新一代 PB 级分布式 HTAP 数据库

Greenplum 能做什么？

姚延栋
Greenplum 中国研发中心副总 &
Greenplum 中文社区创始人
yyao@pivotal.io

Pivotal

内容

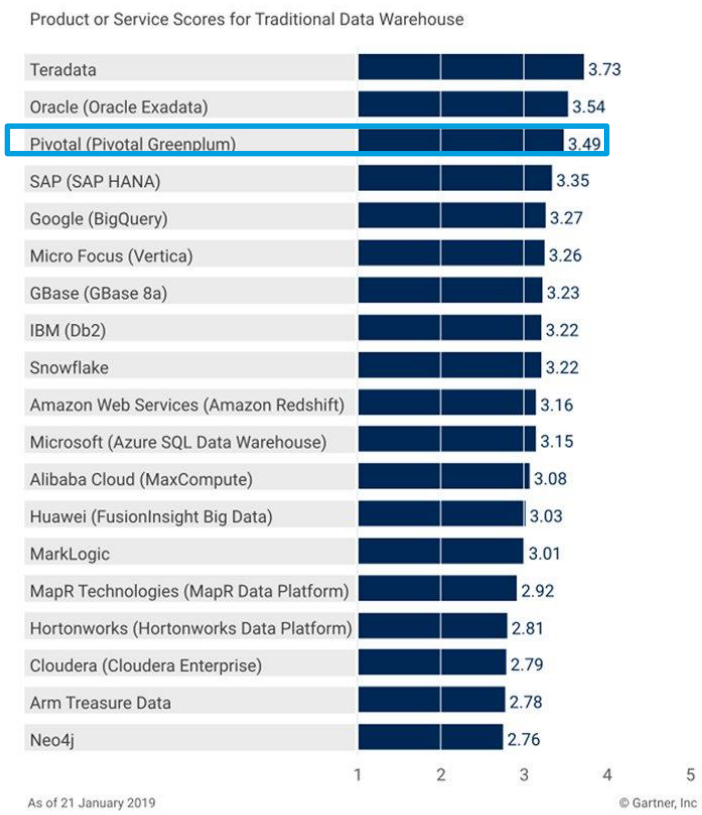
- 数仓/OLAP/即席查询
- 混合负载/HTAP
- 流数据
- 集成数据分析
- 数据库内嵌机器学习
- 现代 SQL

数据仓库、OLAP、即席分析

分析能力卓越：Gartner 2019评测世界第三

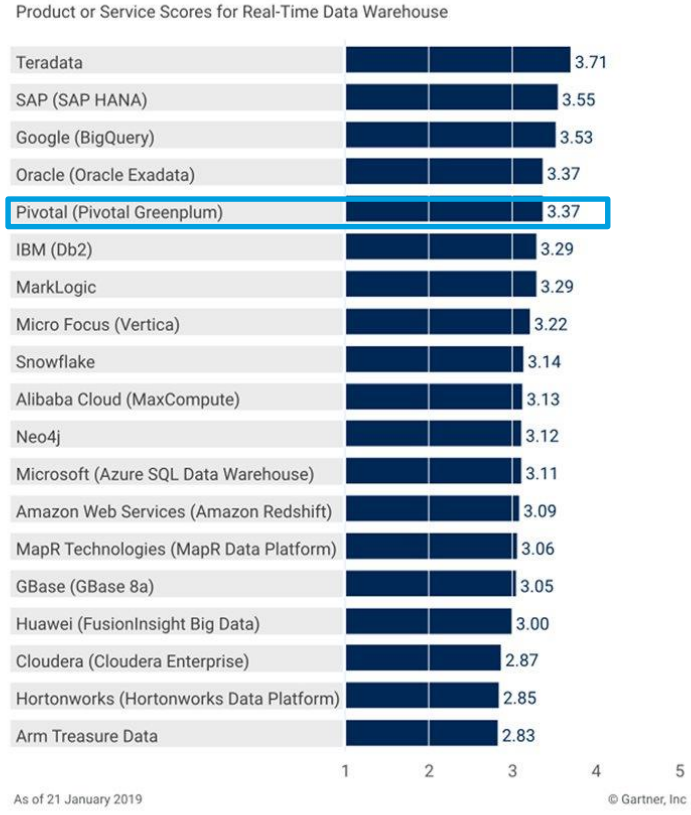
GREENPLUM DATABASE

Figure 1. Vendors' Product Scores for Traditional Data Warehouse Use Case



Source: Gartner (March 2019)

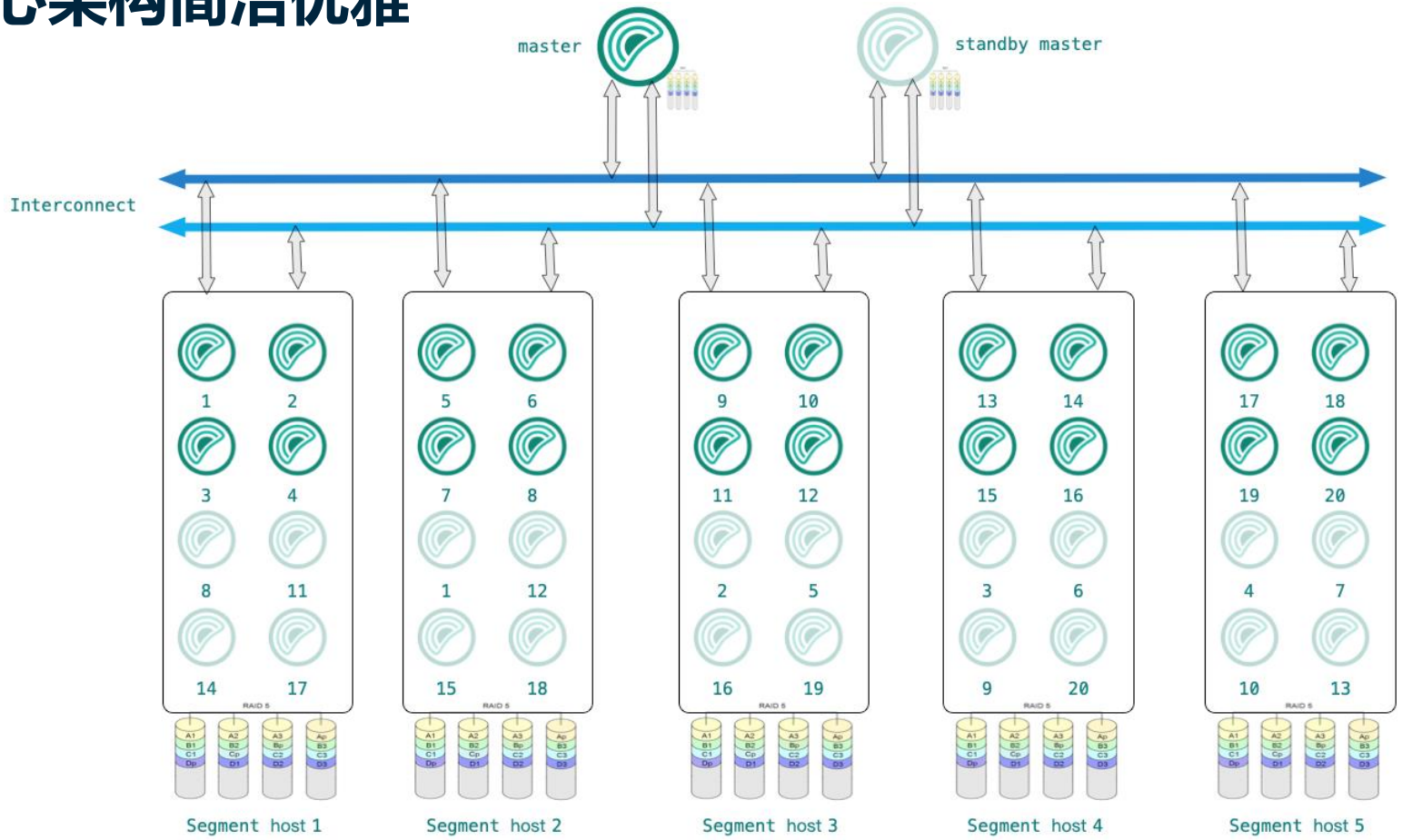
Figure 2. Vendors' Product Scores for Real-Time Data Warehouse Use Case



Source: Gartner (March 2019)

Pivotal

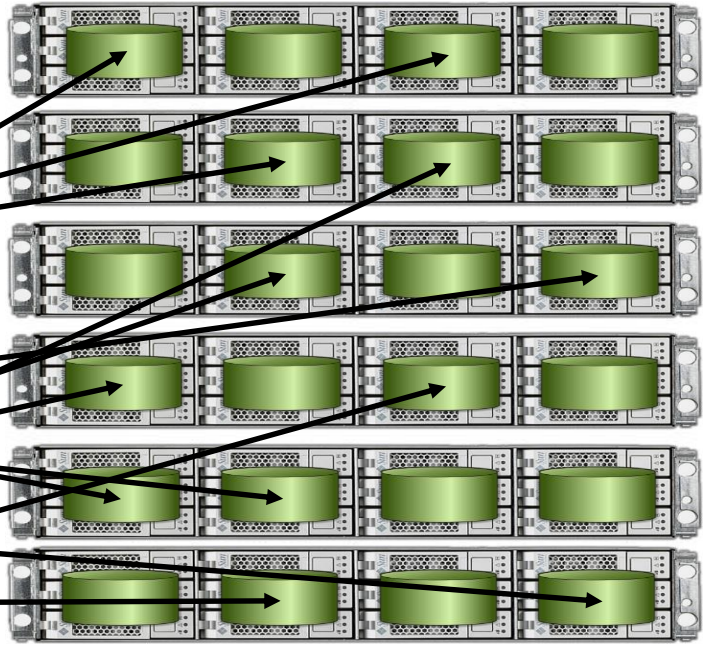
核心架构简洁优雅


master
standby master
Interconnect
1 2 3 4 8 11 14 17
5 6 7 8 1 12 15 18
9 10 11 12 2 5 16 19
13 14 15 16 3 6 9 20
17 18 19 20 4 7 10 13
Segment host 1
Segment host 2
Segment host 3
Segment host 4
Segment host 5

