ARENADATA

Корпоративная платформа хранения
и обработки больших данных

Arenadata DB для Разработчиков

Часть 1

ADB – аналитическая СУБД для больших данных

Введение в ADB:
Назначение ADB. Отличия от других систем. Отличия от GP.

Обзор архитектуры ADB:
Концепция MPP и её имплементация в ADB. Терминология и архитектура СУБД.

Окружение СУБД:
Кластер-менеджер ADCM. Мониторинг. Enterprise Tools.

Подключение к БД:
Реквизиты. Клиент командной строки. Графические клиенты.

Устройство БД:
Шаблоны БД. Логическая схема данных. Обзор дефолтных схем. Основные объекты для работы с БД.

Ролевая модель:
Роль и пользователь, группы. Доступы. Создание пользователей, управление.

Ресурсные группы:
Параметры и ограничения. Создание ресурсных групп.

Дисковая квота:
Информация о модуле. Настройка. Нюансы использования.

ARENADATA

Корпоративная платформа хранения
и обработки больших данных

Введение в ADB

ADB – аналитическая СУБД для больших данных

Основные сведения о системе:

- Arenadata DB (ADB) — SQL-совместимая реляционная аналитическая СУБД, построенная на основе MPP-системы с открытым исходным кодом Greenplum.
- Основное назначение – выполнение сложных запросов аналитического профиля (OLAP) с большими объёмами структурированных данных.
- В качестве инфраструктуры для размещения ADB используются кластеры, которые могут включать до сотен серверов.

ADB – GP с дополнительной функциональностью

Arenadata DB имеет ряд преимуществ перед ванильным Greenplum:

- Arenadata Cluster Manager — инструмент, значительно упрощающий установку и настройку СУБД, в том числе в закрытых ЦОДах без доступа в интернет.

- Нативная интеграция с Arenadata Hadoop, QuickMarts (ClickHouse), Streaming (Kafka&NiFi) в рамках единой платформы Arenadata EDP и другие дополнительные возможности интеграции.

- Система мониторинга и оповещения (Graphite и Grafana).

- Arenadata Command Center — инструмент для работы с запросами. Он дает рекомендации по улучшению запросов и прогнозы времени работы.

- Поддержка x86 и Power-совместимого оборудования (8, 9, OpenPower).

Ванильный GP и ADB Community/Enterprise Edition

Функционал	Open Source дистрибутив	ADB CE	ADB EE
• Core-функционал • PXF • gpbackup • Коннекторы Greenplum <-> Hadoop и Greenplum <-> JDBC-источники	+	+	+
• Коннекторы Greenplum <-> Kafka и Greenplum -> ClickHouse	-	-	+
• Command Center (мониторинг на уровне запросов) • оффлайн установка	-	-	+
• Управление деплоем и апгрейдом, • Расширение кластера, • Мониторинг & alerting	-	+	+
• Client/Loader утилиты	-	-	Redhat 8
• Документация	English only	+	+
• Поддержка • Обучение по продуктам	-	-	+
• Операционная система	Ubuntu 18.04/ Redhat 7 / Redhat 6	CentOS 7/ Redhat 7	CentOS 7 / Redhat 7/ Альт 8 СП Сервер
• Доп.консалтинговые услуги (DBAaS, Smart Start, TAM, Аудит)	-	-	+

ARENADATA

Системы для больших данных: сравнение

Hadoop (ADH)

Роль:
- Подходит для хранения и обработки сырых и агрегированных данных.
- Служит как хранилище для холодных данных.

Плюсы:
- Анализ данных с помощью популярных фреймворков, например Spark, PySpark.

Минусы:
- Трудоёмко писать задачи MapReduce.
- Медленнее, чем GP и ClickHouse.

Greenplum (ADB)

Роль:
- Подходит для КХД, так как может хранить и обрабатывать данные из разных слоёв.

Плюсы:
- Настоящая MPP-система.
- Совместим с ANSI SQL 2008.
- Postgres-совместим. Могут быть подключены типовые BI и ETL системы.
- Транзакционный.
- Локальные и распределенные джойны.
- Хорошо справляется с ELT (ETL с интеграцией слоёв данных для КХД).

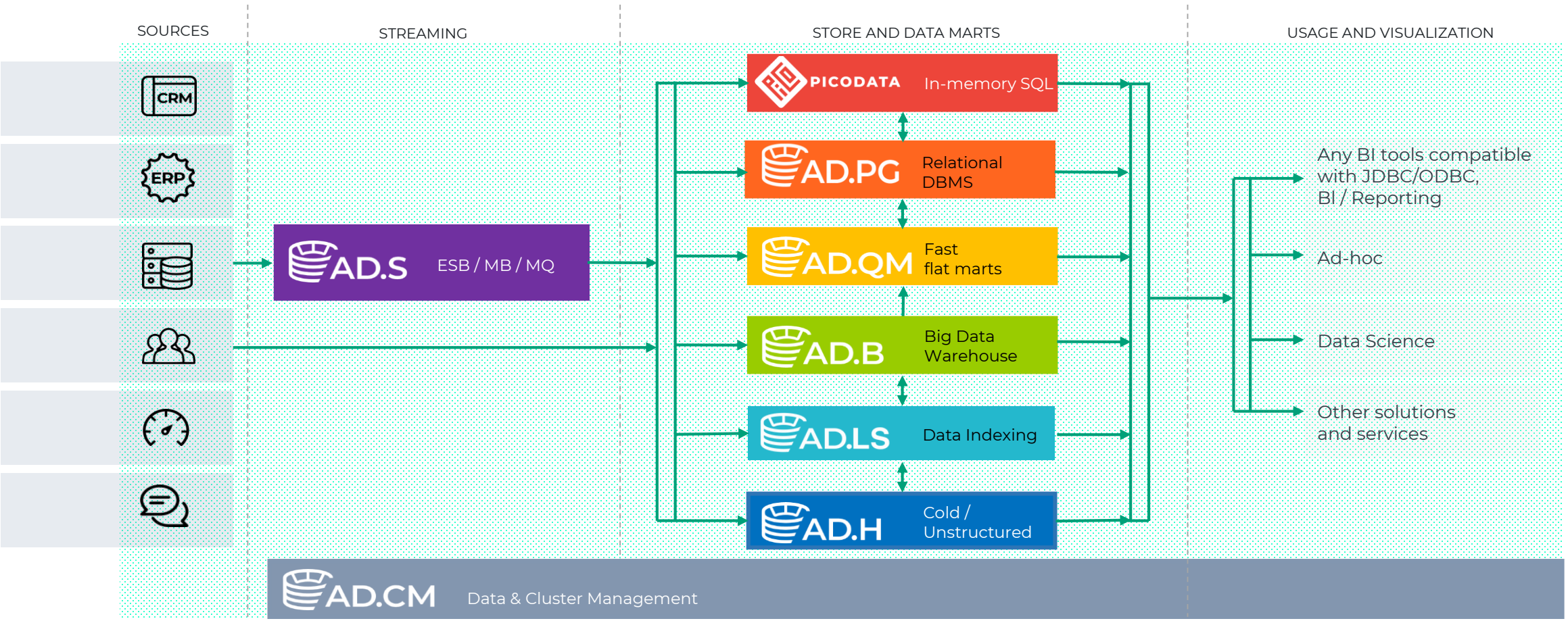
ClickHouse (ADQM)

Роль:
- Выдача готовых данных большому числу пользователей. Подходит для быстрых запросов к широким таблица фактов и к витринам.

Плюсы:
- Очень быстро работает с where-условиями и строит агрегаты очень быстро.
- Позволяет гибко задавать логические схемы шардирования и зеркалирования.

Минусы:
- Сложности с джойнами.
- Нет транзакций.
- Несколько сложен в настройке и эксплуатации.

Arenadata Enterprise Data Platform



ARENADATA

ARENADATA

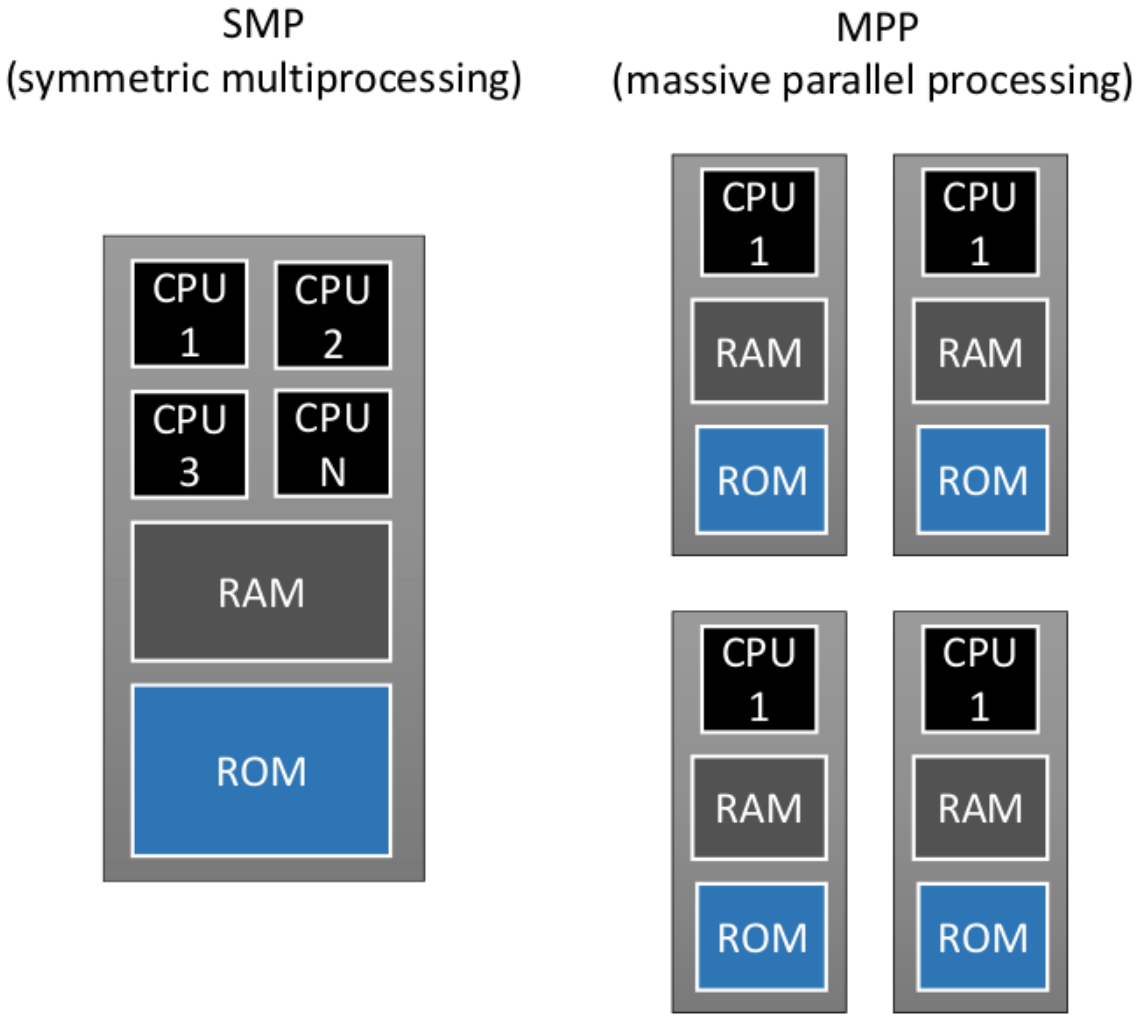
Корпоративная платформа хранения
и обработки больших данных

Архитектура кластера ADB

Обзор архитектуры: SMP и MPP

Преимущества MPP-систем:

