Greenplum 大数据平台：6.0新特性介绍及7.0展望

杨瑜

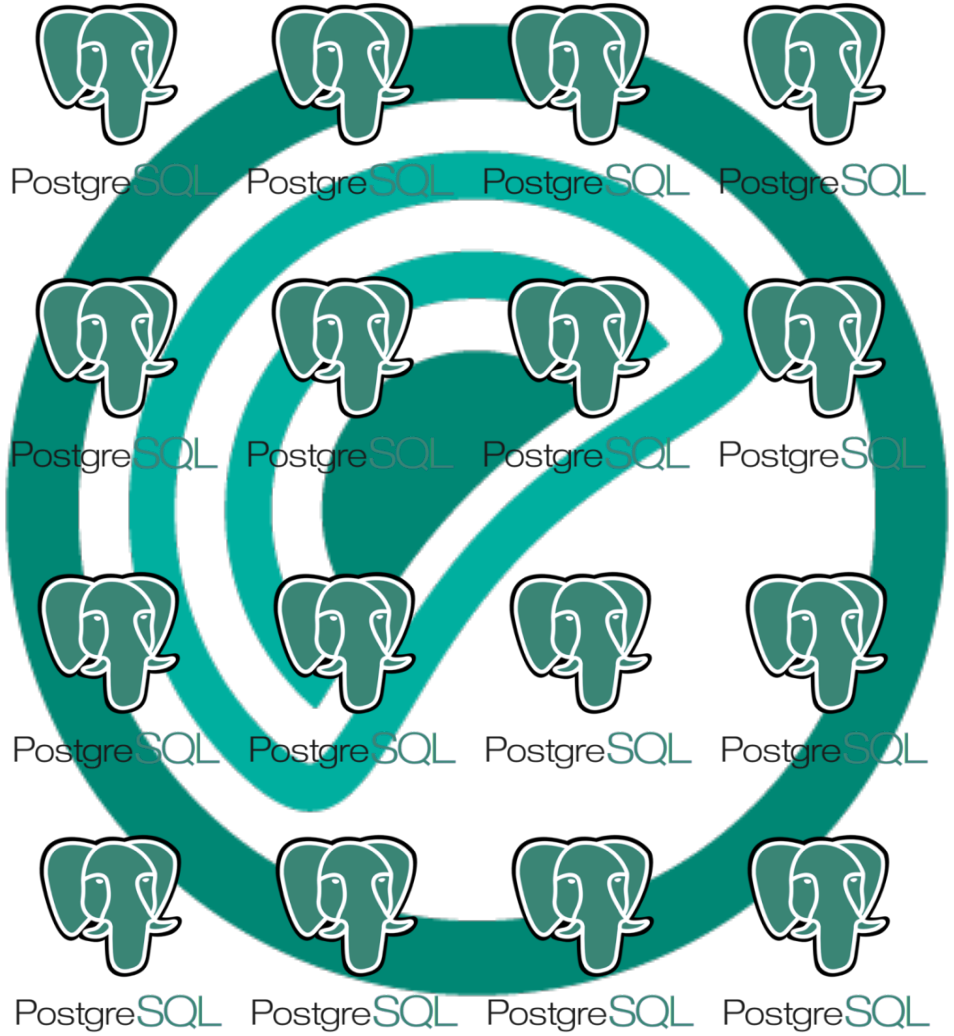
2019年11月

日程

- Greenplum 介绍
- Greenplum 6.0 新特性
- Roadmap 展望

Greenplum 介绍

Greenplum内核：基于MPP架构的分布式并行处理

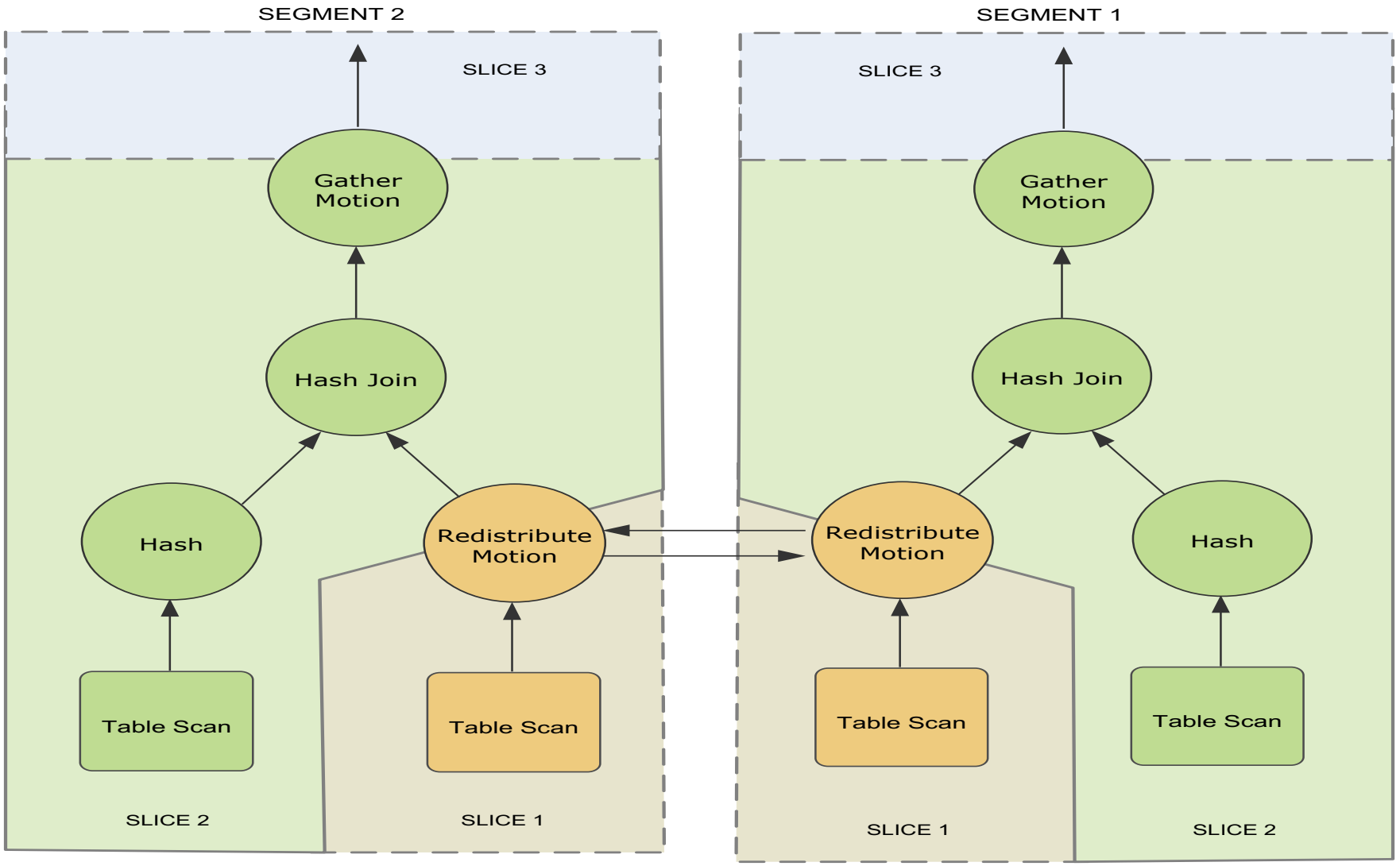


Greenplum量身打造

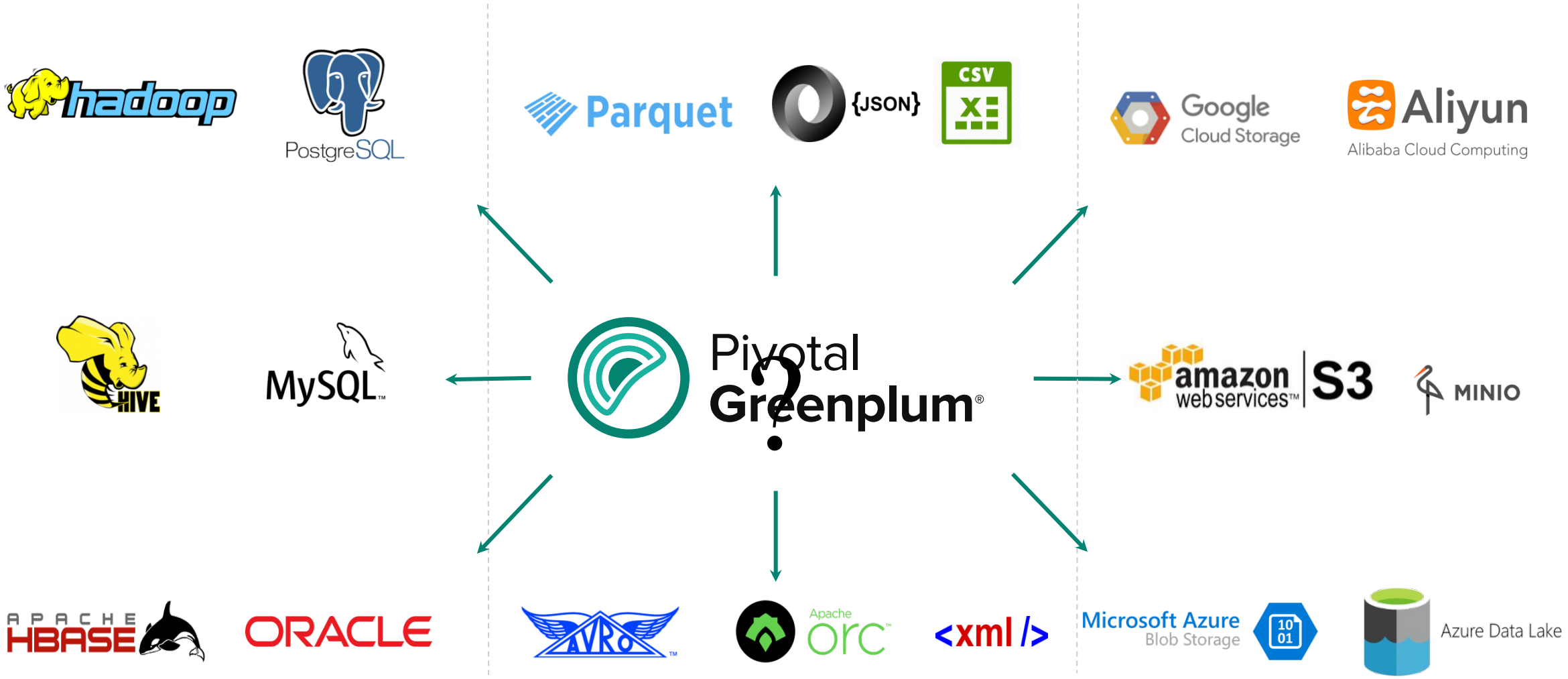
-- 并行查询优化器：planner VS gporca
-- 高速可靠的节点间基础数据交换
-- 集群级系统高可用及容错
-- 针对大数据的支持和丰富扩展

并行查询

```
SELECT customer,

amount
FROM    sales
JOIN    customer
USING (cust_id)
WHERE   date=2008;
```

