APACHE HAWQ

SQL для HADOOP на базе PostgreSQL

Содержание

- Что такое Apache HAWQ?
- Архитектура
- Форматы хранения данных
- Разделение ресурсов
- Выполнение запросов
- Альтернативные решения

EMC² Pivotal RSA EMC Converged Platforms VCE virtustream vmware

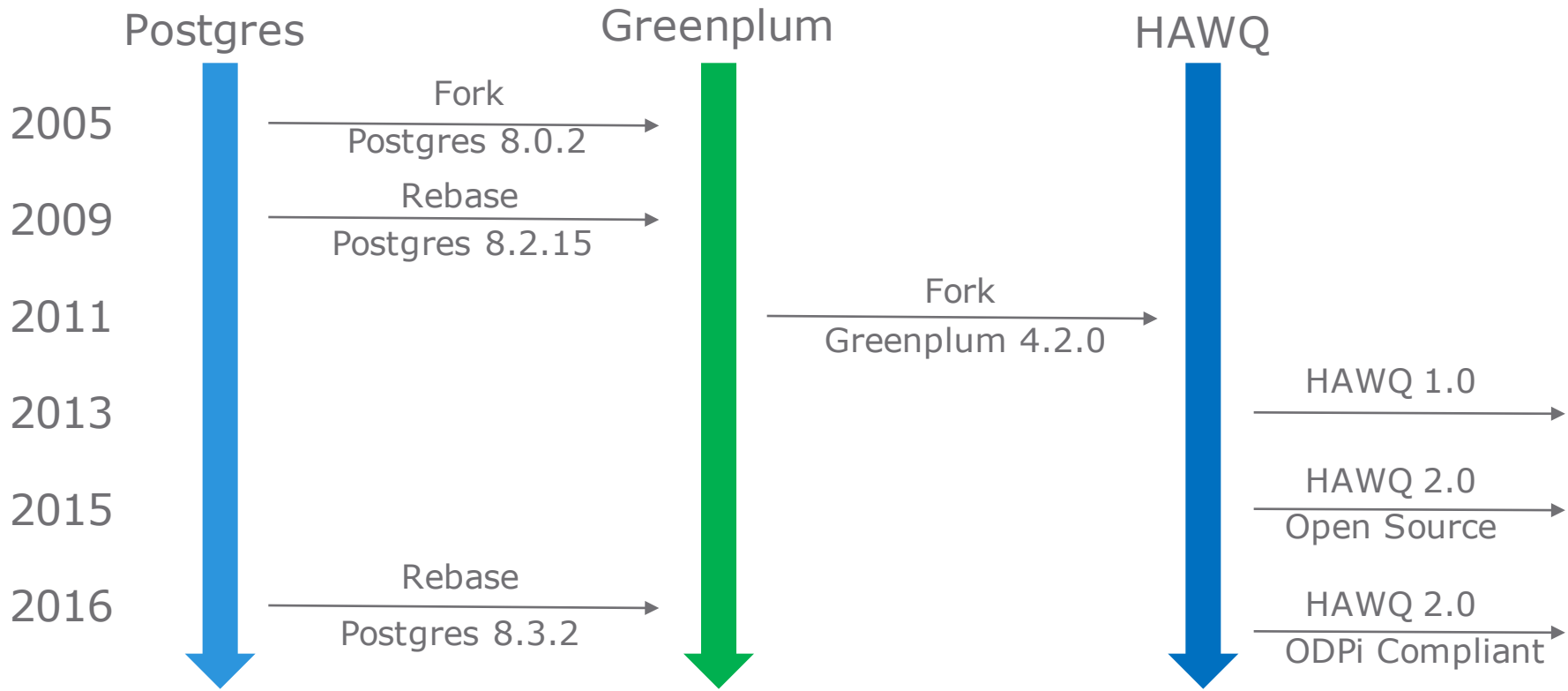
Что такое Apache HAWQ

Apache HAWQ

- HAdoop With Queries
- «Движок» для осуществления SQL запросов к Hadoop с богатыми аналитическими возможностями

"It does look similar—but this one is powered by Hadoop"

История проекта



HAWQ – Это …

- 1’500’000 строк кода C и C++
  - Из которых 200’000 заголовочных файлов
- 180’000 строк кода Python
- 60’000 строк кода Java
- 23’000 строк Makefile’ов
- 7’000 строк shell-скриптов
- Более 50 корпоративных клиентов

HAWQ – Open Source

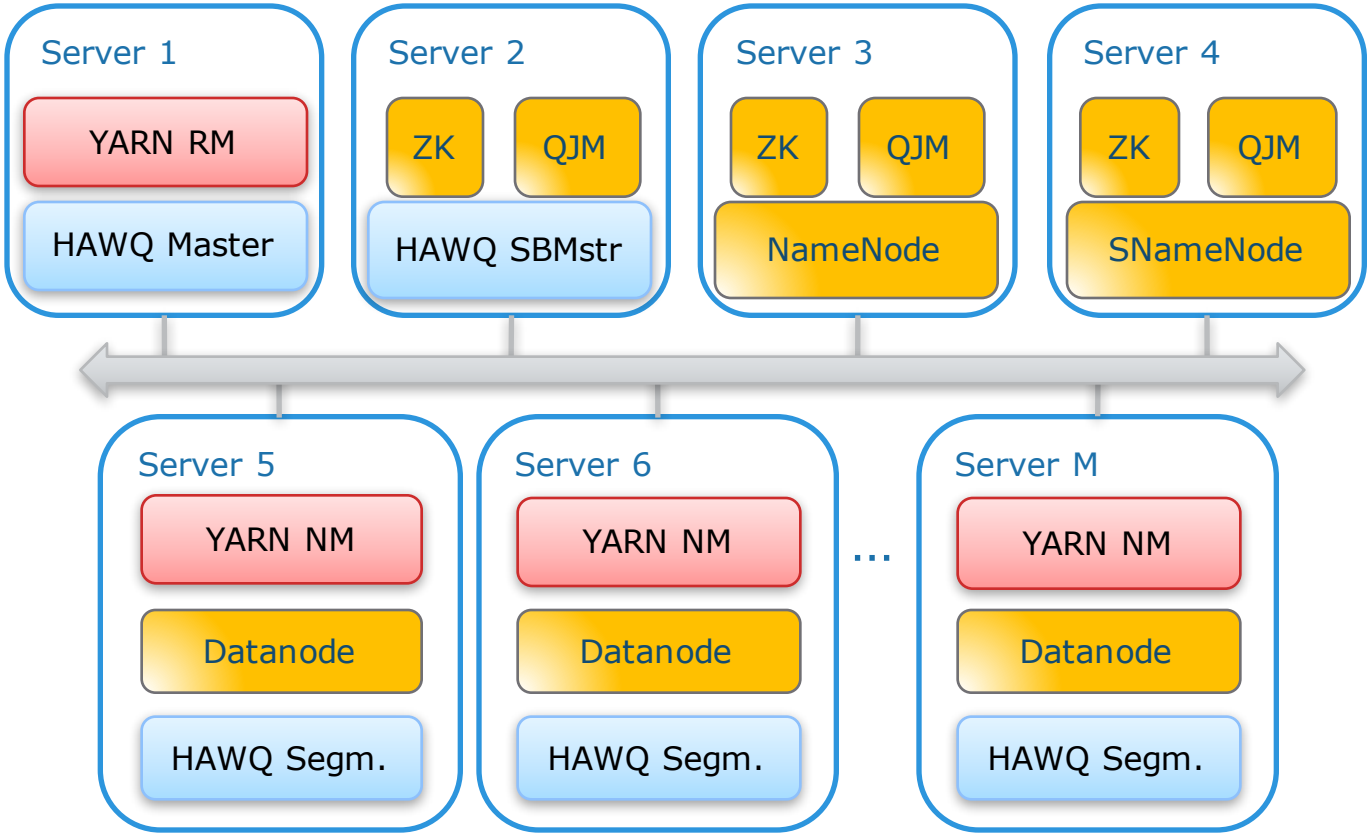
- Apache HAWQ (incubating) с 09.2015
  - http://hawq.incubator.apache.org
  - https://github.com/apache/incubator-hawq
- Что находится в Open Source
  - Исходный код HAWQ 2.0
  - ORCA Optimizer
  - Pivotal Extension Framework (PXF)

