腾讯云数据库CDB技术演进之路

目录

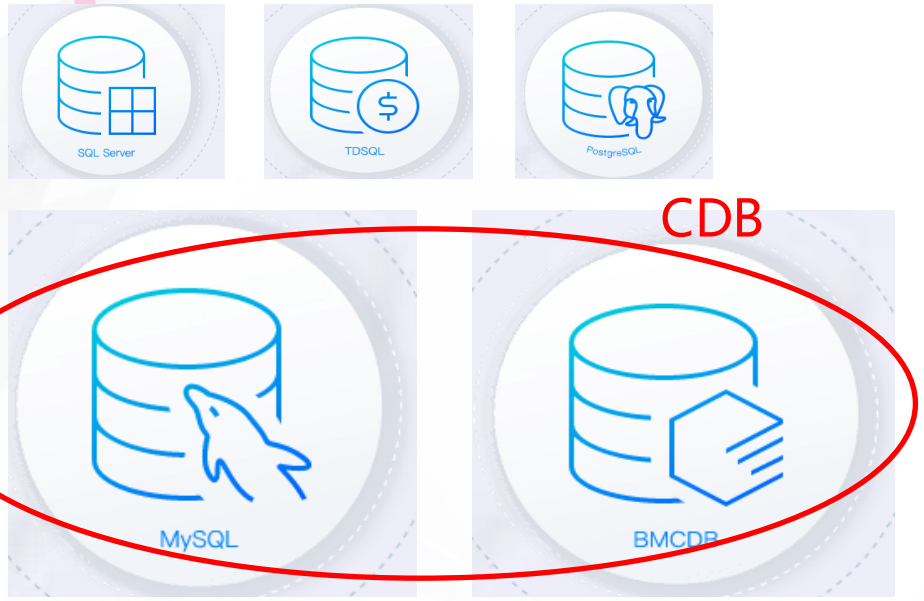
云数据库概览-什么是云数据库

NIST关于云计算服务的基本特征定义

- ✓ 随需应变自助服务
- ✓ 随时随地用任何网络设备访问
- ✓ 多人共享资源池
- ✓ 快速重新部署灵活度
- ✓ 可被监控与量测的服务

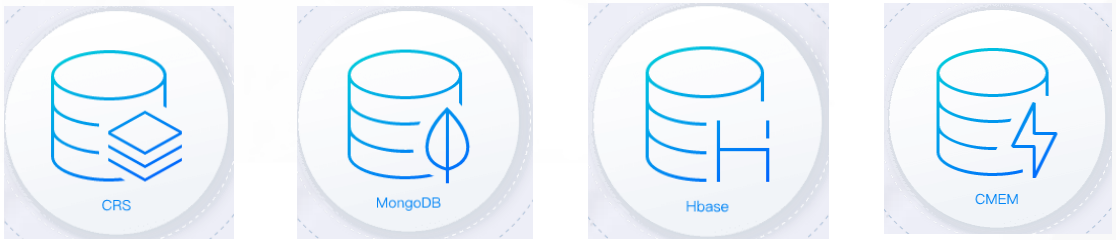
- 满足云计算特征的数据库服务
- 数据库内核+云化功能

云数据库概览-鹅厂的云数据库

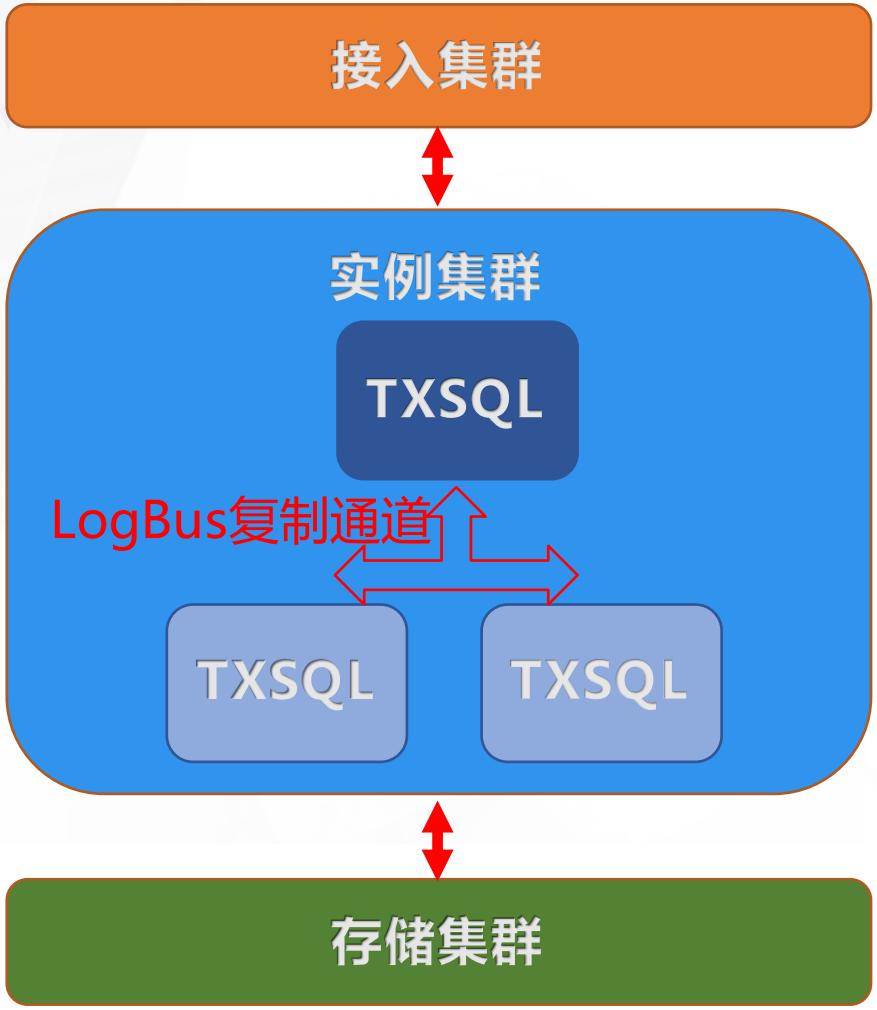


- SQL

- NoSQL



云数据库概览-CDB内核技术栈



- ✓ 路由
- ✓ 安全
- ✓ HA

- ✓ 数据库引擎
- ✓ 多副本复制

- ✓ 数据存储

CDB之存储篇-三次存储革命

阶段1　阶段2　阶段3　阶段4

- 根据存储介质特性，进行数据库存储技术设计
- 数据库存储的本质：面向块的存储

CDB之存储篇-第一次革命

- SAS Raid存储的主要矛盾
  - ✓ 性能问题：随机IO性能不高
  - ✓ 扩展性问题：单机最大存储容量受限
  - ✓ 成本问题：价格贝佐斯定律带给成本的挑战
- SSD Raid存储可行性
  - ✓ 选择SATA SSD：2011年PCI-E SSD刚刚兴起
  - ✓ 性能问题：随机IO性能有较大提升
  - ✓ 扩展性问题：依旧存在
  - ✓ 成本问题：单位存储成本更高
  - ✓ SSD使用的问题：写放大、写性能下降
