# 第 3 章
# HDFS分布式文件系统

学习目标

- 了解文件系统的分类，能够描述不同文件系统的特点。
- 掌握 HDFS 架构，能够描述 HDFS 架构的主要组件及其作用。
- 了解 HDFS 的特点，能够简述 HDFS 的特点。
- 掌握 HDFS 的文件读写流程，能够叙述 HDFS 读写文件的流程。
- 熟悉 HDFS 的健壮性，能够叙述 HDFS 心跳机制、副本机制和负载均衡等策略。
- 掌握 HDFS 的 Shell 操作，能够灵活运用 HDFS Shell 命令操作 HDFS。
- 掌握 HDFS 的 Java API 操作，能够灵活使用 Java API 编写的应用程序操作 HDFS。
- 熟悉 Federation 机制，能够描述 Federation 机制的结构、特点，并实现 Federation 机制。
- 了解 Erasure Coding，能够简述 Erasure Coding 节省存储空间的原理。

HDFS(Hadoop Distributed File System，Hadoop 分布式文件系统)是 Hadoop 体系中的重要组成部分，主要用于解决海量数据的文件存储问题，它是目前应用最广泛的分布式文件系统。本章将从文件系统(File System)的分类开始，带领读者学习 HDFS 的相关内容及常见操作。

## 3.1 文件系统的分类

在计算机中，文件系统是命名文件及放置文件的逻辑存储和恢复的系统，人们熟悉的 Windows、macOS 和 Linux 操作系统都有文件系统。常见的文件系统分为单机文件系统、网络文件系统和分布式文件系统。

### 1. 单机文件系统

单机文件系统是所有文件系统的基础，也是常用的一种文件系统，它通过单台计算机的本地磁盘存储文件，依靠操作系统提供的文件系统实现文件的存储和管理。不过随着互联网的兴起，数据对存储容量的要求越来越高，单机文件系统的缺点逐渐显现。

- 本地磁盘的存储容量很容易达到存储极限。
- 大量并发的读写操作导致读写性能较低。
- 一旦计算机宕机或者损坏，存储的文件将无法访问甚至丢失，存储可靠性低。

### 2. 网络文件系统

网络文件系统(Network File System，NFS)可以看作单机文件系统的网络抽象，其本质与单机文件系统相似，只不过网络文件系统可以通过网络共享文件，用户可以像访问本地磁盘

上的文件一样便捷地访问远端计算机的文件。网络文件系统的出现在一定程度上解决了单机文件系统存储容量的问题，这是因为用户可以将文件存储在网络文件系统和本地文件系统两个位置。但是，网络文件系统没有解决单机文件系统性能低、可靠性低的问题。

### 3. 分布式文件系统

分布式文件系统是指多台服务器组成一个集群，通过相互通信的方式将集群内的所有存储空间资源整合，并对外提供文件访问服务。

在分布式文件系统中，为了解决文件存储的瓶颈问题，可以通过增加服务器数量的方式扩大存储空间，以此提供大型数据文件所需的存储空间，可扩展性高。

虽然文件存储的瓶颈得以解决，但是面对大量文件的读写请求，如果不能解决读写性能低的瓶颈，还是无法解决单机文件系统和网络文件系统存在的问题，那么在分布式文件系统中，针对读写性能方面的问题是如何解决的呢？

考虑到单台服务器的存储容量有限，分布式文件系统在存储一个大文件时，会将其分成几个相对较小的文件，然后将这些相对较小的文件存储在分布式文件系统的不同服务器中。例如，有一个 30GB 的 1.txt 文件，这个文件可以均分成 3 份，分别存储在 3 台不同的服务器中，每台服务器存储的文件大小都是 10GB，如图 3-1 所示。

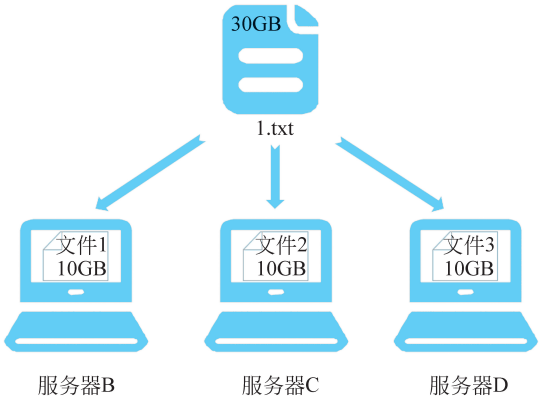


图 3-1　1.txt 文件存储在不同的服务器

图 3-1 展示的是服务器 B、服务器 C 和服务器 D 联合存储大小为 30GB 的 1.txt 文件。当读取 1.txt 文件时，会先从服务器 B、服务器 C 和服务器 D 中读取文件 1、文件 2 和文件 3，然后将这 3 个文件整合为 1.txt。这样做的好处是可以充分利用集群中不同服务器的资源处理单个文件，从而提高读写性能。

为了记录大文件切分后的信息，还需要一台服务器 A，专门用来记录文件被分成几份后的信息以及存储的位置信息，如图 3-2 所示。

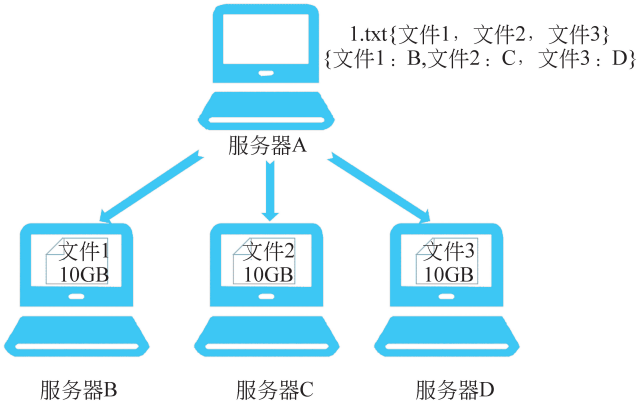


图 3-2　服务器 A 记录文件的切分信息及存储位置信息

在图 3-2 中，服务器 A 记录着文件被分成多少个文件和文件的存储位置。当客户端访问服务器 A 请求文件 1.txt 时，服务器 A 会按照类似查找目录的方式找到对应文件。

