前言

HDFS，全称Hadoop Distributed File System，顾名思义，是Hadoop里面的分布式文件系统。

Hadoop 简介

随着信息数量以爆炸式增长，传统的数据库和计算引擎已经跟不上数据增长的速度以及业务对海量数据处理效率的要求，Hadoop因此产生并得到发展。

Hadoop是Apache开源的分布式软件开发框架，提供可靠的、可扩展的分布式计算能力。主要是通过**分而治之，移动数据不如移动计算**的核心思想，支持通过大量服务器组成集群，将数据存储在多个服务器上，每一台服务器都可以提供计算能力，从而大大提升数据计算能力，解决海量数据存储以及计算的难题，将计算任务分配到每台服务器上进行计算结果，最后统一汇总到管理节点得到最终的结果。

广义上来说，Hadoop通常是指一个更广泛的概念—Hadoop生态圈。

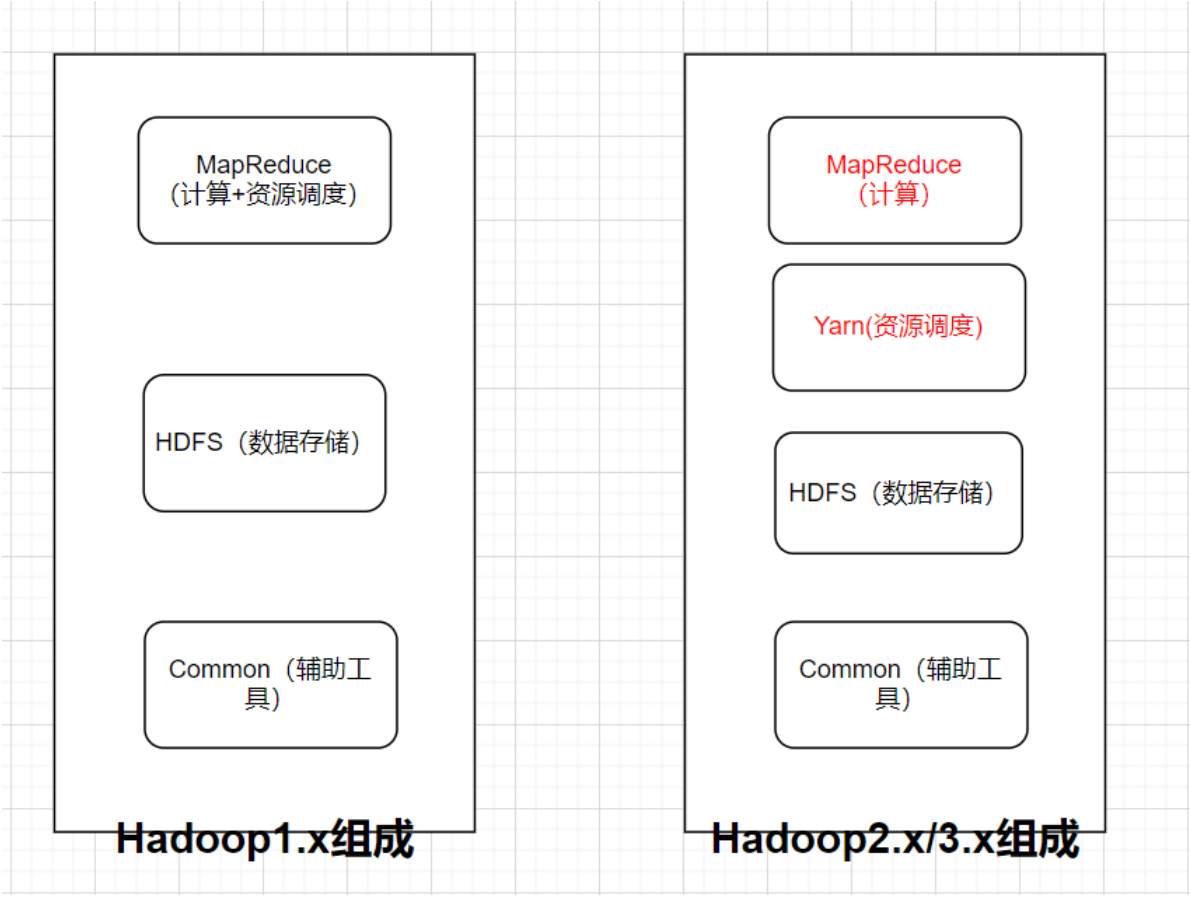
Hadoop的优势：

1. 高可靠性：Hadoop底层维护多个数据副本，所以Hadoop某个计算元素或存储出现故障也不会导致数据的丢失
2. 高扩展性：Hadoop在集群间分配任务数据，可以方便的扩展多个节点
3. 高效性：在Map Reduce的思想下，Hadoop是并行工作的，任务处理速度快
4. 高容错性：能够自动将失败的任务重新分配

