Greenplum 6:
混合负载的理想数据平台

高小明

Pivotal Greenplum®

全球领先的开源MPP大数据平台

mongoing 中文社区

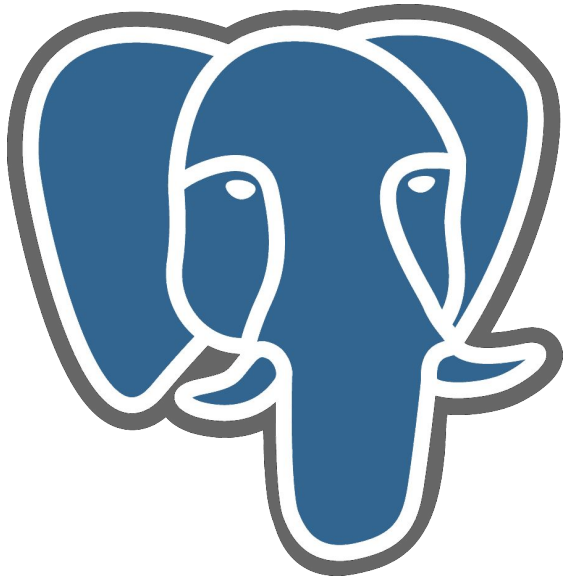
PostgreSQL 中文社区

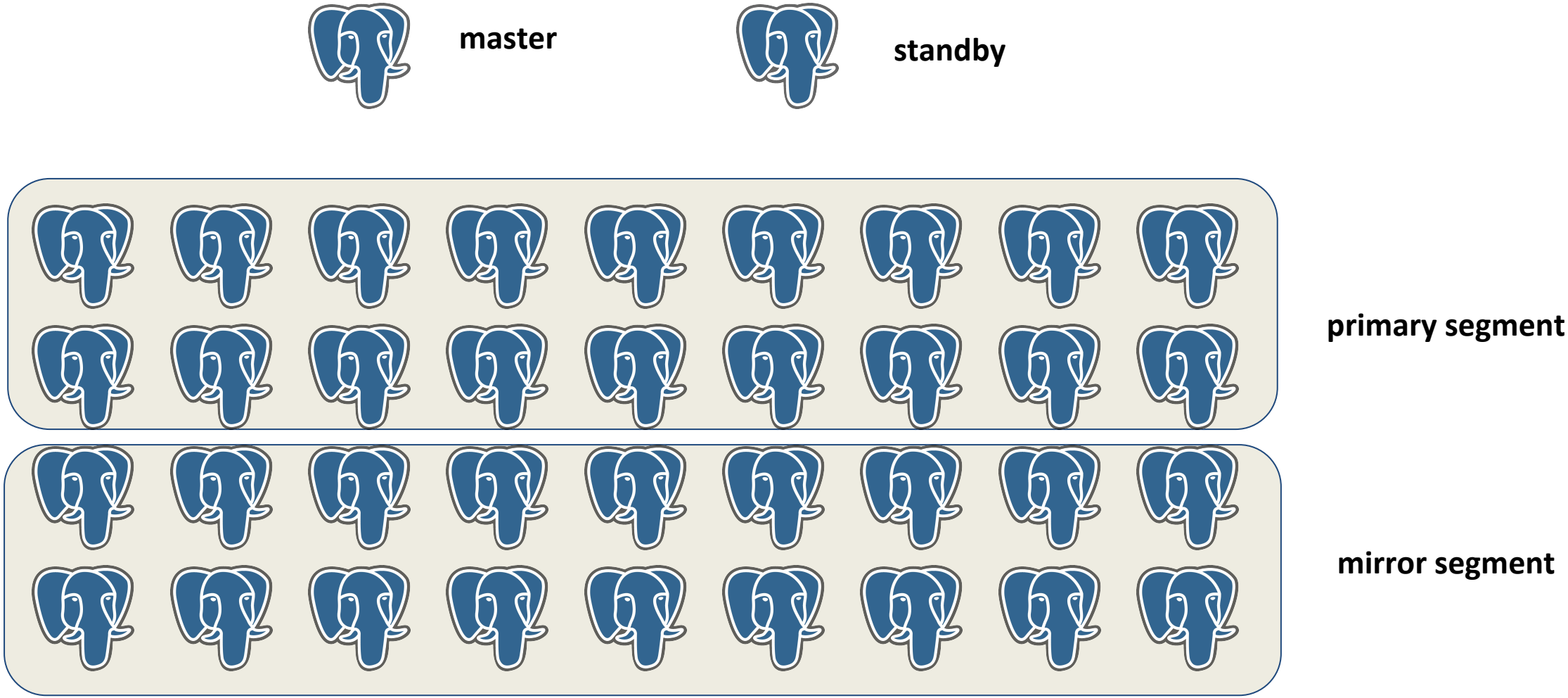
结构化	vs	半结构非结构化
可扩展性	vs	ACID事务
分布式	vs	简单易用
事务型	vs	分析型

Pivotal

MPP

- massively parallel processing
- 大规模并行处理

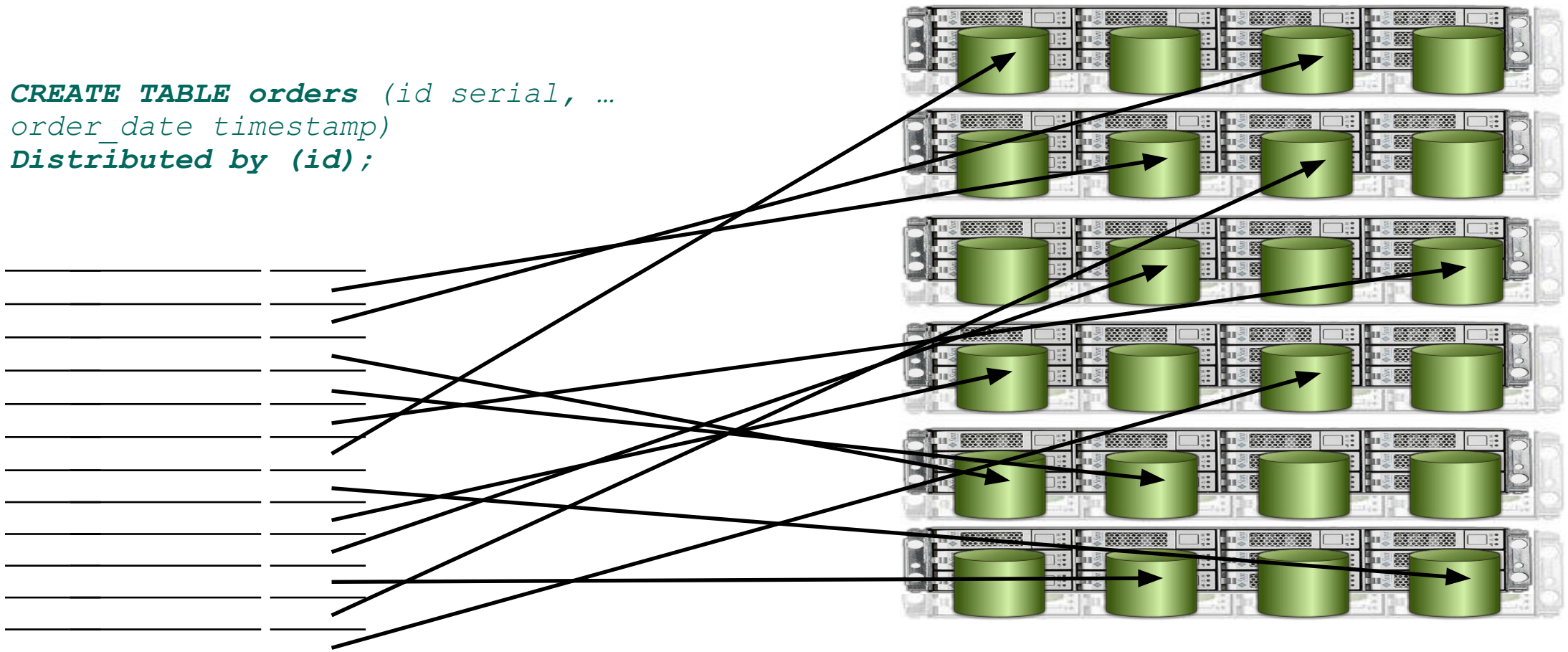



master
standby
primary segment
mirror segment

数据分布: 并行化的根基

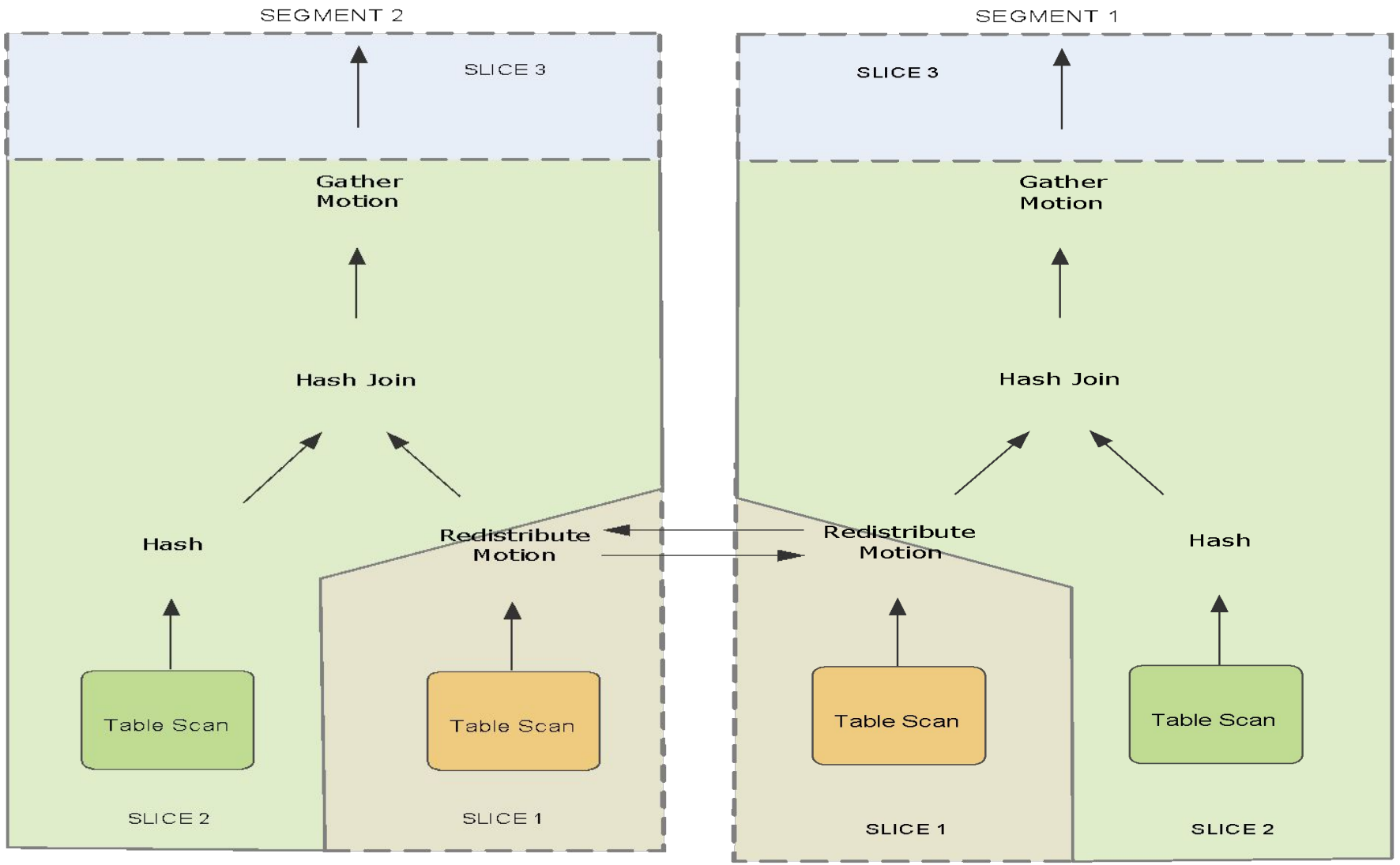
最重要的策略和目标是均匀分布数据到各个数据节点。

```
CREATE TABLE orders (id serial, …
order_date timestamp)
Distributed by (id);
```

