分布式文件系统
——HDFS

Hadoop组成

Hadoop=HDFS + MapReduce + Yarn

HDFS——分布式文件存储（充分的利用磁盘）

MapReduce——分布式计算（充分的使用CPU）

Yarn——负责集群资源的统一管理和调度

HDFS——分布式文件存储

要实现大数据的存储，需要使用几十台、几百台甚至更多的分布式服务器节点。为了统一管理这些节点上存储的数据，必须要使用一种特殊的文件系统——分布式文件系统。为了提供可扩展的大数据存储能力，Hadoop设计提供了一个分布式文件系统HDFS（Hadoop Distributed File System）。

HDFS基本特征

（1）大规模数据分布存储能力

HDFS可以存储GB级到TB级别大小的单个文件，还可以支持在一个文件系统中存储高达数千万量级的文件数量。这种分布式的文件系统使应用程序在访问这些数据的时候完全察觉不到数据在物理上使分布存储在一组不同机器上的。

HDFS基本特征

（2）高并发访问能力

HDFS通过多节点并发访问方式提供很高的数据访问带宽（高数据吞吐率），并且可以把带宽的大小等比例扩展到集群中的全部节点上。

HDFS基本特征

（3）强大的容错能力

大文件存储过程中避免硬件故障带来的数据损失是非常有必要的。所以HDFS在设计的过程中，通过采用多副本数据块形式存储（默认副本数目是3），按照块的方式随机选择存储节点，以快速从故障中得以恢复，并确保了数据不丢失。

HDFS基本特征

（4）顺序文件访问

大数据批处理在大多数情况下都是大量简单数据记录的顺序处理。针对这个特性，为了提高大规模数据访问的效率，HDFS对顺序读进行了优化，支持大量数据的快速顺序读出。

HDFS基本特征

（5）简单的一致性模型（一次写多次读）

HDFS采用了简单的“一次写多次读”模式访问文件，支持大量数据的一次写入、多次读取；不支持已写入数据的更新操作，但允许在文件尾部添加新的数据。

HDFS基本特征

（6）数据块存储模式

与常规的文件系统不同，HDFS采用给予大粒度数据块的方式存储文件，Hadoop2.0默认的块大小是128MB。好处是可以减少元数据的数量，并且可以允许将这些数据块通过随机方式选择节点，分布存储在不同的地方。

*补充：HDFS的数据块（block）存储机制

为了提高硬盘的效率，文件系统中最小的数据读写单位不是字节，而是一个更大的概念——数据块。但是，数据块的信息对于用户来说是透明的，除非通过特殊的工具，否则很难看到具体的数据块信息。

HDFS 同样也有数据块的概念。但是，与一般文件系统中大小为若干 KB 的数据块不同，HDFS 数据块的默认大小是 64MB，而且在不少实际部署中，HDFS 的数据块甚至会被设置成 128MB 甚至更多，比起文件系统上几个 KB 的数据块，大了几千倍。

将数据块设置成这么大的原因是减少寻址开销的时间。在 HDFS 中，当应用发起数据传输请求时，NameNode 会首先检索文件对应的数据块信息，找到数据块对应的 DataNode；DataNode 则根据数据块信息在自身的存储中寻找相应的文件，进而与应用程序之间交换数据。因为检索的过程都是单机运行，所以要增加数据块大小，这样就可以减少寻址的频度和时间开销。

