中国开源软件推进联盟 China Open Source Software Promotion Union

IT大咖说 知识分享平台

腾讯云

PostgreSQL@Tencent

许中清

腾讯云数据库架构师

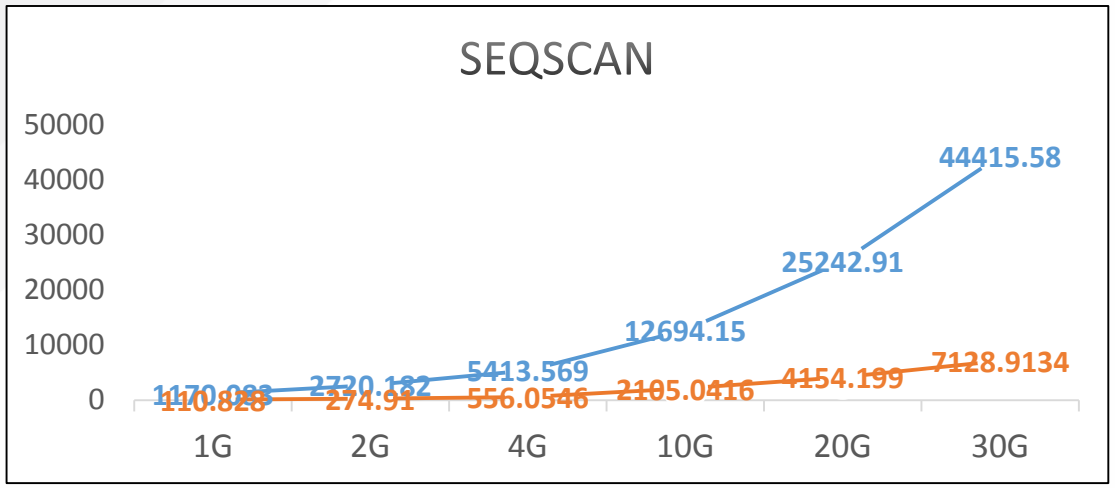
内容

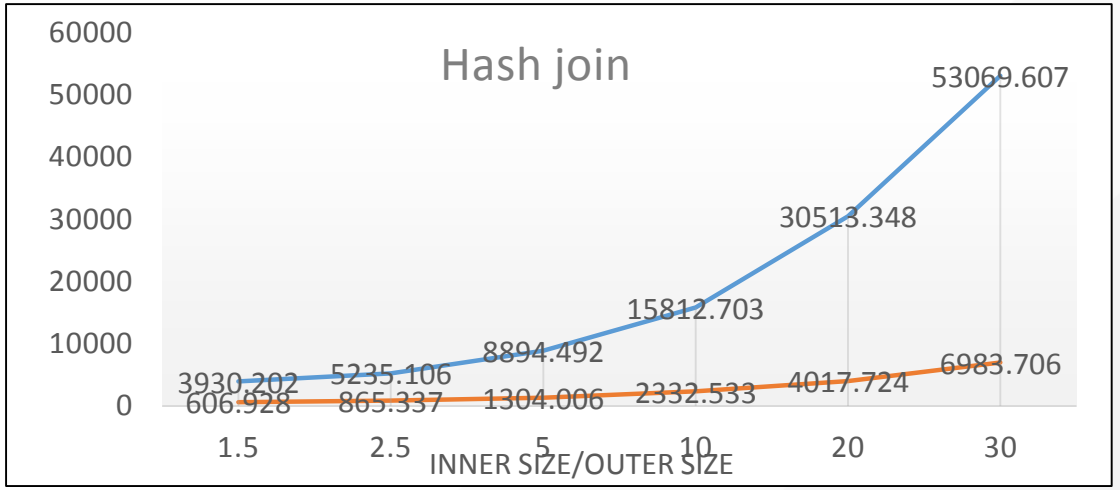
1. CDB for PostgreSQL
2. Postgres-XZ（PGXZ）

