Формализация информации и Big Data

«02.03.03 - Математическое обеспечение и администрирование информационных систем направленность разработка и администрирование информационных систем»

http//vikchas.ru

Лекция 11 «СУБД Greenplum для Big Data цифровой экономике»

Часовских Виктор Петрович
д-р техн. наук, профессор кафедры ШИиКМ

ФГБОУ ВО «Уральский государственный экономический университет»

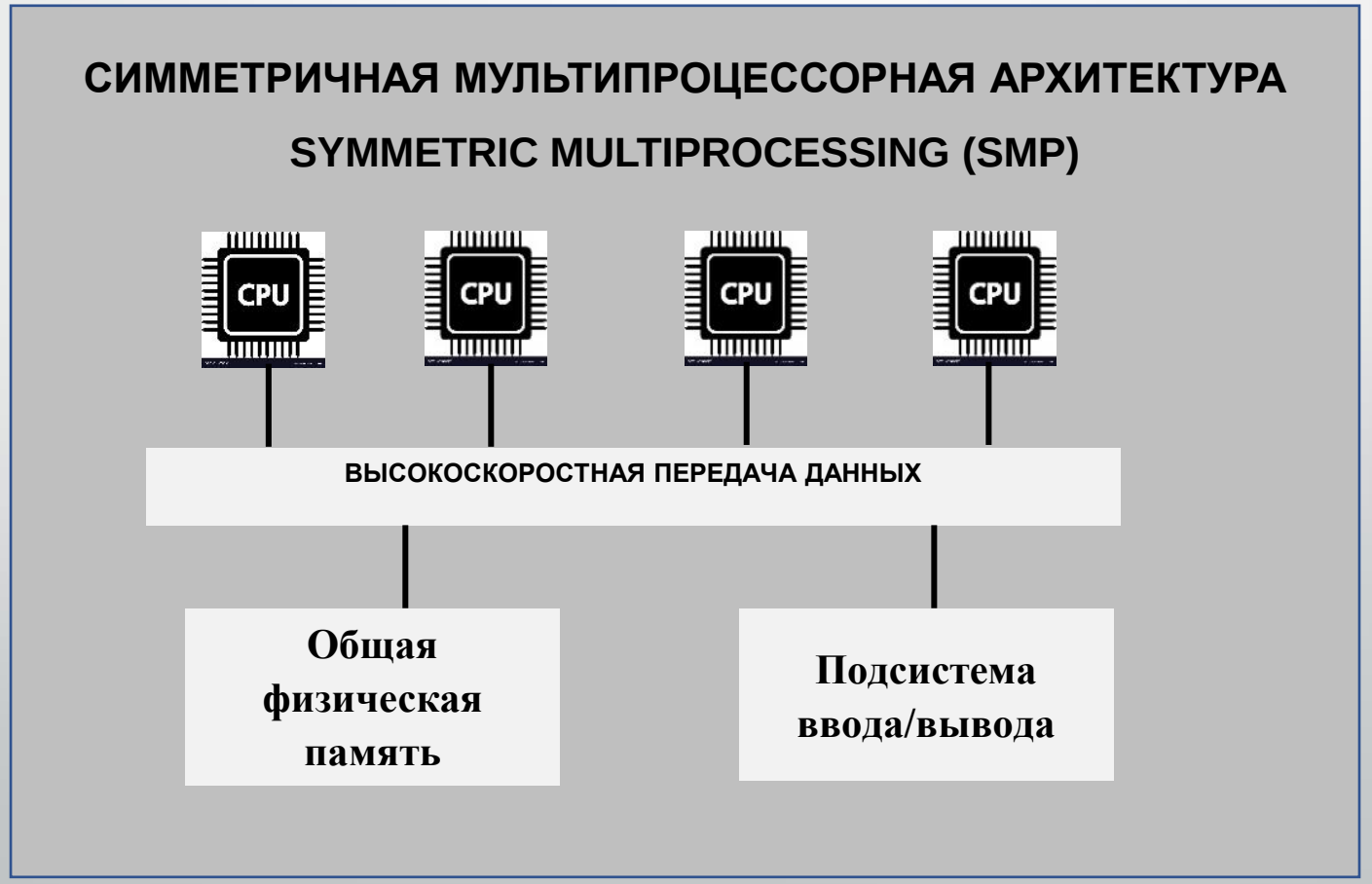
Екатеринбург 2024

Основные сведения о СУБД Greenplum

СУБД Greenplum – open-source продукт, массивно-параллельная реляционная СУБД для хранилищ данных с гибкой горизонтальной масштабируемостью и столбцовым хранением данных на основе PostgreSQL. Благодаря своим архитектурным особенностям и мощному оптимизатору запросов, СУБД Greenplum отличается особой надежностью и высокой скоростью обработки SQL-запросов над большими объемами данных, поэтому эта MPP-СУБД широко применяется для аналитики Big Data в промышленных масштабах.

Эксплуатация системы началась в 2005 г. фирмой США «Greenplum». В 2018 году компания РФ «Arenadata» разработчик первого отечественного дистрибутива Apache Hadoop (программный проект с открытым исходным кодом, предназначенный для эффективной обработки больших пакетов данных), выпустила собственную MPP-СУБД Arenadata DB на основе Greenplum, адаптировав ее для корпоративного использования.