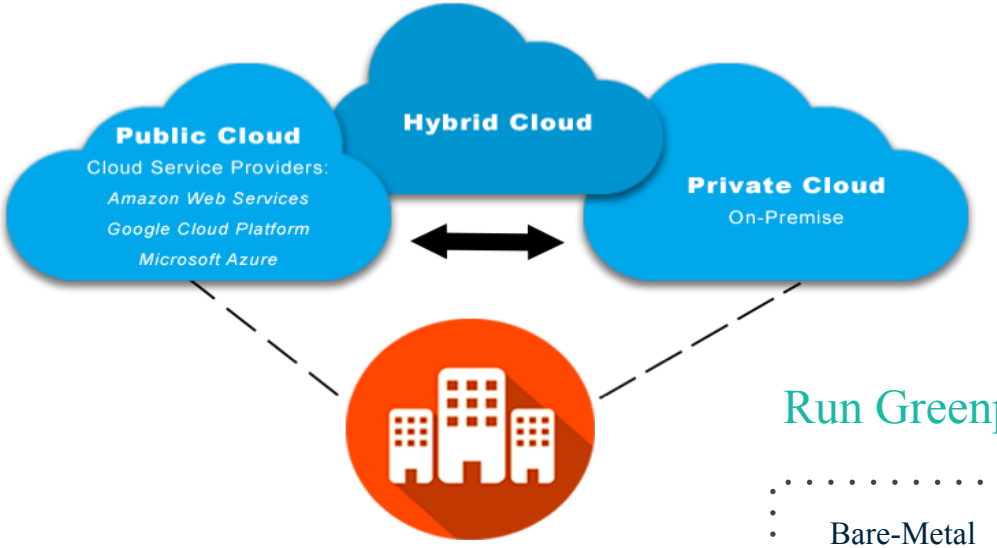


Greenplum 多态存储



Greenplum 混合云部署

GREENPLUM DATABASE



Pivotal Greenplum®

Run Greenplum in Any Environment

Bare-Metal

Private Cloud

vmware vSphere

openstack.

Public Cloud

amazon web services

Google Cloud Platform

Microsoft Azure

Greenplum for Kubernetes

Pivotal Container Service

Enterprise & Essentials(OSS K8s)

Google Container Engine

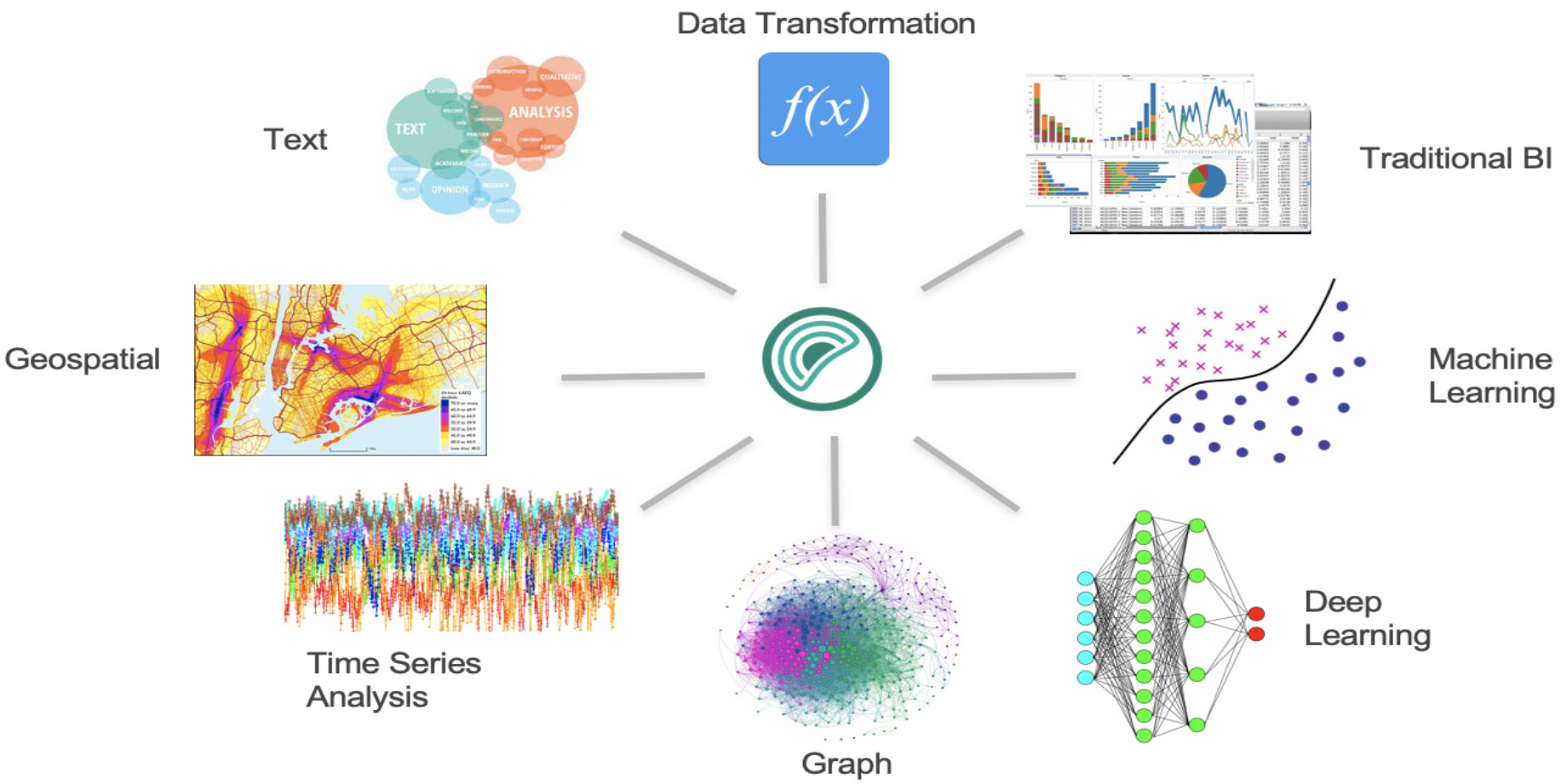
Other Kubernetes (on VMs or not)

Greenplum Building Blocks

- The most performant way to run Greenplum on premise
- Pivotal Blueprint for Dell reference hardware configs
- Superior price/performance; no expensive proprietary hardware
- Cert[illegible] by Pivotal

DELL

Greenplum 集成分析



Greenplum 6.x

GREENPLUM DATABASE®

2019 年 9 月4日正式发布！

Version 6

Massively Parallel Postgres for Analytics

Experience Greenplum Database, an open-source massively parallel data platform for analytics, machine learning and AI

DOWNLOAD

Greenplum 内核大幅升级

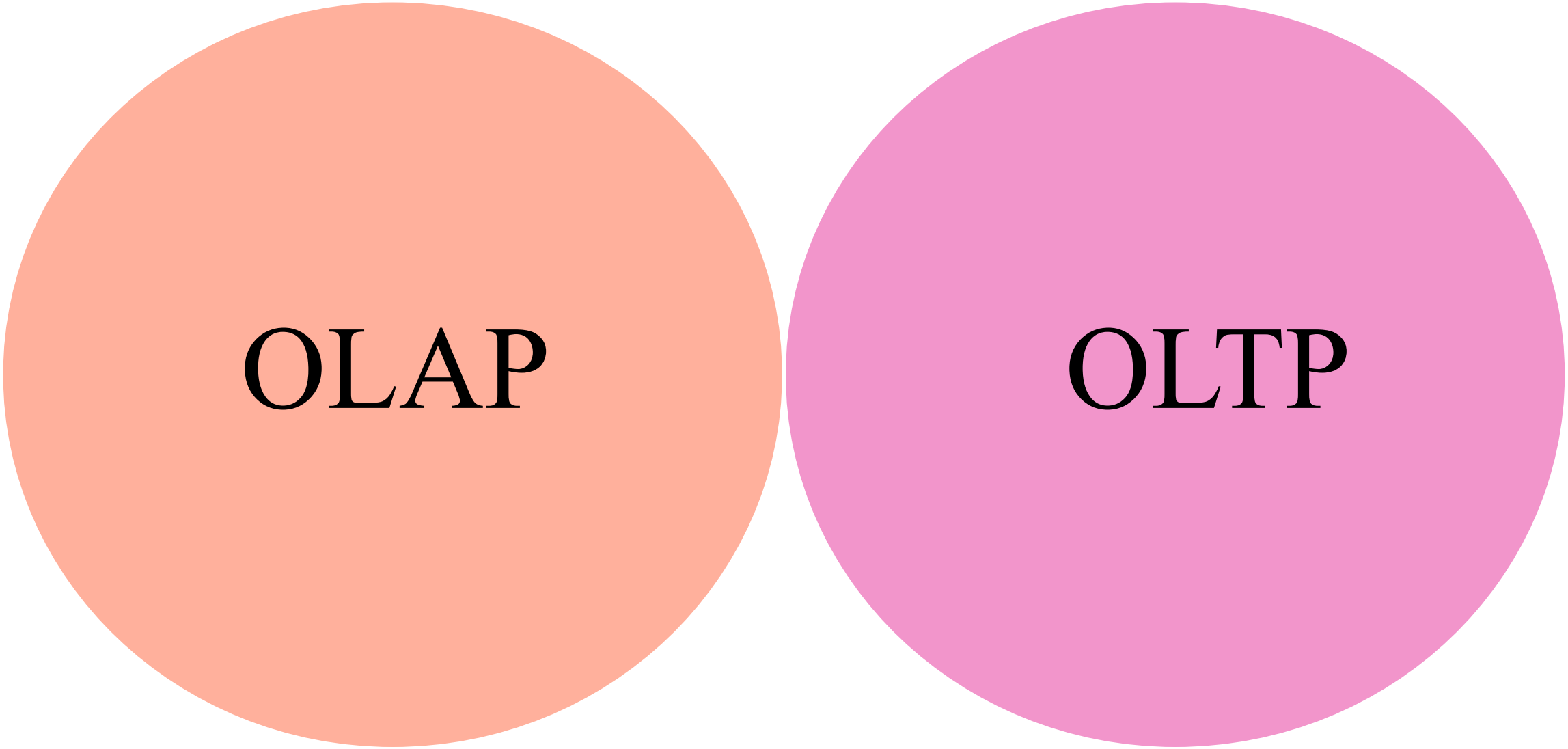
6个Postgres大版本升级

- v8.4 - 2314 commits
- v9.0 - 1859 commits
- v9.1 - 2035 commits
- v9.2 - 1945 commits
- v9.3 - 1603 commits
- v9.4 - 1964 commits