解决了文件存储瓶颈的问题后,其实还有一个关键问题需要处理,就是文件存储的可靠性。大文件分成小文件存储在不同的服务器上,如果某个服务器突然宕机,就会导致获取的文件不完整,也就是在单机文件系统和网络文件系统中存在的存储可靠性低的问题。针对这个问题,分布式文件系统的存储采用了副本机制,该机制通过保存多个副本提高了分布式文件系统的可靠性。

例如,分布式文件系统设置的副本数为 2,那么切分后的每个小文件都会有两份,并且存储在不同的服务器中,如图 3-3 所示。

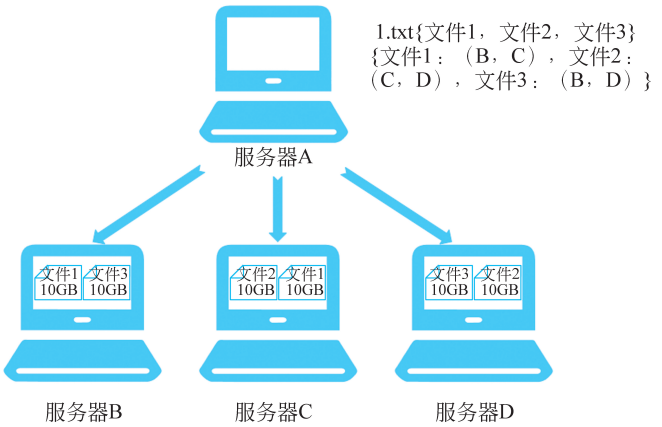


图 3-3 分布式文件系统的副本存储

在图 3-3 中,服务器 B 存储着文件 1 和文件 3,服务器 C 存储着文件 2 和文件 1,服务器 D 存储着文件 2 和文件 3。如果服务器 C 突然宕机,那么服务器 C 上存储的文件 1 和文件 2 同样可以从服务器 B 和服务器 D 中获取。分布式文件系统的副本机制极大地提高了文件存储的可靠性。

## 3.2 HDFS 简介

HDFS 源于 2003 年 10 月发表的论文 *The Google File System*,该论文描述了 Google 开发的一个产品框架,该框架称为 GFS(Google File System,Google 文件系统)。Nutch 开发人员借鉴 GFS 进行开源版本的实现,最终设计了一款类似于 GFS 的分布式文件系统 HDFS。本节将针对 HDFS 架构及其特点进行详细讲解。

### 3.2.1 HDFS 架构

HDFS 采用的是主从(Master/Slave)架构,即一个 HDFS 通常是由一个 Master 和多个 Slave 组成,其中,Master 在 HDFS 中扮演的角色为 NameNode,主要用于管理 HDFS。Salve 在 HDFS 中扮演的角色为 DataNode,主要用于存储文件。除此之外,HDFS 为了缓解单个 NameNode 的压力,还提供了一个 SecondaryNameNode,用于辅助 NameNode。接下来通过一张图详细介绍 HDFS 的架构,如图 3-4 所示。

#### 1. Block

Block 是 HDFS 文件系统中最小的存储单位,通常称为数据块。在 HDFS 文件系统中存

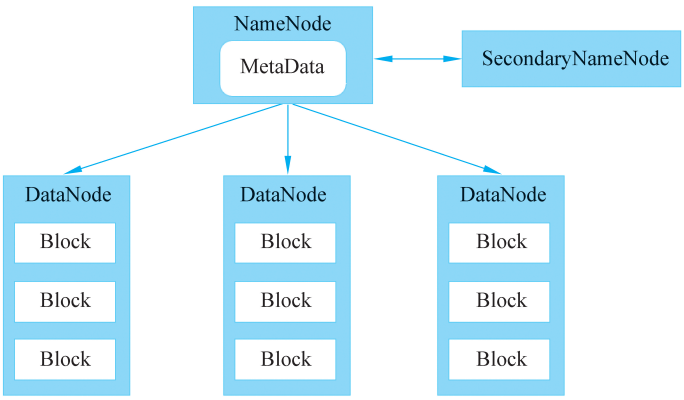


图 3-4　HDFS 架构

储的文件会被拆分成多个 Block，每个 Block 作为独立的单元进行存储，同一文件的多个 Block 通常存放在不同的 DataNode。在 Hadoop3.x 版本中，Block 的默认大小是 128MB。

需要注意的是，如果文件太小或者文件拆分后的 Block 没有达到 128MB，则 Block 的大小也会根据实际情况进行调整，例如，存储在 HDFS 文件系统的一个 300MB 文件如果拆分为 3 个 Block，那么这 3 个 Block 的大小分别为 128MB、128MB 和 44MB。

### 2. MetaData

MetaData 用于记录 HDFS 文件系统的相关信息，这些信息称为元数据，元数据的内容包括文件系统的目录结构、文件名、文件路径、文件大小、文件副本数、文件与 Block 的映射关系，以及 Block 与 DataNode 的映射关系等信息。

