一

这三者的本质差别是使用数据的“用户”不同：块存储的用户是可以读写块设备的软件系统，例如传统的文件系统、数据库；文件存储的用户是自然人；对象存储的用户则是其它计算机软件。

传统的文件系统，是直接访问存储数据的硬件介质的。介质不关心也无法去关心这些数据的组织方式以及结构，因此用的是最简单粗暴的组织方式：所有数据按照固定的大小分块，每一块赋予一个用于寻址的编号。以大家比较熟悉的机械硬盘为例，一块就是一个扇区，老式硬盘是512字节大小，新硬盘是4K字节大小。老式硬盘用柱面-磁头-扇区号（CHS，Cylinder-Head-Sector）组成的编号进行寻址，现代硬盘用一个逻辑块编号寻址（LBA，Logical Block Addressing）。所以，硬盘往往又叫块设备（Block Device）,当然，除了硬盘还有其它块设备，例如不同规格的软盘，各种规格的光盘，磁带等。至于哪些块组成一个文件，哪些块记录的是目录/子目录信息，这是文件系统的事情。不同的文件系统有不同的组织结构，这个就不展开了。为了方便管理，硬盘这样的块设备通常可以划分为多个逻辑块设备，也就是我们熟悉的硬盘分区（Partition）。反过来，单个介质的容量、性能有限，可以通过某些技术手段把多个物理块设备组合成一个逻辑块设备，例如各种级别的RAID，JBOD，某些操作系统的卷管理系统（Volume Manager）如Windows的动态磁盘、Linux的LVM等。

对象存储其实介于块存储和文件存储之间。文件存储的树状结构以及路径访问方式虽然方便人类理解、记忆和访问，但计算机需要把路径进行分解，然后逐级向下查找，最后才能查找到需要的文件，对于应用程序来说既没必要，也很浪费性能。而块存储是排它的，服务器上的某个逻辑块被一台客户端挂载后，其它客户端就无法访问上面的数据了。而且挂载了块存储的客户端上的一个程序要访问里面的数据，不算类似数据库直接访问裸设备这种方式外，通常也需要对其进行分区、安装文件系统后才能使用。除了在网络上传输的数据包效率更高以外，并不比使用文件存储好多少，客户端的文件系统依然需要对路径分解，然后逐级查找才能定位到某一个具体的文件。

二

1. 客户端读取HDFS数据的流程

client访问NameNode，查询元数据信息，获得这个文件的数据块位置列表，返回输入流对象。
就近挑选一台datanode服务器，请求建立输入流 。
DataNode向输入流中中写数据，以packet为单位来校验。
关闭输入流

2．写数据

客户端向NameNode发出写文件请求。
检查是否已存在文件、检查权限。若通过检查，直接先将操作写入EditLog，并返回输出流对象。
（注：WAL，write ahead log，先写Log，再写内存，因为EditLog记录的是最新的HDFS客户端执行所有的写操作。如果后续真实写操作失败了，由于在真实写操作之前，操作就被写入EditLog中了，故EditLog中仍会有记录，我们不用担心后续client读不到相应的数据块，因为在第5步中DataNode收到块后会有一返回确认信息，若没写成功，发送端没收到确认信息，会一直重试，直到成功）
client端按128MB的块切分文件。
client将NameNode返回的分配的可写的DataNode列表和Data数据一同发送给最近的第一个DataNode节点，此后client端和NameNode分配的多个DataNode构成pipeline管道，client端向输出流对象中写数据。client每向第一个DataNode写入一个packet，这个packet便会直接在pipeline里传给第二个、第三个…DataNode。
（注：并不是写好一个块或一整个文件后才向后分发）
每个DataNode写完一个块后，会返回确认信息。
（注：并不是每写完一个packet后就返回确认信息，个人觉得因为packet中的每个chunk都携带校验信息，没必要每写一个就汇报一下，这样效率太慢。正确的做法是写完一个block块后，对校验信息进行汇总分析，就能得出是否有块写错的情况发生）
写完数据，关闭输输出流。
发送完成信号给NameNode。
（注：发送完成信号的时机取决于集群是强一致性还是最终一致性，强一致性则需要所有DataNode写完后才向NameNode汇报。最终一致性则其中任意一个DataNode写完后就能单独向NameNode汇报，HDFS一般情况下都是强调强一致性）

3．HDFS如何选择物理节点

1）当没有配置机架信息时，所有的机器Hadoop都默认在同一个默认的机架下，以名为”/default-rack”，这种情况下，任何一台datanode机器，不管物理上是否属于同一个机架，都会被认为是在同一个机架下。

2）一旦配置topology.script.file.name，就按照网络拓扑结构来寻找datanode。topology.script.file.name这个配置选项的value指定为一个可执行程序，通常为一个脚本。

4．HDFS如何应对数据丢失

增加副本数，提高副本恢复的速度

1）冗余备份

每个文件存储成一系列数据块（Block）。为了容错，文件的所有数据块都会有副本（副本数量即复制因子，课配置）（dfs.replication）

2）副本存放

采用机架感知（Rak-aware）的策略来改进数据的可靠性、高可用和网络带宽的利用率

3）心跳检测

NameNode周期性地从集群中的每一个DataNode接受心跳包和块报告，收到心跳包说明该DataNode工作正常

4）安全模式

系统启动时，NameNode会进入一个安全模式。此时不会出现数据块的写操作。

5）数据完整性检测

HDFS客户端软件实现了对HDFS文件内容的校验和（Checksum）检查（dfs.bytes-per-checksum）。

4．HDFS如何应对主节点失效

1）启动一个拥有文件系统元数据的新NameNode（这个一般不采用，因为复制元数据非常耗时间）

2）配置一对活动-备用（Active-Sandby）NameNode，活动NameNode失效时，备用NameNode立即接管，用户不会有明显中断感觉。

共享编辑日志文件（借助NFS、zookeeper等）

DataNode同时向两个NameNode汇报数据块信息

客户端采用特定机制处理 NameNode失效问题，该机制对用户透明

4．数据块—物理节点表是否需要备份，为什么