第六周总结

首先是第六周的学习笔记：

6.1 分布式关系数据库（一）

1. Mysql的主从复制：通过binlog进行复制，实现了读写分离。
1. 优点：分摊负载；专机专用；便于冷备；高可用（读）
2. 缺点：没有写高可用
2. 主主复制：AB主服务器互相同步binlog，用于提升写操作高可用。为了防止数据冲突，并不是任何时候都可以同时写2个服务器的，要有一个切换的过程。虽然数据库允许这种操作，但是业务逻辑上不应该允许。更新表结构会导致巨大的binlog延迟。问题：没有增加写并发能力。
3. 数据分片：把记录分到多个服务器以提升处理能力
1. 硬编码或者映射表的缺陷：需要大量额外代码，处理逻辑变得复杂；无法执行多分片联合查询；无法使用数据库的事物；随着数据的增长，如何增加更多的服务器。

6.2 分布式数据库（二）

1. 分布式数据中间件把自己伪装成一个数据库。
2. Cobar集群工作模式：接收请求（模拟数据库通讯协议） -> sql解析（解析出分片字段和值） -> sql路由（决定提交哪些服务器） -> sql执行代理（提交到一个或者多个数据库分片） -> 结果合并模块
3. 关系型数据库的伸缩：首先要进行数据迁移。余数法一开始要规划好最终的机器数目，上限从一开始就是定的。在上限达到之前，每次伸缩需要进行数据迁移，并更改路由配置。

6.3 CAP原理与NoSQL数据库架构

1. CAP原理：一致性（consistency），可用性（avalibility），分区耐受性（partition tolerancce）：一个系统在满足分区耐受性的前提下，一致性和可用性是不能同时满足的
2. 最终一致性：数据在一段时间内不一致，但是在网络延迟恢复以后，数据是一致的。解决冲突时可能用时间戳，合并冲突等
3. 投票解决冲突
4. Cassandra服务器：一个写操作写到一个节点，这个节点会和其他的节点同步。一个读操作也会读至少3个服务器，最终通过投票返回最终数据
5. Hbase：底层时hadoop文件存储，上层则实现分布式的数据的逻辑。存储方式则为B+树
6. ACID与BASE（basically available）。NoSql遵循BASE，即允许一些中间状态存在，并提供最终一致性。

6.4 Zookeeper与分布式一致性架构

1. 分布式脑裂：分布式系统中，不同的服务器获得了相互冲突的数据信息或者执行指令，导致整个集群陷入混乱，数据损坏。
2. 分布式一致性算法Paxos，由proposer，acceptors，learner组成，通过投票得出主服务器。zookeeper本身同时充当proposer，acceptors，learner三个角色

3. Zookeeper的简化协议zab：分为leader和proposer两种角色。客户端提交一个请求给任何一个follower，follower转给leader，leader给所有follow发propose，如果多数follower决定接受并返回了ack，leader就会向所有follower提交commit

6.5 搜索引擎的基本架构

1. 搜索引擎包含模块：网络爬虫，网页去重，云存储与云计算（倒排索引，链接关系），网页反作弊，内容相似性，链接分析，网页排序，Cache系统，检索分析
2. 爬虫：种子url-> 爬 -> 分析得到更多url -> 去重 ->爬
3. 像Google这样的公司会和被搜索商家有约定协议，按照协议规定哪些可以爬哪些不可以爬
4. 倒排索引（哈希级别的速度）：词 -> 文档列表
5. 倒排表的词频可以辅助排序
6. elasticSearch把lucene包装成分布式：索引分片，实现分布式；索引备份，实现高可用；API更简单，更高级

6.6 NoSQL 案例：Doris分析案例（一）

1. 分为客户端和实际的集群。集群带有namespace管理，用于区分不同的业务方，客户端自己的key加上namespace才是真正的物理层的key。物理层分片，用于支持更高的性能。
2. 关键技术点：数据分片。客户端拉取config得知当前有多少分片，以及各个分片的ip地址等。客户度拿到这些数据通过自己的路由算法计算目标机器并发送请求获取结果。
3. 路由算法：基于虚拟节点的分区算法。即预先定下1W个虚拟节点，然后按照一定的方式维护一个虚拟节点到物理节点的路由表即可。这样有两个好处，一个是波动性比一致性哈希更小（X/(M+X)对比X/M），另一个是扩容迁移的时候更方便，每个虚拟节点对应一个文件，迁移只需要把整个虚拟节点的文件迁移即可，而不必像一致性哈希那样一个个key进行迁移。

6.7 NoSQL 案例：Doris分析案例（二）：高可用与集群扩容如何设计

1. 数据必须多份都写成功才算写成功，用户可以根据可用性要求配置数据的份数，最少2份
2. 由一个config server去管理整个集群的状态，哪些是有效的，哪些已经失效了，以维护整个集群状态的一致性。有点类似于zk（当时zk还不成熟）
3. 失效状态：瞬时失效，临时失效，永久失效。注意：临时失效2小时后会变成永久失效。
4. 瞬时失效的时候需要由config server进行复查，确认是否真的失效。临时失效的时候，把操作写入日志节点，等服务器恢复过来（2小时以内），再去redo。永久失效的时候，从集群中抓取一个空白服务器（standby），把文件拷贝过去，同时写日志文件，文件拷贝完以后再由日志文件进一步恢复。

6.8 NoSQL 案例：Doris分析案例（三）：扩容伸缩如何设计的?

1. 扩容过程：先把其中一个group的机器全部变成临时失效状态，这个group中增加一台服务器，开始进行虚拟节点的映射，数据的复制。完成后进入失效恢复状态，把之前的日志都追上去，然后即可正常提供服务。这些都完成后对另一个集群也采取相同的行为。

个人总结

这一周主要学习存储系统的分布式架构，存储系统要考虑数据的一致性问题，分布式架构的实现要比无状态服务更为复杂。本周内容主要分mysql类型的分库分表实现以及相应的中间件，利用到排表提供搜索服务的搜索引擎，和可伸缩性较好，并提供最终一致性的NoSql服务。穿插了分布式集群管理中间件zookeeper的介绍。一周的视频课篇幅，内容非常丰富，也是非常好的科普，很多以前看起来比较神秘的中间件，剥离出核心原理来，其实也不是那么复杂的东西，大道至简。