数据分布：每个节点 1/n 数据

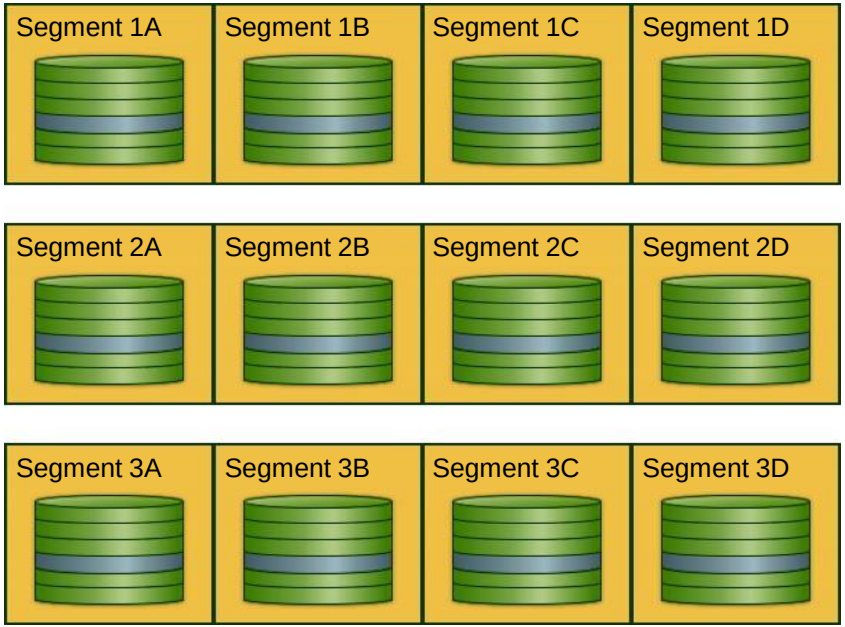
最重要的策略和目标是均匀分布。

Order		
Order #	Order Date	Customer ID
43	Oct 20 2005	12
64	Oct 20 2005	111
45	Oct 20 2005	42
46	Oct 20 2005	64
77	Oct 20 2005	32
48	Oct 20 2005	12
50	Oct 20 2005	34
56	Oct 20 2005	213
63	Oct 20 2005	15
44	Oct 20 2005	102
53	Oct 20 2005	82
55	Oct 20 2005	55



多级分区

```
SELECT COUNT(*)
  FROM orders
 WHERE order_date >= 'Oct 20 2007'
   AND order_date <  'Oct 27 2007'
```



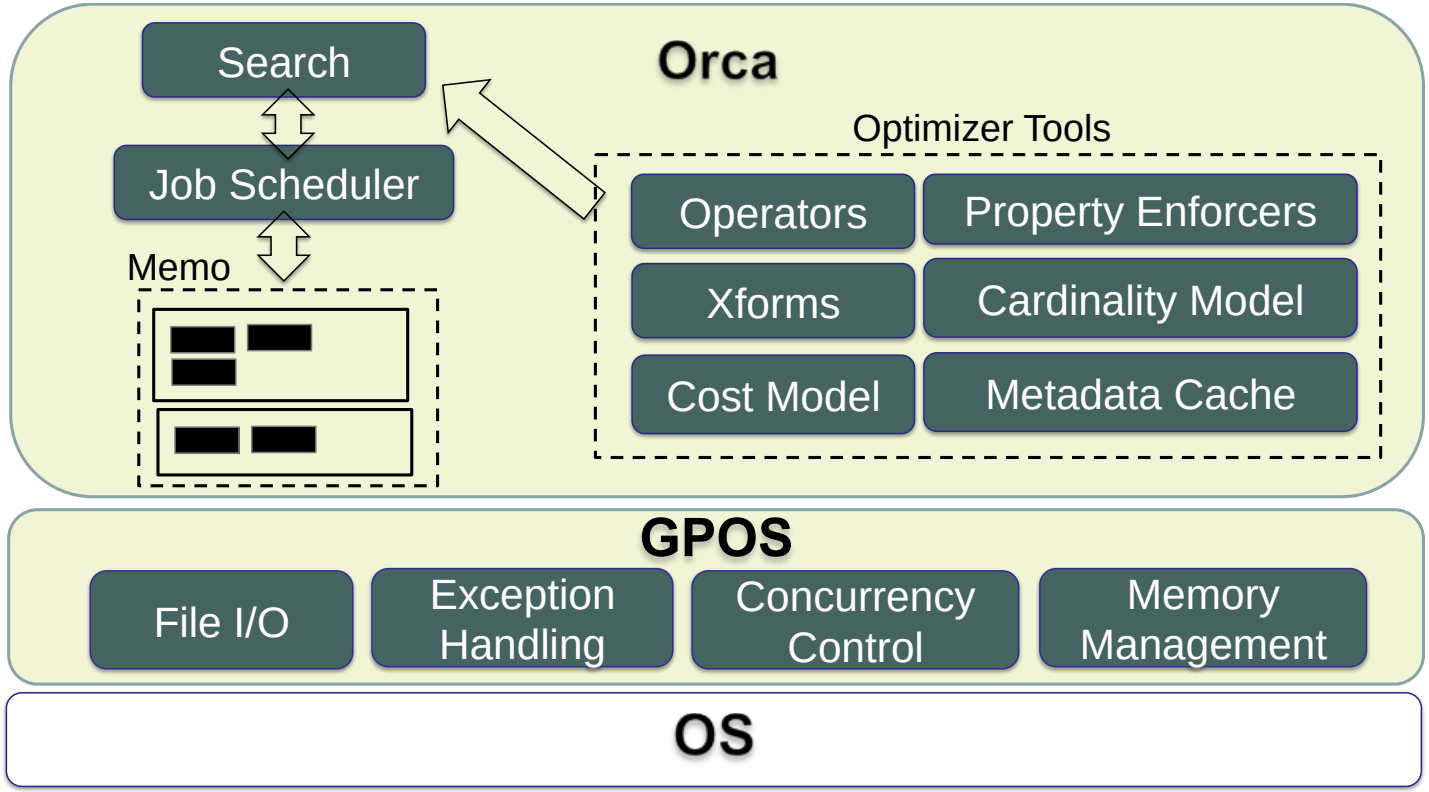
多模存储/多态存储

- 适合 OLTP 业务
- 适合频繁更新或者访问大部分字段的场景

- 列存储更适合压缩
- 查询列子集时速度快
- 不同列可以使用不同压缩方式: gzip (1-9), quicklz, delta, RLE，zstd

- 历史数据和不常访问的数据存储在 HDFS 或者其他外部系统中
- 无缝查询所有数据
- Text，CSV，Binary，Avro，Parquet，ORC 格式

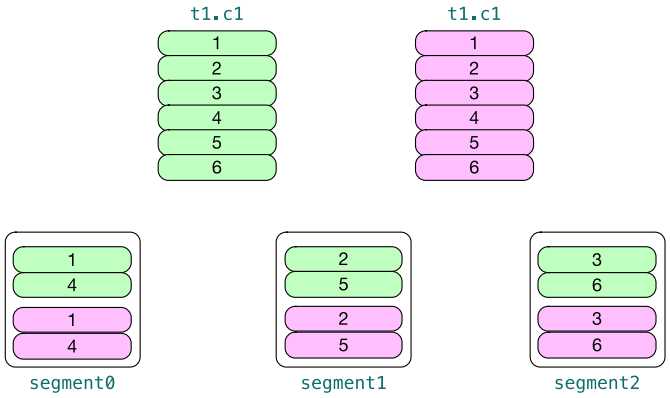
Apache ORCA: 专为复杂查询而生的优化器

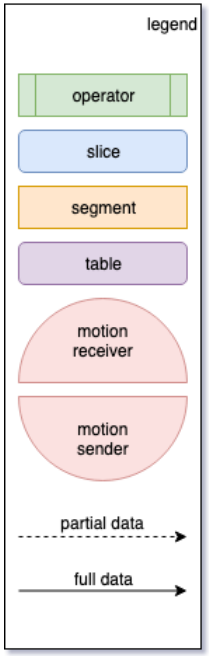


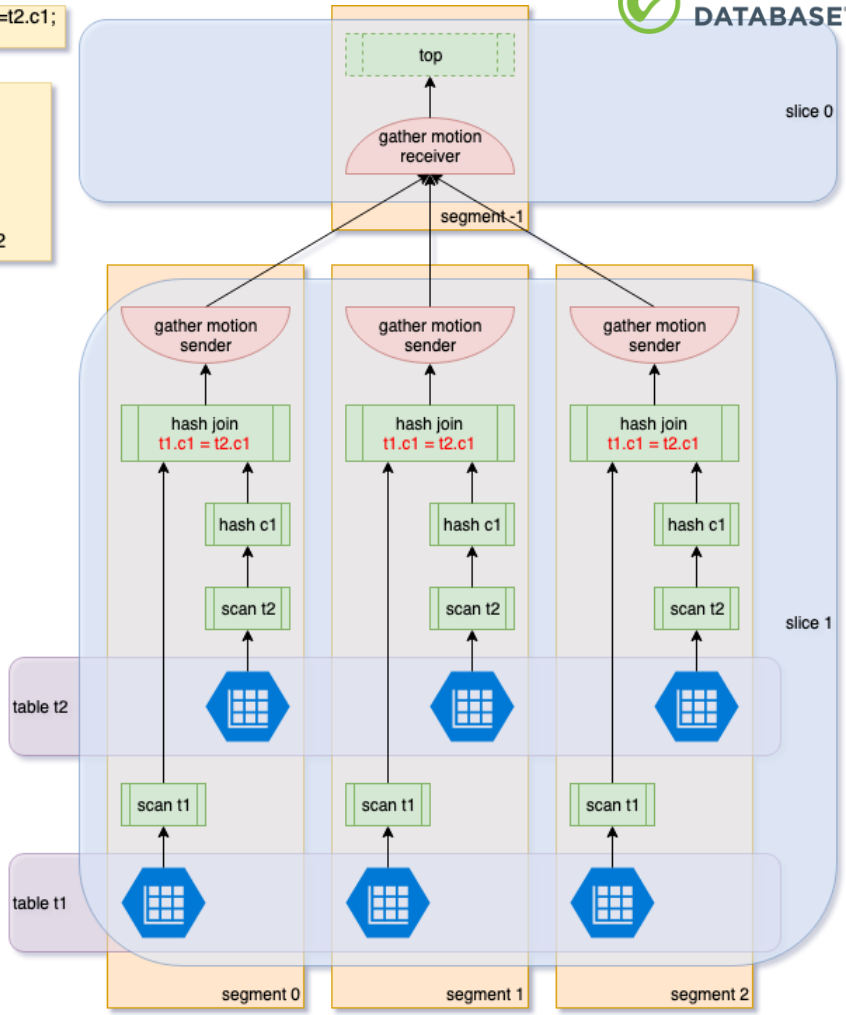
N 个节点并行执行

select * from t1 join t2 on t1.c1=t2.c1;
(Distribution key: t1.c1, t2.c1)

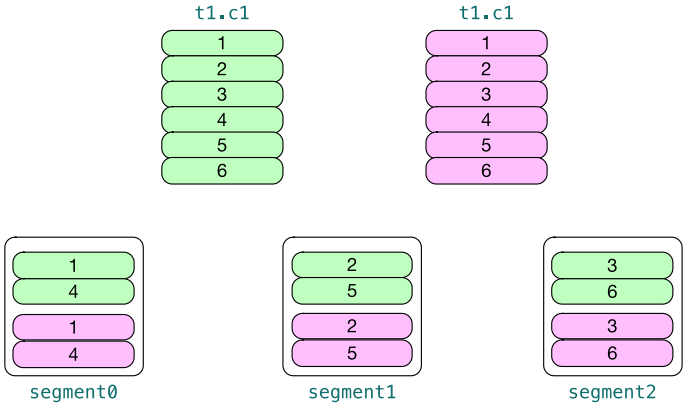
Gather Motion 3:1 (slice1; segments: 3)
-> Hash Join
Hash Cond: t1.c1 = t2.c1
-> Seq Scan on bar t1
-> Hash
-> Seq Scan on bar t2