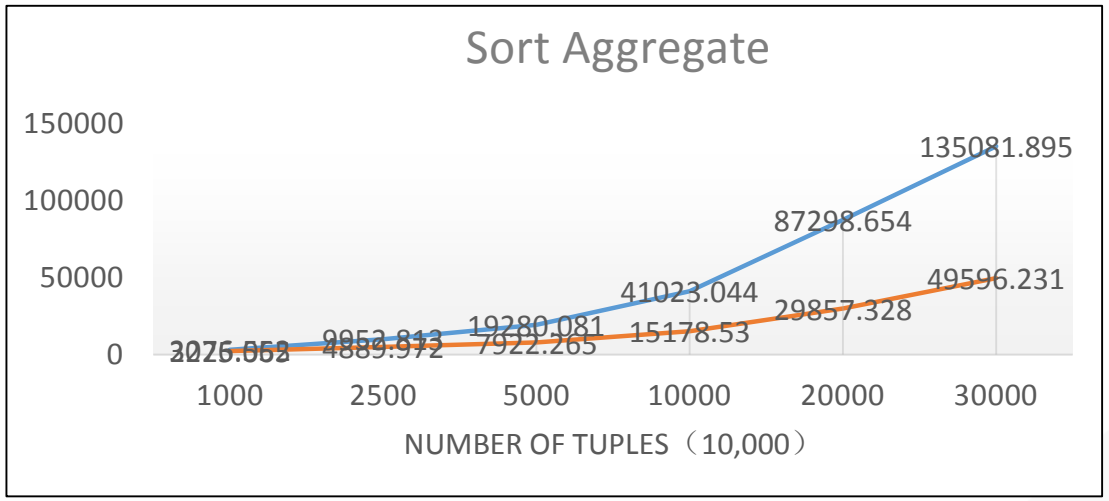
1.1 CDB for PostgreSQL

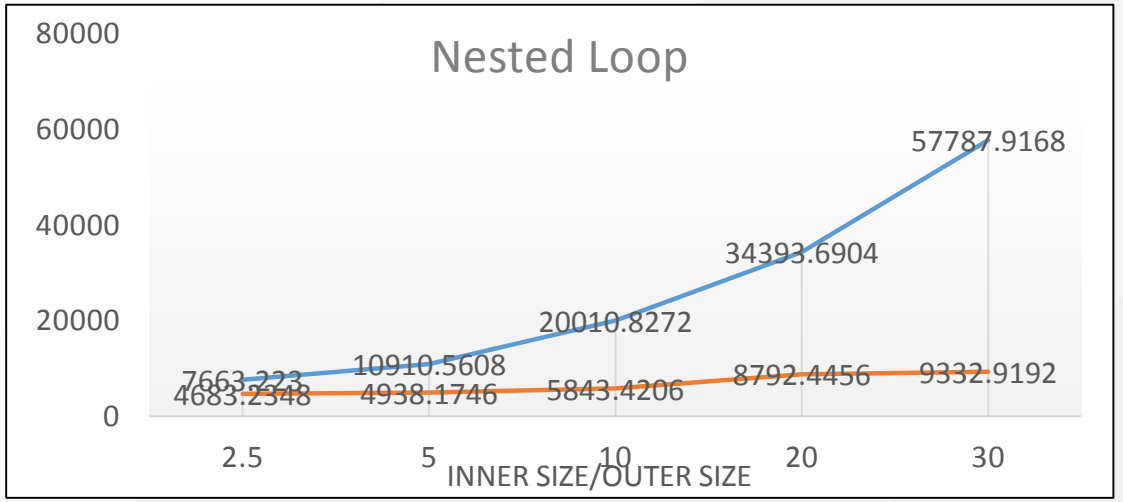
- 2016年2月发布
- 面向中小规模的用户
- 特性：
  - 自动化运维
  - 容灾/备份
  - 扩容
  - 并行执行
  - Vacuum Full Concurrently

1.2 CDB for PostgreSQL并行优化



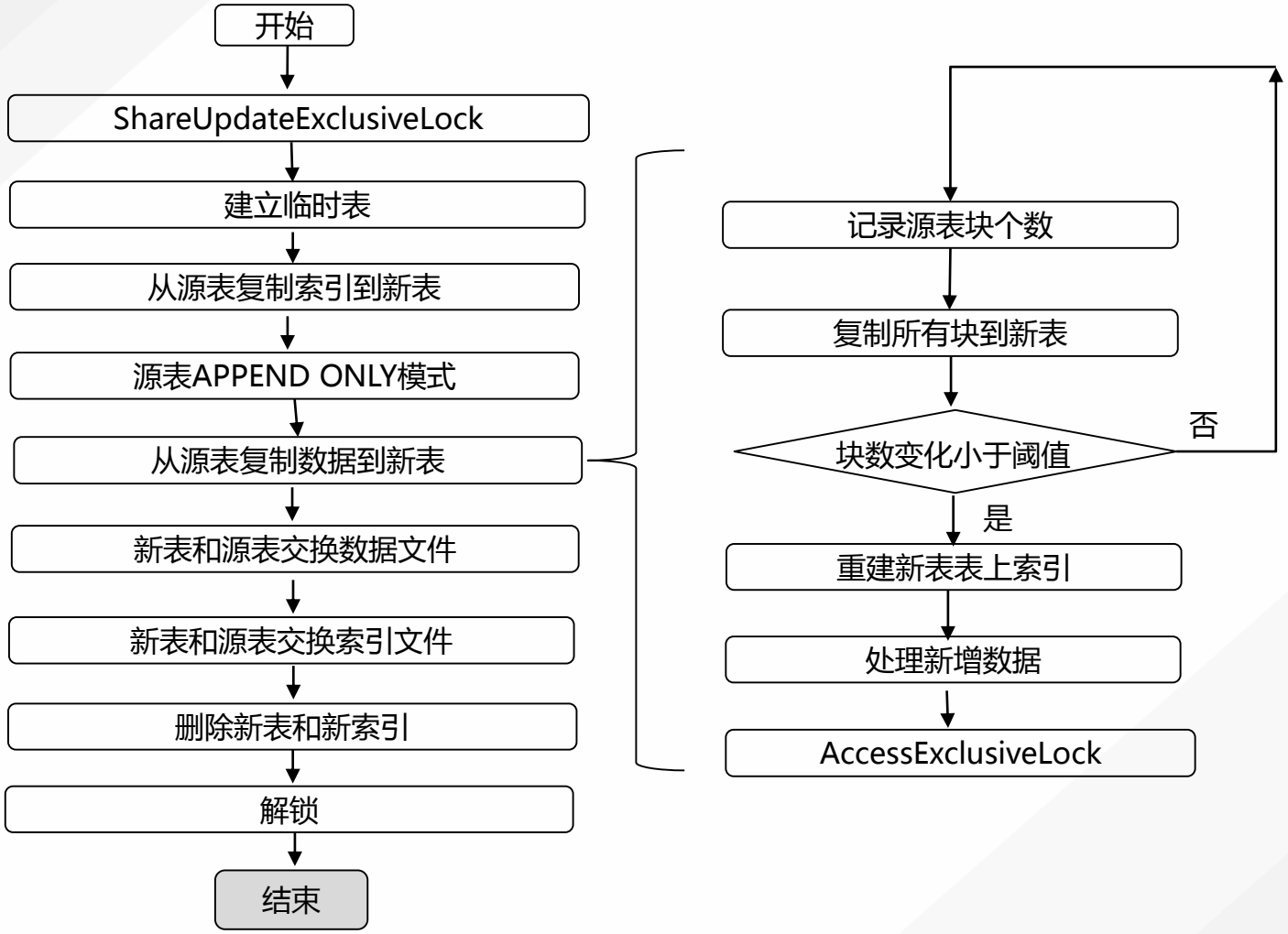






1.3 Vacuum Full Concurrently

方案：增量数据反复迭代



互斥关系

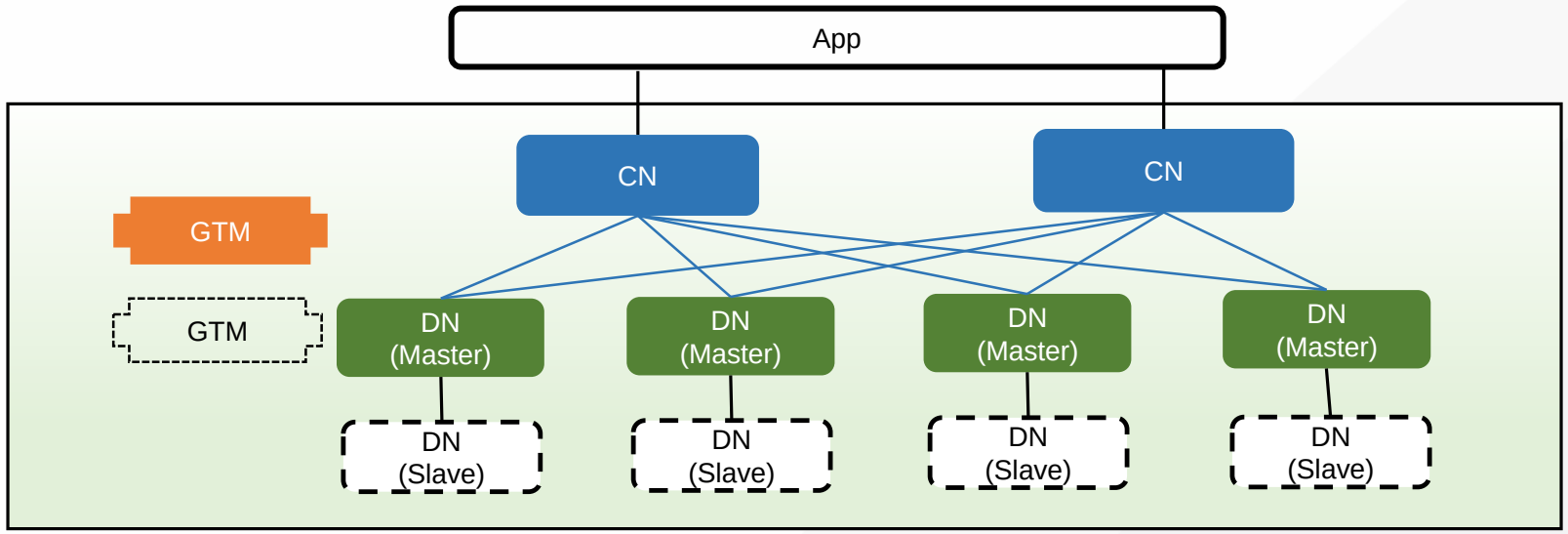
	VF Concurrently
VF Concurrently	BLOCK
VF	BLOCK
Auto Vacuum	BLOCK
DDL	BLOCK
DML	UNBLOCK
SELECT	UNBLOCK