Применение Apache HAWQ

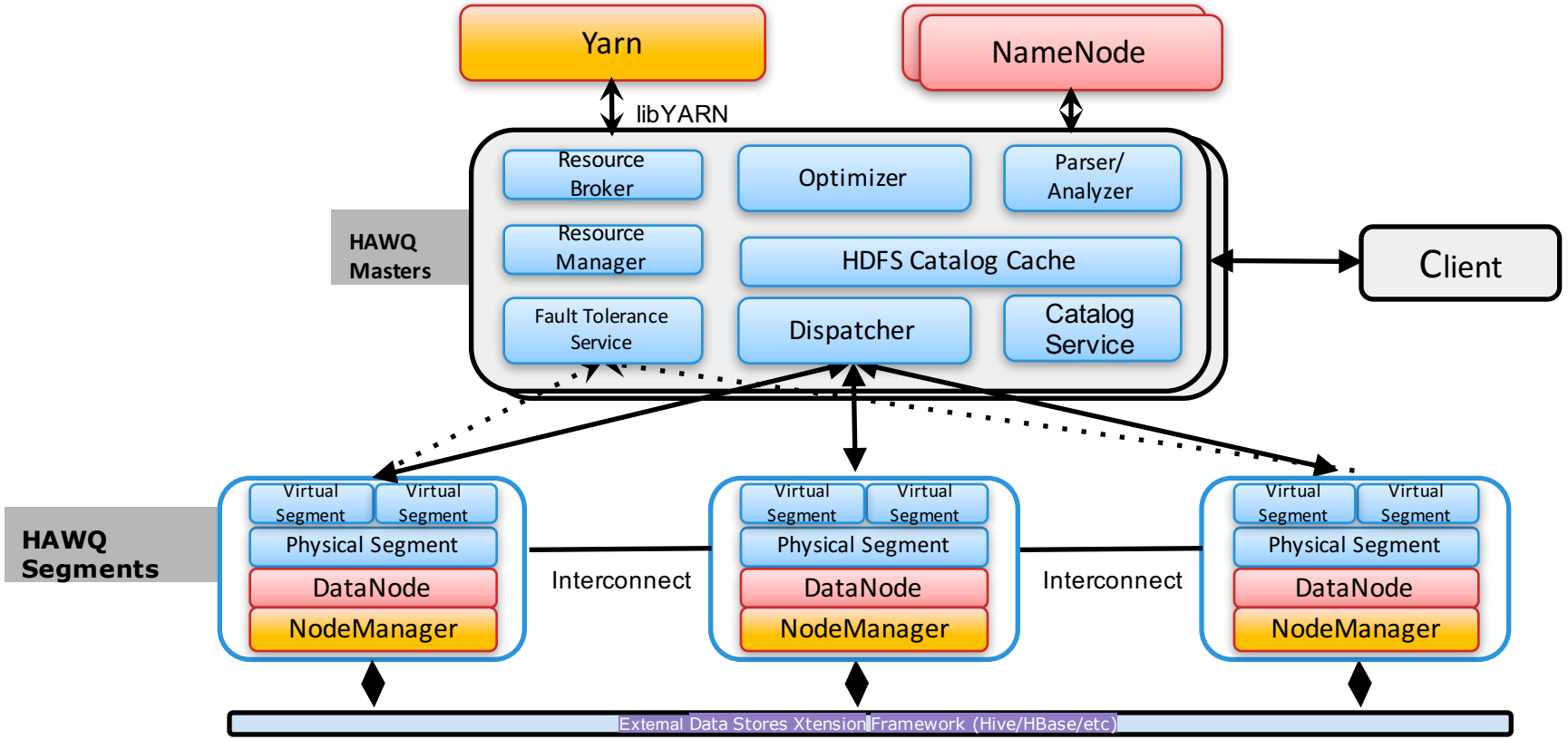
- Универсальный SQL - интерфейс к данным Hadoop для BI с ANSI SQL-93,-99,-2003
  - Пример из практики – запрос Cognos на 5000 строк с множеством функций
- Универсальный инструмент для ad-hoc аналитики
  - Пример из практики распарсить URL, извлечь из него имя хоста и пртокол
- Хорошая производительность
  - Отказ от использования MR и применение оптимизатора плана выполнения

Архитектура HAWQ

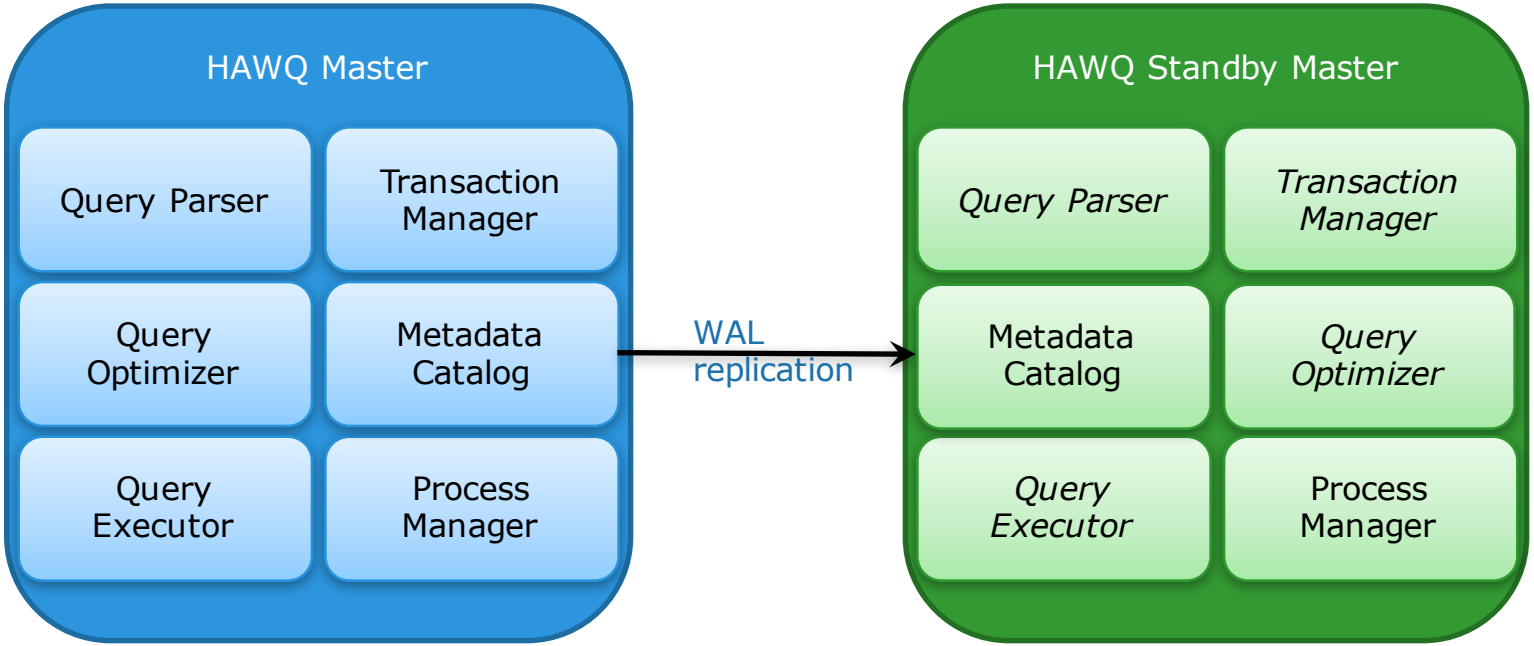
Кластер HAWQ



Архитектура HAWQ 2.0



Мастер сервера



Метаданные

- Структура аналогична структуре каталога Postgres
- Статистика
  - Количество записей и страниц в таблице
  - Наиболее частые значения для каждого поля
  - Гистограмма для каждого числового поля
  - Количество уникальных значений в поле
  - Количество null значений в поле
  - Средний размер значения поля в байтах

Метаданные

- Информация о структуре таблицы
  - Поля распределения
  - Количество hash bucket распределения
  - Партиционирование (hash, list ,range)
- Общие метаданные
  - Пользователи и группы
  - Права доступа к объектам
- Хранимые процедуры
  - PL/pgSQL, PL/Java, PL/Python, PL/Perl, PL/R

Статистика

- Без статистики
  - Join двух таблиц, потребует перебор записей:
    - От 0 до бесконечности
- Количество записей
  - Join двух таблиц по 1000 записей в каждой, потребует перебор записей:
    - От 0 до 1’000’000
- Гистограммы и MCV
  - Join двух таблиц по 1000 записей в каждой, с известной кардинальностью, гистограммой распределения, MCV, null:
    - От 500 до 1’500

Оптимизатор запросов

- Используется cost-based оптимизатор
- Выбрать можно один из двух:
  - Planner – модифицированный оптимизатор Postgres
  - ORCA (Pivotal Query Optimizer) – разработан специально для HAWQ и портирован на Greenplum
- Хинты оптимизатора:
  - Включить/отключить определенную операцию
  - Изменить веса базовых операций

Пример плана

```
SELECT * FROM r JOIN p ON r.a = p.a
```

Legacy Optimizer

```
QUERY PLAN
------------------------------------------------------------------
Gather Motion 2:1  (slice2; segments: 2)  (cost=39.01..3653.51 rows=50450 width=16)
  ->  Hash Join  (cost=39.01..3653.51 rows=50450 width=16)
      Hash Cond: public.p.a = public.r.a
      InitPlan  (slice3)
        ->  Aggregate  (cost=14.50..14.51 rows=1 width=32)
            ->  Gather Motion 2:1  (slice1; segments: 2)  (cost=0.00..12.00 rows=500 width=8)
                  ->  Seq Scan on r  (cost=0.00..12.00 rows=500 width=8)
      ->  Append  (cost=0.00..2101.00 rows=50450 width=8)
            ->  Result  (cost=0.00..1.99 rows=50 width=8)
                  One-Time Filter: 30607::oid = ANY ($0)
                  ->  Seq Scan on p_1_prt_extra p  (cost=0.00..1.99 rows=50 width=8)
            ->  Result  (cost=0.00..12.00 rows=500 width=8)
                  One-Time Filter: 30630::oid = ANY ($0)
                  ->  Seq Scan on p_1_prt_2 p  (cost=0.00..12.00 rows=500 width=8)
...
```

(3014 rows)