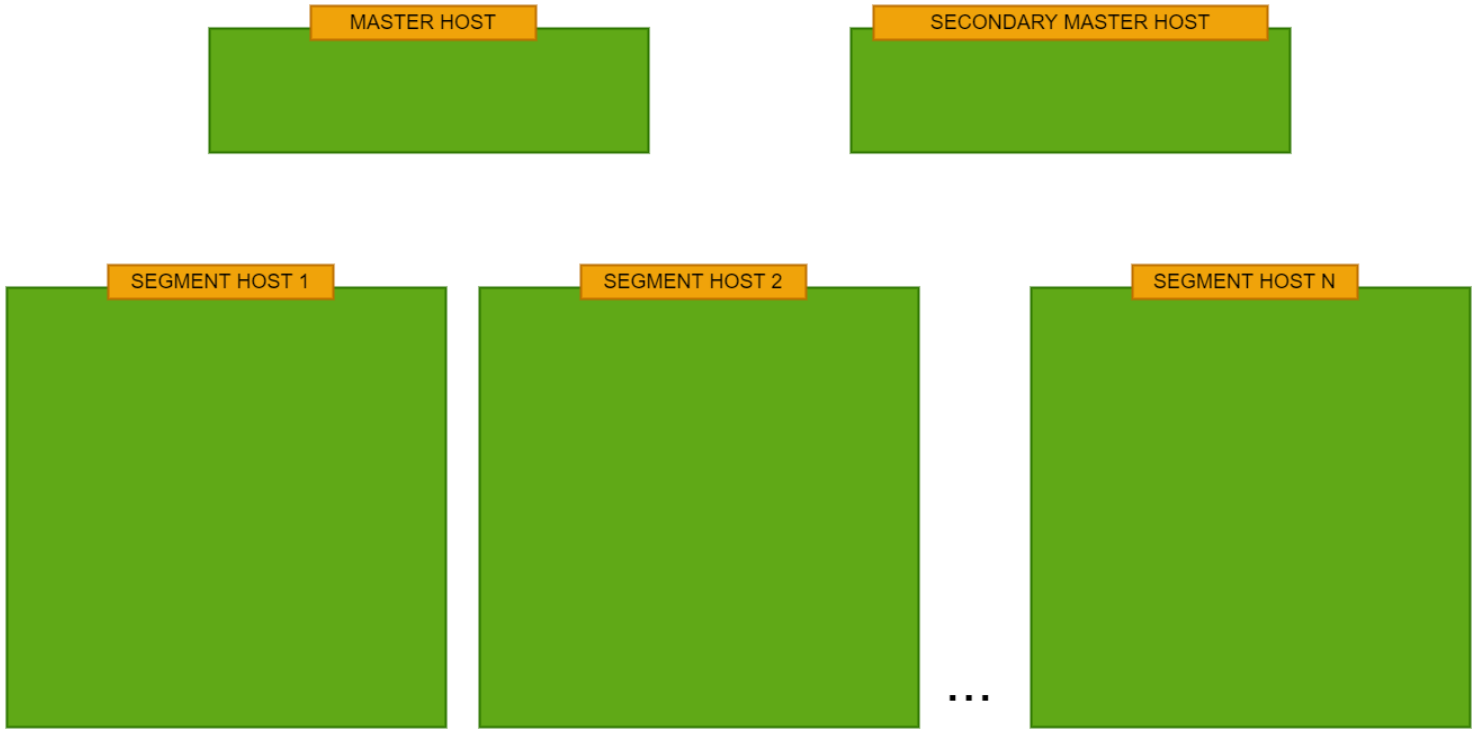
- **Горизонтальная масштабируемость.**
- **Линейный рост производительности.**
- **Большие возможности расширения.**
- **Отказоустойчивость.**



Структура кластера ADB: серверы

Кластер ADB разворачивается на наборе машин, состоящем из:

- Одного мастер-сервера - *master host* и (опционально) одного запасного мастер-сервера - *secondary master*.
- Не менее двух одинаковых сегментных серверов - *segment host*.

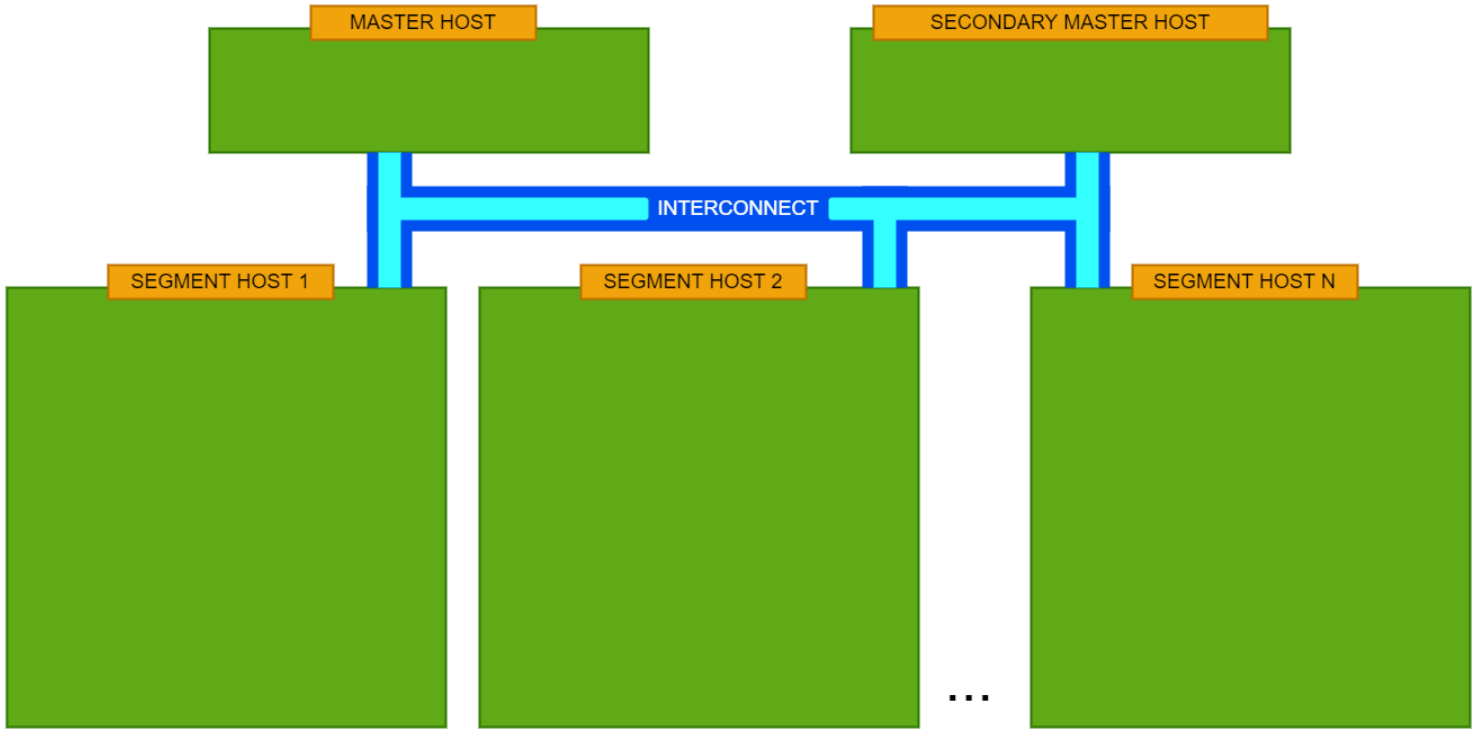


- Сегментные серверы должны быть одинаковыми по характеристикам.
- Мастер-сервер обычно делают вдвое слабее сегментного.
- При выборе из малого числа мощных машин и большого числа, но слабее, обычно лучше второй вариант.
- Количество сегментных серверов можно увеличить (расширить кластер), но нельзя уменьшить.

Структура кластера ADB: интерконнект

Машины соединяются между собой сетью-интерконнектом:

- Не менее 10 Гигабит. Как правило, в типовых конфигурациях используют 2x40 или 1x100.
- По умолчанию ADB использует UDP (своя реализация TCP over UDP), можно переключить на обычный TCP.

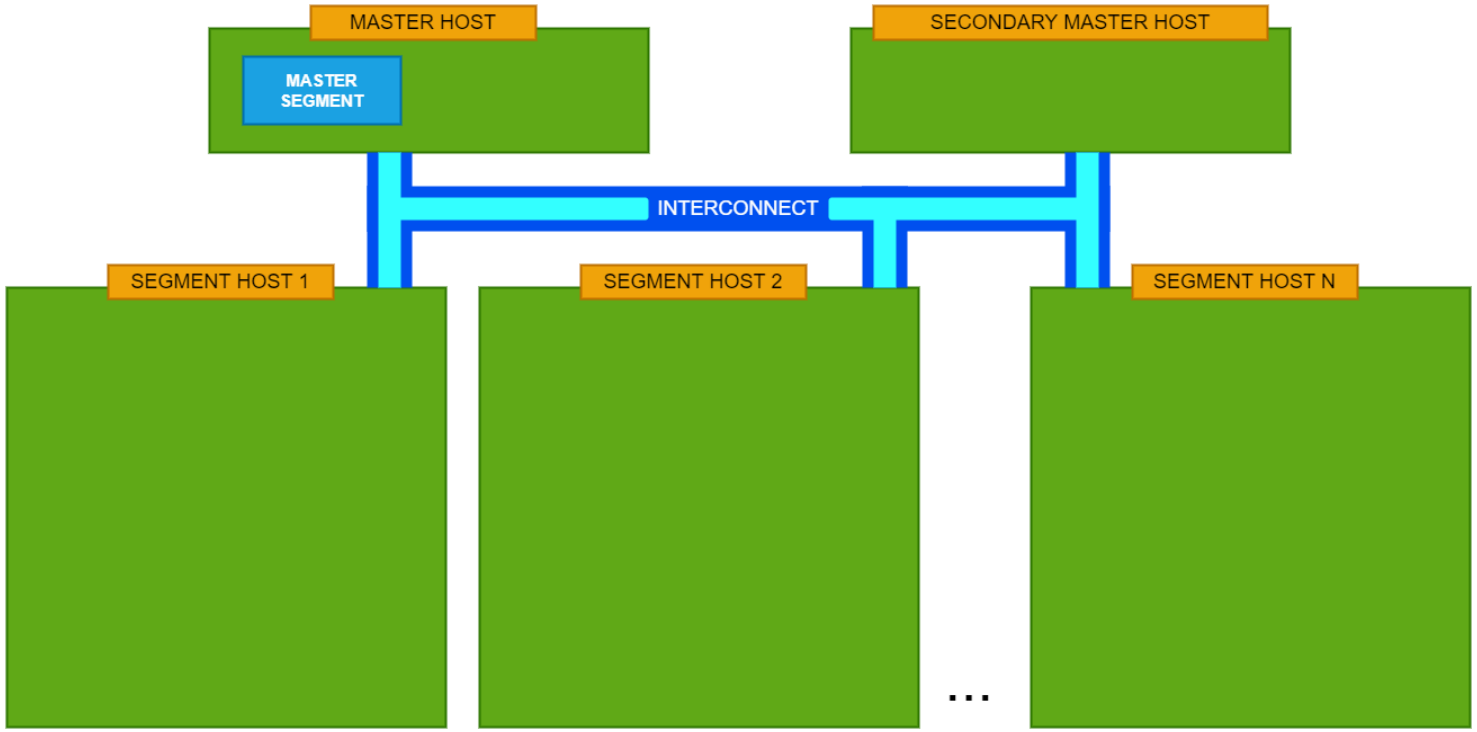


ARENADATA

Структура кластера ADB: сегменты

Мастер-сервер и мастер-сегмент:

- Мастер-сервер является единой точкой входа для работы с СУБД.
- На мастер-сервере размещается одна копия СУБД Postgres, именуемая *мастер-сегментом*. Слушает порт 5432.