2.1 PGXZ架构和现状

PGXZ的架构

PostgreSQL → Postgres-XC → Postgres-XZ (PGXZ)

- 主要面向交易型业务，兼顾部分分析类场景
- 支持分布式事务
- 支持跨节点复杂查询



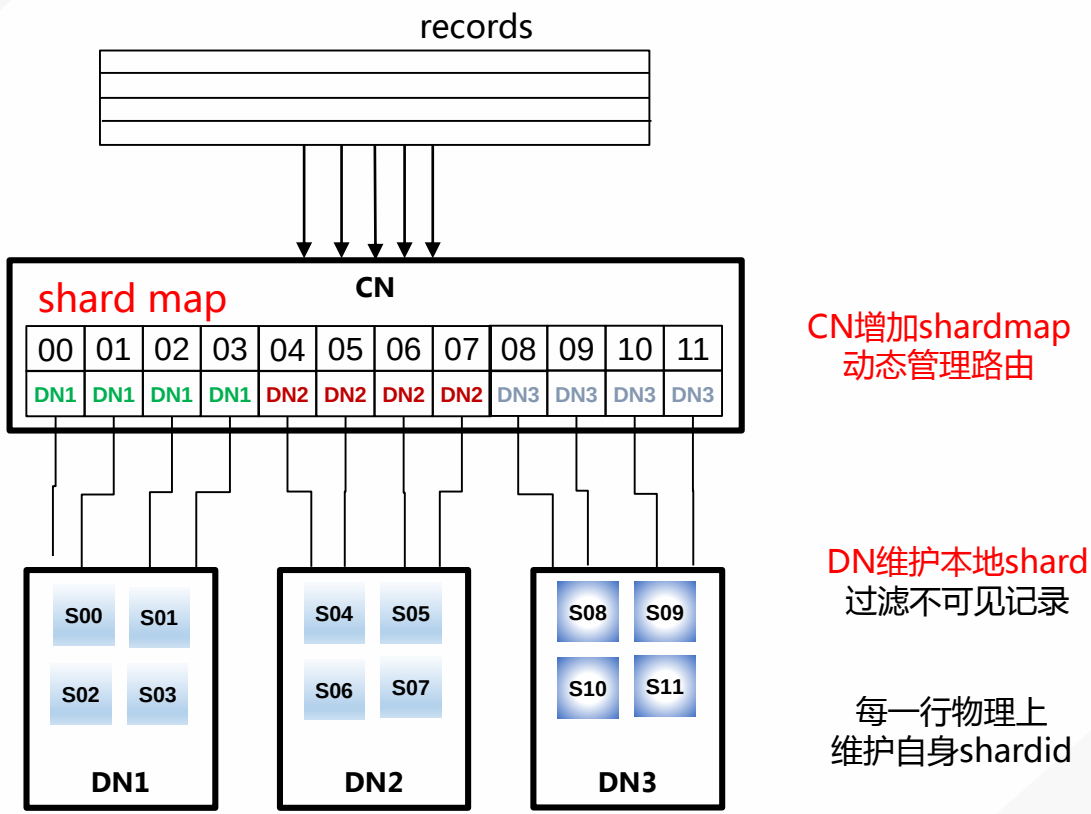
CN: Coordinator Node **DN:** Data Node **GTM:** Global Transaction Manager

2.2 PGXZ主要特性

1. 分布式事务
2. 跨节点的JOIN
3. 在线扩容
4. 冷热数据分治
5. Shardkey数据倾斜治理
6. 多中心部署
7. 滚动升级
8. 监控运维