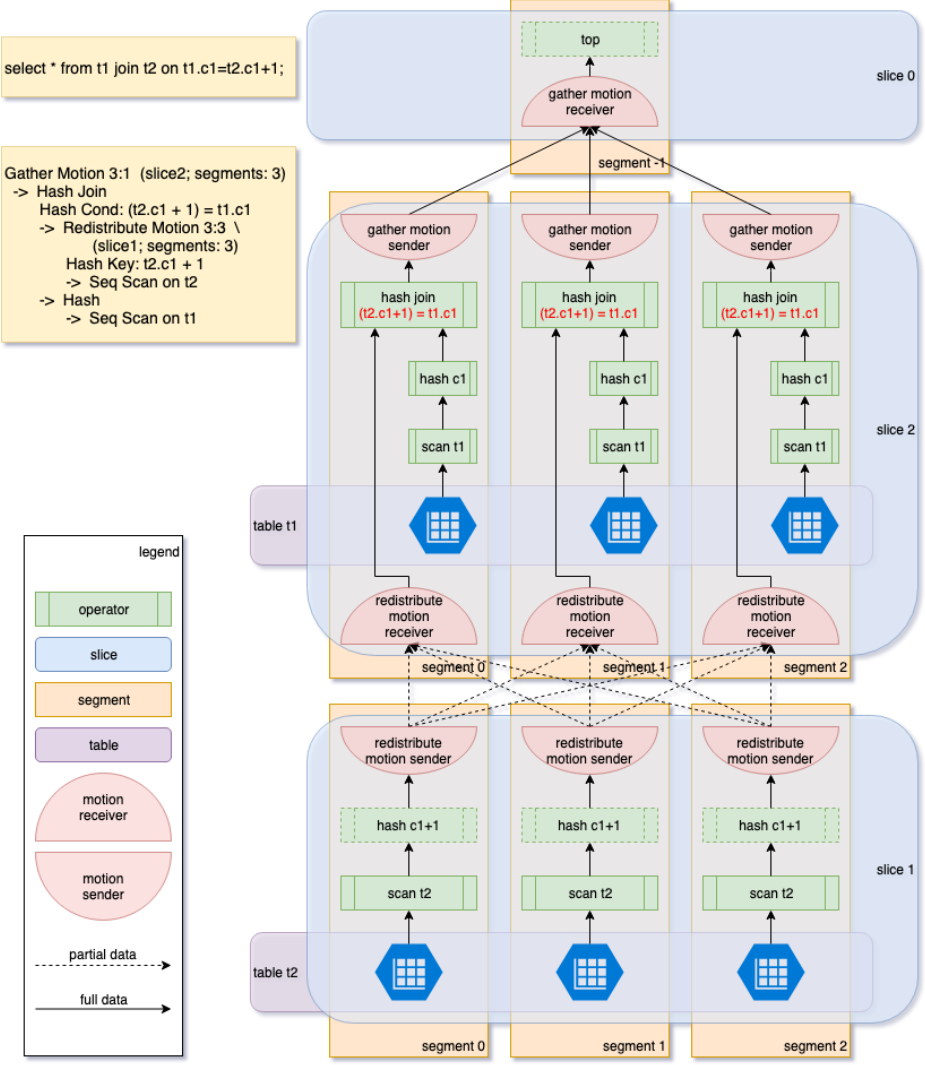


数据shuffle



select * from t1 join t2 on t1.c1=t2.c1+1;

```
Gather Motion 3:1  (slice2; segments: 3)
 ->  Hash Join
       Hash Cond: (t2.c1 + 1) = t1.c1
       ->  Redistribute Motion 3:3  \
             (slice1; segments: 3)
           Hash Key: t2.c1 + 1
           ->  Seq Scan on t2
       ->  Hash
           ->  Seq Scan on t1
```

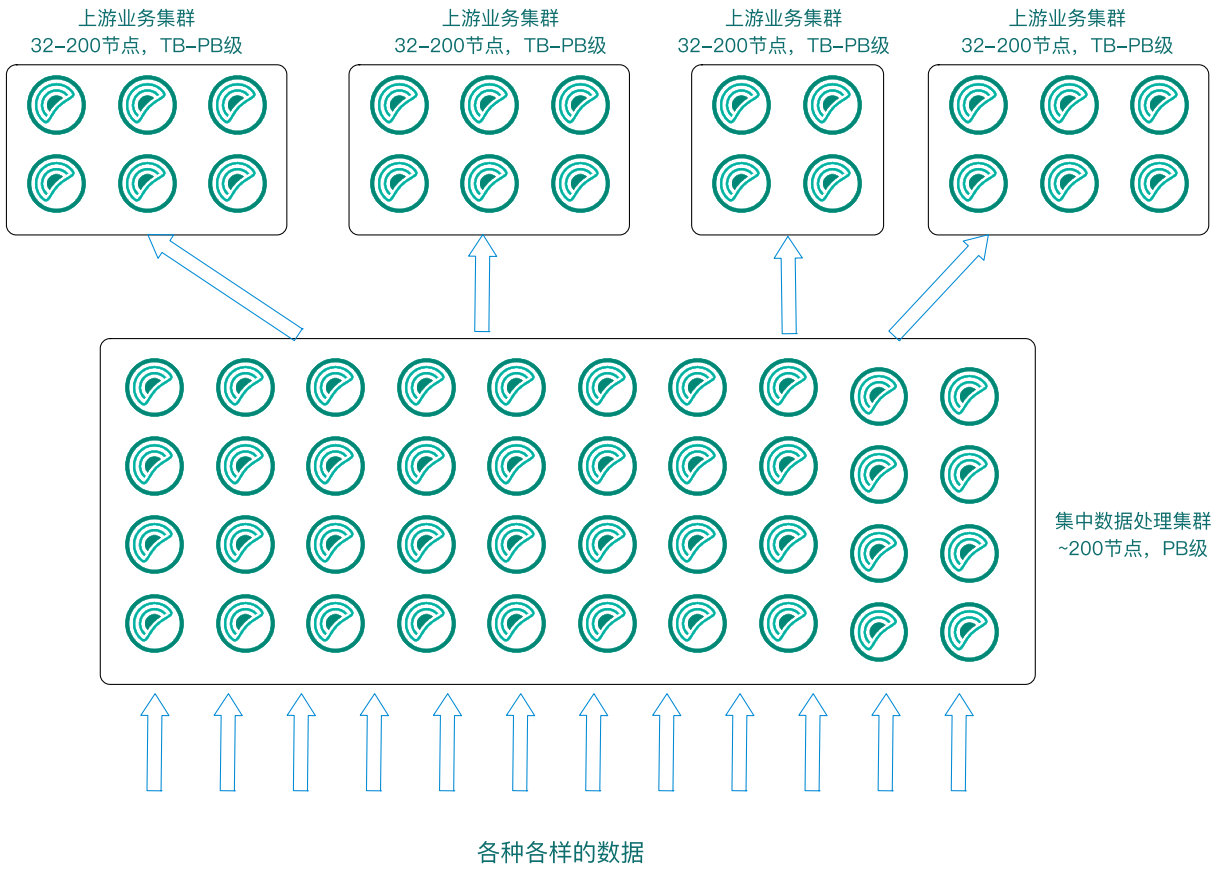


Pivotal

还有很多技术:

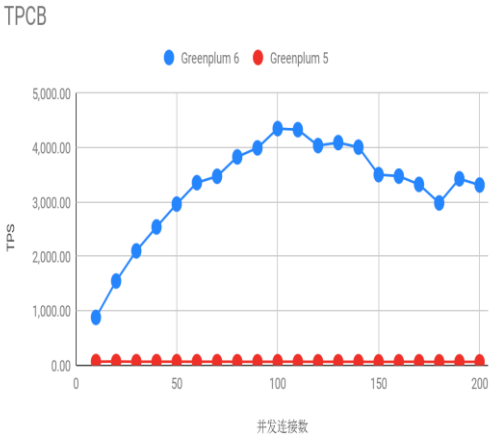
- 高效压缩算法
- 多阶段聚集
- 复制表
- Unlogged table
- 物化视图
- 一致性hash
- 在线扩容
- 安全性

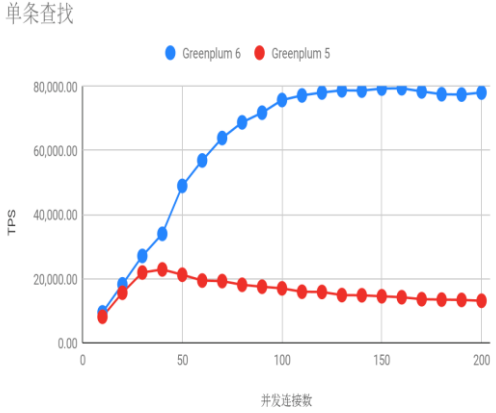
某世界顶级银行案例：中枢集群+上游业务集群

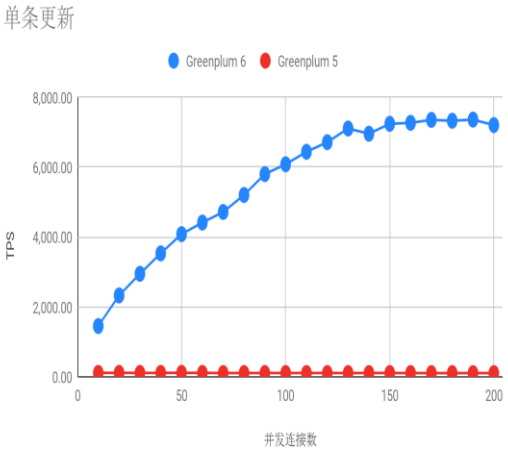


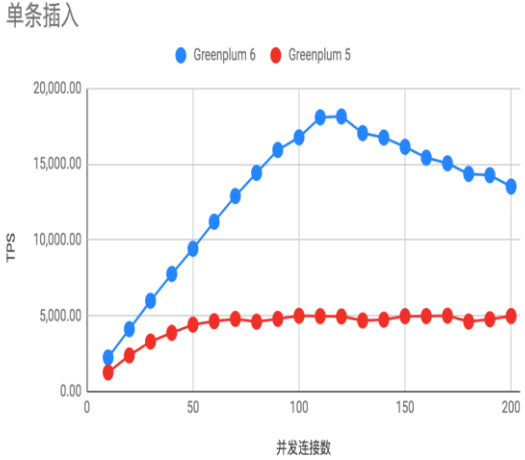
混合负载/HTAP

TPCB 60x；更新70x；单条查找和插入 3.5x









《Greenplum6 JDBC insert性能媲美MySQL》

- 信息来源：锐捷网络汤一波发表在 Pivotal研发中心微信公众号

MySQL 5.6	Greenplum 6.2.1
1926条/秒	2252条/秒

OLTP 优化技术

- 全局死锁检测（GDD）
- 锁优化
- 事务优化
- 复制表
- 多模存储
- 灵活索引
- OLTP 友好的优化器
- 内核升级：PostgreSQL 9.4

