукционных серверов: меньше 10 % места за 5 минут;
   - по памяти для всех продукционных серверов (кроме СУБД): меньше 200 Мб памяти за 5 минут;
   - по CPU для всех продукционных серверов: более 90 % за 5 минут;
   - для СУБД по превышению размера tempdb (SQL Tempdb Log Used Size (MB) [среднее]) более 70 Гб (нужно убедиться в снятом ограничении по размеру tempdb в настройках СУБД).
8. Проведен успешный тест отправки e-mail или SMS из рабочего «Центра контроля качества». Если тест успешно не выполняется (т. е. SMS или письмо не приходят), нужно проверять наличие доступа к соответствующим серверам.
9. На веб-серверах настроен сбор access- и error-логов IIS или Apache.
10. Настроены регламентные процедуры обслуживания:
    - обновление статистик;

- □ очистка процедурного кеша;
- □ дефрагментация индексов;
- □ реиндексация таблиц базы данных;
- □ создание бэкапов.

11. Установлен Windows Performance Toolkit на всех рабочих (1С) Windows-серверах. Windows Performance Toolkit входит в комплект **Windows 8.1 SDK**.

    После установки проверяется возможность запустить Windows Performance Recorder с опцией Resource Analysis\CPU Usage на несколько секунд.

    Наличие файла swpuser.ini с указанием пользователя для рабочих процессов с соответствующими правами в ОС.

    При использовании файла swpuser.ini пользователь, от имени которого работает главный агент кластера (ragent), должен иметь административные права. Дополнительно следует обеспечить наличие этого пользователя в локальных политиках безопасности: Adjust memory quotas for a process (сюда обычно входит группа Администраторы) и Replace a process level token (обычно группа Администраторы сюда не входит). Необходимо помнить, что инсталлятор не подготавливает почву для использования swpuser.ini. Поэтому для использования swpuser.ini все предлагается сделать вручную: создать пользователей, дать им права и т. п. Важно, что пользователям, от имени которых запускаются рабочие процессы и менеджеры кластера, в Windows должен быть создан профиль. Он необходим для правильного размещения:

    - □ каталога временных файлов;
    - □ каталога данных приложения;
    - □ переменных окружения.

    Наиболее простой способ создания профиля – однократный интерактивный вход.

12. Наличие требований назначения функциональности для всех продукционных кластеров с двумя и более рабочими серверами:

    - □ для сервиса лицензирования;
    - □ для сервиса полнотекстового поиска данных;
    - □ для сервиса журнала регистрации.

    Должно быть явно указано расположение именно этих сервисов, так как их корректная работа зависит от расположения файлов кластера. При наличии нескольких рабочих серверов в кластере нужно не допускать возможности переезда этих сервисов между рабочими серверами.

13. Имена пользователей, от имени которых рабочие процессы rphost работают с СУБД, в точности соответствуют именам информационных баз.

    Лучше для каждой продукционной информационной базы создавать отдельного пользователя. Это нужно в первую очередь при расследовании каких-либо проблем на сервере СУБД. Если на сервере СУБД расположено более одной

продукционной базы данных, может возникнуть необходимость максимально быстро расследовать проблему. При этом под рукой в отчетах или трассировках всегда будет имя пользователя, который в данный момент работает с сервером баз данных. Как следствие, будет и понимание, к какой именно информационной базе относится этот пользователь.

14. Настроена утилита BgInfo от SysInternals.

    Желательно на рабочем столе всегда видеть:

    - имя сервера;
    - объем места на дисках;
    - его сетевые адреса.

15. План электропитания: Высокая производительность.

    В Control Panel\All Control Panel Items\Power Options необходимо указать High Performance.

16. Хотя бы один раз проведено тестовое восстановление из бэкапа с замером времени восстановления.

17. Настроены пересчет итогов, перестроение агрегатов.

18. Отсутствует лишняя регистрация изменений в планах обмена.

19. Число процессов rphost адекватно решаемой кластером задаче.

    Например, не установлена настройка 1 rphost на 1 информационную базу при наличии большого числа информационных баз в кластере серверов.

20. Корректно сконфигурированы виртуальные машины.

    На виртуальных машинах обеспечено гарантированное выделение ресурсов, суммарный объем выданной оперативной памяти в виртуальных машинах меньше объема имеющейся памяти на хосте.

