# OceanBase

## 背景

淘宝收藏夹应用，包含info表和item表，info表是收藏信息的条目，item是收藏的宝贝和店铺的详细信息。

如果用户按照商品价格排序，那么要先从item表中读取收藏物品的价格等最新的信息，耗时会比较大；如果要把物品的详细信息冗余到info表，那么就不需要从item取数据了，但是许多商品有非常多的人收藏，这些商品价格的改动会导致info表大量的修改。

## 设计思路

数百TB的数据量、数十万TPS的访问量，采用任何单台的关系型数据库都无法应对，一张做法是根据业务特点对数据库水平拆分，比如根据用户id哈希取模后分布到不同的数据库，再加一个中间层路由到不同的分区，但是这种方式还是有很多的弊端：

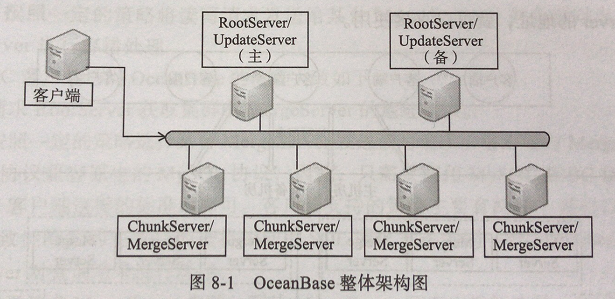
1. 数据量和负载增加后想要添加机器操作很难，需要人工介入。（可以采用一致性哈希）
2. 有些sql语句需要访问几乎所有的分区，比如是按照用户id分区，查询收藏了一个商品的用户需要访问所有的分区；

另一种做法是参考分布式表格的做法，将大表划分成很多个子表，子表之间按照主键有序，某台服务器发送故障，它上面服务的数据能够在很短的时间内自动迁移到集群中其他的服务器。这种方式解决了可扩展性的问题，少量的突发事件对使用者是透明的。这种分布式的做法虽然解决了扩展性的问题，但是却无法支持事务。比如BigTable只支持单行事务，OceanBase希望能够支持跨行跨表事务。

最直接的做法是在BigTable开源实现（HBase或者Hypertable）的基础上引入两阶段提交协议支持分布式事务，但是这种方式对事务的响应时间也是很高的，Percolator中事务的平均响应时间在2-5秒。

虽然在线业务的数据量十分庞大，上百亿条，但是最近一段时间修改量并不多，几千万或者上亿条，OceanBase采用单台更新服务器来记录最近一段时间的修改增量，而之前的数据保持不变，之前的数据成为基线数据。基线数据以类似分布式文件系统的方式存储在多台基线数据服务器中，每次查询要把基线数据和增量数据融合后返回给客户端。写事务都集中在单台更新服务器上，避免了复杂的分布式事务，高效的实现了跨行跨表事务。更新服务器上的修改增量也会定期发给多台基线数据服务器中，避免成为瓶颈。

## 系统架构



### client

类似mysql的JDBC操作；

### RootServer

主要功能包括：集群管理、数据分布以及副本管理。管理集群中所有的MergeServer、ChunkServer以及UpdateServer。

一个集群内部同一时刻只允许一个UpdateServer提供写服务，称为主UpdateServer，牺牲了一定的可用性获取了强一致性，RootServer通过租约机制选择唯一的主UpdateServer。

RootServer采用一主一备的结构，主备之间数据强同步，通过Linux HA软件实现高可用。

### UpdateServer

存储系统中增量更新数据。UpdateServer也是一主一备，主备之间可以配置不同的同步模式。部署时，UpdateServer和RootServer进程共用物理服务器。

UpdateServer是集群中唯一能够接受写入的模块，每个集群只有一个主UpdateServer。UpdateServer中的更新操作首先写入到内存表，当内存表数据量过大时生成快照文件转存到SSD中。

为保证可靠性，主UpdateServer更新内存表前先写操作日志，并同步到备UpdateServer。由于集群中只有一台主UpdateServer提供写服务，因此OceanBase很容易实现跨行跨表事务，而不需要采用传统的两阶段提交协议。但是这样集群中所有的读写操作都必须经过UpdateServer。OceanBase通过定期合并和数据分发这两种机制将UpdateServer一段时间之后的增量更新源源不断的分散到ChunkServer，而UpdateServer只需要服务最新一小段时间新增的数据，这些数据往往可以全部存放在内存中。

### ChunkServer

存储系统中的基线数据。

### MergeServer

**接收并解析用户sql请求**，对sql请求做一系列优化后传给相应的ChunkServer或者UpdateServer。如果请求的数据分布在多台ChunkServer，MergeServer还需要对多台ChunkServer返回的结果合并。