До недавнего времени распространённой структурой системами для аналитической обработки больших объемов данных являются SPM (symmetric multiprocessing) или системы симметричные мультипроцессорной архитектуры



Главная особенностью систем с такой архитектурой является наличие общих физических ресурсов, которые разделяются между несколькими процессорами сравнимой производительности.

Память служит, в частности, для передачи сообщений между процессорами и при этом все вычислительные устройства при обращении к памяти имеют равные права поэтому структура называется симметричной.

Большинство популярных СУБД таких как Oracle, Sybase и PostgreSQL , реализованы именно в такой архитектуре.

PostgreSQL принято обозначать 

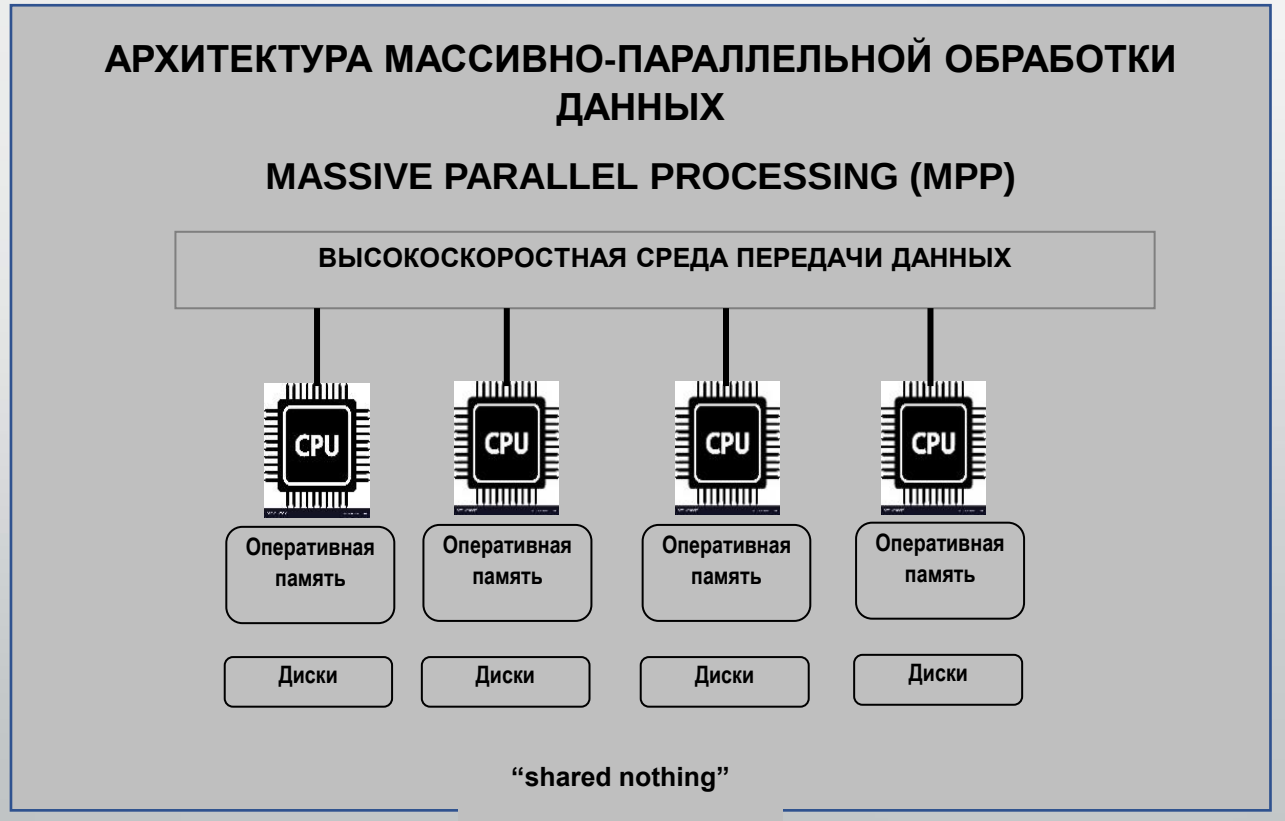
SMP - системы обладают рядом преимуществ, таких как высокая скорость обмена данными между процессорами за счет наличия общей памяти, простота и универсальность обслуживания и относительно невысокая цена.

Существенным недостатком таких систем является плохая масштабируемость.

Каждый раз, когда необходимо увеличить скорость обработки данных, нам необходимо увеличивать мощность сервера за счет увеличения числа процессоров.

Как правило, данная операция является дорогостоящей, а в некоторых случаях и вовсе невозможно из-за физических ограничений в конфигурации сервера.

Для решения проблемы масштабируемости в системах аналитической обработки данных стали применять архитектуру MPP, или архитектуру массивно-параллельной обработки данных



В такой архитектуре система состоит из нескольких независимых узлов, соединенных по сети. При этом в каждом вычислительном узле процессор обладает своими собственными физическими ресурсами, такими как память и диски, которые не разделяются с другими узлами. Именно поэтому такая архитектура также называется Shared nothing (Ничего общего).

В MPP -системах вычислительная мощность и объем хранения данных увеличиваются за счет добавления в систему дополнительных вычислительных узлов.

