DTCC

2017第八届中国数据库技术大会

DATABASE TECHNOLOGY CONFERENCE CHINA 2017

Postgres-XZ数据治理经验分享

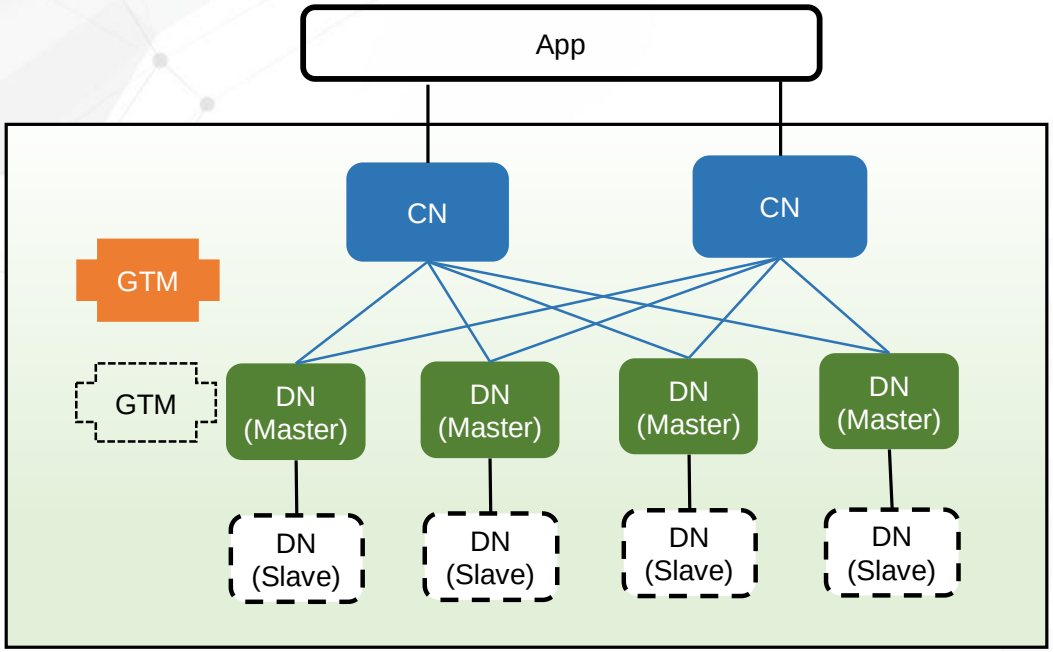
许中清@腾讯

目录

PGXZ—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**

PGXZ 主要特性

1. 分布式事务
2. 跨节点的JOIN
3. 自动在线扩容
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

PGXZ数据治理背景---问题

PGXC数据治理不能满足生产环境的需求

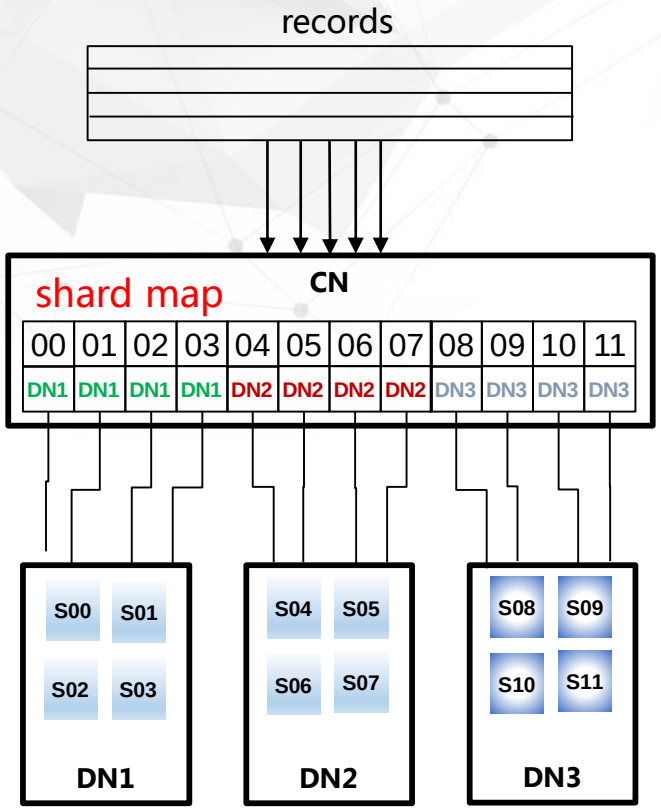
PGXC的问题	导致的后果	解决方案
Sharding ROW直接HASH到DN	❑无法扩容：节点数必须重新HASH所有数据 ❑无法均衡：没办法应对数据倾斜	**Data Sharding** 解决扩容和容量均衡问题
Partition 内核不支持分区表	❑业务逻辑处理分区表：业务逻辑与DB耦合 ❑继承表性能无法接受： • 查询：100个子表时延近1S • 写入：trigger极大影响性能	**Table Partition** 解决易用性问题和性能问题

