Hadoop集群单namenode问题

1. 单节点故障，网络异常

采用多namenode，ha高可用搭建，主备机制，2.x版本会有一个备用namenode，状态为standby，3.x版本最多可以有5个备用namenode，建议用3个备用namenode，越多备用namenode越占用io带宽。

Ha搭建集群分两块，第一块为主备节点和JN集群，可实现手动切换备用节点为主节点。第二块为每个namenode上的ZKFC进程和zookeeper集群，可实现自动切换备用节点为主节点。

第一块

怎样实现主节点和备用节点数据一致性，

1. 高一致性，客户端client发送一个操作到主节点active，主节点active执行后，再发送操作到备用节点standby，备用节点standby操作成功后返回给主节点active信息，主节点active最后返回给客户端client操作成功。若备用节点standby出现故障，则主节点active会一直等待，导致客户端clent也一直等待，从而降低了可用性。
2. 高可用性，客户端client发送一个操作到主节点active，主节点active执行后，发送操作到备用节点standby，同时返回给客户端client操作成功，备用节点操作成功后返回给主节点active信息。若备用节点standby出现故障，则导致主节点active和备用节点standby数据不一致。

引出CAP原理，在一个分布式系统中， Consistency（一致性）、 Availability（可用性）、Partition tolerance（分区容错性），三者不可得兼得。

CAP的定义

(1)C: Consistency 一致性

一致性又称为原子性或者事务性。表示一个事务的操作是不可分割的，要不然这个事务完成，要不然这个事务不完成，不会出现这个事务完成了一半这样的情况。这种事务的原子性使得数据具有一致性。

我们通常情况下在数据库中存在的脏数据就属于数据没有具有一致性的表现。而在分布式系统中，经常出现的一个数据不具有一致性的情况是读写数据时缺乏一致性。比如两个节点数据冗余，第一个节点有一个写操作，数据更新以后没有有效的使得第二个节点更新数据，在读取第二个节点的时候就会出现不一致的问题出现。

传统的ACID数据库是很少存在一致性问题的，因为数据的单点原因，数据的存取又具有良好的事务性，不会出现读写的不一致。

(2)A: Availability 可用性

好的可用性主要是指系统能够很好的为用户服务，不出现用户操作失败或者访问超时等用户体验不好的情况。通常情况下可用性和分布式数据冗余，负载均衡等有着很大的关联。

(3)P:Partition Tolerance分区容错性

分区容错性和扩展性紧密相关。在分布式应用中，可能因为一些分布式的原因导致系统无法正常运转。好的分区容错性要求能够使应用虽然是一个分布式系统，而看上去却好像是在一个可以运转正常的整体。比如现在的分布式系统中有某一个或者几个机器宕掉了，其他剩下的机器还能够正常运转满足系统需求，这样就具有好的分区容错性。

CAP之间的取舍

满足一致性，可用性的系统，通常在可扩展性上不太强大：

· Traditional RDBMSs like Postgres,MySQL, etc (relational)

· Vertica (column-oriented)

· Aster Data (relational)

· Greenplum (relational)

满足一致性，分区容忍必的系统，通常性能不是特别高:

· BigTable (column-oriented/tabular)

· Hypertable (column-oriented/tabular)

· HBase (column-oriented/tabular)

· MongoDB (document-oriented)

· Terrastore (document-oriented)

· Redis (key-value)

· Scalaris (key-value)

· MemcacheDB (key-value)

· Berkeley DB (key-value)

满足可用性，分区容忍性的系统，通常可能对一致性要求低一些:

· Dynamo (key-value)

· Voldemort (key-value)

· Tokyo Cabinet (key-value)

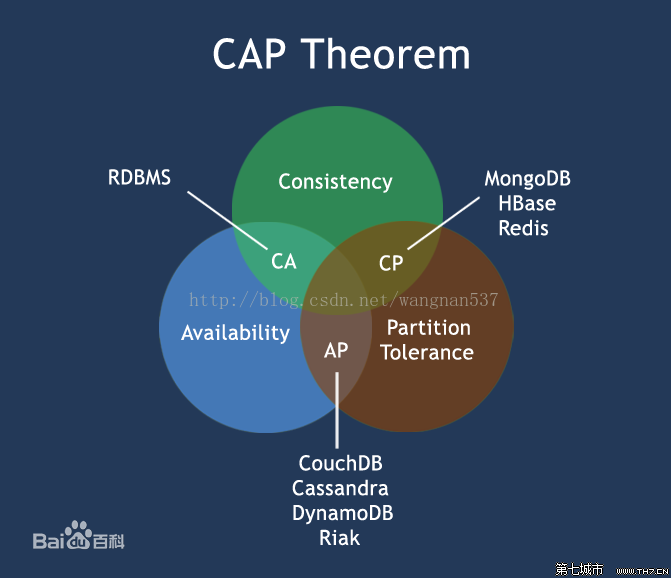
· KAI (key-value)

· Cassandra (column-oriented/tabular)

· CouchDB (document-oriented)

· SimpleDB (document-oriented)

· Riak (document-oriented)



解决办法，引入一个高可用，最终一致性的JN集群,JN的核心采用Paxos算法，当超过一半的操作成功，则认定整个集群操作成功，最终JN集群的数据是一致性的（具体细节未知）。

第二块

每个namenode服务上都有一个ZKFC，用于本地监控namenode状态，同时ZKFC会和zookeeper集群链接。每次启动集群的时候，每个namenode会去zookeeper争抢创建一个锁，哪个namenode先成功创建这个锁，则会作为主节点active。

当主节点active出现故障，ZKFC监测到后会向zookeeper取消之前创建的锁，然后zookeeper中会用回调call back通知所有namenode，让他们重新去争抢创建锁，由于主节点active故障，争抢不了，所以备用节点standby争抢成功，然后备用节点会访问主节点active，确认active故障后，自身转变成主节点active。

当ZKFC故障后，会自动断开与zookeeper的链接，zookeeper同样会用回调call back通知其他namenode，同样重新争抢创建锁，备用节点standby创建锁后去访问主节点，若主节点active没有故障，则将主节点转变成备用节点，自身转变为主节点。若主节点故障，同样，自身直接转变为主节点。

1. 单节点内存不足

采用多namenode，联邦制度搭建

多个namenode服务器，使用相同datanode服务器，datanode服务器之间用不同的目录分隔。

好处：多个namenode，降低内存并发压力。datanode服务器实现资源共享。

坏处：不能同时访问处于不同namenode节点上的元数据。

解决方法：搭建一个虚拟文件系统平台，通过平台封装api，统一查询不同namenode文件，同时可以融入不同种类的文件系统，比如FTP，实现多种文件系统统一查询的功能。