总共11720 commits

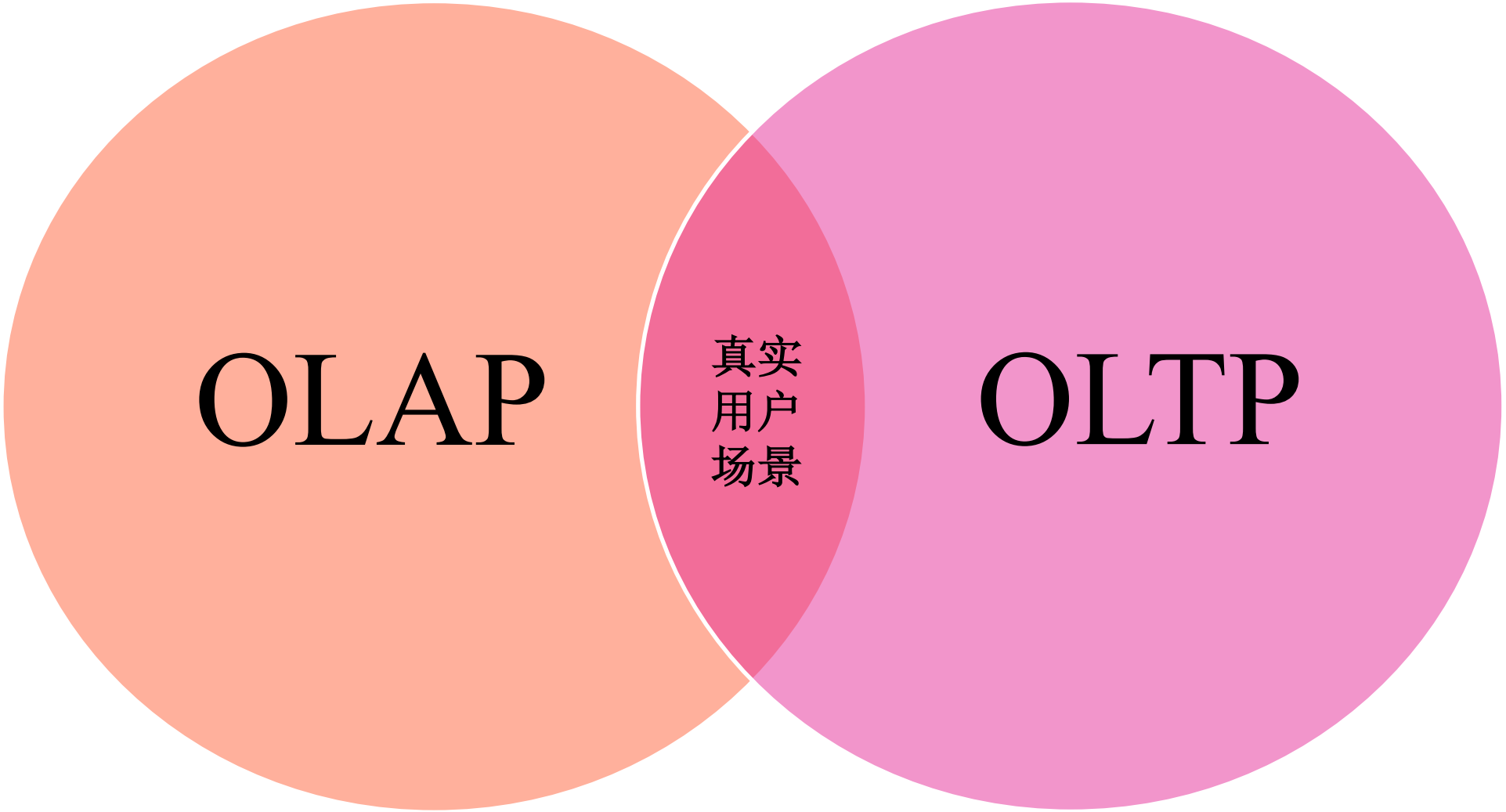
OLAP vs OLTP ?

GREENPLUM DATABASE

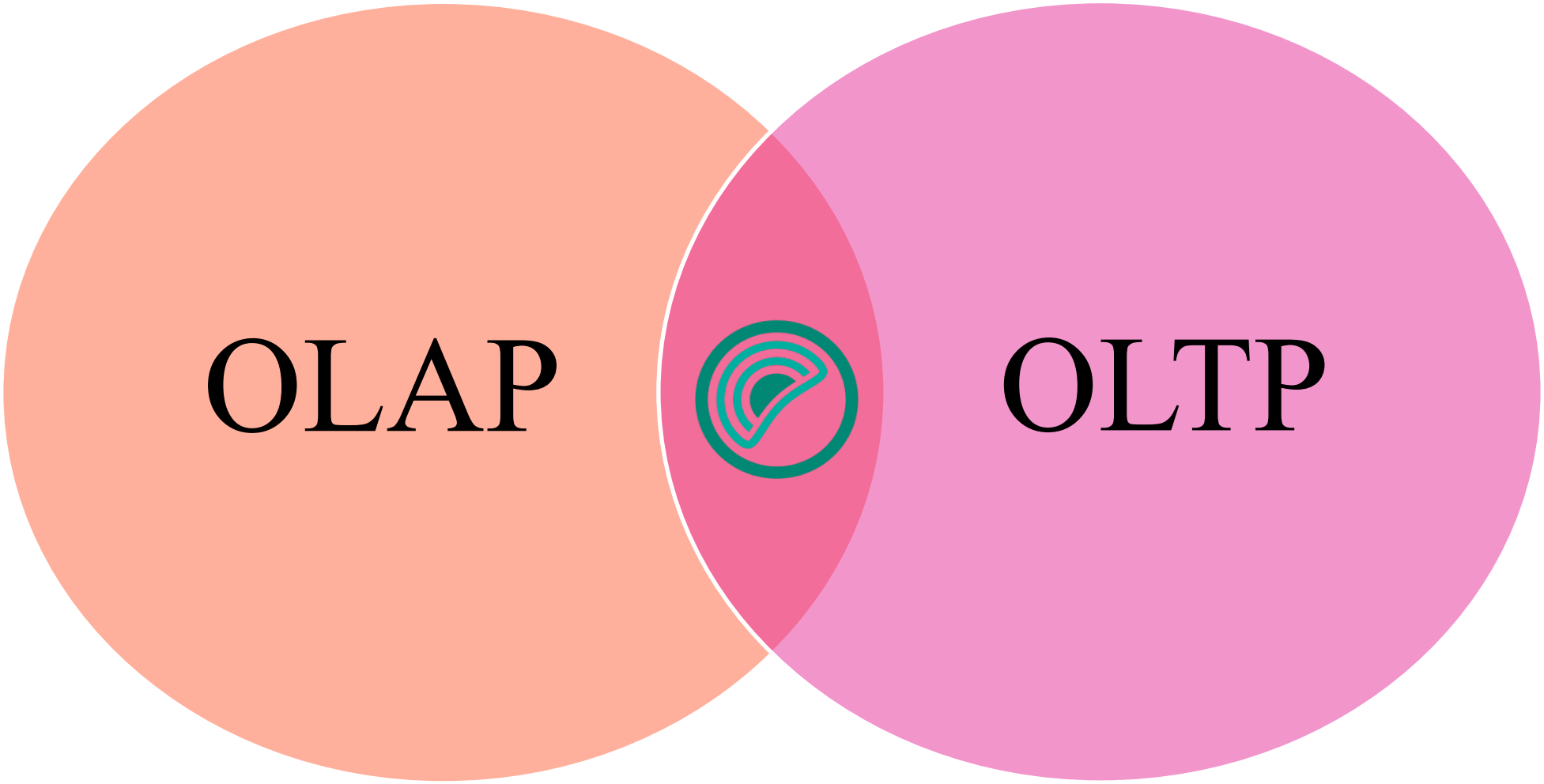


真实用户场景，那么？

GREENPLUM DATABASE



Greenplum HTAP：事务型+分析型融合查询



OLTP性能大幅提升

GREENPLUM DATABASE

Greenplum 6 中核心优化点

- 锁的优化（如tmLock）
- 两阶段事务优化（如特殊情况退化为一阶段）
- 升级内核带来的优化（如fast path lock, lwlock， cache有效性）

平均性能提升高达70+倍

- 许多分析系统需要混合使用OLAP和OLTP查询
- 高并发OLTP操作
- Update TPS 24448
- Insert TPS 46570
- Select TPS 140000