3.1 PGXZ Data Sharding

PGXZ Sharding方案



关键点

关键点：

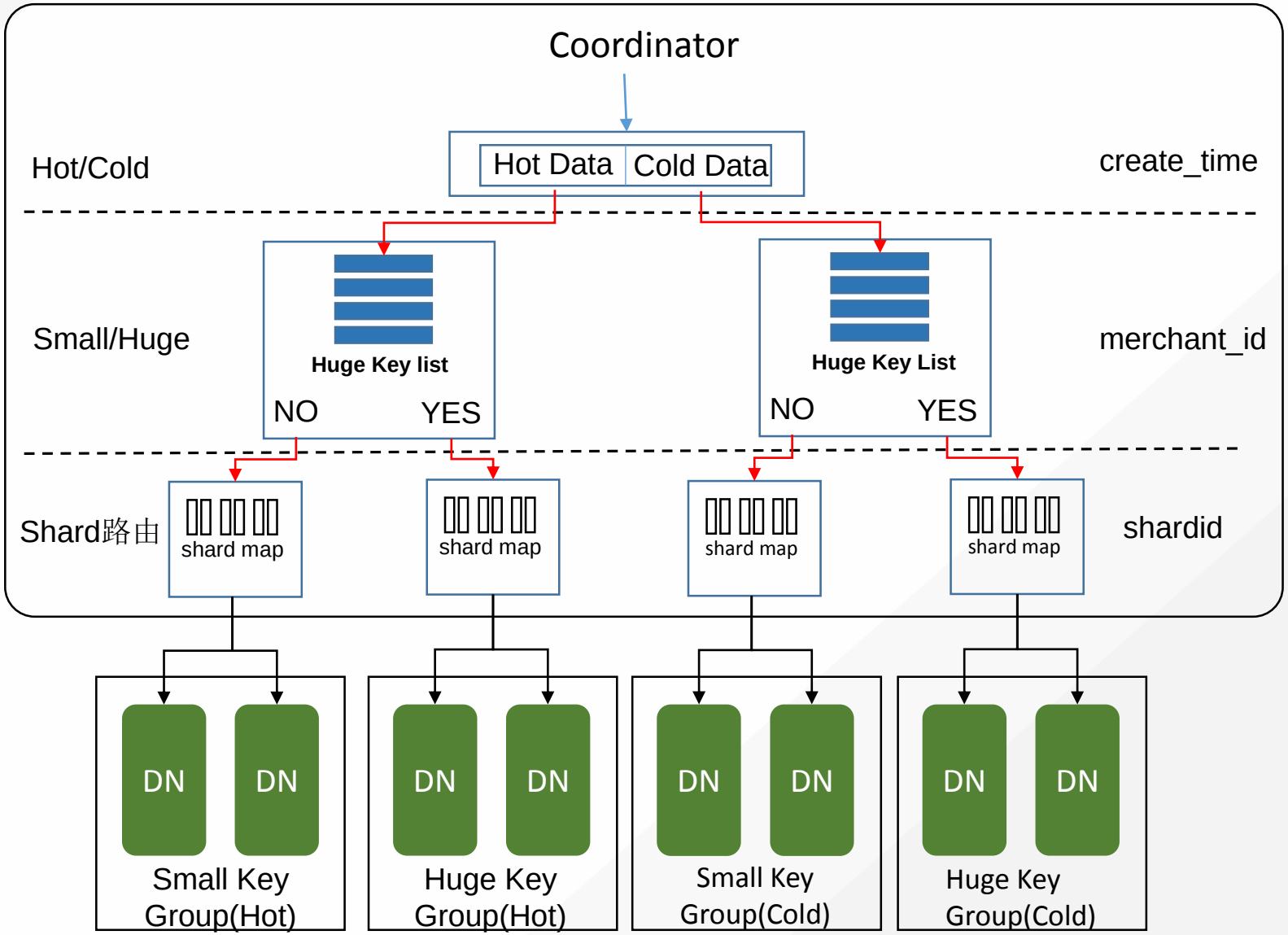
1. **DN过滤**：过滤临时副本

   查询条件没有分布key上的过滤条件。

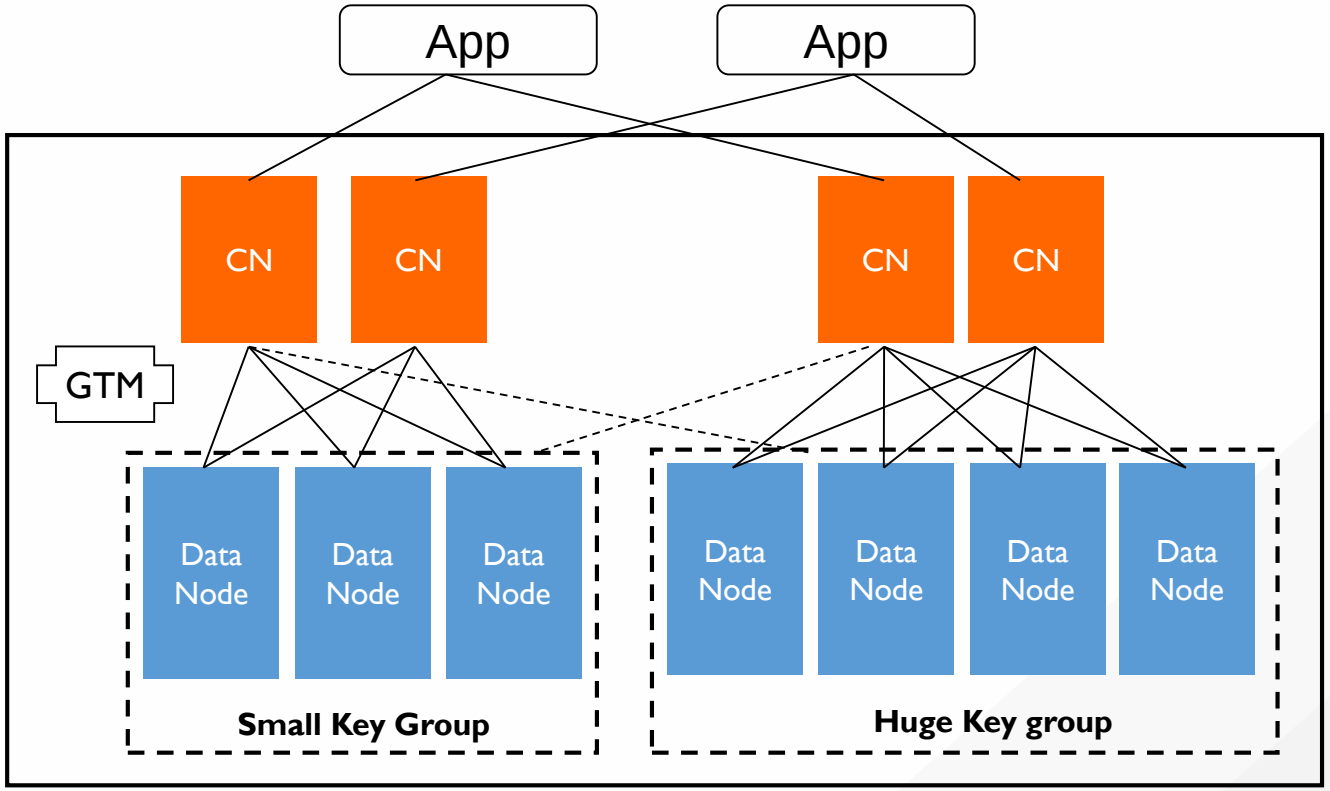
2. **路由一致性**：保证shardmap全局一致性
   1. 二阶段提交
   2. 按顺序生效

3. **Shard在线迁移**
   1. 一致性
   2. 业务无感知
   3. 快

3.2 多组：路由策略(4 Groups)