在 HDFS 文件系统中，为了确保元数据的快速访问，元数据会保存在内存中。同时，为了防止元数据丢失，会在本地磁盘生成 Fsimage 文件以备份元数据。当部署的 Hadoop 集群初次启动前进行格式化 HDFS 操作时，会在本地磁盘的指定目录生成一个 Fsimage 文件，启动 Hadoop 集群时，NameNode 会将 Fsimage 文件加载到内存中。

Hadoop 集群运行的过程中，用户频繁操作 HDFS 文件系统，内存中的元数据变化会非常快。内存中的元数据一旦更新，本地磁盘的 Fsimage 文件就会同步更新，这些操作非常消耗 NameNode 资源。但是，如果不更新本地磁盘的 Fsimage 文件，就会使内存和本地磁盘存储的元数据不一致。另外，一旦 NameNode 宕机，就会丢失元数据。

为了解决上述问题，HDFS 文件系统引入了 Edits 文件，该文件以追加的方式记录内存中元数据的每一次变化，这样，如果 NameNode 宕机，那么可以通过合并 Fsimage 文件和 Edits 文件的方式恢复内存中存储的元数据。需要注意的是，为了避免 Edits 文件过大，NameNode 会定期将 Fsimage 文件和 Edits 文件进行合并。

### 3. NameNode

NameNode 是 HDFS 集群的名称结点，通常称为主结点。如果 NameNode 由于故障原因而无法使用，那么用户就无法访问 HDFS。也就是说，NameNode 作为 HDFS 的主结点，起着至关重要的作用。

NameNode 的主要功能如下。

- 管理文件系统的命名空间(NameSpace)，例如打开文件、删除文件、重命名文件、创建目

录等。

- 处理客户端对文件的读写请求。
- 维护 HDFS 的元数据。
- 维护和管理 DataNode,并协调 DataNode 为客户端发起的读写请求提供服务。

4. DataNode

DataNode 是 HDFS 集群中的数据结点,通常称为从结点,其主要功能如下。

- 存储 Block。
- 根据 NameNode 的指令对 Block 进行创建、复制、删除等操作。
- 定期向 NameNode 汇报自身存储的 Block 列表及健康状态。
- 负责为客户端发起的读写请求提供服务。

5. SecondaryNameNode

SecondaryNameNode 是 HDFS 集群中的辅助结点,它可以定期从 NameNode 复制 Fsimage 文件及合并 Edits 文件,并将合并结果发送给 NameNode,从而辅助 NameNode 合并 Fsimage 文件和 Edits 文件,减轻 NameNode 的工作压力。与此同时,由于 SecondaryNameNode 和 NameNode 保存的 Fsimage 和 Edits 文件相同,因此它可以作为 NameNode 的冷备份,即当 NameNode 无法使用时,可以通过获取 SecondaryNameNode 中保存的 Fsimage 和 Edits 文件恢复 NameNode 的数据和运行状态。

### 3.2.2 HDFS 的特点

HDFS 是应用最广泛的分布式文件系统,它的特点具体如下。

1. 存储大文件

HDFS 支持 GB 级别甚至 TB 级别的文件存储,每个文件都会被切分为多个 Block 存储在集群的不同 DataNode 结点,使得对大型文件进行读写操作时可以采用并行的方式提高吞吐量。

2. 高容错性

HDFS 的副本机制会为 DataNode 中的每个 Block 创建多个副本,这样,即使某个 DataNode 宕机,也不会造成数据丢失,这是因为 HDFS 可以从保存 Block 副本的其他 DataNode 中读取相同的 Block。

3. 简单的一致性模型

HDFS 采用的是一次写入、多次读取的文件访问模型。在 HDFS 中,一个文件经过创建、写入和关闭后,只能追加,不能修改,并且不支持并发多用户的写操作,这样可以有效保证数据的一致性。

4. 移动计算比移动数据更经济

如果应用请求的计算在其操作的数据附近执行,那么该应用的效率要高很多。在数据达到海量级别时更是如此,这样可以最大限度地减少网络阻塞,并增加系统的整体吞吐量。应用运行时,将计算迁移到更靠近数据所在的位置通常比将数据迁移到计算所在的位置更好。HDFS 为应用提供了接口,以使自己更靠近数据所在的位置。

5. 可移植性

HDFS 可以轻松地从一个平台移植到另一个平台,这有助于广泛应用 HDFS 作为大量应用程序的首选平台。

## 3.3　HDFS 的文件读写流程

通过对 HDFS 架构的学习，读者已经了解 HDFS 主要由 NameNode 和 DataNode 组成，那么它们是如何协作处理 HDFS 文件的读写请求的呢？接下来针对 HDFS 文件的读写流程进行详细介绍。

