《08\_面向高并发场景的NoSQL数据库：KV存储（3）》

互联网公司里，几乎是绕不过去一个话题，NoSQL数据存储中的一种，分布式KV存储

核心业务系统，数据库 -> 分库分表 -> 分布式数据库，如果有搜索场景，用ES作为一种数据存储的补充技术，互联网遇到这样的一些场景，核心业务系统，他们也不需要事务，也不需要复杂的库表数据模型，也不需要执行复杂的SQL语句

数据模型很简单的，key-value，kv数据格式，简单的kv读写的操作，互联网公司核心业务场景里会遇到这类场景，读写并发很高，比较高并发的写和比较高并发的读，当同时满足这些业务场景和情况的时候，一般来说，就可以引入NoSQL数据库里的分布式kv存储

社交系统，数据模型比较简单，不需要大量的表和复杂的字段，简单的读写操作，并发量很高，直接面向C端用户的，社交场景的系统，不适合使用MySQL / TiDB这种关系型的数据库来支持的

数据模型简单（key-value） + 读写操作（kv读写）+ 读写并发很高，挂很多的从库，分库分表，TiDB，浪费！高并发的支持就不是特别好，8核16GB的机器，2000 QPS，2万QPS，此时你需要10台机器来支撑

NoSQL数据库

Redis（开启持久化，保证数据可能会丢失一点点，但是不会丢失太多），kv，kv读写，数据结构和基于数据结构的复杂操作，读写并发能力极为强悍，主要是基于内存来实现读写操作，轻松可以支持几万QPS，甚至10万QPS

数据存储主要是基于内存，虽然会基于磁盘做持久化，避免数据丢失，全量数据都是放在内存里的，他不适合海量数据的场景，kv存储，热点数据缓存

中大型互联网公司，自研分布式kv存储，淘宝开源的Tair，支持存储海量数据，读写自动内存热点数据+磁盘存储海量全量数据，性能很高，QPS也是几万甚至十万，都会自己研发类似的分布式kv存储

数据库 + 分库分表 / TiDB

Elasticsearch

分布式KV存储：自研，HBase

HBase，大数据架构那边的课程，会对他的原理有一个很深入的理解

面向列的存储，核心就是kv存储，半结构化，

不停的灌入大量的简单的kv模型的数据，分布式存储，扩容、运维都很方便，写入kv数据，对他来说，并发能力是超高的，不支持事务，直接写入内存buffer缓冲就可以了，WAL预写日志

只要你是根据rowkey去进行读取，有一定的概率会命中内存里的缓存，直接从内存里加载数据出来，布隆过滤器、LSM索引树，大幅度的提升和优化了rowkey的读取，性能和并发也是不错的

海量数据，模型简单，操作简单，读写并发能力和性能也还是不错的

HBase的NoSQL数据库也好，分布式KV存储也好，其实可以支持模型简单、操作简单、数据量很大、并发很高，HBase来支撑类似的场景也是比较多的，在电商卖家日志系统里，去做一个项目实战

3616118202

《05\_分布式搜索引擎：关系型数据库的一种补充》

普通数据库架构 -> 读写分离 + 分库分表 + 数据库中间件（MyCat / Sharding-Sphere）- > 最近这一年，出现了一个趋势，涌现出了一个非常成功和比较好的分布式数据库的技术，TiDB

TiDB，分布式关系型数据库，SQL语法支持、事务支持、各种功能支持，都是要有的，适合的也是那种核心业务系统，商家系统，商品系统，订单系统

要做一些搜索类的业务，要做一些简单的语义分析和基于语义的搜索

1 我喜欢吃鸡腿

2 我除了鸡腿，还是吃拉面

3 鸡腿蛋炒饭的一个做法

select \* from xxx where id=1

搜索“鸡腿”，只要跟鸡腿有关的文本，都需要给我查询出来

MySQL，关系型数据库，也可以通过建立全文索引来支持类似这种基于文本和语义的搜索，他们对这块的支持是非常的不好的

Elasticseach，就是用来做搜索的，对你的各种文本建立倒排索引，然后可以基于文本和语义去进行搜索，在数据存储架构里，不是必须要有的，一般来说是你出现搜索业务场景的时候，会作为补充引入ES技术

