

CONTENTS

目录

1

Greenplum简介

2

应用实践

3

平台化

CONTENTS

目录

1

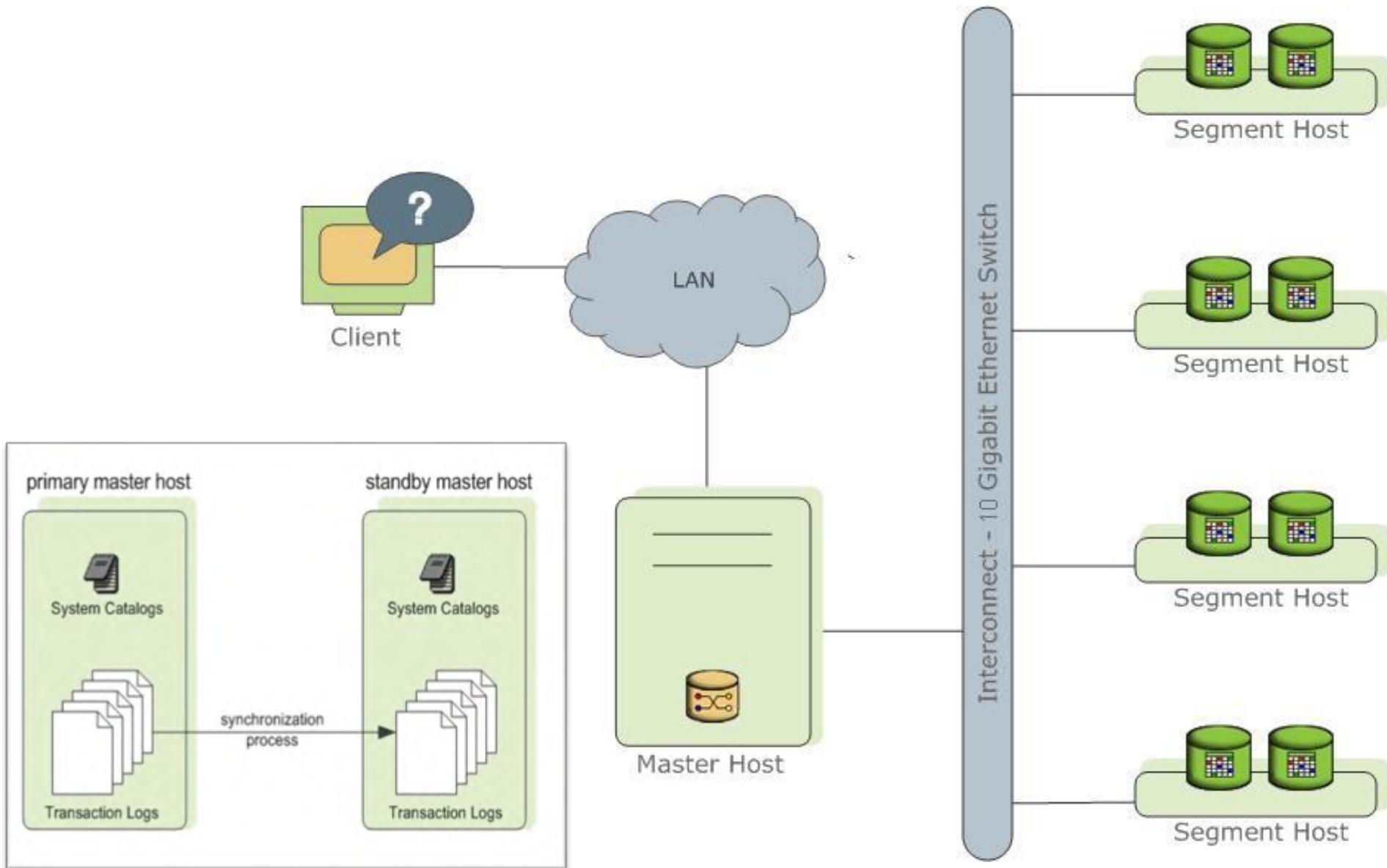
Greenplum简介

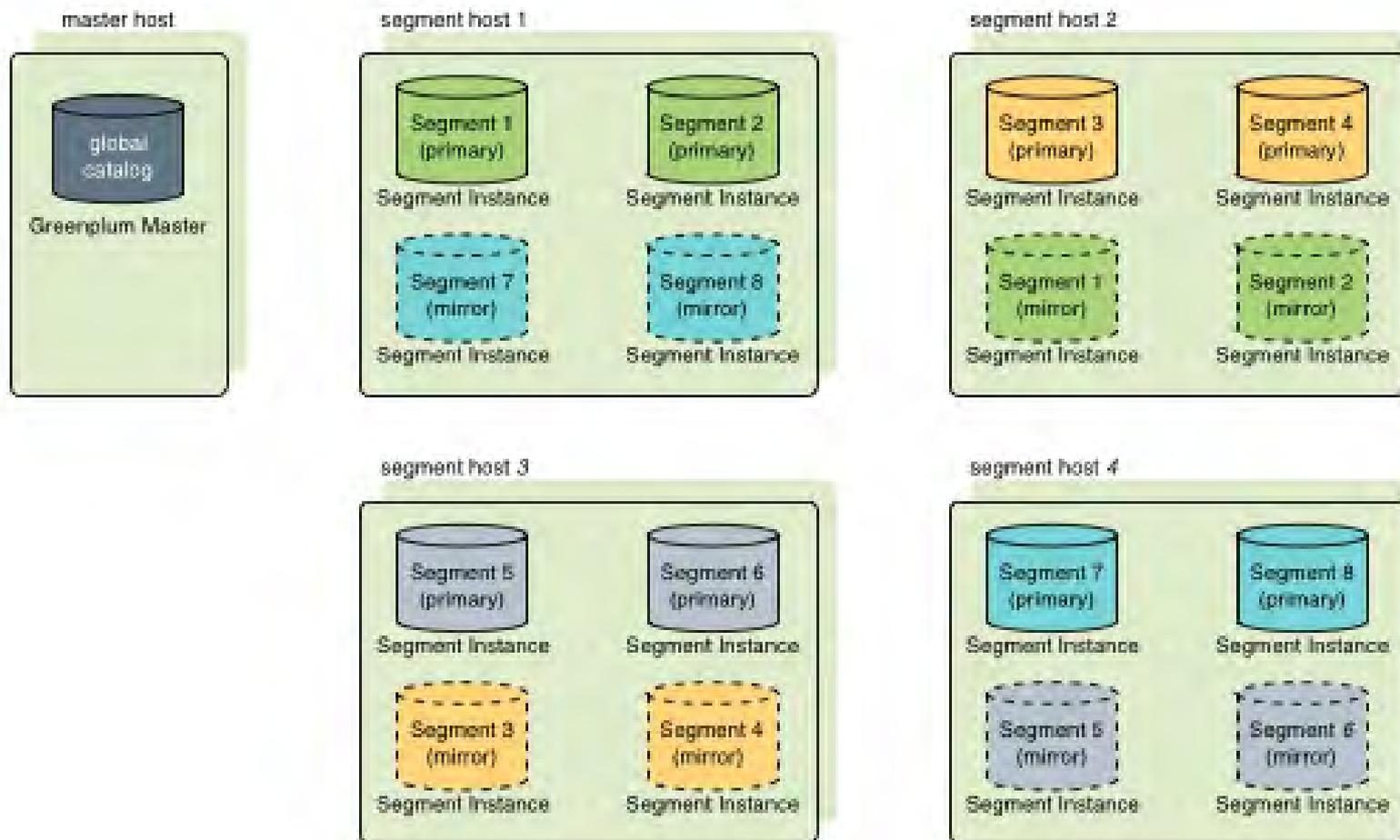
2

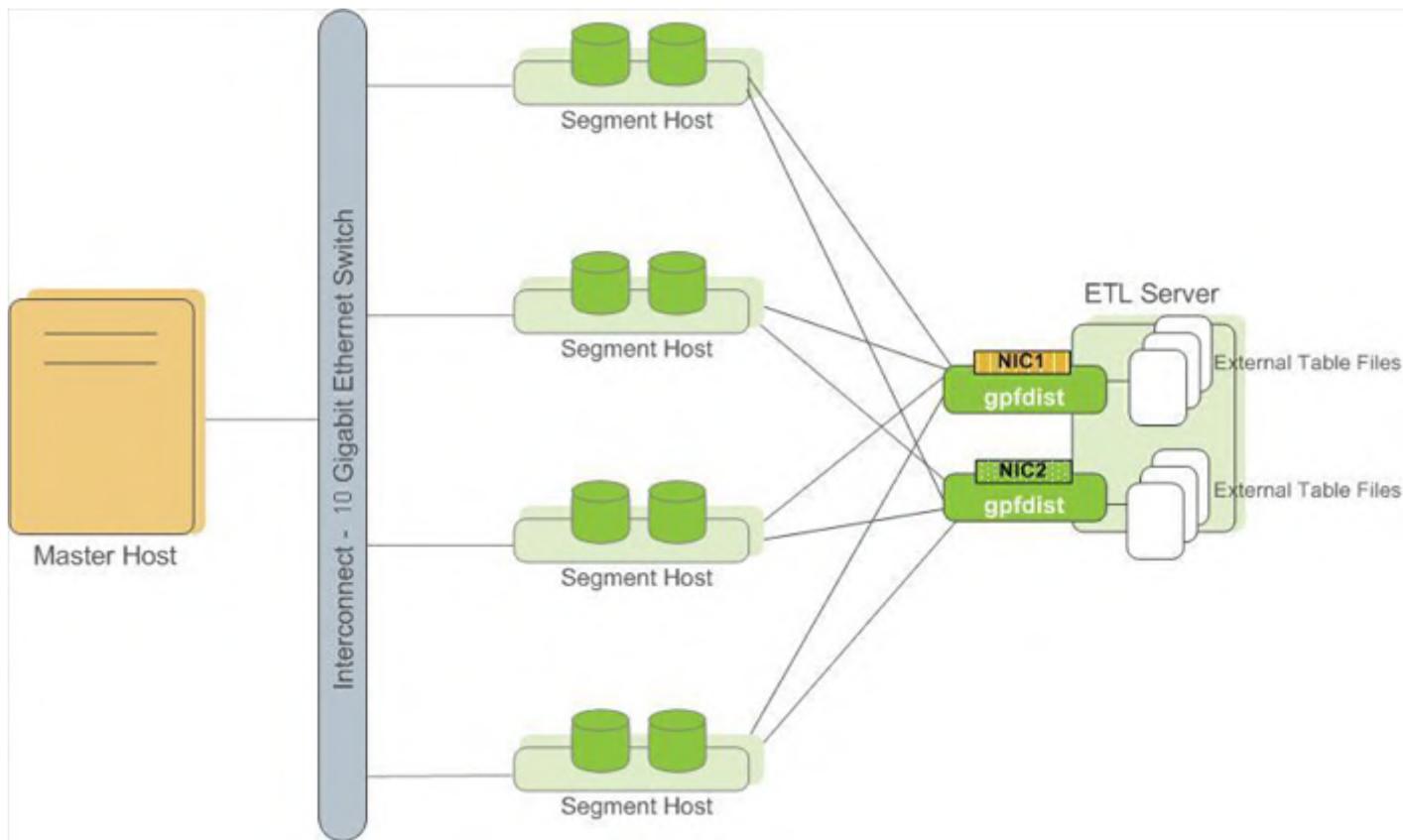
应用实践

3

平台化







CONTENTS 目录

- 1 Greenplum简介
- 2 应用实践
- 3 平台化

■ 极个别节点存储资源不足

查看数据分布情况

```
select gp_segment_id,count(*) from gp_test group by gp_segment_id order by 1;
```

