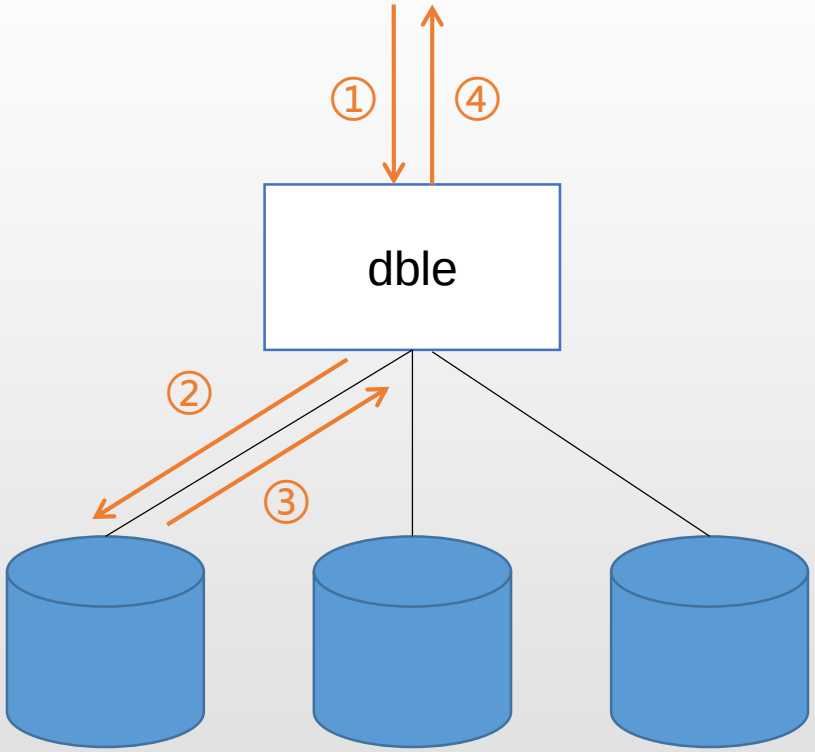
如何愉快地与dble玩耍

2019.06.15 钟悦

话题

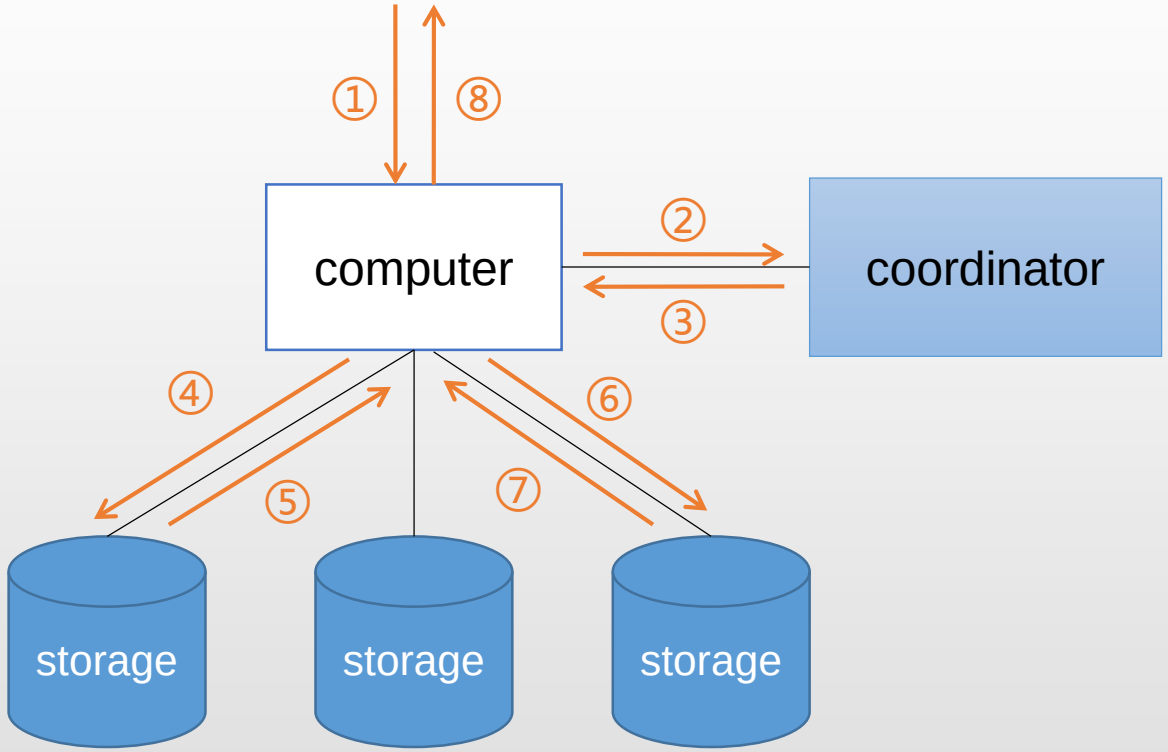
- dble和mycat-like架构的优势和极限
- 实施上的最佳实践

dble和mycat-like架构的极限



- 不知道数据的真实分布
  - 所有非sharding key访问都需要广播
  - 难以建立二级索引
  - 优化器优化空间有限
- 前端会话与后端连接强耦合
  - 前端并行执行能力受限于后端单个MySQL的最大连接数
  - 跨片访问会有“连接数放大”现象