3616118202

3616118202

《01\_学习海量数据架构设计实战之前，先夯实理论基础》

官网公布的2020年新版本大纲里，在业务架构之后，海量数据架构，符合演进的过程，大部分的系统，刚开始主要都是做业务，DDD业务建模，软件工程理论和流程，项目管理的流程，推动一个业务系统的开发，基础框架+分布式架构+微服务架构+持续集成/持续交付，数据库的表设计，基本上一个系统已经可以出来了

高并发、高性能、高可用、稳定性的问题

国内就只有少数公司是真的有高并发的场景和问题的，大多数公司的并发都是很低的，哪怕是一些互联网公司，并发也没那么高；高性能，系统的数据量就很少，就压根儿就没有性能问题；比较简单的话，也谈不上高可用；稳定性

海量数据问题，数据量可能会比较大一些，海量数据存储架构如何设计，技术选型

MySQL、分库分表、数据库中间件、TiDB、MongoDB、HBase、Elasticsearch、Redis，各自的特点是什么，各自的优劣势是什么，分别适合在哪些场景下使用，了解清楚了，再针对不同的业务场景，学习不同场景下的数据存储架构设计实战，结合具体的项目来做

不同的数据存储技术，考虑哪些因素：

运维成本：监控告警、备份恢复、升级和迁移、社区活跃度、性能优化、troubleshooting

稳定性：数据多副本、高可用、多写多活

性能：延迟、QPS，等等

扩展性：横向扩容和纵向扩容，等等

安全：权限，认证，审计，等等

《06\_面向高并发场景的NoSQL数据库：KV存储》

互联网公司里，几乎是绕不过去一个话题，NoSQL数据存储中的一种，分布式KV存储

核心业务系统，数据库 -> 分库分表 -> 分布式数据库，如果有搜索场景，用ES作为一种数据存储的补充技术，互联网遇到这样的一些场景，核心业务系统，他们也不需要事务，也不需要复杂的库表数据模型，也不需要执行复杂的SQL语句

数据模型很简单的，key-value，kv数据格式，简单的kv读写的操作，互联网公司核心业务场景里会遇到这类场景，读写并发很高，比较高并发的写和比较高并发的读，当同时满足这些业务场景和情况的时候，一般来说，就可以引入NoSQL数据库里的分布式kv存储

数据模型简单（key-value） + 读写操作（kv读写）+ 读写并发很高，挂很多的从库，分库分表，TiDB，浪费！高并发的支持就不是特别好，8核16GB的机器，2000 QPS，2万QPS，此时你需要10台机器来支撑

3616118202

3616118202

《09\_MongoDB这种NoSQL数据存储一般适合什么场景？》

数据库 + 分库分表 / TiDB，适合的场景是核心业务系统

Elasticsearch，适合的场景是搜索

分布式KV存储：自研，HBase，简单的KV模型+简单的KV读写+海量数据+高并发，有一些特殊的场景，电商卖家日志系统

NoSQL：MongoDB，适合的场景类似于社交系统，电商买家社交系统

不太好说把MongoDB划分到分布式KV存储里去，数据模型有点类似于ES，是面向文档（Document）的数据模型，他支持的数据模型相对来说会复杂一些，也可以建立索引，数据存储经过了大量的优化，稍微有一点点复杂度的数据模型，支持稍微复杂一点的功能，事务，海量数据+高并发，分布式架构

3616118202

《04\_分库分表架构的未来演进趋势：分布式数据库》

数据库普通开发架构 -> 读写分离 + 分库分表 + 数据库中间件（MyCat / Sharding-Sphere）- > 最近这一年，出现了一个趋势，涌现出了一个非常成功和比较好的分布式数据库的技术，TiDB

分布式数据库，彻底替代比较复杂的、有很多痛点的数据库架构：读写分离 + 分库分表 + 数据库中间件（MyCat / Sharding-Sphere）

后续做海量数据存储架构实战，分库分表那块，我们会用真实的复杂订单系统来做实战，包括如何设计方案，如何做数据迁移，如何无缝接入系统，如何基于中间件支持全部订单功能，如何基于ES做订单搜索，这是一个完整的实战，但是大家也必须意识到，这种分库分表架构终将成为一个过渡方案，因为无论如何，这个SQL语法支持是不完善的，同时运维起来是极为麻烦的，所以会进入分布式数据库的阶段