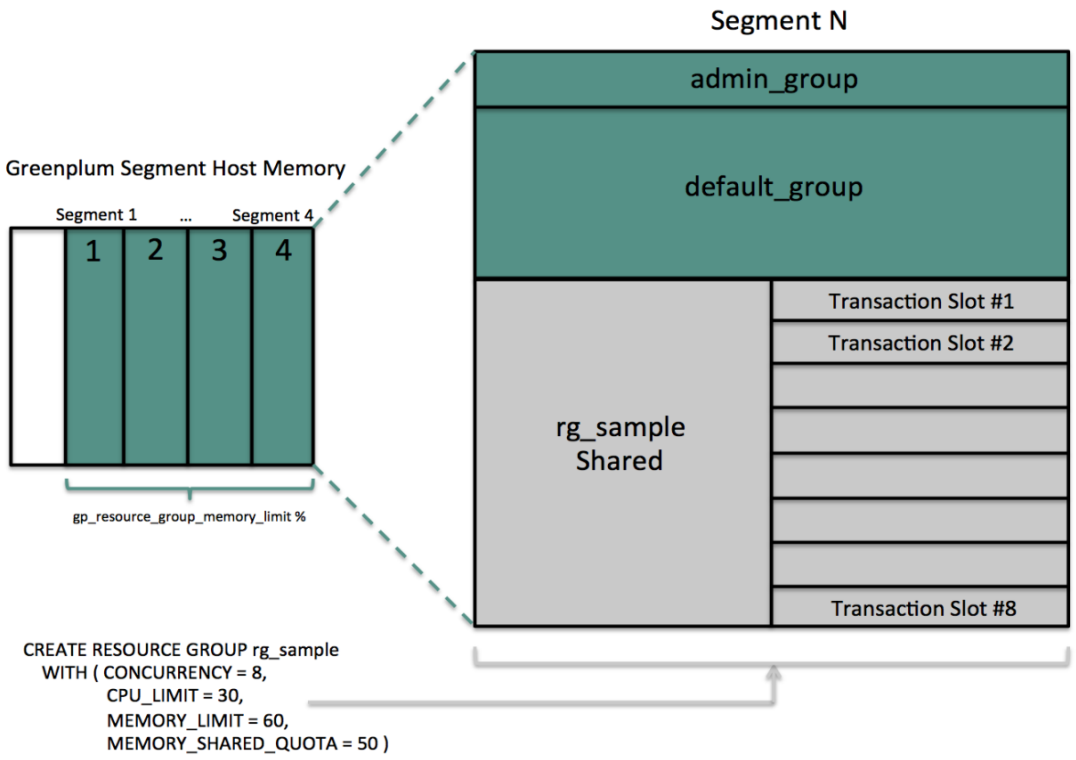
多租户：精细化资源管理

目标

- 多租户资源隔离
- 更好地支持混合工作负载
- 更好地支持高并发

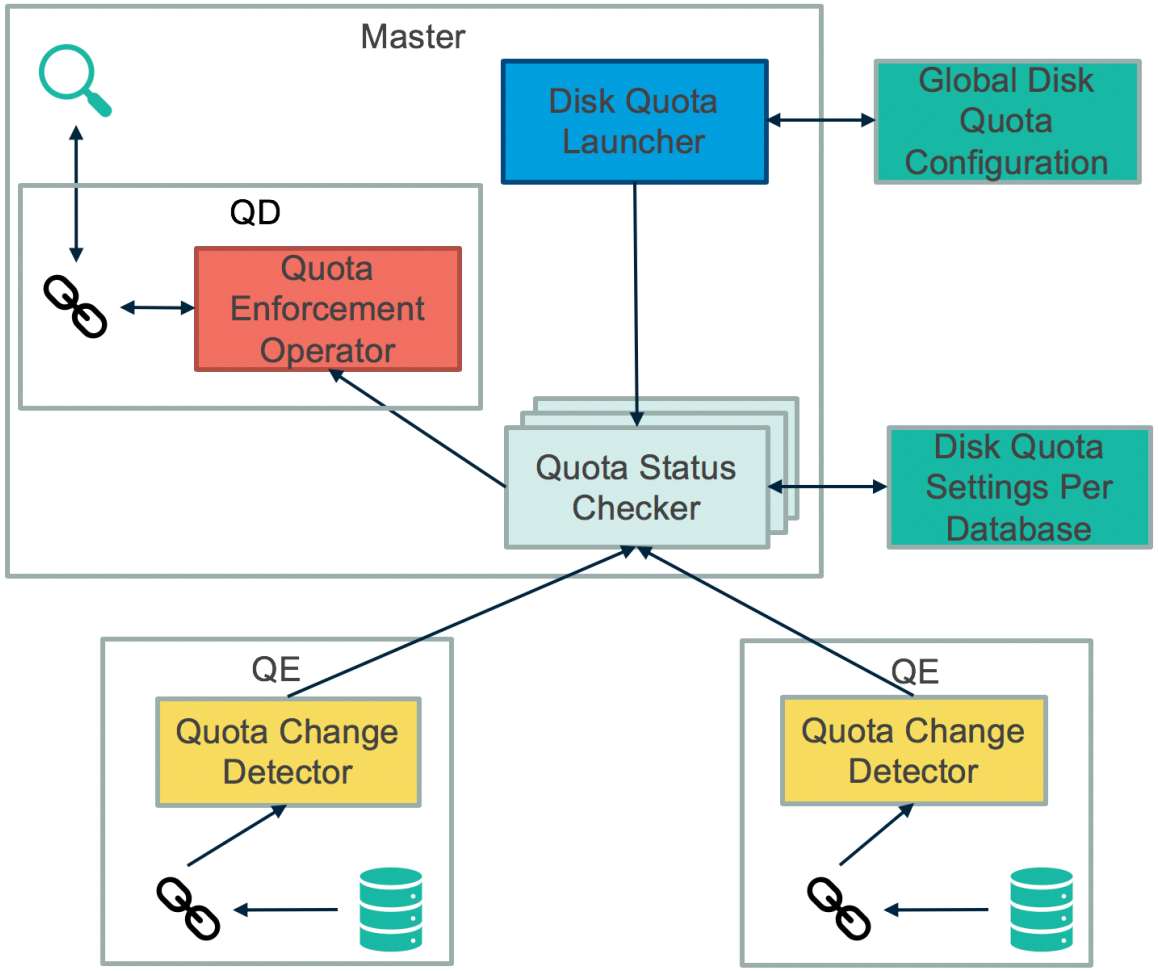
特征

- 指定每个用户组的最大CPU
- 指定每个用户组和限制每个查询的最大内存
- 指定每个用户组的最大并发数
- 利用Linux Cgroup实现



多租户：磁盘配额管理

- 多租户的场景下，磁盘配额监控，减少不同查询间相互影响
- 支持Schema和用户级别的磁盘配额创建和管理
- 当磁盘使用量达到限额时，阻止用户继续占用磁盘空间
- 基于Greenplum Extension框架构建
- 最小化IO性能的影响
- 能达到近似实时的响应

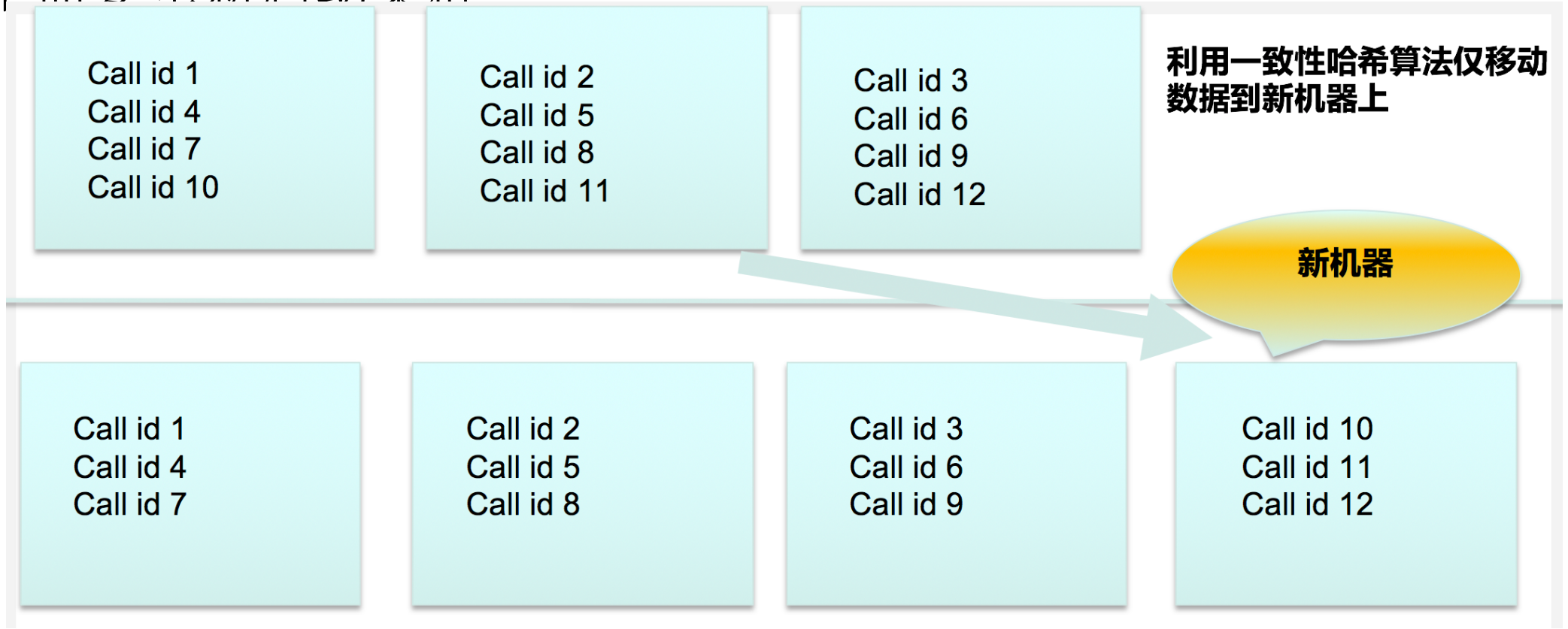


更快更高效的集群扩容

GREENPLUM DATABASE

10倍以上性能提升

- 新的一致性哈希算法
- 不需要重新分布集群所有数据，仅移动少量数据
- 不需要重新启动集群

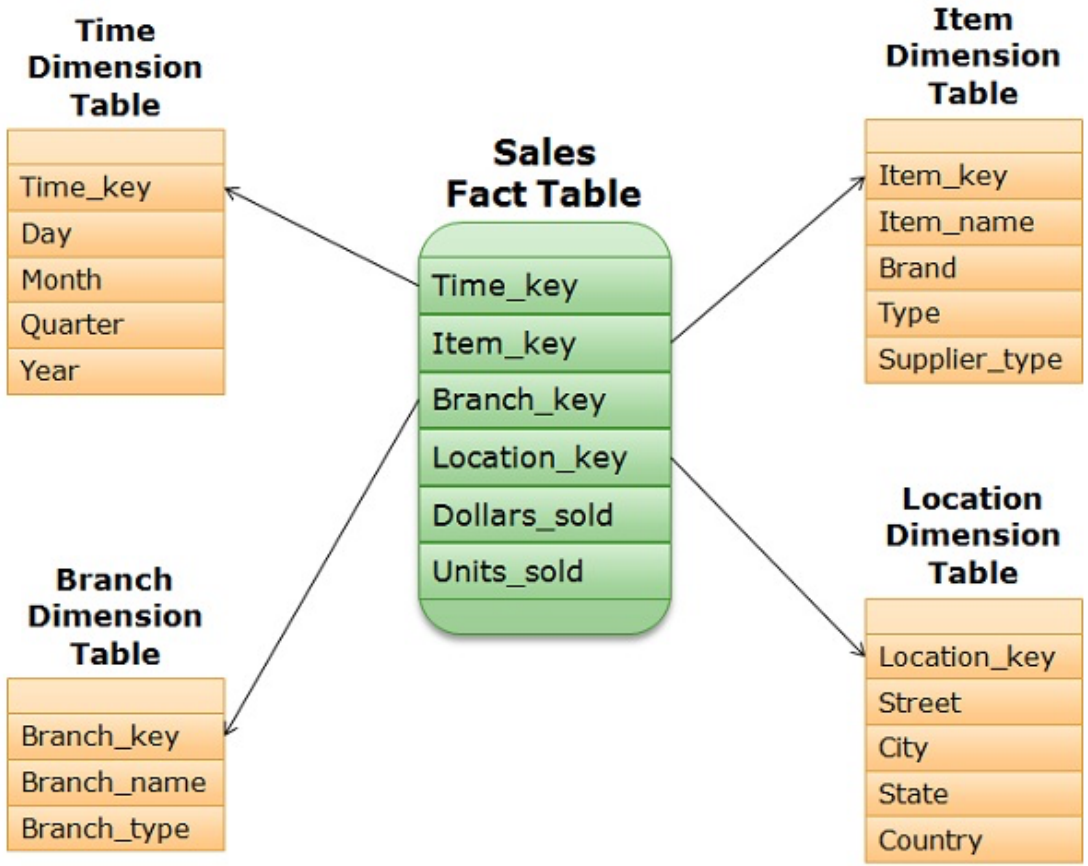


复制表

Greenplum 6 新语法 “DISTRIBUTED REPLICATED”

- 表不会分片存放，而是整个完全复制到各个节点上
- 做连接操作时不会做额外的数据重分布或广播
- 复制表同时支持 PRIMARY KEY 和 UNIQUE 约束，但分布表需要包含分布键

复制表加速本地连接（Local Join）操作



复制表（续）

非复制表

```
explain select count(*) from table_fact f inner join table_non_replicated d on f.a = d.a;
                                                    QUERY PLAN
-----------------------------------------------------------------------------------------------------------
 Aggregate  (cost=0.00..874.31 rows=1 width=8)
   ->  Gather Motion 2:1  (slice3; segments: 2)  (cost=0.00..874.31 rows=1 width=8)
         ->  Aggregate  (cost=0.00..874.31 rows=1 width=8)
               ->  Hash Join  (cost=0.00..874.31 rows=50000 width=1)
                     Hash Cond: (table_fact.a = table_non_replicated.a)
                     ->  Redistribute Motion 2:2  (slice1; segments: 2)  (cost=0.00..433.15 rows=50000 width=4)
                           Hash Key: table_fact.a
                           ->  Seq Scan on table_fact  (cost=0.00..432.15 rows=50000 width=4)
                     ->  Hash  (cost=431.22..431.22 rows=5000 width=4)
                           ->  Redistribute Motion 2:2  (slice2; segments: 2)  (cost=0.00..431.22 rows=5000 width=4)
                                 Hash Key: table_non_replicated.a
                                 ->  Seq Scan on table_non_replicated  (cost=0.00..431.12 rows=5000 width=4)
 Optimizer: PQO version 3.29.0
```

