# 第三章 Hadoop分布式文件系统

## 3.1 HDFS设计

HDFS以流式数据访问模式来存储超大文件。

流式数据访问：HDFS的构建思路是这样的，**一次写入，多次读写**是最高效的访问模式。

记住HDFS是为**高数据吞吐量应用优化**的，这可能以提高实践延迟为代价。对低延迟的访问需求，Hbase是更好的选择。

由于namenode是将文件系统的元数据储存在内存中，因此文件系统存储的文件总数受限于namenode的内存。根据经验一个文件，目录和数据块的存储信息大约占150字节。

HDFS不支持多个写入者的操作，也不支持文件在任意位置进行修改。

## 3.2 HDFS的概念

### 3.2.1 数据块

HDFS的数据块默认是64MB。HDFS中小于一个块大小的文件不会占据整个块的空间。、

为什么HDFS的块如此之大？目的是为了最小化寻址开销。

对于分布式文件系统的块抽象的好处：

1. 文件的大小可以大于网络上任何一个磁盘的大小，文件的所有块不一定要存储在同一个磁盘上。
2. 简化 了存储子系统的设计

HDFS的fsck指令可以显示块信息。

### 3.2.2 namenode 和datanode

如果运行namenode服务的机器被毁，文件系统上的所有文件将会丢失，因为我们不知道根据datanode的块重构文件。**对于namenode的容错性很重要；**

1. 备份那些组成文件系统元数据持久状态的文件。Hadoop可以通过配置在多个文件系统保存元数据持久状态，这些写操作是实时同步，原子操作，一般配置是将持久状态写入本地磁盘的同时，写入一**个远程挂载的网络文件系统（NFS）**
2. **运行一个辅助namenode**，辅助namenode保存的状态总是滞后于主节点。一般把NFS的namenode元数据复制到辅助namenode并作为namenode运行。

### 3.2.3 联邦HDFS

对于一个拥有大量文件的超大集群来说，内存将成为限制系统横向扩展的瓶颈。在2.x版本引入的联邦HDFS允许系统**添加namenode**实现扩展。每个namenode管理文件系统命名空间的一个部分。

客户端挂载数据表将文件路径映射到namenode。

### 3.2.4 HDFS的高可用性

虽然在多个文件系统备份namenode的元数据和通过备份namenode创建监测点能防止数据丢失。但是依旧无法实现文件系统的高可用性。Namenode存在单点失效（SPOF）的问题。

如果一个namenode失效，系统管理员得启动存有数据副本的新的namenode。新的namenode直到满足以下情况才能响应服务：

1. 将命名空间的映像导入内存
2. 重做编辑日志
3. 接受足够多的来自datanode的数据映像并退出安全模式

系统恢复的时间太长了，会影响日常维护。

在Hadoop的2.X中针对上述问题在HDFS增加了对高可用性（HA）的支持。配置了一对活动备用的namenode。为了实现这个目标需要在架构上做如下修改：

1. namenode之间需要通过高可用的共享储存实现编辑日志的共享。
2. datanode需要向两个那么node发生数据块处理报告
3. 客户端需要使用特定机制处理namenode的失效问题。

在活动的namenode失效后，备用的namenode能够迅速在几十秒实现任务的接管。其中系统需要确定活动namenode是否真的失效了。

管理员可以手动发起故障转移

## 3.3命令行接口

HDFS有很多接口，但是命令行是最简单的。

文件系统的基本操作。书上是$ hadoop fs 我的使用经历是 $hdfs dfs

## 3.4 Hadoop 文件系统

Hadoop有一个抽象文件系统的概念，HDFS只是其中的一种实现，JAVA抽象类org.apache.hadoop.fs.FileSystem 定义了Hadoop的一个文件系统接口。

通过Java API可以调用所有Hadoop文件系统的交互操作，一个接口通常和HDFS一起使用，因为Hadoop其他文件系统一般都有访问基本文件系统的工具（对于FTP ，有FTP客户端，对于S3 有S3工具）；

1. HTTP

通过HTTP访问HDFS有两种方法：直接访问，HDFS后台进程直接服务来自客户端的请求；通过代理（一个对应多个）访问。

1. C语言

Hadoop提供了一个名为libhdfs的c语言库。

1. FUSE

用户空间文件系统（Filesystem in Userspace，FUSE）允许把按照用户空间实现的文件系统整合一个unix文件系统。

功能： 任何一个Hadoop文件系统可以作为一个标准文件系统进行挂载；还可以使用任何一种编程语言通过POSIX库来访问文件系统。

## JAVA 接口

在这一节要深入探索Hadoop的Filesystem类：它是与Hadoop某一个文件系统进行交互的API。

**在eclipse下寻找包含这个类的jar包，之后需要你指定这个jar包的源码包，就在这个jar包目录下的source目录下。**

### 从hadoop url读取数据

### 通过FileSystem API 读取数据

### 3.5.3写入数据

### 3.5.4 目录

### 3.5.5 查询文件系统

## 3.6 数据流

## 3.7 通过Flume 和Sqoop导入数据

Apache Flume 是一个将大规模流数据导入HDFS的工具。最典型的应用是从另一个系统中收集日志数据并实现在HDFS中的聚集操作以便用于后期的分析。

Apache Sqoop是为了将结构化存储设备批量导入HDFS设计的。例如关系型数据库。Sqoop的应用场景，是组织将白天生产的数据库中的数据在晚间导入Hive数据仓库中进行分析。

## 3.8 通过distcp并行复制

前面着重介绍了单线程访问的HDFS访问模型，但是为了提高性能，需要写一个程序要并行的处理文件。Hadoop有一个有用的distcp分布式复制程序。它典型的应用场景就是两个HDFS集群之间传输数据

$ hadoop distcp hdfs://namenode/foo hdfs://namenode2/bar

不去深究它的使用，遇到的时候查一下就行

Distcp是作为一个MapReduce作业来实现的，通过集群中并行的map来完成的。

在两个运行不同版本HDFS的集群上使用distcp复制数据并使用hdfs协议会失败的。因为两个版本的RPC不兼容。

使用 均衡器（balancer）进而改善集群中块分布的均匀程度。

## 3.9 Hadoop存档

Hadoop存档文件或HAR文件，是一个更高效的文件存档工具。它将文件存入HDFS块中，在减少namenode内存使用的同时，允许文件进行透明的访问，Hadoop存档文件可以用作MapReduce的输入。

### 3.9.1 使用hadoop存档工具

Hadoop存档是通过一个archive工具根据一组文件创建而来的，该存档工具运行一个MapReduce作业来并行处理所有输入文件。

如果要删除HAR文件，要使用递归格式删除，因为对基础文件系统来说，HAR文件系统是一个目录

### 3.9.2 不足

新建一个存档文件会创建原始文件的一个副本；

一旦创建，存档文件便不能修改