3.3 多组: 治理ShardKey倾斜



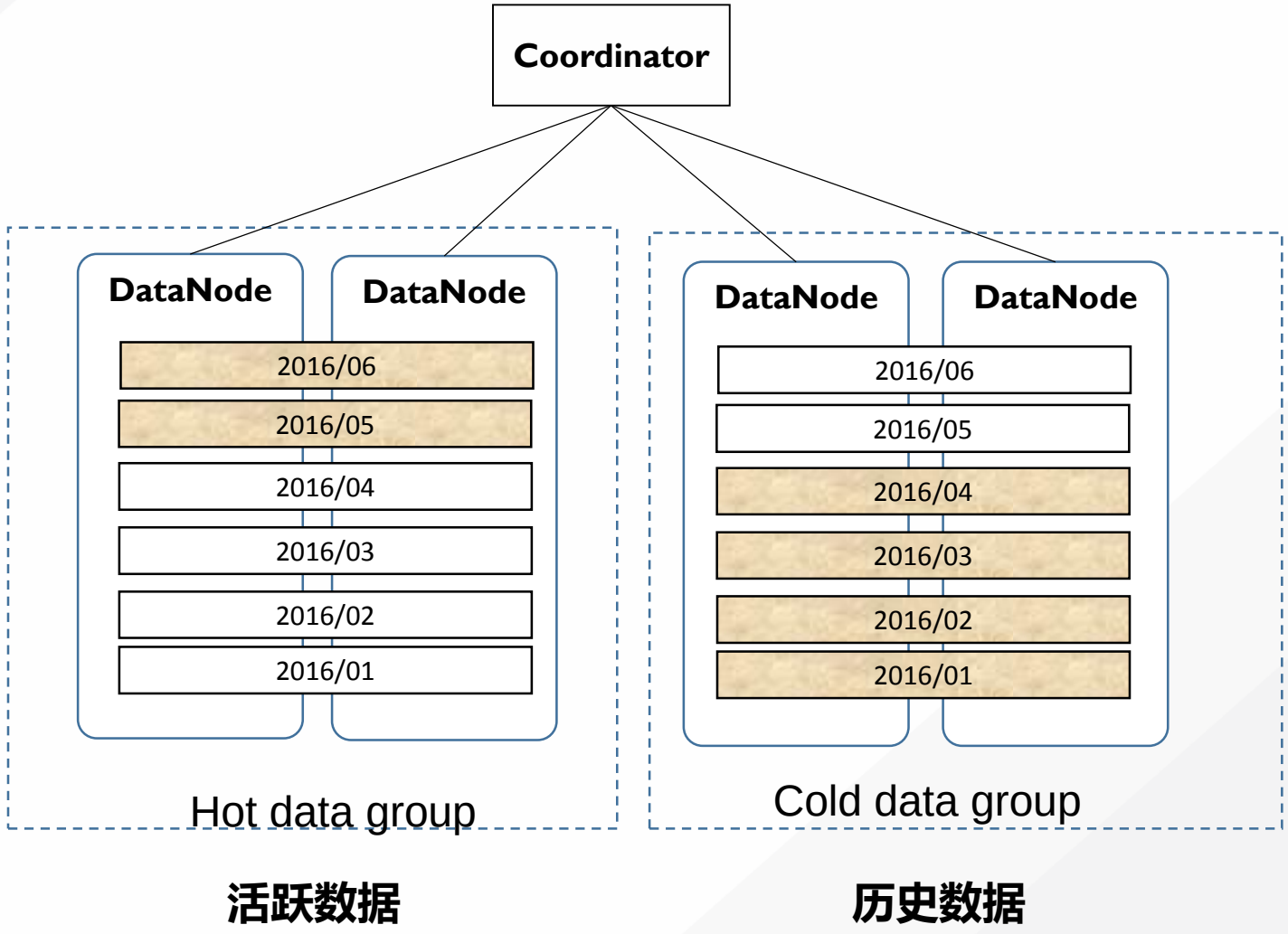
小商户组: 一个商户的数据只落在一个DataNode上.

- 在一个商户上的写不会引入分布式事务.
- 一个商户上的读不会有跨节点的Join.

大商户组: 一个大商户同一天的数据只落到一个DataNode；一个大商户的数据打散到组内的所有DataNode上.

- 超出单节点存储空间的大商户就可以存储到集群中了.

3.4 多组: 冷热数据分治


Coordinator
DataNode
DataNode
2016/06
2016/05
2016/04
2016/03
2016/02
2016/01
Hot data group
DataNode
DataNode
2016/06
2016/05
2016/04
2016/03
2016/02
2016/01
Cold data group
活跃数据
历史数据

4.1 PGXZ 在线迁移

- **组内迁移**(扩容/存储均衡)

  1. DataNode -> DataNode

- **组间迁移**

  2. ShardKey跨组迁移

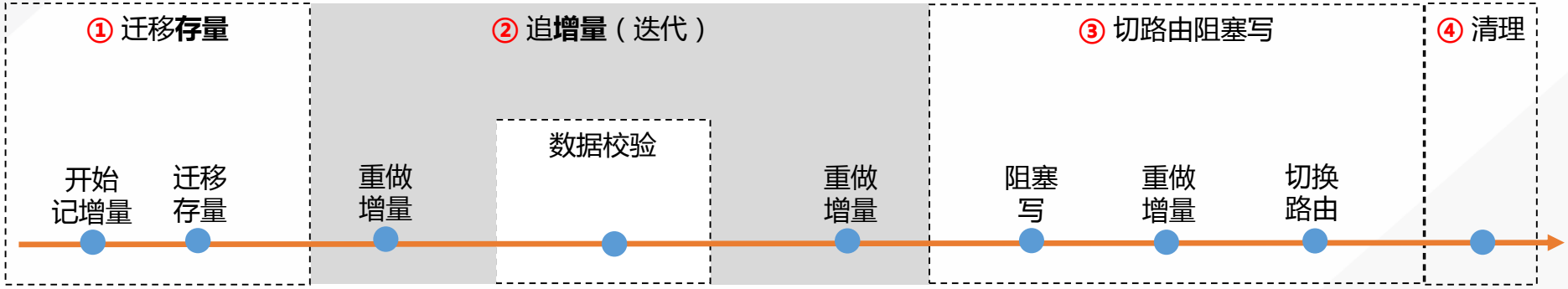
     （用户ID为某一个值的所有数据跨组迁移）

  3. 冷热数据迁移

     Hot Group -> Cold Group（时间纬度）

4.2 PGXZ 在线迁移一致性

在线迁移流程保证数据一致性



三个关键问题：

1. **存量/增量临界点：**
   - 怎么保证存量和增量之间 无重复 无遗漏： 数据库快照
2. **数据校验：**
   - 源/目标节点 如何保证完全一样的校验对象：数据库快照
   - 支持条数校验和内容校验
3. **切路由阻塞写：**
   - 阻塞时间在20ms左右，满足大部分业务