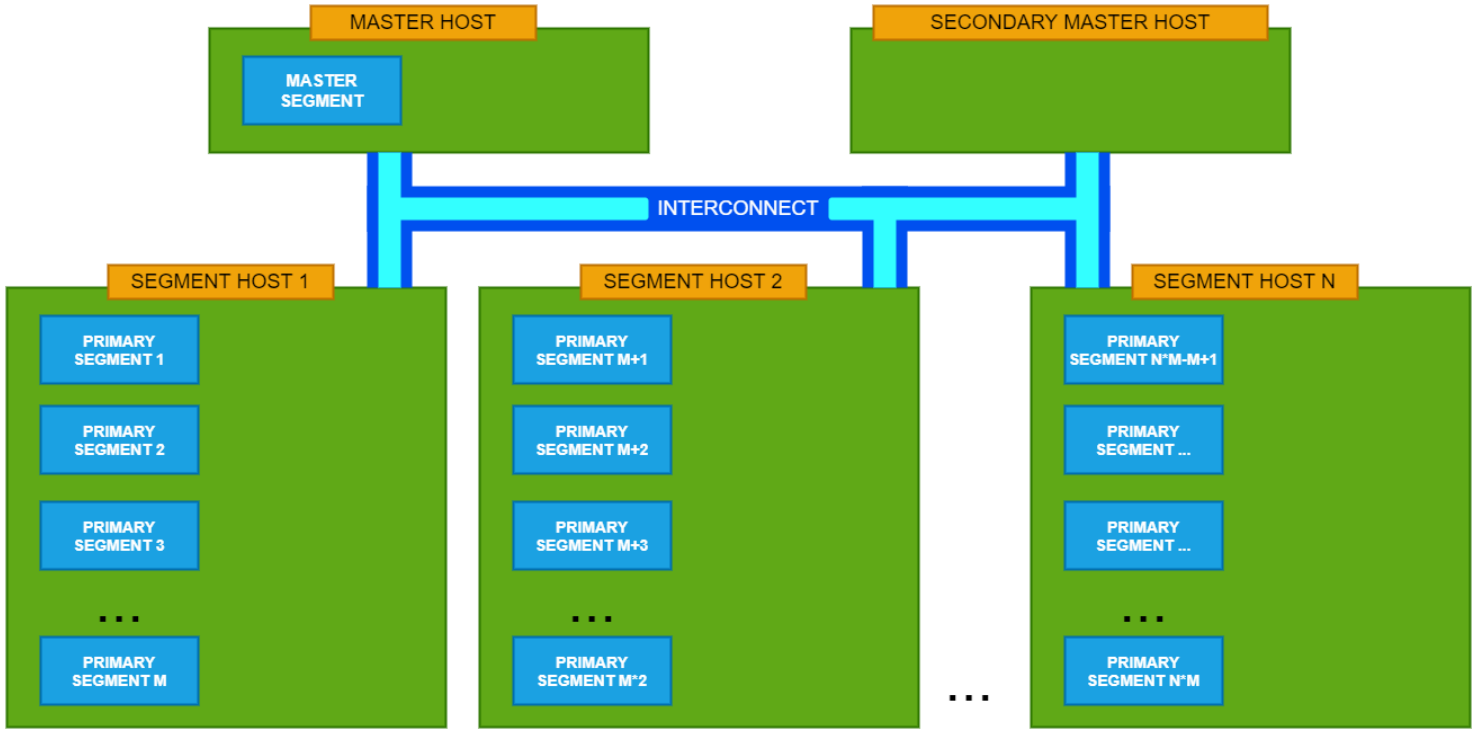


- Мастер-сегмент не хранит содержимого пользовательских объектов.
- На мастере хранятся данные системного каталога.
- Мастер *может* выполнять некоторые финальные операции с данными – агрегации, сортировки и т.д..
- Большинство административных действий выполняются только на мастере.

Структура кластера ADB: сегменты

Праймари-сегменты на сегментных хостах:

- На каждом сегментном сервере присутствует одинаковое число копий СУБД Postgres - праймари-сегментов. Типовая конфигурация – 1 сегмент на каждые 2 ядра процессора сегментного сервера.
- Каждый сегмент хранит свою отдельную часть от общего набора данных.



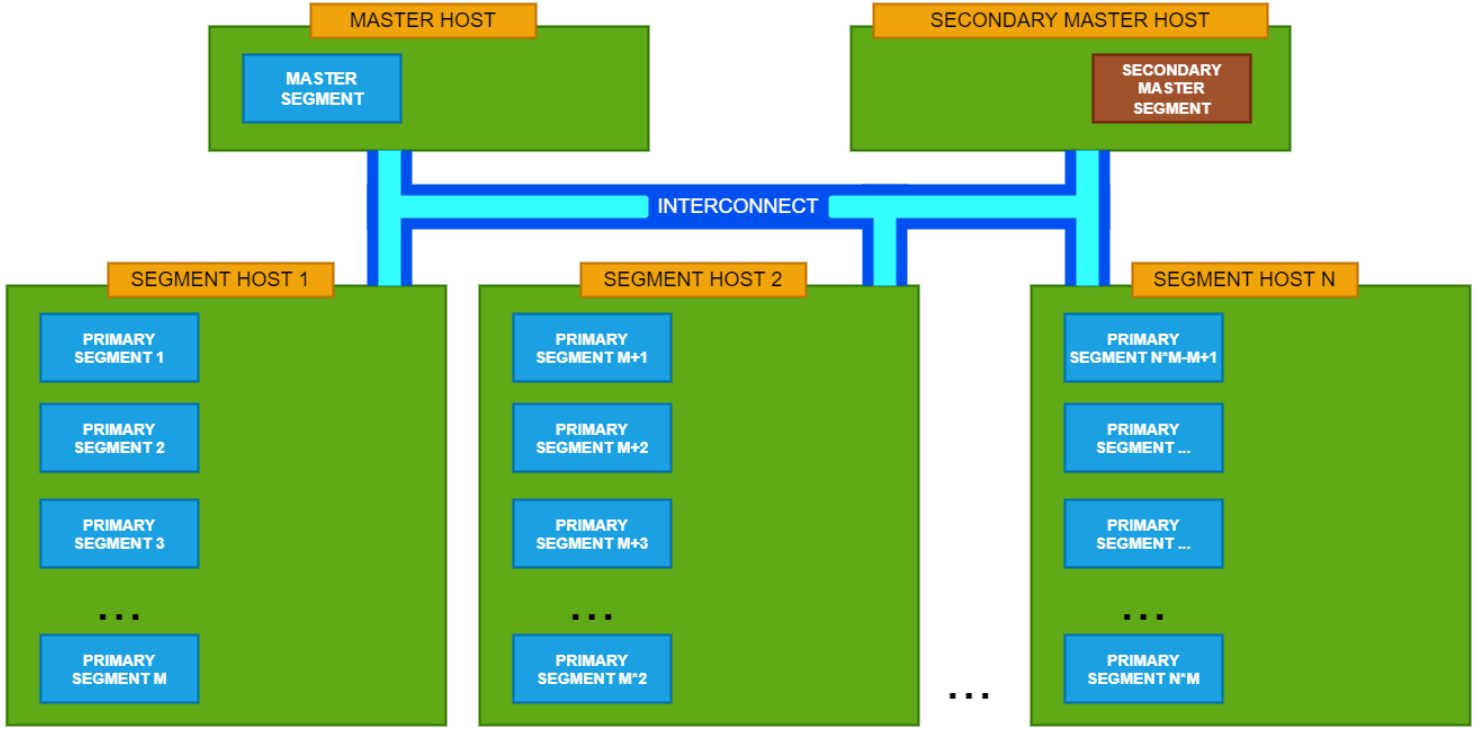
- Любая таблица в БД распределяет свои данные сразу по всем праймари-сегментам.
- Каждый праймари-сегмент имеет копию системного каталога с мастера.
- Большинство действий, связанных с обработкой данных, выполняется локально на сегментах.
- Сегменты обмениваются данными между собой по интерконнекту во время выполнения запроса.

ARENADATA

Структура кластера ADB: сегменты

Вторичный мастер и его сегмент:

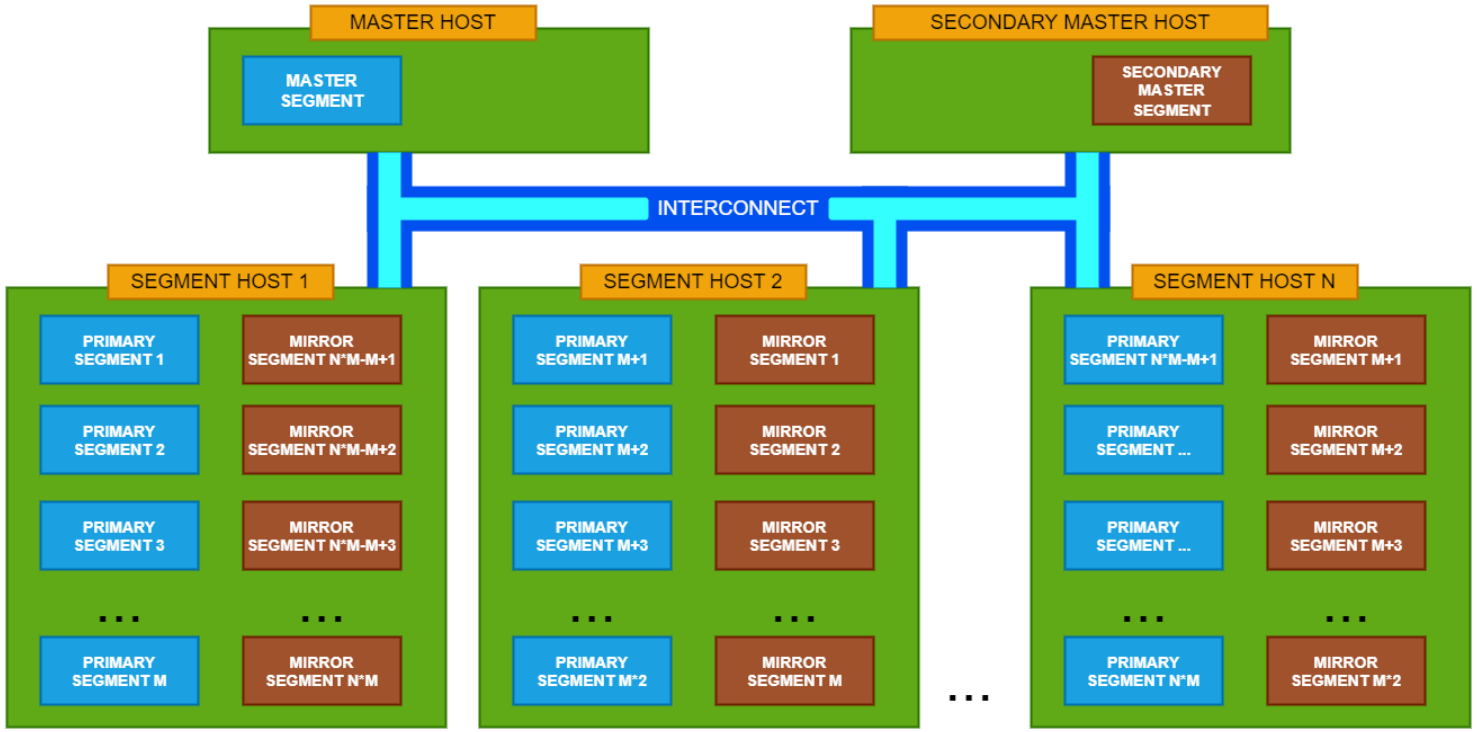
- На запасной машине размещается одна копия Postgres, которая является зеркальной копией мастер-сегмента.
- Вторичный мастер-сегмент не выполняет никаких действий, кроме резервного копирования данных мастер-сегмента. При сбое на мастер-сервере администратор вручную назначает вторичному серверу роль мастера.



Структура кластера ADB: сегменты

Зеркала праймари-сегментов:

- У каждого праймари-сегмента по умолчанию есть одно (больше нельзя) зеркало на другом сервере кластера.
- Зеркала служат только для резервного копирования данных, включаясь в основную работу только при сбое в праймари-сегменте. Переключение на зеркало происходит автоматически, обратное – вручную.



ARENADATA

Структура кластера ADB: отказоустойчивость

Отказоустойчивость мастера:

- Warm standby (реплика каталога).
- Синхронная репликация на основе WAL.
- Автоматического переключения при сбое нет, необходимо сделать это вручную.