### 1. HDFS 写文件流程

客户端向 HDFS 中写文件的具体流程如图 3-5 所示。

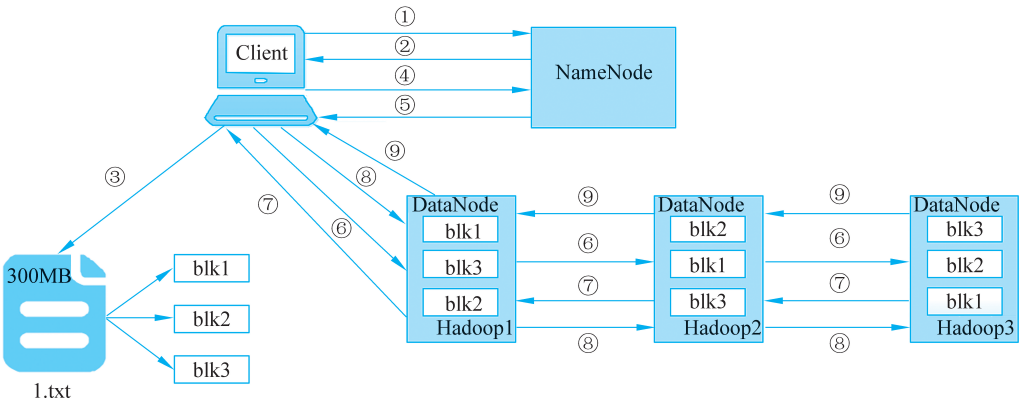


图 3-5　HDFS 写文件的具体流程

下面以 300MB 大小的 1.txt 文件为例，针对图 3-5 所示的 HDFS 写文件流程进行介绍。

（1）客户端(Client)上传 1.txt 文件到指定目录的请求，与 NameNode 建立通信。

（2）NameNode 检查客户端是否有上传文件的权限，以及文件是否存在。若通过检查，则通知客户端上传文件，并向其发送分块策略。

（3）客户端根据分块策略(以默认 128MB 为例)对文件 1.txt 进行切分，形成 3 个 Block，分别是 blk1、blk2 和 blk3。

（4）客户端向 NameNode 请求上传第一个 Block，即 blk1。

（5）NameNode 根据副本机制和机架感知向客户端返回可上传 blk1 的 DataNode 列表，这里指的是 Hadoop1、Hadoop2 和 Hadoop3。

（6）客户端从 NameNode 接收到 blk1 上传的 DataNode 列表之后，首先根据就近原则从 DataNode 列表中选择一台 DataNode(如 Hadoop1)并与之建立管道(Pipeline)，用于传输数据，然后 Hadoop1 会与第二台 DataNode(如 Hadoop2)建立管道，最后 Hadoop2 会与第三台 DataNode(如 Hadoop3)建立管道。

（7）首先，Hadoop3 向 Hadoop2 汇报管道建立成功；然后，Hadoop2 向 Hadoop1 汇报管道建立成功；最后，Hadoop1 向客户端汇报管道建立成功，此时客户端与所有 DataNode 列表中的所有 DataNode 都建立了管道。

（8）客户端开始传输 blk1，传输过程以流式写入的方式实现，具体过程如下。

① 将 blk1 写入内存进行缓存。

② 将 blk1 按照 packet(默认为 64KB)为单位进行划分。

③ 将第一个 packet 通过管道发送给 Hadoop1。

④ Hadoop1 接收第一个 packet 之后,客户端会将第二个 packet 发送给 Hadoop1,同时 Hadoop1 通过 Pipeline 将第一个 packet 发送给 Hadoop2。

⑤ Hadoop2 接收第一个 packet 之后,Hadoop1 会将第二个 packet 发送给 Hadoop2,同时 Hadoop2 通过 Pipeline 将第一个 packet 发送给 Hadoop3。

⑥ 以此类推,直至 blk1 上传完成。

(9) 首先,Hadoop3 向 Hadoop2 发送 blk1 写入完成的信息;然后,Hadoop2 向 Hadoop1 发送 blk1 写入完成的信息;最后,Hadoop1 向客户端发送 blk1 写入完成的信息。

客户端成功上传 blk1 后,重复第(4)~(9)的流程,依次上传 blk2 和 blk3,最终完成 1.txt 文件的上传。

### 2. HDFS 读文件流程

客户端从 HDFS 中读文件的流程如图 3-6 所示。

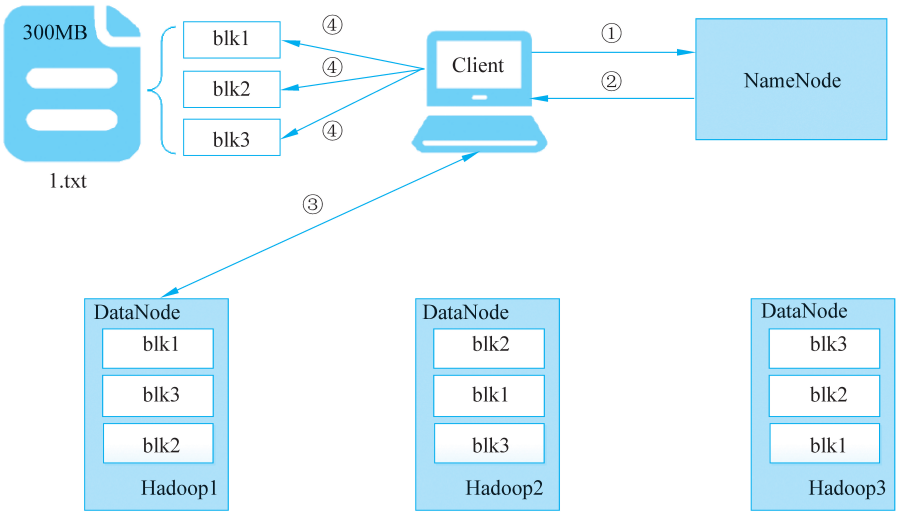


图 3-6 HDFS 读文件流程

下面以 300MB 大小的 1.txt 文件为例,针对图 3-6 所示的 HDFS 读文件流程进行介绍。

(1) 客户端发起读取 1.txt 文件的请求,与 NameNode 建立通信。

(2) NameNode 检查客户端是否有读取文件的权限,以及文件是否存在。若通过检查,则通知客户端读取文件,并向其发送 1.txt 文件的 Block 列表,该列表中记录了每个 Block 及其副本所在 DataNode 的地址,例如 Hadoop1、Hadoop2 和 Hadoop3 的主机地址。