■ SQL执行效率差

- 最慢的节点会成为系统的瓶颈
- 不必要的广播和重分布引起性能问题

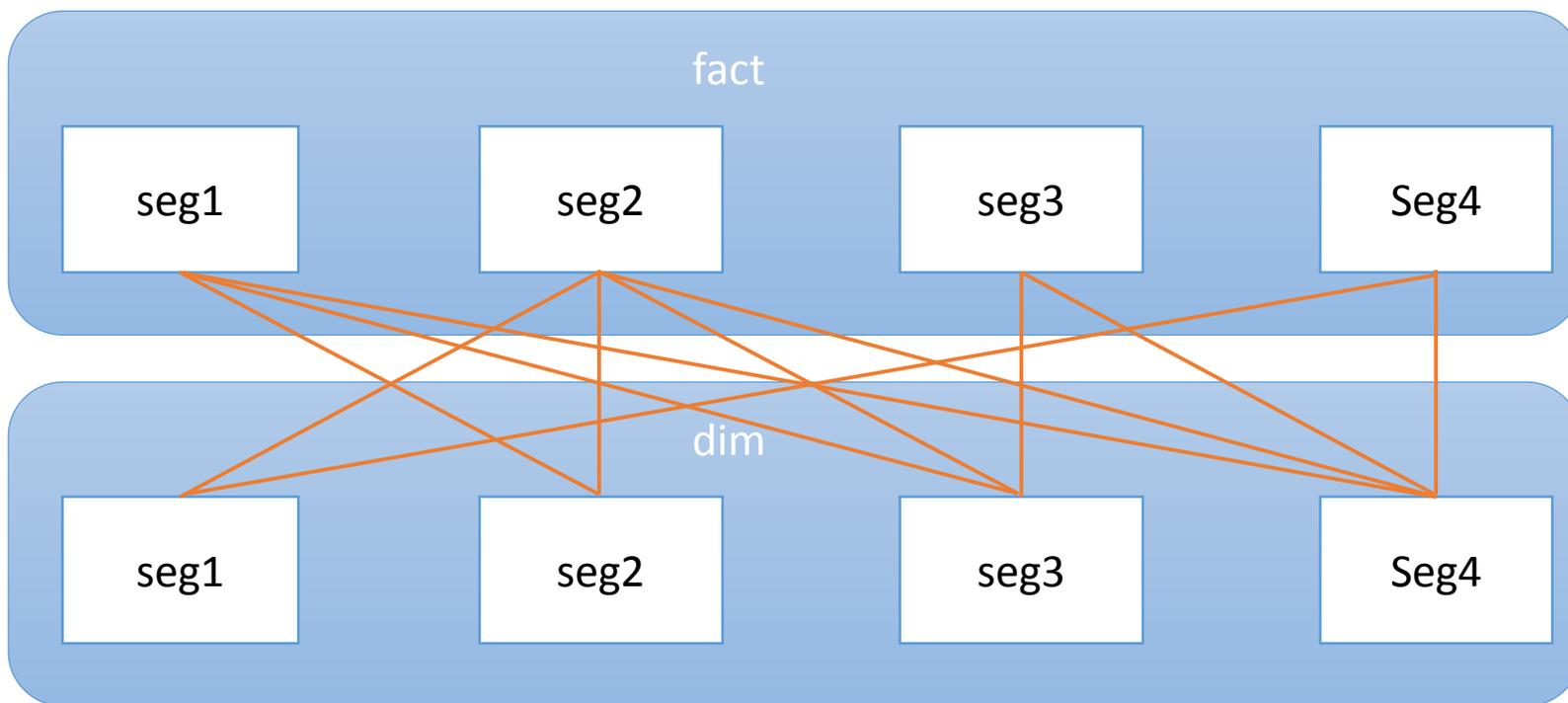
一个水桶无论有多高，它盛水的高度取决于其中最低的那块木板。

——木桶理论

```
select count(1) from dim d join fact f on d.shop_id =f.shop_id;