Отказоустойчивость сегментов:

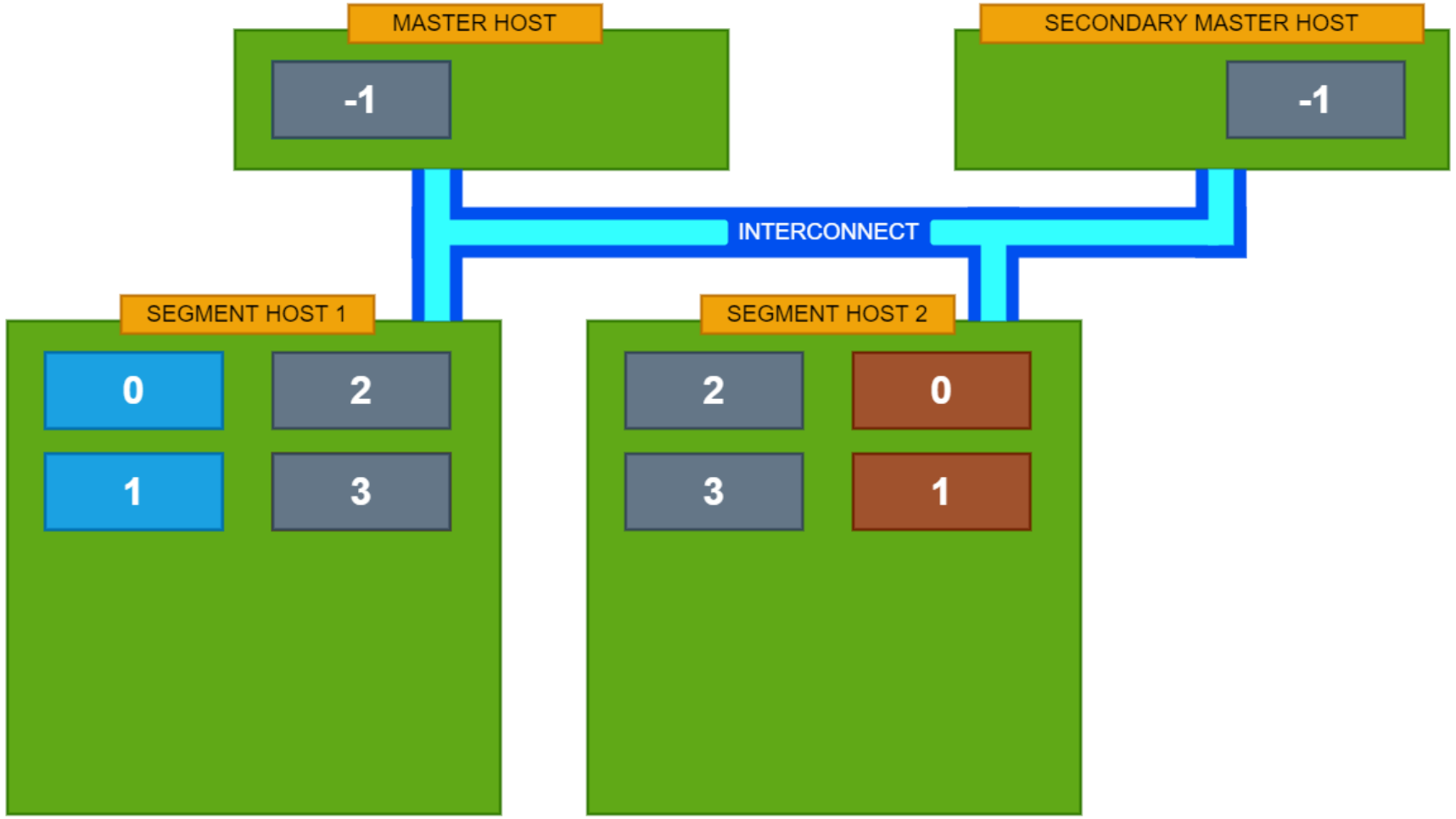
- Для каждого основного сегмента создаётся зеркало.
- Синхронная репликация на основе WAL.
- Автоматическое переключение при сбое.
- Для обратного переключения нужно выполнить процедуру восстановления.
- Есть два варианта расположения зеркал. Один из них выбирается на этапе установки СУБД.

Политики размещения зеркал: Group mirroring

В ADB зеркала могут размещаться двумя отдельными способами. Первый способ – это размещение группами (Group mirroring):

- Все зеркала для основных сегментов, находящихся на одном физическом сервере, вместе размещаются на каком-то другом.
- Выходит, что один сервер целиком страхует другой и возьмет на себя всю его нагрузку в случае выхода того из строя.
- При выходе из строя одного сегментного сервера скорость работы кластера сразу замедляется вдвое.
- Конфигурация возможна при любом соотношении количества сегментных серверов и количества сегментов на одном сервере, но в ADB при возможности автоматически выбирается другой вариант размещения зеркал.

Политики размещения зеркал: Group mirroring



Политики размещения зеркал: Spread mirroring

Второй вариант – это распределенное размещение (Spread mirroring):

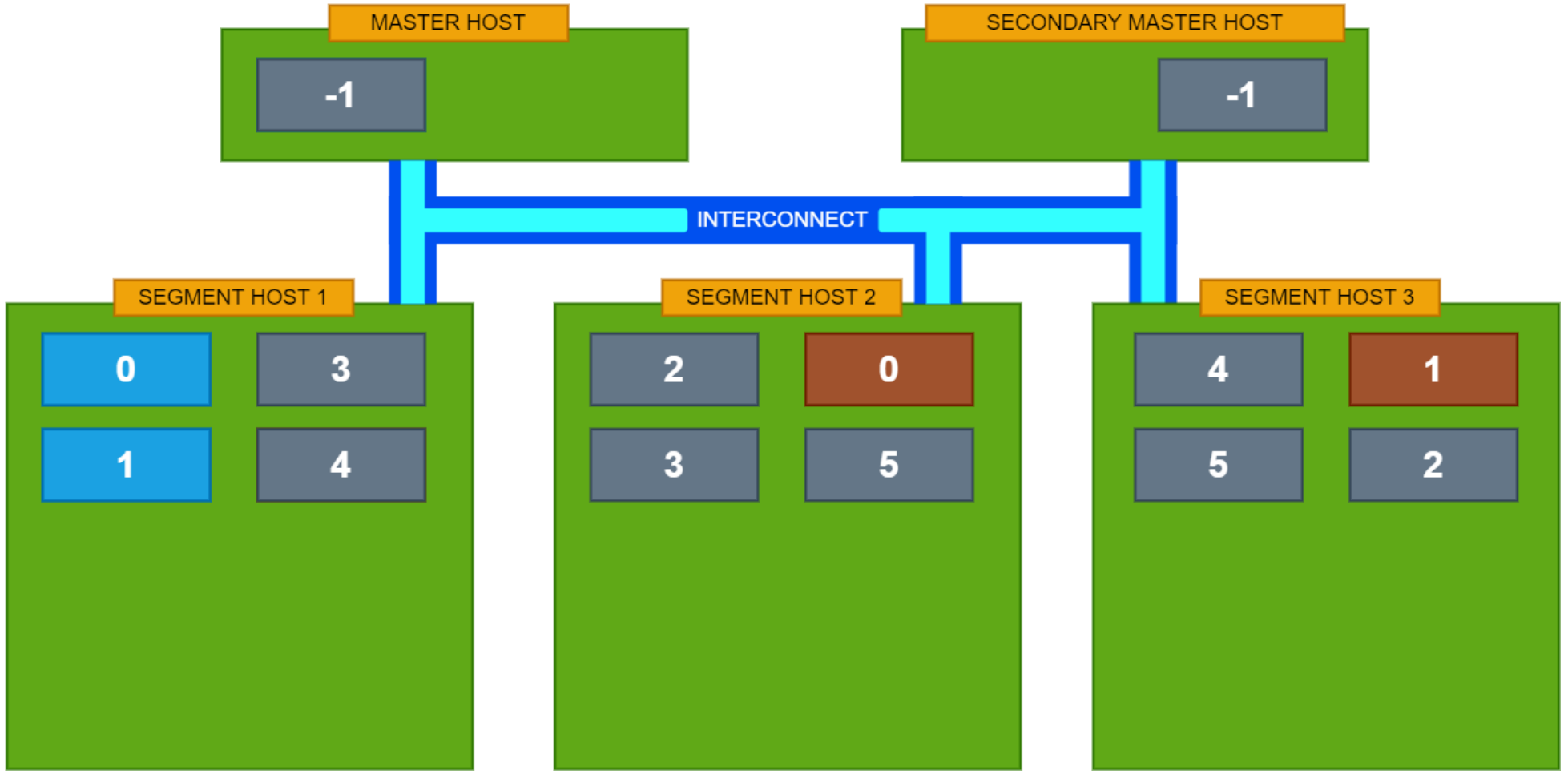
- Зеркала праймари-сегментов с одного сегментного сервера попадают на разные сегментные серверы (по одному на сервер). Для этого число сегментных серверов должно как минимум на единицу превышать число сегментов на сервере.

- При выходе из строя одного сегментного сервера скорость работы кластера падает не столь значительно, чем при Group mirroring, так как возросшая нагрузка распределяется между набором серверов.

- Уязвимость к повторному сбою при Spread mirroring выше. В случае с Group Miroring при выходе из строя сервера к отказу системы может привести поломка одной машины - на которой лежат зеркала уже упавшей. Падение других условно безопасно.

- При Spread mirroring система откажет при падении любой из машин, на которые были распределены зеркала.

Политики размещения зеркал: Spread mirroring



Структура кластера ADB: резюме

- Кластер в норме состоит из управляющей машины (мастер-сервер) и набора одинаковых по характеристикам серверов для хранения и обработки данных (сегментные хосты).
- У мастер-сервера может быть резервная замена – вторичный мастер-сервер. Все машины соединены интерконнектом с пропускной способностью не менее 10 гигабит.
- На машинах размещены отдельные экземпляры СУБД Postgres, именуемые сегментами.
- На мастер-сервере присутствует только один сегмент. На сегментных серверах - один или более основных (праймари), а также зеркала сегментов с других серверов.
- Конфигурирование СУБД и подключение пользователей к базам выполняется на мастер-сервере. Основную работу с данными базы выполняется на праймари-сегментах сегментных серверов, мастер агрегирует их результаты и отдает пользователю результат выполнения запроса.
- Сегменты бывают основные и зеркальные. Зеркальные сегменты не отдают данные по запросу, на них выполняется только запись.
- Есть две политики зеркалирования: групповая и распределенная. Политика определяет то, как размещаются зеркала основных сегментов на сегментных серверах. Зеркало мастер-сегмента всегда находится на отдельном сервере.
- Переключение на зеркало мастера при сбое производится вручную. Переключение на зеркало при сбое рабочего сегмента происходит автоматически.

ARENADATA

Корпоративная платформа хранения
и обработки больших данных

Кластер-менеджер ADCM

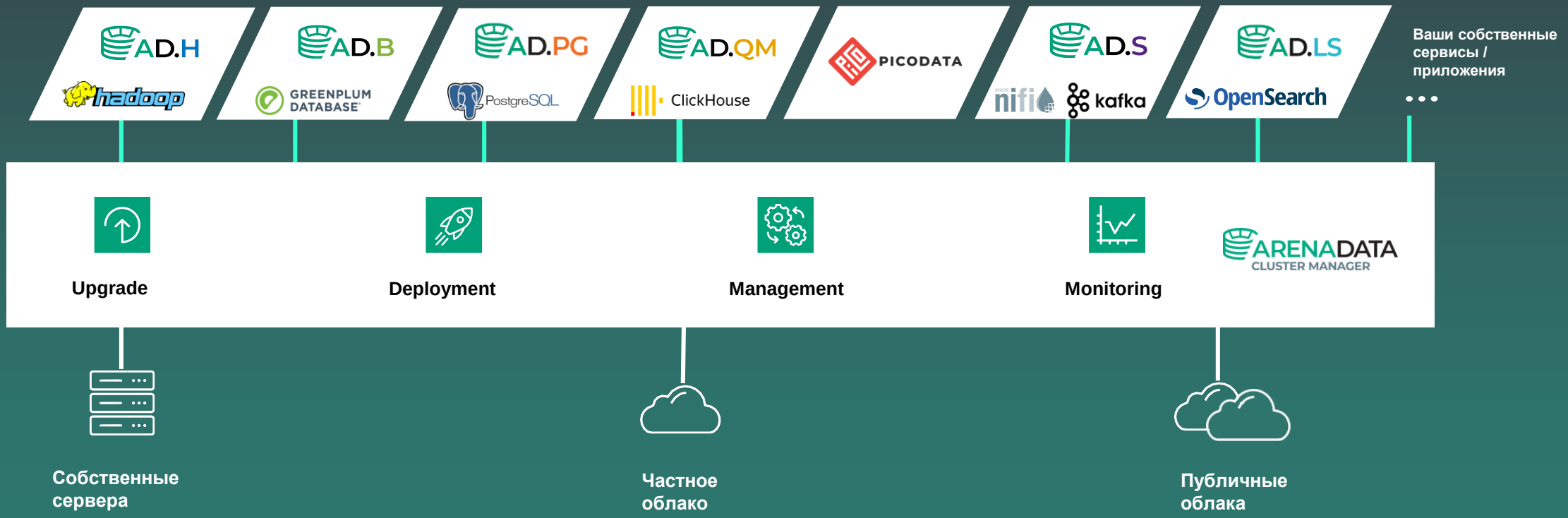
ADCM – система развертывания и управления

- Автоматизацию процессов обновления и управления обеспечивает Arenadata Cluster Manager (ADCM) — отдельный разработанный продукт с открытым исходным кодом.

- ADCM – это универсальный оркестратор гибридного ИТ-ландшафта, благодаря которому в автоматическом режиме происходят установка, настройка и обновление кластеров, настройки операционной системы, сервисов, сети и монтирование дисков.

- С помощью ADCM можно разворачивать data-сервисы как в облаке, так и on-premise, а также она может служить в качестве PaaS-сервисов.

ADCM – система развертывания и управления



ARENADATA

ADCM – возможности

Уровень Data Service:

- Установка.
- Обновление.
- Управление.
- Мониторинг.
- Управление доступом.
- Интеграция между Data Service.

Уровень Infrastructure:

- Создание и удаление виртуальных машин.
- Настройка OS.
- Мониторинг.
- Управление пользователями.
- Управление доступом.