目录

PGXZ Data Sharding—方案关键点

PGXZ Sharding方案

核心思想：类似一致HASH算法



CN增加shardmap
动态管理路由

DN维护本地shard
过滤不可见记录

每一行物理上
维护自身shardid

关键点

关键点：

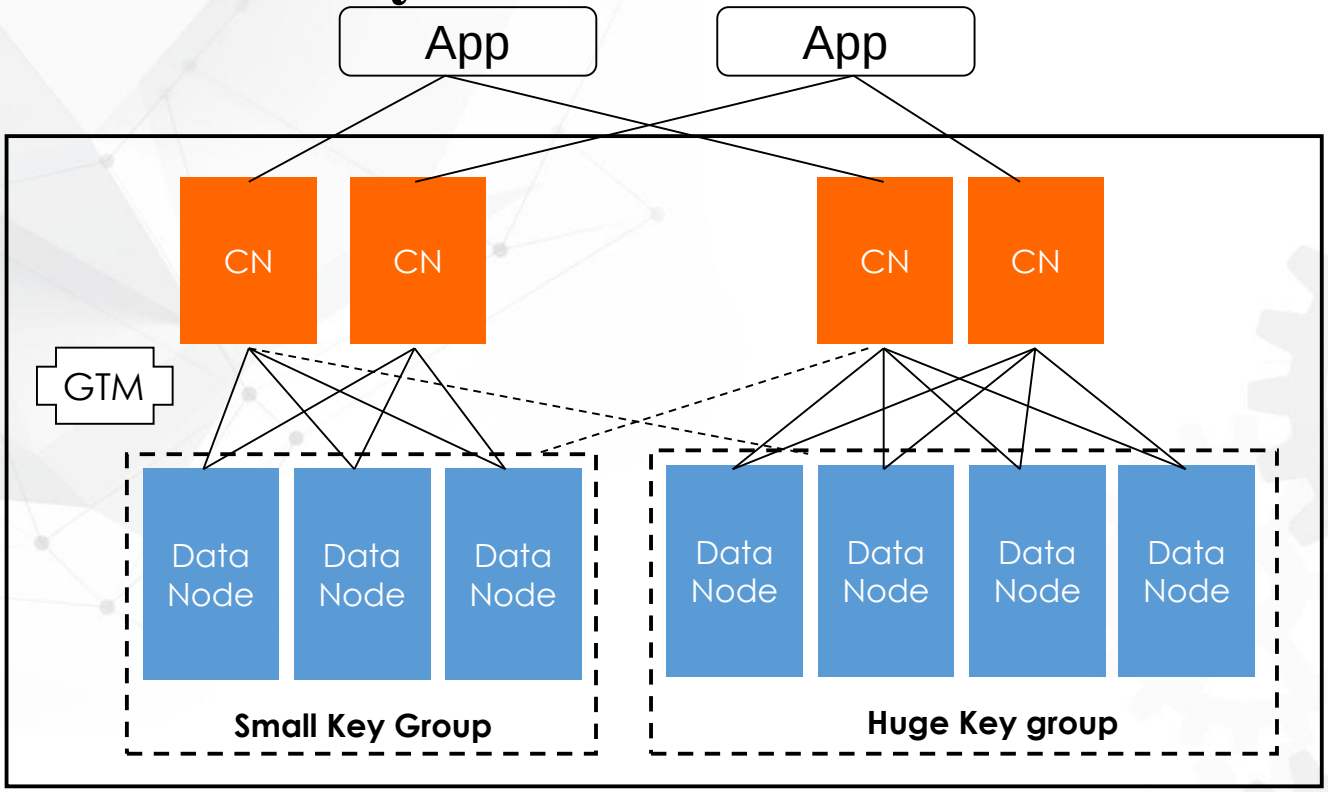
1. **DN过滤**：过滤临时副本