(3) 客户端按照就近原则从 NameNode 返回的 Block 列表读取 Block。首先客户端从 Block 列表中选择距离自身最近的 DataNode(如 Hadoop1) 读取 Block,如果该 DataNode 可以读取文件的所有 Block,则结束读取;如果该 DataNode 没有读取文件的所有 Block,则客户端从 Block 列表中继续选择距离自身最近的 DataNode 读取剩余的 Block,直至获取文件的所有 Block 才结束读取。

(4) 客户端将读取的所有 Block 按照顺序进行合并,最终形成 1.txt 文件。需要注意的是,如果文件过大,导致 NameNode 无法一次性读取所有 Block,则会分批次将 Block 列表返回客户端。

## 3.4　HDFS 的健壮性

负责、敬业的精神和态度对于保证各行各业的稳定和发展具有重要意义。犹如 HDFS 的主要目标是在出现故障的情况下可靠地存储数据，其运用了心跳机制、副本机制、数据完整性校验、安全模式和快照 5 种策略保证存储数据的可靠性，关于这 5 种策略的具体介绍如下。

### 1. 心跳机制

HDFS 在运行期间，为了确保 NameNode 可以实时获取每个 DataNode 的健康状态，会在 NameNode 和每个 DataNode 之间建立一种心跳机制，即每个 DataNode 会根据固定间隔时间(默认为 3s)周期性地向 NameNode 发送心跳信息。如果 NameNode 在固定间隔时间内接收到 DataNode 发送的心跳信息，则认为该 DataNode 处于存活状态，否则认为该 DataNode 处于假死状态。不过，此时 NameNode 暂时还不会标注该 DataNode 处于宕机状态，如果在最大间隔时间(默认是 10min)内仍然没有接收到处于假死状态的 DataNode 发送的心跳信息，那么此时 NameNode 会认为该 DataNode 处于宕机状态。当某个 DataNode 处于宕机状态后，可能会造成某些 Block 的副本数无法达到系统要求，这时 NameNode 会启动复制操作，令其他处于存活状态的 DataNode 为这些 Block 创建新的副本。

### 2. 副本机制

副本机制可以确保存储在 DataNode 的每个 Block 都存在多个副本，默认的副本数为 3(包含自身)，即每个 Block 除自身外还存在 2 个副本，并且每个副本会分配到不同的 DataNode。若其中一台 DataNode 宕机而导致 Block 的副本数不足 3，HDFS 会为该 Block 创建新的副本，并将其存储到处于存活状态的 DataNode 中，以确保每个 Block 的副本数为 3。

### 3. 数据完整性校验

客户端从某个 DataNode 获取的 Block 有可能是已损坏的，损坏的原因可能是由于 Datanode 的存储设备错误或者网络错误造成的，针对这一问题，HDFS 的客户端实现了对文件内容进行校验和检查的功能。当客户端从 HDFS 读取文件时，会计算这个文件中每个 Block 的校验和(Checksum)，并将校验和作为一个单独的隐藏文件保存在 HDFS 中。当客户端获取文件内容后，它会校验从 DataNode 获取 Block 的校验和，并与保存为隐藏文件的校验和进行匹配，如果不匹配，客户端可以选择从其他 DataNode 中获取该文件的 Block。

### 4. 安全模式

HDFS 的安全模式是一种特殊状态，该状态下，客户端只能对 HDFS 读，不能写。HDFS 的安全模式分为两种情况，一种情况是 HDFS 启动时进入安全模式，此时 DataNode 会向 NameNode 报告自身的 Block 状态，当 Block 的副本数符合 HDFS 规定的副本数时，该 Block 的数据被认为是完整的，HDFS 会退出安全模式；另一种情况是 HDFS 中某个 Block 的副本丢失，HDFS 也会进入安全模式，此时 NameNode 会自动复制 Block 到 DataNode 上，当 Block 的副本数满足 HDFS 规定的副本数时，HDFS 会退出安全模式。

### 5. 快照

快照是 HDFS 整个文件系统或某个目录在某个时间点的镜像。当用户因错误操作导致 HDFS 中的数据缺失时，可以利用快照恢复缺失数据，使缺失数据的 HDFS 恢复至已知的良好时间点。

# 3.5 HDFS 的 Shell 操作

## 3.5.1 HDFS Shell 介绍

HDFS Shell 类似于 Linux 操作系统中的 Shell,是一种命令语言,可以对 HDFS 上的文件和目录进行一系列操作。

HDFS Shell 的语法格式如下。

```
hdfs [OPTIONS] SUBCOMMAND [SUBCOMMAND OPTIONS]
```

上述语法格式中,各参数的介绍具体如下。

- OPTIONS:可选参数,用来调试 Hadoop。例如关闭 HDFS 或 YARN 集群的某个服务、启动 HDFS 或 YARN 集群的某个服务、指定 Hadoop 配置文件所在目录等。
- SUBCOMMAND:表示 HDFS Shell 的子命令,用于操作 HDFS。根据操作类型的不同,HDFS Shell 提供了 3 种类型的子命令,分别是管理员命令(Admin Commands)、客户端命令(Client Commands)和进程命令(Daemon Commands)。其中,管理员命令主要用于管理员调试 HDFS、对 HDFS 进行加密和配置 HDFS 的缓存等;客户端命令主要用于操作人员查看 HDFS 版本信息和对 HDFS 文件进行操作等;进程命令主要用于操作人员启动 HDFS 集群的指定服务。
- SUBCOMMAND OPTIONS:表示 HDFS Shell 子命令的选项。例如,通过 Client Commands 提供的子命令选项"-mkdir"可以在 HDFS 中创建目录。