Hadoop的诞生源于Google三篇论文，也就是我们经常说的“三辆马车”，2006年Hadoop出了第一个发行版本，Hadoop到目前为止发展已经有10余年，版本经过了无数次的更新迭代，目前业内大家把Hadoop大的版本分为Hadoop1，hadoop2，Hadoop3三个版本。

Google三篇论文	**Hadoop生态**
GFS分布式文件系统	HDFS
MapReduce计算模型	MapReduce
BigTable	Hbase

> 本文图片均来自draw.io手工绘制



HDFS

HDFS（Hadoop Distributed File System），它是一个文件系统，用于存储文件，通过目录树来定位文件；其次，它是分布式的，由很多服务器联合起来实现其功能，集群中的服务器有各自的角色。

HDFS的使用场景：适合一次写入，多次读出的场景。一个文件经过创建、写入和关闭之后就不需要改变。

核心概念

HDFS有两个核心特征：分布式、文件系统。

从文件系统的角度来看，HDFS提供了一个统一的命名空间——**目录树**来组织文件，其操作命令的形式跟Linux操作系统基本保持一致，目录树，是逻辑上的概念，用来屏蔽底层复杂的存储和相关操作的细节，让用户感觉像在操作本地文件系统一样。

因为从分布式的角度来看，一个文件可能会被切割成多个**数据块**，分散存储到多台机器上。数据块，是物理上的概念，决定了文件的具体存储形式。

另一方面，Hadoop在设计和使用上，有一个前提：允许集群中少量机器在某个时刻发生故障。为了达到这个目的，需要将文件拷贝多份放在不同的机器上，即**Replication**。

为什么需要引入数据块？其实是我们常说的“分而治之”的思想。假设没有数据块，直接以整体的形式存储一个文件，就容易出现下面问题：

- 集群中的机器使用不均匀。比如将一个20GB的文件完整存放到机器A上，势必会导致机器A的负载更重。
- 故障恢复慢。对于一个20GB文件，如果所在的某台机器故障了，整个系统需要重新搬移数据，保持足够的Replica时，需要一次性移动20GB的数据，会带来较高的负载。
- 无法并行加载文件。文件作为整体存储时，很难利用并行计算的优势来并行加载文件。

架构演进

HDFS 1.0架构

我们一般把架构分成两种：

- 主从（master/slave）
- 对等（peer/peer）

HDFS是主从式（master/slave）结构，由NameNode和DataNode以及SecondaryNamenode组成。

Namenode：

属于集群中的中心服务器，管理节点，主要负责管理文件系统的命名空间(namespace)以及客户端对文件的访问，比如打开、关闭、重命名文件或目录，同时管理集群元数据的存储，记录文件中数据块（block）的映射关系。

Namenode中文件系统树及整棵树内所有的文件和目录，即元数据（MetaData）有三种存储方式：

- 内存元数据，目的是提升性能，定期从磁盘加载一份镜像到内存中。
- 命名空间镜像文件（fsImage），保存整个文件系统的目录树。
- 编辑日志文件（edits），记录文件系统元数据发生的所有更改，如文件的删除或添加等操作信息。

