**8-分布式基础**

**Mapreduce**

**1、MapReduce**

Hadoop并行计算框架，为海量的数据提供了计算。

**2、MapReduce的原理**

分而治之，一个大任务分成多个小的子任务（map），并行执行后，合并结果（reduce）。

**3、基本概念**

（1）Job

作业，表示MapReduce程序，一个MapReduce程序可对应若干个作业，而每个作业会被分解成若干个MapReduce任务（Task）。

（2）Task

任务，分为Map Task和Reduce Task两种，均由TaskTracker启动。

（3）JobTracker

任务调度器，负责作业调度；分配任务、监控任务执行进度；监控TaskTracker的状态。

（4）TaskTracker

任务跟踪器，负责执行任务；向JobTracker汇报任务状态。

**4、MapReduce体系结构**

MapReduce采用Master/Slave的架构。

Master，是整个集群的唯一的全局管理者，负责作业调度、分配任务、监控任务执行进度和监控TaskTracker的状态，即MapReduce中的JobTracker。

Slave，负责任务的执行和任务状态的汇报，即MapReduce中的TaskTracker。

**5、MapReduce作业执行过程**

（1）首先对输入数据源进行切片。

（2）JobTracker调度TaskTracker并分配Map任务。

（3）TaskTracker读取输入源片段。

（4）TaskTracker执行Map任务，将任务输出保存在本地。

（5）JobTracker调度TaskTracker并分配Reduce任务。

（6）TaskTracker读取Map任务的输出文件。

（7）TaskTracker执行Reduce任务，将任务输出保存到HDFS。

**一致性哈希算法**

这个映射可以想到哈希算法，但是如果节点数量发生了变化，也就是在对系统做扩容或者缩容时，必须迁移改变了映射关系的数据，否则会出现查询不到数据的问题。这就需要我们进行迁移数据，假设总数据条数为 M，哈希算法在面对节点数量变化时，最坏情况下所有数据都需要迁移，所以它的数据迁移规模是 O(M)，这样数据的迁移成本太高了。

一致性哈希算法就很好地解决了分布式系统在扩容或者缩容时，发生过多的数据迁移的问题。一致哈希算法也用了取模运算，但与哈希算法不同的是，哈希算法是对节点的数量进行取模运算，而一致哈希算法是对 2^32 进行取模运算，是一个固定的值。

一致性哈希是需要两次哈希操作的，一致性哈希是指将「存储节点」和「数据」都映射到一个首尾相连的哈希环上。对「数据」进行哈希映射得到一个结果，映射的结果值往顺时针的方向的找到第一个节点，就是存储该数据的节点。也就是说，需要对指定的 key 值进行读写需要 2 步： - 首先，对 key 进行哈希计算，确定此 key 在环上的位置； - 然后，从这个位置沿着顺时针方向走，遇到的第一节点就是存储 key 的节点。

在一致哈希算法中，如果增加或者移除一个节点，仅影响该节点在哈希环上顺时针相邻的后继节点，其它数据也不会受到影响。但是一致性哈希算法并不保证节点能够在哈希环上分布均匀，这样就会带来一个问题，会有大量的请求集中在一个节点上。所以，一致性哈希算法虽然减少了数据迁移量，但是存在节点分布不均匀的问题。

要想解决节点能在哈希环上分配不均匀的问题，就是要有大量的节点，节点数越多，哈希环上的节点分布的就越均匀。但问题是，实际中我们没有那么多节点。所以这个时候就加入虚拟节点，也就是对一个真实节点做多个副本。具体做法是，不再将真实节点映射到哈希环上，而是将虚拟节点映射到哈希环上，并将虚拟节点映射到实际节点，所以这里有「两层」映射关系。虚拟节点除了会提高节点的均衡度，还会提高系统的稳定性。当节点变化时，会有不同的节点共同分担系统的变化，因此稳定性更高。带虚拟节点的一致性哈希方法不仅适合硬件配置不同的节点的场景，而且适合节点规模会发生变化的场景。