然后，有了NewSQL架构



- 计算层能获取数据真实分布

单条语句耗时长*

- 前端会话与后端连接解耦合

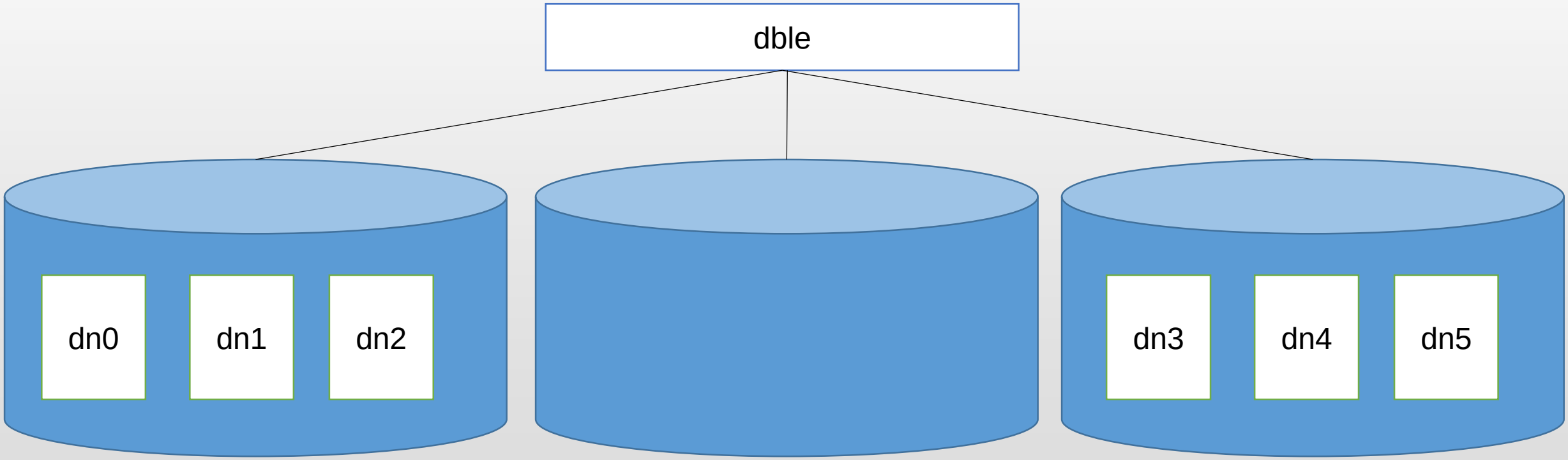
相反就是dble架构的相对优势

* 简单语句且无法命中内部row_id的情况（一般的客户语句都无法满足）

实施上的通用经验

超量分片策略

按照最大数据规模提前分好逻辑分片（dataNode）数量，对后端MySQL扩缩容时，仅需要增加物理分片（dataHost）数量，并迁移部分逻辑分片即可。从而保证方案具有横向扩容能力。



状态表访问场景 表存储的是对象的某个时刻的状态，例如订单表、客户信息表等

- 场景特点

数据量大（TB级），**单条语句耗时要求高（平均低于4ms）**

- 实践经验

◆ 语句必须利用上sharding key
◆ 语句必须利用上MySQL内的索引
◆ 禁止广播访问
◆ 限制跨分片访问（跨分片数在4个以内）
◆ 善用全局表来改善JOIN
◆ 妥协：由应用层使用分布式事务框架进行跨库事务的保障

日志访问场景

表存储的是对象的变动过程，例如系统活动日志、用户操作历史表等

- 场景特点

数据量大（TB级），一直在写入，**联机写入时IO压力大**，**批量清理时希望减少对联机写入的干扰**

- 实践经验

◆ dble层面上按**时间**进行分片，分散写入压力到不同的物理节点上

◆ MySQL层面上按**日期**进行分区，为使用TRUNCATE PARTITION和DROP PARITION的快速清理手段提供支持

◆ 妥协：日志表内要同时有日期和时间两部分信息

谢谢静听