Datanode：

Datanode主要负责存储用户数据，处理来自文件系统客户端的请求，保持与Namenode的通信，执行Namenode的调度指令。

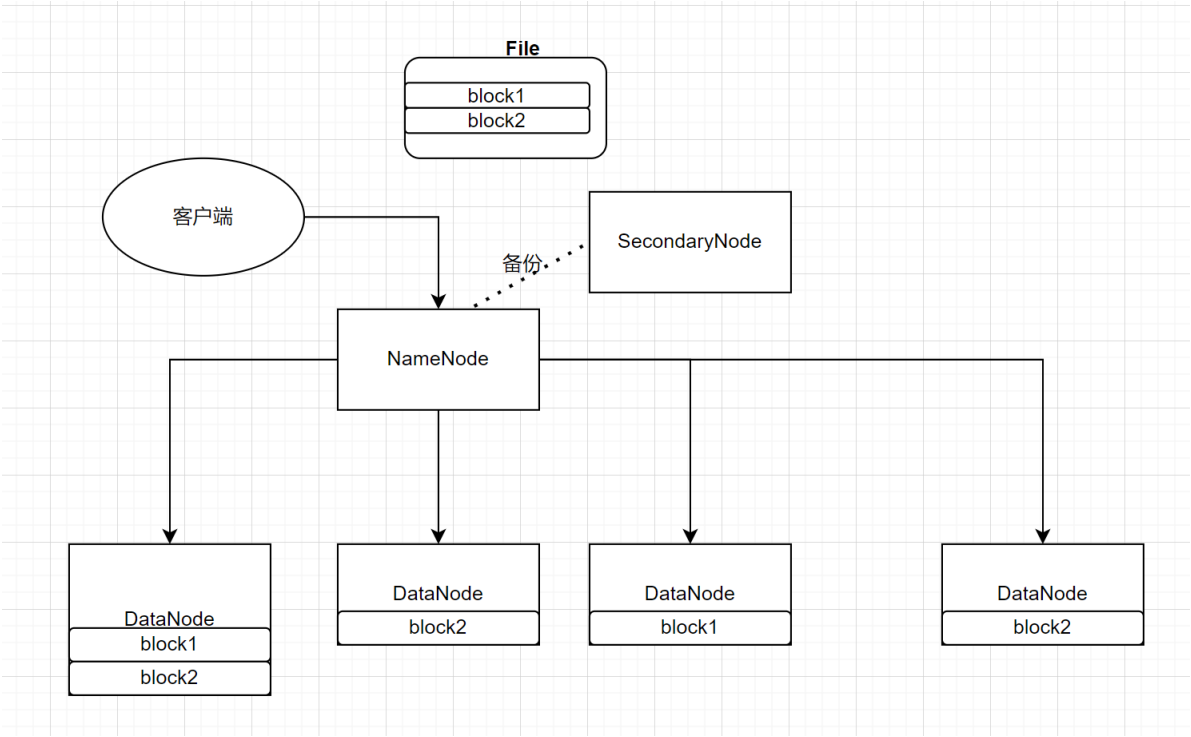
SeconddaryNameNode：

主要是合并NameNode的edit logs到fsimage文件。（属于NameNode的备份）

其数据合并的过程如下：

1. NameNode初始化时会产生一个edits文件和一个fsimage文件。
2. 随着edits文件不断增大，当达到设定的阀值时，Secondary NameNode把edits文件和fsImage文件复制到本地，同时NameNode会产生一个新的edits文件替换掉旧的edits文件，这样以保证数据不会出现冗余。
3. Secondary NameNode拿到这两个文件后，会在内存中进行合并成一个fsImage.ckpt的文件（这个过程称为checkpoint），合并完成后，再将fsImage.ckpt文件复制到NameNode下。
4. NameNode文件拿到fsImage.ckpt文件后，会将旧的fsimage文件替换掉，并且改名成fsimage文件。