   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

多组: 治理ShardKey倾斜



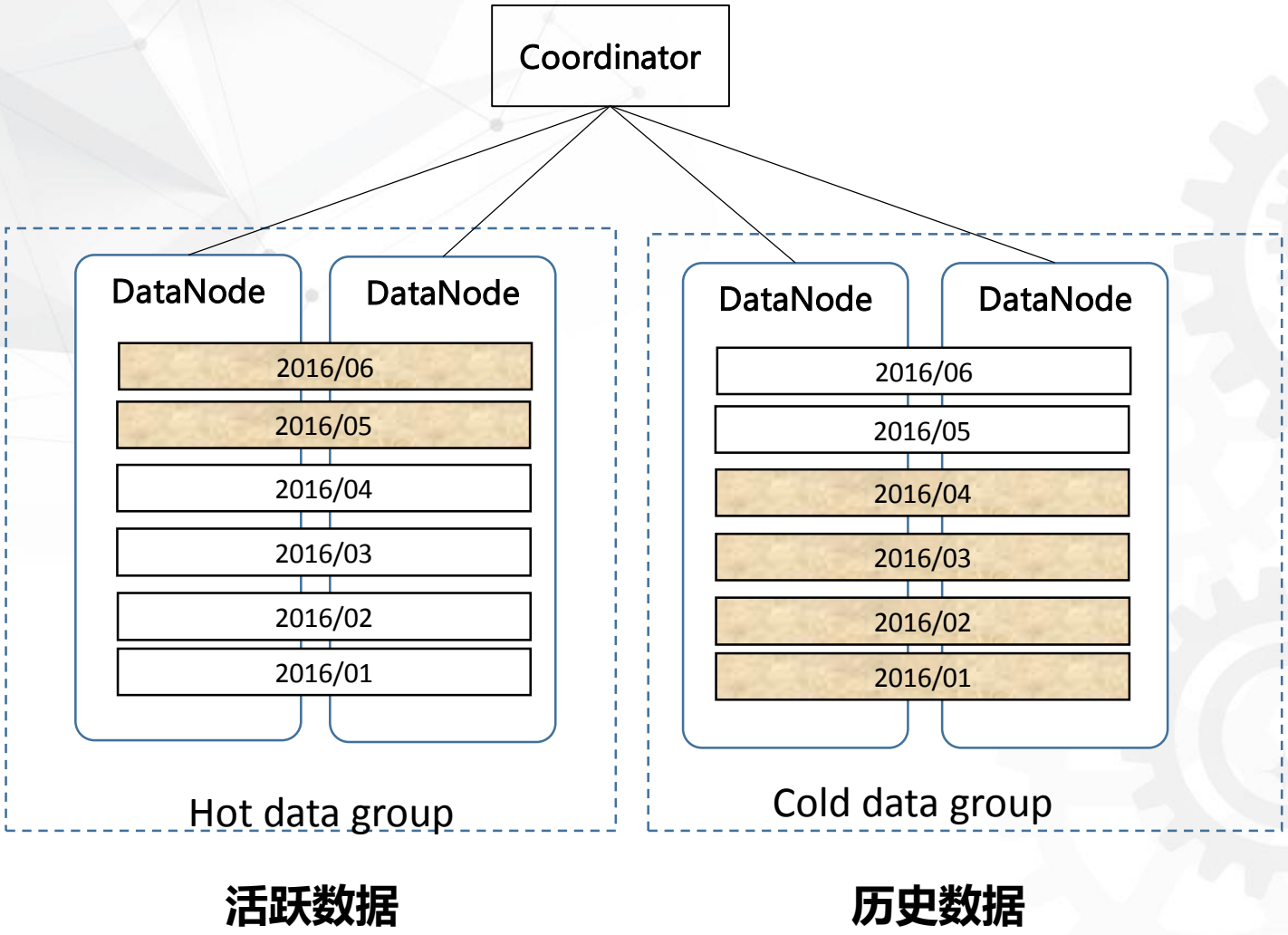
小商户组: 一个商户的数据只落在一个DataNode上.

- 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.