复制表

```
explain select count(*) from table_fact f inner join table_replicated d on f.a = d.a;
                                                QUERY PLAN
-----------------------------------------------------------------------------------------------------
 Aggregate  (cost=0.00..874.73 rows=1 width=8)
   ->  Gather Motion 2:1  (slice1; segments: 2)  (cost=0.00..874.73 rows=1 width=8)
         ->  Aggregate  (cost=0.00..874.73 rows=1 width=8)
               ->  Hash Join  (cost=0.00..874.73 rows=50000 width=1)
                     Hash Cond: (table_fact.a = table_replicated.a)
                     ->  Seq Scan on table_fact  (cost=0.00..432.15 rows=50000 width=4)
                     ->  Hash  (cost=431.23..431.23 rows=10000 width=4)
                           ->  Seq Scan on table_replicated  (cost=0.00..431.23 rows=10000 width=4)
 Optimizer: PQO version 3.29.0
```

1 slice vs 3 slices
不需要数据重分布

zStandard压缩

- 源于Facebook，Greenplum 6.0 发布
- 更快的压缩和解压速度
- 更大的压缩率，节省存储空间
- 创建表时可以通过如下选项开启：

WITH (compresstype=zstd)

Compressor name	Ratio	Compression	Decompress.
zstd 1.3.4 -1	2.877	470 MB/s	1380 MB/s
zlib 1.2.11 -1	2.743	110 MB/s	400 MB/s
brotli 1.0.2 -0	2.701	410 MB/s	430 MB/s
quicklz 1.5.0 -1	2.238	550 MB/s	710 MB/s
lzo1x 2.09 -1	2.108	650 MB/s	830 MB/s
lz4 1.8.1	2.101	750 MB/s	3700 MB/s
snappy 1.1.4	2.091	530 MB/s	1800 MB/s
lzf 3.6 -1	2.077	400 MB/s	860 MB/s

列级安全管理

SELECT name, address FROM people;

SELECT ssn FROM people;

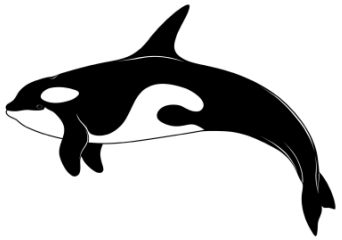
SELECT * from people;

```
CREATE TABLE
INSERT 0 1
INSERT 0 1
INSERT 0 1
GRANT
     ssn      |   name   |   address
--------------+----------+-------------
 123-45-6789  | John Doe | main street
 123-45-6790  | Jane Doe | fare street
 123-45-6791  | Jill Doe | hill street
(3 rows)

[gpadmin@mdw ~]$
```

```
[gpadmin@mdw ~]$ psql -f col_perms2.sql -U analyst1 testdb
   name    |    address
----------+-------------
 John Doe | main street
 Jane Doe | fare street
 Jill Doe | hill street
(3 rows)

psql:col_perms2.sql:2: ERROR:  permission denied for relation people
psql:col_perms2.sql:3: ERROR:  permission denied for relation people
[gpadmin@mdw ~]$ cat col_perms2.sql
```

GPORCA 查询优化器处理复杂查询

Differences between good and mediocre optimizers produce large performance differences as data size increases[1]



[1] Z. Gu, M. A. Soliman, and F. M. Waas. Testing the Accuracy of Query Optimizers. In DBTest, 2012.

2011: 自主研发查询优化器 GPORCA 立项开始

2015: GPORCA 查询优化器发布

2019: 6.0 发布，GPORCA 诸多提升，多于1000 个代码提交

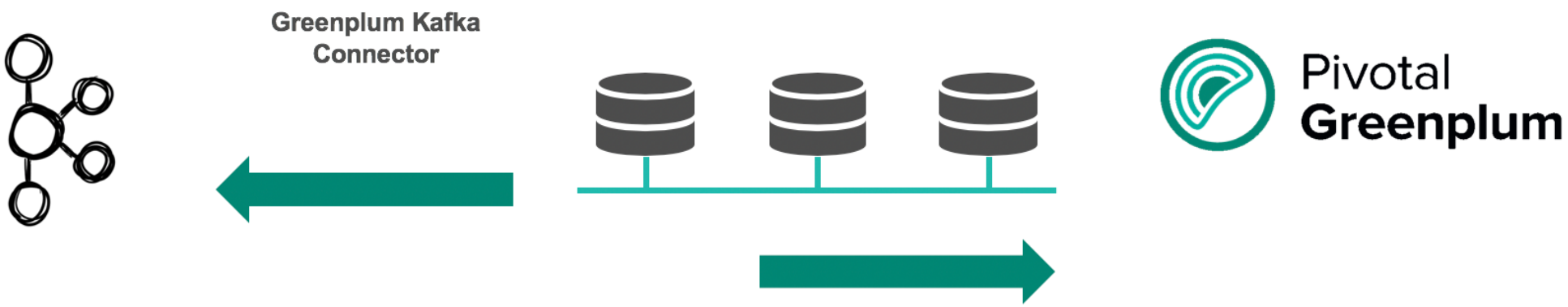
Unlogged 表用于更快的数据加载

GREENPLUM DATABASE

- 批量加载通常会包含一些列的事务和准备阶段
- 但是有时候加载目的只是为了计算，这些数据是“用完即删”
- Unlogged 表不需要写WAL日志以及拷贝日志到Mirror节点重放
- Unlogged 表创建更快
- 数据会持久化到磁盘，并且有Checksum保护，但是不会写日志，如果系统重启或硬件故障存在丢失的风险
- 和临时表不同在于，会话结束数据仍然存在

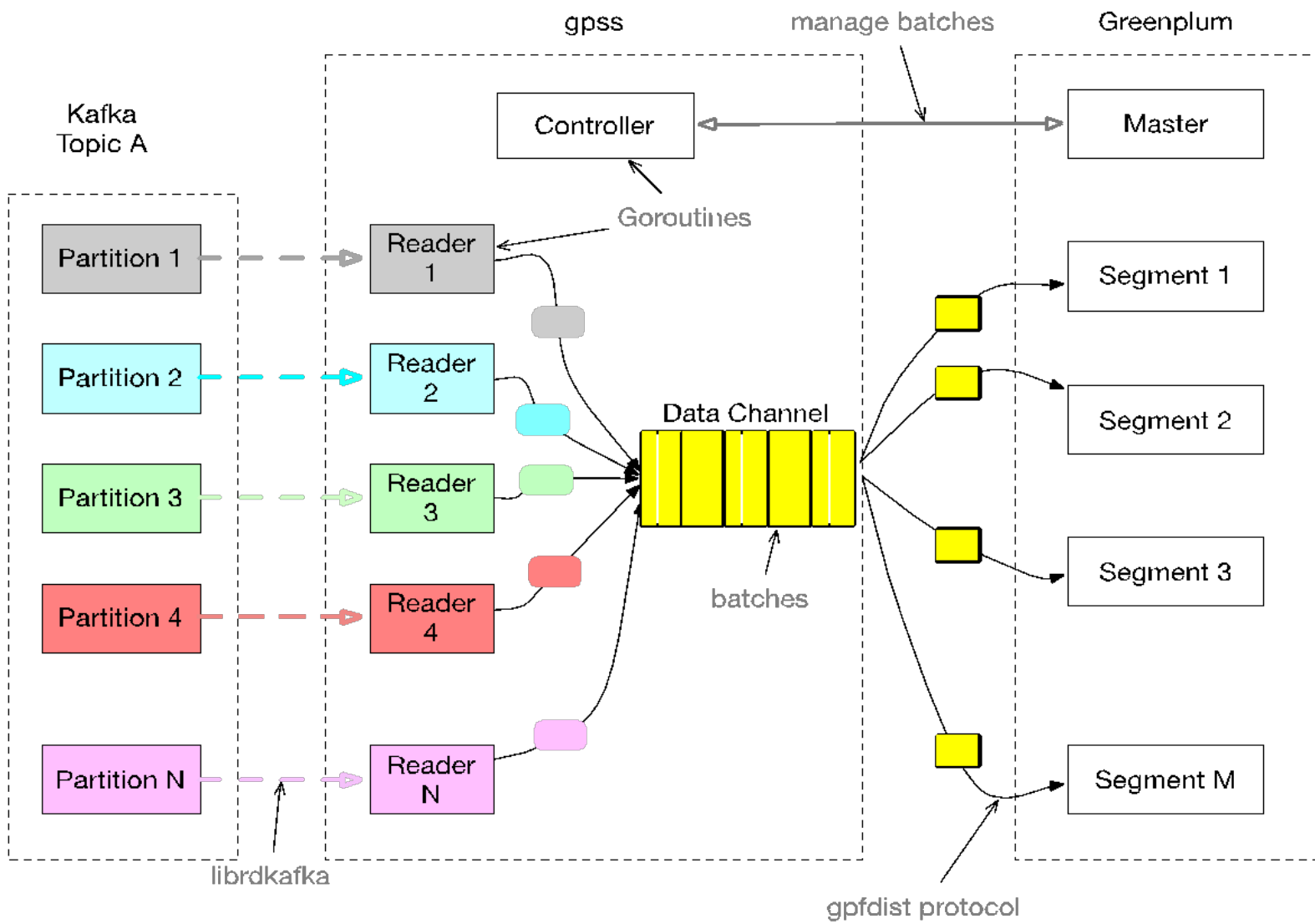
```
create unlogged table
table_unlogged
(a int , b text)
distributed randomly;
```