4.3 PGXZ 在线迁移效率

并行追增量提升迁移效率

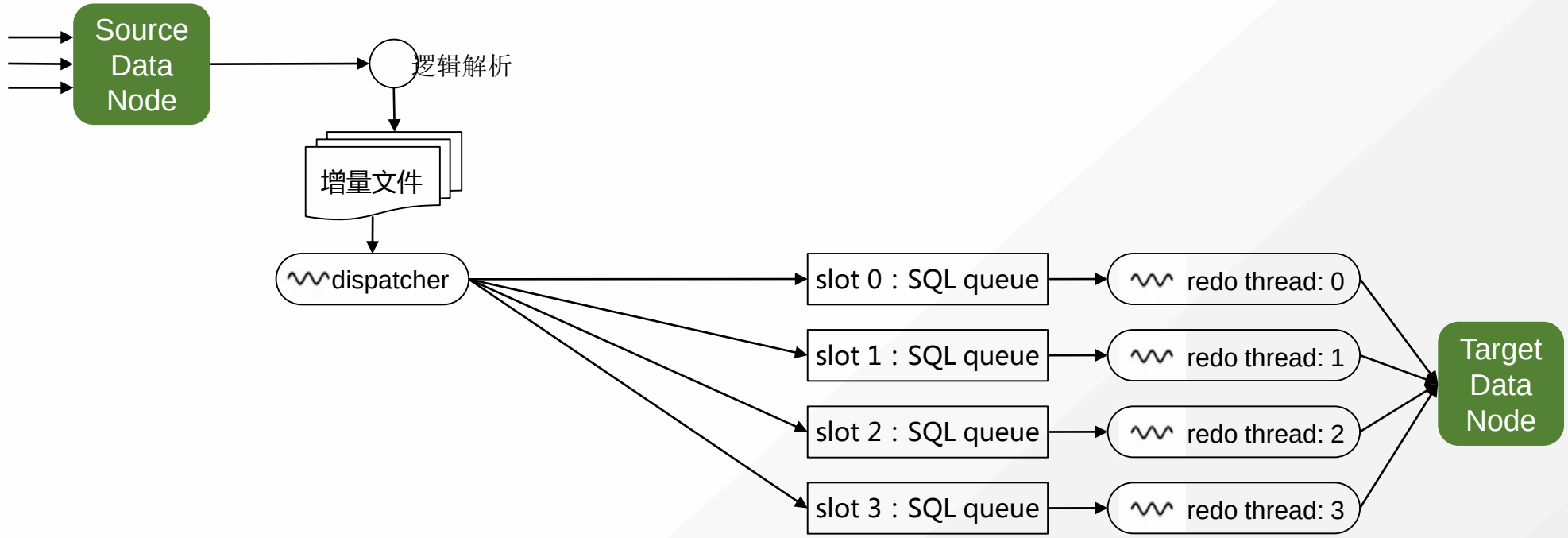
问题：追增量太慢(串行)

- 微信支付PGXZ出现过追增量速度比产生增量速度还慢
- 恶化磁盘空间紧张的局面（100G增量文件）

效果：微信支付现网（8个并发）

- **速度提升6X**（17min/G -> 2.8min/G）
- **追增量时间段缩短~10X**

方案：并行追增量如何保证顺序



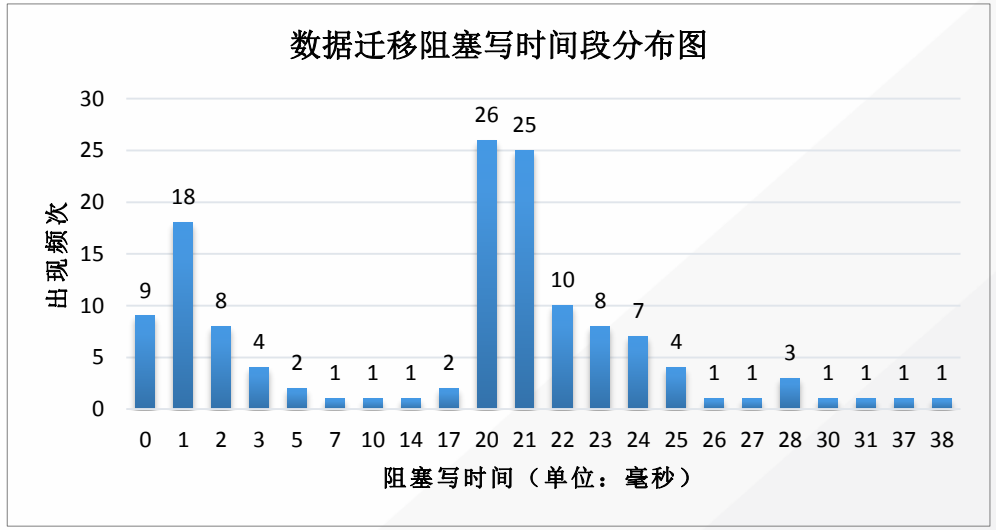
4.4 PGXZ 在线迁移效果

PGXZ扩容路径（微信支付商户系统）

- 2015.12 灰度上线
- 2016.02 正式上线：42组Data Node
- 2016.04 扩容：+6组热Data Node
- 2016.07 扩容：+6组热Data Node
- 2016.08 扩容：+12组冷Data Node
- 2016.12 扩容：+5组热Data Node
- 2017.02 扩容：+6组冷Data Node