- 解决问题的思路
  - ✓ 基于SSD的存储系统有吗？ 分布式KV
  - ✓ 分布式KV解决了性能、扩展性、SSD使用问题
  - ✓ 分布式KV和数据库存储的联系：block
  - ✓ 块block的数据模型(Disk, LBA, Value)<->(Key, Value)

基于SAS阵列的数据库存储

CDB之存储篇-第一次革命(Cont.1)

- SSD Cluster1.0存储
  - ✓ 虚拟块设备+IO网络化→块读写转化为KV操作
  - ✓ 适配SSD特性的KV存储系统
  - ✓ MySQL层优化：千兆网络瓶颈 & 去double write
  - ✓ 文件系统&块设备调优
  - ✓ 成本问题：按需分配、动态伸缩、
- 运营中的故事
  - ✓ 存储集群的蝴蝶效应

基于SSD Cluster1.0的数据库存储

CDB之存储篇-第二次革命

- SSD Cluster1.0的主要矛盾
  - ✓ 开发商的变化：IO密集型核心应用上云
  - ✓ 存储介质的变化：PCI-E SSD量大价低
  - ✓ 性能不够，性价比没优势
- PCI-E SSD Cluster存储可行性
  - ✓ IO性能：相比本地，网络和分布式带来额外开销
  - ✓ 扩展性：单机最大本地存储6T
  - ✓ SSD使用的问题：SSD FTL的进一步优化
  - ✓ 成本问题：相比本地，优势有，但已不是主要矛盾
  - ✓ 运维成本：相比本地，更高
- 解决问题的思路
  - ✓ 选择本地PCI-E SSD
  - ✓ 选择新的技术制高点：数据库引擎本身的性能和稳定性