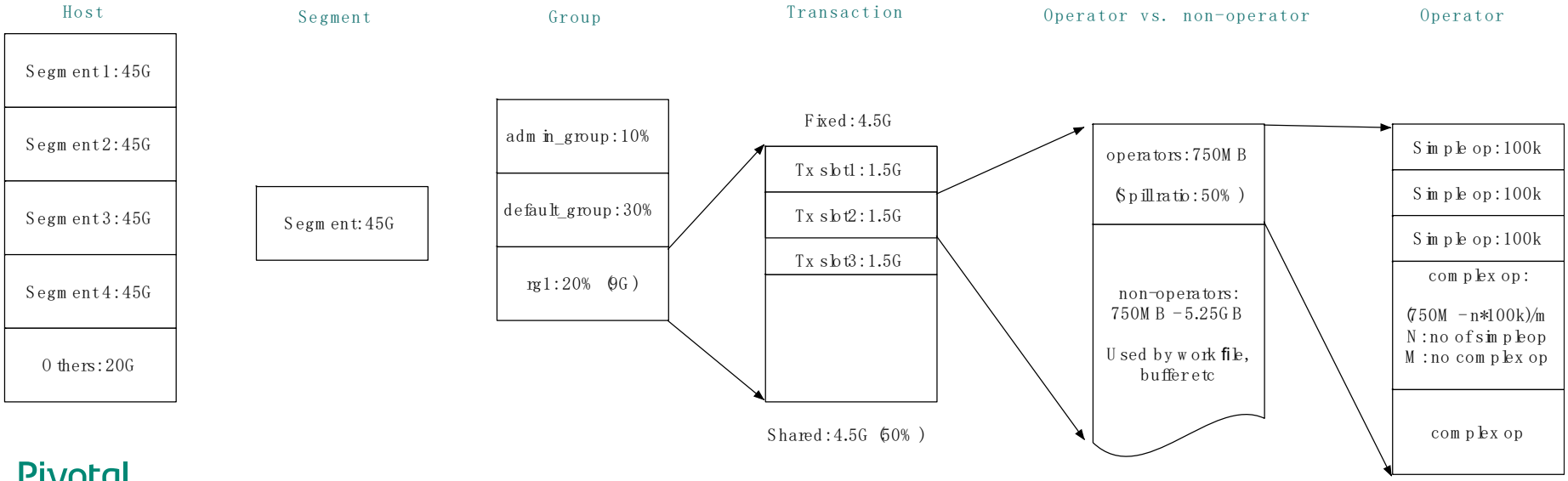
资源管理

特性	Resource group	Resource queue
并发控制单元	transaction	statement
并发控制死锁	no	yes
CPU 分配	Percentage, cgroup	Priority
CPU: Burst	yes	some
CPU: CPUSET 绑定	yes	no
Memory: 严格内存控制	yes	no
Memory: 组内内存共享	yes	no
Memory: 什么时候等待	No concurrency slot or memory quota	No concurrency slot
磁盘配额	是	否

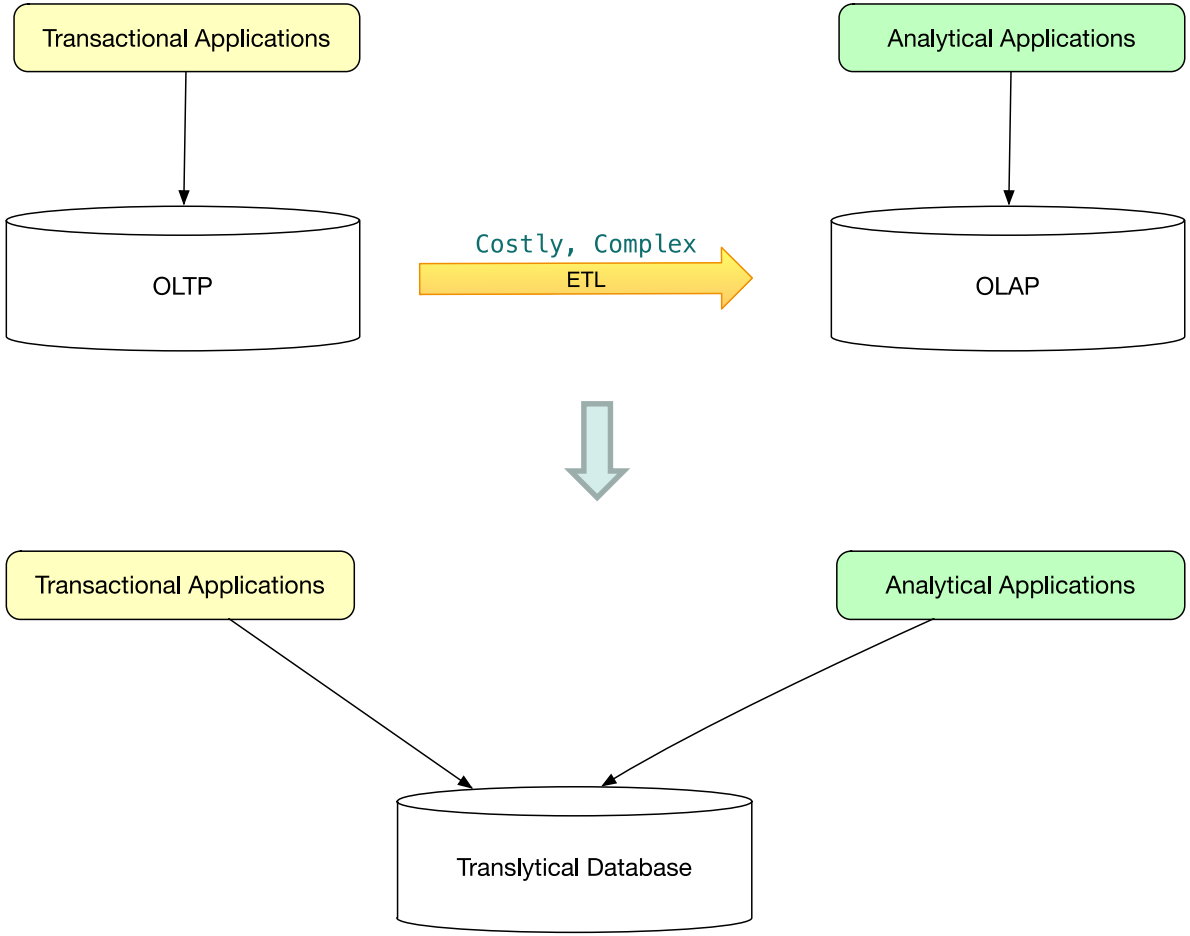
细粒度多级内存管理

Host Memory: 200G, 4 Segments
gp_resource_group_memory_limit: .9
Segment mem: 200*.9/4 = 45G

rg1:
Concurrency: 3
memory_limit: 20 —> 9G
memory_shared_quota: 50
memory_spill_ratio: 50