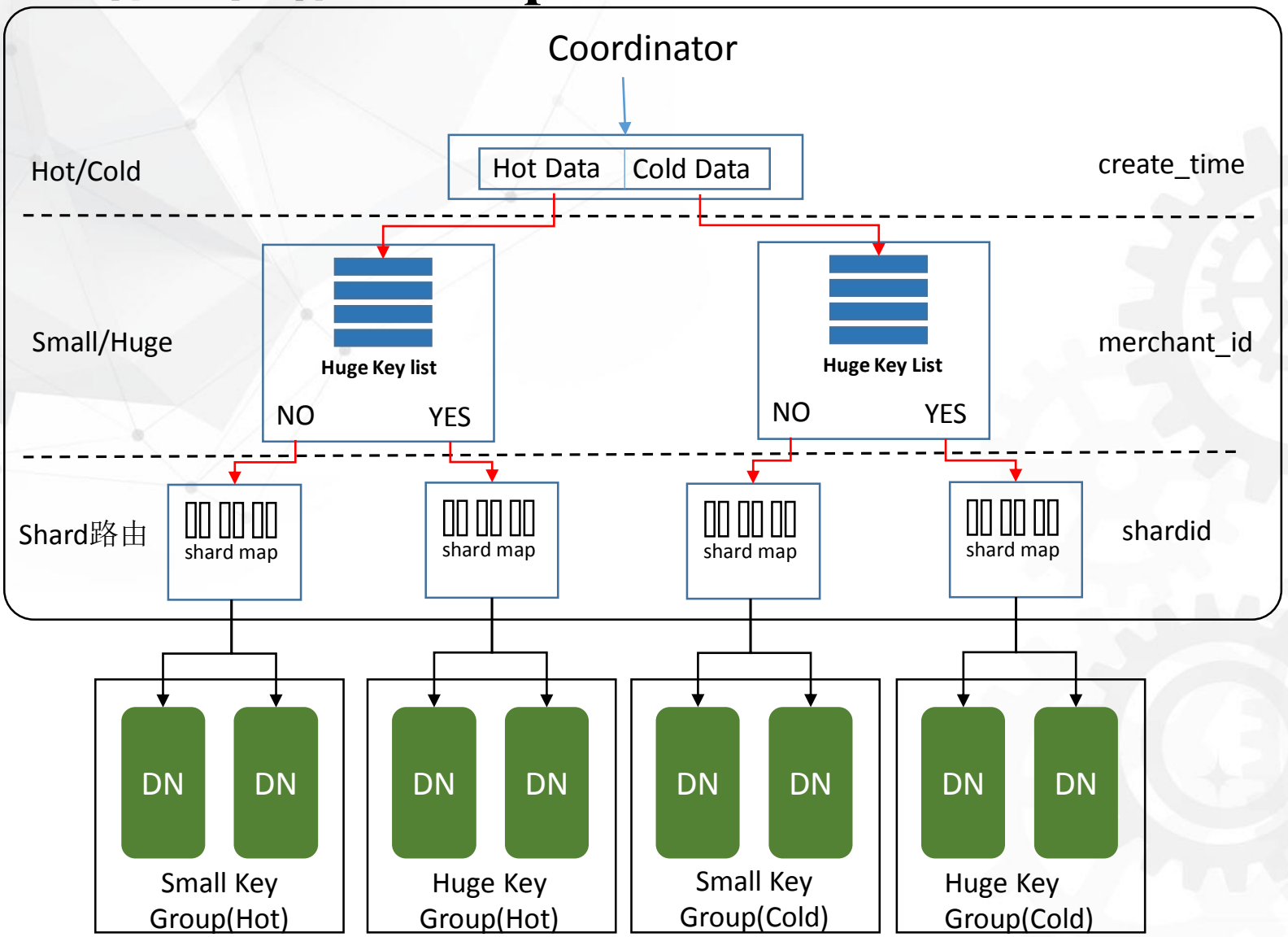
多组: 冷热数据分治



活跃数据

历史数据

多组：路由策略(4 Groups)