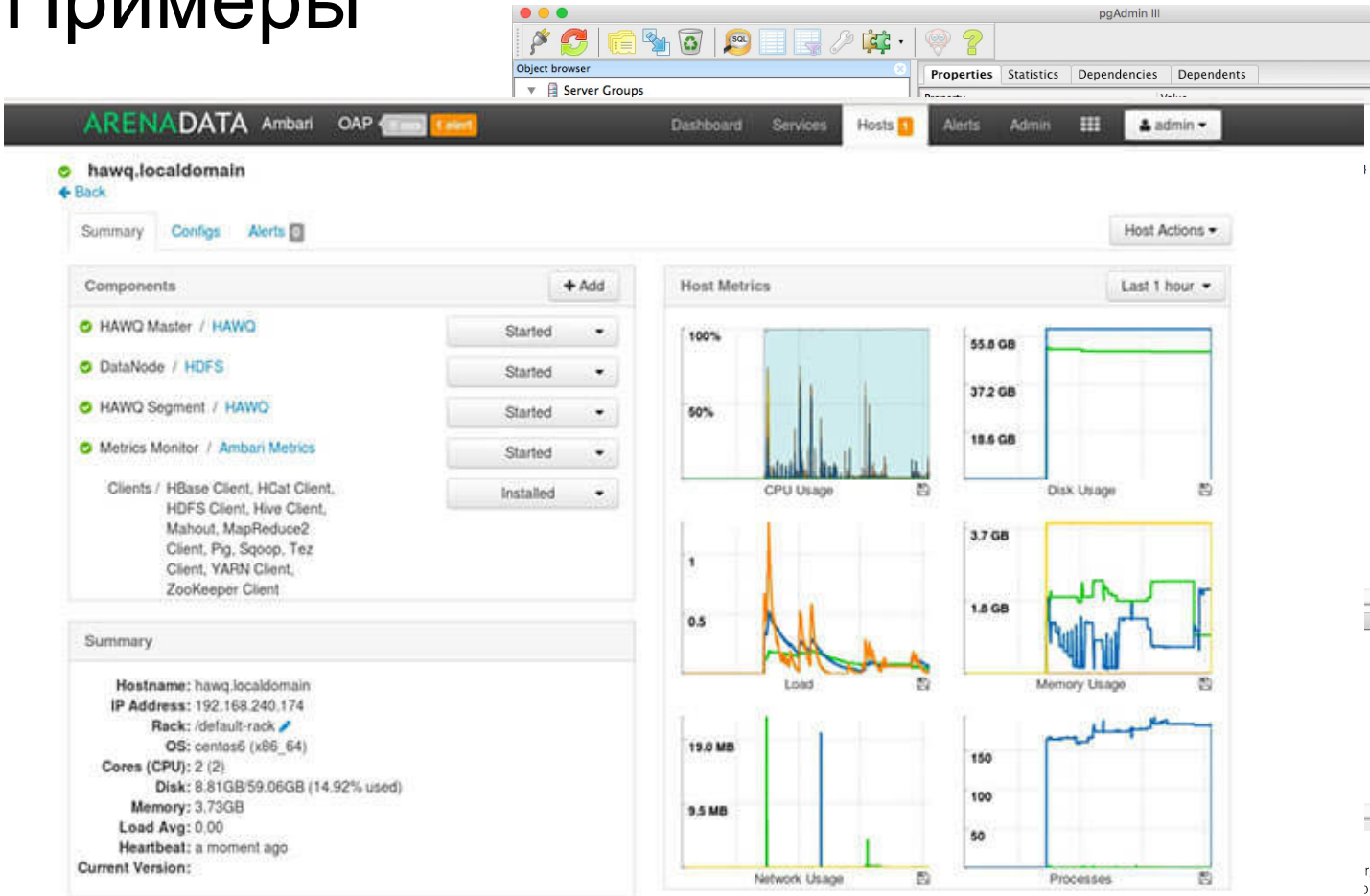
Orca

```
QUERY PLAN
------------------------------------------------------------------------------------------
 Gather Motion 2:1  (slice1; segments: 2)  (cost=10.00..100.00 rows=1 width=12)
   ->  Hash Join  (cost=10.00..100.00 rows=1 width=12)
       Hash Cond: p.a = r.a
       -> Dynamic Table Scan on p (partIndex: 1)  (cost=10.00..100.00
rows=500 width=12)
       ->  Hash  (cost=100.00..100.00 rows=500 width=12)
             ->  Result  (cost=10.00..100.00 rows=500 width=12)
                   ->  Seq Scan on r  (cost=10.00..100.00 rows=500 width=12)
Settings:  optimizer=on
```

(8 rows)

Адмнистрирование и мониторинг

Примеры


pgAdmin III
Object browser
Server Groups
Properties
Statistics
Dependencies
Dependents
ARENADATA
Ambari
OAP
Dashboard
Services
Hosts
Alerts
Admin
admin
hawq.localdomain
Back
Summary
Configs
Alerts
Host Actions
Components
+ Add
HAWQ Master / HAWQ
Started
DataNode / HDFS
Started
HAWQ Segment / HAWQ
Started
Metrics Monitor / Ambari Metrics
Started
Clients / HBase Client, HCat Client, HDFS Client, Hive Client, Mahout, MapReduce2 Client, Pig, Sqoop, Tez Client, YARN Client, ZooKeeper Client
Installed
Summary
Hostname: hawq.localdomain
IP Address: 192.168.240.174
Rack: /default-rack
OS: centos6 (x86_64)
Cores (CPU): 2 (2)
Disk: 8.81GB/59.06GB (14.92% used)
Memory: 3.73GB
Load Avg: 0.00
Heartbeat: a moment ago
Current Version:
Host Metrics
Last 1 hour
100%
50%
CPU Usage
55.8 GB
37.2 GB
18.6 GB
Disk Usage
1
0.5
Load
3.7 GB
1.8 GB
Memory Usage
19.0 MB
9.5 MB
Network Usage
150
100
50
Processes

Форматы хранения данных

Какой формат является оптимальным?

Зависит от критерия оптимальности:

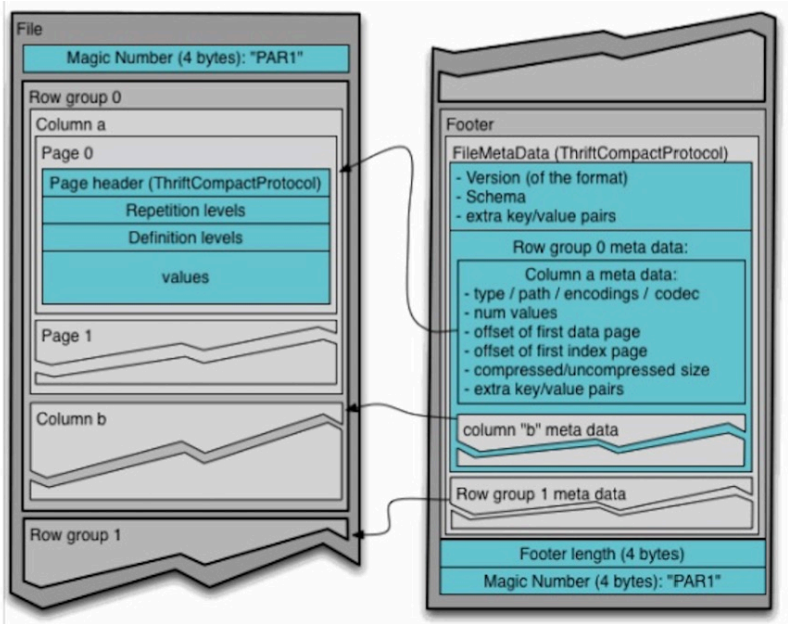
- Минимальное потребление ресурсов CPU
- Минимальный объем занимаемого дискового пространства
- Минимальное время извлечения записи по ключу
- Минимальное время извлечения подмножества столбцов
- Etc…

Формат хранения

- Построчное хранение
  - Модифицированный формат postgres
    - Без toast
    - Без ctid, xmin, xmax, cmin,
  - Сжатие
    - Без сжатия
    - Quicklz
    - Zlib уровни 1 - 9

Формат хранения

- Apache Parquet
  - Поколоночное хранение блоков последовательности строк (“row group”)



Формат хранения

- Apache Parquet
  - Поколоночное хранение блоков последовательности строк (“row group”)
  - Сжатие
    - Без сжатия
    - Snappy
    - Gzip уровни 1 – 9
  - Размер «row group» и страницы задается для каждой таблицы отдельно

Внешние данные

- PXF (Pivotal Extension Framework)
  - Фреймворк для доступа к внешним данным
  - Легко расширяется, возможно использовать свои плагины
  - Официальные плагины: CSV, SeqFile, Avro, Hive, Hbase
  - Open Source плагины: JSON, Accumolo, Cassandra, JDBC, Redis, Pipe
- HCatalog
  - HAWQ «видит» таблицы из Hcatalog как свои внутренние таблицы

Разделение ресурсов

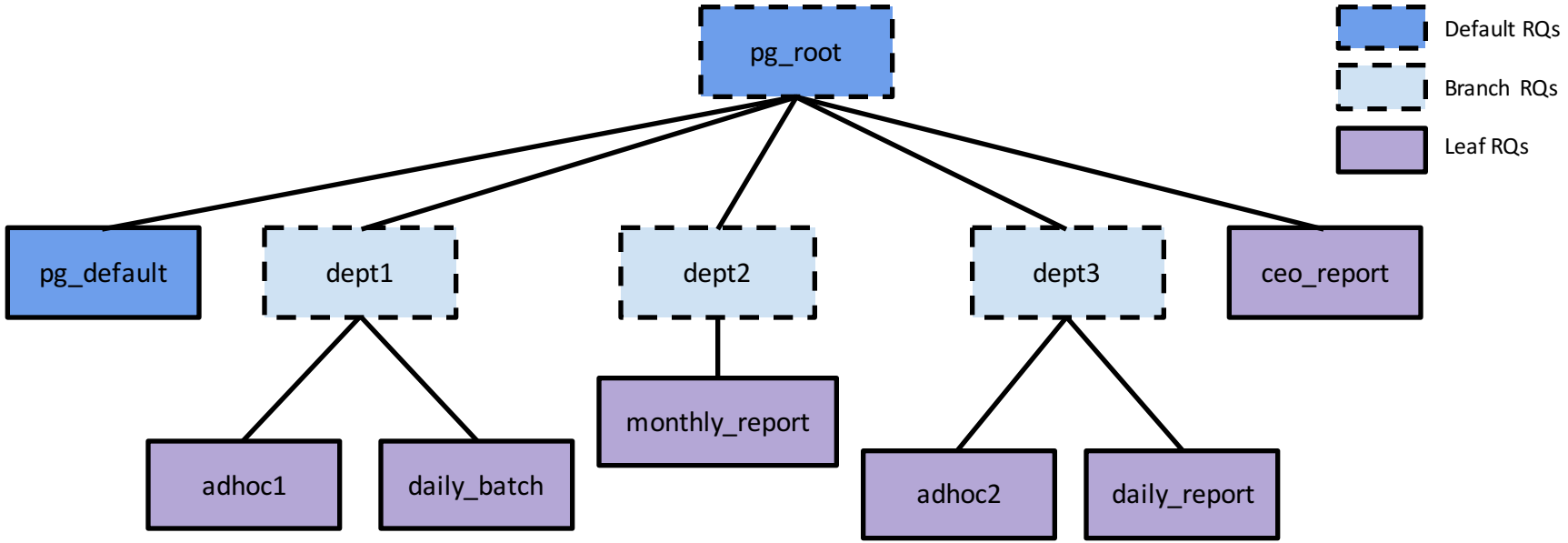
Разделение ресурсов

- Два основных варианта
  - Независимое разделение HAWQ и YARN не знают друг о друге
  - HAWQ запрашивает у YARN выделение ресурсов через YARN RM
- Гибкая утилизация кластера
  - Запрос может выполняться на части узлов
  - Запрос может иметь несколько потоков исполнения на каждом узле
  - Желаемый параллелизм можно задать в ручную