TiDB，分布式数据库，架构上就是完全基于多台服务器形成的一个分布式的架构，灌入数据，内部自动把数据均匀分散在多台服务器上面，本质上就跟分库分表是一个概念，运维，他内部给你透明的完成了，不需要你自己去关注这个东西，扩容，加几台服务器给TiDB集群，内部可以自动的数据迁移和扩容，分布式事务，支持的也会更好，复杂SQL语法的支持，也会更好

所以未来的一个趋势就是：分布式数据库，以TiDB为首，完全分布式，扩容、运维，都极为的方便，支持更加完善的复杂的SQL语法，同样包含事务支持，让研发人员使用起来，不用去care所谓的分多少库和表，扩容的时候也不用care这些细节

过去一年多里，很多互联网公司都开始在引入TiDB这个技术，部分开始替代原来的分库分表的架构，坑，踩坑，解决他自己的一些问题，还是比自己去玩儿原始的分库分表要好多了，TiDB会成为替代分库分表的主流

接着我们会给大家讲解，如何用分布式数据库去重构订单系统数据存储架构

3616118202

《03\_基于MySQL的分库分表架构：痛并快乐着！》

备份恢复+高可用，都是mysql比较常规的功能和架构，网上一搜，多的是

一般中小型公司，如果没有百亿级数据、每秒上万TPS写入压力的话，其实就做一个MySQL数据库的备份恢复方案+高可用自动故障转移方案，就足够了，但是如果是有点规模的互联网公司，数据量几十亿到百亿级，写入TPS轻松破万，那么互联网公司玩读写分离+分库分表基本就是个标配

读写分离，主库 + 几个从库，主库 -> 从库会同步数据，半同步，数据会有一定的延迟，几十ms，写入就往主库里写，查询可以从主库或者是其他的从库里查，多挂几个从库，就可以分摊你的查询QPS

数据量很大，上百亿数据量，一台数据库服务器磁盘空间都放不下，磁盘都快满了；写入TPS压力极大，每秒钟写入压力要达到上万，分库分表

其实分库分表往简单了说也并不难理解，就是搞好几台数据库服务器，创建一堆库，每个表搞一堆表，分散在各个数据库服务器上，然后这样一个表的数据就分散在多台服务器上了，每秒上万的写入TPS就分散在多台服务器上了

写入和查询，如果是简单的那条根据id主键的单条数据，很简单，做一个路由，路由到某个库某个表，写入到里面，以后也从这个库这个表里查就行了

但是，真的这么简单吗？万一你要搞一些很复杂很恶心的查询呢？跨表？跨库？这怎么办，要不然基于mycat、sharding-sphere之类的中间件执行，他们支持的复杂SQL语法还有限制，而且会基于内存来做，效果也不是太好，要不然就是用ES之类的额外数据存储，全量数据打进去，然后重建数据模型来做复杂查询

另外，运维成本呢？相当高，修改表字段，建表之类的，一搞就是多个库多个表，弄的很麻烦不是吗？还有最恶心的就是你扩容的时候，怎么扩？做数据迁移？那是相当的麻烦，所以是痛并快乐着

分布式事务，SQL，跨多个数据库和多个表去执行的，事务如何保证？

《07\_面向高并发场景的NoSQL数据库：KV存储（2）》

互联网公司里，几乎是绕不过去一个话题，NoSQL数据存储中的一种，分布式KV存储

核心业务系统，数据库 -> 分库分表 -> 分布式数据库，如果有搜索场景，用ES作为一种数据存储的补充技术，互联网遇到这样的一些场景，核心业务系统，他们也不需要事务，也不需要复杂的库表数据模型，也不需要执行复杂的SQL语句

数据模型很简单的，key-value，kv数据格式，简单的kv读写的操作，互联网公司核心业务场景里会遇到这类场景，读写并发很高，比较高并发的写和比较高并发的读，当同时满足这些业务场景和情况的时候，一般来说，就可以引入NoSQL数据库里的分布式kv存储