目录

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

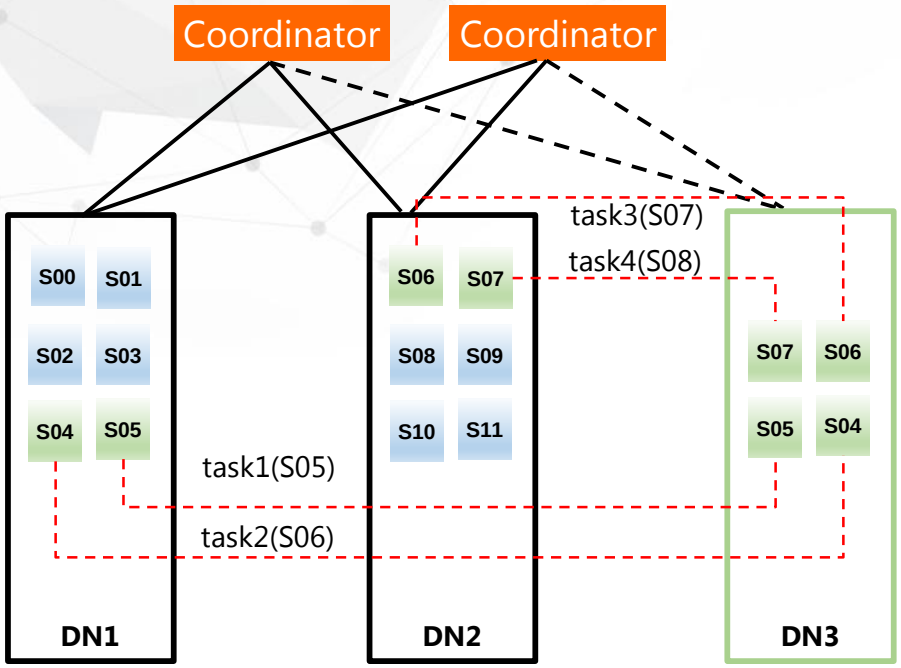
     （用户ID为某一个值的所有数据跨组迁移）

  3. 冷热数据迁移

     Hot Group -> Cold Group（时间纬度）

PGXZ 在线迁移的问题

扩容/DN均衡 的关键在于数据迁移



数据迁移关键问题：

1. **数据一致性**

   如何校验数据的正确性

1. **业务无感知**

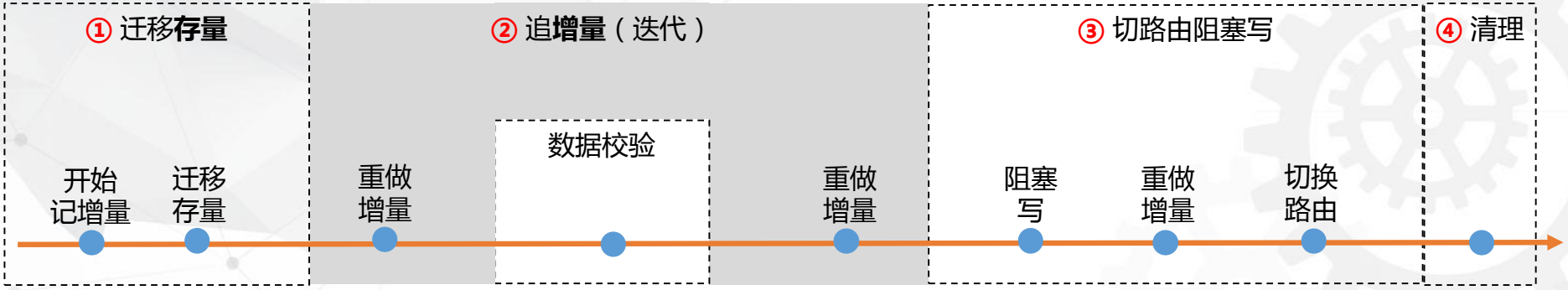
   不影响业务读写

2. **效率要高**

   增量追不上：并行写入/串行增量

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

PGXZ 在线迁移效率

并行追增量提升迁移效率

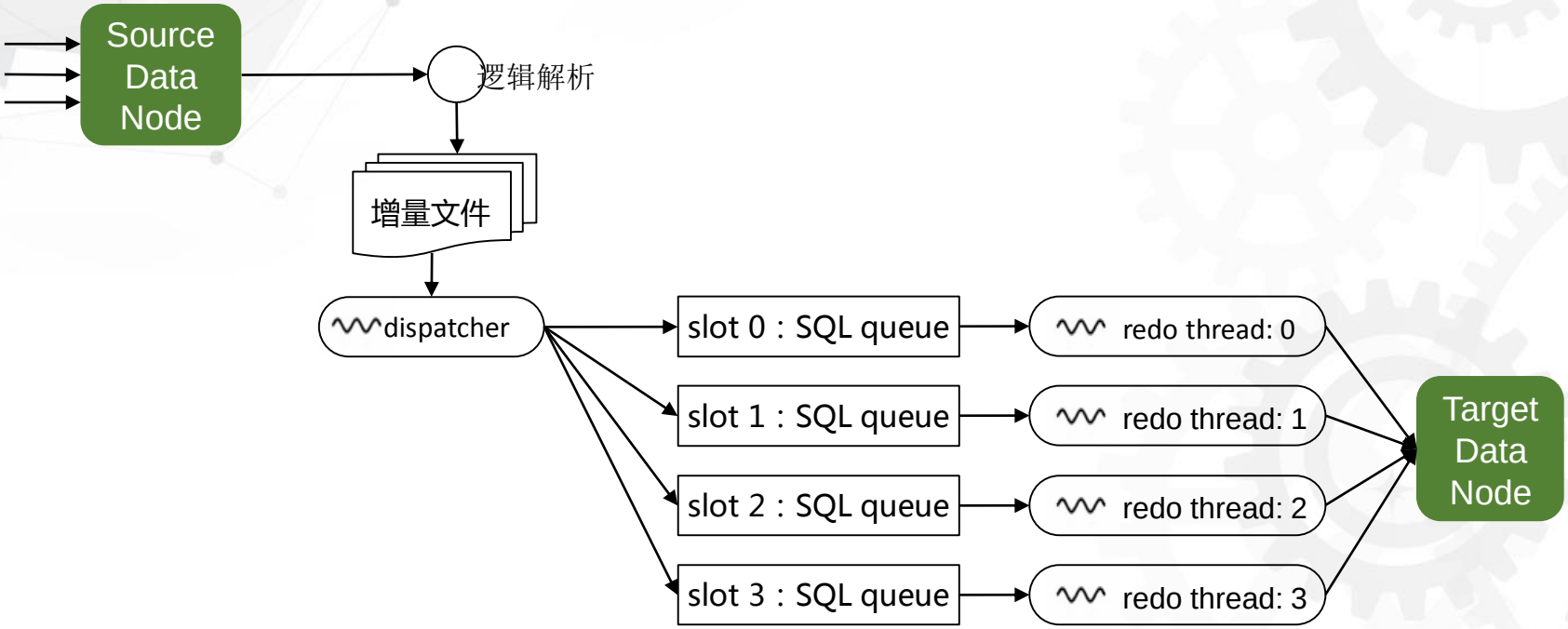
问题：追增量太慢(串行)