Данный подход позволяет достигнуть линейный рост производительности и в зависимости от количества узлов в системе.

СУБД Greenplum переназначена до хранения и обработки больших объемов данных методом распределения данных и обработки запросов на нескольких серверах.

Данная СУБД лучше всего подходит для построения корпоративных хранилищ данных, решение аналитических задач и задач машинного обучения и искусственного интеллекта.

В основе СУБД Greenplum лежит СУБД PostgreSQL. По сути СУБД Greenplum представляет из себя множество модифицированных экземпляров дисковых баз данных PostgreSQL, работающих совместно как одна связанная система управления базами данных.

В большинстве случаев похож на PostgreSQL, например, в части синтаксиса SQL, функции параметров, конфигурации и функциональности для конечного пользователя. Пользователи базы данных взаимодействуют с СУБД Greenplum также как с обычной СУБД PostgreSQL.

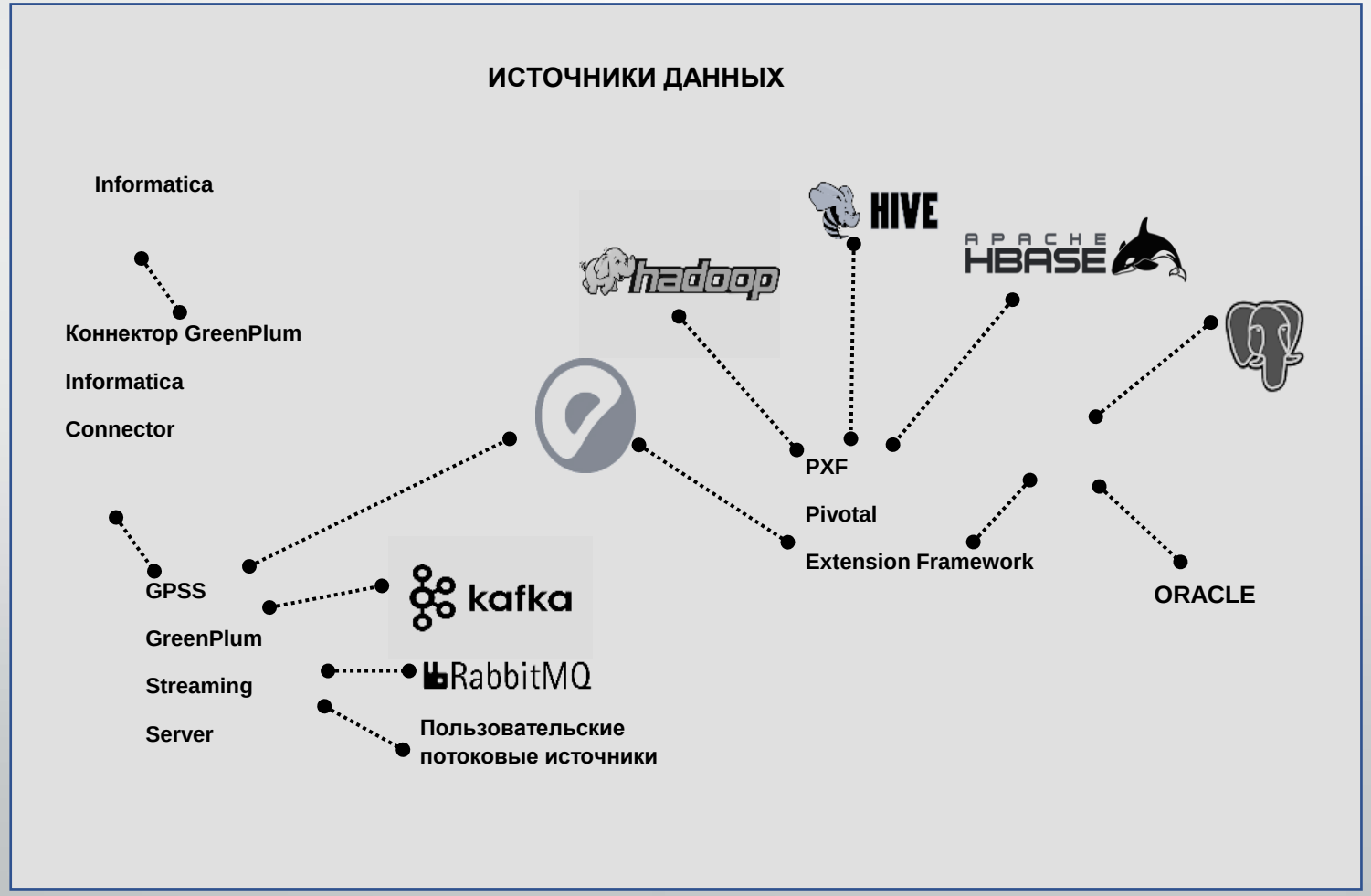
Внутреннее устройство СУБД PostgreSQL было изменено или дополнено для поддержки параллельной структуры баз данных СУБД Greenplum.

Например, компоненты системного каталога, оптимизатора, исполнителя запросов и диспетчера транзакции были изменены и улучшены, чтобы они могли выполнять запросы одновременно в параллель во всех экземплярах базы данных.

СУБД Greenplum является исключительно программным решением. Он поставляется в виде дистрибутива, который может быть установлен на различные серверные платформы. Производительность напрямую зависит от оборудования, на котором он установлен.