在 HDFS 集群的日常使用过程时,主要通过 Client Commands 类型的 HDFS Shell 子命令操作 HDFS。Hadoop 提供了多种 Client Commands 类型的 HDFS Shell 子命令,包括 dfs、envvars、classpath 等,其中,dfs 主要用于操作 HDFS 的文件和目录,是最常用的 HDFS Shell 子命令。dfs 常用的子命令选项如表 3-1 所示。

表 3-1 dfs 常用的子命令选项

| 子命令选项 | 功 能 描 述 |
| --- | --- |
| -ls | 查看指定目录信息 |
| -du | 查看指定目录下每个文件和子目录的大小,子目录也可以看作单独的目录,因为它也可以存在子目录 |
| -mv | 移动指定文件或目录 |
| -cp | 复制指定文件或目录 |
| -rm | 删除指定文件或目录 |
| -put | 将本地文件系统中的指定文件上传到 HDFS 指定目录 |
| -cat | 查看指定文件的内容 |
| -help | 查看帮助文档 |
| -mkdir | 创建目录 |
| -get | 将 HDFS 的指定文件下载到本地文件系统 |

表 3-1 只介绍了 HDFS Shell 子命令 dfs 常用的子命令选项,如果需要了解 dfs 其他的子

命令选项，可以通过 dfs 提供的子命令选项“-help”进行查看。接下来对表 3-1 中的子命令选项进行讲解。

### 1. -ls

-ls 用于查看 HDFS 指定目录信息，语法格式如下。

```
hdfs dfs -ls [-S][-C][-r][-h][-R] <path>
```

上述语法格式中，各参数的介绍具体如下。

- 参数-S：可选，用于根据文件的大小，按照由大到小的顺序显示指定目录的内容。
- 参数-C：可选，用于显示指定目录下文件和子目录的路径，不显示关于文件和子目录的其他信息。
- 参数-r：可选，用于根据文件的大小，按照由小到大的顺序显示指定目录的内容。
- 参数-h：可选，用于将默认的文件大小(字节数)格式化为便于查看的格式进行显示，例如将 1024 格式化为 1MB。
- 参数-R：可选，用于递归显示指定目录及其子目录的信息。
- 参数 path：用于指定查看的目录。

假设 HDFS 存在目录/data，该目录下包含文件 dataA、dataB 和子目录/dataChild，子目录/dataChild中存在文件 dataC。接下来演示如何使用子命令选项-ls 查看 HDFS 指定目录信息。

(1) 查看目录/data 的信息，具体命令如下。

```
$ hdfs dfs -ls -S /data
```

上述命令的执行效果如图 3-7 所示。

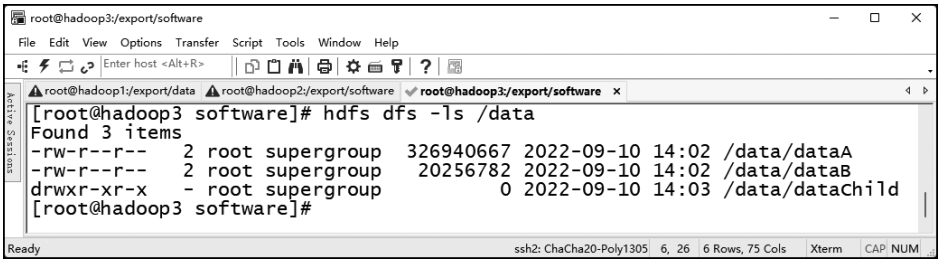


图 3-7　查看目录/data 的信息(1)

图 3-7 显示了 3 条信息，每条信息从左到右依次表示属性、副本数、所有者、所属组、大小(子目录大小为 0)、创建时间和路径。其中，属性包含 4 部分内容，以 drwxr-xr-x 为例，属性每部分的含义如图 3-8 所示。

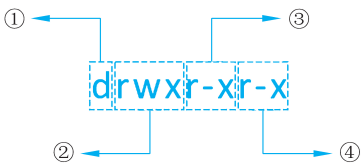


图 3-8　属性每部分的含义

在图 3-8 中，属性每部分含义的介绍如下。

① 用于标识文件或目录，目录使用 d 表示，文件使用“-”表示。

② 用于标识所有者的操作权限，包括读(r)、写(w)和执行(x)3 种权限，这 3 部分的位置固定，如不存在读、写或执行权限，则对应位置使用字符“-”代替。图 3-8 展示的所有者操作权限是读、写和执行。

③ 用于标识用户所属组的其他用户的操作权限，图 3-8 展示的用户所属组的其他用户的

操作权限为读和执行。

④ 用于标识其他组用户的操作权限,图 3-8 展示的其他组用户的操作权限为读和执行。

(2) 根据文件的大小,按照由小到大的顺序显示目录/data 的内容,并将默认的文件大小格式化为便于查看的格式进行显示,具体命令如下。

```
$ hdfs dfs -ls -r -h /data
```

上述命令的执行效果如图 3-9 所示。

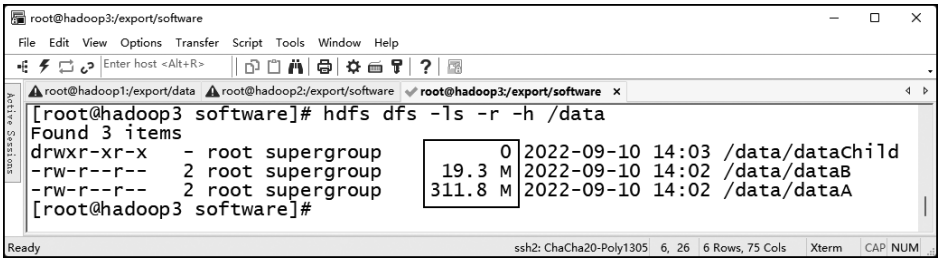


图 3-9 查看目录/data 的信息(2)