社交系统，数据模型比较简单，不需要大量的表和复杂的字段，简单的读写操作，并发量很高，直接面向C端用户的，社交场景的系统，不适合使用MySQL / TiDB这种关系型的数据库来支持的

数据模型简单（key-value） + 读写操作（kv读写）+ 读写并发很高，挂很多的从库，分库分表，TiDB，浪费！高并发的支持就不是特别好，8核16GB的机器，2000 QPS，2万QPS，此时你需要10台机器来支撑

NoSQL数据库

Redis（开启持久化，保证数据可能会丢失一点点，但是不会丢失太多），kv，kv读写，数据结构和基于数据结构的复杂操作，读写并发能力极为强悍，主要是基于内存来实现读写操作，轻松可以支持几万QPS，甚至10万QPS

数据存储主要是基于内存，虽然会基于磁盘做持久化，避免数据丢失，全量数据都是放在内存里的，他不适合海量数据的场景，kv存储，热点数据缓存

中大型互联网公司，自研分布式kv存储，淘宝开源的Tair，支持存储海量数据，读写自动内存热点数据+磁盘存储海量全量数据，性能很高，QPS也是几万甚至十万，都会自己研发类似的分布式kv存储

3616118202

《02\_国内互联网公司标配：MySQL数据库》

国内互联网公司标配，传统软件公司，Oracle、SQL Server + .NET

MySQL的特点是，完善的关系型数据库的模型，支持完善的SQL语法，而且有事务支持，性能还不错，而且数据库必备的一些功能通常也都有，非常适合互联网公司的核心系统的开发，免费，开源

但是缺点也比较明显，因为他支持复杂SQL以及事务，儒猿技术窝，《从0开始带你成为MySQL实战高手》，面向磁盘，用很多内存来读写数据，支持锁，事务也要做很多的事情，所以TPS是比较低的，一般8核16G的机器，大概也就支持两三千的TPS，机器负载就很高了，有点扛不住了

MySQL，天生就不是用来抗高并发的

所以一般在互联网公司里用MySQL，一定会做这么几块架构：

（1）数据备份和恢复，一般是日备份、周备份，会设定几个级别

（2）高可用架构，也就是做master-slave主从架构，数据实时同步，读写分离开来，一定程度上扩展数据库的读QPS，同时可以做故障自动切换，当然也可以采用MHA之类的其他高可用架构

（3）读写分离：就是上面说的，如果读压力比较大，就多挂几个从库扩容读QPS

（4）分库分表：搞多台数据库服务器组成一个分库分表的集群，然后基于mycat、sharding-sphere之类的中间件去支持，这样可以对数据存储容量进行扩容，同时可以扩展数据库写入TPS

至于说业务场景，说实话，一般的系统，只要没特殊要求，刚开始就都用MySQL就行了，如果后续感觉有点瓶颈了，再用别的数据存储就行3616118202

《10\_海量数据场景下的难题：数据迁移和数据同步》

数据库 + 分库分表 / TiDB，适合的场景是核心业务系统，订单系统

Elasticsearch，适合的场景是搜索，订单系统

分布式KV存储：自研，HBase，简单的KV模型+简单的KV读写+海量数据+高并发，有一些特殊的场景，电商卖家日志系统

NoSQL：MongoDB，适合的场景类似于社交系统，电商买家社交系统

不太好说把MongoDB划分到分布式KV存储里去，数据模型有点类似于ES，是面向文档（Document）的数据模型，他支持的数据模型相对来说会复杂一些，也可以建立索引，数据存储经过了大量的优化，稍微有一点点复杂度的数据模型，支持稍微复杂一点的功能，事务，海量数据+高并发，分布式架构

数据迁移，要从一个数据存储里把一大堆的数据迁移到另外一个数据存储里去，其实里面的技术方案和不同的场景，案例 -> 方案；数据同步，你要做异构数据存储架构，要在多种数据存储之间做同步，案例 -> 方案

SQL、NoSQL、NewSQL

可能会把数据存储技术选型和架构设计，理论知识铺垫，也可能没有

3616118202