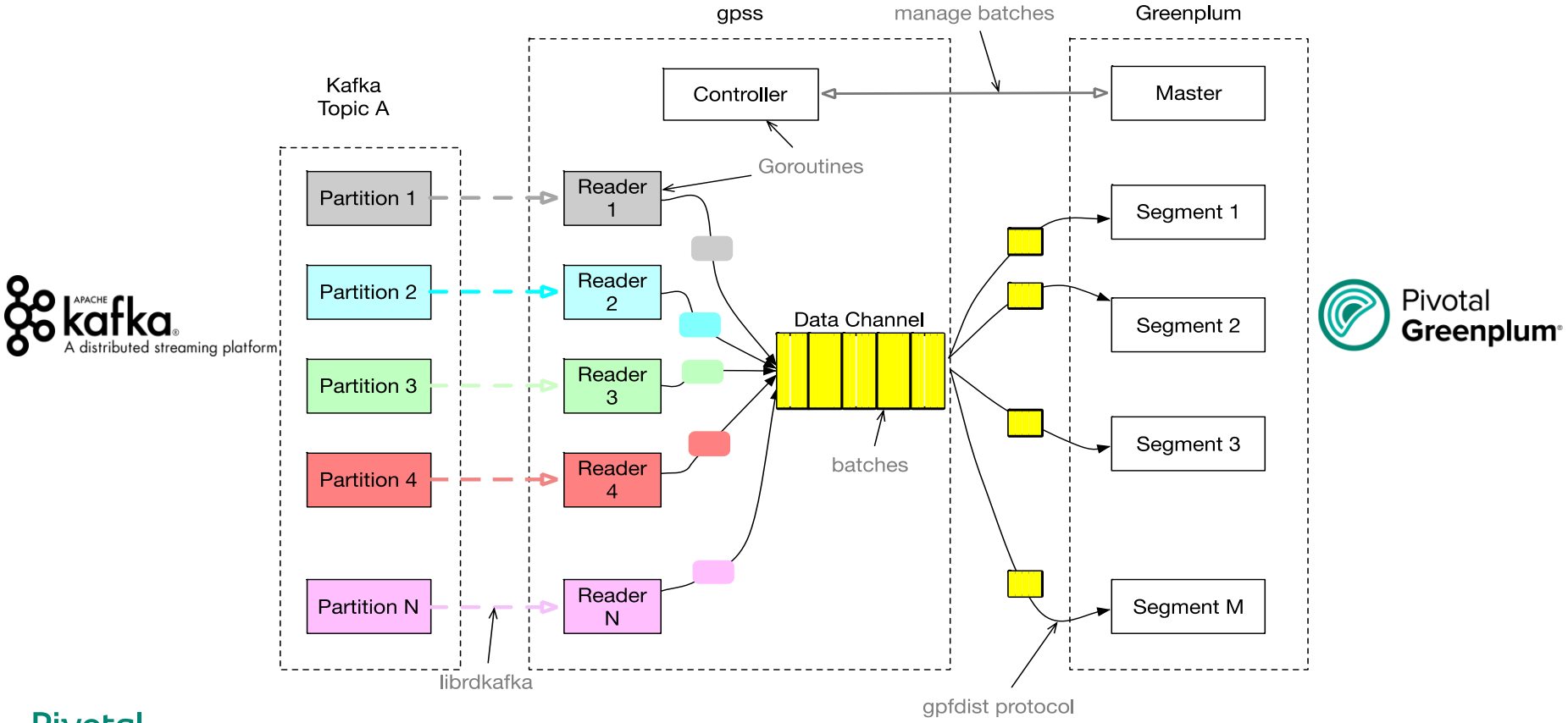


某互联网金融企业案例

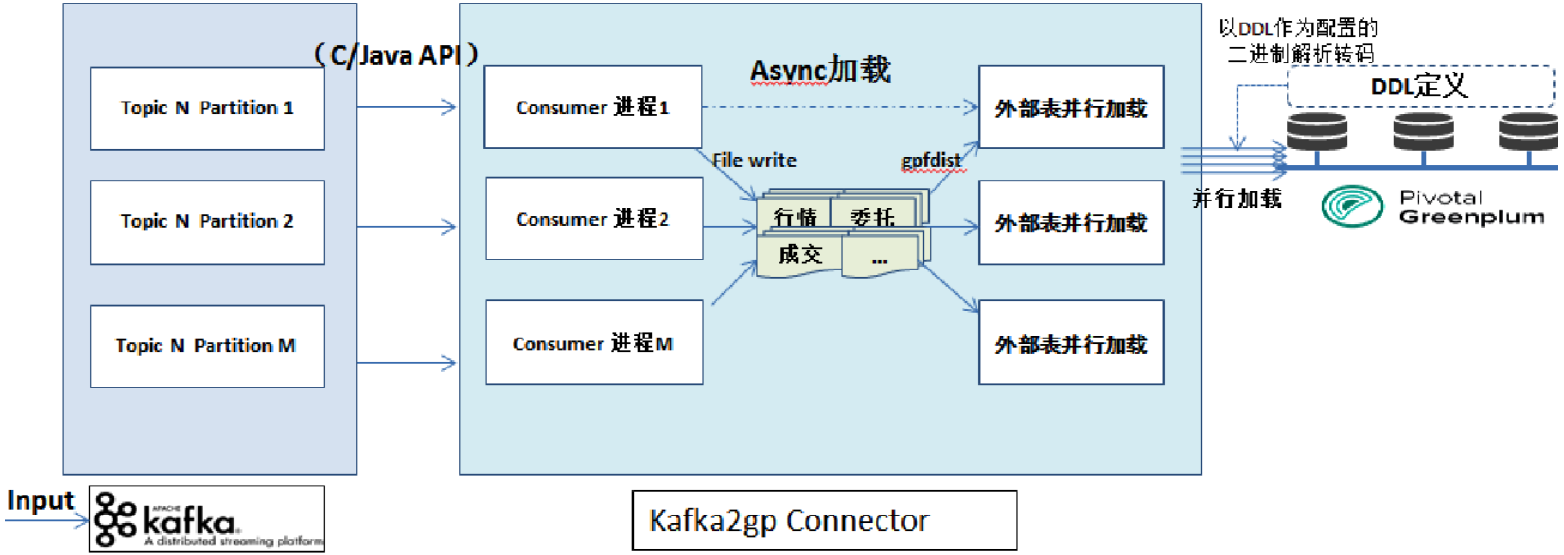


流数据、准实时

Greenplum Kafka Connector



某世界顶级证券交易所案例



数据量	机器数	表个数	索引个数	并发数	插入间隔	平均时延	最长时延	插入速度
9.8亿	18	4	12	16	500ms	170ms	1100ms	300万/s

集成数据分析

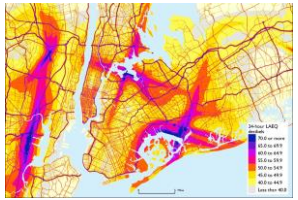
各种数据类型：结构化、半结构化、非结构化

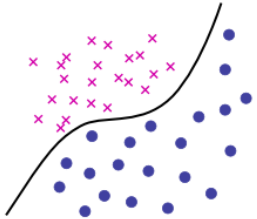
GREENPLUM DATABASE

Text

Traditional BI

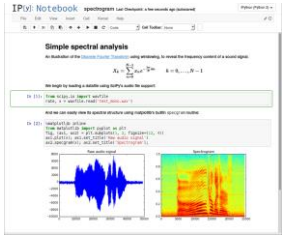
Geospatial

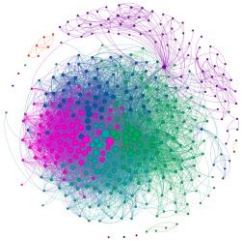




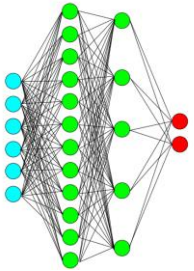
Machine Learning

Data Science Productivity Tools





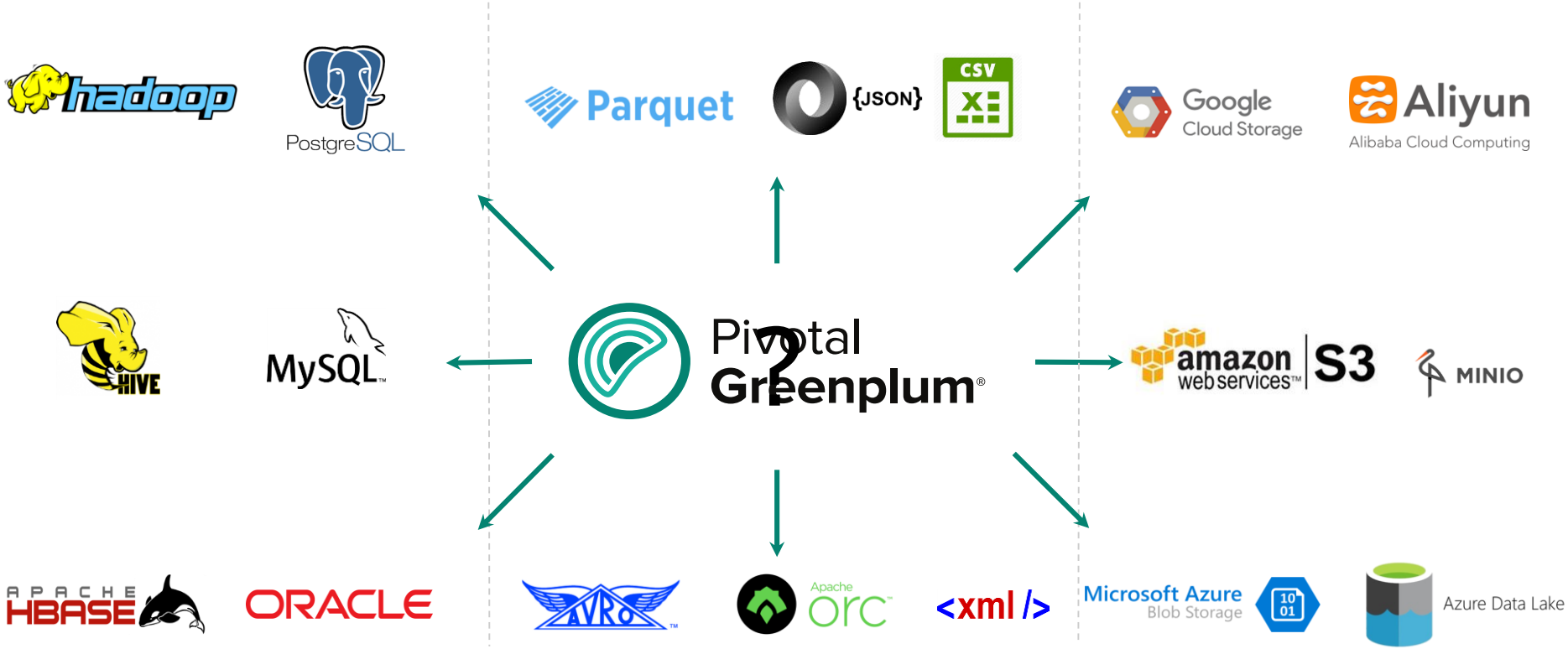
Graph



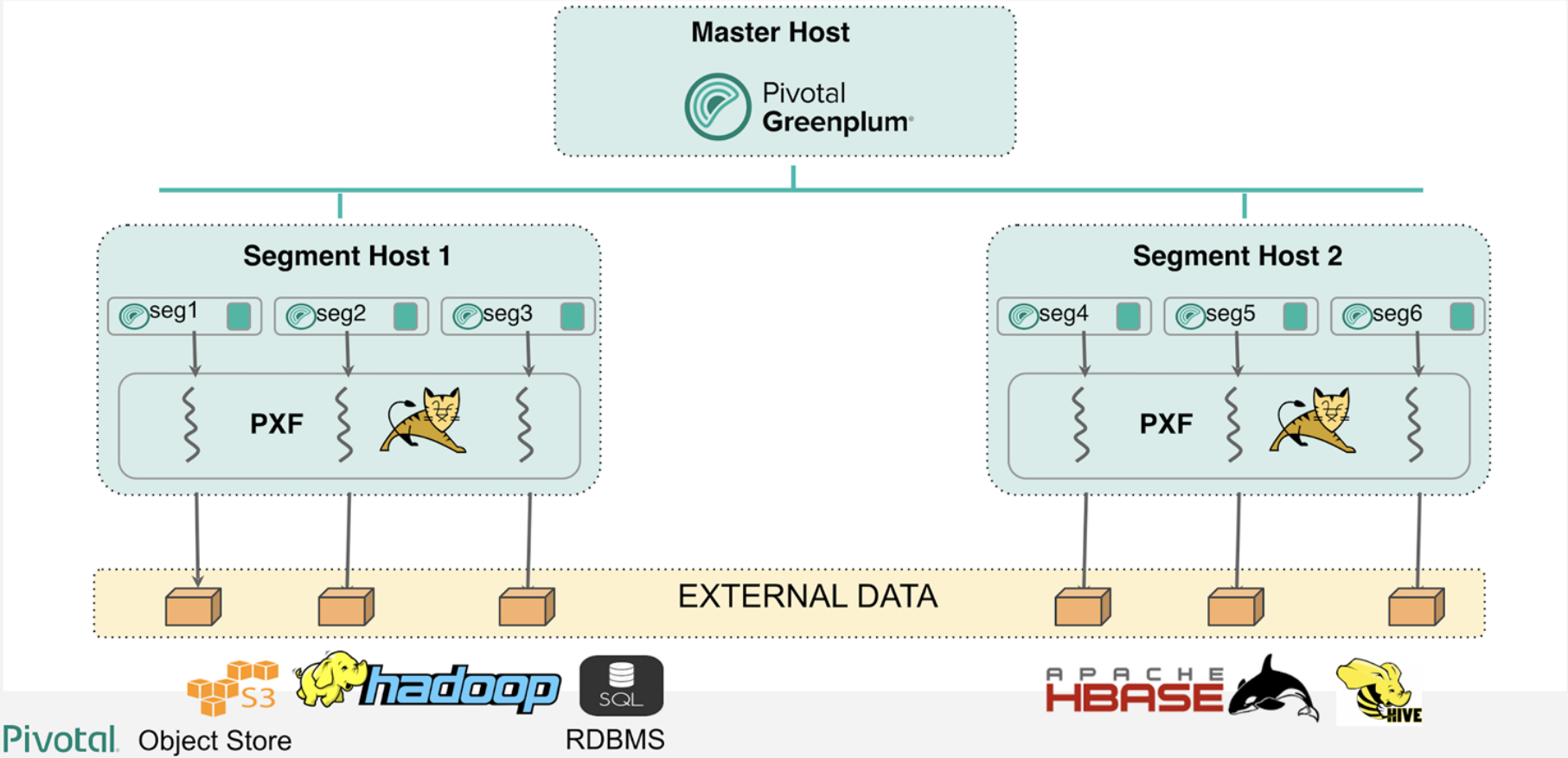
Deep Learning

Pivotal

数据融合


hadoop
PostgreSQL
Parquet
{JSON}
CSV
Google Cloud Storage
Aliyun
Alibaba Cloud Computing
HIVE
MySQL
Pivotal Greenplum
amazon web services
S3
MINIO
APACHE HBASE
ORACLE
AVRO
Apache orc
<xml />
Microsoft Azure Blob Storage
Azure Data Lake