СУБД Greenplum предоставляет широкий выбор инструментов для решения аналитических задач

СУБД Greenplum обладает большим количеством различных коннекторов, которые предоставляют доступ к другим источникам данных, таким как другие СУБД-компоненты экосистемы Hadoop и стриминговые источники данных, например, Кафка



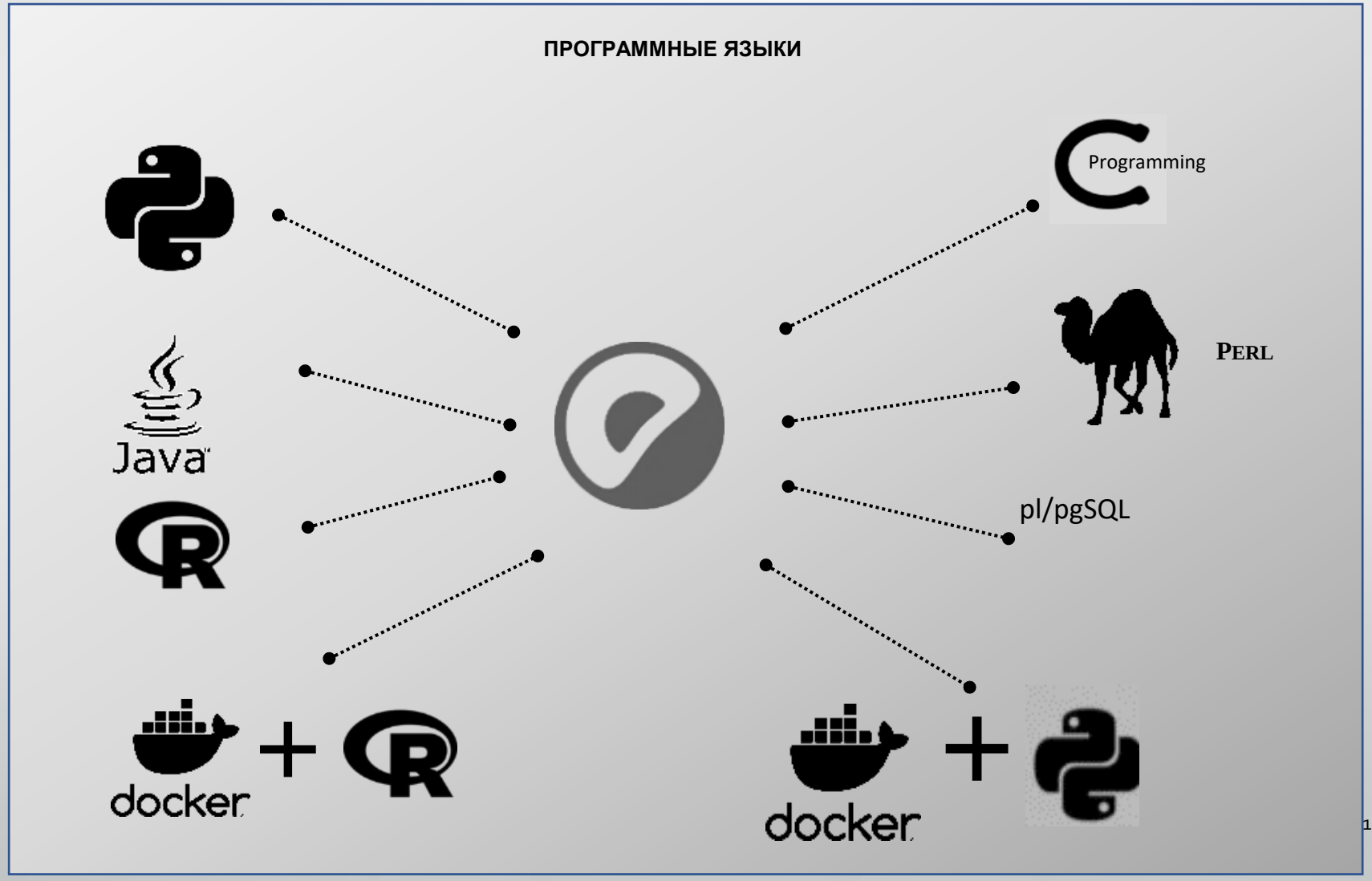
Hadoop — проект фонда Apache Software Foundation, свободно распространяемый набор утилит, библиотек и фреймворк для разработки и выполнения распределённых программ, работающих на кластерах, из сотен и тысяч узлов.

Также в СУБД Greenplum можно разрабатывать хранимые процедуры не только на встроенном процедурном языке SQL, но и на многих других популярных языках программирования, которые будут компилироваться и выполняться внутри базы данных.

Также есть возможность разрабатывать процедуры, которые будут работать в контейнерах.