ПРИЛОЖЕНИЕ 7

Примеры документов, формируемых в процессе тестирования прикладного решения

Протокол приемки конфигурации «Зарплата и управление персоналом» версии 3.1.2.278.6

Информация о релизе

Версия конфигурации	Зарплата и управление персоналом, 3.1.2.278.6 (доработанная от типовой 3.1.2.278)
Дата поступления	01.07.2017
Дата приемки	02.07.2017
Платформа 1С	8.3.10.2561

Check-лист приемки

№	Действие	Исполнитель	Результат	Комментарий
	Проверка версии конфигурации	Иванов И.И.	+	
	Проверка состояния поддержки конфигурации	Иванов И.И.	-	Конфигурация снята с поддержки
	Выполнение синтаксического контроля	Иванов И.И.	-	Выявил 1 ошибку в нетиповом функционале. См. ниже
	Проверка конфигурации на предмет удаленных объектов, реквизитов, табличных частей	Иванов И.И.	+	

Ошибки, выявленные при синтаксическом контроле

{Справочник.абвУниверсальныйНетиповойСправочник.Форма.Форма.Форма(555,12)}: Процедура или функция с указанным именем не определена (ВыполнитьМетодНаСервере)

Пока <<?>>ВыполнитьМетодНаСервере (ИмяМетода,ПолныеПрава) Цикл

Протокол тестирования конфигурации «Зарплата и управление персоналом» версии 3.1.2.278.6

Информация о релизе и подготовительном стенде

Версия конфигурации	Зарплата и управление персоналом, 3.1.2.278.6 (доработанная от типовой 3.1.2.278)
Дата поступления	01.07.2017
Даты проведения тестирования	02.07.2017 – 10.07.2017
Платформа 1С	8.3.10.2561
Состав оборудования	Кластер 1С: Srv-test-1c-01, Srv-test-1c-02, Srv-test-1c-03 Сервер СУБД: Srv-test-db-01

Тестирование ключевых операций

№	Действие	Исполнитель	Результат	Комментарий
	Вход в базу	Иванов И.И	+	
	Открытие формы списка справочника «Сотрудники»	Иванов И.И	+	
	Заведение нового «Физ. лица»	Иванов И.И	+	
	Расчет зарплаты за месяц	Иванов И.И	+	
	Создание и проведение документа «Прием на работу»	Иванов И.И	+	Сильно поменялась форма документа, может потребоваться дополнительное описание для пользователей

№	Действие	Исполнитель	Результат	Комментарий
	Создание и проведение документа «Кадровый перевод»	Иванов И.И	+	
	Создание и проведение документа «Увольнение»	Иванов И.И	+	
	Формирование табеля за месяц	Иванов И.И	+	
	Формирование расчетной ведомости	Иванов И.И	+	

Тестирование нетиповых доработок в релизе

№	Тестируемое изменение	Исполнитель	Результат	Комментарий
	Изменение в форме 4-ФСС в соответствии с постановлением правительства № 1 от 01.01.2017	Петров П.П.	+	
	Групповое заполнение семейного положения по списку сотрудников	Петров П.П.	+	
	Функционал отложенного проведения кадровых документов	Петров П.П.	+	

Заключение

В данном учебном пособии изложены ключевые моменты, связанные с настройкой программного обеспечения для эффективной работы системы на базе платформы «1С:Предприятие», описаны базовые методики для расследования проблем доступности и производительности, рассмотрены вопросы организации эксплуатации.

Пособие предназначено для слушателей курса «1С:Эксплуатация крупных информационных систем», а также для специалистов, занимающихся эксплуатацией информационных систем, построенных с использованием технологической платформы «1С:Предприятие». Рекомендуется в качестве методического материала при подготовке к аттестации «1С:Эксплуататор крупных информационных систем».

Данный материал служит существенным методическим дополнением к материалам, изложенным в документации на ИТС, и иным на тему эксплуатации. Пособие призвано систематизировать и упорядочить уже имеющиеся у читателя практические навыки по эксплуатации, предоставить в их распоряжение методики, отработанные специалистами фирмы «1С» на крупных корпоративных внедрениях. Мы предполагаем, что пособие будет использоваться в ежедневной работе эксплуататоров и администраторов информационных систем. Рассчитываем, что часть материала будет использоваться как check-листы, отступать от которых имеет смысл только при понимании, «почему нам это не подходит».

Предложенные примеры команд, запросов и скриптов имеет смысл попробовать выполнить, научиться пользоваться ими. Данные примеры не раз экономили время при расследовании технологических проблем.

Мы рассчитываем, что специалисты в области эксплуатации, применяя полученные знания, смогут повысить свою эффективность в вопросах обеспечения технологического качества крупных информационных систем. Полученный набор инструментов и знаний поможет специалистам оперативно собирать необходимую информацию о возникающих проблемах для их оперативного устранения.

Надеемся, что применение полученных знаний и рассмотренных подходов к эксплуатации систем повысит их технологическое качество.

Фирма «1С»
123056, Москва, а/я 64, Селезневская ул., 21.
Тел.: (495) 737-92-57, факс: (495) 681-44-07.
1c@1c.ru, http://www.1c.ru/

Издательство ООО «1С-Паблишинг»
127434, Москва, Дмитровское ш., д. 9.
Тел.: (495) 681-02-21, факс: (495) 681-44-07.
publishing@1c.ru, http://books.1c.ru

Об опечатках просьба сообщать по адресу publishing@1c.ru.