从图 3-9 可以看出,文件 dataA 和 dataB 的大小由字节数格式化为便于查看的格式,并且文件按照由小到大的顺序显示。

(3) 递归显示目录/data 及其子目录的信息,信息中仅显示文件和子目录的路径,具体命令如下。

```
$ hdfs dfs -ls -R -C /data
```

上述命令的执行效果如图 3-10 所示。

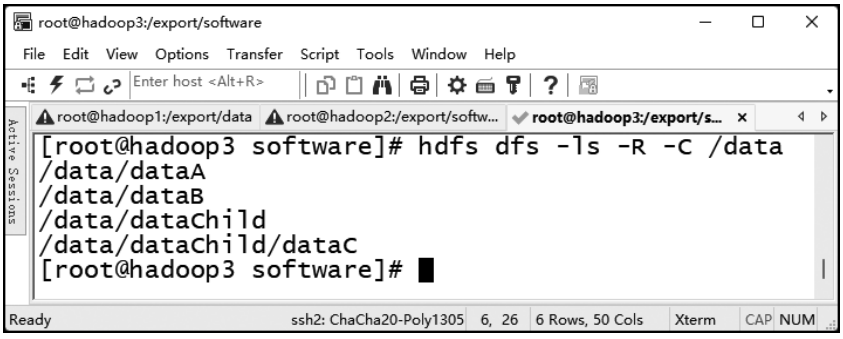


图 3-10 查看目录/data 的信息(3)

从图 3-10 可以看出,不仅显示了目录/data 的信息,而且显示了目录/data/dataChild 的信息。

### 2. -mkdir

-mkdir 用于创建目录,语法格式如下。

```
hdfs dfs -mkdir [-p] <path>
```

上述语法格式中,参数-p 为可选,它有两个作用,第一是创建目录,如果要创建的目录存在,则不会返回错误信息,也不会重新创建;第二是递归创建目录及其子目录。

接下来,在 HDFS 的目录/data 中创建子目录/dataChild1,并在子目录/dataChild1 中创

建子目录/dataChild2，具体命令如下。

```
$ hdfs dfs -mkdir -p /data/dataChild1/dataChild2
```

上述命令执行完成后，执行“hdfs dfs -ls -R /data”命令递归显示目录/data 及其子目录的信息，如图 3-11 所示。

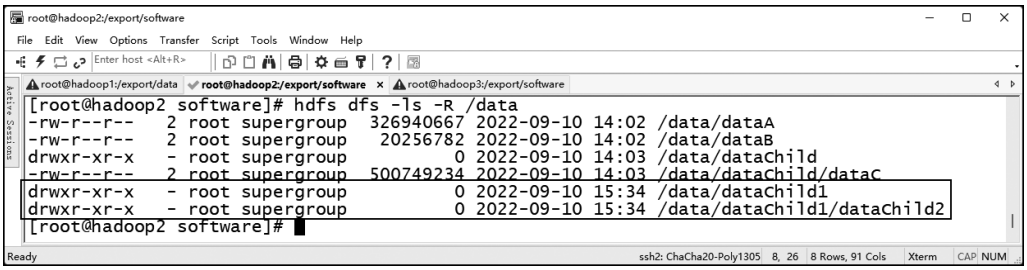


图 3-11　查看目录/data 的信息(4)

从图 3-11 可以看出，目录/data 存在子目录 dataChild1，并且该子目录同样存在子目录/dataChild2。

### 3. -du

-du 用于查看 HDFS 指定目录下每个文件和子目录的大小，语法格式如下。

```
hdfs dfs -du [-s] [-h] <path>
```

上述语法格式中，参数-s 为可选，用于查看指定目录下所有文件和子目录的总大小；参数-h 为可选，用于将默认的文件和子目录大小(字节数)格式化为便于查看的格式进行显示，例如将 1024 格式化为 1MB。

接下来演示如何使用子命令选项-du 查看 HDFS 的目录/data 中的每个文件和子目录的大小，并将默认的文件和子目录大小格式化为便于查看的格式进行显示，具体命令如下。

```
$ hdfs dfs -du -h /data
```

上述命令的执行效果如图 3-12 所示。

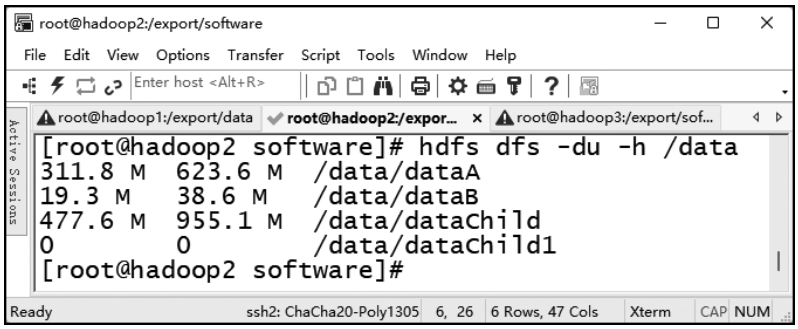


图 3-12　查看目录/data 中每个文件和子目录的大小

从图 3-12 可以看出，文件和子目录的大小显示了两部分，前者为文件或子目录中文件的实际大小，后者为文件及其副本的大小，以文件 dataA 为例，文件 dataA 的实际大小为 311.8MB。由于 HDFS 的副本数为 2，所以除文件 dataA 自身外，还存在一个副本，这两者大小之和便是 623.6MB。