基于PCI-E SSD的数据库存储

CDB之存储篇-第三次革命

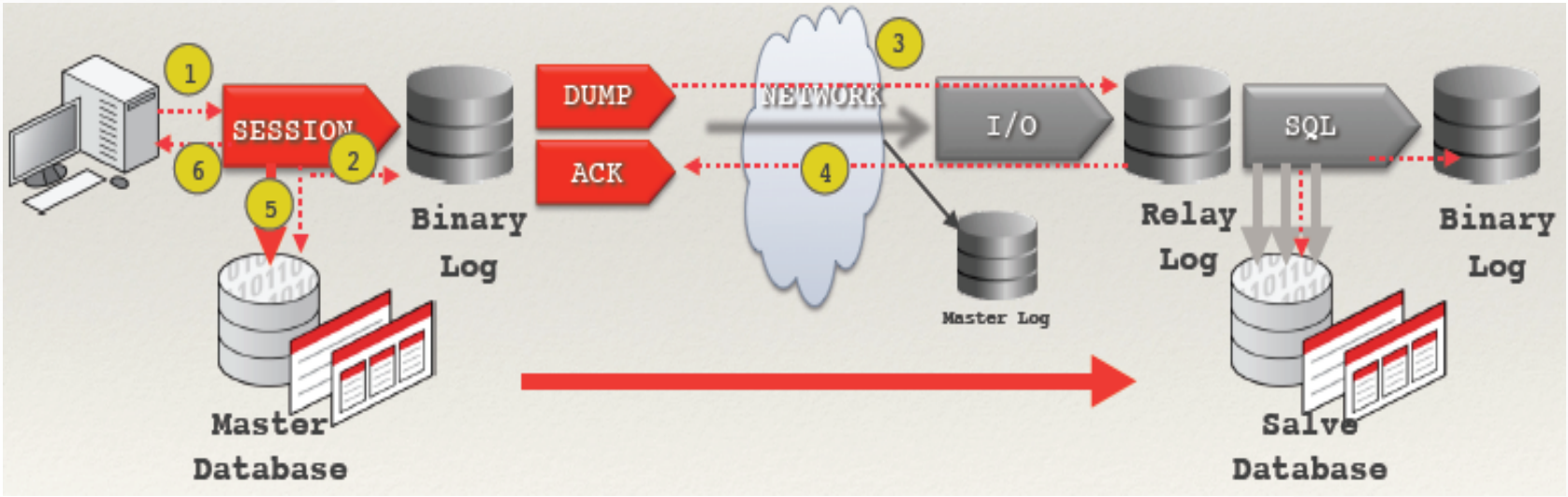
- PCI-E SSD的主要矛盾
  - ✓ 开发商的变化：金融、政企等数据库上云
  - ✓ DB要求：兼容性、RTO、RPO、扩展性四个都不能少
  - ✓ 扩展性问题：容量无法和SAN\NAS\专有存储相比
  - ✓ 中间件sharding解决了扩展性问题，但兼容性有问题
- 解决问题的思路
  - ✓ 主要目标：100TB以下，SQL完全兼容的传统行业DB服务
  - ✓ SDP：Shared Disk Parallel
  - ✓ 数据库节点和存储分离，数据库节点有主从之分
  - ✓ 尽量减少IO次数：主数据库节点才能写存储集群，从节点不会写

CDB之复制篇-三种复制模式

- 复制结合机房部署，灵活选择，达到最有效的容灾效果

CDB之复制篇-半同步的一些问题

- MySQL半同步的主要问题
  - ✓ 无法做到强一致：自动降级、灵活的多zone部署下的一致性要求
  - ✓ 半同步下性能下降多



MySQL Native Semi Sync Replication

CDB之复制篇-性能优化

- 性能关键因子
  - ✓ 单个事务耗时：用户层面的响应时间，每个事务耗时多少ms
  - ✓ 系统整体吞吐：服务层面的处理能力，一台机器每秒能处理事务数(TPS)
- 单个事务耗时
  - ✓ $T_{total}=T_{sql}+T_{engine}+T_{replicate}$
  - ✓ T_{sql}和T_{engine}在同步复制下不是关键瓶颈
  - ✓ $T_{replicate}=T_{binlog网络传输}+T_{slave落地binlog}$，$T_{binlog网络传输}$取决于RTT值
  - ✓ 测试数据：
    - ✓ 全cache下MySQL异步单事务耗时3.37ms；
    - ✓ 半同步单事务耗时8.33ms，测试环境RTT值2.6ms，$T_{slave落地binlog}$为1.9ms
    - ✓ 测试机器顺序512B的write+fsync延时0.13ms，$T_{slave落地binlog}$优化空间大
- 系统整体吞吐
  - ✓ 吞吐=并发数/单个事务耗时
  - ✓ 并发数取决于各种资源的利用率，如master的事务线程、master的binlog发送/响应线程、slave的binlog接收线程等利用率情况