```

fact: distributed by(city_id)
Dim: distributed by(shop_id)

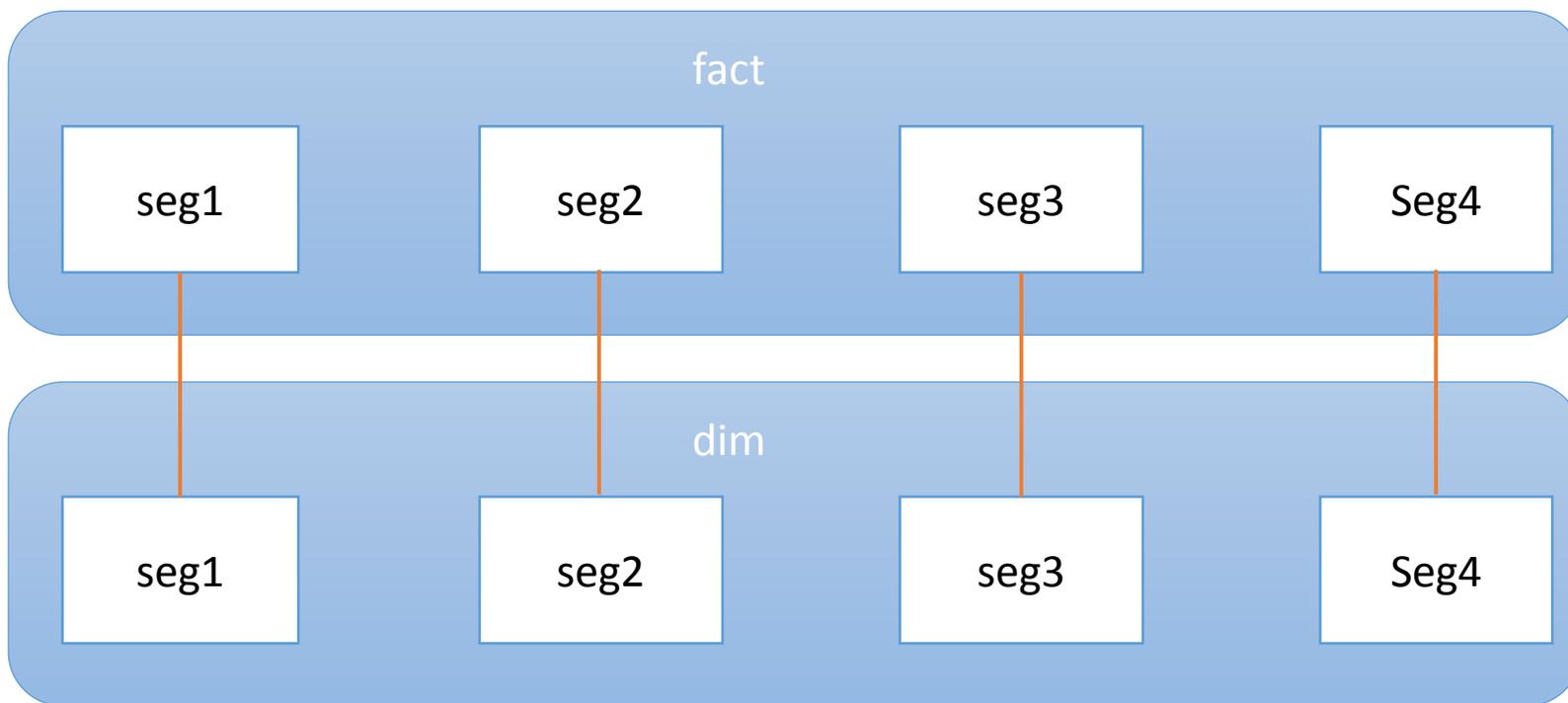
重分布或者广播



```
select count(1) from dim d join fact f on d.shop_id =f.shop_id;
```