当前共77组Data Node

数据迁移对微信支付商户系统的影响分析



阻塞写时间主要分布在20ms~25ms内

结论：

一共执行了135个Shard Moving Task

平均阻塞时间：15.6ms

分析：

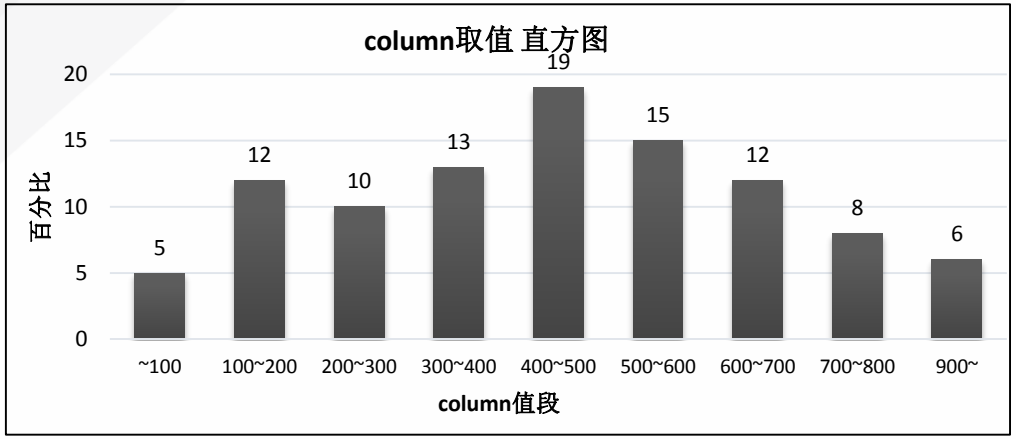
小于10ms：上线初期迁移shard的选择策略不同

大于30ms：可以通过调整迁移参数来缩短。

5.1 PGXZ Table Partition

分区表的代价模型

（父表上的结果集规模如何估计）



- **问题：**
  1. **计划不优**：子表直方图无法合并主表直方图
  2. **优化太慢**：通过子表直接估算父表结果集太慢

- **解决方案：**
  1. **同时抽样：**同时抽样统计子表和父表的直方图信息
  2. **分开处理：**

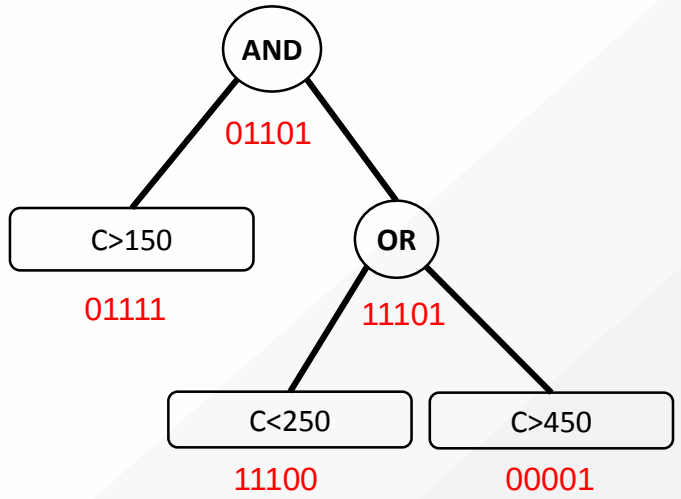
     落到一个子表上：直接用子表上的统计信息来估算

     落到多个子表上：使用父表上的统计信息来估算

根据查询条件对子表剪枝

分区表：p_table(100,200,300,400,500)

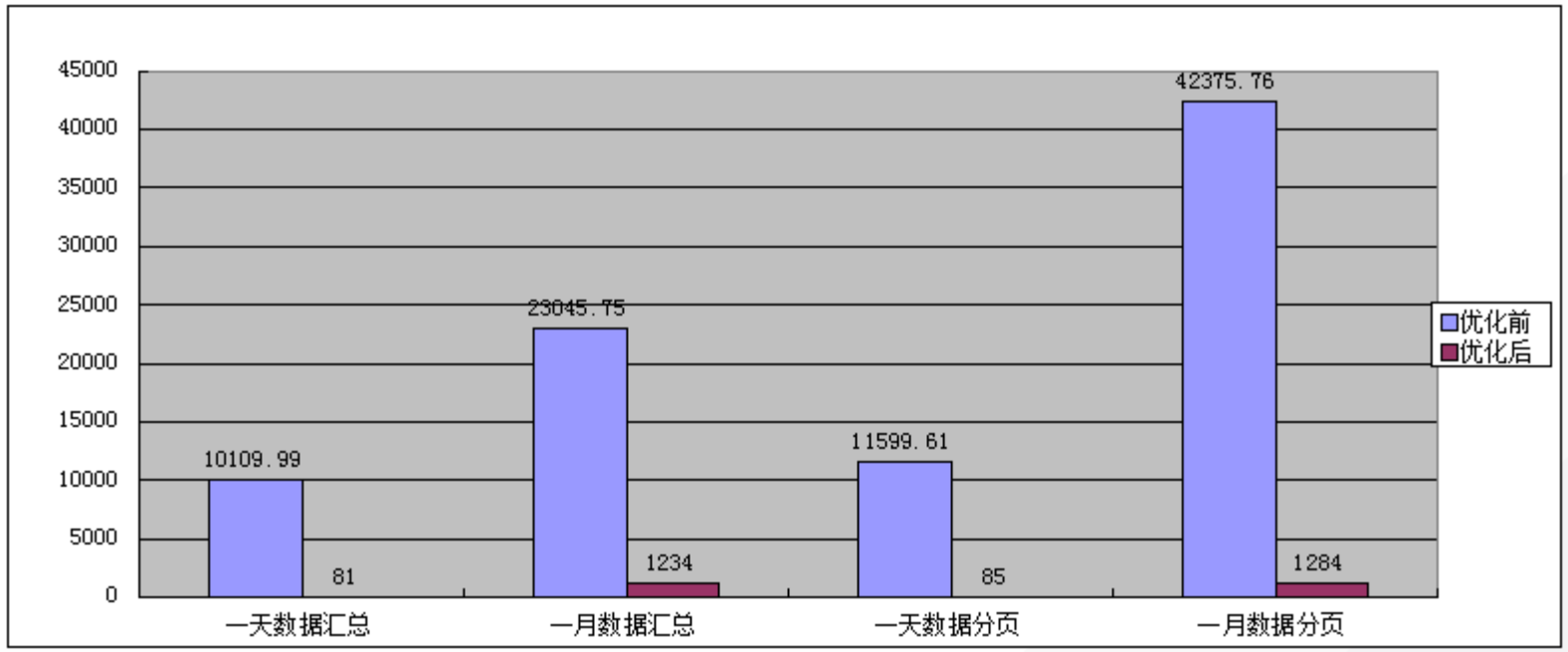
WHERE **c>150 and (c<250 or c>450)**



- **难点：**
  1. 准确性：剪多了，数据不对；剪少了，影响性能
  2. 通用性：查询表达式变幻无穷，如何能尽可能处理更多的场景
  3. 效率：快速剪枝，直接影响时延
- **方案：**
  1. 算法：树形递归+bitmap
  2. 按天分区：固化一年366天，空置2月29日换取性能