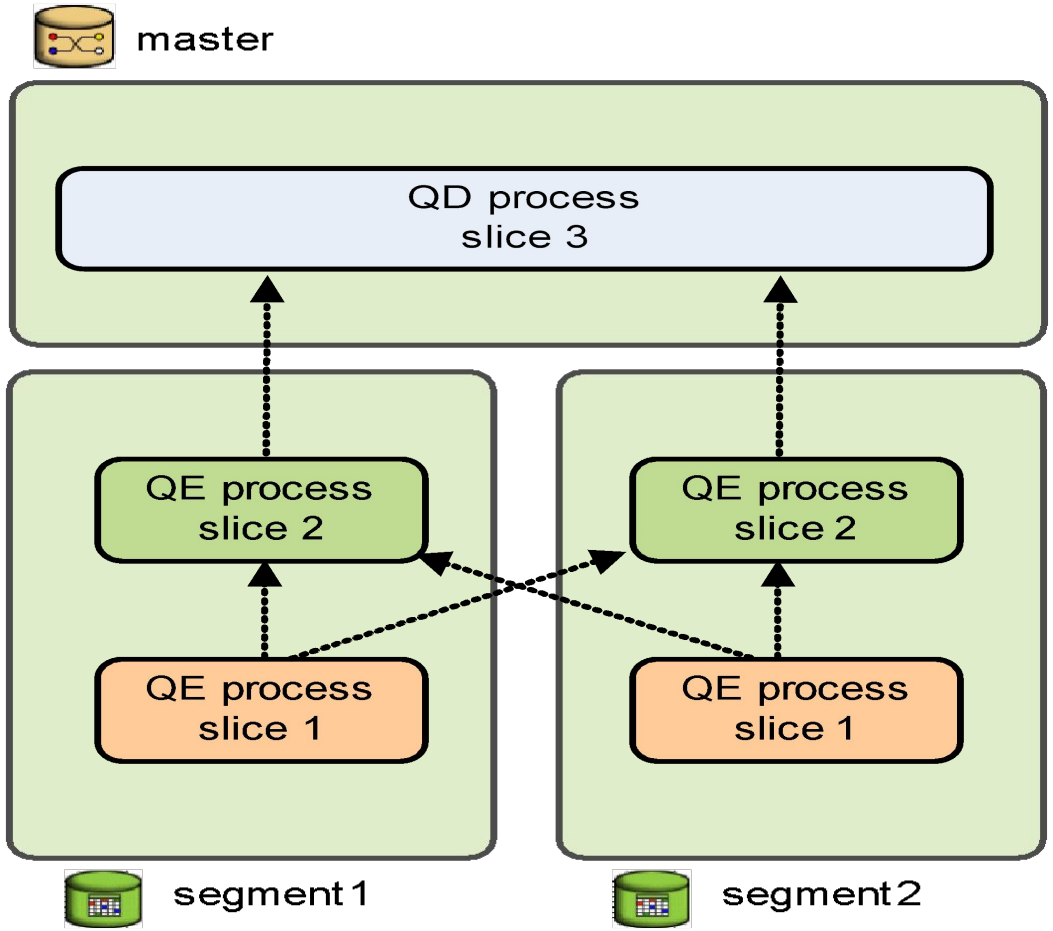


生成并行查询计划

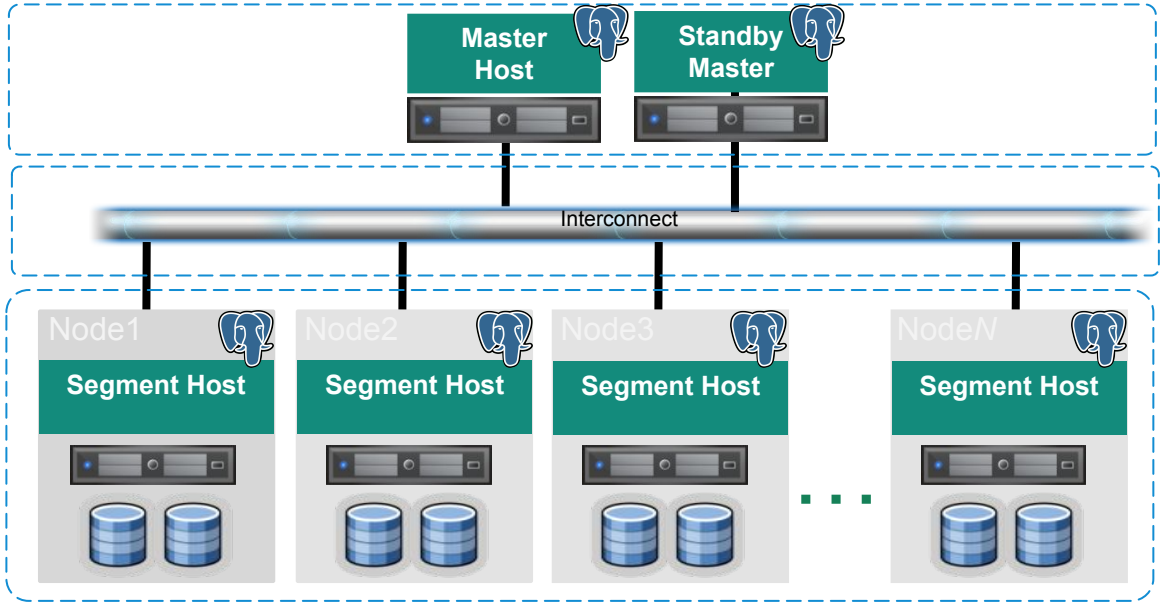
```
SELECT customer,
       amount
FROM   orders
JOIN   customer
USING (cust_id)
WHERE  date=2008;
```



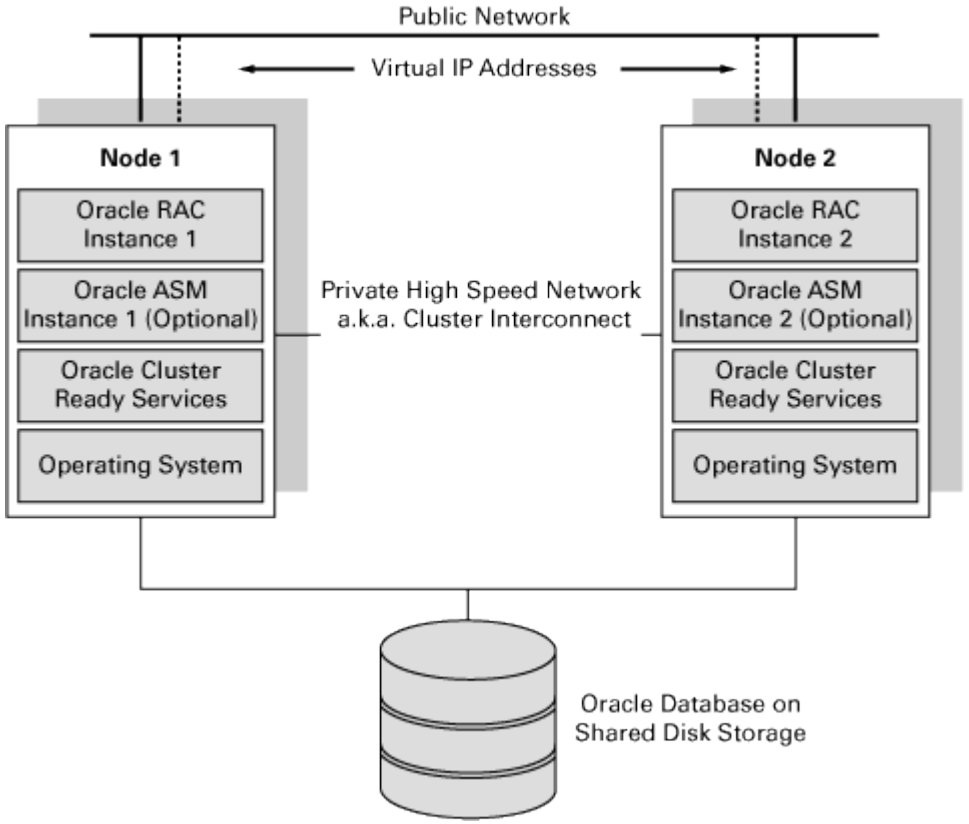
执行并行计划


master
QD process
slice 3
QE process
slice 2
QE process
slice 2
QE process
slice 1
QE process
slice 1
segment 1
segment 2

Greenplum (MPP)



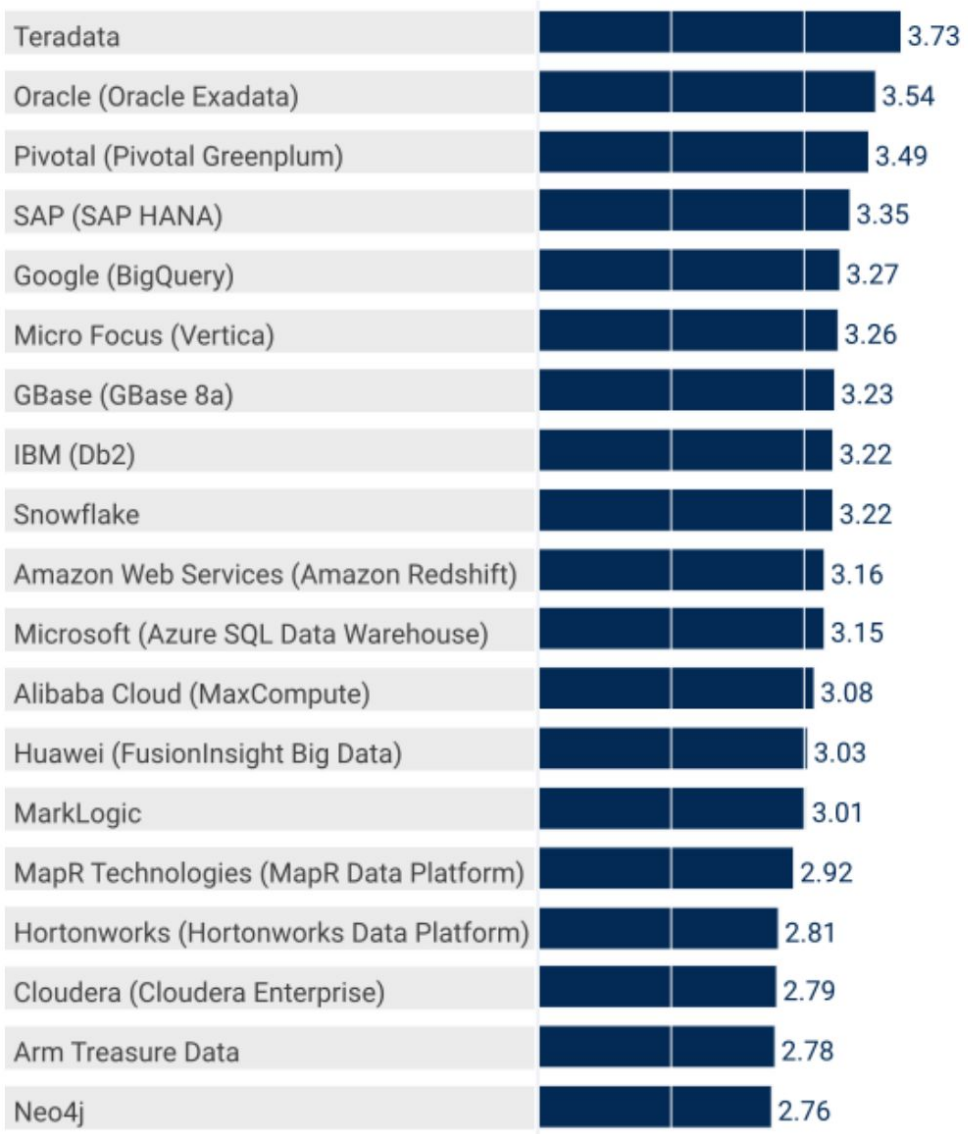
Oracle (SMP)



OLAP

- Online Analytical Processing
- 联机分析处理

Gartner 2019数据分析行业报告



Pivotal Greenplum scored highly this year in all four use cases, positioning among the top vendors in all bar the context-independent data warehouse use cases. This reflects one of the major trends in the DMSA market this year: **rediscovery. End users are turning to traditional technologies in order to meet their DMSA requirements**, and Pivotal Greenplum's strong capabilities here as an MPP relational database are well-showcased

卓越的OLAP特性

列式存储

分区、压缩

高级特性

递归查询、窗口函数

ORCA

复杂查询优化器

Madlib: 机器学习

数据库内并行模型训练和预测、分类

集成分析

多格式、多语言

成熟稳定

完备生态、支撑核心生产系统

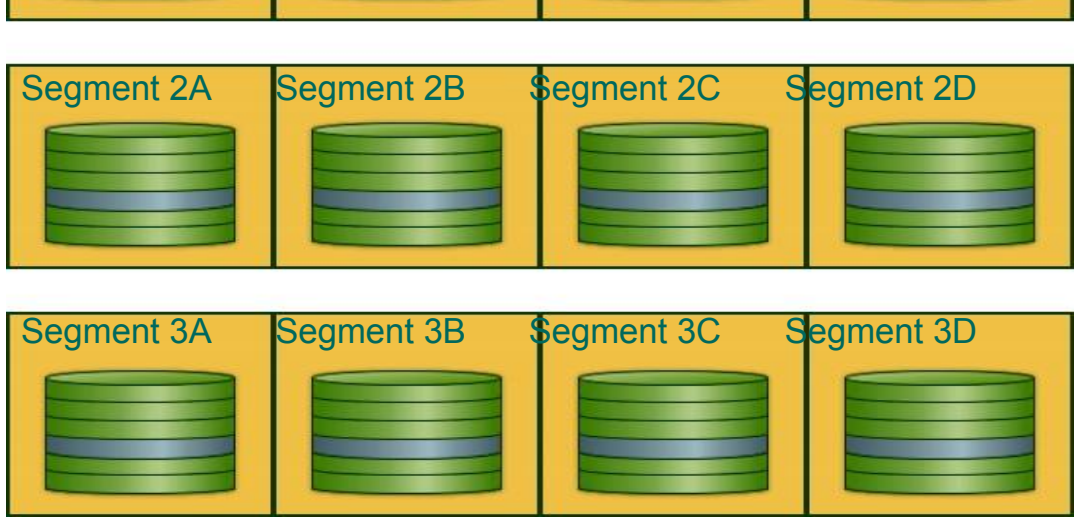
列式存储

- 更适合压缩
- 查询部分列时速度快
- 不同列可以使用不同压缩方式

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

分区

```
SELECT COUNT(*)
  FROM orders
 WHERE order_date >= 'Oct 1 2007'
   AND order_date <= 'Oct 31 2007'
```