PXF 架构


Master Host
Pivotal Greenplum
Segment Host 1
seg1
seg2
seg3
PXF
Segment Host 2
seg4
seg5
seg6
PXF
EXTERNAL DATA
S3
hadoop
SQL
APACHE HBASE
HIVE
Pivotal Object Store
RDBMS

问题描述

GREENPLUM DATABASE

"Find anyone who *works at Pivotal* and knows each other *directly* and whose names *sound* like 'Peter' or 'Pavan' and have *withdrawn an amount* > $200 *within 24 hours* at an ATM less than *2 KM from a reference* latitude and longitude"

Find anyone who works at 'Pivotal' and know each other 'directly' and whose names sound like 'Peter' or 'Pavan' and have withdrawn an amount > $200 within 24 hours at an ATM less than 2 KM from reference latitude and longitude.

```
drop function if exists get_people(text,text,integer,integer,float,float);
CREATE FUNCTION get_people(text,text,integer,integer,float,float) RETURNS integer
AS $$
declare
linkchk integer; v1 record; v2 record;
begin
 execute 'truncate table results;';
 for v1 in select distinct a.id,a.firstname,a.lastname,amount,tran_date,c.lat,c.lng,address,a.description,d.score from people a,transactions b,location c,
    (SELECT w.id, q.score FROM people w, gptext.search(TABLE(SELECT 1 SCATTER BY 1), 'gpadmin.public.people' , 'Pivotal', null) q
     WHERE (q.id::integer) = w.id order by 2 desc) d
     where soundex(firstname)=soundex($1) and a.id=b.id and amount > $3 and (extract(epoch from tran_date) - extract(epoch from now()))/3600 < $4

and st_distance_sphere(st_makepoint($5, $6),st_makepoint(c.lng, c.lat))/1000.0 <= 2.0 and b.locid=c.locid and a.id=d.id
 loop
   for v2 in select distinct a.id,a.firstname,a.lastname,amount,tran_date,c.lat,c.lng,address,a.description,d.score from people a,transactions b,location c,
    (SELECT w.id, q.score FROM people w, gptext.search(TABLE(SELECT 1 SCATTER BY 1), 'gpadmin.public.people' , 'Pivotal', null) q
     WHERE (q.id::integer) = w.id order by 2 desc) d
     where soundex(firstname)=soundex($2) and a.id=b.id and amount > $3 and (extract(epoch from tran_date) - extract(epoch from now()))/3600 < $4
    and st_distance_sphere(st_makepoint($5, $6),st_makepoint(c.lng, c.lat))/1000.0 <= 2.0 and b.locid=c.locid and a.id=d.id
 loop
     execute 'DROP TABLE IF EXISTS out, out_summary;';
     execute  'SELECT madlib.graph_bfs(''people'',''id'',''links'',NULL,'||v1.id||',''out'');'  ;
     select 1 into linkchk from out where dist=1 and id=v2.id;
     if linkchk is not null  then
          insert into results values (v1.id,v1.firstname,v1.lastname,v1.amount,v1.tran_date,v1.lat,v1.lng,v1.address,v1.description,v1.score);
          insert into results values (v2.id,v2.firstname,v2.lastname,v2.amount,v2.tran_date,v2.lat,v2.lng,v2.address,v2.description,v2.score);
      end if;
    end loop;
 end loop;
 return 0;
end
$$ LANGUAGE plpgsql;
--                  person1 , person 2, amount, duration in hours, longtitude, latitude (in question)
select get_people('Pavan','Peter',200,24,103.912680, 1.309432) ;
```

Greenplum Fuzzy String Match function **Soundex()** to know if people name sounds like 'Pavan' or 'Peter'

GPText.search() function is used to know if both people work at 'Pivotal'

Amount > $200

Greenplum and **Apache MADlib BFS** search to know if there are direct or indirect links between people

Greenplum **Time functions** to calculate difference in amount withdrawn time < 24 hours

Greenplum POSTGIS functions **st_distance_sphere() and st_makepoint()** calculate distance between ATM location and reference latitude, longitude < 2 KM

Pivotal

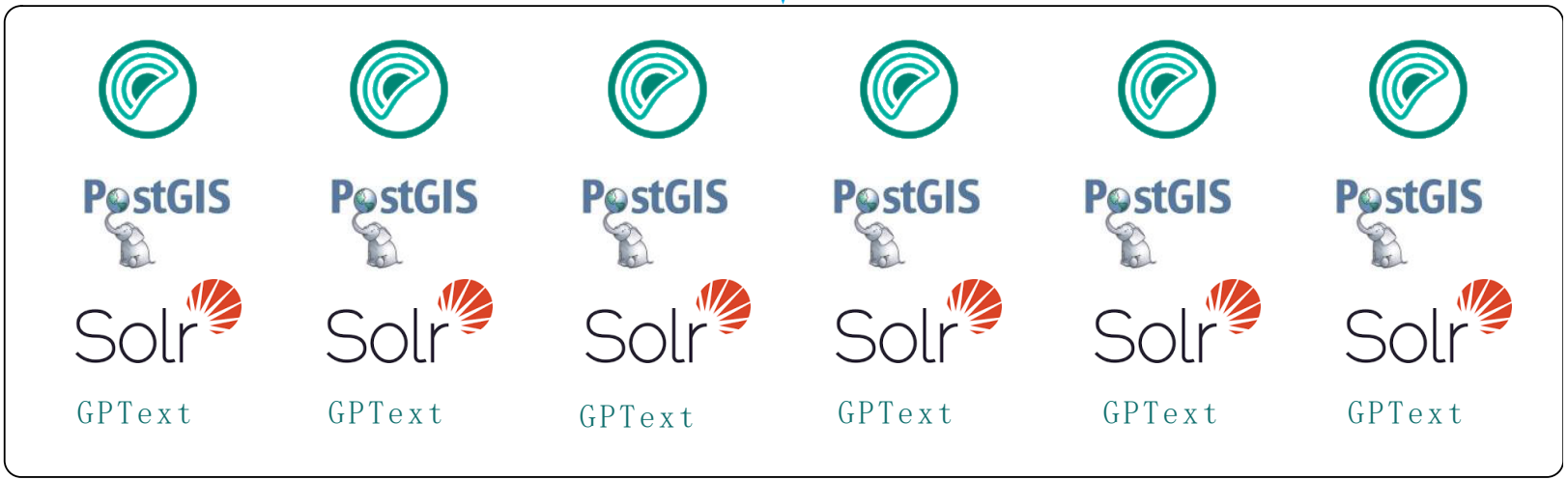
某大型服务提供商案例

GREENPLUM DATABASE

文本数据 +
地理信息数据 +
结构化数据

OLAP 操作
下钻、上卷等

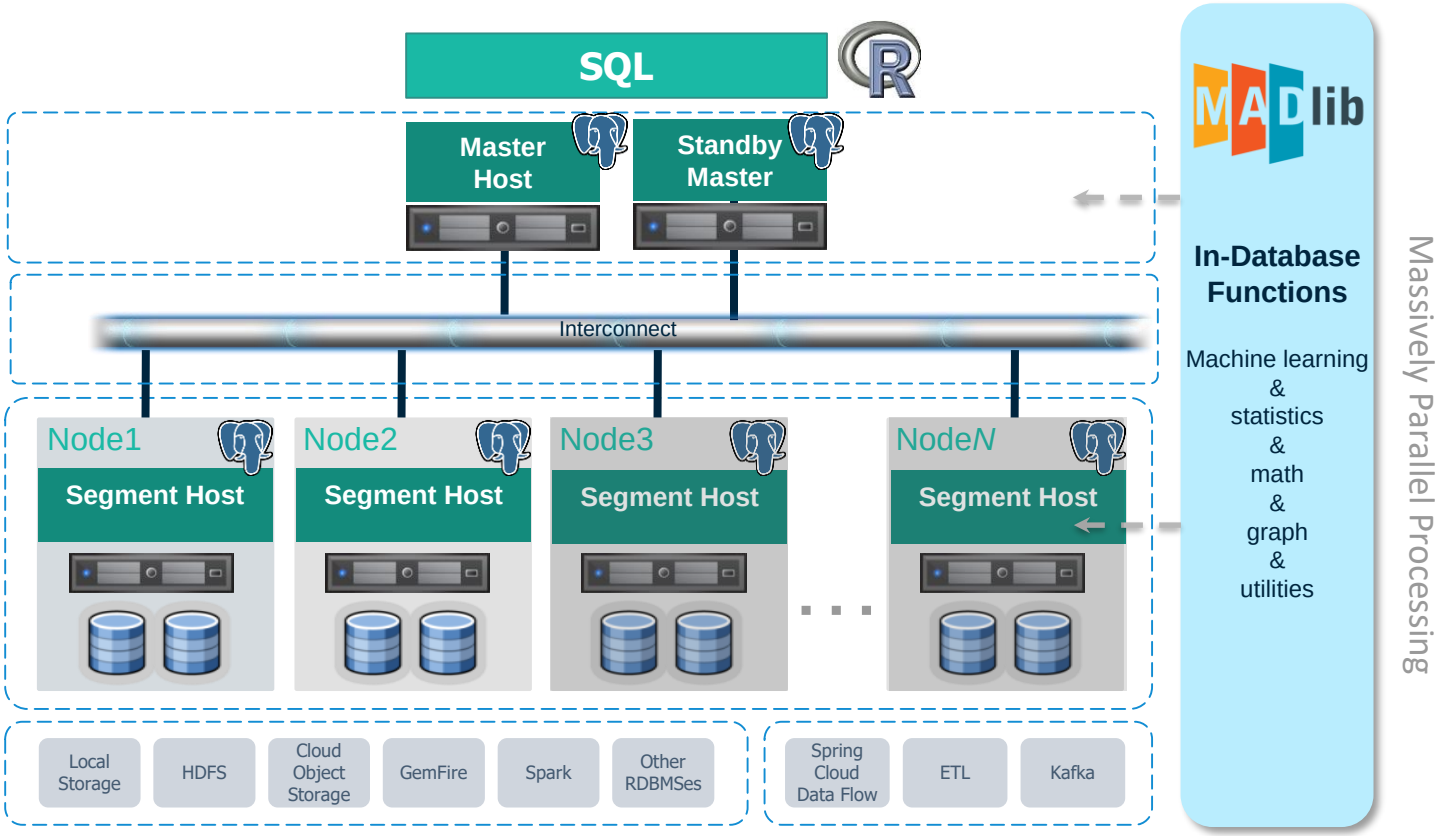
每天 5亿 数据，单表最大1000亿，200+ 用户访问，1分钟间隔



Pivotal

基于 SQL 的数据库内嵌机器学习

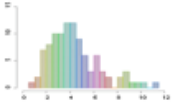
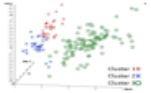
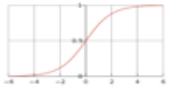
MADLib 架构