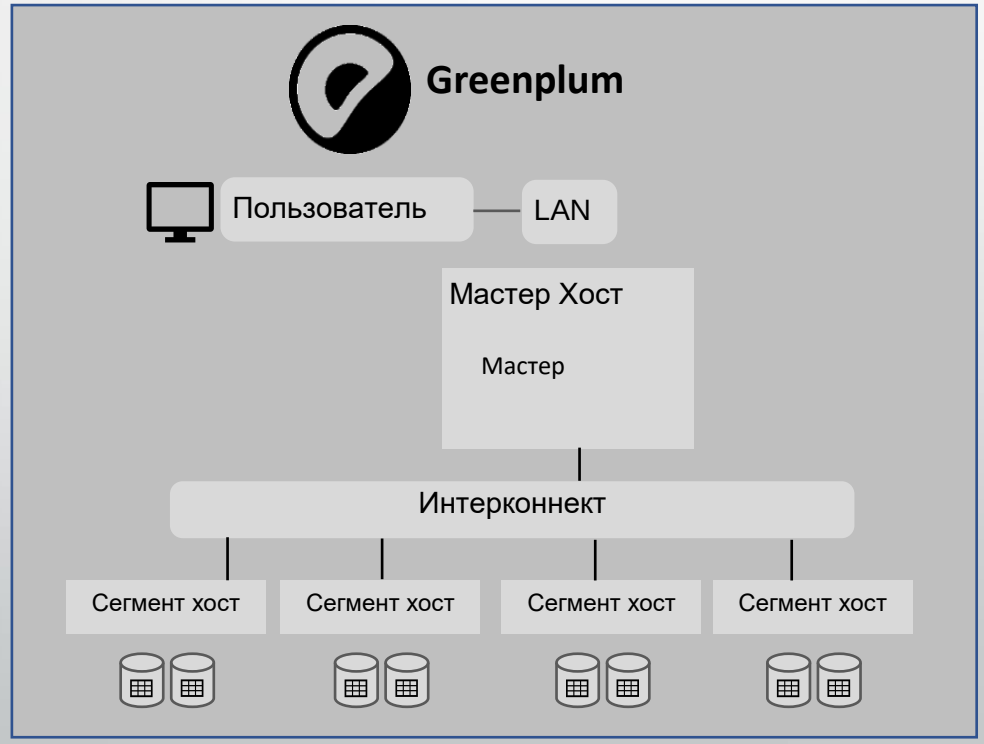
Такой подход позволяет сделать разработку безопасной путем изолирования исполняемого кода от операционной системы, на которой установлена СУБД.

Языки программирования, поддерживаемые СУБД Greenplum



Архитектура СУБД Greenplum

СУБД Greenplum представляет из себя множество модифицированных баз данных PostgreSQL, которые расположены на нескольких серверах и взаимодействуют друг с другом, образуя единую, цельную базу данных



Логическая архитектура СУБД Greenplum выглядит следующим образом:

1. **Мастер** — это входная точка для СУБД Greenplum — это экземпляр базы данных СУБД PostgreSQL, к которому клиенты подключаются и отправляют свои SQL-запросы. Мастер координирует работу с другими экземплярами базы данных системы, которые называются «сегментами». Пользователи взаимодействуют с мастером также, как если бы это была обычная база данных СУБД PostgreSQL. Подключение может выполняться при помощи клиентских программ, таких как:

PSQL (SQL СУБД PostgreSQL);

JDBC (Java DataBase Connectivity — соединение с базами данных на Java);

ODBC (Open Database Connectivity - программный интерфейс (API) доступа к базам данных, разработанный компанией Microsoft).

На **мастере** располагается глобальный системный каталог — это набор системных таблиц, которые содержат метаданные о всех объектах базы данных СУБД Greenplum.

Важно отметить, что мастер не хранит какие-либо пользовательские данные, все пользовательские данные хранятся на сегментах. Функции мастера заключаются в следующем: мастер выполняет авторизацию клиентских соединений, обрабатывает входящие SQL-команды, распределяет нагрузку между сегментами, координирует результаты, возвращаемое каждым сегментом, выполняет финальные операции над данными предоставляет конечный результат клиенту.

2. Сегменты — это отдельные экземпляры базы данных СУБД PostgreSQL, которые хранят определенную порцию пользовательских данных и выполняют большую часть обработки запросов.

Когда пользователь подключается к базе данных через мастер и запускает запрос, на каждом сегменте создаются процессы для обработки этого запроса. Затем каждый сегмент параллельно обрабатывает свою порцию данных и возвращает финальный результат своей работы на мастер-сегменты работают на серверах, которые называются сегмент-хосты или ноды.

На каждом ноде обычно располагается от двух до восьми сегментов Greenplum. Количество сегментов конфигурируется при первоначальной установке системы и напрямую зависит от профиля рабочих нагрузок системы и технических характеристик сервера, таких как количество и производительность ядер процессора, объема оперативной и постоянной памяти и сетевых интерфейсов.

Предполагается, что все ноды(*Нода (узел) сети — это обычный компьютер, на котором запущена и постоянно работает определенная программа*) системы будут настроены идентично ключом к достижению максимальной производительности от СУБД Greenplum, является равномерное распределение данных и рабочих нагрузок по большому количеству сегментов с одинаковой производительностью. Это необходимо для того, чтобы все сегменты одновременно начинали работать над задачей и одновременно завершали свою работу. В этом случае ни один из узлов не станет узким горлышком всей системы.

3.Интерконнект это сетевой слой архитектуры СУБД Greenplum. Интерконнект обеспечивает взаимосвязь между мастером-сегментами и сетевой инфраструктурой, в которой развернут кластер.