仅仅扫描 orders 表2017年十月份数据所在的分区C

递归查询

- 层次结构
- 树状结构

```
WITH RECURSIVE included_parts(sub_part, part,
quantity) AS (
    SELECT sub_part, part, quantity FROM parts
WHERE part = 'our_product'
  UNION ALL
    SELECT p.sub_part, p.part, p.quantity
    FROM included_parts pr, parts p
    WHERE p.part = pr.sub_part
  )
SELECT sub_part, SUM(quantity) as
total_quantity
FROM included_parts
GROUP BY sub_part
```

窗口函数

- 计算移动平均值或各种时间间隔的总和
- 分组内重置聚合和排序

```
SELECT last_name,
       salary,
       department,
       rank() OVER w
FROM employees
       WINDOW w as (PARTITION BY department
ORDER BY salary DESC)
```

last_name	salary	department	rank
Jones	45000	Accounting	1
Williams	37000	Accounting	2
Smith	55000	Sales	1
Adams	50000	Sales	2
Johnson	40000	Marketing	1

Pivotal

ORCA优化器

超过8年的投资，多位博士的长期贡献

基于Cascades / Volcano框架, Goetz Graefe

优化分布式大数据系统中特别复杂的查询

01

高效处理相关子查询

02

公共表达式的下推

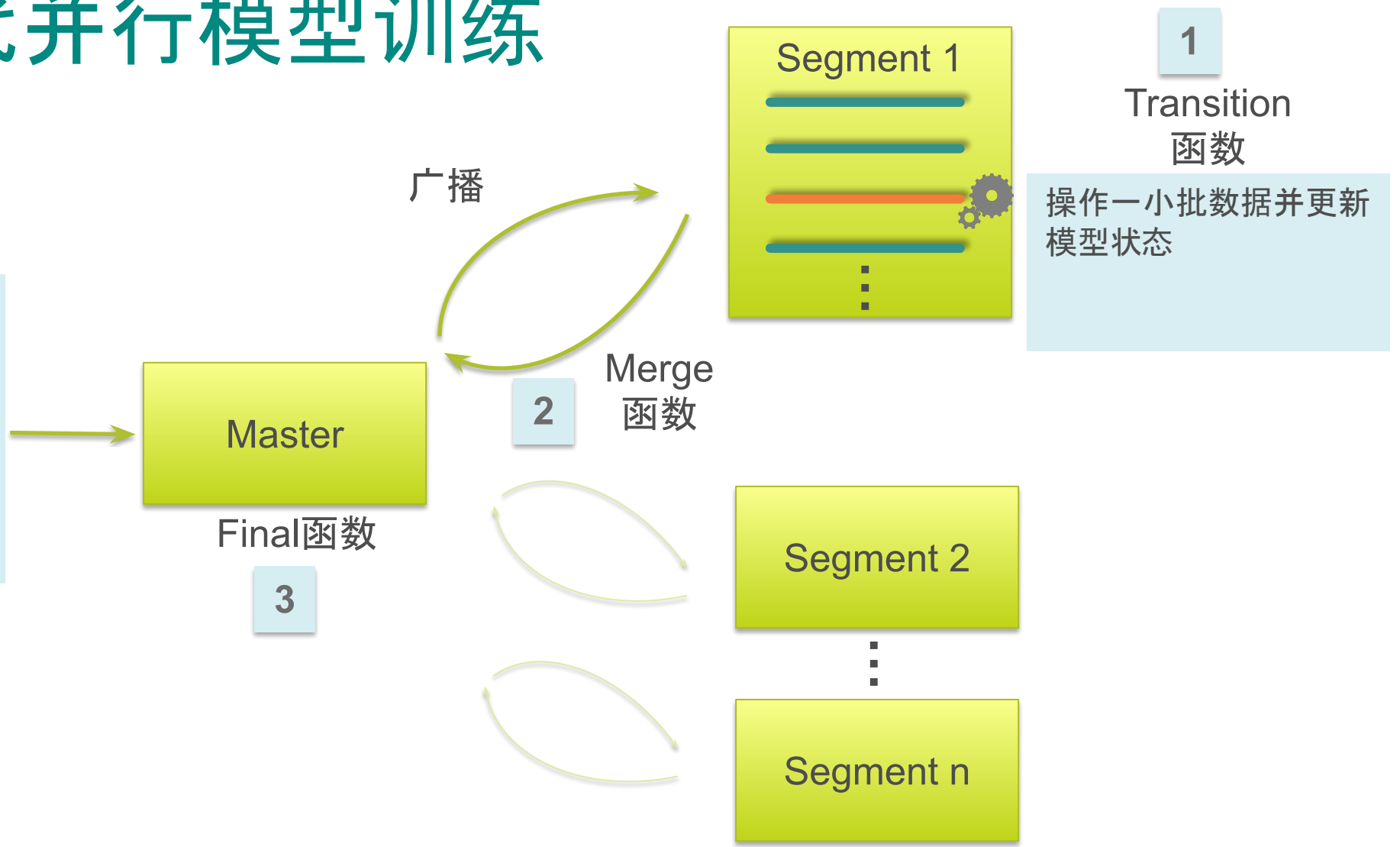
03

动态分区裁剪

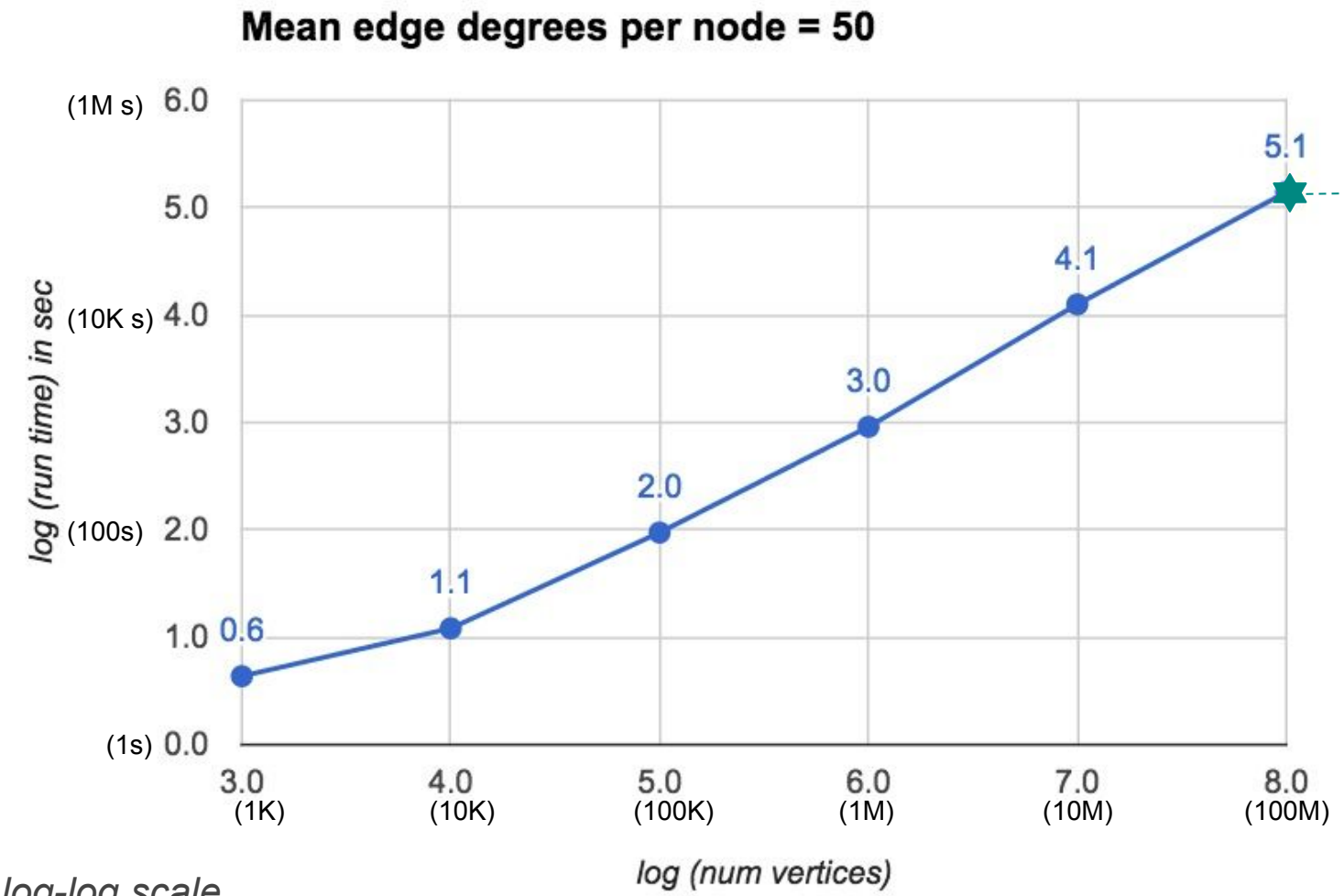
Madlib: 迭代并行模型训练

模型存储过程

```
model = init(…)
WHILE model not converged
  model =
    SELECT
      model.aggregation(…)
    FROM
      data table
ENDWHILE
```



Madlib: PageRank性能



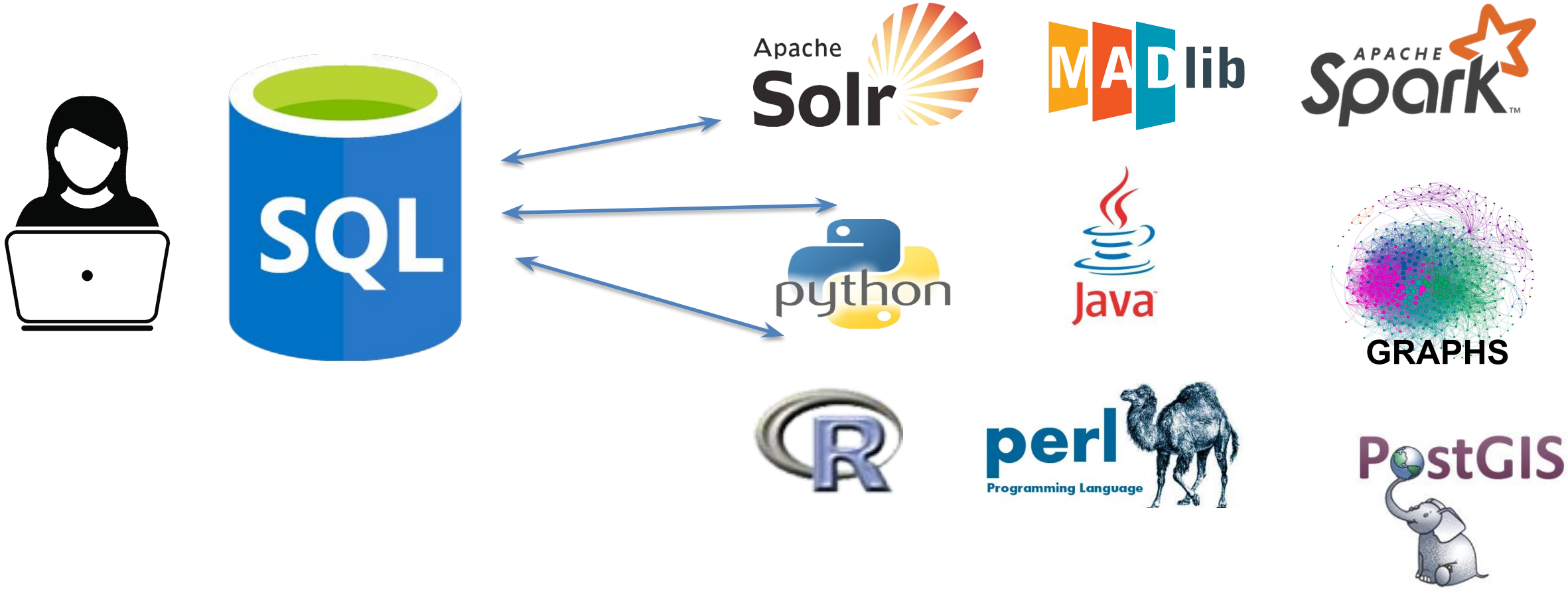
50亿条链接

Greenplum集群:

- 1 master
- 4*6 segment

Note: log-log scale

数据库内集成分析



“请找出这样的员工，在Pivotal工作，互相直接认识，有一个人名字听起来像是‘Peter’或者‘Pavan’，并且最近24小时从一个给定经纬度的参考点方圆2KM的ATM机上取出了多于$200的现金”

Pivotal

```
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模糊字符串匹配函数**Soundex()** 可以知道姓名是否发音是‘Pavan’或‘Peter’

GPText.search() 函数可以知道是否一个人在Pivotal工作

金额 > $200

Greenplum **MADlib BFS** 算法可以知道两个之间是否有直接联系

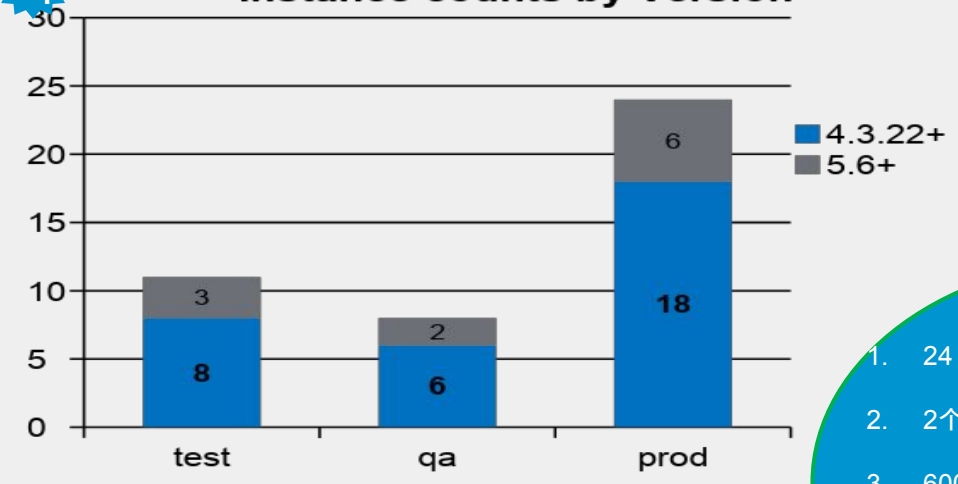
Greenplum **Time 函数**计算24小时内的取款时间

Greenplum POSTGIS 函数 **st_distance_sphere() and st_makepoint()** 计算给定经纬度方圆2KM的范围

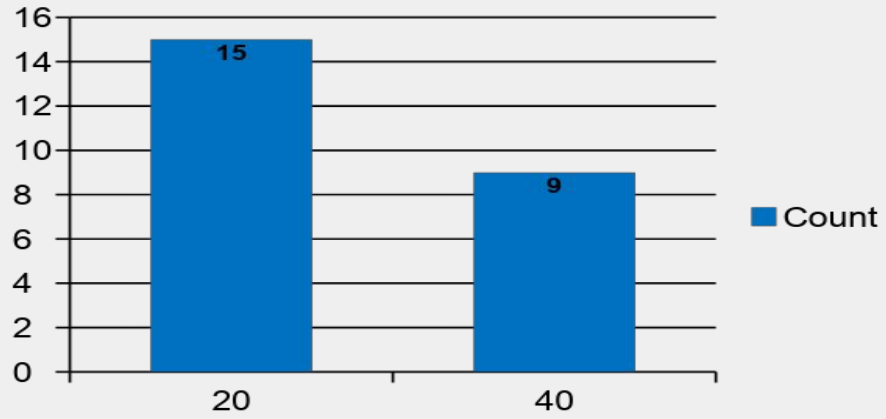
Pivotal

Greenplum在摩根士丹利

1

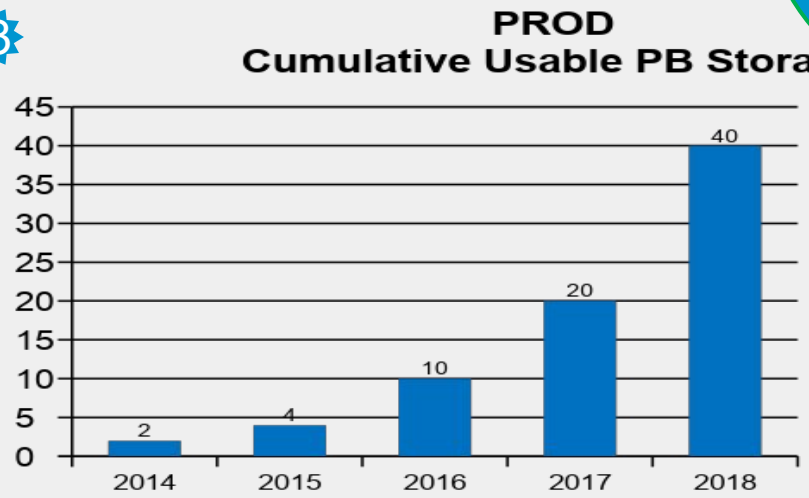


2

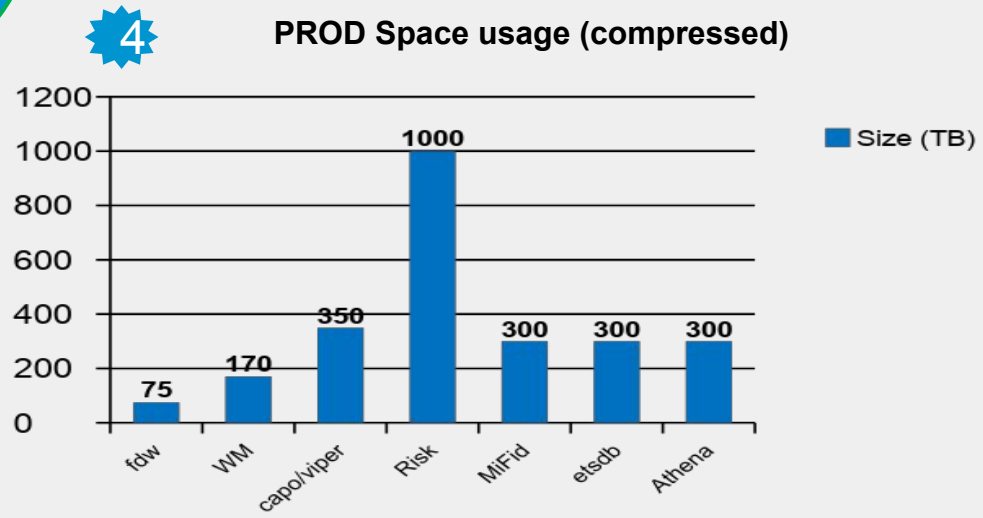


1. 24 个生产集群
2. 2个选项: 20个节点或 40个节点
3. 600+ 服务器, 13k+ 核, 81PB存储(增长中)
4. 2.5PB 或 25PB 原始数据 按10x压缩率

3



4



OLTP

- Online transaction processing
- 联机事务处理

出色的OLTP特性

天生的优势

- 行式存储
- 索引
- 直接分发
- 完整的增删改

Greenplum 6 增强

- 并发修改、删除
- 系统性的优化事务和锁

行式存储

- 更适合OLTP负载
- 高效更改和删除
- 适合需要全部或者多数列的查询

索引

Greenplum支持以下索引:

- Btree
- Bitmap
- Gist
- GIN
- BRIN（开发中）

B-Tree

Bitmap

Gist

GIN

<10ms的访问时间，即使是上亿条记录

<~ 100 不同值

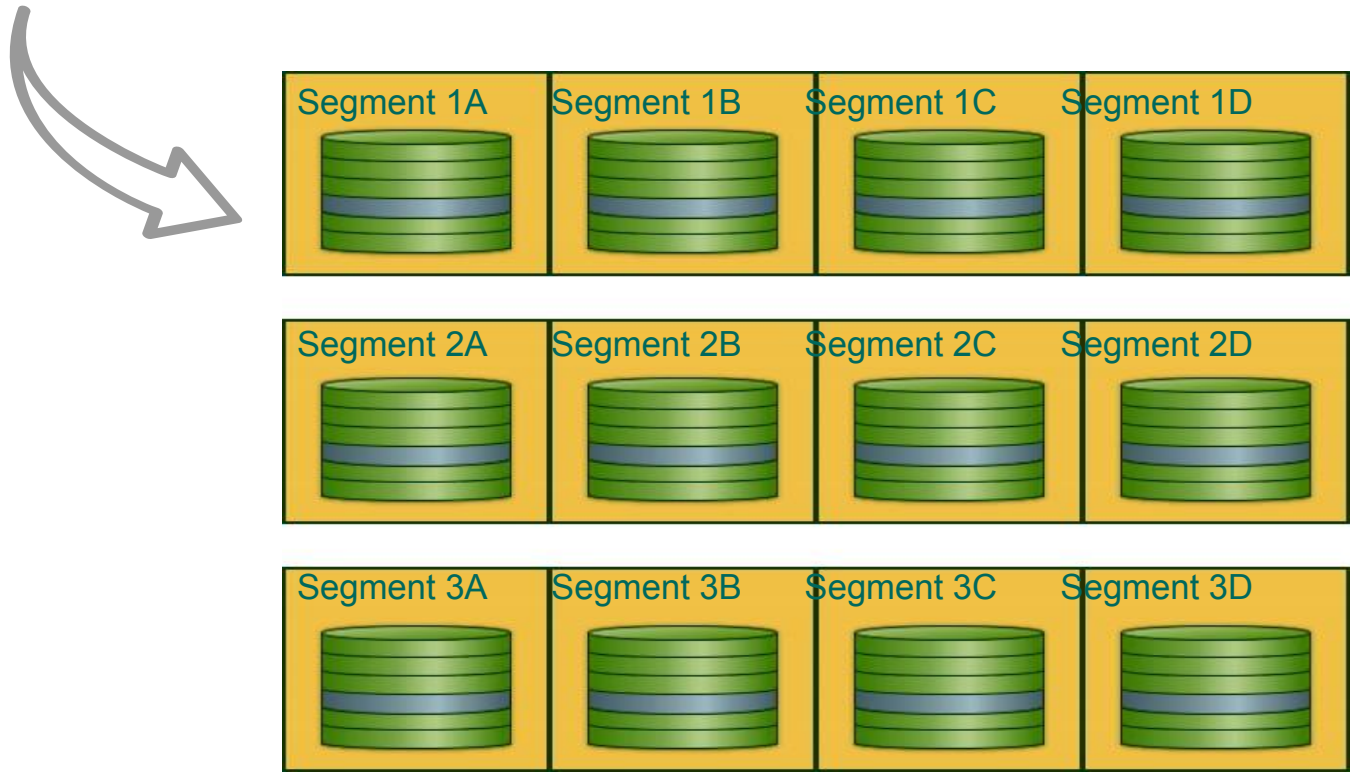
地理空间区域

文本倒排索引

直接分发