SQL
Master Host
Standby Master
Interconnect
Node1
Segment Host
Node2
Segment Host
Node3
Segment Host
NodeN
Segment Host
Local Storage
HDFS
Cloud Object Storage
GemFire
Spark
Other RDBMSes
Spring Cloud Data Flow
ETL
Kafka
MADlib
In-Database Functions
Machine learning & statistics & math & graph & utilities
Massively Parallel Processing

基于 SQL 的机器学习

GREENPLUM DATABASE

MADlib Functions


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

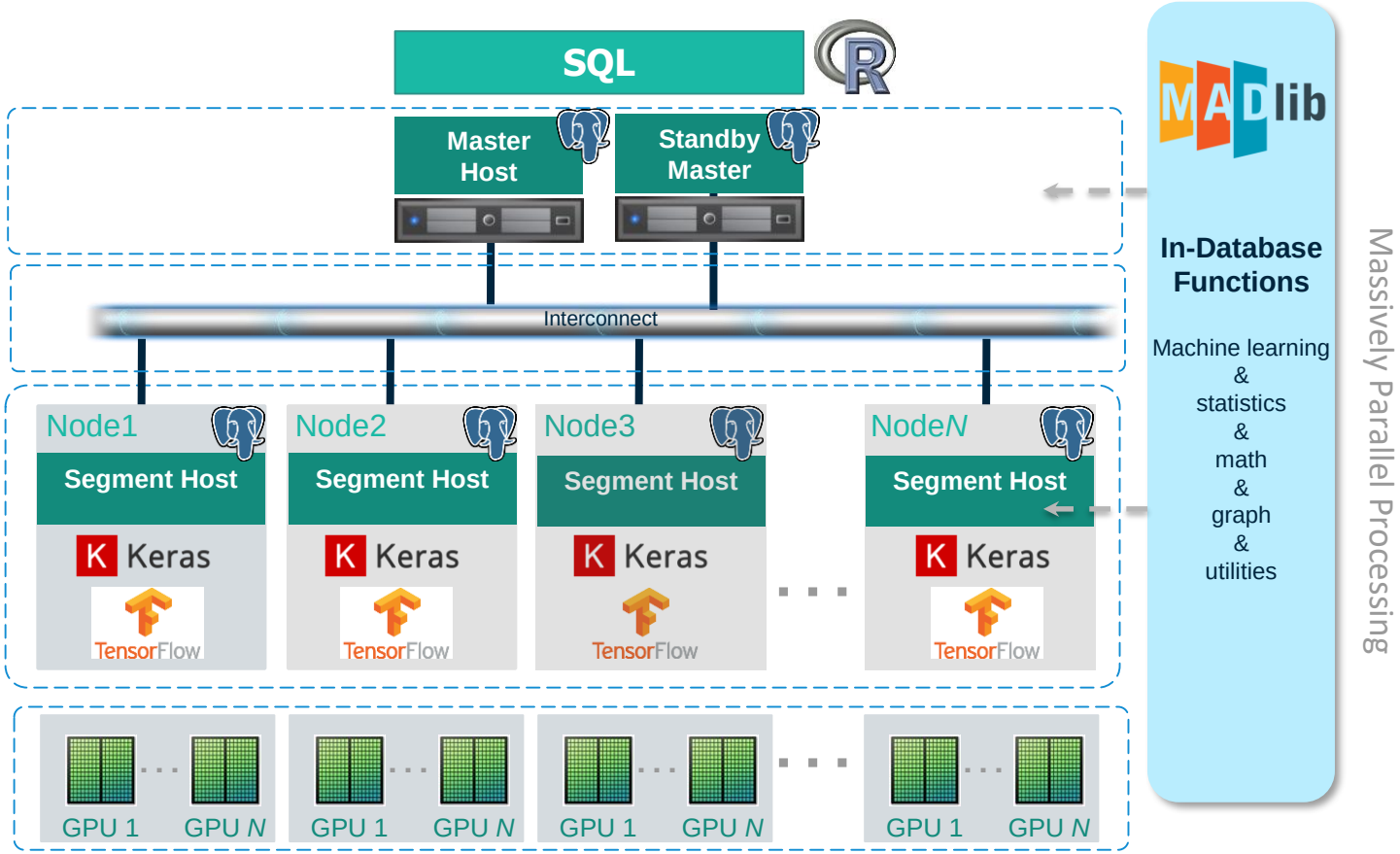
Statistics
Descriptive Statistics
- Cardinality Estimators
- Correlation and Covariance
- Summary

Inferential Statistics - Hypothesis Tests
Probability Functions

Model Selection
Cross Validation
Prediction Metrics
Train-Test Split

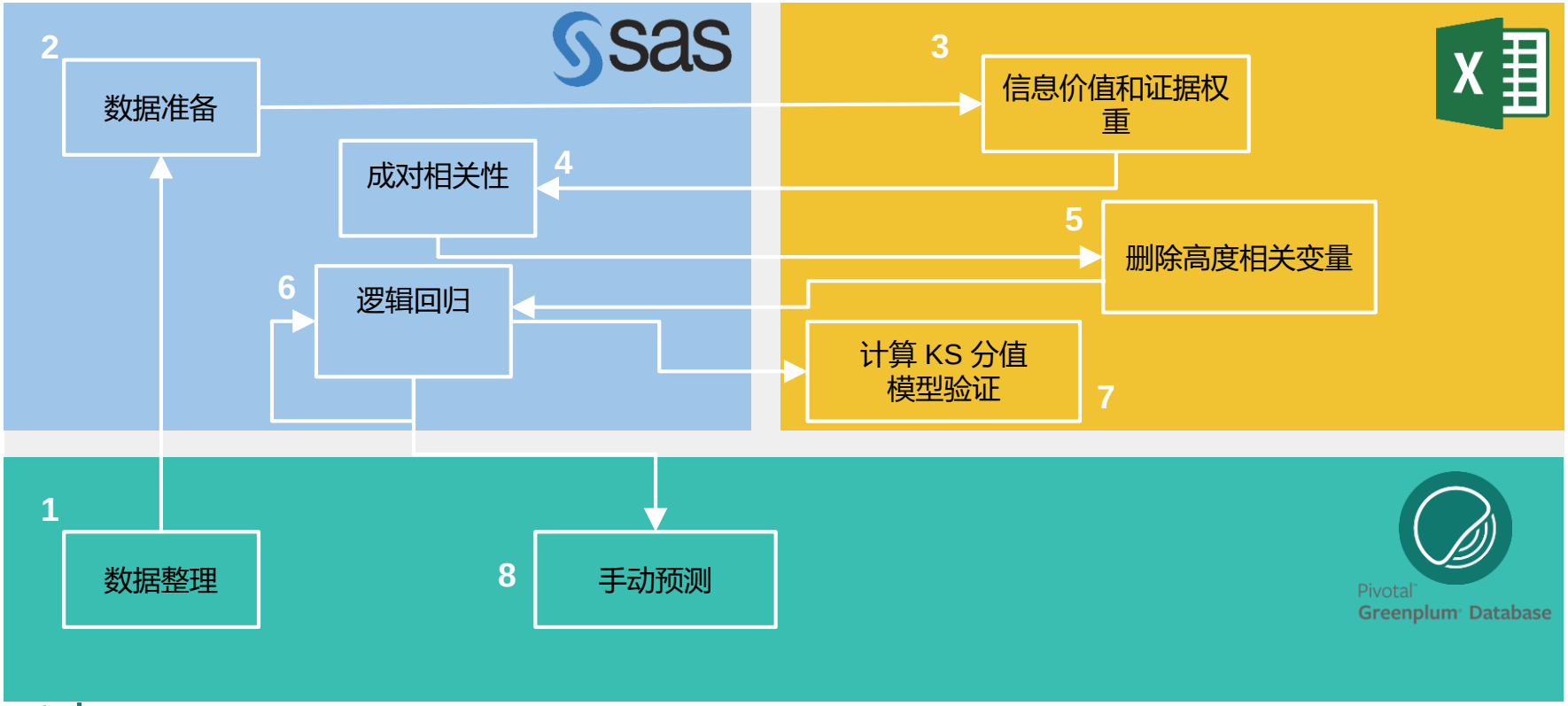
Sampling
Balanced
Random
Stratified

基于SQL的深度学习


SQL
Master Host
Standby Master
Interconnect
Node1
Node2
Node3
NodeN
Segment Host
Segment Host
Segment Host
Segment Host
Keras
TensorFlow
GPU 1
GPU N
MADlib
In-Database Functions
Machine learning & statistics & math & graph & utilities
Massively Parallel Processing

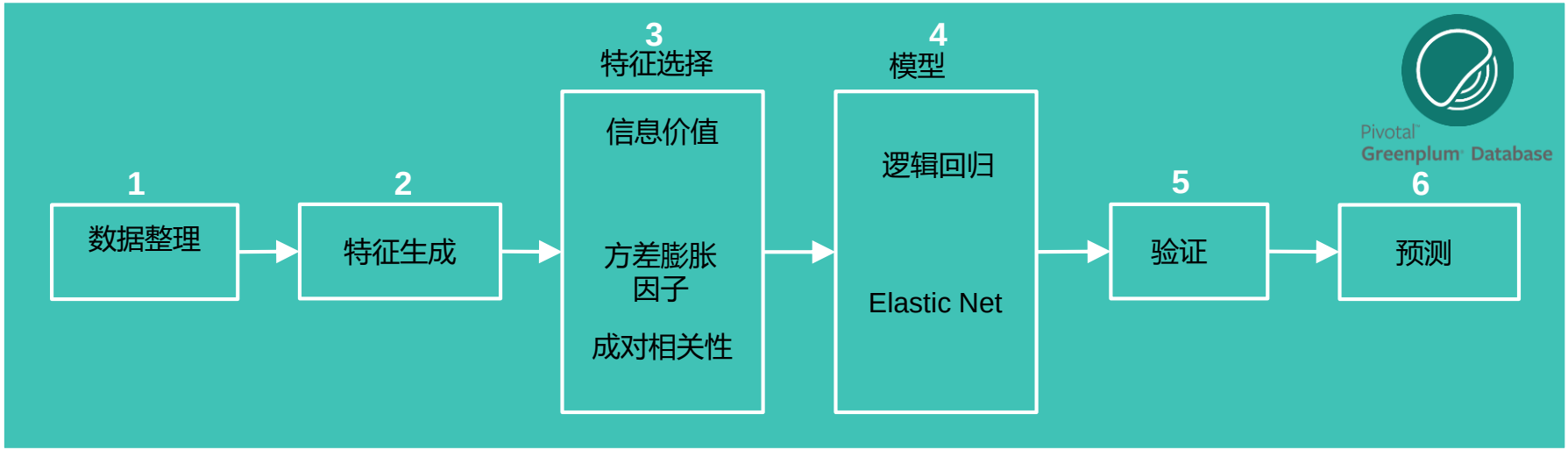
某跨国传媒和娱乐公司案例

TB级数据，400个特征



采用 Greenplum 数据库内分析方案后

TB级数据，~1000个特征



前后对比

	之前	之后	性能提升
数据编辑/整理	• 181 行代码 • 75 分钟	• 116 行代码 • 8 分钟	9.35x
特征编辑	• 439 特征 • 4,517 行代码 • 100 分钟	• 934 特征 • 1,438 行代码 • 30 分钟	多 495 个特征，快 3.33x
信息价值	• ~450 个变量，~30分钟计算结果并写入 excel	• 在 GPDB 中花 58 秒计算 ~200 个变量的IV	13.7x/变量
建模	• < 50 个变量，运行一次逻辑回归迭代需要 ~30 分钟	• 376 个变量，运行一次逻辑回归迭代需要 ~1.86 分钟	~16x/迭代

