SQL到NOSQL的思维转变

NOSQL系统一般都会宣传一个特性，那就是性能好，然后为什么呢？关系型数据库发展了这么多年，各种优化工作已经做得很深了，NOSQL系统一般都是吸收关系型数据库的技术，然后，到底是什么因素束缚了关系型数据库的性能呢？我们从系统设计的角度看这个问题。

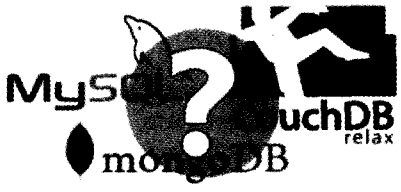
1. 索引支持。关系型数据库创立之初没有想到今天的互联网应用对可扩展性提出如此高的要求，因此，设计时主要考虑的是简化用户的工作，SQL语言的产生促成数据库接口的标准化，从而形成了Oracle这样的数据库公司并带动了上下游产业链的发展。关系型数据库在单机存储引擎支持索引，比如Mysql的Innodb存储引擎需要支持索引，而NOSQL系统的单机存储引擎是纯粹的，只需要支持基于主键的随机读取和范围查询。NOSQL系统在系统层面提供对索引的支持，比如有一个用户表，主键为user_id，每个用户有很多属性，包括用户名，照片ID，照片URL，在NOSQL系统中如果需要对photo_id建立索引，可以维护一张分布式表，表的主键为<photo_id, user_id>形成的二元组。关系型数据库由于需要在单机存储引擎层面支持索引，大大降低了系统的可扩展性，使得单机存储引擎的设计变得很复杂。

2. 事务并发处理。关系型数据库有一整套的关于事务并发处理的理论，比如锁的粒度是表级，页级还是行级，多版本并发控制机制MVCC，事务的隔离级别，死锁检测，回滚，等等。然而，互联网应用大多数的特点都是多读少些，比如读和写的比例是10：1，并且很少有复杂事务需求，因此，一般可以采用更为简单的copy-on-write技术：单线程写，多线程读，写的时候执行copy-on-write，写不影响读服务。NOSQL系统这样的假设简化了系统的设计，减少了很多操作的overhead，提高了性能。

3. 动态还是静态的数据结构。关系型数据库的存储引擎总是一颗磁盘B+树，为了提高性能，可能需要有insert buffer聚合写，query cache缓存读，经常需要实现类似Linux page cache的缓存管理机制。数据库中的读和写是互相影响的，写操作也因为时不时需要将数据flush到磁盘而性能不高。简而言之，关系型数据库存储引擎的数据结构是通用的动态更新的B+树，然而，在NOSQL系统中，比如Bigtable中采用SSTable + MemTable的数据结构，数据先写入到内存的MemTable，达到一定大小或者超过一定时间才会dump到磁盘生成SSTable文件，SSTable是只读的。如果说关系型数据库存储引擎的数据结构是一颗动态的B+树，那么SSTable就是一个排好序的有序数组。很明显，实现一个有序数据比实现一个动态B+树且包含复杂的并发控制机制要简单高效地多。

4. Join操作。关系型数据库需要在存储引擎层面支持Join，而NOSQL系统一般根据应用来决定Join实现的方式。举个例子，有两张表：用户表和商品表，每个用户下可能有若干个商品，用户表的主键为<user_id, item_id>，用户和商品的关联属性存放在用户表中，商品表的主键为item_id，商品属性包括商品名，商品URL，等等。假设应用需要查询一个用户的所有商品并显示商品的详细信息，普通的做法是先从用户表查找指定用户的所有item_id，然后对每个item_id去商品表查询详细信息，即执行一次数据库Join操作，这必然带来了很多的磁盘随机读，并且由于Join带来的随机读的局部性不好，缓存的效果往往也是有限的。在NOSQL系统中，我们往往可以将用户表和商品表集成到一张宽表中，这样虽然冗余存储了商品的详细信息，却换来了查询的高效。

关系型数据库的性能瓶颈往往不在SQL语句解析上，而是在于需要支持完备的SQL特性。互联网公司面临的问题是应用对性能和可扩展性要求很高，并且DBA和开发工程师水平比较高，可以通过牺牲一些接口友好性来换取更好的性能。NOSQL系统的一些设计，比如通过宽表实现Join操作，互联网公司的DBA和开发工程师也做过，NOSQL系统只是加强了这种约束。从长远来看，可以总结一套约束集合，并且定义一个SQL子集，只需要支持这个SQL子集就可以在不牺牲可扩展性的前提下支持比如90%以上的互联网应用。我想，NOSQL技术发展到这一步的时候就算是比较成熟了，这也是我们最终想做的事情。我们在设计和使用NOSQL系统的时候也可以适当转化一下思维，如下：



1. 更大的数据量。很多人在使用Mysql的过程遇到记录条数超过一定值，比如2000W的时候，数据库性能开始下降，这个值的得出往往需要经过大量的测试。然而，大多数的NOSQL系统可扩展性都比较好，能够支持更大的数据量，因此也可以采用一些空间换时间的做法，比如通过宽表的方式实现Join。

2. 性能预估更加容易。关系型数据库由于复杂的并发控制，insert buffer及类似page cache的读写优化机制，性能估算相对较难，很多时候需要凭借经验或者经过测试才能得出系统的性能。然后，NOSQL系统由于存储引擎实现，并发控制机制等相对简单，可以通过硬件的性能指标在系统设计之处大致预估系统的性能，性能预估可操作性相对更强。

LUPA开源社区

SQL到NOSQL的思维转变

刊名：硅谷

英文刊名：

年，卷(期)：2012(4)