HDFS基本架构

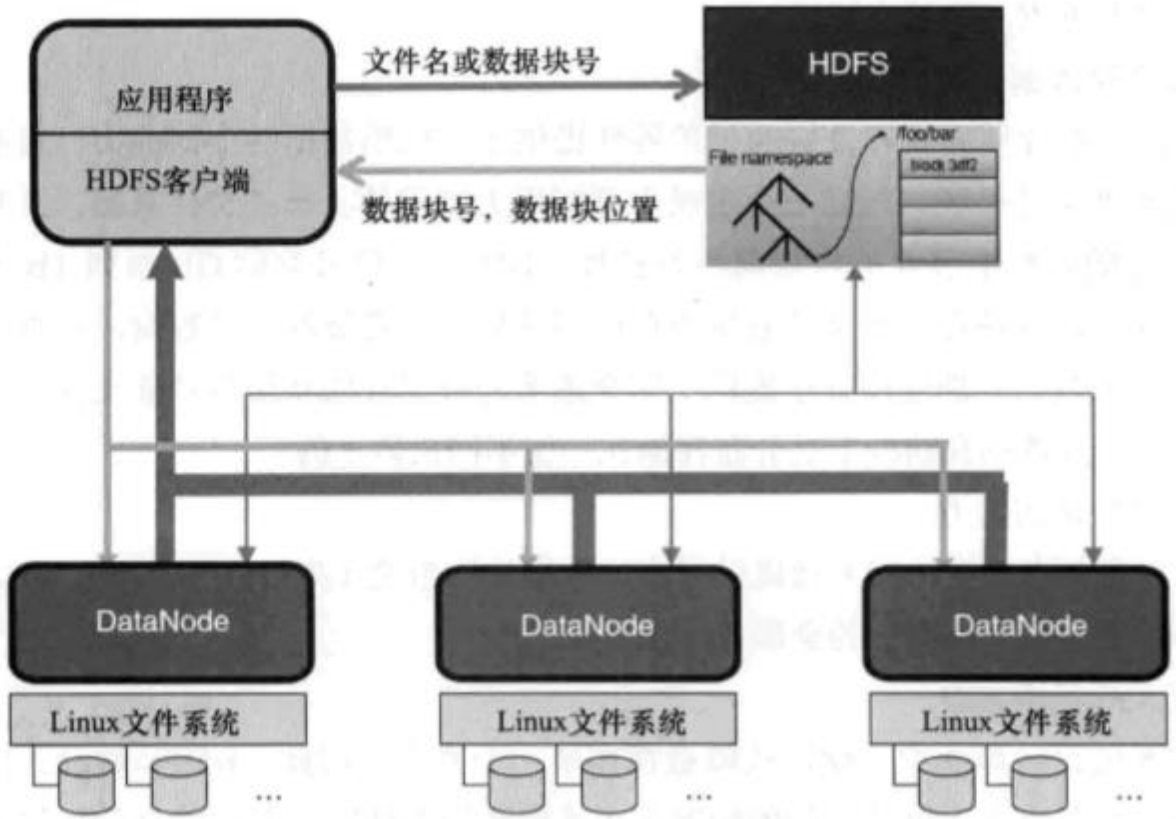

图 3-1　HDFS 的基本组成结构

HDFS基本架构

一个 HDFS 文件系统包括一个主控节点 NameNode 和一组 DataNode 从节点。NameNode 是一个主服务器，用来管理整个文件系统的命名空间和元数据，以及处理来自外界的文件访问请求。NameNode 保存了文件系统的三种元数据：1）命名空间，即整个分布式文件系统的目录结构；2）数据块与文件名的映射表；3）每个数据块副本的位置信息，每一个数据块默认有 3 个副本。

HDFS 对外提供了命名空间，让用户的数据可以存储在文件中，但是在内部，文件可能被分成若干个数据块。DataNode 用来实际存储和管理文件的数据块。文件中的每个数据块默认的大小为 64MB；同时为了防止数据丢失，每个数据块默认有 3 个副本，且 3 个副本会分别复制在不同的节点上，以避免一个节点失效造成一个数据块的彻底丢失。

每个 DataNode 的数据实际上是存储在每个节点的本地 Linux 文件系统中。

在 NameNode 上可以执行文件操作，比如打开、关闭、重命名等；而且 NameNode 也负责向 DataNode 分配数据块并建立数据块和 DataNode 的对应关系。DataNode 负责处理文件系统用户具体的数据读写请求，同时也可以处理 NameNode 对数据块的的创建、删除副本的指令。

HDFS基本架构

概括来讲：一个典型的HDFS部署情况是：NameNode程序单独运行于一台服务器节点上，其余的服务器节点，每一台运行一个DataNode程序。

HDFS文件访问过程

（1）首先，用户的应用程序通过HDFS的客户端程序将文件名发送至NameNode。

（2）NameNode接收到文件名之后，在HDFS目录中检索文件名对应的数据块，再根据数据块信息找到保存数据块的DataNode地址，将这些地址回送给客户端。

（3）客户端接收到这些DataNode地址之后，与这些DataNode并行地进行数据传输操作，同时将操作结果的相关日志（比如是否成功，修改后的数据块信息等）提交到NameNode。