Kafka Connector

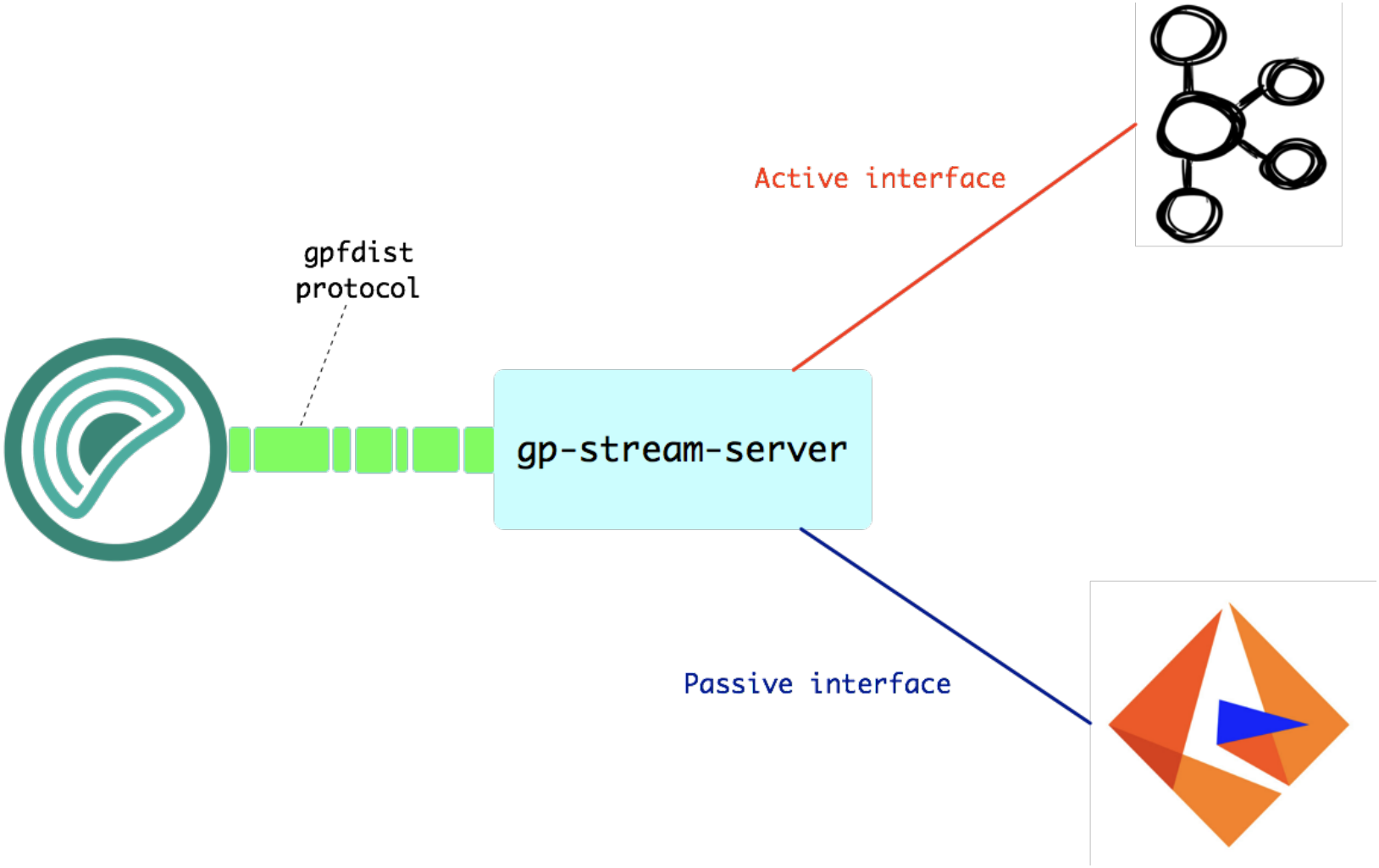


- Minibatch模式的流加载和计算引擎
- 端到端不重不丢，高速，可靠，低延迟
- Confluent认证
- 丰富的数据格式支持（Avro, Json, Text, Binary及自定义类型）
- 支持并行数据Transform
- 跟机器学习，文本分析，地理信息等分析工具无缝衔接，并通过UDF扩展
- 未来查询下推，减少数据流动

Kafka Connector 工作原理



GPSS 流式数据加载


Active interface
gpfdist
protocol
gp-stream-server
Passive interface

GPCC 6.0

GREENPLUM DATABASE

Pivotal Greenplum Command Center

欢迎使用 GPCC 6.0

新特性

- 首次支持GPDB 6
- 无需运行gpperfmon服务
- 提供日志级别来控制日志粒度
- 搜索历史查询时，使用过滤器缩小搜索范围
- 监测Standby Master服务器状态

我们需要您的反馈

关于 GPCC 和 WLM
gpcc-feedback@pivotal.io

关于所有其他Greenplum的主题
gpdb-feedback@pivotal.io

登入

服务器: **gpcc600**

gpmon

••••••••

☐ 记住我的用户名

登入

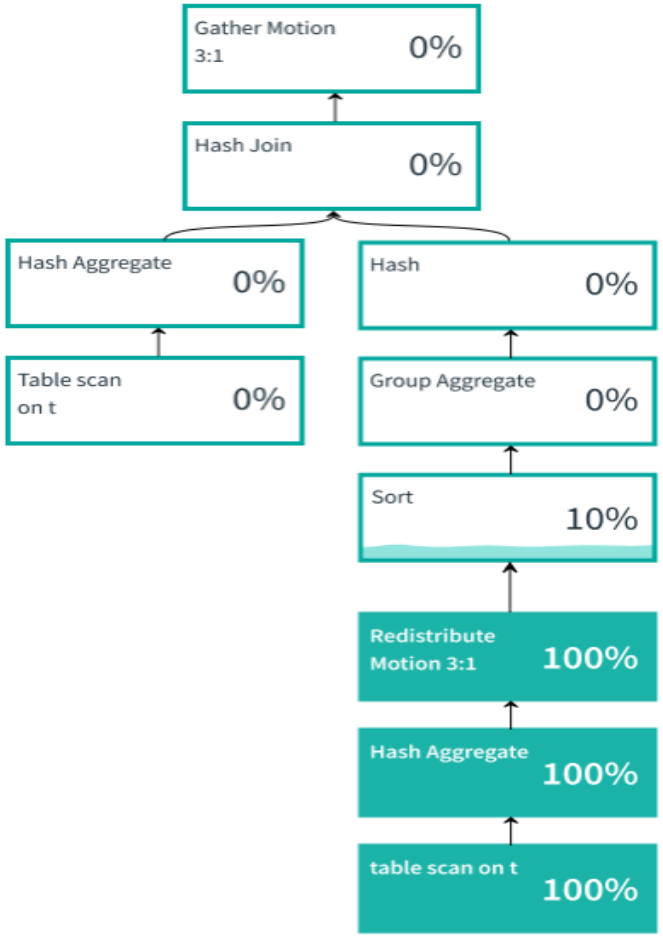
查询计划可视化展示

GREENPLUM DATABASE

Query Text

```
select s_name, count(*) as numwait from supplier, lineitem l1,
 orders, nation where s_suppkey = l1.l_suppkey and
o_orderkey = l1.l_orderkey and o_orderstatus = 'F' and
l1.l_receiptdate > l1.l_commitdate and exists ( select * from
  lineitem l2 where l2.l_orderkey = l1.l_orderkey and
l2.l_suppkey <> l1.l_suppkey ) and not exists ( select * from
  lineitem l3 where l3.l_orderkey = l1.l_orderkey and
l3.l_suppkey <> l1.l_suppkey and l3.l_receiptdate >
l3.l_commitdate ) and s_nationkey = n_nationkey group by
 s_name order by numwait desc, s_name ;
```

Plan & Progress | Textual Explain



递归 CTEs 和 可更新 CTEs

利用可更新CTE在一个语句中实现数据的查询和更新

```
test-# SELECT * FROM t;
 productname |  price
-------------+---------
 galaxy      | 1080.45
 mi          |  992.25
 iphone      |  1102.5
(3 rows)

test=# WITH t AS (UPDATE products SET price = price * 1.05 RETURNING *)
test-# SELECT * FROM t;
 productname |   price
-------------+-----------
 iphone      |  1157.625
 galaxy      | 1134.4725
 mi          | 1041.8625
(3 rows)
```

```
test=# WITH RECURSIVE t(n) AS (VALUES (1) UNION ALL SELECT n+1 FROM t WHERE n < 100)
test-# SELECT sum(n) FROM t;
 sum
------
 5050
(1 row)
```

利用递归CTE实现复杂处理逻辑

更好地支持半结构化数据

GREENPLUM DATABASE

- 结构化数据可以很好的存放在表里
- 半结构化数据由文档组成
- 结构动态，不需要修改表的DDL
- 应用使用更方便
- JSON, XML, KeyValue 等多种格式
- 可用通过条件查询
- 也可以通过数据域作聚合

```
{
    hey: "guy",
    anumber: 243,
  - anobject: {
        whoa: "nuts",
      - anarray: [
            1,
            2,
            "thr<h1>ee"
        ],
        more: "stuff"
    },
    awesome: true,
    bogus: false,
    meaning: null,
    japanese: "明日がある。",
    link: http://jsonview.com,
    notLink: "http://jsonview.com is great"
}
```

```
<?xml version="1.0"?>
<quiz>
 <qanda seq="1">
  <question>
   Who was the forty-second
   president of the U.S.A.?
  </question>
  <answer>
   William Jefferson Clinton
  </answer>
 </qanda>
 <!-- Note: We need to add
  more questions later.-->
</quiz>
```

XML

```
k => v
foo => bar, baz => whatever
"1-a" => "anything at all"
```

Whether we're analyzing social media, documents, call recordings or other language, we need to know how words impact decisions and actions

But 80-90% of enterprise data is communications is in a language that machines can't analyze, like human-readable formats, or binaries