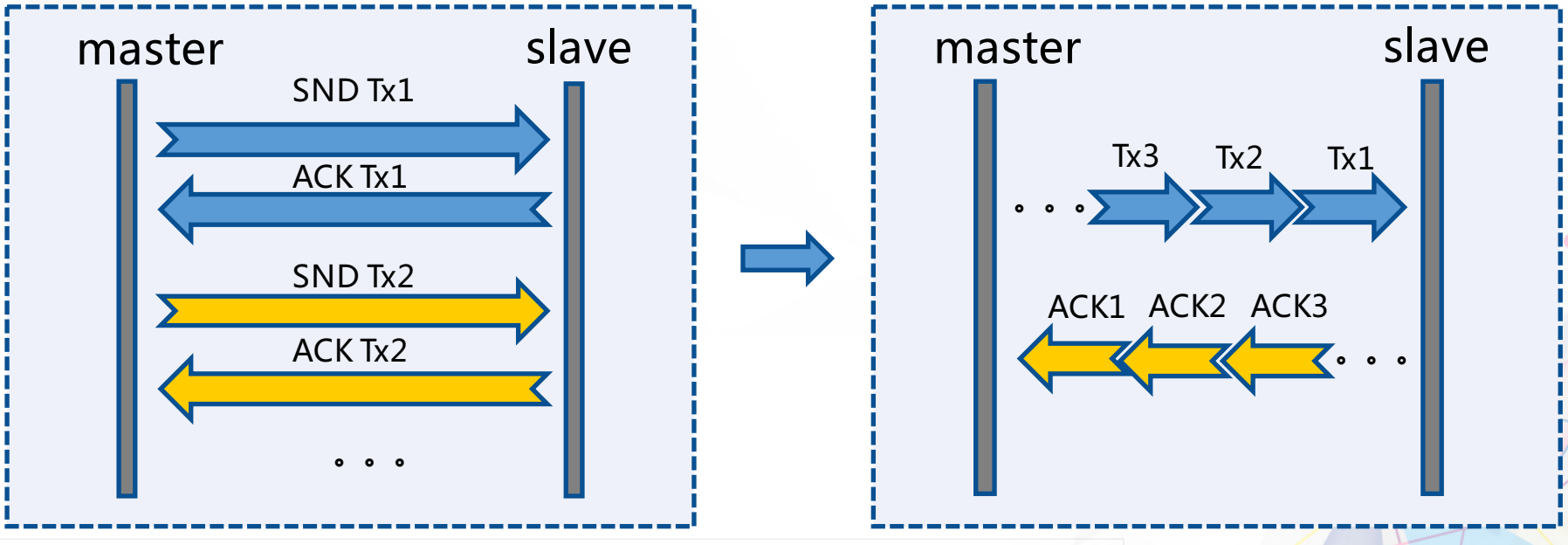
CDB之复制篇-性能优化(Cont.1)

- MySQL slave的IO线程接收binlog耗时原因
  - ✓ 锁冲突：IO/SQL线程间的锁冲突，如元数据文件锁
  - ✓ 小IO消耗：IO线程离散小磁盘IO消耗过多的IOPS
  - ✓ 串行化：IO线程接收和落盘操作串行
- 构建独立于MySQL的快速复制通道logbus
  - ✓ 基于semisync协议，模拟slave向master建立主从关系，同步binlog
  - ✓ 避免原生相关耗时瓶颈
  - ✓ 外置logbus，减少对MySQL的侵入，方便各种分支兼容

数据库版本	同步类型	TPS	单事务耗时(ms)	同步RTT	性能基准对比
MySQL5.7	异步	33193	3.82	2.60	100%
MySQL5.7	半同步	15395	8.32	2.60	46.30%
MySQL5.7	logbus	22169	5.92	2.60	66.79%

CDB之复制篇-性能优化(Cont.2)

- 优化MySQL 5.6的binlog发送、响应
  - ✓ binlog同步阻塞传输模型：master binlog dump线程发送某个事务binlog后，等待slave回包后，再发送下一个事务binlog。理论上最大QPS=1000/RTT(ms)。一般情况同城跨园区之间RTT为3ms左右，$QPS_{峰值}$约330
  - ✓ 方案： master binlog发送和接收异步化，dump线程负责发送，ack线程负责处理回包，通知事务线程继续提交



CDB之引擎篇-TXSQL

阶段1　阶段2　阶段3　阶段4

- 自研技术：20+
- 社区红利：30+

CDB之引擎篇-TXSQL特性概览

THANKS

SequeMedia 盛拓传媒 | IT168.com 专注导购16年 | ChinaUnix.net | ITPUB www.itpub.net