Присутствует Rest API, библиотеки для Python, модули и плагины Ansible для автоматизации.

Возможно создавать свои бандлы для расширения функциональности.

ADCM – бандлы

- ADCM bundle – это специальный архивный файл, который содержит файлы конфигураций сервисов и Ansible-скрипты по управлению ими.

- Существует два типа бандлов: Cluster Bundle и Infrastructure Bundle.

- Cluster Bundle необходим для развертывания кластеров систем, в то время как Infrastructure Bundle позволяет добавить поддержку различных хост-провайдеров (таких как Google Cloud, Yandex Cloud или обычный ssh-хост).

Для ADB Community понадобятся: инфраструктурный бандл хоста (SSH в нашем случае), бандл ADB и бандл мониторинга.

Для ADB Enterprise потребуется дополнительно бандл Enterprise Tools.

ADCM как прикладное ПО

- Устанавливается в виде Docker-контейнера. Резервное копирование при этом – бекап файловой системы.

- Достаточно одной инсталляции ADCM для всех дата-сервисов. Не следует размещать ADCM на сервере, который будет входить в состав системы, предоставляющей дата-сервис. При необходимости можно совмещать с системой мониторинга.

- Поддерживаемые ОС: CentOS 7/RHEL 7 (также возможна установка на Alt Linux 8.2).

- Для установки необходим доступ к репозиториям ОС (для CentOS - CentOS Extras и CentOS Base).

- Перед установкой необходимо отключить SELinux.

- Пакеты, необходимые для установки:
  - yum-utils.
  - docker.
  - device-mapper-persistent-data.
  - lvm2.

- После установки консоль ADCM доступна по порту 8000.

Установка ADCM

На машине, куда ставится ADCM, следует выполнить последовательно команды:

1. sudo sed -i 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/selinux/config
2. sudo setenforce 0
3. sudo yum install -y yum-utils docker device-mapper-persistent-data lvm2
4. sudo systemctl enable docker --now
5. sudo docker pull arenadata/adcm:latest
6. sudo docker create --name adcm --restart unless-stopped -p 8000:8000 -v /opt/adcm:/adcm/data arenadata/adcm:latest
7. sudo docker start adcm

Интерфейс ADCM будет доступен в браузере по адресу http://<адрес машины>:8000 (ip-адрес машины adcm, порт 8000) с логином и паролем admin.

Обновление ADCM

На машине, где развернут ADCM, следует выполнить последовательно команды:

1. sudo docker stop adcm
2. sudo docker rm adcm
3. sudo docker pull arenadata/adcm:latest
4. sudo docker create --name adcm --restart unless-stopped -p 8000:8000 -v /opt/adcm:/adcm/data arenadata/adcm:latest
5. sudo docker start adcm

Интерфейс обновленной версии ADCM будет доступен в браузере по адресу http://<адрес машины>:8000 (ip-адрес машины adcm, порт 8000) с логином и паролем admin.

Лабораторная работа. Обновление ADCM

1. Откройте веб-интерфейс своего сервера ADCM и проверьте версию.
2. Подключитесь по SSH к этому серверу.
3. Выполните обновление.
4. Откройте веб-интерфейс и проверьте версию системы.

Логин и пароль для SSH-подключения и входа в веб-интерфейс одинаковые.

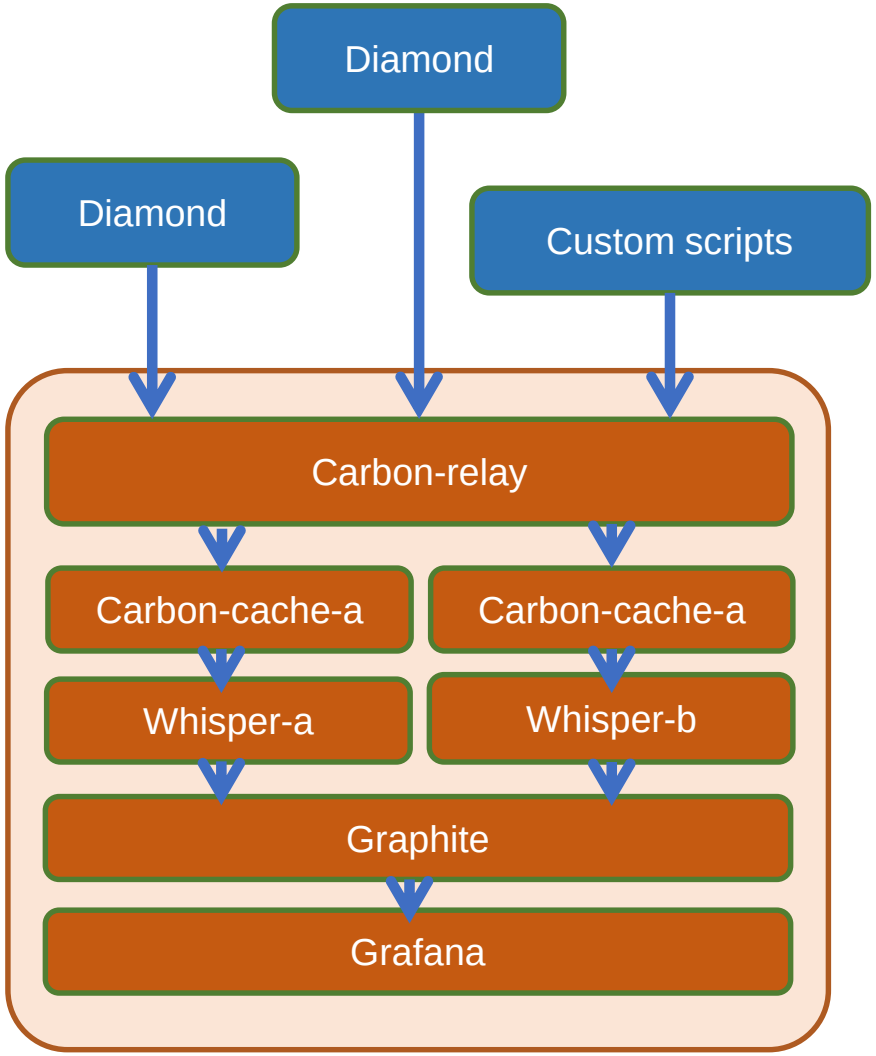
ARENADATA

Корпоративная платформа хранения
и обработки больших данных

Мониторинг в ADCM

Мониторинг

- Сервис мониторинга модульный, состоит из нескольких компонент:
  - Diamond – собирает и отправляет системные метрики с серверов.
  - Carbon-relay – принимает метрики, преобразует в бинарный формат и отправляет Carbon-caches.
  - Carbon-caches – принимают метрики и записывают их в Whisper DB.
  - Whisper DB – файловая временная БД.
  - Graphite – фронтенд, читающий метрики из Whisper DB и отдающий их в ответ на особые запросы.
  - Grafana – визуализатор метрик Graphite и алерты.
- Модульность системы позволяет выполнять масштабирование при росте числа метрик.



Мониторинг: Diamond

- Сервис: diamond.
- Работает под root, так как обращается к низкоуровневым метрикам.
- Установлен на всех серверах кластера.
- /var/log/diamond/archive.log – архив всех отосланных метрик.
- /var/log/diamond/diamond.log – ошибки и другие сообщения.
- /etc/diamond/diamond.conf – настройка адреса отсылки, периодичности и т.д..
- Diamond имеет модульную архитектуру, расширяется благодаря коллекторам:

```
/etc/diamond/collectors/
├── CPUCollector.conf
├── DiskSpaceCollector.conf
├── DiskUsageCollector.conf
├── LoadAverageCollector.conf
├── MemoryCollector.conf
├── NetworkCollector.conf
└── SmartCollector.conf
```

Мониторинг: Carbon

Сервисы: carbon-relay, carbon-cache-a, carbon-cache-b

Размещаются на сервере мониторинга внутри docker-контейнера graphite.

Логи:

- /var/log/carbon-relay.log - логи сервиса carbon-relay (директория внутри контейнера).
- /var/log/carbon.log - логи сервиса carbon-cache (директория внутри контейнера).

Конфигурационный файл:

- /etc/graphite/carbon.conf – секция relay, cache:a и cache:b. Размещается на сервере мониторинга.

Мониторинг: Wishper

- Просто директории в файловой системе сервера мониторинга:

```
/opt/monitoring/whisper/
├── Arenadata
│   └── DB
│       ├── database
│       │   ├── available
│       │   │   └── is_available.wsp
│       │   ├── db_datfrozenxid
│       │   │   ├── adb
│       │   │   │   ├── seg_2.wsp
```

- Файлы с метриками имеют расширение .wsp
- Точки в названии метрики преобразуются в структуру каталогов:

  Database.hardware.iops.sdw1 => Database/hardware/iops/sdw1.wsp

- Удаление файлов = удаление метрик.

Мониторинг: Graphite

- Устанавливается в виде docker-контейнера graphite.
- Конфигурационные файлы вынесены наружу в файловую систему сервера: /etc/graphite/.
- Приложение HTTP, сервис httpd.
- Интерфейс доступен на порту 80.
- Логин и пароль – root.
- /opt/graphite/webapp/graphite/local_settings.py:

```
DATA_DIRS = ['/opt/graphite/whisper_a','/opt/graphite/whisper_b'].
```

- Можно использовать как источник данных для Zabbix.

Мониторинг: Grafana

- Устанавливается в виде docker-контейнера grafana.
- Конфигурационные файлы вынесены наружу в файловую систему сервера: /var/lib/docker/volumes/
  - grafana_data.
  - grafana_etc.
  - grafana_log.
- Интерфейс доступен на порту 3000.
- Логин и пароль задаются при установке.

Установка мониторинга (community)

- Откройте в браузере интерфейс ADCM (ip машины adcm, порт 8000) и войдите в систему.
- Создайте хост с названием monitoring и установите на него statuschecker.
- В настройках хоста укажите логин, пароль (или ключ) и IP-адрес.
- Загрузите бандл мониторинга.
- Создайте кластер с названием Monitoring Cluster на основе бандла мониторинга.
- Добавьте в кластер сервисы Graphite и Grafana.
- Укажите пароль в настройках сервиса Grafana.
- Добавьте хост monitoring в кластер.
- Установите добавленный хост как место размещения обоих сервисов.
- Запустите процесс установки кластера и проконтролируйте его до успешного завершения.
- Откройте в браузере интерфейс Grafana (ip машины, порт 3000) и войдите в систему с логином и паролем, заданными при установке.

Установка мониторинга (enterprise)

- Откройте в браузере интерфейс ADCM (ip машины adcm, порт 8000) и войдите в систему.
- **У вас должен быть создан кластер Enterprise Tools на основе соответствующего бандла.**
- Создайте хост с названием monitoring. В настройках хоста укажите логин, пароль (или ключ) и IP-адрес.
- Установите на него statuschecker.
- **Добавьте в кластере Enterprise Tools сервисы Graphite и Grafana.**
- Укажите пароль в настройках сервиса Grafana.
- Добавьте хост monitoring в кластер.
- Установите добавленный хост как место размещения обоих сервисов.
- Запустите процесс установки мониторинга в кластере Enterprise Tools и проконтролируйте его до успешного завершения.
- Откройте в браузере интерфейс Grafana (ip машины, порт 3000) и войдите в систему с логином и паролем, заданными при установке.

ARENADATA

Корпоративная платформа хранения
и обработки больших данных

Enterprise Tools

Инструментарий Enterprise Tools

Если согласно политике информационной безопасности доступ к ресурсам в нешним ресурсам ограничивается и требуется офлайн-установка, необходим инструментарий Arenadata Enterprise Tools.

Он позволяет создать локально инфраструктуру для развертывания продуктов Arenadata в закрытом контуре. Развертывание репозиториев выполняется из офлайн-пака, который поставляется с ADB Enterprise.

В состав входят:

- Docker registry. – *для организации офлайн-установки.*
- HTTP Mirror. – *для организации офлайн-установки.*
- Сервисы мониторинга. – *отдельный бандл мониторинга для ADB Enterprise не нужен.*
- CoreDNS.

Обновление ADB

1. Откройте веб-интерфейс своего сервера ADCM.
2. Загрузите бандл enterprise tools.
3. Создайте хост с ip-адресом, логином и паролем сервера adcm.
4. Создайте кластер enterprise tools.
5. Добавьте в кластер сервисы http mirror и docker registry.
6. Добавьте в кластер созданный хост.
7. Разместите сервисы на хосте.
8. Запустите установку (не офлайн).
9. Дождитесь окончания установки и запустите задачу upload pack.
10. Запустите процесс удаления расширения GPPerfmon в ADB и дождитесь его завершения.
11. Переведите кластер ADB в режим обновления.
12. Выполните импорт Docker Registry и HTTP Mirror в кластере ADB.
13. Запустите процесс обновления и проконтролируйте его до успешного завершения.
14. Добавьте сервис ADBCC, разместите его на хосте mdw и запустите процесс установки.

