NewSQL 简介

2018-04-08

数据库的发展历史

- 传统关系型数据库
  - 单机, 一体的(monolithic)
  - 例子: MySQL/PostgreSQL/...

数据库的发展历史

- 传统关系型数据库
  - 单机, 一体的(monolithic)
  - 例子: MySQL/PostgreSQL/...
- 数据库中间层/集群模式(Middleware/Cluster/Sharding)
  - 模式: 读写分离, 分库分表
  - 解决的问题? 导致的问题?
  - 例子: MaxScale/GreenPlum/Vitess/...

数据库的发展历史

- 传统关系型数据库
  - 单机, 一体的(monolithic)
  - 例子: MySQL/PostgreSQL/...
- 数据库中间层/集群模式(Middleware/Cluster/Sharding)
  - 模式: 读写分离, 分库分表
  - 解决的问题? 导致的问题?
  - 例子: MaxScale/GreenPlum/Vitess/...
- NoSQL运动(Not only SQL)
  - 分布式, 扩展性好, 不要求SQL的全部特性
  - 数据模型不同: Row vs KV/Document/Graph/...
  - 牺牲事务/一致性要求
  - 例子: DynamoDB/HBase/Redis/Cassandra/MongoDB/...
  - 最终一致性(eventual consistency) 带来的问题?

What is NewSQL ?

RDS (SQL) with ACID guarantees, and NoSQL-like scalable performance for OLTP workloads.

- 关系性数据库的使用方式，同时兼有NoSQL数据库的性能和可扩展性，也能保证事务性(ACID).
- 专注于处理OLTP类型的数据需求. OLAP不是NewSQL首要解决问题领域.
- 例子: Google Spanner / CockroachDB / TiDB / VoltDB / MemSQL /AWS Aurora / ...

NewSQL 特点

- SQL API: 大家还是习惯SQL表达的灵活性
- 强一致性保障, 简化业务逻辑处理复杂度
- 从设计上就是分布式/ 冗余/ 横向扩展/ 高可用/ 自愈的
- DBaaS (Database as a service) 按需/实际使用量付费
- 和SQL over Hadoop HDFS 体系区别: 自己管理分布式存储, 计算找数据节点以就近计算, 而不是读取数据以计算.
- 未来趋势: HTAP (hybrid transaction-analytical processing), OLTP / OLAP 一套带走.

AWS Aurora 简介

- 计算与存储解耦, 将存储从数据库引擎中剥离, MySQL代码, IO相关重写
- (redo) log is the database
- quorum writes to distributed storage layer
- MySQL数据页淘汰时写回磁盘, Aurora数据页直接丢弃; 缺页按需构建, 以及后台通过redo日志主动构建
- Online DDL Support
- 目前还是一主多从策略, single-writer, multiple-reader
- (Preview) Multi-Master Support
- (Preview) DBaaS, Serverless, 按需收费

Spanner & F1 简介

Google 内部服务演化历史

- GFS (HDFS) -> Colossus / GFS II
- BigTable (HBase): versioned key-value store, single row level transaction
- BigTable + SQL -> MegaStore
- Spanner: successor of BigTable, cross-region replication, multi-row transaction support
- F1, external RDS over Spanner
- Spanner with built-in SQL support

F1

F1 (Filial 1 hybrid / 第一代杂交种): NoSQL + RDS

Pre-F1: MySQL + Sharding

F1 Server 本身无状态, 存储及事务依赖底层Spanner (可扩展, 支持事务的KV 存储). 可以理解为在Spanner外部的RDS实现.

表主键必须有继承关系, 从而实现row key 的局部性.

Spanner

- scalable KV store + mult-row transaction + strongly-typed schema system, SQL support, ... -> RDS
- 跨区域同步, 自动扩容, 故障切换, ...
- 适合读多的场景, 乐观锁, 读写实现依赖2PC/2PL.
- 外部一致性(external consistent / linearizability)
- 严重依赖全局时钟服务(TrueTime API), 时间戳作为版本号
- replicated WAL redo log over consensu group (paxos)
- Cloud Spanner from GCP

TiDB & CockroachDB

- 相同点
  - 思路来自F1 / Spanner
  - RocksDB 作为底层KV存储
  - Raft 算法
  - Go 语言
- 区别:
  - multi-layered (tisql/pd/tikv)
  - Global Distributed vs Single region cluster
  - PostgresSQL 协议/ MySQL 协议支持

Reference

- 2017 Amazon Aurora: Design Considerations for High Throughput Cloud-Native Relational Databases
- 2012 Spanner: Google's Globally-Distributed Database
- 2013 F1: A Distributed SQL Database That Scales
- 2017 Spanner: Becoming a SQL System
- 2017 Spanner, TrueTime and the CAP Theorem
- https://aws.amazon.com/rds/aurora/
- https://cloud.google.com/spanner/
- PingCAP
  - https://pingcap.com/blog-cn/
  - https://pingcap.com/blog-cn/tidb-internal-1/
  - https://pingcap.com/blog-cn/tidb-internal-2/
  - https://pingcap.com/blog-cn/tidb-internal-3/