典型应用场景

GREENPLUM DATABASE

信息安全/欺诈检测

通过交易记录检查可疑信息

通过分析文件发现法律风险

客户服务

客服数据分析，分类，打分

发现潜在的客户需求

医疗健康

电子病例自动分析

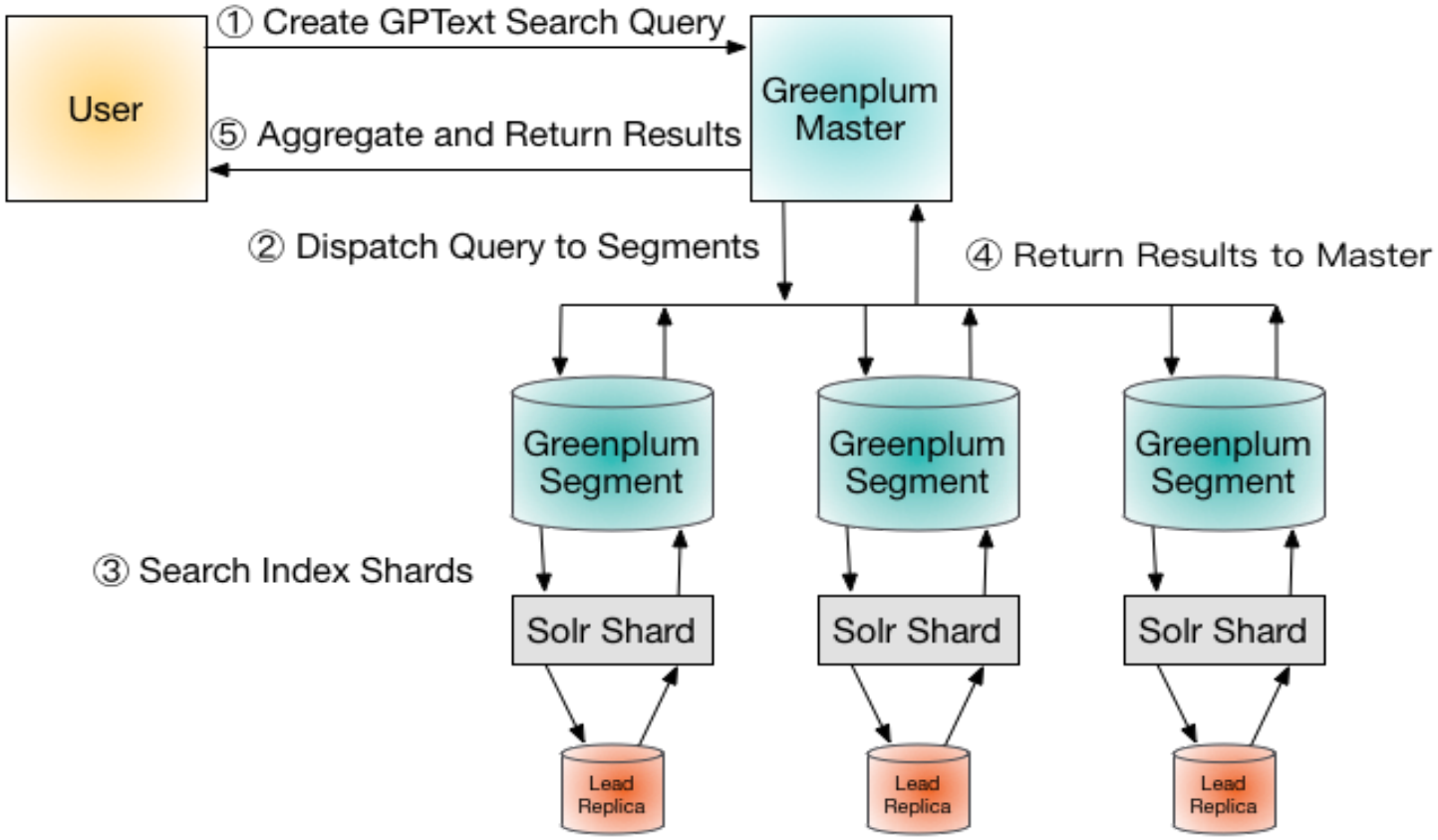
图片等医疗资料深度挖掘，疾病自动筛查

索引和全文检索

GPTEXT：Greenplum + Apache Solr

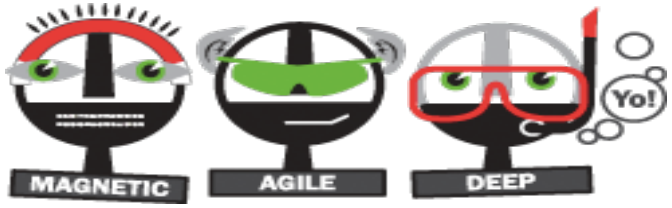
每次搜索基于相关性评分

基于文本挖掘和词性标注的NLP



GREENPLUM DATABASE

可扩展的，数据库内的机器学习



Apache MADlib: Big Data Machine Learning in SQL

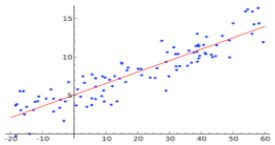
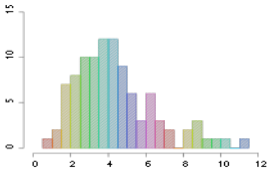
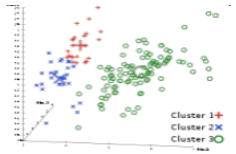
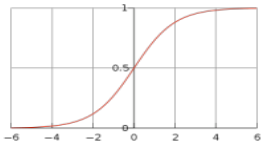
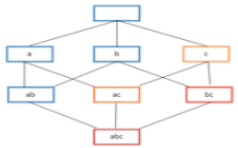
Apache顶级开源项目

PostgreSQL & Greenplum

为数据科学家提供强大的机器学习，图计算，和统计分析能力

- 开源地址 https://github.com/apache/madlib
- 下载和文档 http://madlib.apache.org/
- Wiki https://cwiki.apache.org/confluence/display/MADLIB/

MADlib 函数簇

Supervised Learning
Neural Networks
Support Vector Machines (SVM)
Conditional Random Field (CRF)
Regression Models
- Clustered Variance
- Cox-Proportional Hazards Regression
- Elastic Net Regularization
- Generalized Linear Models
- Linear Regression
- Logistic Regression
- Marginal Effects
- Multinomial Regression
- Naïve Bayes
- Ordinal Regression
- Robust Variance

Tree Methods
- Decision Tree and Random Forest

Unsupervised Learning
Association Rules (Apriori)
Clustering (k-Means)
Principal Component Analysis (PCA)
Topic Modelling (Latent Dirichlet Allocation)

Nearest Neighbors
- k-Nearest Neighbors

Time Series Analysis
- ARIMA

Deep Learning
Keras Fit/Evaluate/Predict
Load Model Architectures
Preprocessor for Images
Parallel Image Loading

Graph
All Pairs Shortest Path (APSP)
Breadth-First Search
Hyperlink-Induced Topic Search (HITS)
Average Path Length
Closeness Centrality
Graph Diameter
In-Out Degree
PageRank and Personalized PageRank
Single Source Shortest Path (SSSP)
Weakly Connected Components

Utility Functions
Columns to Vector
Conjugate Gradient
Linear Solvers
- Dense Linear Systems
- Sparse Linear Systems

Mini-Batching
PMML Export
Term Frequency for Text
Vector to Columns

Data Types and Transformations
Array and Matrix Operations
Matrix Factorization
- Low Rank
- Singular Value Decomposition (SVD)

Norms and Distance Functions
Sparse Vectors
Encoding Categorical Variables
Path Functions
Pivot
Sessionize
Stemming

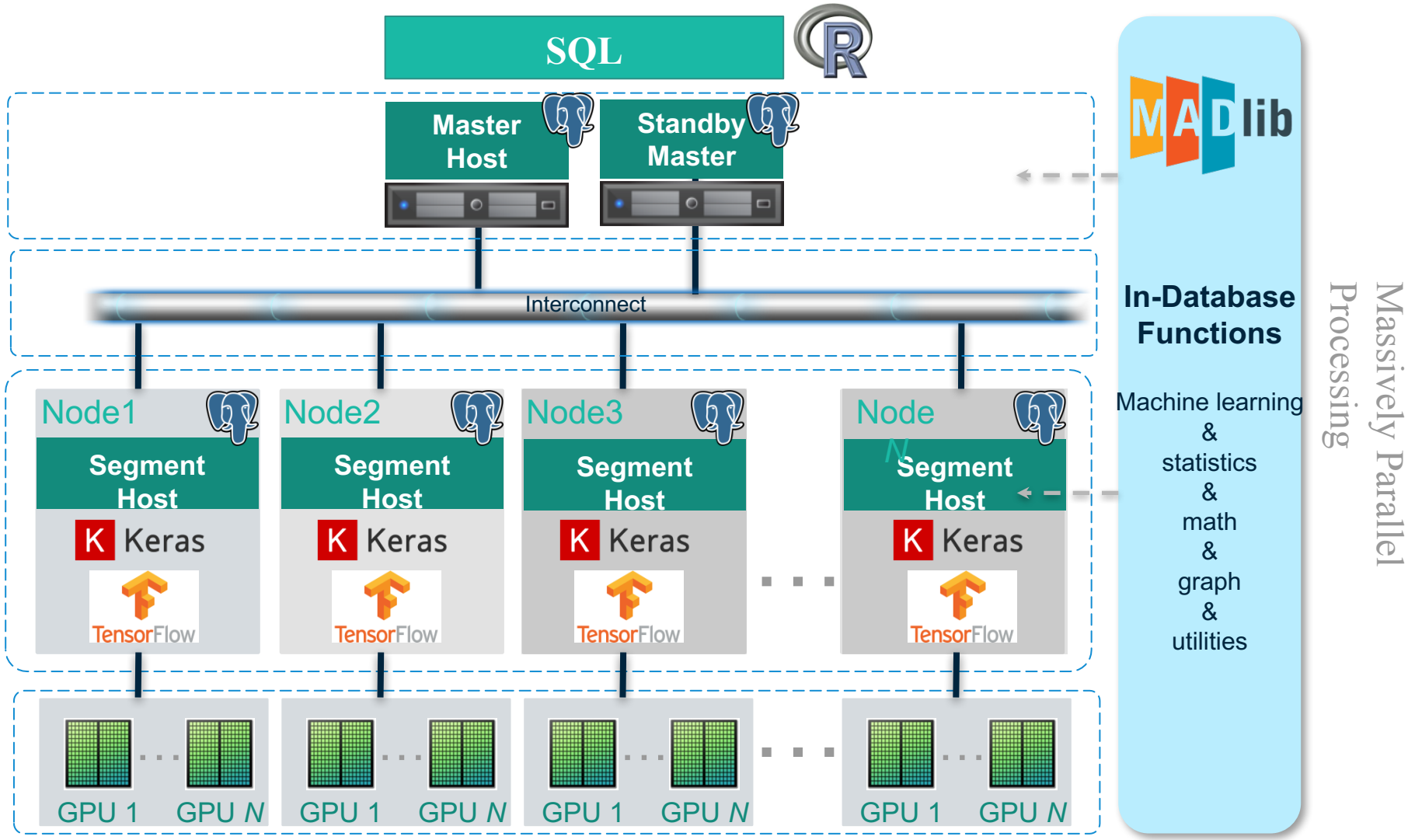
Statistics
Descriptive Statistics
- Cardinality Estimators
- Correlation and Covariance
- Summary

Inferential Statistics - Hypothesis Tests
Probability Functions

Model Selection
Cross Validation
Prediction Metrics
Train-Test Split

Sampling
Balanced
Random
Stratified

基于Greenplum 并行深度学习框架



同时利用 GPU 执行与 CPU 执行

线性可扩展性

支撑完备的数据科学负载:
数据准备,特征提取,机器学习,地理信息, 文本挖掘, 深度学习, 等

简单易用的算法接口

GREENPLUM DATABASE

训练模型

```
SELECT madlib.linregr_train( 'houses',                        -- Historical prices
                             'houses_linregr_bedroom',        -- Output model table
                             'price',                         -- Variable to predict
                             'ARRAY[1, tax, bath, size]',      -- Features
                             'bedroom'                        -- Diff models by #bedrooms
                           );
```

模型预测

```
SELECT houses_test.*,
       madlib.linregr_predict( model.coef,                -- Trained model
                               ARRAY[1,tax,bath,size]     -- Features
                              ) as predicted_price
FROM houses_test, houses_linregr_bedroom as models
WHERE houses_test.bedroom = model.bedroom;
```

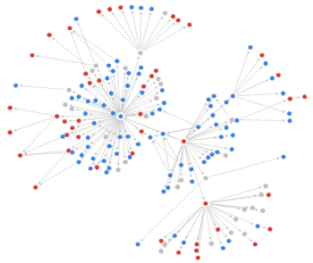
大规模图计算问题

GREENPLUM DATABASE

Social Network

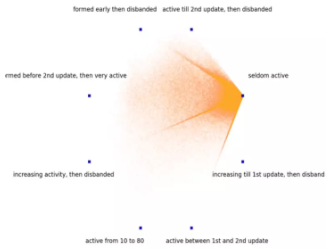
* Grandjean, M. (2016)