- 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）

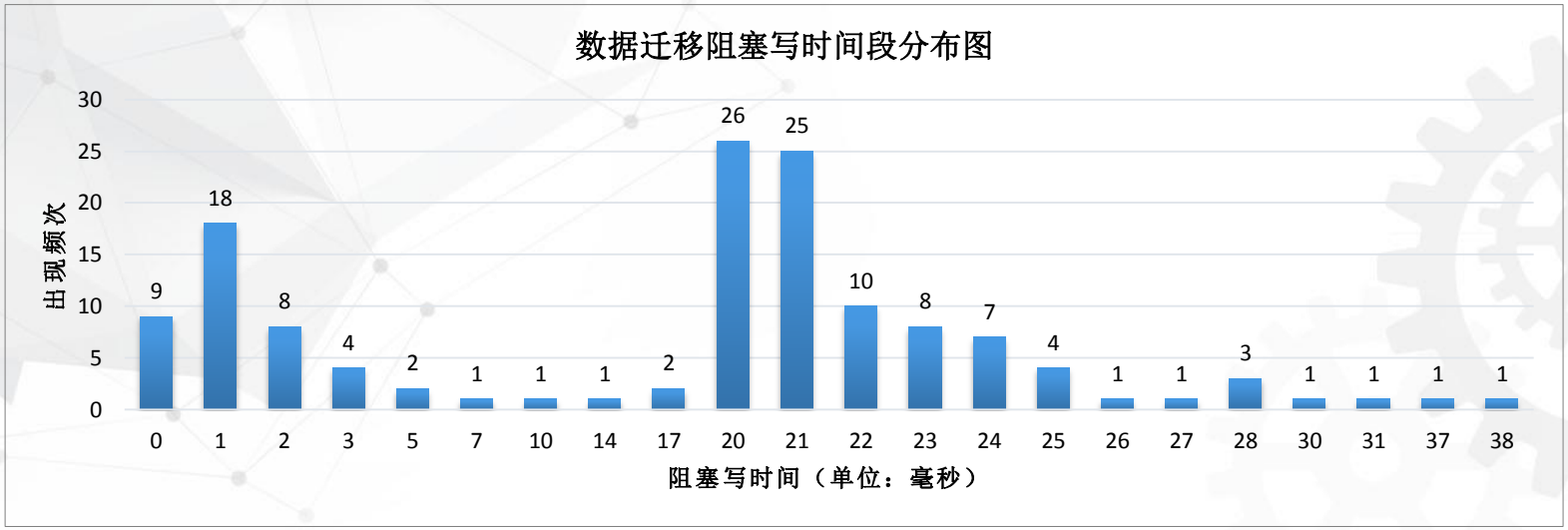
- **速度提升6X** （17min/G -> 2.8min/G ）
- **追增量时间段缩短~10X**