Разделение ресурсов

- Пулы ресурсов (Resource Queues) задают
  - Количество параллельных запросов
  - Приоритет использования CPU
  - Лимит по памяти
  - Лимит по ядрам CPU
  - MIN/MAX потоков исполнения в целом по системе
  - MIN/MAX потоков исполнения на каждом узле
- Задаются по пользователю или группе

Иерархические ресурсные очереди

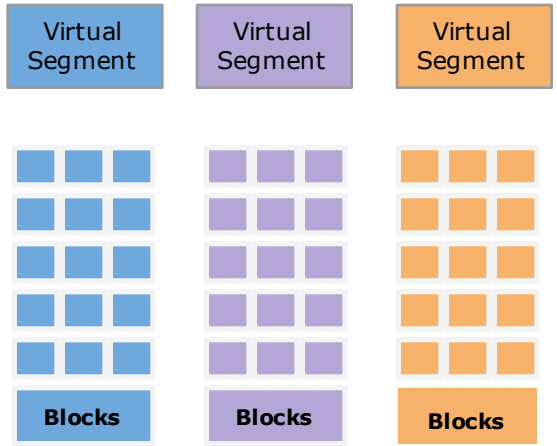


Выполнение запросов

Выполнение запросов

- Выполнение запросов является динамическим и гибким процессом
  - Возможно Scale-up/down масштабирование
  - Возможно Scale-in/out масштабирование

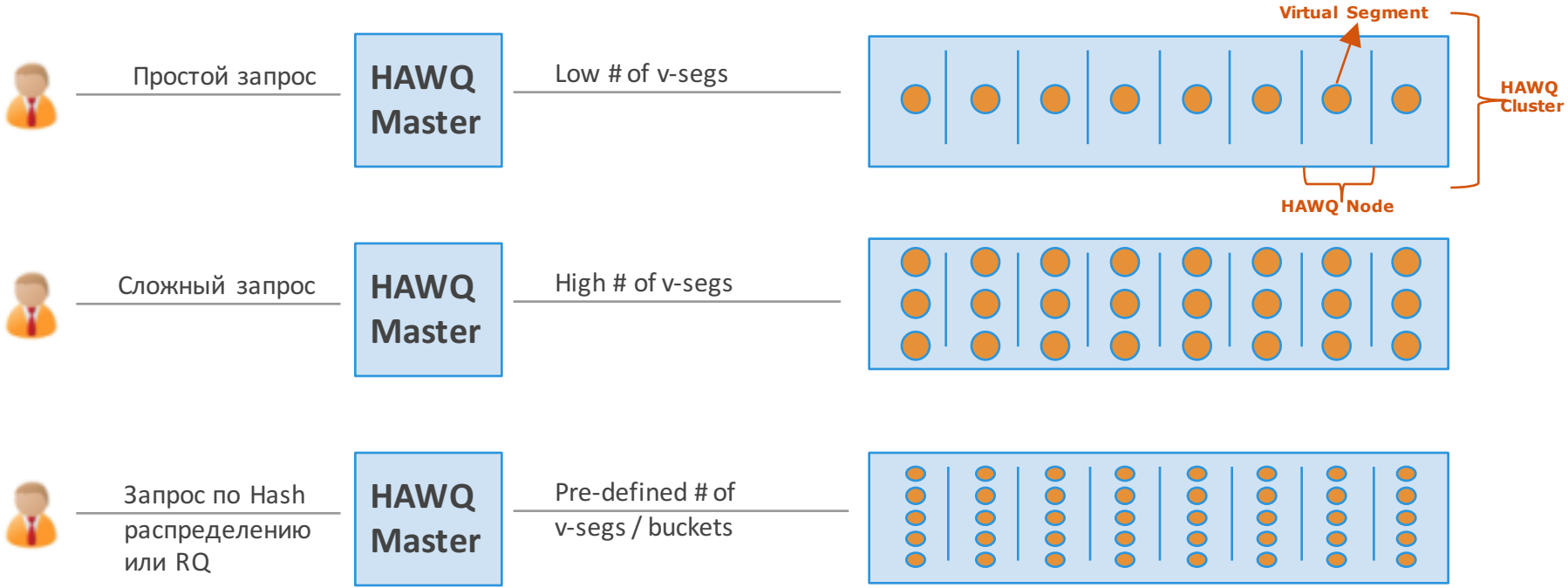
- Использование: “block level storage” и “virtual segments”
  - Поддержка использования блоков данных
  - Использование узлов хранящих требуемые блоки
  - Запуск запросов на узлах, обладающих требуемыми свободными ресурсами
  - Запуск ”virtual segment” по запросу



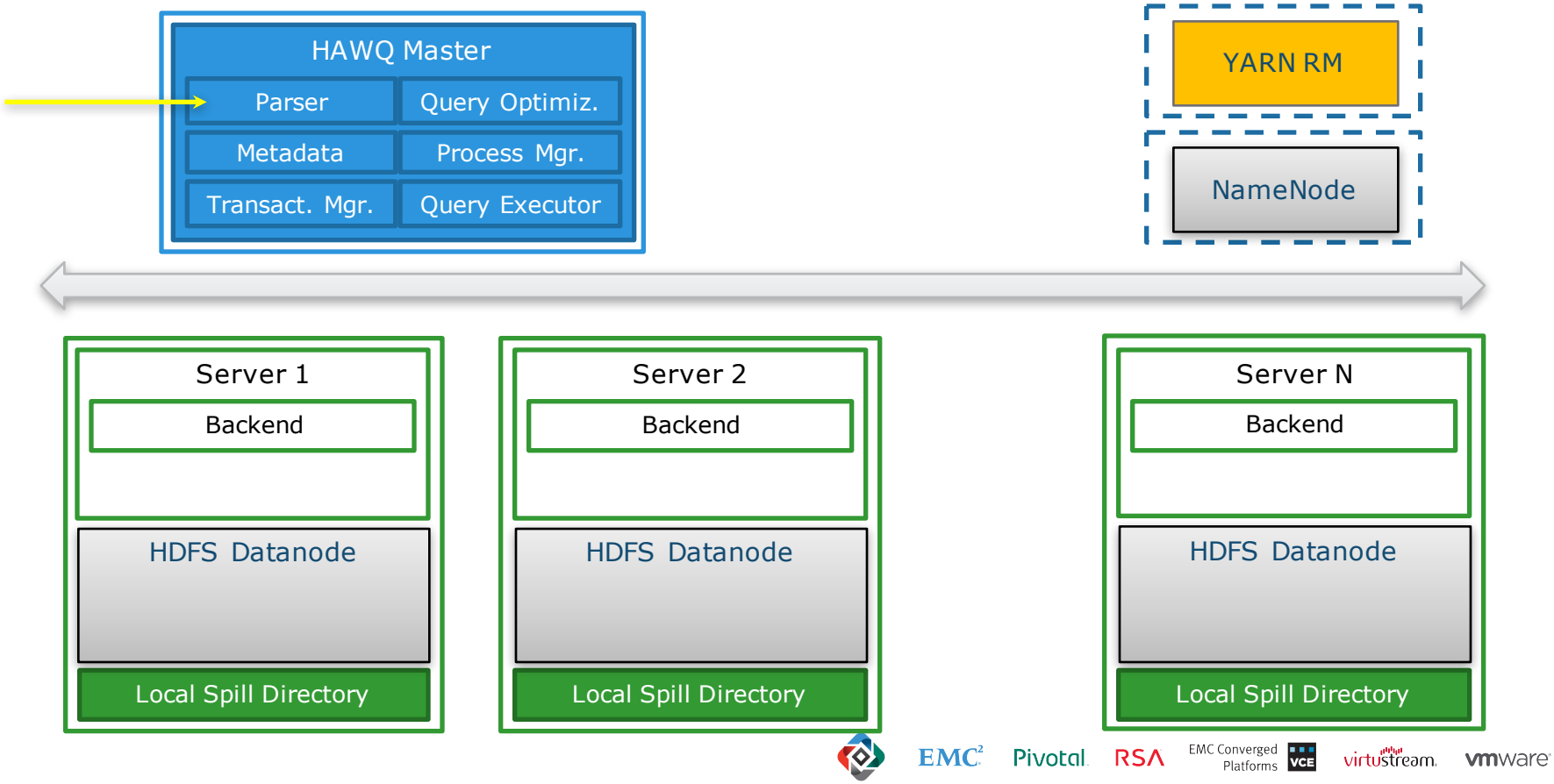
Выполнение запросов

- Каждый узел имеет только один логический сегмент HAWQ
  - Простота в настройке и установке
  - Простата в оптимизации производительности
- Количество “virtual segments” используемых для запроса
  - Определяет параллелизм запроса
  - Каждый ”virtual segment” равен одному процессу Postgres
- Как определить количество ”virtual segment” необходимых для выполнения запроса
  - Объем ресурсов необходимых для выполнения запроса
  - «Стоимость» запроса, Data locality, RQ Definitions
  - Распределение таблицы: Hash vs Random
  - UDFs и external tables