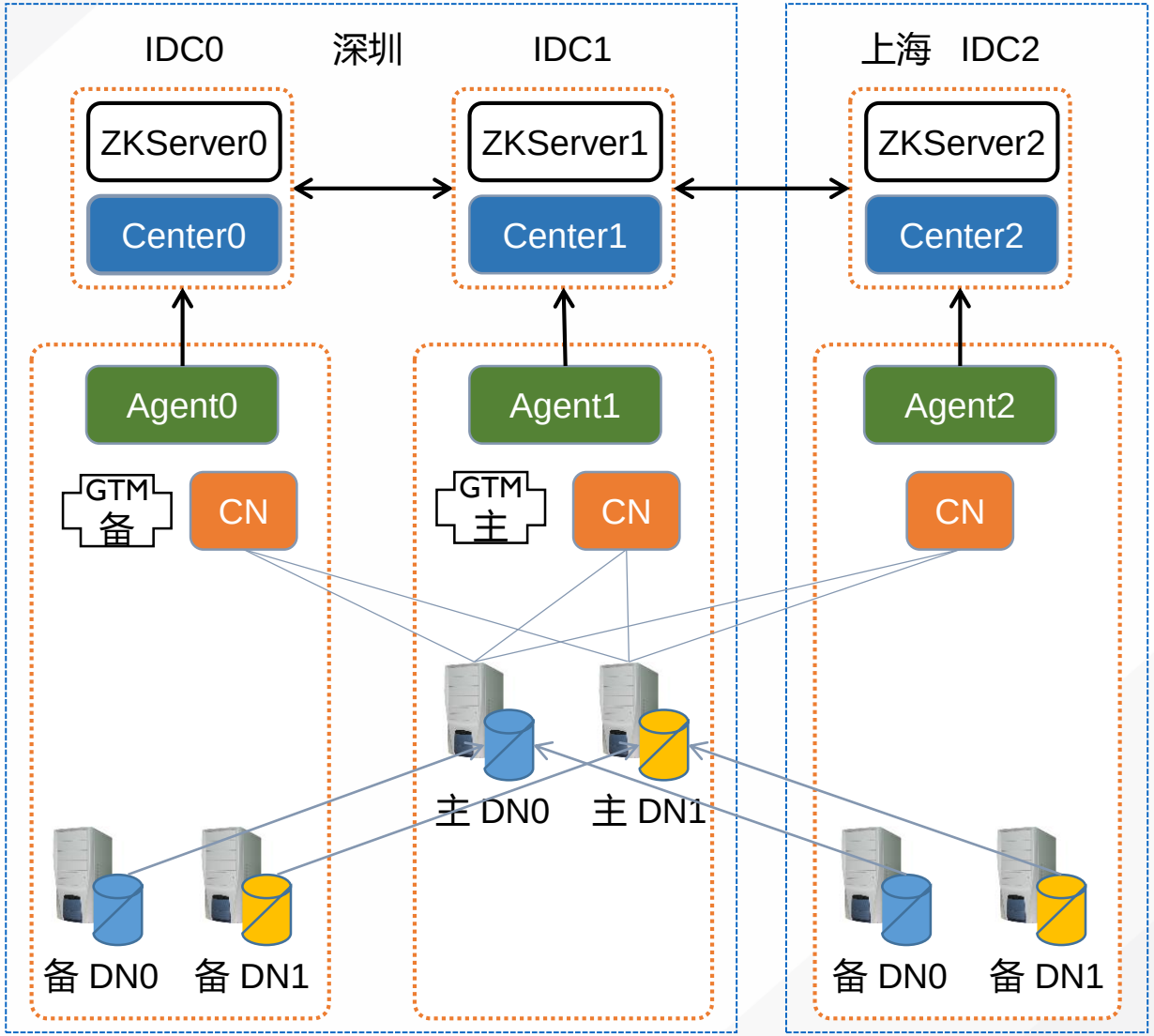
5.2 PGXZ Table Partition—效果

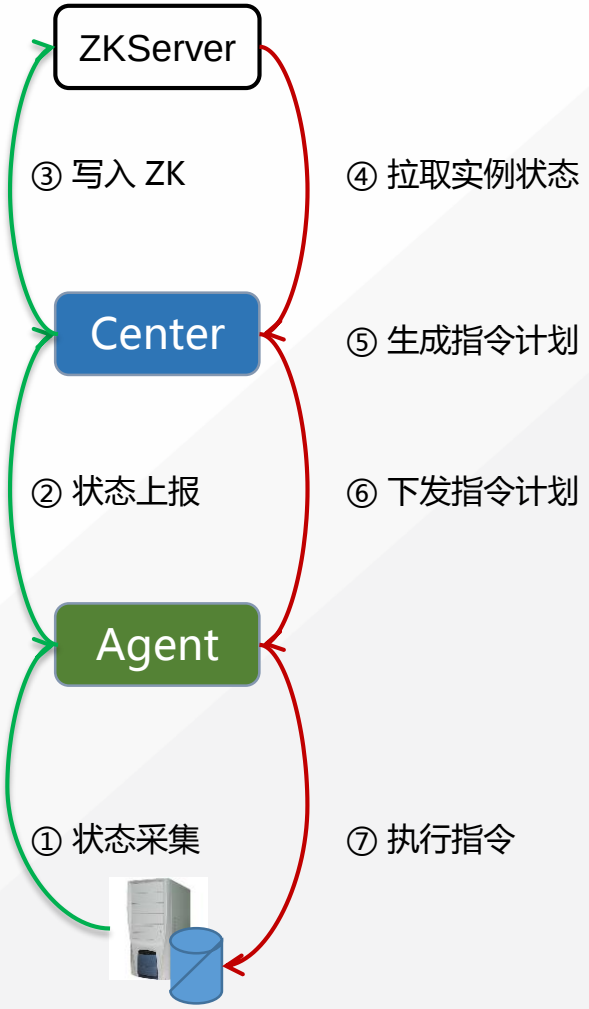
测试环境： PGXZ微信支付商户系统测试集群（Z3为主），6组DN
数据量　： 600G数据量
测试目的： 分区表对汇总类业务的性能提升



操作	优化前	优化后	sql语句
一天数据汇总	10109.991	81	select count(ftotal_amount) ,sum(ftotal_amount) from t_trade_ref where ftotal_amount between 1 and 683771 and fcreate_time >= '2015-07-31 00:00:00' AND fcreate_time < '2015-08-01 00:00:00' and fmerchant_id = 7777777;
一月数据汇总	23045.746	1234	select count(ftotal_amount) ,sum(ftotal_amount) from t_trade_ref where ftotal_amount between 1 and 683771 and fcreate_time >= '2015-07-31 00:00:00' AND fcreate_time < '2015-08-30 00:00:00' and fmerchant_id = 7777777;
一天数据分页	11599.605	85	select * from t_trade_ref where (ftotal_amount between 1 and 683771 and fcreate_time >= '2015-07-31 00:00:00' AND fcreate_time < '2015-08-01 00:00:00' and fmerchant_id = 7777777) ORDER BY fcreate_time DESC LIMIT 10 offset 0;
一月数据分页	42375.755	1284	select * from t_trade_ref where (ftotal_amount between 1 and 683771 and fcreate_time >= '2015-07-31 00:00:00' AND fcreate_time < '2015-08-31 00:00:00' and fmerchant_id = 7777777) ORDER BY fcreate_time DESC LIMIT 10 offset 0;

6.1 PGXZ容灾


IDC0
深圳
IDC1
上海
IDC2
ZKServer0
ZKServer1
ZKServer2
Center0
Center1
Center2
Agent0
Agent1
Agent2
GTM
备
CN
GTM
主
CN
CN
主 DN0
主 DN1
备 DN0
备 DN1
备 DN0
备 DN1


ZKServer
③ 写入 ZK
④ 拉取实例状态
Center
⑤ 生成指令计划
② 状态上报
⑥ 下发指令计划
Agent
① 状态采集
⑦ 执行指令

7. 其他部分特性

1. 滚动升级
2. 自动化运维/监控
3. Coordinator Pooler优化
4. 分布式事务优化

Q&A