方案：并行追增量如何保证顺序



PGXZ 在线迁移效果

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。

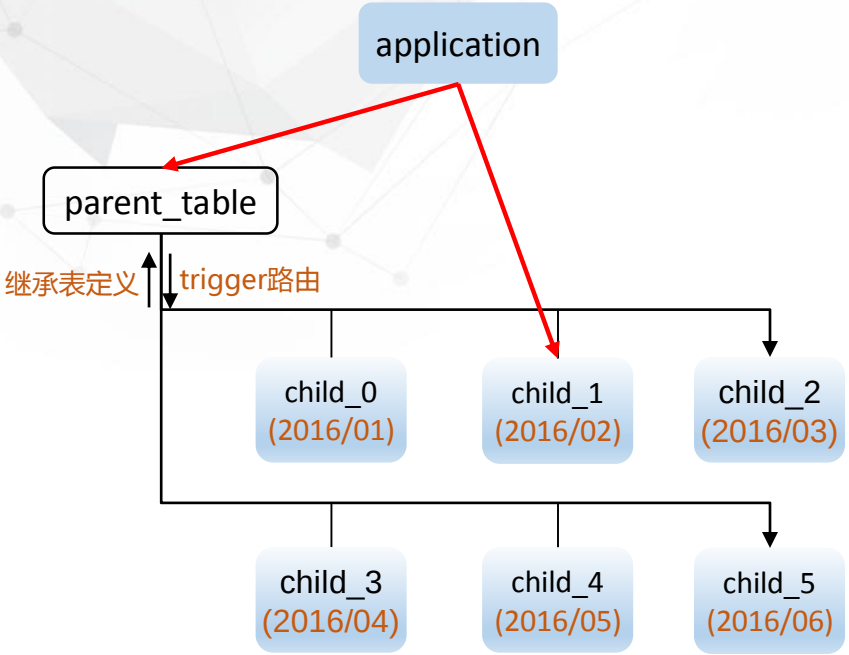
目录

PGXZ Table Partition—问题

PostgreSQL9.3开源现状

Trigger实现路由

遍历子表来剪枝



问题

管理不易：创建维护复杂

子表/父表、约束、路由规则、索引

性能慢：

trigger影响写性能

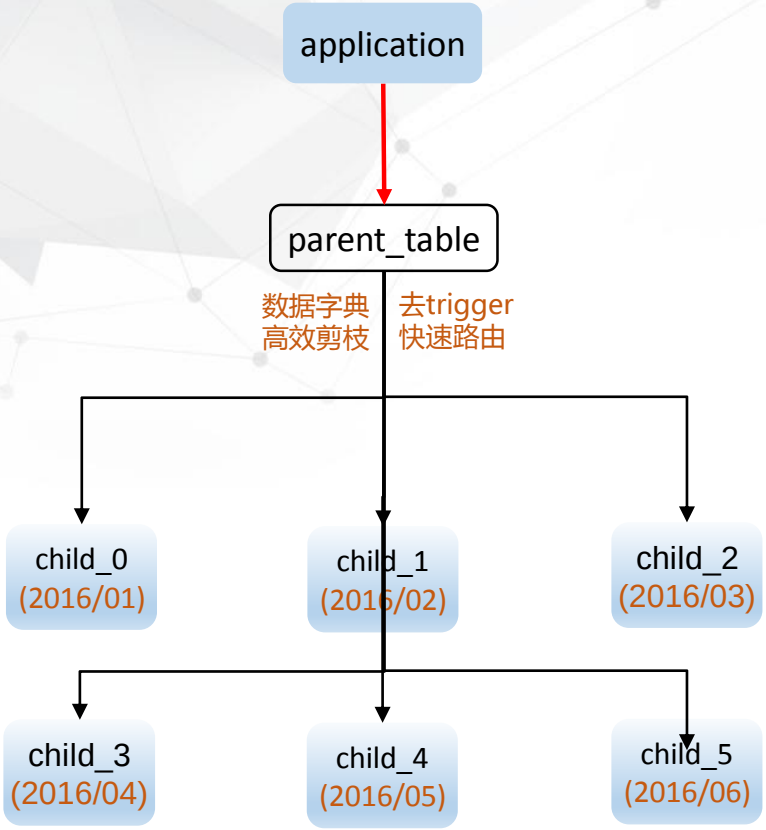
查询优化影响读性能（500子表时延1s）

表模式不安全：

用户可以直接操作子表

PGXZ Table Partition—方案

实现内置的分区表特性



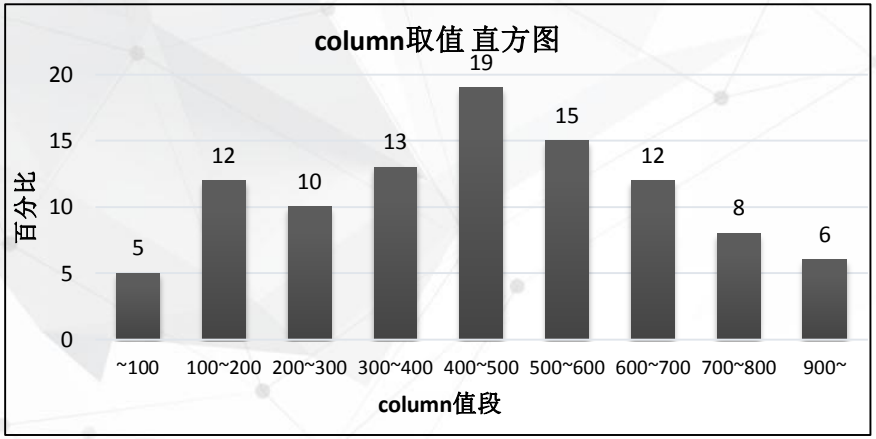
策略

- **提升易用性：**
  1. **维护简单：**DDL一条SQL即可
  2. **表模式安全：**用户不能直接操作子表
- **提升性能**
  1. **快速路由：**抛弃trigger
  2. **高效剪枝：**直接定位目标子表
  3. **内核优化：**代价模型/优化算法
- **提升可用性**
  1. **迁移冷数据**
  2. **冷热分治**