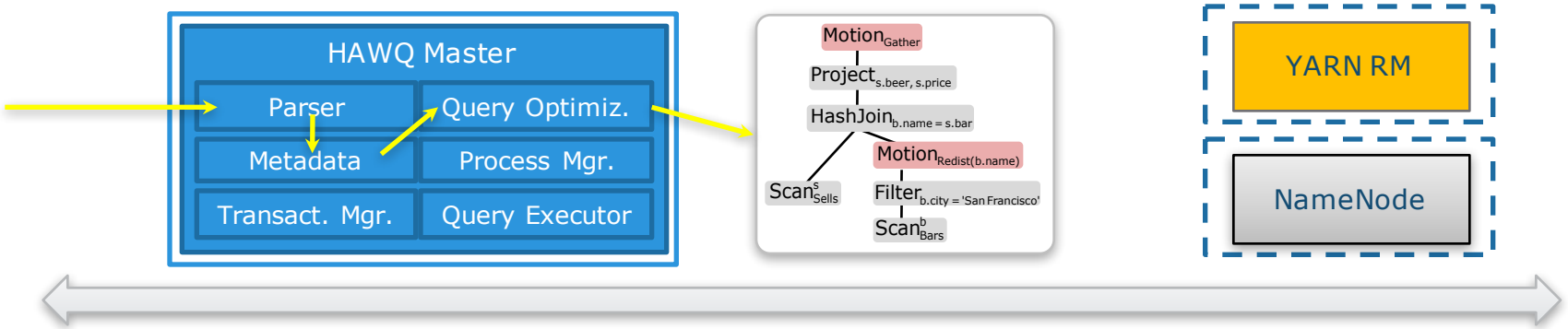
Выполнение запросов

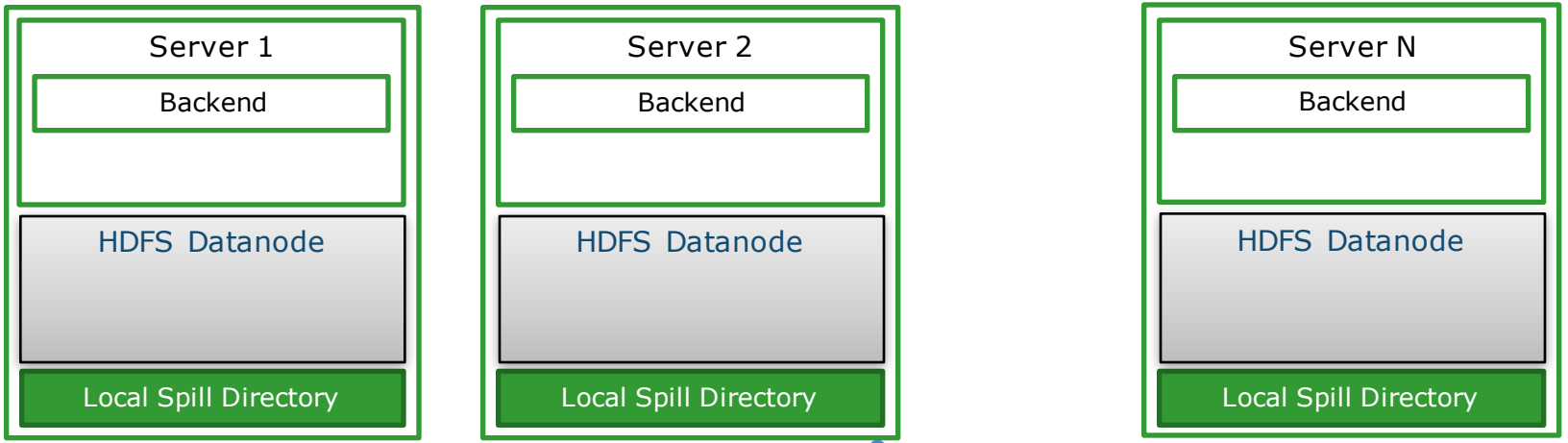


Выполнение запроса

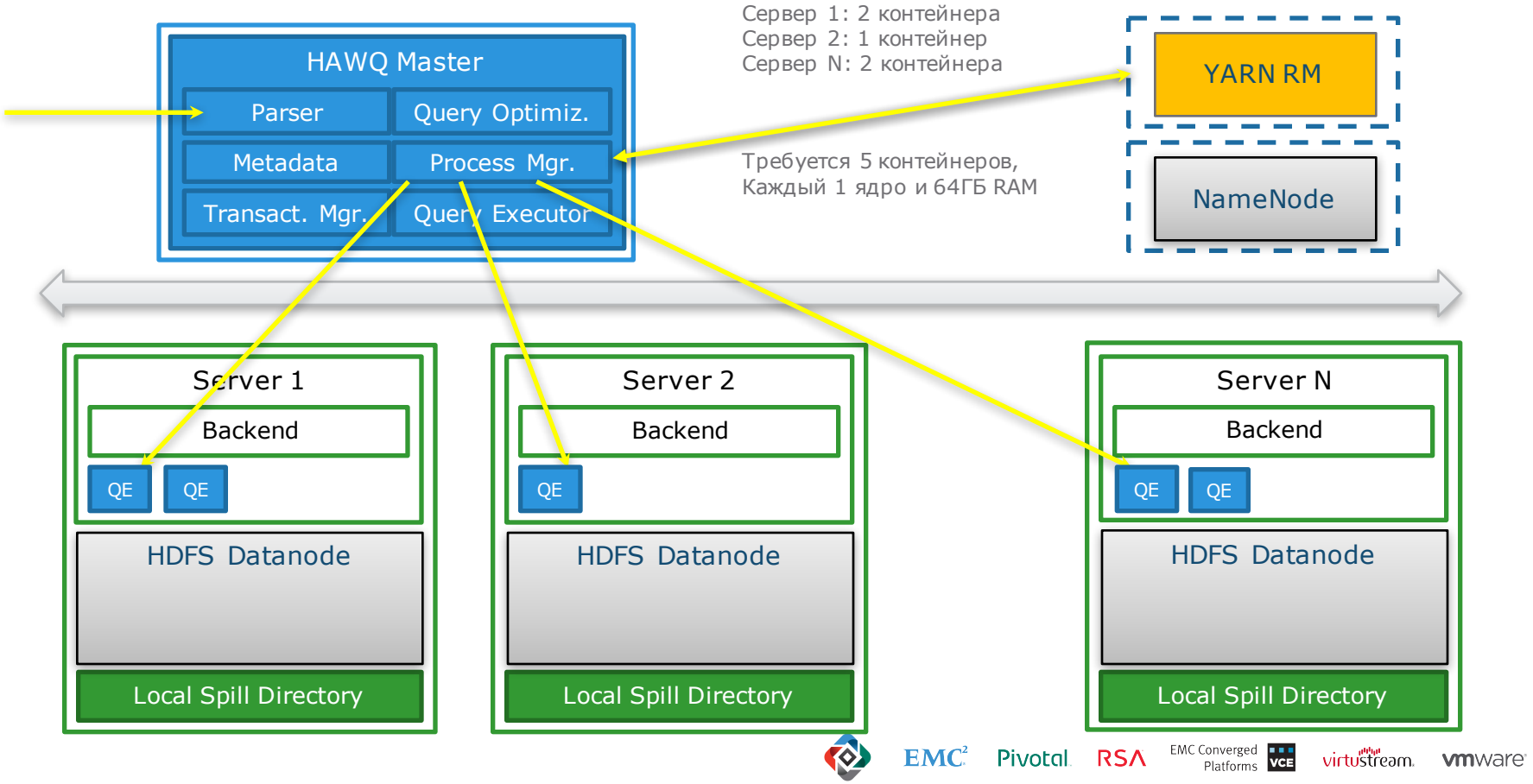


Выполнение запроса