*补充：通信协议

作为一个分布式文件系统，HDFS 中大部分的数据都是通过网络进行传输的。为了保证传输的可靠性，HDFS 采用 TCP 协议作为底层的支撑协议。应用可以向 NameNode 主动发起 TCP 连接。应用和 NameNode 交互的协议称为 Client 协议，NameNode 和 DataNode 交互的协议称为 DataNode 协议（这些协议的具体内容请参考其他资料）。而用户和 DataNode 的交互是通过发起远程过程调用（Remote Procedure Call，RPC）、并由 NameNode 响应来完成的。另外，NameNode 不会主动发起远程过程调用请求。

*补充：HDFS客户端（Client）（网页形式和命令行形式）

*网页形式：50070

*命令行形式：balabala

**文件的分块大小和存储的副本数量都由客户端决定（改变配置参数：切块大小dfs.blocksize副本数量dfs.replication）

***客户端在哪里运行没有约束，只要运行客户端的机器能够跟hdfs集群通信即可

HDFS数据读取过程

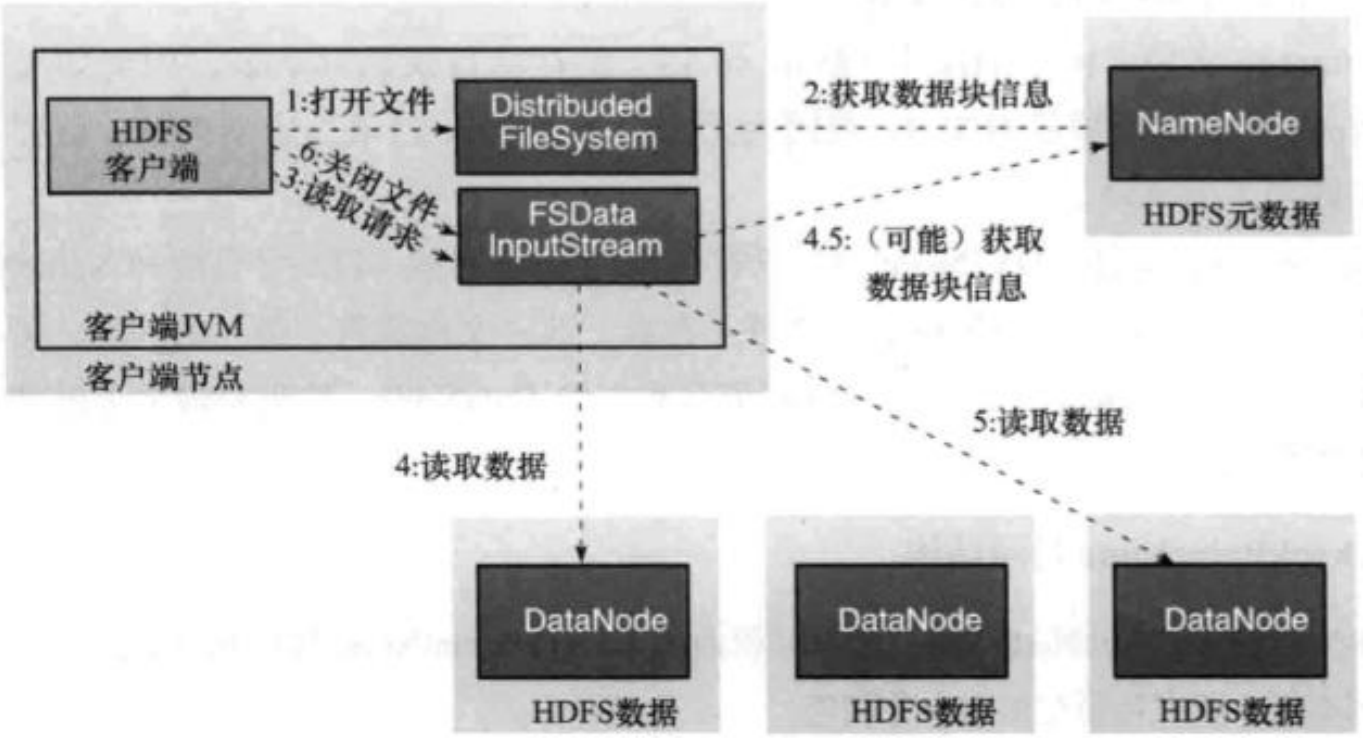

图 3-6　HDFS 数据读取过程

HDFS数据读取过程

以下是客户端读取数据的过程，其中 1、3、6 步由客户端发起：

客户端首先要获取 FileSystem 的一个实例，这里就是 HDFS 对应的实例。

1）首先，客户端调用 FileSystem 实例的 open 方法，获得这个文件对应的输入流，在 HDFS 中就是 DFSInputStream。

2）构造第 1 步中的输入流 DFSInputStream 时，通过 RPC 远程调用 NameNode 可以获得 NameNode 中此文件对应的数据块保存位置，包括这个文件的副本的保存位置（主要是各 DataNode 的地址）。注意，在输入流中会按照网络拓扑结构，根据与客户端距离对 DataNode 进行简单排序。

3 ~ 4）获得此输入流之后，客户端调用 read 方法读取数据。输入流 DFSInputStream 会根据前面的排序结果，选择最近的 DataNode 建立连接并读取数据。如果客户端和其中一个 DataNode 位于同一机器（比如 MapReduce 过程中的 mapper 和 reducer），那么就会直接从本地读取数据。

5）如果已到达数据块末端，那么关闭与这个 DataNode 的连接，然后重新查找下一个数据块。

不断执行第 2 ~ 5 步直到数据全部读完，然后调用 close。

6）客户端调用 close，关闭输入流 DFSInputStream。

另外，如果 DFSInputStream 和 DataNode 通信时出现错误，或者是数据校验出错，那么 DFSInputStream 就会重新选择 DataNode 传输数据。