ARENADATA

Корпоративная платформа хранения
и обработки больших данных

Подключение к СУБД

Работа в ADB: консольный клиент psql

В стандартной поставке ADB присутствует консольный клиент psql. Им можно подключиться к базе, вызвав его по названию в командной строке и передав параметры через ключи:

1. -h адрес мастер-сервера.
2. -p порт СУБД.
3. -d название БД.
4. -U имя пользователя.

При этом:

1. При запуске утилиты непосредственно на мастер-сервере адрес можно не указывать.
2. Порт тоже можно не указывать, будет взят PGPORT.
3. Имя БД нужно обязательно. Если это единственный параметр, ключ -d не нужен, достаточно названия.
4. Если не указать имя пользователя, будет взят пользователь ОС.

Т.е. для работы на учебном кластере достаточно перейти под пользователя gpadmin командой `sudo su - gpadmin` и вызвать клиент с указанием имени БД командой `psql adb`.

Работа в ADB: консольный клиент psql

При работе с консольным клиентом могут пригодится следующие команды:

1. `-c 'sql query'` – выполнить SQL-запрос и завершить сессию.
2. `-f filename` – выполнить запрос из файла и завершить сессию.
3. Использование в скриптах: `psql -d adb -Atc 'select * from pg_class'`.
4. `\?` – справочник по внутренним командам psql.
5. `\q` – закрыть подключение к СУБД.
6. `\d+` – вывести подробное описание объекта (таблицы или представления).
7. `\x` – включить построчное отображение.
8. `\timing` - включить отображение времени выполнения запроса.

При этом следует помнить, что:

- Запросы завершаются точкой с запятой.
- Точка с запятой после ввода специальной команды клиента команды не требуется.

Лабораторная. Подключение к ADB

Подключитесь к БД при помощи консольного клиента.

Для учебного кластера:

1. Адрес мастер-сервера у каждого свой. Правильно определите IP-адрес мастера в описании.
2. Порт был в материале ранее.
3. Название БД: adb.
4. Имя пользователя: gpadmin.
5. Пароль: *пароля у пользователя gpadmin в нашем случае нет*.

Проверьте возможность выполнения запроса с использованием объекта в служебной схеме.

Следующий запрос должен выполниться: `select rsgname from gp_toolkit.gp_resgroup_status;`

- Выполните запрос: `select * from pg_catalog.pg_class where relname = 'pg_class';`
- Просмотрите информацию о структуре таблицы `pg_catalog.pg_class`.
- Просмотрите информацию о структуре таблицы `gp_toolkit.__gp_log_master_ext`.

ARENADATA

Корпоративная платформа хранения
и обработки больших данных

Обзор системы

Директории с настройками и данными

- Мастер-сегмент: `/data1/master/gpseg-1`
- Основные сегменты: `/data<disk>/primary/gpseg<contentid>/`

В них будут директории и файлы:

- `base` – директория с файлами таблиц `filespace pg_system`. Внутри данные в директориях вида:

`<db1 oid>/<table1 oid>`

`<db2 oid>/<table1 oid>`

- `db_analyze` – директория с информацией работы утилиты analyzedb (только на мастере, может переполниться, следите).
- `global` – директория с файлами global-таблиц.
- `gpperfmon` – логи, конфиги и т.д. Gpperfmon (только на мастере).
- `gpssh.conf` – тонкие настройки gpssh.
- `pg_clog` – журналы метаданных транзакций.
- `pg_distributedlog` – файлы синхронизации *основной сегмент -> зеркало*.
- `pg_hba.conf` – настройки доступа к БД (можно изменять только на мастере).

Директории с настройками и данными

- `pg_log` – логи (можно изменять).
- `pg_multixact` – файлы синхронизации *основной сегмент -> зеркало*.
- `pg_stat_tmp` – временные файлы сборщика статистики.
- `pg_subtrans` – служебная информация о транзакциях.
- `pg_tblspc` – символьные ссылки на tablespaces (не используется).
- `pg_twophase` – файлы синхронизации *основной сегмент -> зеркало*.
- `pg_utilitymodedtmredo` – служебная директория для redo в служебном режиме.
- `PG_VERSION` – актуальная версия PG.
- `pg_xlog` – журнал транзакций.
- `plcontainer_configuration.xml` – конфигурация расширения PL/Container.
- `postgresql.conf` – настройки БД (не рекомендуется менять напрямую).
- `postmaster.opts` – опции, с которыми был запущен процесс БД.
- `postmaster.pid` – PID процесса БД и дополнительная информация.

Процессы: Мастер

- `/usr/lib/gpdb/bin/postgres -D /data1/master/gpseg-1 -p 5432 –E`
- `/usr/lib/gpdb/bin/gpmmon -D /data1/master/gpseg-1/gpperfmon/conf/gpperfmon.conf -p 5432`
- `/usr/lib/gpdb/bin/gpsmon -m 0 -t 150 -l /data1/master/gpseg-1/gpperfmon/logs -v 0 8888`

Каждый запрос порождает процесс следующего вида:

```
postgres:  5432, gpadmin adb [local] con748725 seg-1 cmd9 slice1 MPPEXEC INSERT
```

По con[0-9] (это session_id) можно связать процесс на мастере с процессами на сегментах.

Процессы: Сегменты

- `/usr/lib/gpdb/bin/postgres -D /data1/mirror/gpseg3 -p 10501`
- `/usr/lib/gpdb/bin/gpsmon -m 0 -t 150 -l /data2/primary/gpseg5/gpperfmon -v 0 8888`

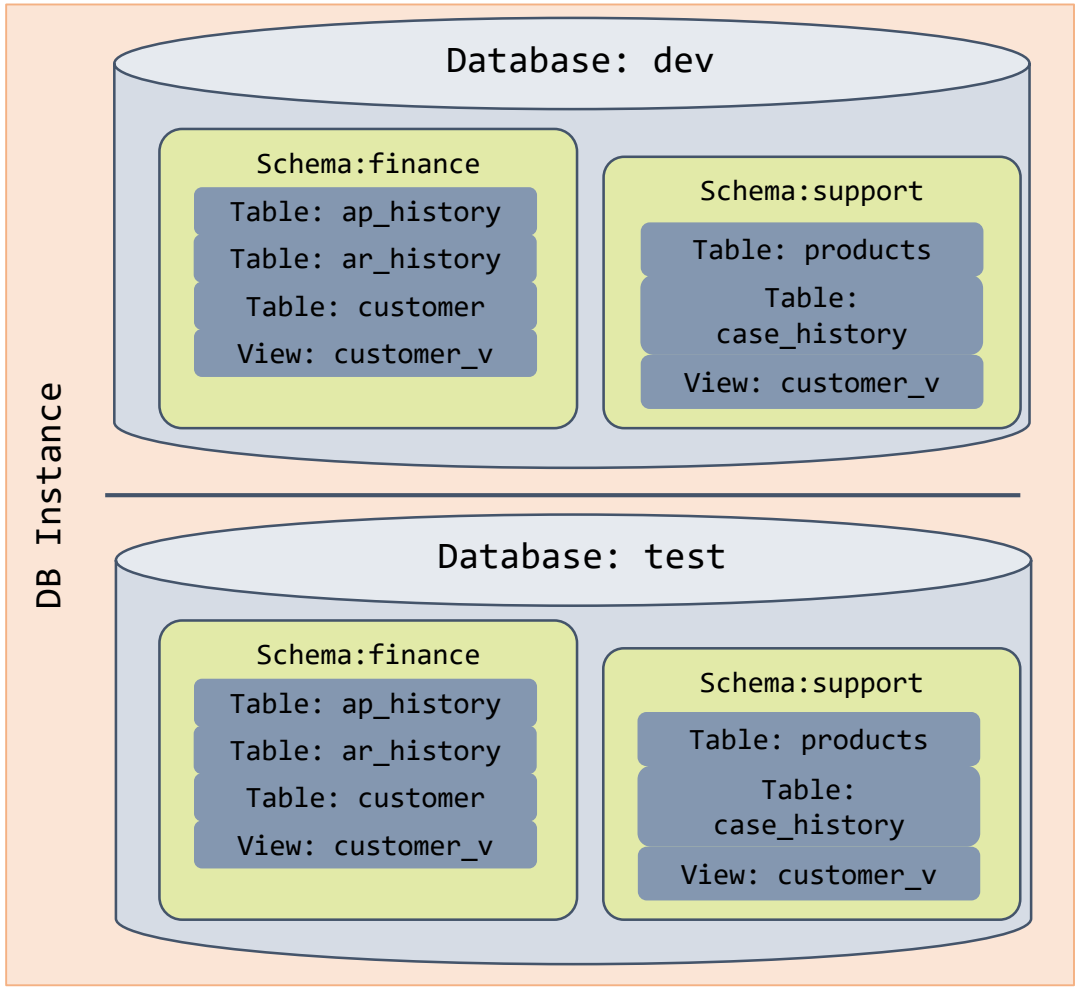
Каждый запрос порождает процесс следующего вида:

```
postgres: 40001, gpadmin adb 192.168.100.13(19273) con748725 seg1 cmd9 MPPEXEC INSERT
```

Иерархия

Уровни:

- База Данных:
  - Несколько БД в одной СУБД.
  - Перенос данных между БД невозможен.
  - БД разделены физически.
- Схема:
  - Логическое объединение сущностей внутри БД.
- Объект:
  - Таблица, представление, функция, т.д..
- Имя объекта:
  - ASCII standard: `database.schema.object`
  - Но лучше: `schema.object`



Шаблоны

- При инициализации СУБД создаётся три БД-шаблона:
  - `postgres`.
  - `template0`.
  - `template1`.
- `CREATE DATABASE` копирует `template1` со всеми объектами.
- Не следует изменять `template0`, это шаблон пустой БД, используемый утилитой `gprestore`.

Схемы по умолчанию

В ADB создаются автоматически схемы:

- `arenadata_toolkit` – настройки и информация о распределении таблиц и выполненных операциях.
- `gp_toolkit` – информация о размерах таблиц, раздувании таблиц, индексах, логах, ресурсных группах, спилл-файлах.
- `information_schema` – информация об объектах БД согласно стандарту ANSI SQL 2008.
- `pg_aoseg` – служебная схема информации AO-таблиц.
- `pg_bitmapindex` – служебная схема информации bitmap-индексов.
- `pg_catalog` – полная информация о БД и её объектах.
- `pg_toast` – служебная схема информации хранилища toast.
- `public` – схема для данных, доступная всем пользователям.

Поисковый путь

- Полное наименование объекта состоит из явно указанной схемы и имени объекта: `<схема.имя>`.
- Если не указать схему, то системе нужно понять, в какой схеме искать или создавать объект.
- Определяется схема с помощью пути поиска, который задается в переменной `search_path`.
- В параметре search_path можно через запятую перечислить схемы, в которых система будет искать объект, если схема в запросе не была указана явно.
- Несуществующие схемы и схемы, на которые отсутствуют права доступа, будут пропускаться.
- Схема `pg_catalog` включается по умолчанию, даже если не указана.
- При создании нового объекта, он будет помещаться в первую указанную в `search_path` схему.
- Проверить текущее значение переменной можно запросом `SHOW search_path;`
- Реальное значение search_path показывает функция current_schemas(): `SELECT current_schemas(true);`

Полезные объекты в системных схемах

- `pg_catalog.pg_class` – справочник всех объектов в БД. Содержит OID – системный идентификатор объекта (как и большинство системных таблиц).

```
adb=# select oid,relname, relnamespace, relstorage, relpersistence, reltuples from pg_class where relname
='pg_class';
 oid  | relname  | relnamespace | relstorage | relpersistence | reltuples
 1259 | pg_class |           11 |      h     |       p        |    520
```

`Relstorage`: a - row-oriented append-optimized, c- column-oriented, h - heap, v - представление, m - матвью, x - внешняя таблица.
`Relpersistence:` p - обычная таблица, u - unlogged временная таблица, t – простая временная таблица

- `pg_catalog.pg_namespace` – справочник всех схем БД.

```
adb=# select oid,nspname from pg_namespace  where oid=11;
 oid |  nspname
-----+------------
  11 | pg_catalog
```

- `Information_schema.tables`, `information_schema.schemata` – то же самое, но меньше информации и согласовано с ANSI SQL 2008.