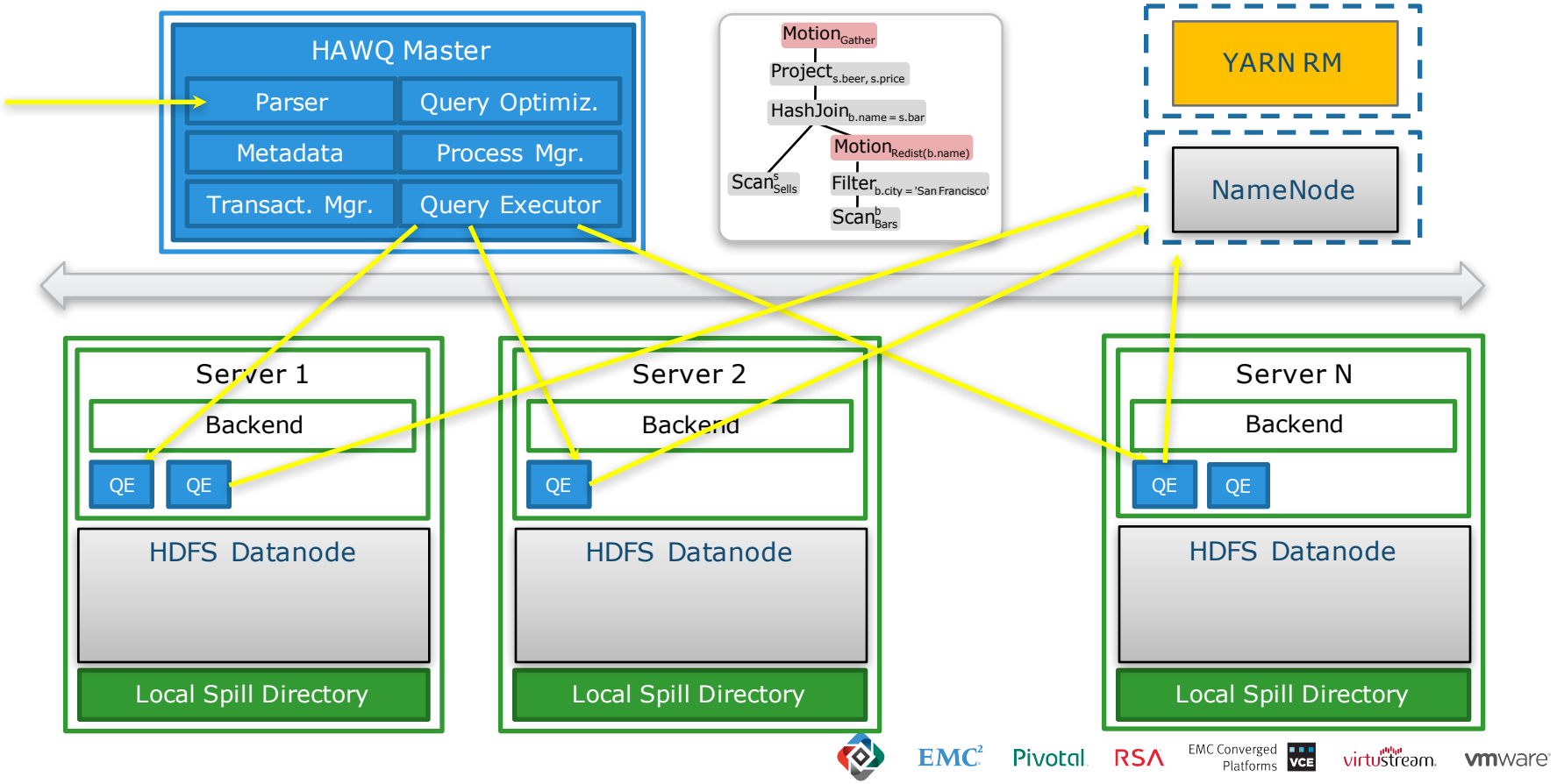




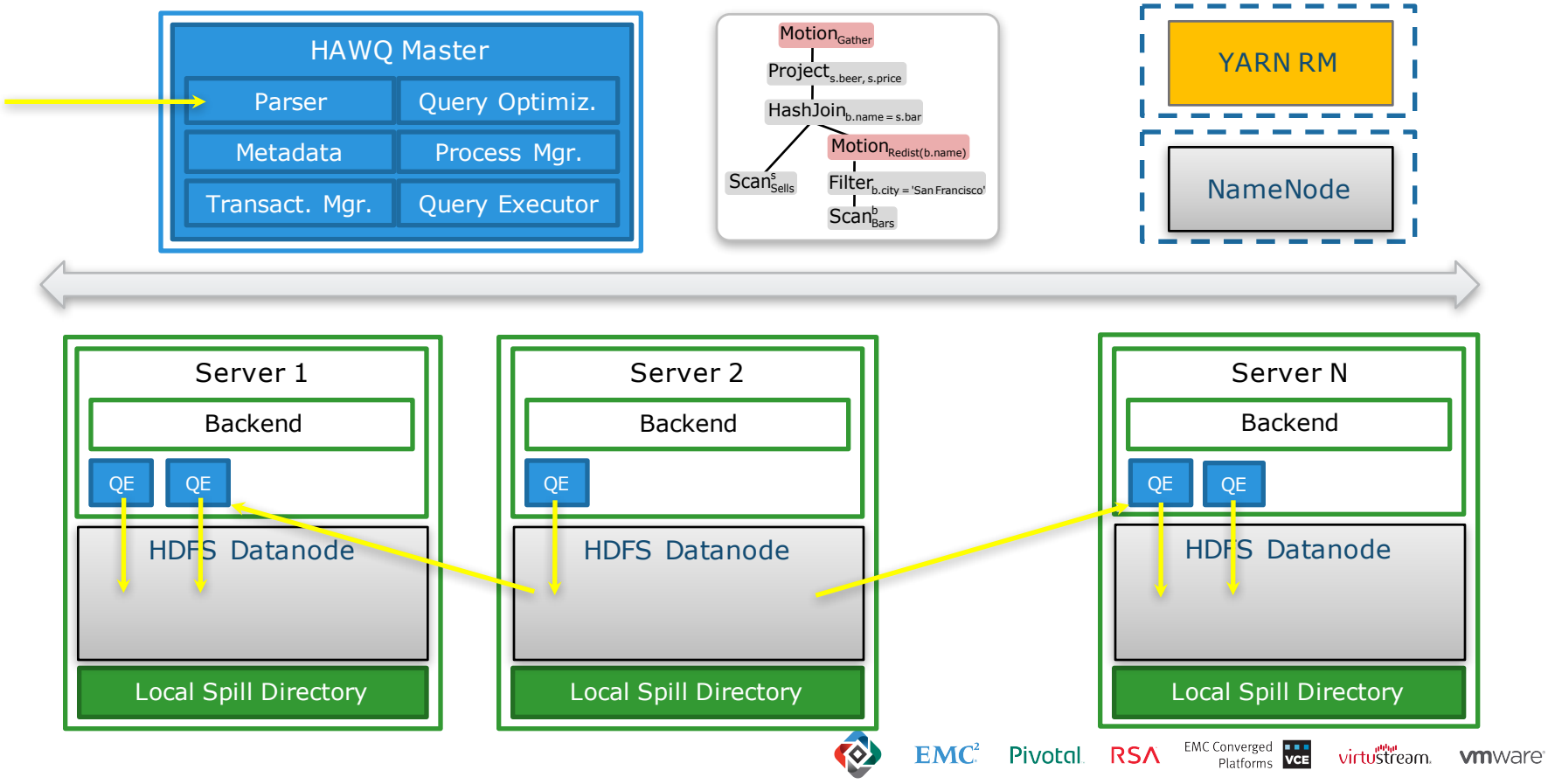
Выполнение запроса



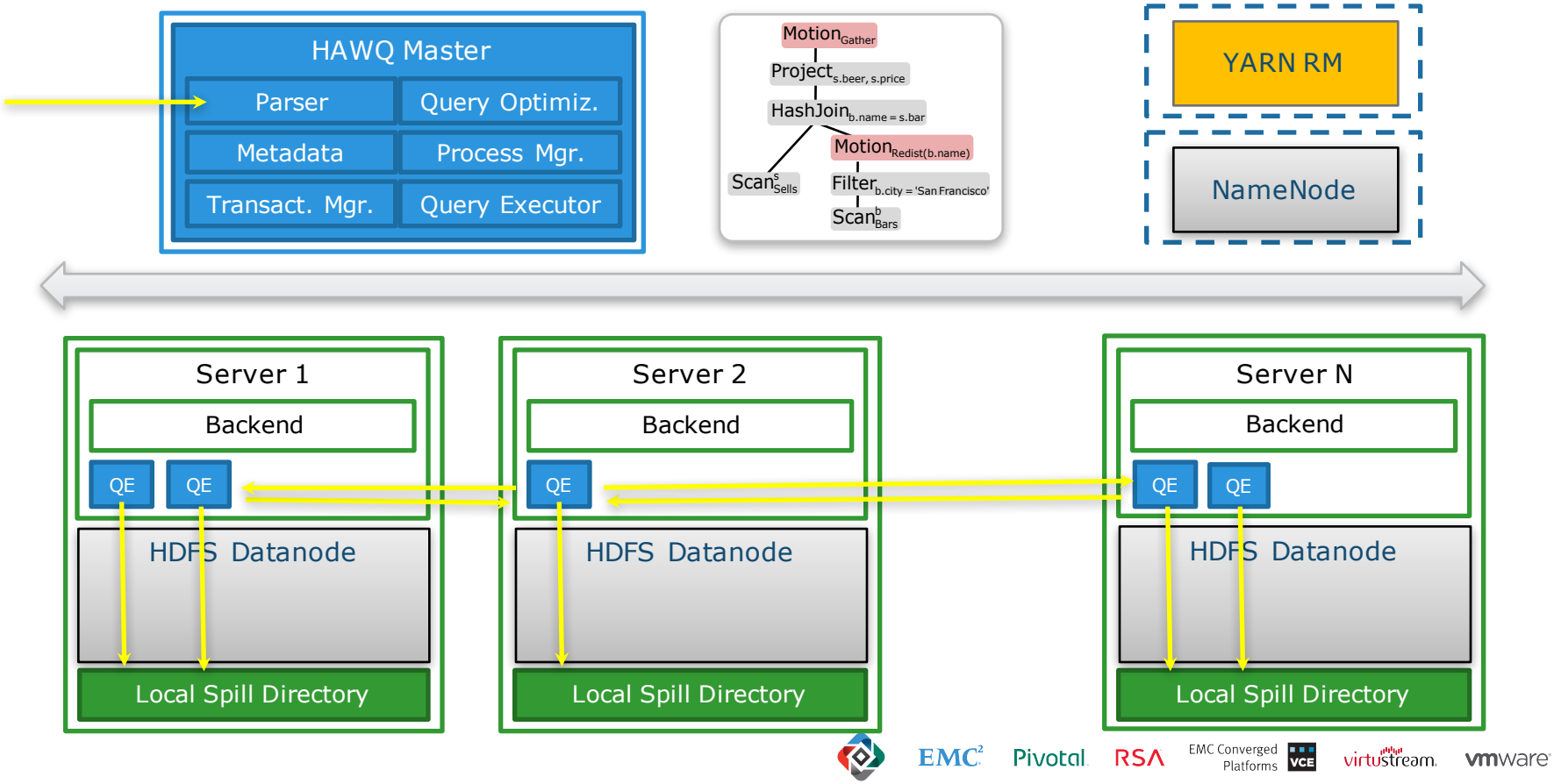
Выполнение запроса



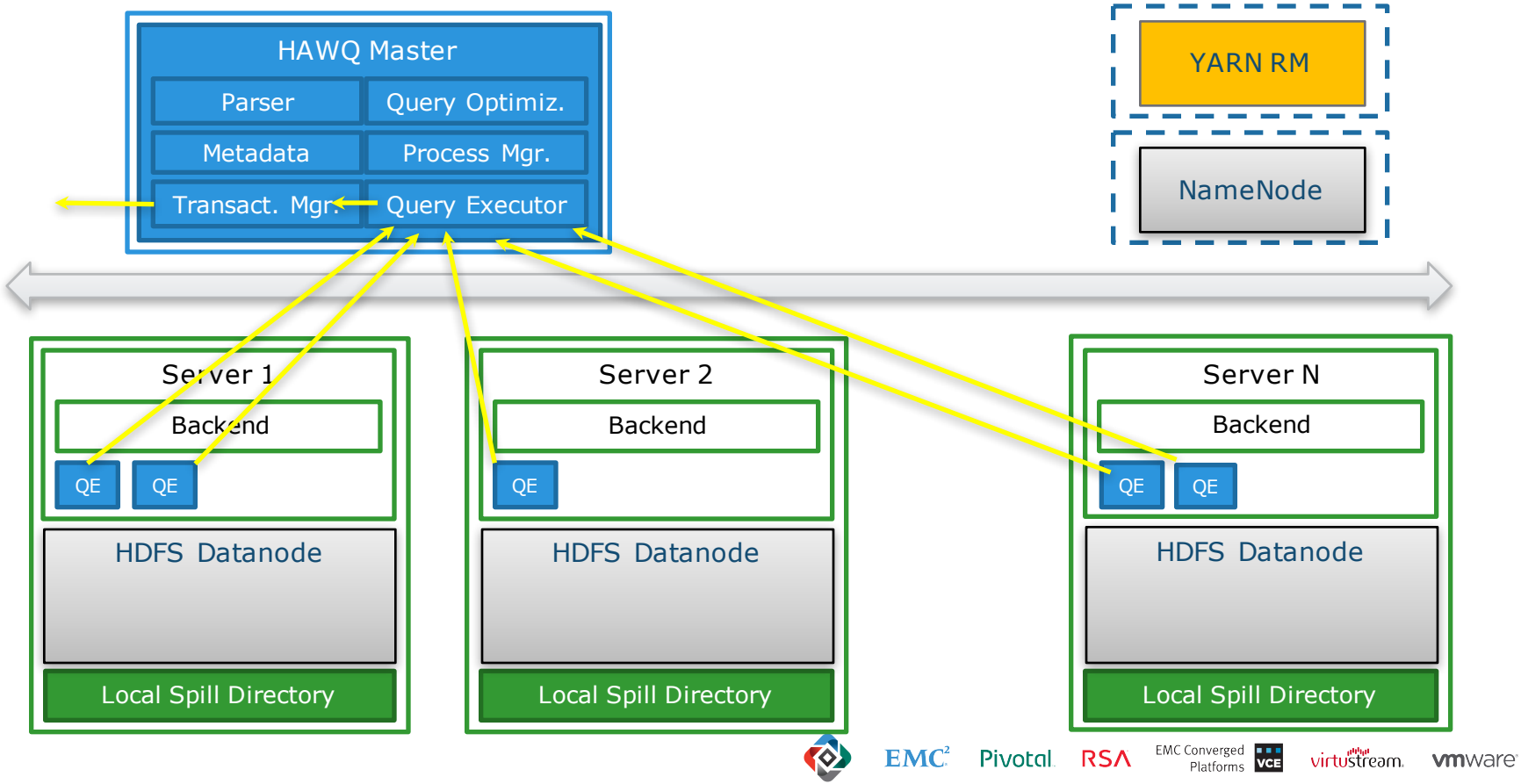