```
SELECT *
FROM orders
WHERE id = 1;
```

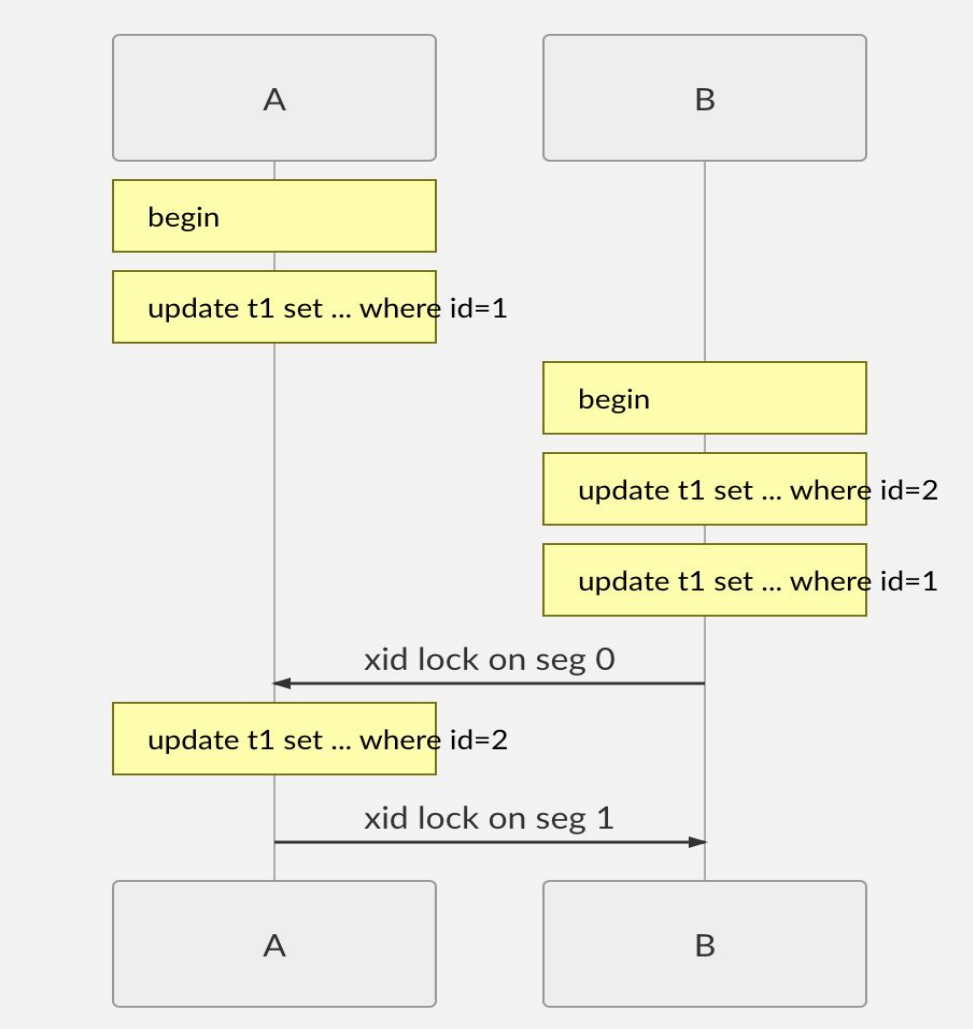
```
UPDATE orders
SET cust_id = 2
WHERE id = 2;
```

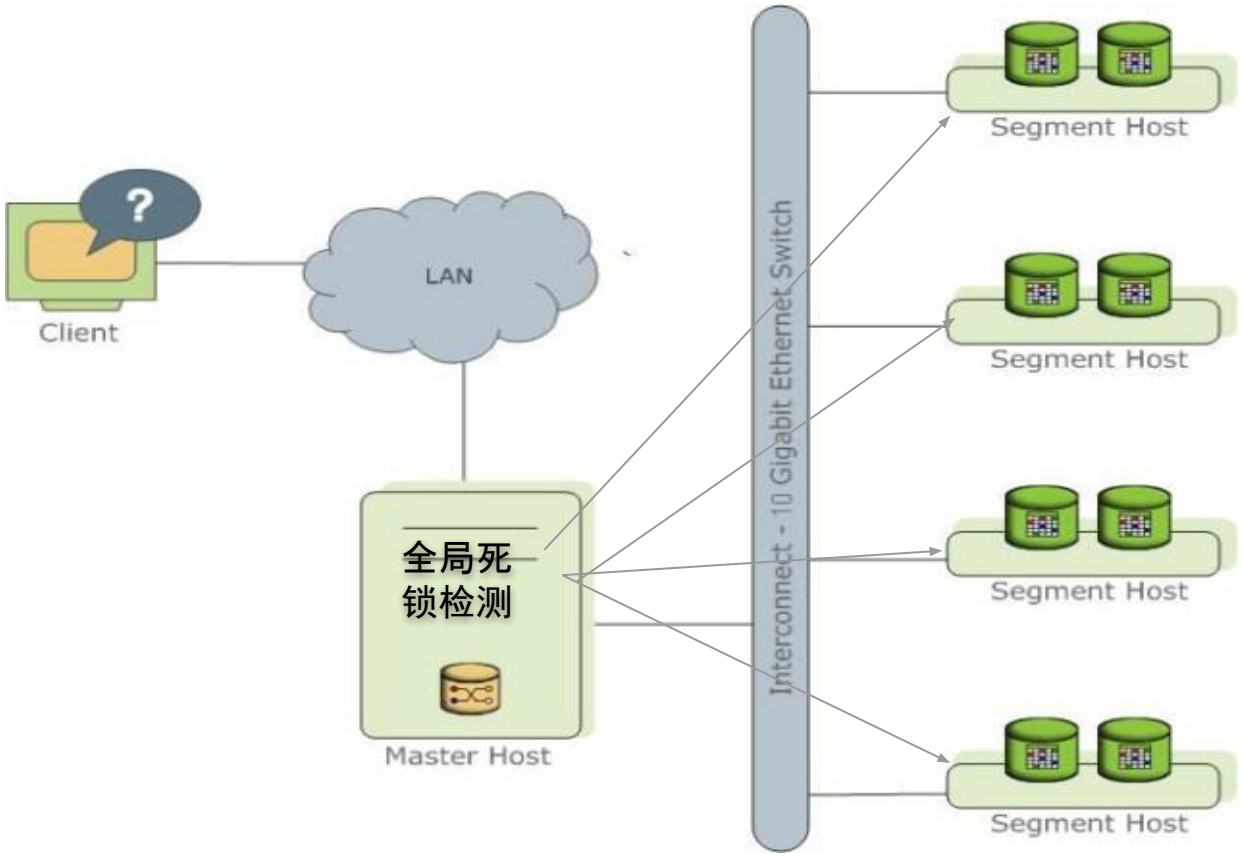


完整的增删改查

- 读和写不阻塞
- 支持更改删除、删除
- 支持更改分布键、主键（将数据从一个节点移到另一个节点）

Greenplum 6：并发改删和分布式死锁检测





gpconfig -c ‘gp_enable_global_deadlock_detector’ -v on

Greenplum 6：锁和事务的优化

- 大幅减少事务开始和结束时的锁冲突
- 消除隐式只读操作（单条SELECT）的锁冲突
- 避免显式只读事务（BEGIN-SELECT-END）的两阶段提交（开发中）
- fastpath锁（PostgreSQL合并）

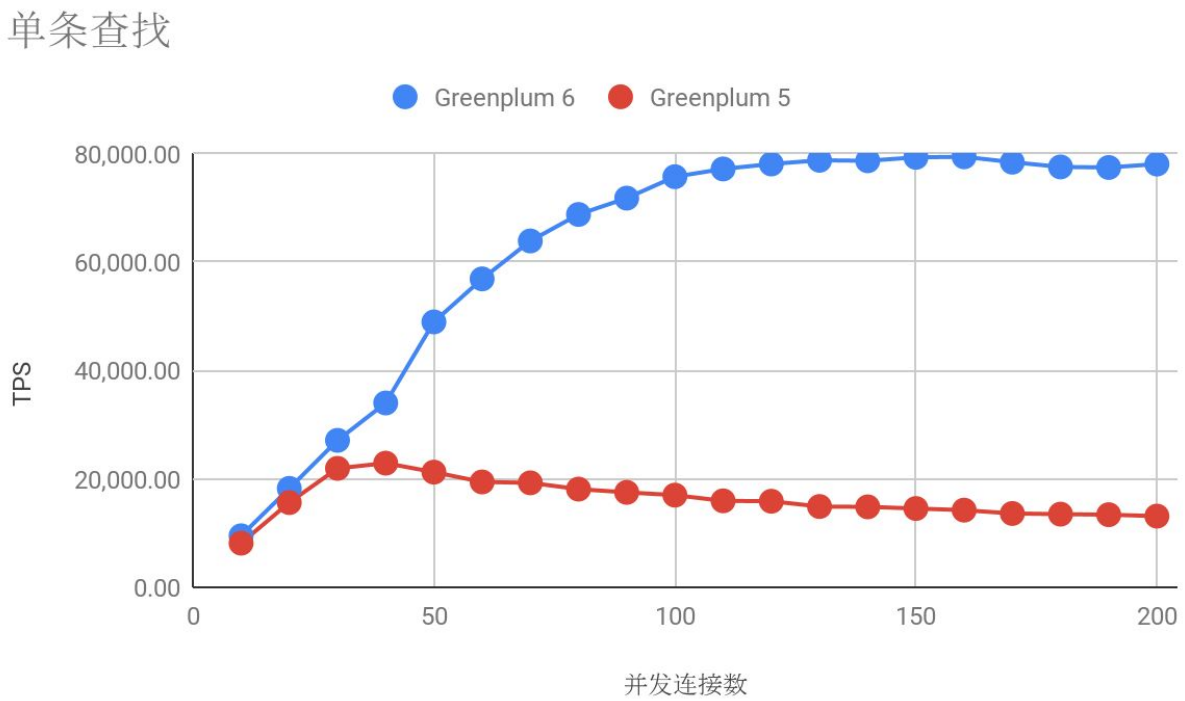
TPC-B基准测试：环境

基于谷歌云平台（Google Cloud Platform，简称GCP），为5个虚拟主机的集群，包含一个master主机和四个segment主机，master和segment虚拟主机的配置信息如下