Для обеспечения высокой производительности системы рекомендуемая пропускная способность сети должна быть не менее 10.0 гигабит.

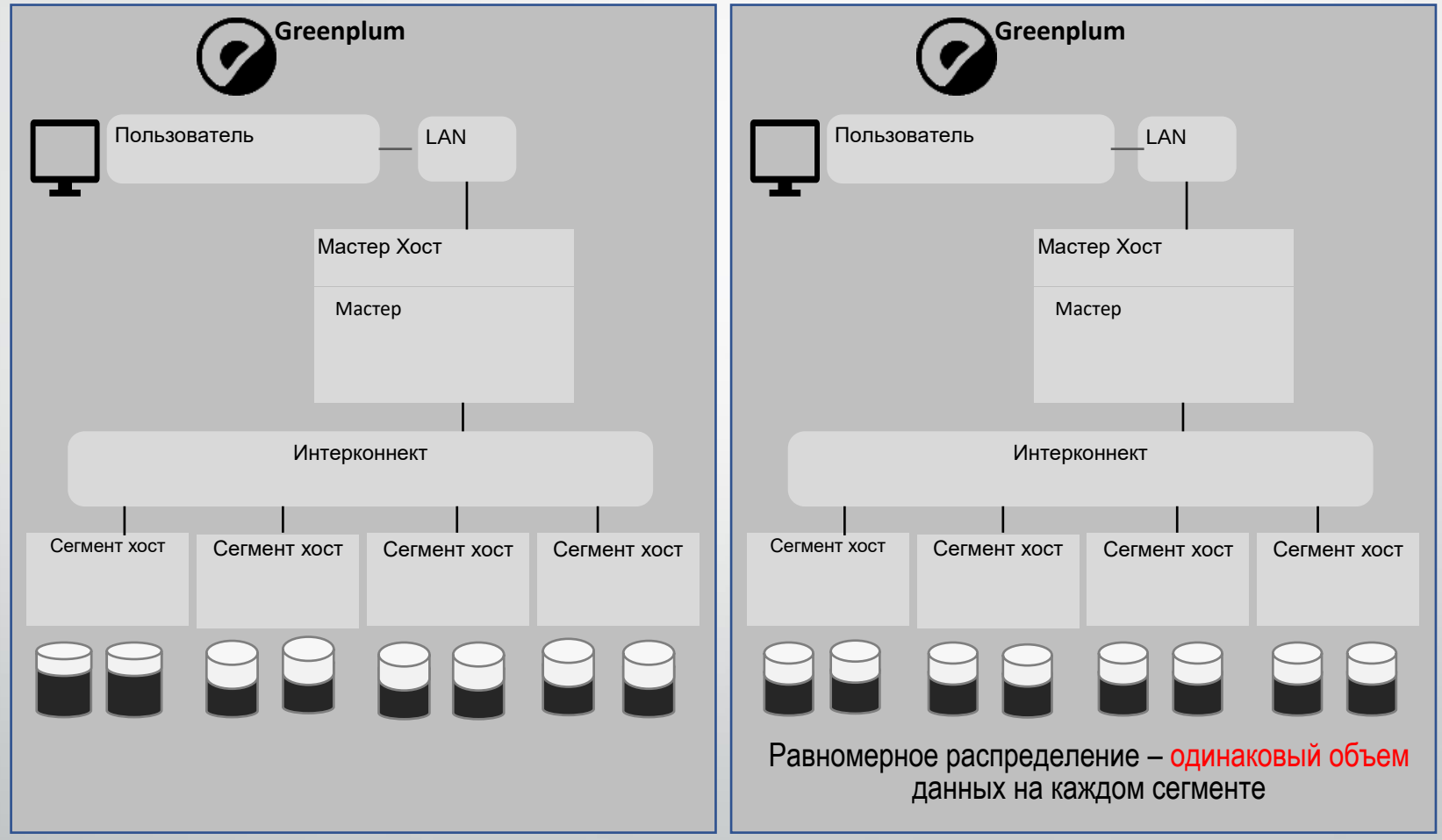
По умолчанию обмен данными происходит по протоколу UDP (User Datagram Protocol — протокол пользовательских блоков информации; один из ключевых элементов набора сетевых протоколов для Интернета).

Также программное обеспечение СУБД Greenplum выполняет дополнительную проверку пакетов поверх UDP. Таким образом, надежность остается такой же, как у TCP (Transmission Control Protocol - один из основных протоколов передачи данных Интернета). При этом производительность и возможности масштабирования намного выше.

Дистрибьюция

Дистрибьюция или распределении данных — это один из важнейших факторов быстродействия системы. В СУБД Greenplum общее время выполнения запроса измеряется временем выполнения на всех сегментах. Так как обработка на сегментах выполняется в параллель то максимальная скорость системы ограничивается работой самого медленного сегмента. Если данные распределены неравномерно, сегменты с большим количеством данных будут выполнять работу дольше, что снижает общую производительность. Таким образом, для повышения производительности необходимо стремиться к равномерному распределению данных по всем сегментам системы

Неравномерное и равномерное распределение данных по сегментам



Понимание и правильное использование такой ключевой особенности СУБД Greenplum, как распределение данных по всем сегментам с учетом особенностей данных, поможет добиться наилучшей производительности. Даже одна таблица в СУБД Greenplum хранится не на одном, а на нескольких сегментах. Каждый сегмент хранит уникальную порцию данных. Для каждой таблицы задается своя политика дистрибьюции.