Выполнение запроса



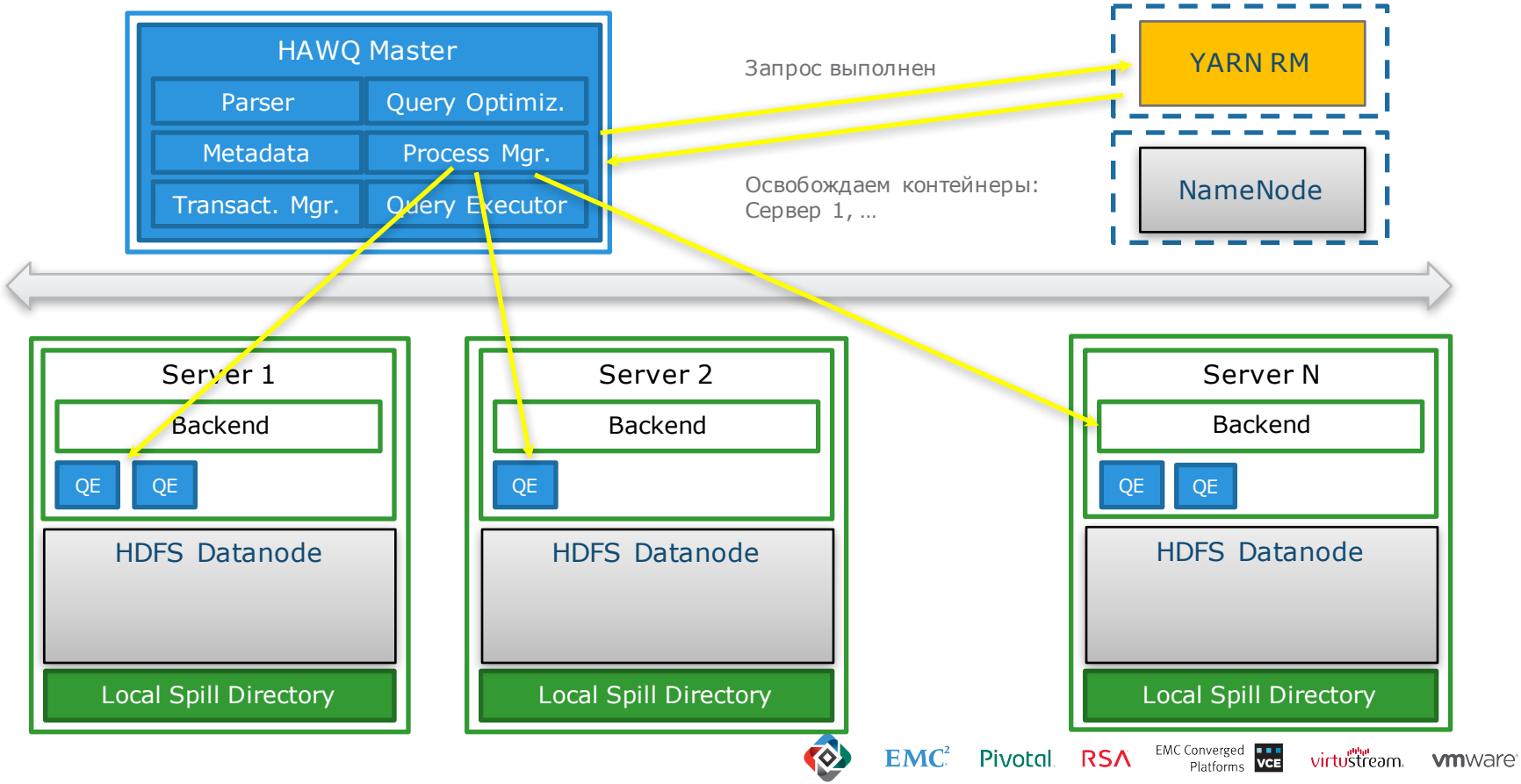
Выполнение запроса



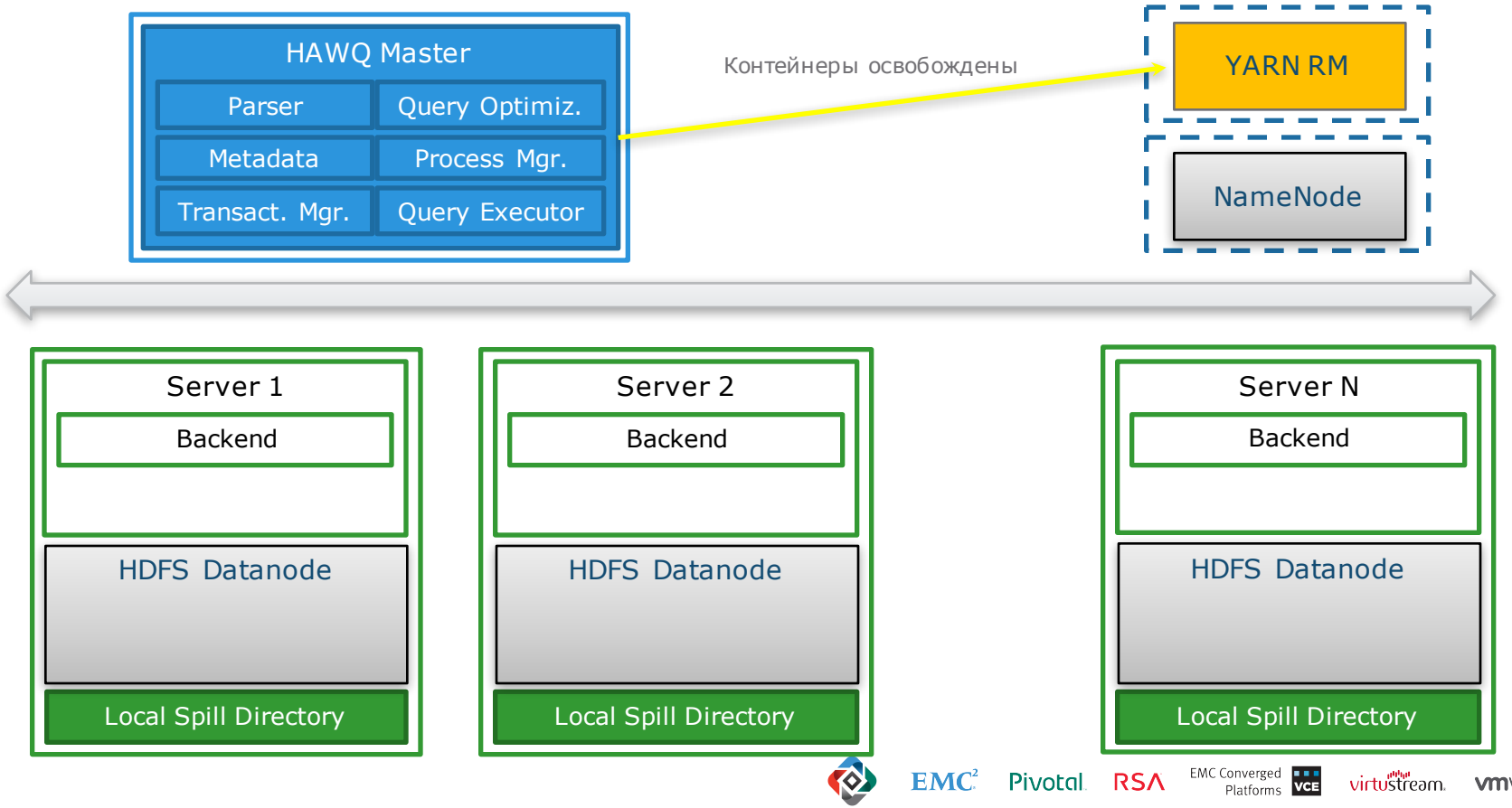
Выполнение запроса



Выполнение запроса



Выполнение запроса



Итого...