这个架构如下图：



现里为了数据的安全性，HDFS进行了数据冗余，就是把一个 block 块增加了多个副本，其实分布式存储就是分区和冗余，这里叫分块和冗余。

但是HDFS1 有两个明显缺点：

1）NameNode 单点故障的问题

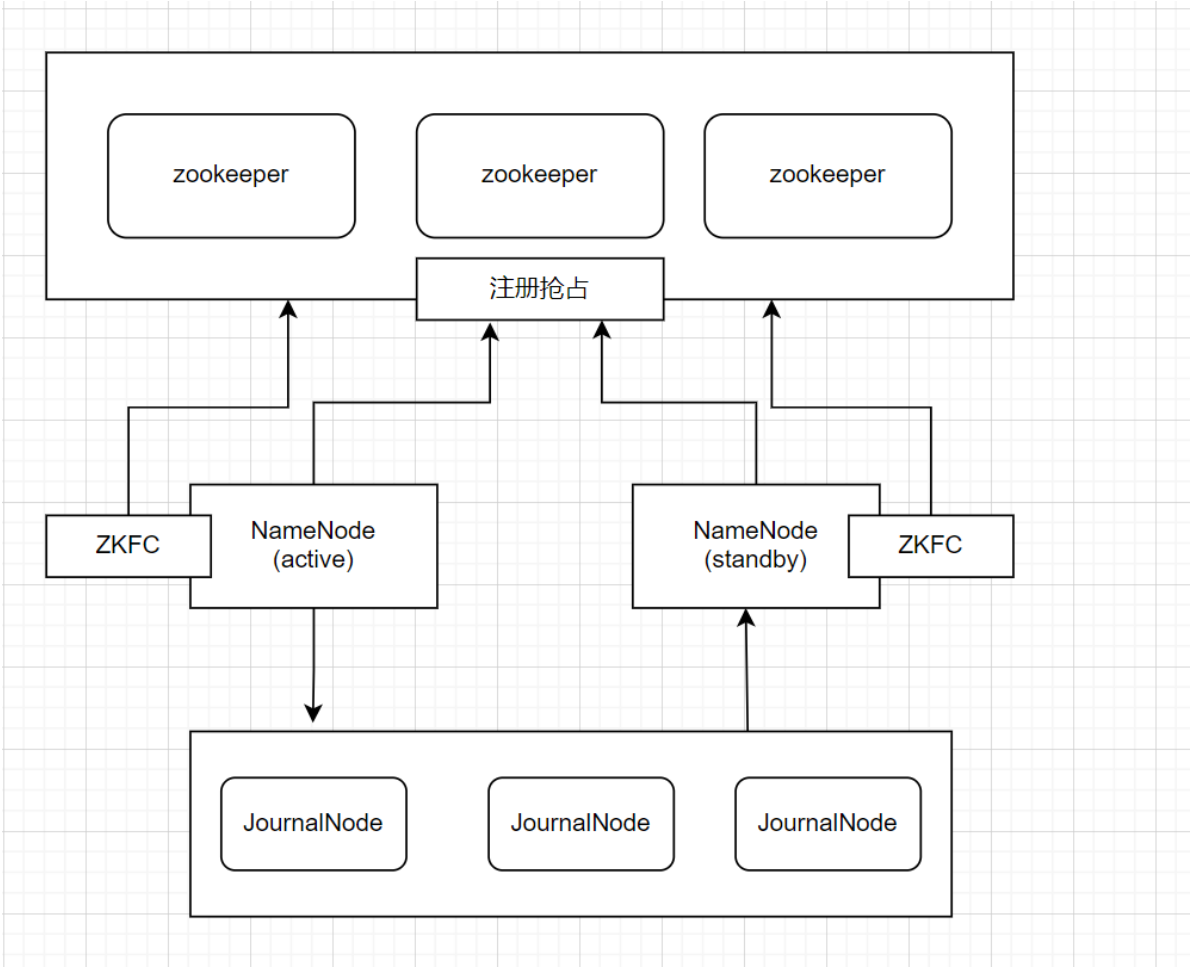
NameNode 这个主节点进程只有一个，如果这个进程宕机挂了，会导致 HDFS 不可用。

2）NameNode 内存受限的问题

当我们的元数据原来越多的时候，我们的 NameNode 单台服务器的内存会不够用。这个时候有人说，我们可以给 NameNode 加内存，但是你再加内存，元数据还是会越来越多。再说单台机器的内存再去增加是有上限的，所以这个方案不可取。

HDFS 2.0架构

HDFS 2.0采用HA（高可用）的方案，相对于HDFS 1.0来说，**Namenode会区分两种状态，active和standby**，正常工作的时候时候由active Namenode对外提供服务，standby Namenode则会从journalnode同步元数据，保证和active保持元数据一致，当active Namenode出现故障或者宕机的时候，standby会自动切换为新的active Namenode对外提供服务，并且HA对外提供了统一的访问名称，对于用户来说，不管访问的Namenode是active状态还是standby状态都是无感知的。