ТАБЛИЦА С РАСПРЕДЕЛЕНИЕМ

id	group	name
101	A1	Student1
102	A1	Student2
103	A1	Student3
104	A1	Student4
105	B2	Student5
106	C3	Student6

РАВНОМЕРНОЕ РАСПРЕДЕЛЕНИЕ + УЧЕТ ОСОБЕННОСТЕЙ ДАННЫХ = ПРОИЗВОДИТЕЛЬНОСТЬ

Способ, согласно которому строки распределяются по сегментам задается при создании таблицы в команде Create Table, а также может быть изменен впоследствии командой Alter Table.

Ключ дистрибьюции — это поле или набор полей, по значениям которых определяется, на каком сегменте будет храниться существующая строка таблицы.

При выборе ключа дистрибьюции необходимо учитывать 2 основных момента:

Ключ должен обеспечивать равномерность распределения данных;

Добиться локальности операций.

Под локальными операциями подразумевается ситуации, когда соединяемые таблицы имеют равный ключ дистрибьюции, что позволяет выполнять соединение локально на сегментах без необходимости перераспределять данные, что, в свою очередь, дает существенное ускорение выполнения запроса СУБД Greenplum.

Существует 3 вида дистрибьюции:

1 - distributed by;

2 - distributed randomly;

3 - distributed replicated.

1 - Distributed by — это дистрибьюция по hash указанных полей. Идея соединения с помощью хеширования состоит в поиске подходящих строк с помощью заранее подготовленноё хеш-таблицы.

Хеш-таблица позволяет сохранять пары, составленные из ключа хеширования и значения, а затем искать значения по ключу за фиксированное время, не зависящее от размера хеш-таблицы.

В нашем примере из значения колонки или нескольких колонок вычисляется хэш-значения, по которому в дальнейшем определяется, на какой сегмент попадает запись. В нашем примере вычисления хеш-значения осуществляется по значениям одной или двух колонок.

2 - Distributed randomly - это метод, при котором планировщик сам равномерно размещает данные по всем сегментам. Данный метод лучше всего подходит для таблиц, в которых отсутствует или сложно определить ключ, который обеспечит равномерное распределение данных. Важно учесть, что при таком алгоритме соединения с другими таблицами будет всегда вызывать перераспределение данных, так как строки распределены случайным образом.

3 - Distributed replicated – это метод, при котором на каждом сегменте хранится полная копия таблицы.

Такой метод подходит для небольших таблиц, например, справочников, которые регулярно участвуют в соединениях с другими таблицами.

Хранение полной копии таблицы на каждом сегменте позволяет сэкономить время, которое в противном случае было бы потрачено на перераспределение во время выполнения запроса.

При выборе ключа дистрибьюции необходимо руководствоваться следующими правилами:

- в качестве ключа дистрибьюции необходимо выбирать поле с большой селективностью, то есть поля с большим количеством уникальных значений. Как правило, под этот критерий подходит поля с идентификаторами.

Если данные распределены по ключу с низкой селективностью, то велика вероятность неравномерного распределения данных, так как одинаковые значения будут располагаться на одном сегменте;

- в ключе не должно быть Null или значений по умолчанию. Иначе все Null или другие значения по умолчанию будут попадать на один сегмент, что приведет к перекосу данных.

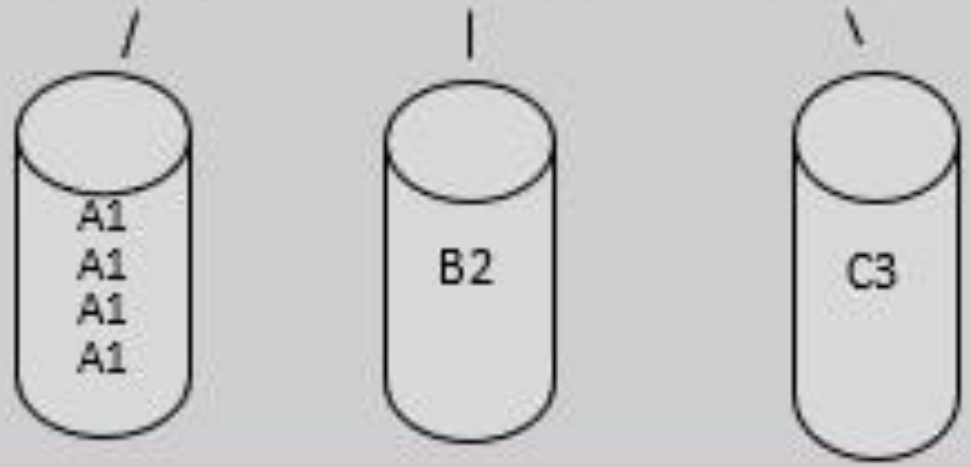
Вернемся к рассмотрению примера.

Выберем в качестве ключа дистрибьюции поле ID. В результате распределения на каждый сегмент падает ровно по две строки. Распределение получается равномерное, а значит, ключ дистрибьюции хороший.