HDFS数据写入过程

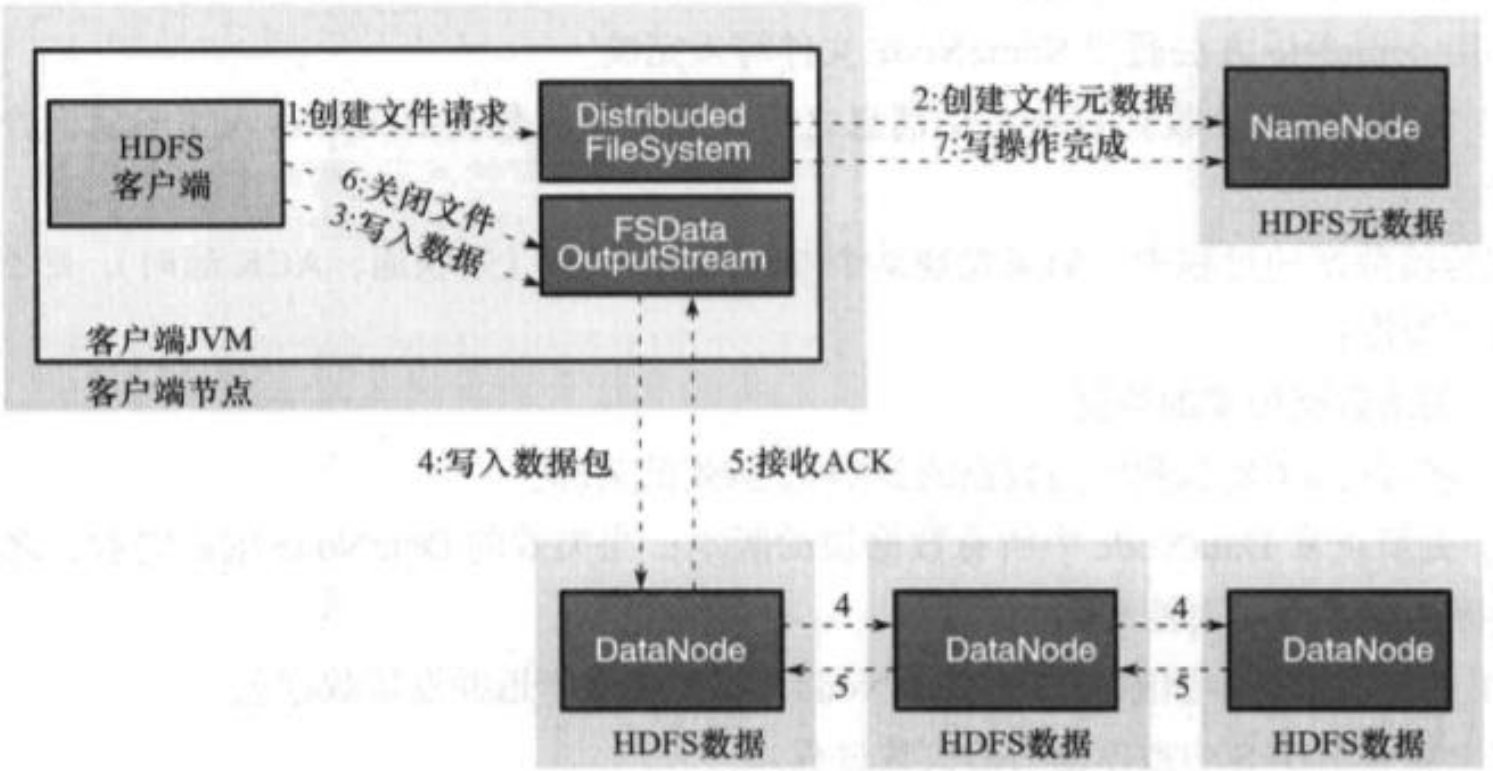

图 3-7　HDFS 数据写入过程

HDFS数据写入过程

以下是客户端写入数据的过程，其中 1、3、6 步由客户端发起：

客户端首先要获取 FileSystem 的一个实例，这里就是 HDFS 对应的实例。

1 ~ 2）客户端调用 FileSystem 实例的 create 方法，创建文件。NameNode 通过一些检查，比如文件是否存在，客户端是否拥有创建权限等；通过检查之后，在 NameNode 添加文件信息。注意，因为此时文件没有数据，所以 NameNode 上也没有文件数据块的信息。创建结束之后，HDFS 会返回一个输出流 DFSDataOutputStream 给客户端。

3）客户端调用输出流 DFSDataOutputStream 的 write 方法向 HDFS 中对应的文件写入数据。数据首先会被分包，这些分包会写入一个输出流的内部队列 Data 队列中，接收完数据分包，输出流 DFSDataOutputStream 会向 NameNode 申请保存文件和副本数据块的若干个 DataNode，这若干个 DataNode 会形成一个数据传输管道。

4）DFSDataOutputStream 会（根据网络拓扑结构排序）将数据传输给距离上最短的 DataNode，这个 DataNode 接收到数据包之后会传给下一个 DataNode。数据在各 DataNode 之间通过管道流动，而不是全部由输出流分发，这样可以减少传输开销。