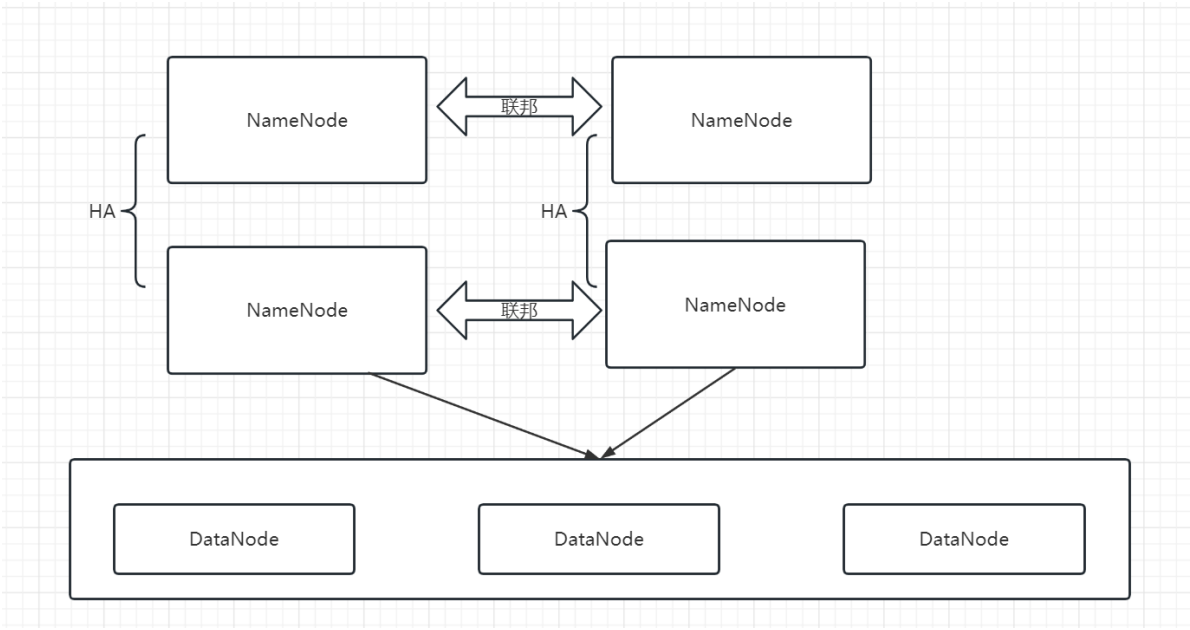


这里使用了 Zookeeper 来监控 NameNode 状态，当两个NameNode 启动的时候，它们俩都会去抢一把分布式锁，谁抢到了，谁的状态就是 Active ，另外一个状态就是 StandBy。

但是有些情Zookeeper 不一定能感受的到 NameNode 的，比如 NameNode 发生了 分钟级别的 FullGC，发生 FullGC 的时候，只有GC 线程去运行的，我们本地的工作线程是不运行的。

所以同时引入ZKFC 监控NameNode ，其会把 NameNode 的监控状态报告给 Zookeeper （与 Zookeeper 保持心跳）。

如此当状态为 Active 的 NameNode 宕机之后，状态为 StandBy 的 NameNode 可以自动顶上去，也就实现了故障的自动切换。



HDFS 2.0通过**federation（联邦）**机制解决了内存受限的问题。一个集群的内存是有限，当内存不足的时候我们要加内存，加机器，可是一个集群之间的通信需要成本，管理节点能力也是有限的。因此HDFS 2.0采用了建仓库的办法，也就是federation机制。

这里要注意两点：

- 上图中互为联邦的两个 NameNode 它们存储的数据是不相同的，因为这是当一台 NameNode 水平扩展的。
- 互为 HA 的两个 NameNode 所存储的数据是相同的，因为这是为了解决 NameNode 单点故障的问题。

HDFS 3.0架构

HDFS 3.0在架构上相对于HDFS 2.0没什么大的调整，HDFS 2.0只支持至多两个Namenode，而HDFS 3.0在2.0的基础上增加了多个Namenode的支持，提供集群可用性。主要聚焦提升底层数据存储优化，降低数据开销的成本，采用纠错码技术提高集群的容错性。

EC纠删码技术(EC: Erasure Coding)

Erasure coding纠删码技术简称EC，是一种数据保护技术。最早用于通信行业中数据传输中的数据恢复，是一种编码容错技术。他通过在原始数据中加入新的校验数据,使得各个部分的数据产生关联性。**在一定范围的数据出错情况下,通过纠删码技术都可以进行恢复。**

即EC可以将n份原始数据，增加m份数据，并能通过n+m份中的任意n份数据，还原为原始数据。如果把n+m份数据分布在存储系统的不同节点上，那么任意小于等于m个节点故障（m份数据失效），都可以通过其他剩余的数据还原出原始数据，从而达到不影响业务的目的。

目前，纠删码技术在分布式存储系统中的应用主要有三类，阵列纠删码（Array Code: RAID5、RAID6等）、RS(Reed-Solomon)里德-所罗门类纠删码和LDPC(LowDensity Parity Check Code)低密度奇偶校验纠删码。

EC技术可以防止数据丢失，又可以解决HDFS存储空间翻倍的问题。创建文件时，将从最近的祖先目录继承EC策略，以确定其块如何存储。与备份复制相比，默认的EC策略可以节省50%的存储空间，同时还可以承受更多的存储故障。