Выберем в качестве ключа дистрибьюции поле group.

id	group	name
101	A1	Student1
102	A1	Student2
103	A1	Student3
104	A1	Student4
105	B2	Student5
106	C3	Student6

ALTER TABLE client DISTRIBUTER BY group



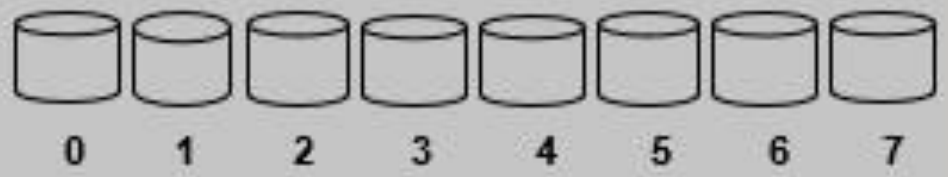
После распределения мы видим, что на одном из сегментов данных в 4 раза больше, чем на других. Как следствие, и работать этот сегмент будет в 4 раза дольше. Делаем вывод о том, что ключ выбран не оптимально. Проверить равномерность распределения строк между сегментами можно, используя служебный столбец **GP** сегмент ID, который присутствует в каждой таблице

```
SELECT gp_segment_id, count (1)
FROM sales
GROUP BY gp.segmentjd
```

В данном столбце содержится идентификатор сегмента, на котором лежит строка пример запроса с группировкой по полю GP сегмент ID для определения равномерности распределения строк по сегментам, приведен на экране. По результату запроса видно, что распределение данных таблицы Sails практически равномерное.

КАК ПРОВЕРИТЬ РАВНОМЕРНОСТЬ РАСПРЕДЕЛЕНИЯ?

gp_segment_id	count
0	6251759
5	6249646
2	6246734
7	6249281
3	6251719
4	6253486
1	6247555
6	6249820



КАК ПРОВЕРИТЬ РАВНОМЕРНОСТЬ РАСПРЕДЕЛЕНИЯ?

gp_segment_id	count

Ключ распределения можно изменить без пересоздания таблицы с помощью предложения

ALTER TABLE tabi SET DISTRIBUTED BY (coll)

ALTER TABLE tab! SET DISTRIBUTED RANDOMLY

Также есть возможность перераспределения данных по прежнему ключу дистрибьюции при помощи опции

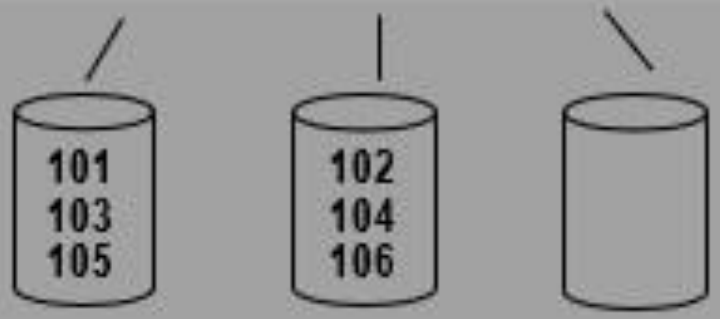
ALTER TABLE tabi SET WITH (REORGANIZE=true)

Эта опция будет полезна в двух случаях: при добавлении нового сегмента и для избежания раздутия таблиц, результат следующий:

Перераспределение данных без изменения ключа

id	group	name
101	A1	Student1
102	A1	Student2
103	A1	Student3
104	A1	Student4
105	B2	Student5
106	C3	Student6

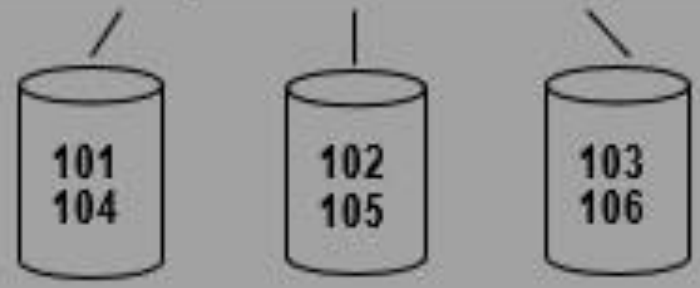
CREATE TABLE client DISTRIBUTED BY id



Перераспределение данных без изменения ключа

id	group	name
101	A1	Student1
102	A1	Student2
103	A1	Student3
104	A1	Student4
105	B2	Student5
106	C3	Student6

ALTER TABLE client SET WITH (REORGANIZE=true)



При добавлении нового сегмент хоста в кластер возникает необходимость заново перераспределить данные, чтобы избежать перекоса в распределении данных. Когда новый сегмент фактически не задействован для хранения данных, а все данные по-прежнему распределены по старым сегментам.

Некоторые важные рекомендации, которые помогут вам правильно распределять данные:

- явно задавать способ и ключ распределения при создании таблицы, в том числе для временных таблиц;
- использовать RANDOMLY-распределение для небольших таблиц, где нет хороших кандидатов на ключ распределения из одного атрибута;
- избегать использования в качестве ключа распределения атрибутов, которые будете часто указывать в предложении WHERE и атрибутов с типом данных date, timestamp;
- стремиться использовать в качестве ключа распределения тип данных integer (предпочтительнее, чем string) и столбец, который будете использоваться как ключ соединения (JOIN ON).

Благодарю за внимание!