PGXZ Table Partition—关键难点

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：**

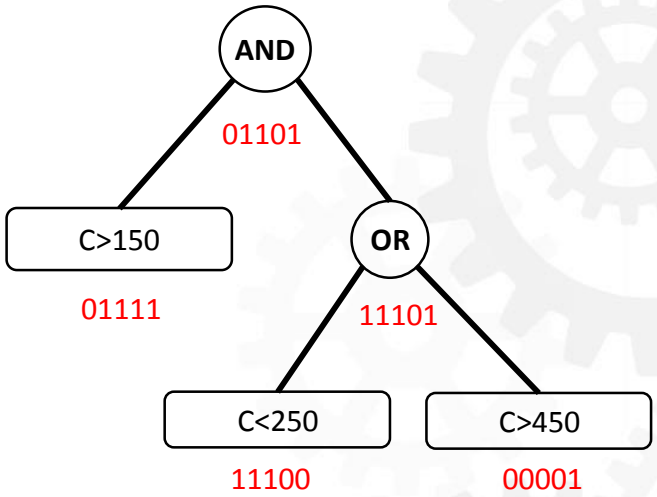
     落到一个子表上：直接用子表上的统计信息来估算

     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)

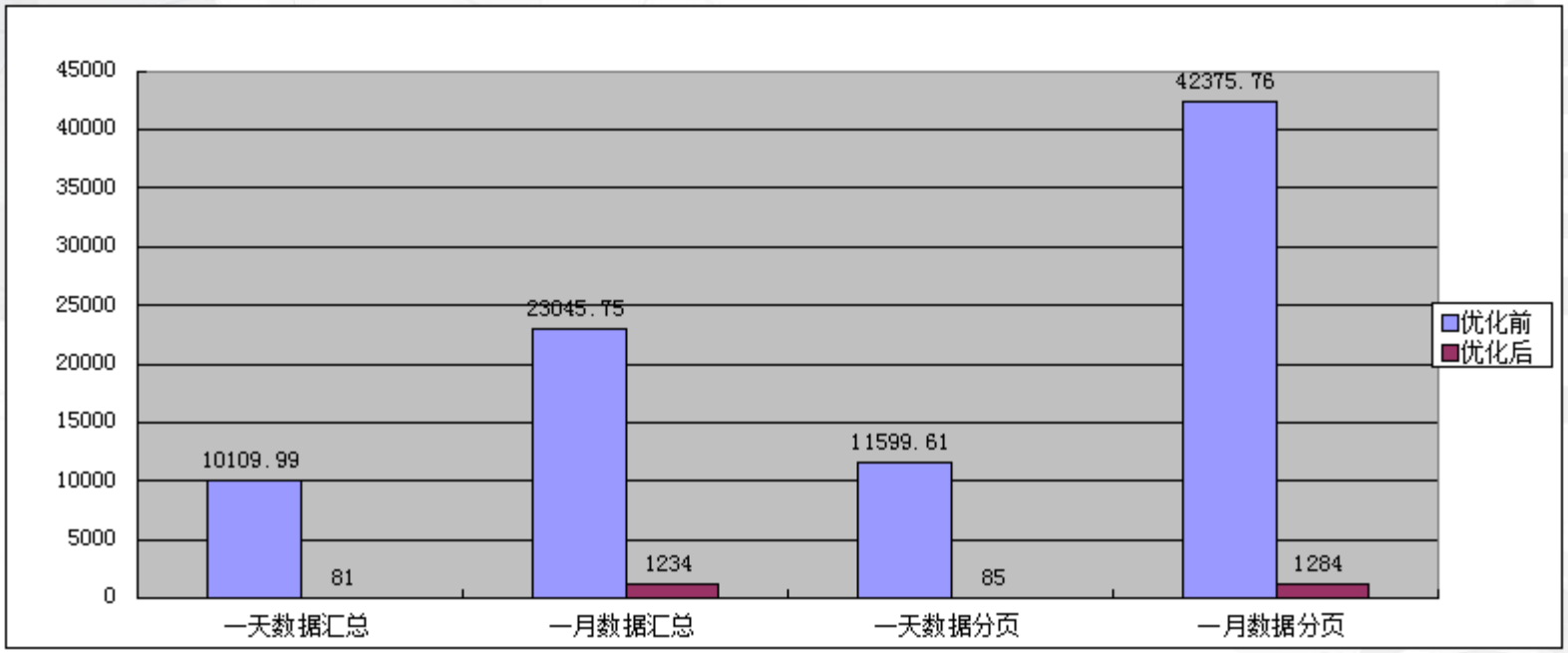
WHERE **c>150 and (c<250 or c>450)**



- **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

PGXZ Table Partition—效果(高性能)

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



THANKS

SequeMedia 盛拓传媒 IT168.com ITPUB ChinaUnix