Epidemiology



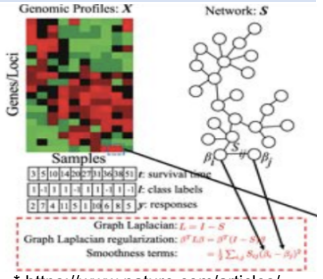
* http://www.netminer.com/community

MMO Role-Playing Game



* www.researchgate.net

Chemistry

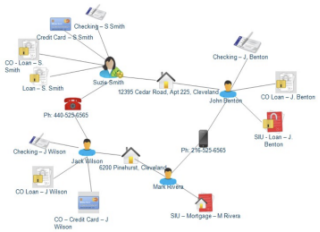


* https://www.nature.com/articles/

Bank Risk

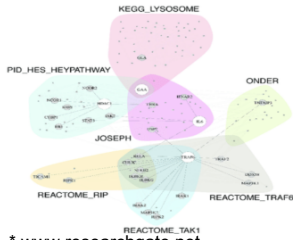
* https://cambridge-intelligence.com

1st Party Fraud



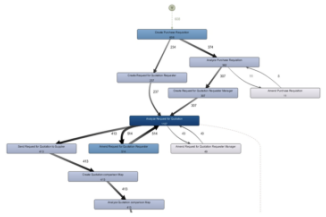
* www.infoglide.com

Gene



* www.researchgate.net

Manufacturing



* https://blog.trifinance.com

利用 **MADlib** 基于 **Greenplum** 进行大规模图计算

- 为超大图设计 (数以亿计的边和顶点)
- 数据存储在 **Greenplum** 中利用熟悉的 **SQL** 接口操作

- 直接在Greenplum中支持，不需要额外购买图引擎
- 不需要将原数据导出或修改
- 一个数据库内进行部署和管理
- 简化备份恢复过程以及DBA的工作

Graph Algorithms and Measures

	Types	Question	Features
Graph-based Features	1 Group	*"What are the sub-graphs, component, communities?"*	weakly-connected component
	2 Structure	*"What is the character of the network structure?"*	Density, Diameter, Average path length, Modularity
	3 Path	*"What is the shortest path (distance) among vertices?"*	Single source shortest path, All pairs shortest path, Breadth-First Search
	4 Centrality	*"What are the most important vertices within a graph?"*	Degree (in/out, weight), Closeness, PageRank, Hub, Authority, Betweenness, Clustering coefficient

Roadmap

Postgres 9.5 升级完成

GREENPLUM DATABASE

将会包含在Greenplum 7.0里

- Upsert (Insert on Conflict)
- 行级安全管理
- Block Range Indices
- 排序性能大幅提升
- 多CPU机器性能大幅提升

Postgres 9.6 升级总体完成

GREENPLUM DATABASE

将会包含在Greenplum 7.0里

- 并行执行顺序扫描，Join和聚合
- 避免在Vacuum操作期间不必要地扫描存储页
- 全文检索具备搜索短语能力（多个相邻单词）
- postgres_fdw支持远程Join，排序，更新和删除
- 大幅提升性能，尤其是在多CPU插槽服务器上的可扩展性方面

多数据中心复制

GREENPLUM DATABASE

将会包含在Greenplum 7.0里

- 利用GPDB 6中实现的预写日志
- 支持故障转移和故障恢复
- 最大化数据可用性
- 支持冷热待机
- 在现有的网络设置中可以工作
- 只需要配置一次

跨集群联合查询

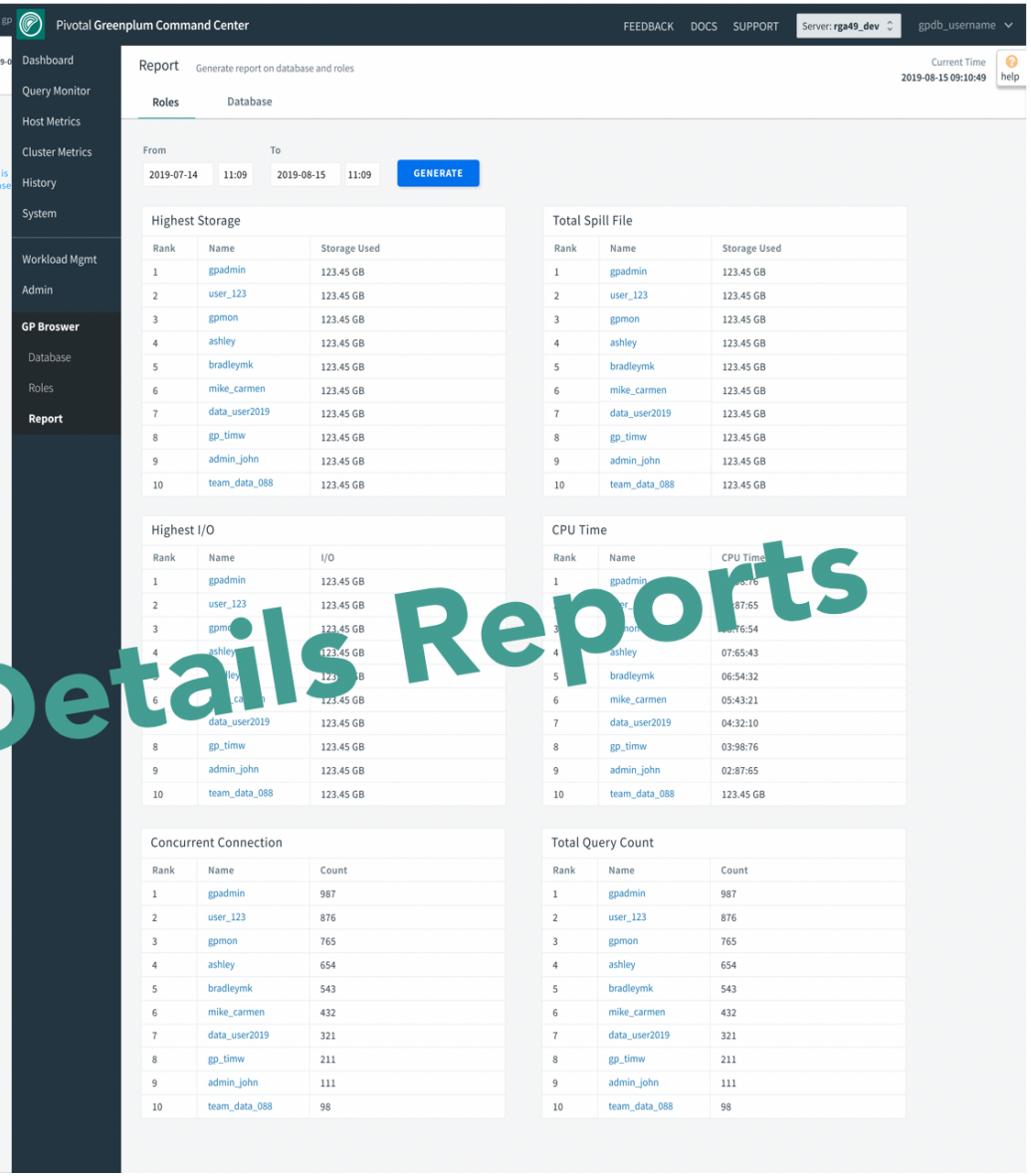
GREENPLUM DATABASE

将会包含在Greenplum 7.0里

- 让用户可以快速可靠地跨Greenplum集群查询
- “扩大”Greenplum集群规模
- 查询优化器倾力打造：
  - 减少数据冗余
  - 支持跨集群表间Join
  - 支持谓词下推，减少数据网络传输

GPCC更强的警报功能和更加智能的查询推荐

GREENPLUM DATABASE



GPCC Object Details Reports

让R并行起来

典型场景：

小明同学是R语言的专家，擅长统计学和数据建模

他写了很多R的代码，包括一些库和常用的工具

他对R的生态非常了解

但是，，，，

当处理上T或者上P的数据时，他需要之前写的R代码能够继续使用

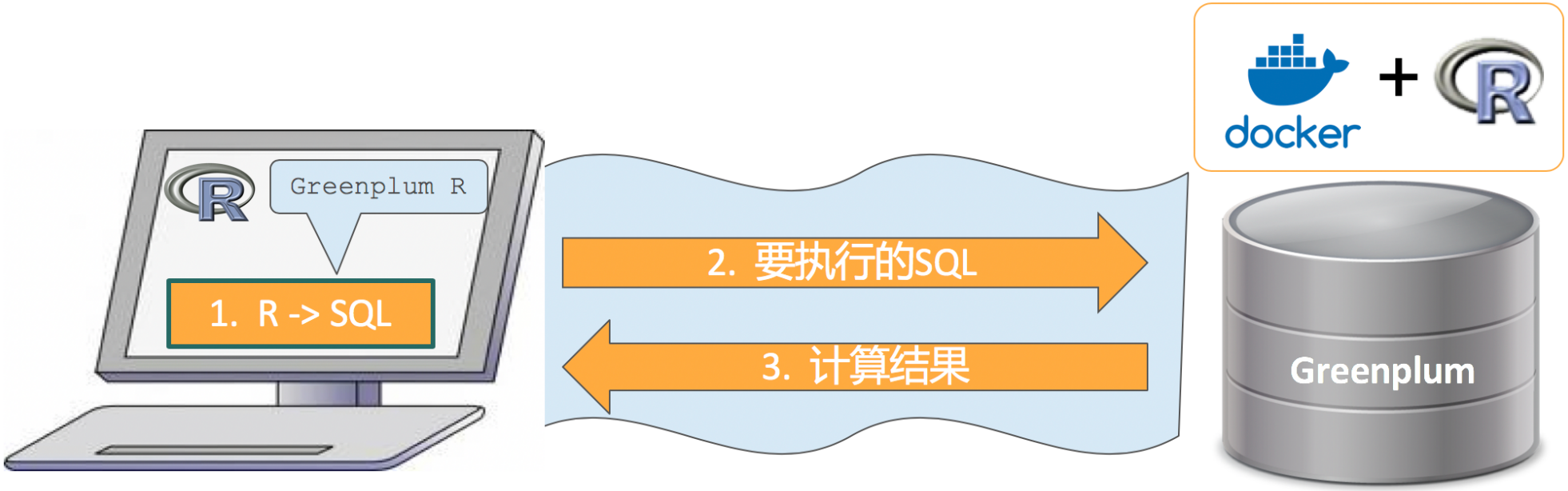
数据需要存放在统一的位置并确保安全

他需要一种办法能够和其它用户共享模型和代码

他希望能像以前一样单机写代码，但是并行的跑起来

Greenplum R

- R开发者在本地编写代码
- R函数推送到GP服务器
- 在Docker容器中并行执行R函数
- R函数在GP处理数据
- 结果存储在数据库中或者返回给客户端



- 数据无需离开GP
- 将代码推送到GP

Greenplum：面向未来打造的新一代数据平台

GREENPLUM DATABASE

- A Relational Database
- A Massively Parallel Object Relational Database
- An Analytics Database
- A Transactional Database
- A Columnar Database
- A Traditional Row Oriented Database
- A Cloud Native Database
- A Graph Database at Mass Scale
- A Geospatial Database
- A Time Series Database
- A Structured and Semi-Structured Platform leveraging ANSI-SQL
- An Image Recognition Engine
- An R and Python High Performance Computing Cluster (HPC)

Q & A

GREENPLUM DATABASE®

GREENPLUM DATABASE®

扫描加入 Greenplum 技术讨论群

微信群组

QQ 群组

欢迎订阅 Greenplum中文社区：greenplum.cn