	master	segment
虚拟机类型	n1-standard-16	n1-standard-8
CPU核数	16	8
内存大小（GB）	60	30
CPU平台	Intel Haswell	
存储类型	SSD persistent disk	
存储大小（GB）	512	
Linux发行版	Ubuntu Linux 18.04	
Linux内核版本	4.15.0	
GCC版本	7.3.0	

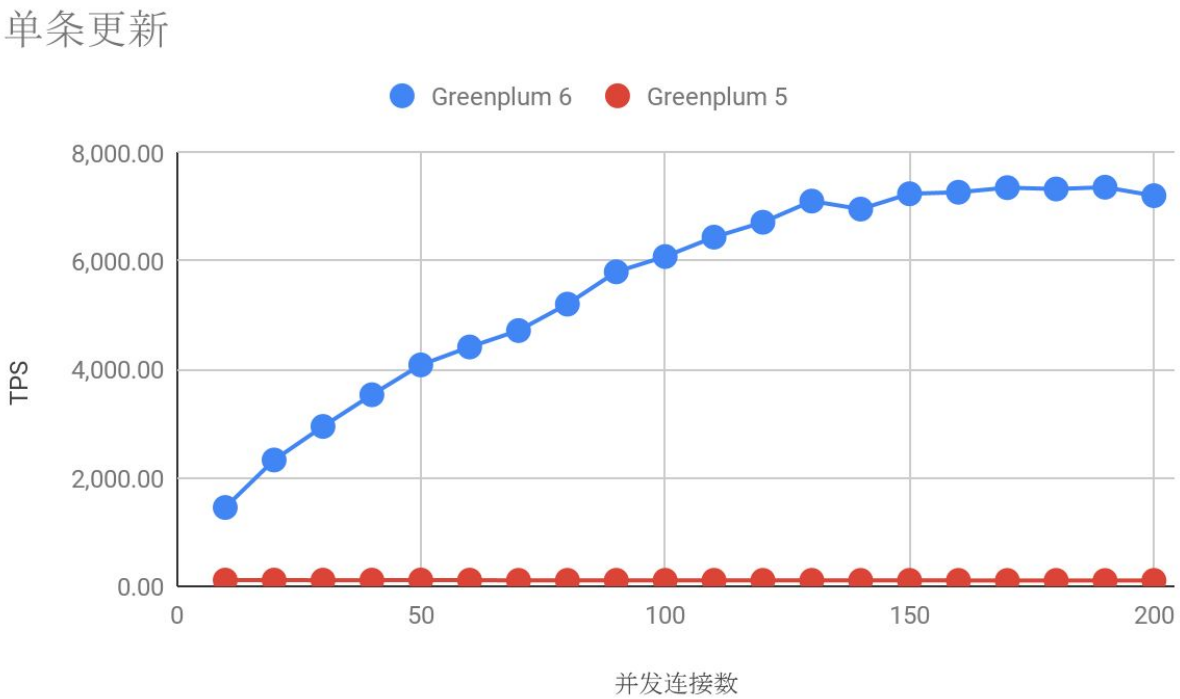
TPC-B基准测试：SELECT

- 3.5倍的TPS提升
- master CPU使用率大幅提高
- TPS随着master CPU核数增加同步提高
- 22万 TPS（192核单机部署，master+18 segments）



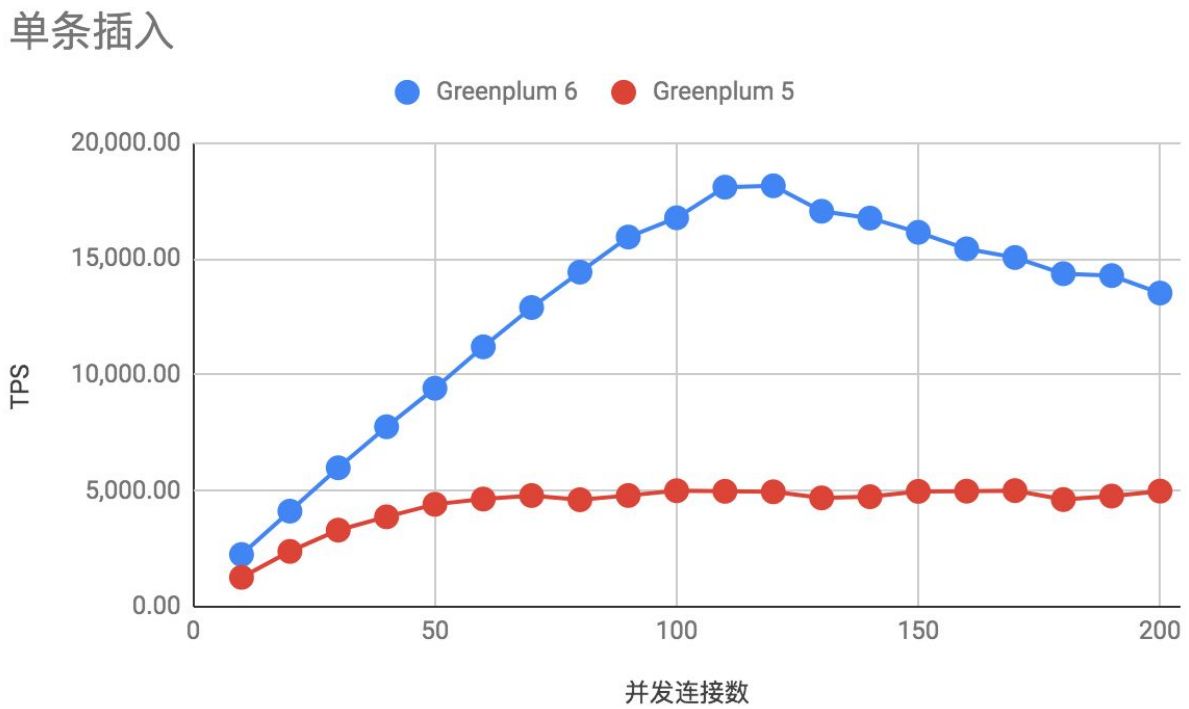
TPC-B基准测试：UPDATE

- 得益于并发更改特性
- 70倍的TPS提升



TPC-B基准测试：INSERT

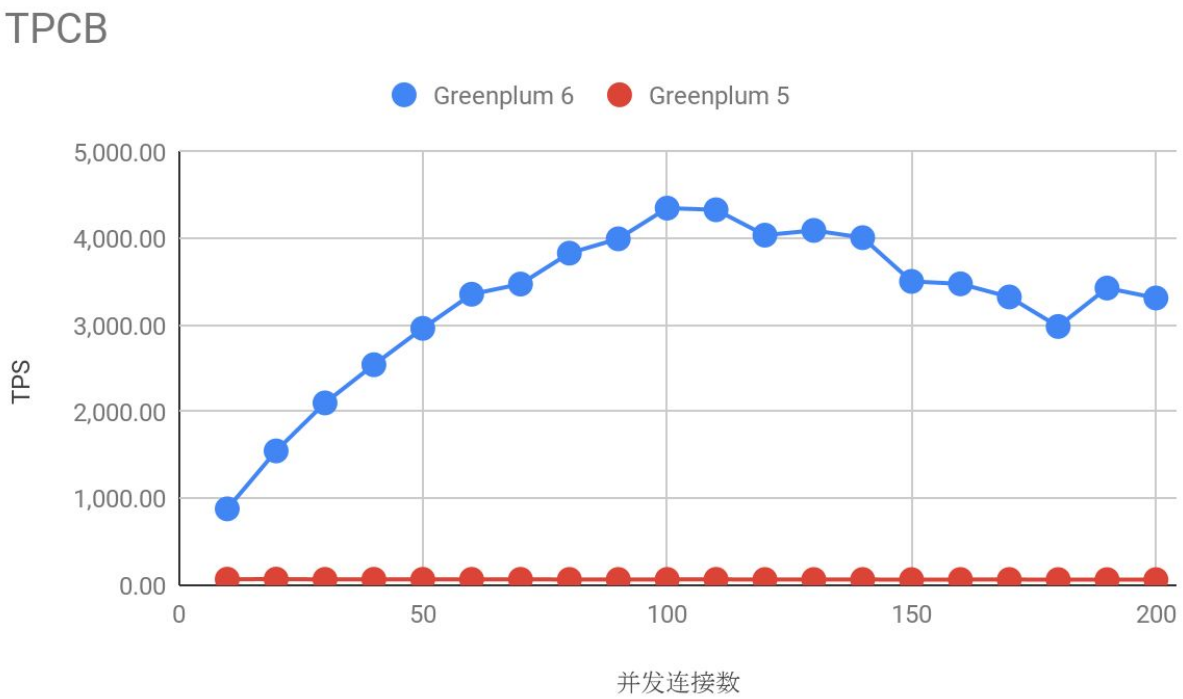
- 峰值TPS提升3.6倍



TPC-B基准测试：多语句

- 峰值TPS提升60倍

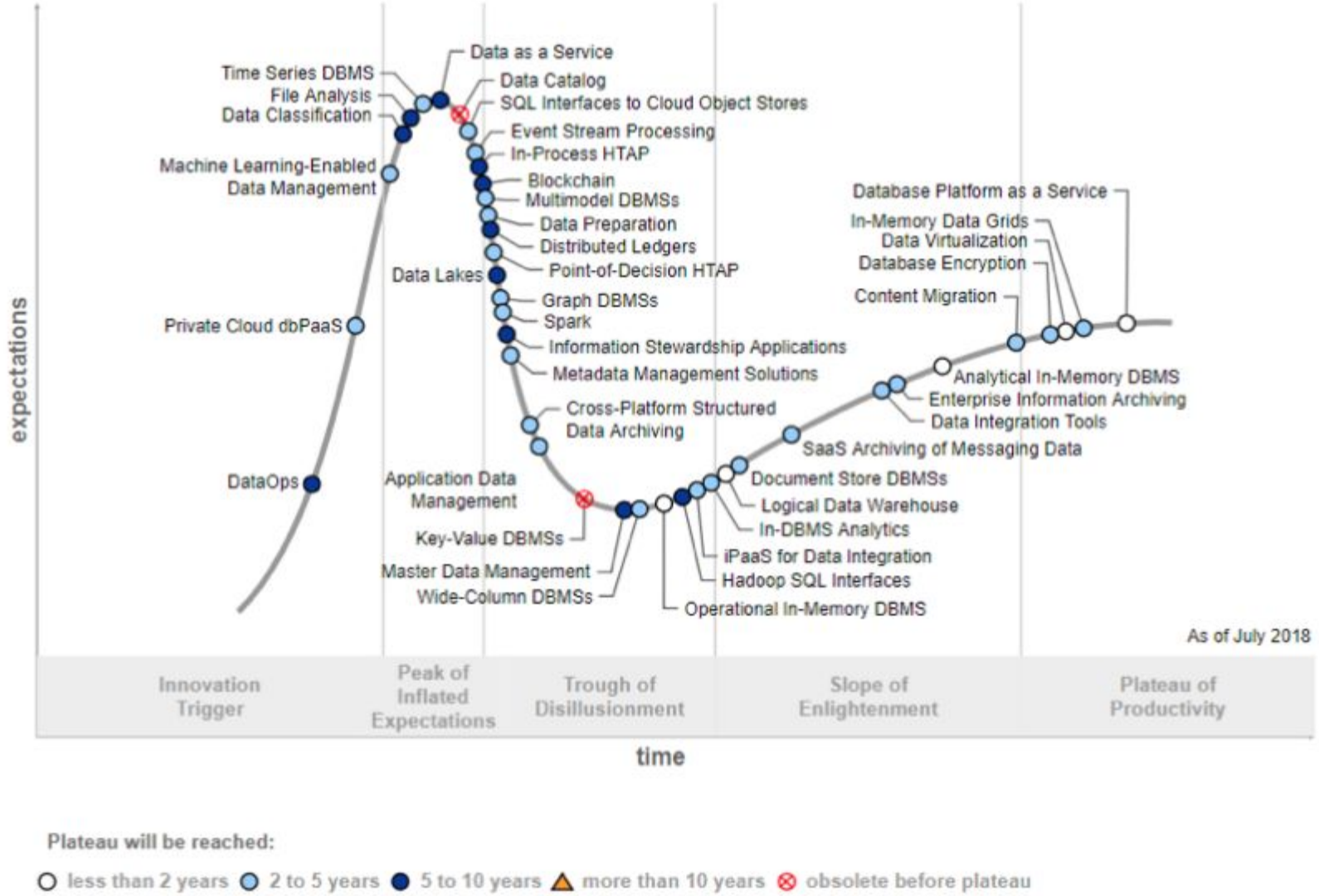
```
BEGIN;