fact_order: distributed by(**shop_id**)
Dim_business: distributed by(shop_id)

Segment内部join操作



1

选择需要Join的列，优先考虑并发高的列

2

防止数据倾斜

3

避免条件字段

4

考虑一下随机分布

■ 数据移动

- Broadcast Motion：每个节点向其它节点广播需要发送的数据
- Redistribute Motion：利用join列值hash不同，将筛选后的数据在其他segment重新分布
- Gather Motion：每个节点将join后的数据发到单节点去处理（一般是master）

■ 关联操作

- Hash Join： $O(n)$
- Merge Join： $O(n \cdot \log N)$
- Nested Loop： $O(N \cdot N)$

■ 两种聚合

- HashAggregate
- GroupAggregate

Query Plan:

-> Aggregate (cost=543353.77..543353.78 rows=1 width=8)

Slice 2

-> Gather Motion 232:1 (cost=543351.42..543353.76 rows=1 width=8)

-> Aggregate (cost=543351.42..543351.43 rows=1 width=8)

-> Hash Join (cost=155446.02..524212.24 rows=32999 width=0)
Hash Cond: f.shop_id = d.shop_id

Slice 1

-> Redistribute Motion 232:232 (cost=0.00..264111.23 rows=15447 width=8)
Hash Key: f.shop_id

-> Append (cost=0.00..192438.41 rows=15447 width=9)

-> Append-only Scan on fact_..._key_20170520 f (cost=0.00..68315.92 rows=8065 wi

-> Seq Scan on fact_..._key_20170521 f (cost=0.00..124122.49 rows=7383 width=9)

-> Hash (cost=115284.90..115284.90 rows=13849 width=8)

-> Seq Scan on dim_... d (cost=0.00..115284.90 rows=13849 width=8)

Query Plan:

-> Aggregate (cost=471335.92..471335.93 rows=1 width=8)

Slice 1

-> Gather Motion 232:1 (cost=471333.56..471335.90 rows=1 width=8)

-> Aggregate (cost=471333.56..471333.57 rows=1 width=8)

-> Hash Join (cost=155446.02..452191.16 rows=33005 width=0)
Hash Cond: f.shop_id = d.shop_id

-> Append (cost=0.00..192072.50 rows=15450 width=8)

-> Append-only Scan on fact_..._key_20170520 f (cost=0.00..67891.92 rows=8065 width=8)

-> Seq Scan on fact_..._key_20170521 f (cost=0.00..124180.58 rows=7386 width=8)

-> Hash (cost=115284.90..115284.90 rows=13849 width=8)