现代 SQL vs. SQL-1992

现代 SQL: SQL 演变为 transformation 语言

特性	SQL 标准	Greenplum 版本
关系模型、关系操作	SQL-1992	Greenplum 3
复杂数据类型	SQL-1999	Greenplum 4
LATERAL	SQL-1999	Greenplum 6
XML	SQL-2003	Greenplum 4
Window 函数	SQL-2003	Greenplum 4
OVER frames	SQL-2003	Greenplum 4
OVER GROUPS	SQL-2011	Greenplum 4
Percentile	SQL-2003	Greenplum 4
JSON	SQL-2016	Greenplum 5

看一个问题：求每科前三名

Student	Class	Score
A	MATH	98
B	MATH	60
C	MATH	100
D	MATH	95
E	MATH	70
A	ENGLISH	90
B	ENGLISH	99
X	ENGLISH	80
Y	ENGLISH	59

用你最熟悉的方式

- 手工计算
- Excel
- Python
- Java
- golang
- C
- Hadoop
- Spark
- ……

考虑下：

- **多久出解决方案**
- **该解决方案需要多久获得结果**

然后考虑下：

- 1000亿条记录而不是几条记录
- 存在大量分组（内存容纳不下）
- 其他人同时在修改数据时，如何避免数据不一致
- 对数据进行加密，防止数据泄露
- 数据量过大，如何使用压缩技术节省空间
- 避免不该看到数据的人看到数据，适当的认证和访问控制
- 如何实现服务高可用，避免一台机器崩溃，整个业务不可用
- 数据存储在不同的结构或者体系中（SQL 术语：JOIN）
- 利用CPU多核性能提高性能（多线程、多进程）
- 利用CPU SIMD指令集或者L3 Cache提高性能
- 需要访问的数据存储在其他系统中，譬如 HDFS、S3 等
- ……

Window 函数实例

创建数据库表

```
CREATE TABLE student_score (
    student    TEXT
  , class      TEXT
  , score      INT
);
```

插入数据

```
INSERT INTO student_score VALUES
('A', 'MATH', 98),
('B', 'MATH', 60),
('C', 'MATH', 100),
('D', 'MATH', 95),
('E', 'MATH', 70),
('A', 'ENGLISH', 90),
('B', 'ENGLISH', 99),
('X', 'ENGLISH', 80),
('Y', 'ENGLISH', 59);
```

获得每科前三名的 SQL

```
SELECT * FROM (
  SELECT
    *,
    rank() OVER (
      PARTITION BY class
      ORDER BY score DESC) rn
  FROM student_score
) dt
WHERE rn <= 3;
```

结果

```
 student |  class   | score | rn
---------+----------+-------+----
 B       | ENGLISH  |    99 |  1
 A       | ENGLISH  |    90 |  2
 X       | ENGLISH  |    80 |  3
 C       | MATH     |   100 |  1
 A       | MATH     |    98 |  2
 D       | MATH     |    95 |  3
```

JSON 数据类型实例 (提供了十几个操作符，二十多个函数)

GREENPLUM DATABASE

```
CREATE TABLE books (
  book_id serial NOT NULL,
  data jsonb
);
```

```
CREATE INDEX idx_published ON books (data->'published');
```

```
INSERT INTO books VALUES (1, '{"title": "Sleeping Beauties", "genres": ["Fiction", "T
INSERT INTO books VALUES (2, '{"title": "Influence", "genres": ["Marketing & Sales",
INSERT INTO books VALUES (3, '{"title": "The Dictator''s Handbook", "genres": ["Law",
INSERT INTO books VALUES (4, '{"title": "Deep Work", "genres": ["Productivity", "Refe
INSERT INTO books VALUES (5, '{"title": "Siddhartha", "genres": ["Fiction", "Spiritua
```

```
SELECT data->'title' AS title FROM books;
```

```
SELECT * FROM books WHERE data->'published' = 'false';
```

```
SELECT jsonb_array_elements_text(data->'genres') AS genre
FROM books
WHERE book_id = 1;
```

```
SELECT COUNT(*) FROM books WHERE data ? 'authors';
```

Pivot

综述

- Greenplum 是成熟的企业级HTAP数据库
  - 为全球
  - 来自各行各业
  - 的大量大型客户（世界500强）
  - 之生产系统
  - 支撑关键数据分析业务 （大数据）
  - 还开放源代码
  - 且协议友好（Apache 协议）

- 数仓/OLAP
- 混合负载/HTAP
- 流数据
- 数据库内嵌机器学习
- 集成数据分析
- 现代 SQL

Volume

Velocity

Variety

数据仓库


Greenplum

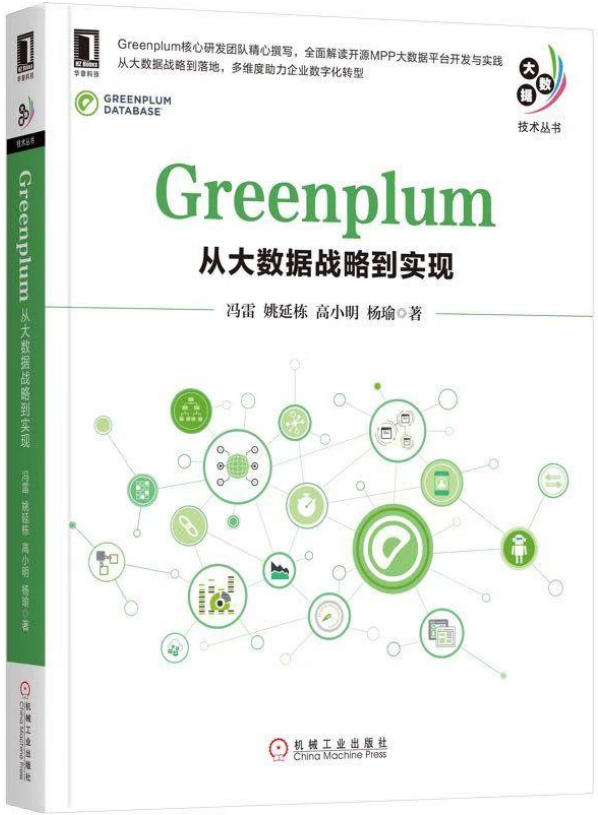
大数据


hadoop
HIVE
cloudera
IMPALA
APACHE
HBASE
APACHE
Spark


Greenplum

《Greenplum：从大数据战略到实现》

Greenplum 中国研发中心出品



长按识别二维码购买

敬请关注 Greenplum

GREENPLUM DATABASE®

扫码加入Greenplum技术讨论群

微信群：gp_assistant

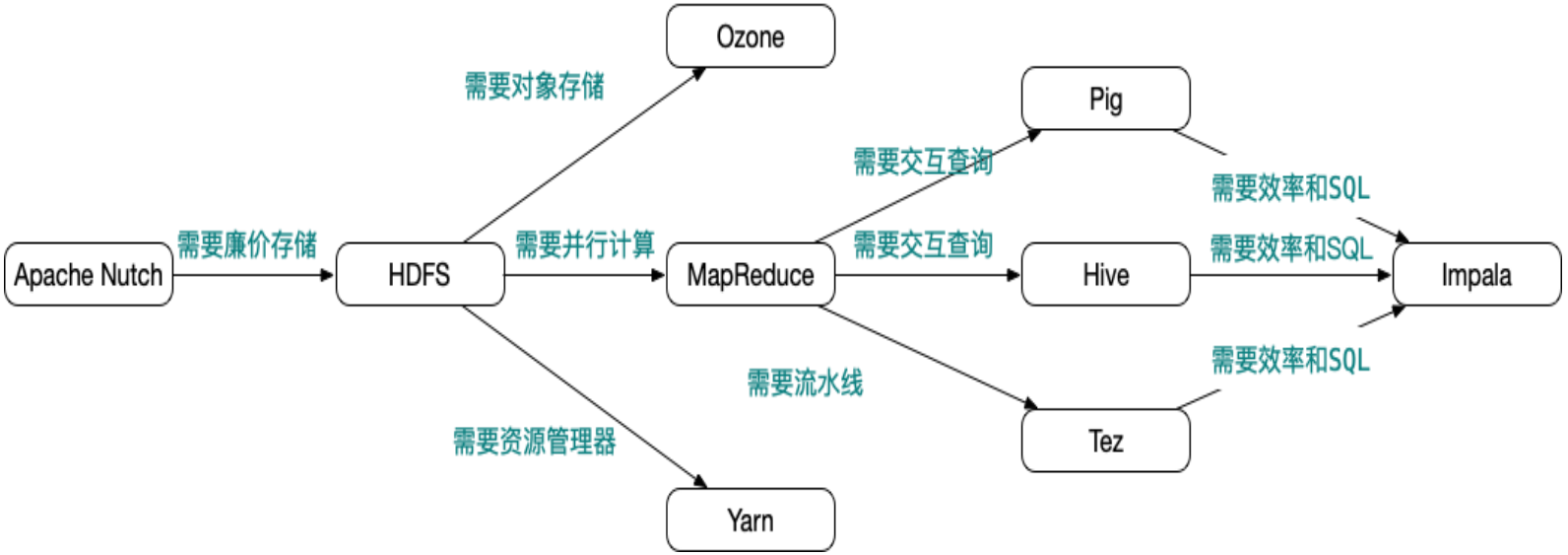
QQ群：99194625

钉钉群：https://dwz.cn/23XPHVOD

获取实时资讯，欢迎订阅中文社区网站 greenplum.cn

Pivotal

Q&A


Ozone
需要对象存储
Pig
需要交互查询
需要效率和SQL
需要廉价存储
需要并行计算
需要交互查询
需要效率和SQL
Apache Nutch
HDFS
MapReduce
Hive
Impala
需要效率和SQL
需要流水线
需要资源管理器
Tez
Yarn

NoSQL 局限性

- 不支持SQL，开发人员自己实现复杂的代码，进行聚集分析等。
- 不支持ACID和事务，实现大量代码处理数据不一致
- 不支持关联，只得使用宽表，引起数据冗余，维护代价高
- 使用低级查询语言，数据独立性差，灵活性差，维护代价高。
- 缺少标准接口，学习代价高，应用使用代价高，需要大量胶水代码
- 缺少生态，从数据迁移、ETL、报表、BI、可视化都要从头开发
- 多种 NoSQL 产品的引入，数据整合代价高
- 人才缺乏，企业积累的大量SQL人才和资产浪费