UPDATE pgbench_accounts SET abalance =
abalance + :delta WHERE aid = :aid;

SELECT abalance FROM pgbench_accounts WHERE
aid = :aid;

UPDATE pgbench_tellers SET tbalance =
tbalance + :delta WHERE tid = :tid;

UPDATE pgbench_branches SET bbalance =
bbalance + :delta WHERE bid = :bid;

INSERT INTO pgbench_history (tid, bid, aid,
delta, mtime) VALUES (:tid, :bid, :aid,
:delta, CURRENT_TIMESTAMP);

END;
```



HTAP

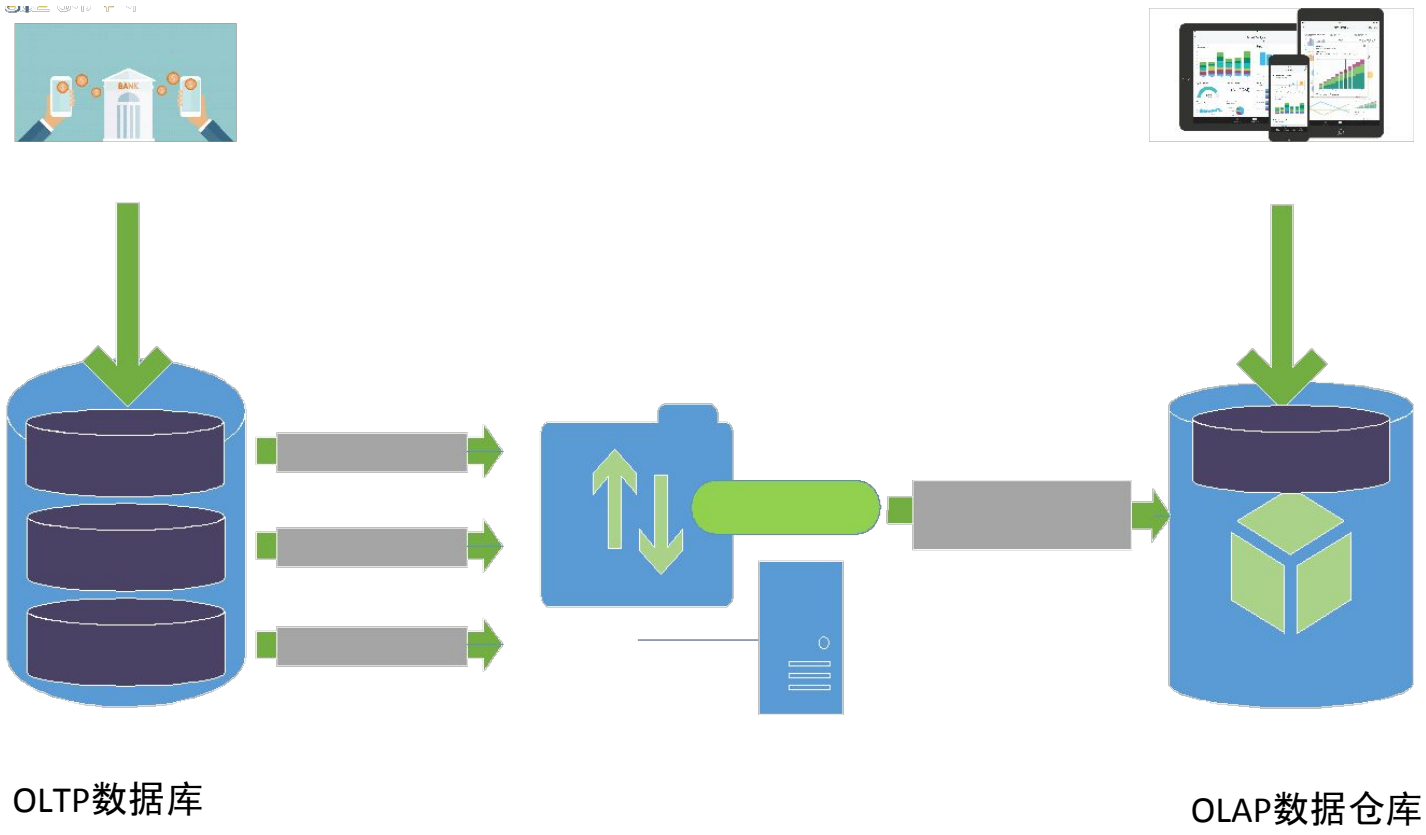
- Hybrid transactional/analytical processing
- 混合事务/分析处理

Gartner技术成熟度曲线

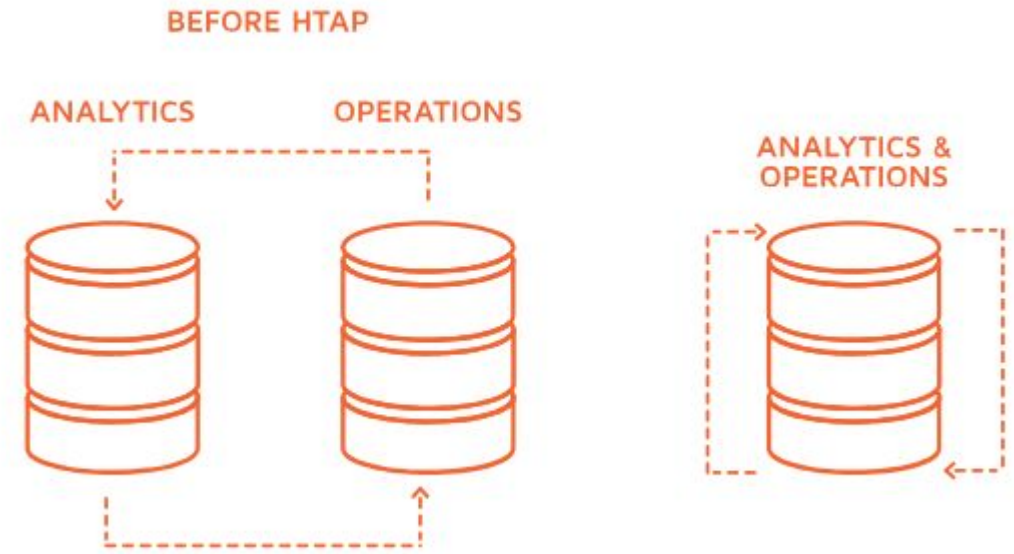


OLTP-OLAP独立部署

- 实时性
- 数据同步复杂性
- 应用复杂性



HTAP

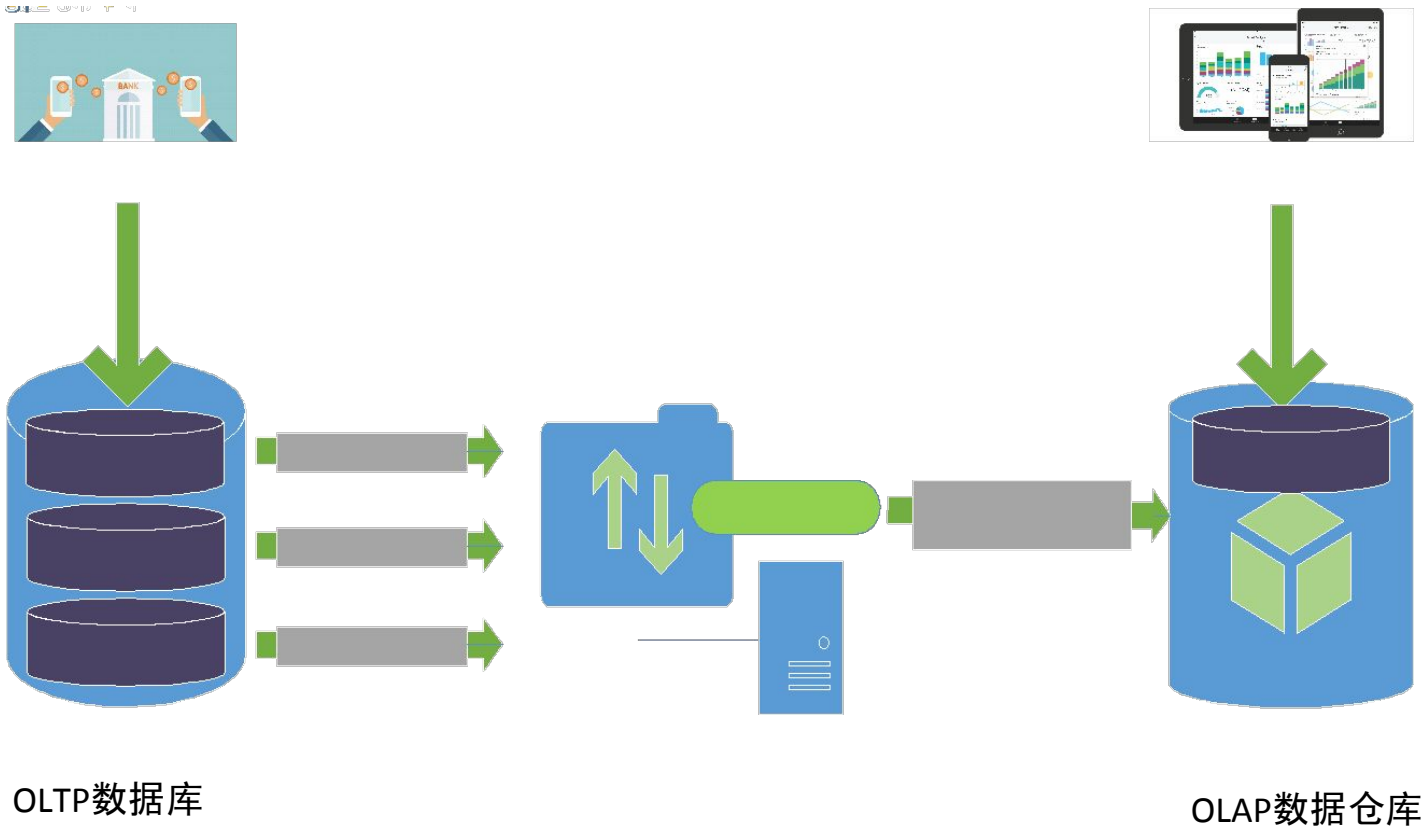


HTAP = ?

- 卓越的OLAP特性
- 出色的OLTP特性
- 多态存储
- 有效的并发和资源管理

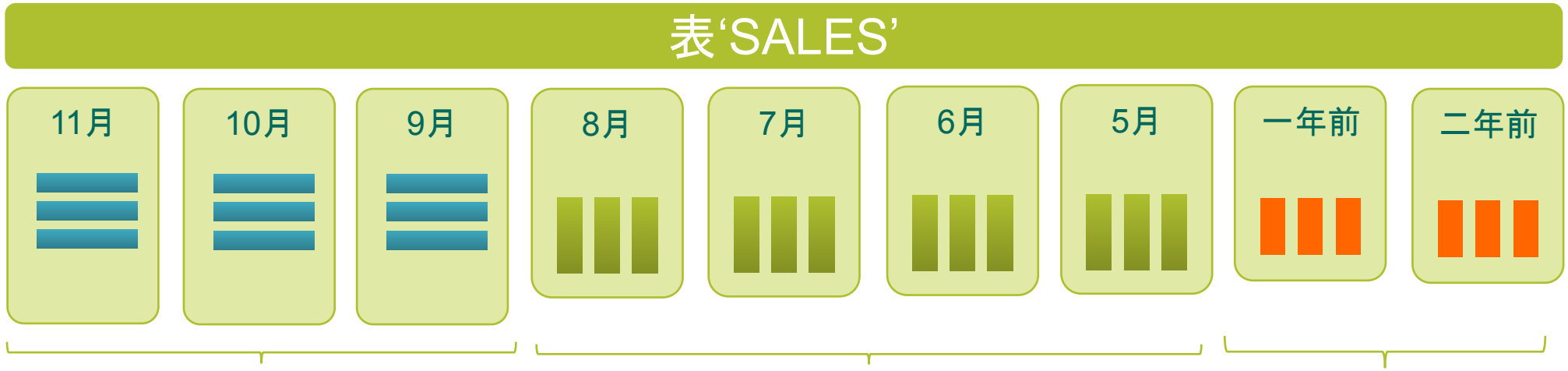
OLTP-OLAP独立部署

- 实时性
- 数据同步复杂性
- 应用复杂性



多态存储

用户自定义数据存储格式



行存储

- 访问多列时速度快
- 支持高效更新和删除
- AO 主要为插入而优化

列存储

- 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: zstd, gzip (1-9), quicklz, delta, RLE

外部表

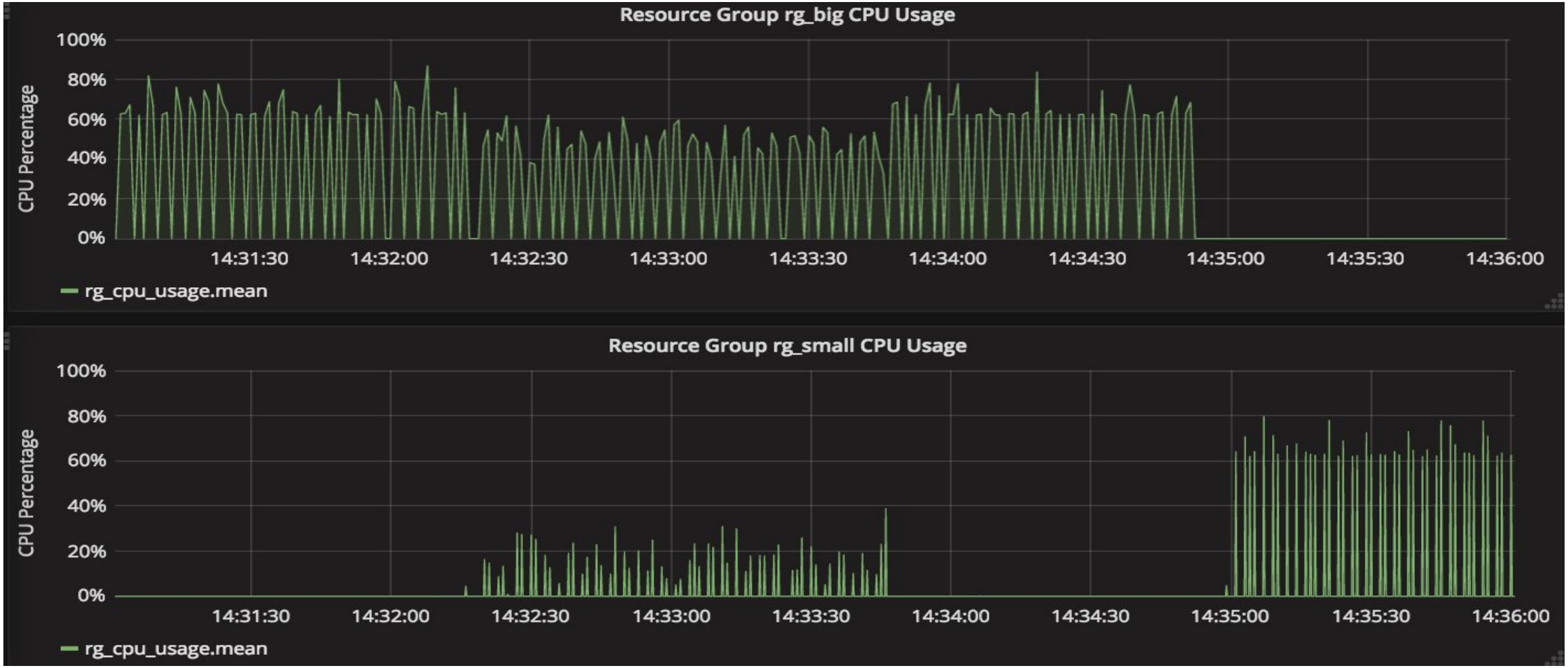
- 历史数据和不常访问的数据存储在 HDFS 或者其他外部系统中
- 无缝查询所有数据
- Text, CSV, Binary, Avro, Parquet 格式

并发管理

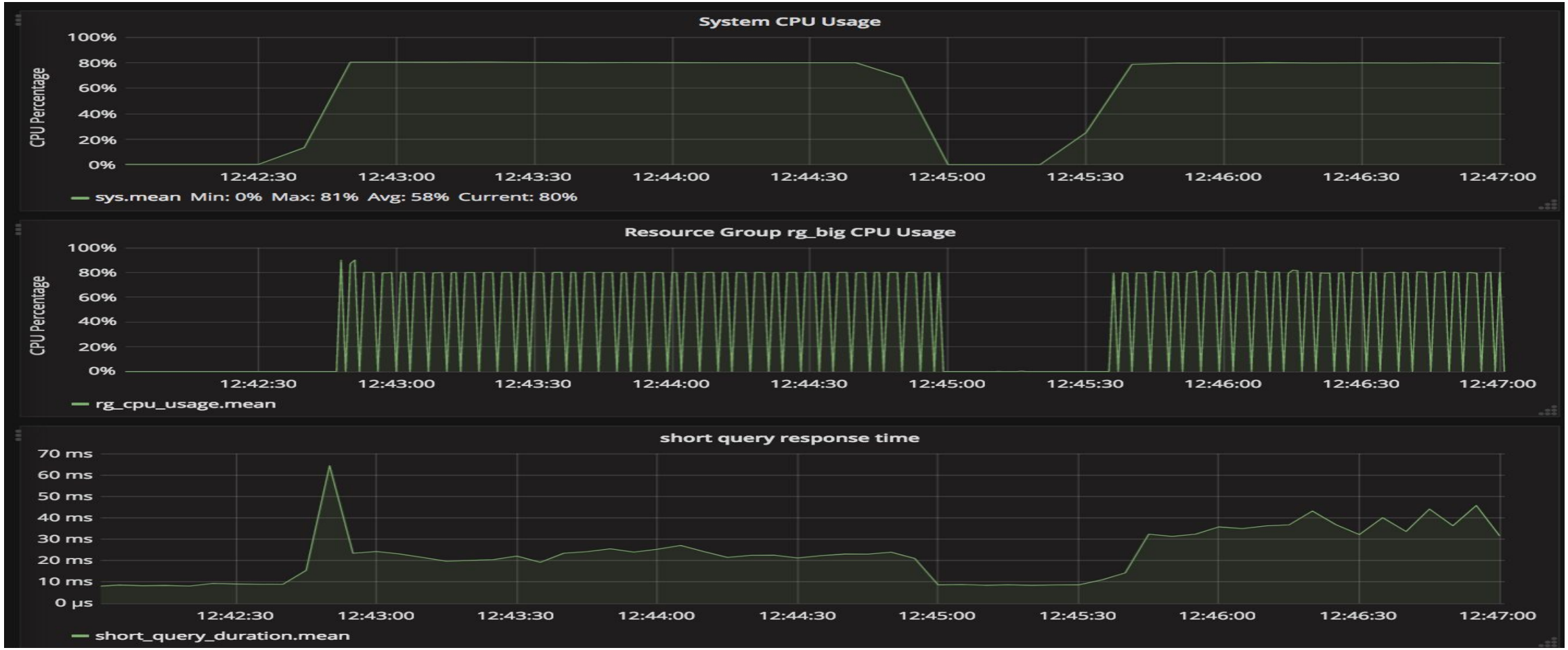
- pgbouncer
- 资源组（resource group）