- Данные не сохраняются на диске без необходимости
- Данные не буферизируются на сегментах без необходимости
- Данные передаются между узлами по протоколу UDP
- Отличный cost-based оптимизатор
- Быстрый код написанный на C++
- Гибкая настройка параллелизма

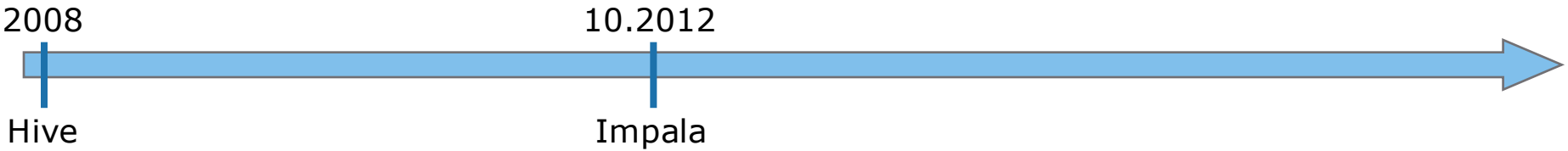
EMC² Pivotal RSA EMC Converged Platforms VCE virtustream vmware

Альтернативные решения

Решения SQL-on-Hadoop

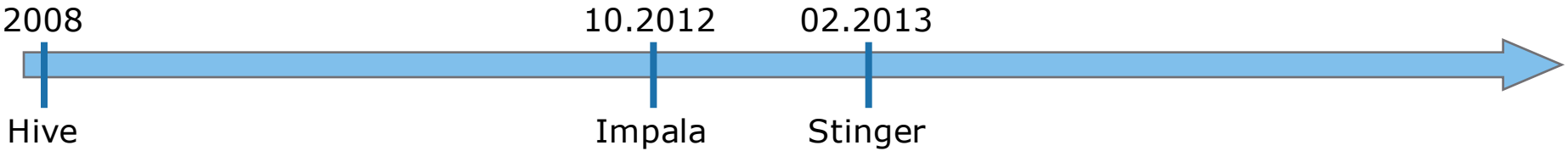
- Разработка Facebook
  - Используется для анализа данных в их хранилище
  - Хранилище имеет размер ~300PB, ежедневно загружается ~600TB данных. Данные сжаты ORCFile, коэффициент ~8x
- Синтаксис HiveQL не соответствует ANSI SQL-92
- Множество ограничений на подзапросы
- Cost-based оптимизатор (Optiq) пока в стадии technical preview

Решения SQL-on-Hadoop



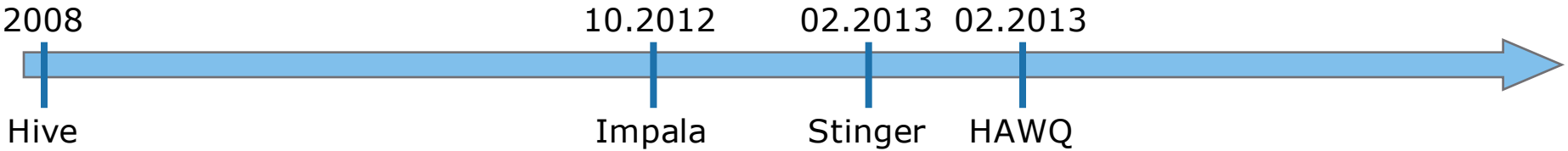
- Разработка Cloudera
  - Open-source решение
  - Cloudera продает поддержку этого решения корпоративным клиентам
  - До мая 2013 находилась в стадии бэта
- Поддержка HiveQL, развитие в сторону поддержки ANSI SQL-92
- Написана на C++ и не использует Map-Reduce для исполнения
- Требует много памяти, Join больших таблиц часто приводит к OOM

Решения SQL-on-Hadoop



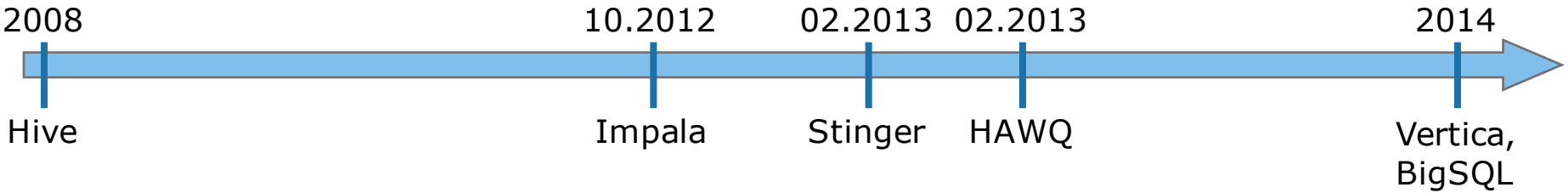
- Разработка Hortonworks
  - Содержит в себе шаги по улучшению производительности Hive
- Tez – выполнение сгенерированного Hive кода без приземления промежуточных данных на диск между независимыми map-reduce
- Optiq – cost-based оптимизатор запросов (пока technical preview)
- ORCFile – адаптивное сжатие данных и встроенный индекс
- Hive-5317 – ACID и поддержка update/delete (релиз через ~2 месяца)

Решения SQL-on-Hadoop



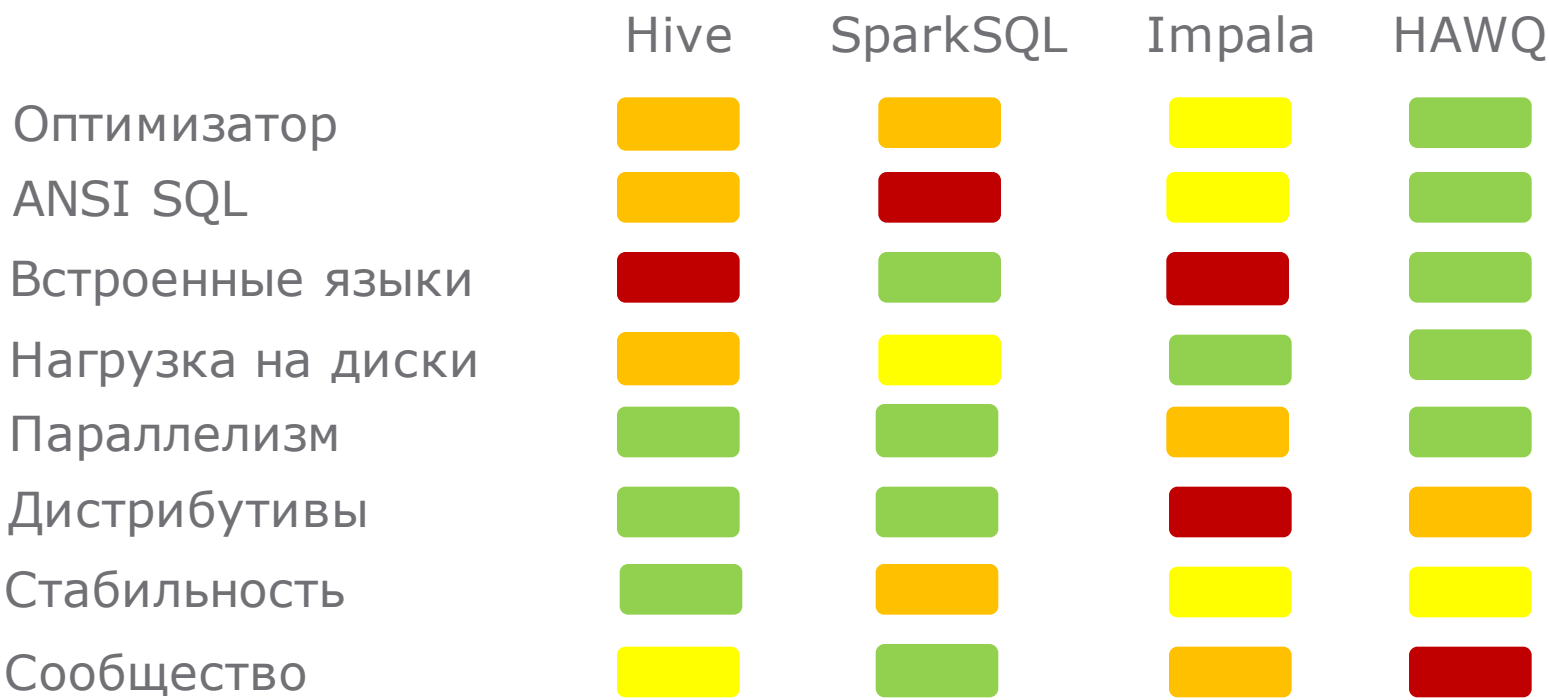
- Разработка Pivotal
  - СУБД Greenplum, портированная на HDFS
  - Написан на C, оптимизатор запросов переписан для этого решения
- Поддержка ANSI SQL-92, аналитических расширений SQL-2003
- Поддержка сложных запросов с корелированными подзапросами, сложными оконными функциями и различными join'ами
- Данные приземляются на диск только в случае нехватки памяти

Решения SQL-on-Hadoop



- HP Vertica
  - Поддержка ANSI SQL-92, SQL-2003
  - Поддержка UPDATE/DELETE
- IBM BigSQL v3
  - IBM DB2, портированная на HDFS
  - Федеративные запросы, полноценный оптимизатор запросов, etc.

Сравнение



Планы развития

Планы развития

- Интеграция с AWS и S3
- Интеграция с Mesos
- Улучшенная интеграция с Ambari и Ambari Metrics
- Полноценная поддержка GPText, MADLib
- Реализация для Cloudera, MapR, IBM
- Развитие комьюнити

EMC² Pivotal RSA EMC Converged Platforms VCE virtustream vmware