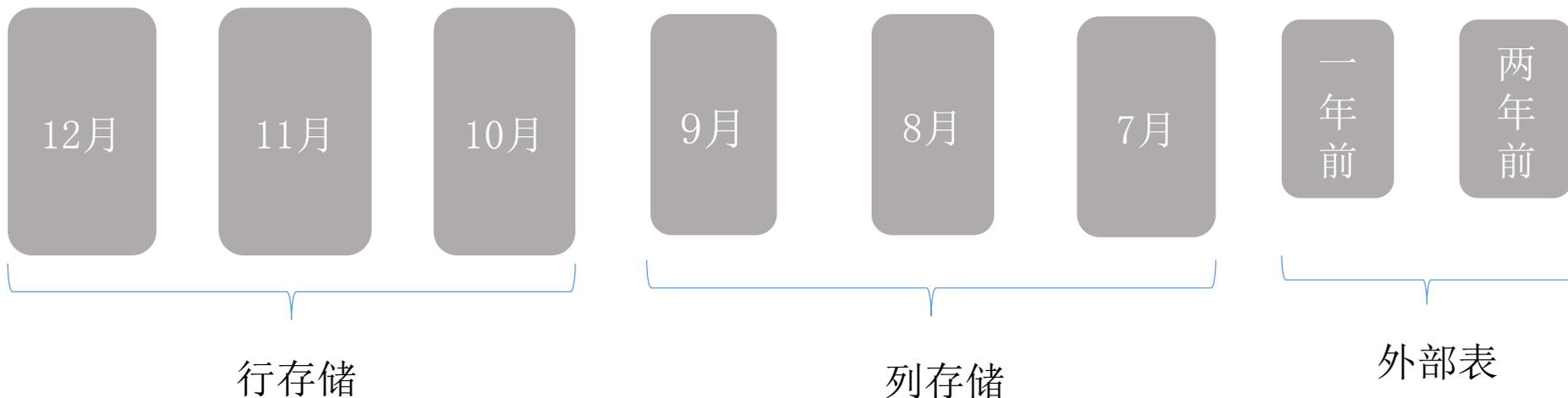
-> Seq Scan on dim_... d (cost=0.00..115284.90 rows=13849 width=8)

- GPDB基于cost base的查询计划，不正确的统计信息导致错误的执行计划
- 使用ANALYZE命令收集统计信息
- 仅对必要的字段执行收集统计信息，不在join,where,order等操作中的字段无需收集
- EXPLAIN [ANALYZE]分析执行计划
 - 分区裁剪
 - 广播或者重分布

- 一般不需要创建索引
- 对于条件筛选性强的SQL，建议增加索引
- 区分度高的建议使用B-Tree索引，区分度低得建议使用Bitmap索引

- 频繁查询的宽表使用列存，其他表使用行存
- 过多的使用列存，会造成小文件过多
- 对于经常更新的表，使用行存
- 分区表支持行列混合存储
- 不建议通过adhoc查询外部表

表（以订单表为例）



- 通过资源队列控制集群资源消耗，避免系统超负荷运行
- 基于ROLE体系的资源队列，为所有ROLE分配资源队列
- 设置队列优先级，HIGH > MEDIUM > LOW
 - 在并发争用CPU资源时，高优先级资源队列可以获得更多的CPU资源
 - 资源队列的缺省优先级为MEDIUM
- 动态资源队列

- 不同时段各队列的资源不同
- 不同负载各队列的资源不同

选择创建资源队列集合模板:

集合名称	时间类型	日期范围	开
default	-	-	-
生产队列	天类型	-	00
生产队列2	天类型	-	21

创建资源队列集合

- 集合名称:
- 时间类型: 天
- 开始时间: 00:04
- 结束时间: 09:00

名称	优先级	qj配额限制	总cross最大限值	空间限制是否允许突破最大xroot	内存限制	最小本地盘xroot
adhoc	medium	100	-1	否	3000MB	0
pg_default	medium	20	-1	否	-1	0
production	medium	20	-1	否	7000MB	0

集群通过安全可以修改

取消 确认

现象

- 用户报查询队列满，监控发现资源利用率很低，怀疑SQL被锁

如何查找原因

- `gp_toolkit.pg_stat_activity`：大量sql处于waiting状态，怀疑被锁
- `gp_toolkit.gp_locks_on_relation`：查看被锁状况

解决方法

- 查找被哪个会话锁住，等待结束或者cancel掉该会话

■ 系统维护：

- 对系统表定期做vacumm analyze，对经常更新的表做vacumm
- Down节点自动拉起，IDLE连接定时查杀
- 参数调优（深入理解GPDB，系统表）