```
create resource group rg1 (cpu_rate_limit=20, memory_limit=10, concurrency=5)
```

资源管理:CPU使用受限和超限



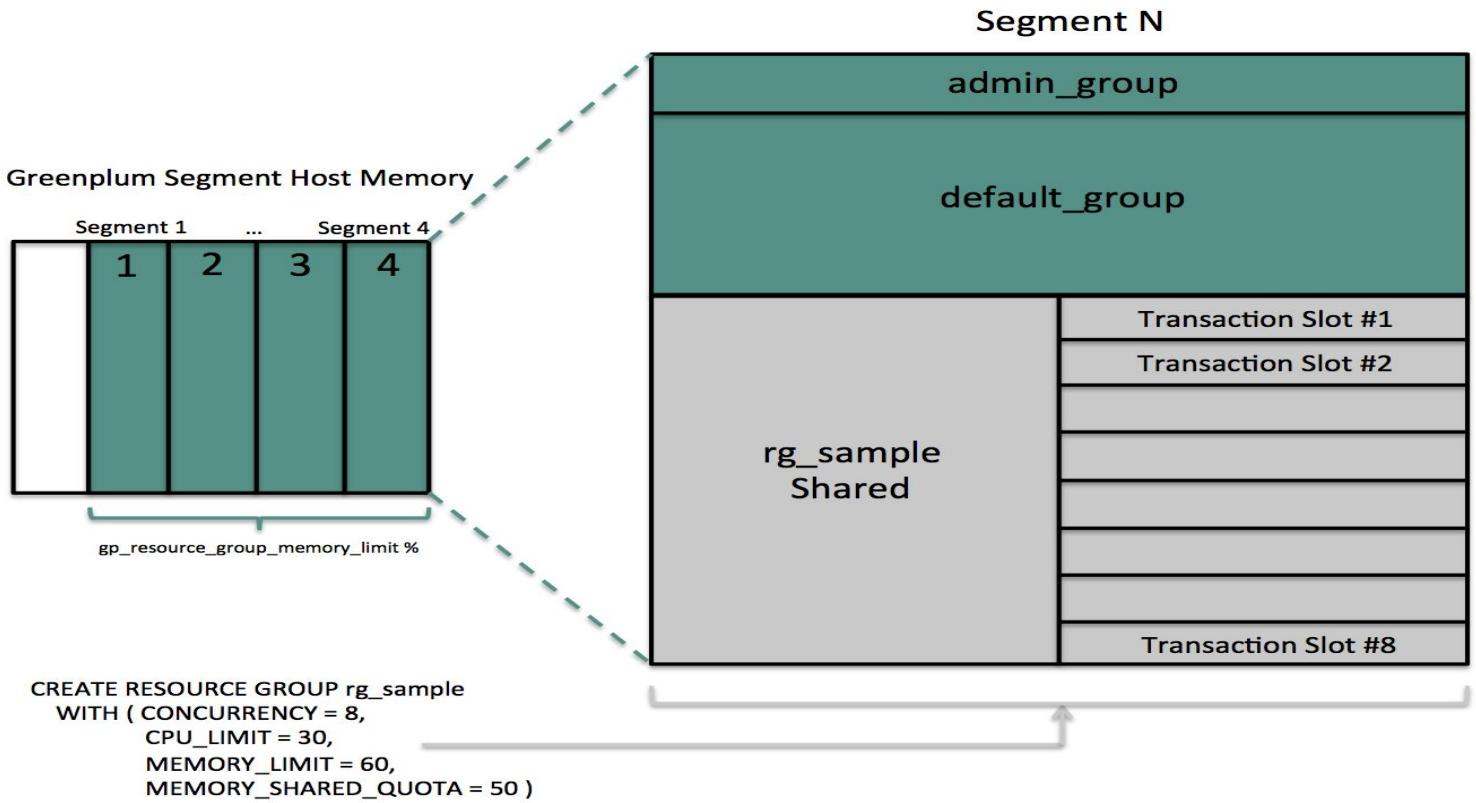
资源管理:CPU，短查询延迟



- 更稳定延迟，CPUSET特性：**create resource group rg1 (cpu_set='4,5', memory_limit=10, concurrency=5)**

资源管理:内存

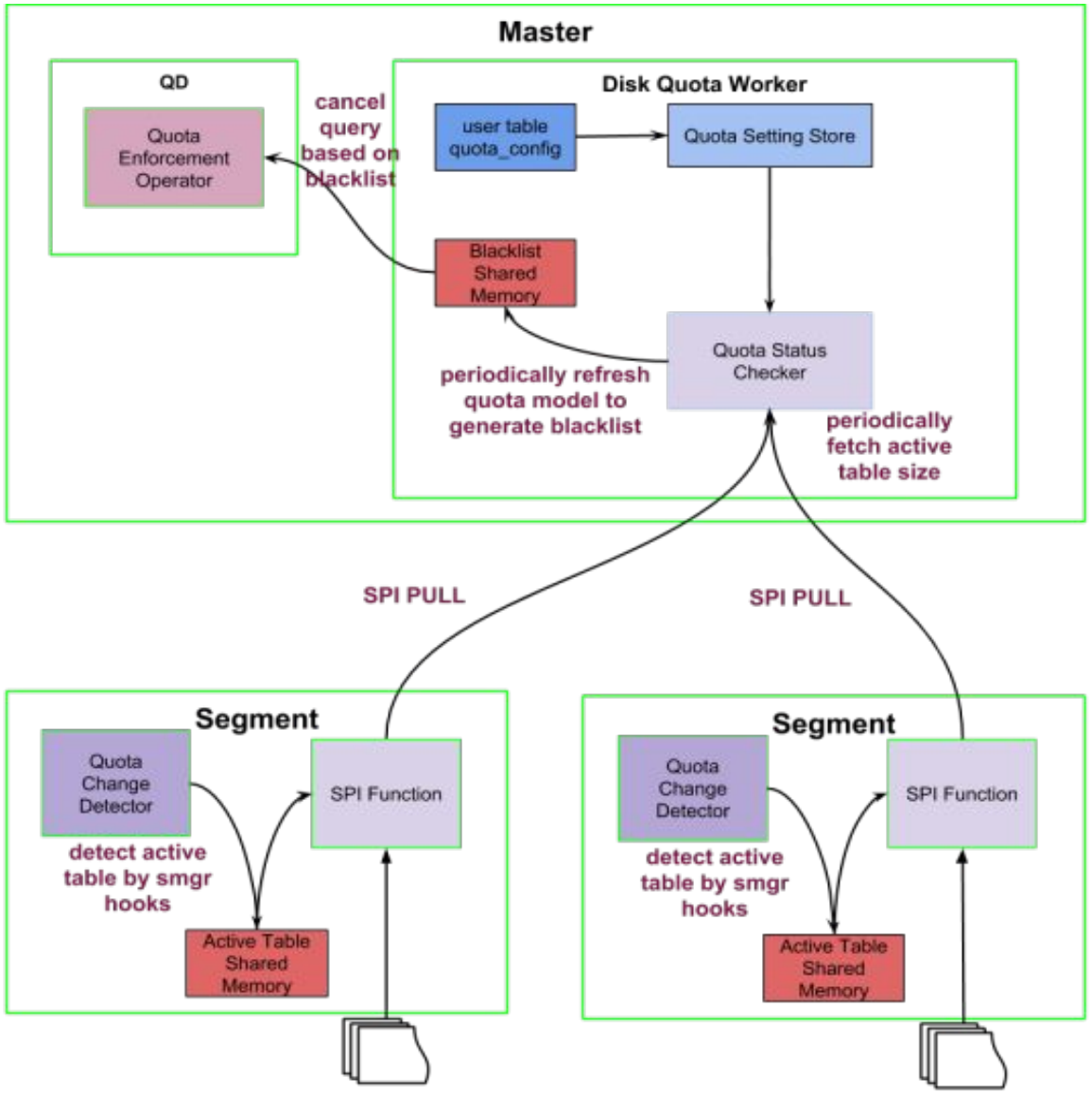
- 隔离
  - segment级
  - 资源组
  - 查询
- 共享
  - 全局segment级
  - 资源组内



资源管理:磁盘配额

SELCT diskquota.set_schema_quota ('s1', '1 MB');

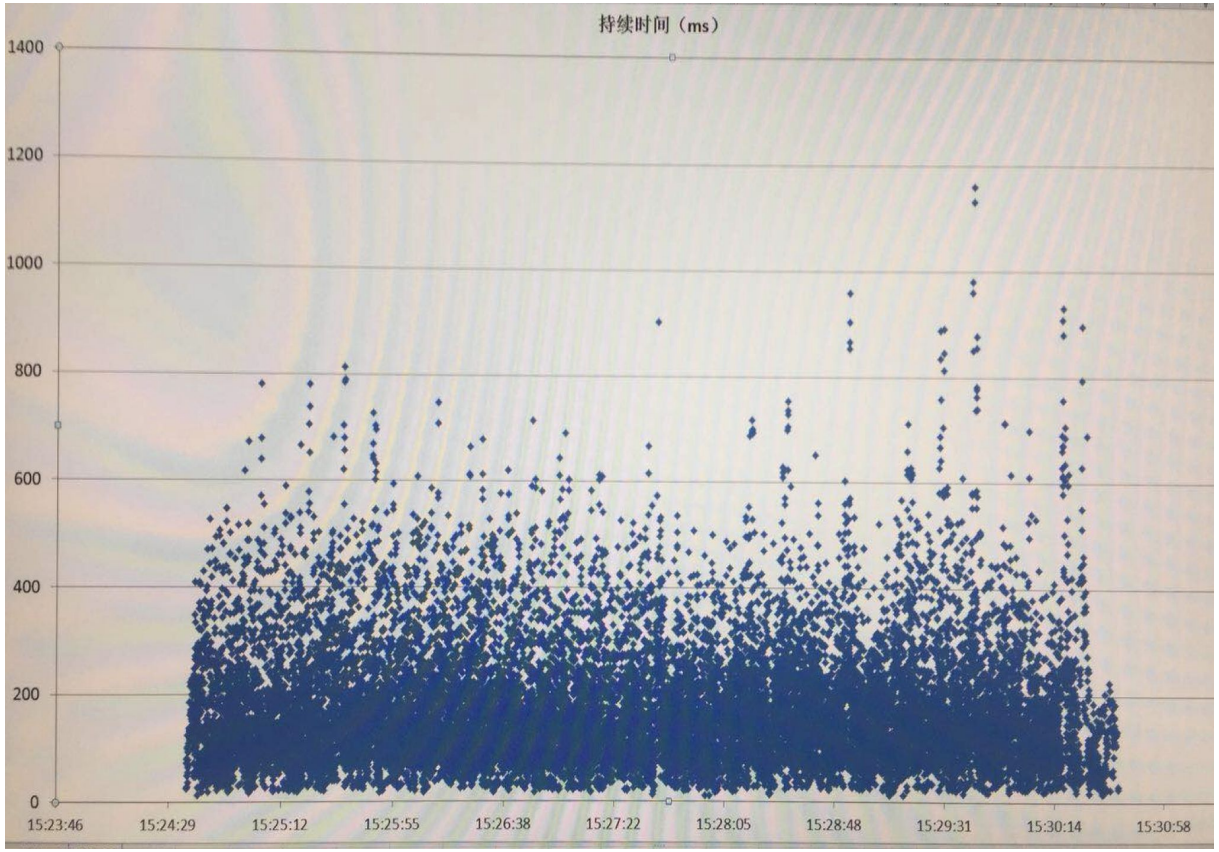
SELECT diskquota.set_role_quota ('u1', '1 MB');



客户案例

- 通过kafka近实时(500ms~1s)间隔加载:100万/s
- 简单查询1000并发:1s内返回
- 复杂关联查询:s级返回

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



展望

Greenplum 6.x/7

- PostgreSQL合并：BRIN索引和并行扫描
- 锁和事务的优化
- 磁盘IO的资源管理
- 更多思路？

资源

- 中文社区：http://greenplum.cn
- 文档：https://gpdb.docs.pivotal.io/6-0Beta/main/index.html
- 代码：https://github.com/greenplum-db/gpdb

由于微信群组规定限制，请首先添加这个入群助手个人微信，会拉您入群

QQ群