Полезные объекты в системных схемах

- `pg_catalog.gp_segment_configuration` – информация о статусе сегментов, зеркал и мастеров:
  - `dbid` – уникальный id инстанса (разный у праймари и зеркала). Целое число, отсчёт с 1.
  - `content` – уникальный id шарда данных (одинаковый у праймари и зеркала). Целое число, отсчет с -1.
  - `role` – текущая роль сегмента. Варианты: p – основной сегмент, m – зеркало.
  - `preferred_role` – изначальная роль сегмента. Варианты: p – основной сегмент, m – зеркало.
  - `mode` – статус синхронизации основного сегмента с зеркалом. Варианты: s – норма, n – нет синхронизации.
  - `status` – отвечает ли сегмент на запросы мастера. Варианты: u – отвечает, d – недоступен.
  - `datadir` – физическое расположение рабочих директорий на сегменте и мастере.
  - port – порт сегмента.
  - hostname – имя сервера.
  - address – адрес сервера.

- `pg_catalog.gp_configuration_history` – историческая информация с описанием событий.

Заглядывайте сюда при сбоях – важно выявить первый произошедший отказ.

Полезные объекты в системных схемах

- `pg_catalog.pg_stat_activity` – текущие сессии СУБД:

```
adb=#  select pid, usename,  waiting, waiting_reason, rsgname from pg_catalog.pg_stat_activity;
  pid  | usename | waiting | waiting_reason |   rsgname
 73830 | gpadmin | f       |                | admin_group
```

- `pg_catalog.pg_settings` – все настройки СУБД.

- `gp_toolkit.gp_log_database` – внешняя таблица с логами СУБД на всех сегментах и мастере.
Могут быть проблемы с кодировкой и производительностью.

- `gp_toolkit.gp_resgroup_status` – детальная информация о ресурсных группах.

Лабораторная. Служебные объекты ADB

1. Выясните с помощью одного SQL-запроса, на каком сервере располагается зеркало сегмента с dbid=2.
2. Выясните, сколько внешних таблиц содержится в БД.
3. Выясните, в какой схеме находится таблица `stg_clients_rnd`.

ARENADATA

Корпоративная платформа хранения
и обработки больших данных

Ролевая модель

Ролевая модель

Ролевая модель построена на использовании универсального объекта ROLE:

- Роль – сущность в СУБД, которая может владеть объектами и иметь привилегии.
- Доступна иерархия. Объект ROLE может быть пользователем или группой.
- Роли могут включаться в другие роли, наследуя их привилегии (по-умолчанию).
- Для удобства пользовательские роли создаются с опцией LOGIN (возможность входа), а группы – без.
- Роли глобальны для всех БД внутри СУБД.
- Существует базовая роль public, в которую автоматически включаются все созданные роли. От неё наследуется право работать со схемой public.

Создание роли

```
CREATE ROLE name [[WITH]
      SUPERUSER | NOSUPERUSER – будет ли обладать роль правами администратора (доп. привелегии)
    | CREATEDB | NOCREATEDB
    | CREATEROLE | NOCREATEROLE
    | CREATEUSER | NOCREATEUSER
    | CREATEEXTTABLE | NOCREATEEXTTABLE
      [ ( attribute='value'[, ...] ) ]
           where attributes and value are:
           type='readable'|'writable'
           protocol='gpfdist'|'http'|'pxf'|'s3'
    | INHERIT | NOINHERIT
    | LOGIN | NOLOGIN
    | REPLICATION | NOREPLICATION
```

Создание роли

```
CREATE ROLE name [[WITH]
      SUPERUSER | NOSUPERUSER
    | CREATEDB | NOCREATEDB – может ли создавать базы
    | CREATEROLE | NOCREATEROLE
    | CREATEUSER | NOCREATEUSER
    | CREATEEXTTABLE | NOCREATEEXTTABLE
      [ ( attribute='value'[, ...] ) ]
           where attributes and value are:
           type='readable'|'writable'
           protocol='gpfdist'|'http'|'pxf'|'s3'
    | INHERIT | NOINHERIT
    | LOGIN | NOLOGIN
    | REPLICATION | NOREPLICATION
```

Создание роли

```
CREATE ROLE name [[WITH]
      SUPERUSER | NOSUPERUSER
    | CREATEDB | NOCREATEDB
    | CREATEROLE | NOCREATEROLE – может ли создавать роли
    | CREATEUSER | NOCREATEUSER – устаревшее, осталось для совместимости
    | CREATEEXTTABLE | NOCREATEEXTTABLE
      [ ( attribute='value'[, ...] ) ]
           where attributes and value are:
           type='readable'|'writable'
           protocol='gpfdist'|'http'|'pxf'|'s3'
    | INHERIT | NOINHERIT
    | LOGIN | NOLOGIN
    | REPLICATION | NOREPLICATION
```

Создание роли

```
CREATE ROLE name [[WITH]
      SUPERUSER | NOSUPERUSER
    | CREATEDB | NOCREATEDB
    | CREATEROLE | NOCREATEROLE
    | CREATEUSER | NOCREATEUSER
    | CREATEEXTTABLE | NOCREATEEXTTABLE – может ли создавать внешние таблицы
      [ ( attribute='value'[, ...] ) ] – и какие именно
           варианты опций:
           type='readable'|'writable'
           protocol='gpfdist'|'http'|'pxf'|'s3'
    | INHERIT | NOINHERIT
    | LOGIN | NOLOGIN
    | REPLICATION | NOREPLICATION
```

Создание роли

```
CREATE ROLE name [[WITH]
      SUPERUSER | NOSUPERUSER
    | CREATEDB | NOCREATEDB
    | CREATEROLE | NOCREATEROLE
    | CREATEUSER | NOCREATEUSER
    | CREATEEXTTABLE | NOCREATEEXTTABLE
      [ ( attribute='value'[, ...] ) ]
           where attributes and value are:
           type='readable'|'writable'
           protocol='gpfdist'|'http'|'pxf'|'s3'
    | INHERIT | NOINHERIT - наследование
    | LOGIN | NOLOGIN – возможность входа
    | REPLICATION | NOREPLICATION – полномочия по репликации
```

Создание роли

| CONNECTION LIMIT connlimit – **лимит подключений**

| [ENCRYPTED | UNENCRYPTED] PASSWORD 'password' – **пароль**

| VALID UNTIL 'timestamp' - **устаревание пароля**

| [DENY deny_point] | [DENY BETWEEN deny_point AND deny_point] – **ограничение по входу**

| SYSID uid [, ...]

| RESOURCE QUEUE queue_name

| RESOURCE GROUP group_name

| IN ROLE rolename [, ...]

| ROLE rolename [, ...]

| ADMIN rolename [, ...]

Создание роли

```
| CONNECTION LIMIT connlimit
| [ ENCRYPTED | UNENCRYPTED ] PASSWORD 'password'
| VALID UNTIL 'timestamp'
| [ DENY deny_point ] | [ DENY BETWEEN deny_point AND deny_point]
| SYSID uid [, ...]
| RESOURCE QUEUE queue_name – устаревшее, оставлено для совместимости
| RESOURCE GROUP group_name – привязать роль к ресурсной группе
| IN ROLE rolename [, ...]
| ROLE rolename [, ...]
| ADMIN rolename [, ...]
```

Создание роли

```
| CONNECTION LIMIT connlimit
| [ ENCRYPTED | UNENCRYPTED ] PASSWORD 'password'
| VALID UNTIL 'timestamp'
| [ DENY deny_point ] | [ DENY BETWEEN deny_point AND deny_point]
| SYSID uid [, ...]
| RESOURCE QUEUE queue_name
| RESOURCE GROUP group_name
| IN ROLE rolename [, ...] – включить создаваемую роль в указанные роли как в группы
| ROLE rolename [, ...]
| ADMIN rolename [, ...]
```

Создание роли

```
| CONNECTION LIMIT connlimit
| [ ENCRYPTED | UNENCRYPTED ] PASSWORD 'password'
| VALID UNTIL 'timestamp'
| [ DENY deny_point ] | [ DENY BETWEEN deny_point AND deny_point]
| SYSID uid [, ...]
| RESOURCE QUEUE queue_name
| RESOURCE GROUP group_name
| IN ROLE rolename [, ...]
| ROLE rolename [, ...] – включить указанные роли в создаваемую как членов группы
| ADMIN rolename [, ...] – то же самое, но с расширенными полномочиями
```

Изменение роли

Смена имени:

```
ALTER ROLE name RENAME TO newname;
```

Смена параметра конфигурации СУБД:

```
ALTER ROLE name SET config_parameter {TO | =} {value | DEFAULT};
```

Сброс параметра конфигурации СУБД:

```
ALTER ROLE name RESET config_parameter;
```

Привязка к ресурсной группе:

```
ALTER ROLE name RESOURCE GROUP {group_name | NONE};
```

Модификация опций роли (примечание: связи с другими ролями или объектами так изменить не получится):

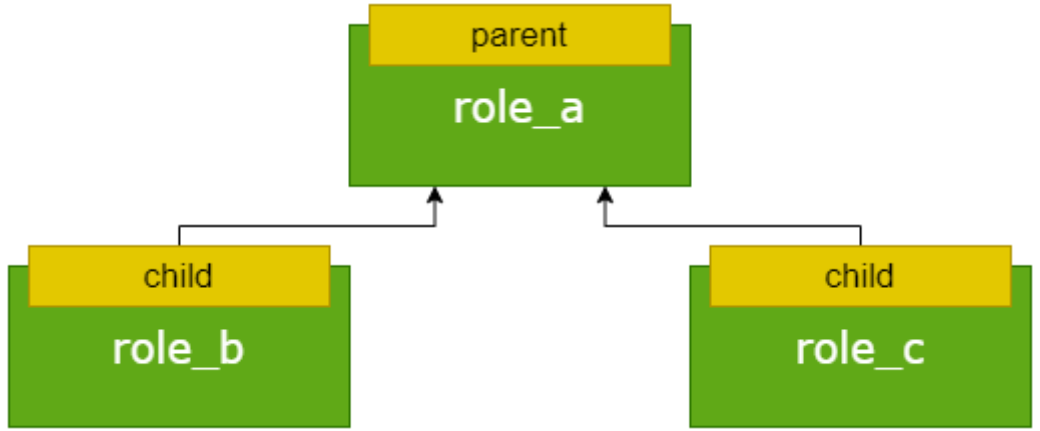
```
ALTER ROLE name [ [WITH] option [ ... ] ];
```

Установка иерархии после создания роли

Для включения ролей в группы используется команда GRANT:

```
GRANT parent_role [, ...] TO member_role [, ...] [WITH ADMIN OPTION]
```

```
GRANT role_a TO role_b, role_c;
```



Для удаления связи используется команда REVOKE:

```
REVOKE [ADMIN OPTION FOR] parent_role [, ...] FROM member_role [, ...] [CASCADE | RESTRICT]
```

Выдача полномочий. Таблица

Полномочия выдаются той же командой GRANT и отзываются при помощи REVOKE. Доступны:

- На таблицу:

```
GRANT { {SELECT | INSERT | UPDATE | DELETE | TRUNCATE } [,...] | ALL [PRIVILEGES] }
    ON [TABLE] tablename [, ...]
       | ALL TABLES IN SCHEMA schema_name [, ...] }
    TO {rolename | PUBLIC} [, ...] [WITH GRANT OPTION]
```

- На отдельные колонки таблицы, если не ко всем данным положено иметь доступ:

```
GRANT { { SELECT | INSERT | UPDATE } ( column_name [, ...] )
    [, ...] | ALL [ PRIVILEGES ] ( column_name [, ...] ) }
    ON [ TABLE ] table_name [, ...]
    TO { role_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]
```

Выдача полномочий. База и схема

Для доступа к таблице обязательно необходимо иметь доступ к базе и схеме, в которых она лежит:

- Схема:

```
GRANT { {CREATE | USAGE} [,...] | ALL [PRIVILEGES] }
    ON SCHEMA schemaname [, ...]
    TO {rolename | PUBLIC} [, ...] [WITH GRANT OPTION]
```

- БД:

```
GRANT { {CREATE | CONNECT | TEMPORARY | TEMP} [,...] | ALL [PRIVILEGES] }
    ON DATABASE dbname [, ...]
    TO {rolename | PUBLIC} [, ...] [WITH GRANT OPTION]
```

Выдача полномочий. Другие полномочия

- Последовательности:

```
GRANT { {USAGE | SELECT | UPDATE} [,...] | ALL [PRIVILEGES] } ON SEQUENCE sequencename …
```

- Функции:

```
GRANT { EXECUTE | ALL [PRIVILEGES] } ON FUNCTION funcname ( [ [argmode] [argname] argtype [, ...]] )…
```

- Языки для написания функций и безымянных блоков:

```
GRANT { USAGE | ALL [PRIVILEGES] } ON LANGUAGE langname …
```

- Тейблспейсы:

```
GRANT { CREATE | ALL [PRIVILEGES] } ON TABLESPACE tablespacename …
```

- Протоколы для внешних таблиц:

```
GRANT { SELECT | INSERT | ALL [PRIVILEGES] } ON PROTOCOL protocolname …
```

Отзыв полномочий и удаление роли

Для отзыва полномочий используется команда REVOKE:

- Отзыв связей в структуре ролей:

```
REVOKE [ADMIN OPTION FOR]  parent_role [, ...]  FROM member_role [, ...] [CASCADE | RESTRICT]
```

- Отзыв полномочий на использование таблицы:

```
REVOKE [GRANT OPTION FOR] { {SELECT | INSERT | UPDATE | DELETE | REFERENCES | TRIGGER | TRUNCATE } [, ...] |
 ALL [PRIVILEGES] } ON { [TABLE] table_name [, ...] | ALL TABLES IN SCHEMA schema_name [, ...] }
 FROM { [ GROUP ] role_name | PUBLIC} [, ...] [CASCADE | RESTRICT]
```

- Отзыв полномочий на администрирование групповой роли:

```
REVOKE [ADMIN OPTION FOR] parent_role [, ...] FROM [ GROUP ] member_role [, ...] [CASCADE | RESTRICT]
```

Доступ к СУБД

Помимо роли с параметром LOGIN, пользователю необходимо для подключения к СУДБ иметь запись в файле `pg_hba.conf` на мастер-сервере. Этот файл отвечает за проверку параметров коннекта. На сегментах править этот файл не следует.