■ SQL优化

- 分布键（数据均匀分布）
- 统计信息（正确）
- 高耗SQL优化
- 慢查询优化

■ 资源队列，合理的资源划分，动态调整

CONTENTS

目录

1

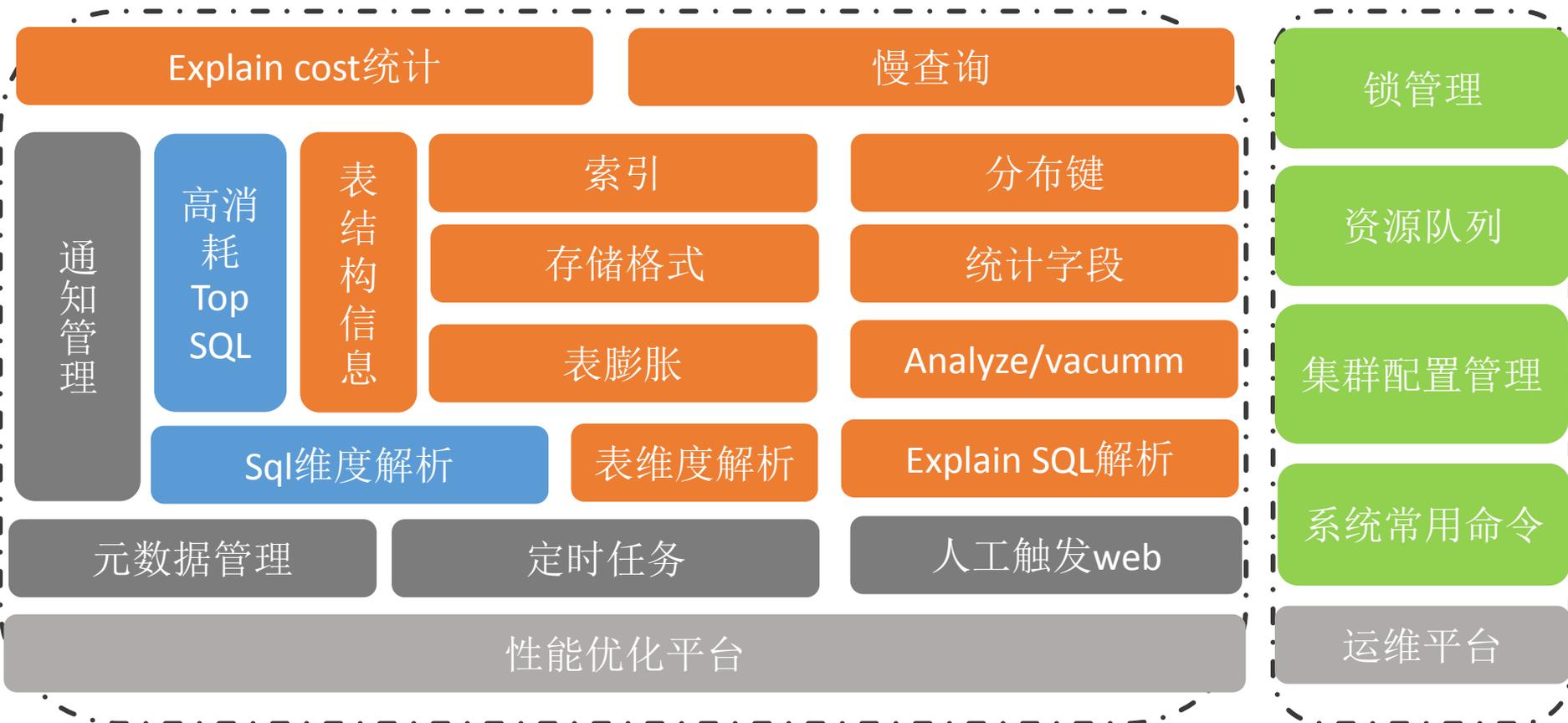
Greenplum简介

2

应用实践

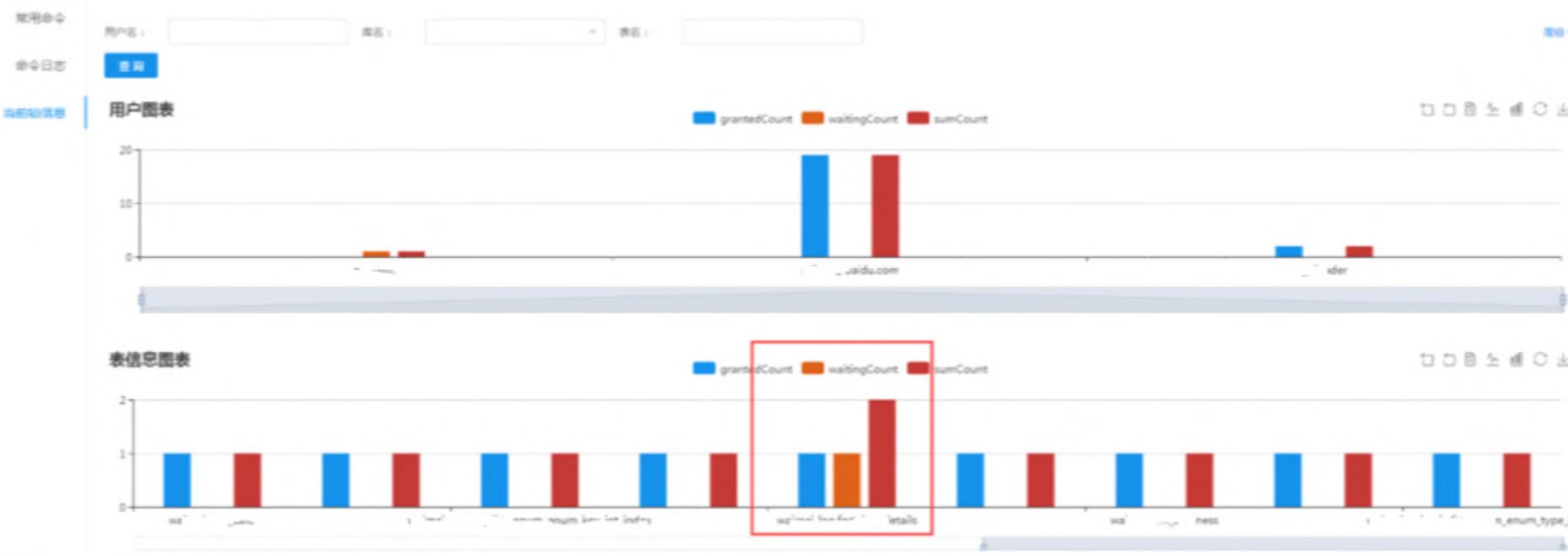
3

平台化



详情信息

库名	表名	锁模式	进程id	用户名	始时间	segmentId	查看	操作
		AccessExclusiveLock	6409		16:03	-1	被本锁阻塞的锁 导致本锁阻塞的锁	删除SessionID





小男孩

首页

通知

资源队列

膨胀表

慢查询

代价统计

运维

配置

GPCC

设置

聚合方式：

时间： -

adhocId	花费时间	默认Schema	查询执行时间	结果数目	用户名	查询语句	sqlInfold	操作
41683335	10h	xxxxxx	2017-05-27 08:34:02	25	xxxxxx@baidu.com	<code>select dim_date.d...</code>	1697573	<input type="button" value="查看"/>
41683283	10h	xxxxxx	2017-05-27 08:33:17	8	xxxxxx@baidu.com	<code>select dim_date.d...</code>	1697533	<input type="button" value="查看"/>
41683284	10h	xxxxxx	2017-05-27 08:33:17	8	xxxxxx@baidu.com	<code>select dim_date.d...</code>	1697534	<input type="button" value="查看"/>
41683293	10h	xxxxxx	2017-05-27 08:33:20	2	xxxxxx@baidu.com	<code>select dim_date.d...</code>	1697543	<input type="button" value="查看"/>
41683286	10h	xxxxxx	2017-05-27 08:33:17	8	xxxxxx@baidu.com	<code>select dim_date.d...</code>	1697536	<input type="button" value="查看"/>
41683337	10h	xxxxxx	2017-05-27 08:34:03	8	xxxxxx@baidu.com	<code>select dim_date.d...</code>	1697575	<input type="button" value="查看"/>
41683327	10h	xxxxxx	2017-05-27 08:33:56	0	xxxxxx@baidu.com	<code>select dim_date.d...</code>	1697567	<input type="button" value="查看"/>
41683277	10h	xxxxxx	2017-05-27 08:33:16	1	xxxxxx@baidu.com	<code>select dim_date.d...</code>	1697528	<input type="button" value="查看"/>

SDCC 2017 深圳站
大数据技术实战峰会

CSDN

THANKS

<http://waimai.baidu.com>