5）因为各 DataNode 位于不同机器上，数据需要通过网络发送，所以，为了保证所有 DataNode 的数据都是准确的，接收到数据的 DataNode 要向发送者发送确认包（ACK

HDFS数据写入过程

Packet）。对于某个数据块，只有当DFSDataOutputStream收到了所有DataNode的正确ACK，才能确认传输结束。DFSDataOutputStream内部专门维护了一个等待ACK队列，这一队列保存已经进入管道传输数据、但是并未被完全确认的数据包。

不断执行第3 ~ 5步直到数据全部写完，客户端调用close关闭文件。

6）客户端调用close方法，DFSDataInputStream继续等待直到所有数据写入完毕并被确认，调用complete方法通知NameNode文件写入完成。

7）NameNode接收到complete消息之后，等待相应数量的副本写入完毕后，告知客户端即可。

在传输数据的过程中，如果发现某个DataNode失效（未联通，ACK超时），那么HDFS执行如下操作：

1）关闭数据传输的管道。

2）将等待ACK队列中的数据放到Data队列的头部。

3）更新正常DataNode中所有数据块的版本；当失效的DataNode重启之后，之前的数据块会因为版本不对而被清除。

4）在传输管道中删除失效的DataNode，重新建立管道并发送数据包。

以上就是HDFS中数据读写的大致过程。