Внутри файла текстовые строки для описания локалных и удаленных подключений:

- host database role address authentication-method
- local database role authentication-method

Добавить всех пользователей группы можно указав имя роли с плюсом: +group_name.

Добавить всех пользователей из отдельного файла: @filename.

Адрес указывается с подсетью: 172.20.143.0/24.

Можно указать 0.0.0.0/0 для коннектов с любого адреса.

Лабораторная. Создание ролей

1. Создайте суперпользователя adb_super с правами входа в систему, ресурсной группой admin_group и паролем “superpassword”.
2. Создайте группу adb_group без прав входа в систему.
3. Создайте пользователя adb_1 с правом входа в систему добавьте его в группу adb_group.
4. Создайте пользователя adb_2 с правом входа в систему.
5. Добавьте пользователя adb_2 в группу adb_group.
6. Дайте полные права на протокол PXF группе adb_group.
7. Дайте права на чтение таблицы arenadata_toolkit.db_files_current группе adb_group.
8. Смените пароль пользователю adb_1 на “password_1”.

ARENADATA

Корпоративная платформа хранения
и обработки больших данных

Ресурсные группы

Ресурсные группы

- **Resource queue** – ресурсные очереди – старый механизм (до версии 5.3).
- **Resource groups** – ресурсные группы (РГ) – новый механизм (начиная с версии 5.3):
  - РГ – набор ограничений, накладываемых на роли (пользователей).
  - Используют Linux Control Groups (cgroups) и vmtracker.
  - Также позволяют контролировать внешние компоненты (например, PL/Container).

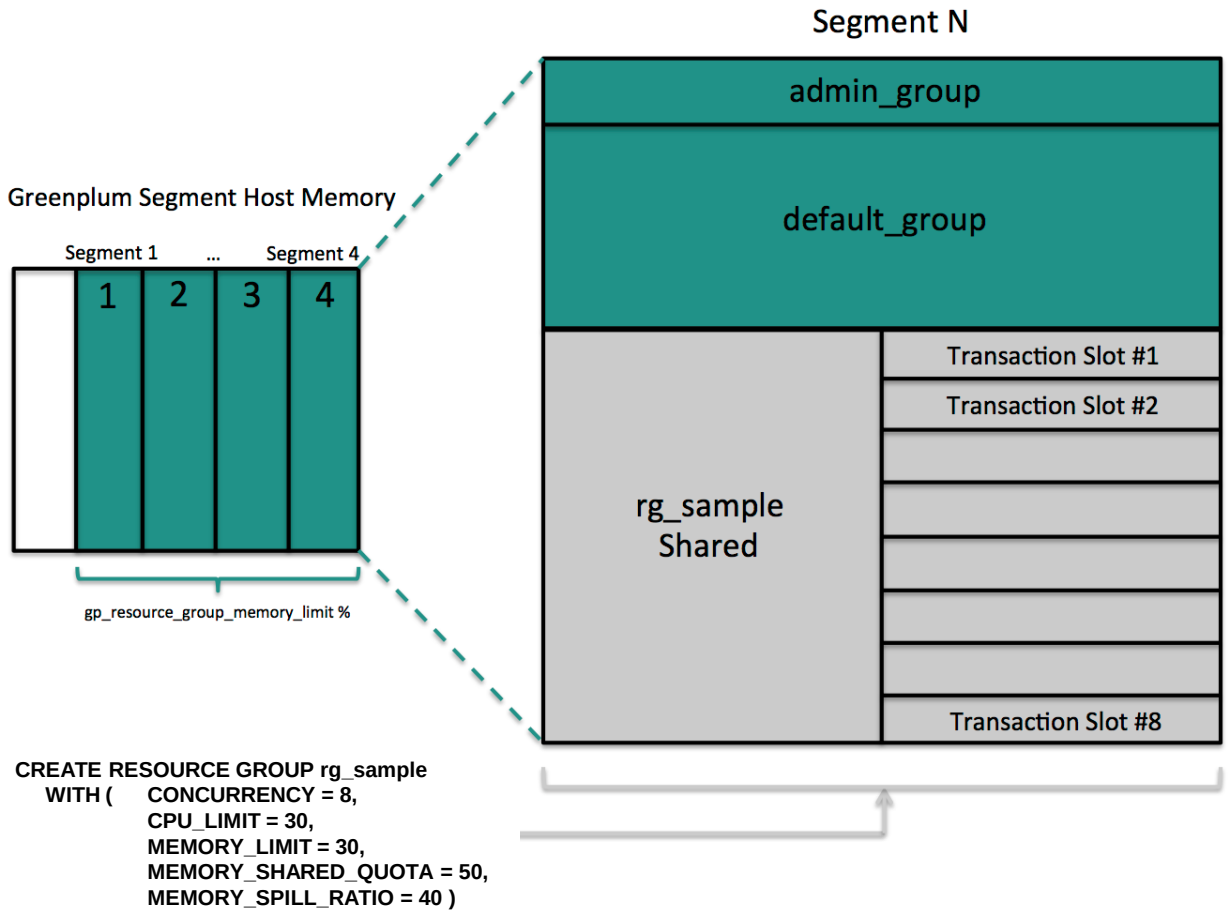
Параметры ресурсных групп

Атрибут	Описание	Значение по умолчанию
MEMORY_AUDITOR	Vmtracker – для управления ролями СУБД; Cgroups – для управления внешними компонентами	vmtracker
CONCURRENCY	Число параллельных транзакций (включая IDLE). Можно обойти с помощью gp_resource_group_bypass=true, но при этом запросу доступно только 10 MB RAM	20
CPU_RATE_LIMIT	Процент CPU кластера, доступный РГ. Сумма параметров всех РГ должна быть <=100. Общее количество CPU задаётся gp_resource_group_cpu_limit . Не совместим с CPUSET . Выделение гибкое – пока есть свободные ресурсы, будут выдаваться всем	-
CPUSET	Перечень ядер, зарезервированных под данную РГ. Запросы РГ будут утилизировать только данные ядра. Не совместим с CPU_RATE_LIMIT. Если данная РГ не использует ядра, они будут простаивать. Не используйте ядро 0. Формат: '1,3-4'	-
MEMORY_LIMIT	Процент RAM кластера, доступный РГ. Сумма параметров всех РГ должна быть <=100. Общее количество памяти задаётся gp_resource_group_memory_limit . Выделение гибкое – пока есть свободные ресурсы, будут выдаваться всем	0
MEMORY_SHARED_QUOTA	Процент RAM РГ, который будет доступен всем транзакциям РГ. (100-MEMORY_SHARED_QUOTA)/CONCURRENCY – будет гарантировано каждой транзакции в РГ	80
MEMORY_SPILL_RATIO	Процент RAM, по достижении которого запрос спилит данные на диск. Можно задавать в сессии параметром memory_spill_ratio. Параметр влияет на начальное выделение памяти. Малое значение может ускорить простые запросы. Если memory_spill_ratio = 0, то для выделения первоначальной памяти используется значение параметра statement_mem.	0

Нюансы

- **gp_resource_group_cpu_limit** – по умолчанию задаётся 0.9. Если у вас есть другие критичные ресурсоёмкие приложения на кластере, уменьшите значение;
- **gp_resource_group_memory_limit** – по умолчанию задаётся 0.7. Если у вас есть другие критичные ресурсоёмкие приложения на кластере, уменьшите значение. Выбирайте параметр, исходя из стратегии зеркалирования.
- Оставляйте 10-20% памяти неаллоцированной для непредсказуемых запросов.
- СУБД опирается на статистику при подсчёте необходимого количества памяти (однако, также считает и актуальное использование).
- При превышении concurrency запрос будет поставлен в очередь.
- Используйте gp_toolkit.gp_resgroup_config, gp_toolkit.gp_resgroup_status, gp_toolkit.gp_resgroup_status_per_host и gp_toolkit.gp_resgroup_status_per_segment для контроля потребления ресурсов.

Распределение памяти




```
CREATE RESOURCE GROUP rg_sample
    WITH (    CONCURRENCY = 8,
              CPU_LIMIT = 30,
              MEMORY_LIMIT = 30,
              MEMORY_SHARED_QUOTA = 50,
              MEMORY_SPILL_RATIO = 40 )
```

ARENADATA

Создание ресурсной группы

```
CREATE RESOURCE GROUP name WITH (

        CPU_RATE_LIMIT=integer | CPUSET=tuple

        MEMORY_LIMIT=integer

        [ CONCURRENCY=integer ]

        [ MEMORY_SHARED_QUOTA=integer ]

        [ MEMORY_SPILL_RATIO=integer ]

        [ MEMORY_AUDITOR= {vmtracker | cgroup} ]

)

ALTER RESOURCE GROUP name SET group_attribute value

ALTER ROLE name RESOURCE GROUP {group_name | NONE}
```


ARENADATA

Корпоративная платформа хранения
и обработки больших данных

Расширение Diskquota

Diskquota

- **diskquota** – модуль, который позволяет ограничить используемое место на дисках для схем и ролей.
- Квота на уровне схем: устанавливает лимит на размер для всех таблиц принадлежащих конкретной схеме.
- Квота на уровне ролей: устанавливает лимит на размер для всех таблиц владельцем которых является конкретная роль.
- Устанавливается в виде расширения и использует отдельную базу данных с именем «diskquota».
- В дистрибутиве ADB, расширение устанавливается опционально с помощью ADCM.

Использование расширения

- Если расширение **diskquota** было добавлено после того, как в СУБД уже были данные, то необходимо выполнить первоначальную инициализацию модуля для каждой БД:

  SELECT diskquota.init_table_size_table();

- Для установки квоты на уровне схем используется функция diskquota.set_schema_quota():

  SELECT diskquota.set_schema_quota('stg', '250GB');

- Для установки квоты на уровне ролей используется функция diskquota.set_role_quota():

  SELECT diskquota.set_role_quota('test_user', '500GB');

- Для снятия квоты на уровне ролей или схем используются функции diskquota.set_role_quota() и diskquota.set_schema_quota() с параметром «-1»:

  SELECT diskquota.set_schema_quota('stg', '-1');

  SELECT diskquota.set_role_quota(‘test_user', '-1');

Нюансы

- Кол-во доступных баз данных для включения дисковых квот – не больше 10.
- Для каждой базы данных создаётся дополнительный процесс на мастер-сервере.
- Используемое место – это размер таблиц на всех сегментах, включая индексы и служебные таблицы.
- Частота обновления данных по дисковым квотам зависит от значения параметра diskquota.naptime.
- В некоторых случаях квота может быть превышена.
- Для просмотра установленных квот используйте представления diskquota.show_fast_schema_quota_view и diskquota.show_fast_role_quota_view.
- diskquota-модуль не учитывает размер таблиц, которые были созданы внутри ещё не закомиченной транзакции или с помощью команды CREATE TABLE AS.

Конец первой части