4. -mv

-mv 用于移动 HDFS 指定目录或文件,语法格式如下。

```
hdfs dfs -mv <src> <dst>
```

上述语法格式中,参数 src 用于指定要移动的目录或文件。参数 dst 用于将目录或文件移动到指定的目录,如果指定的目录不存在,并且与移动的目录或文件处于同一路径下,那么会对文件或者目录进行重命名操作。需要注意的是,移动的目录或文件在指定的目录中不能存在。

接下来演示如何使用子命令选项-mv 移动 HDFS 的目录/data 中的文件和子目录。

(1) 将目录/data 中的子目录/dataChild1 移动到目录/data/dataChild 中,具体命令如下。

```
$ hdfs dfs -mv /data/dataChild1 /data/dataChild
```

上述命令执行完成后,执行“hdfs dfs -ls -R /data”命令递归显示目录/data 及其子目录的信息,如图 3-13 所示。

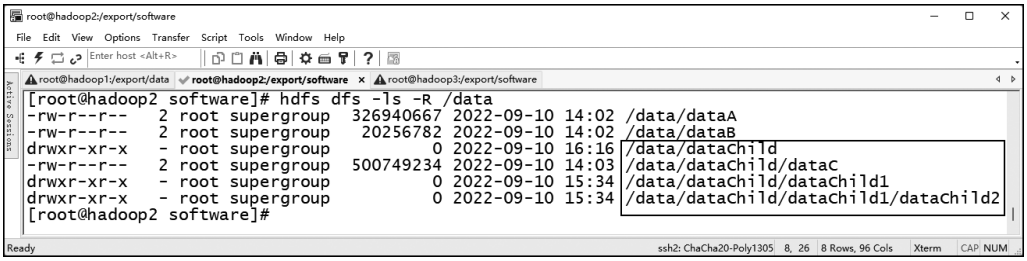


图 3-13 查看目录/data 的信息(5)

从图 3-13 可以看出,目录/data 的子目录/dataChild1 已经移动到目录/data/dataChild 中。

(2) 将目录/data 中的文件 dataA 重命名为 dataA_New,具体命令如下。

```
$ hdfs dfs -mv /data/dataA /data/dataA_New
```

上述命令执行完成后,执行“hdfs dfs -ls /data”命令查看目录/data 的信息,如图 3-14 所示。

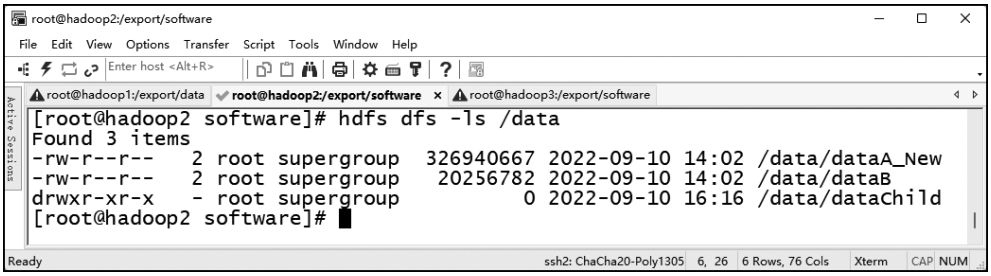


图 3-14 查看目录/data 的信息(6)

从图 3-14 可以看出,目录/data 下的文件 dataA 变为 dataA_New。

5. -cp

-cp 用于复制 HDFS 指定目录或文件,语法格式如下。

```
hdfs dfs -cp <src> <dst>
```

上述语法格式中，参数 src 用于指定要复制的目录或文件，可以同时复制多个文件或目录，每个文件或目录用空格进行分隔。参数 dst 用于将目录或文件复制到指定的目录，该目录必须已经存在，且要复制的文件或目录在指定的目录中不能存在。如果复制的是单文件或目录，则可以重命名复制后的文件或目录名称。

接下来演示如何使用子命令选项-cp 复制 HDFS 的目录/data 中的文件和子目录。

（1）将目录/data 下的文件 dataA_New 和 dataB 复制到目录/data/dataChild，具体命令如下。

```
$ hdfs dfs -cp /data/dataA_New /data/dataB /data/dataChild
```

上述命令执行完成后，执行“hdfs dfs -ls -R /data”命令递归显示目录/data 及其子目录的信息，如图 3-15 所示。

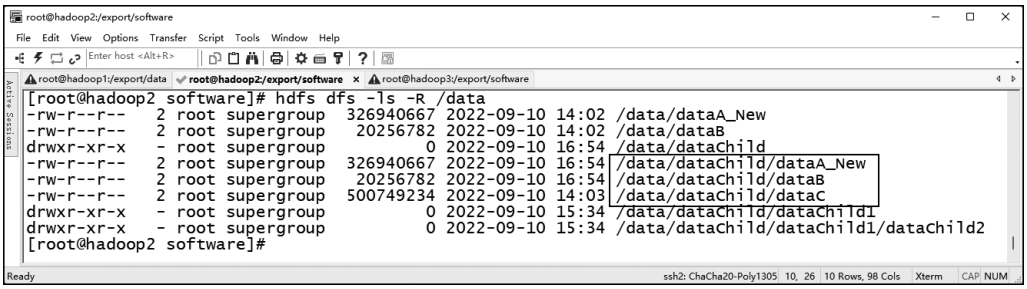


图 3-15　查看目录/data 的信息（7）

从图 3-15 可以看出，目录/data/dataChild 中已经存在文件 dataA_New 和 dataB。

（2）将目录/data 下的文件 dataA_New 复制到子目录/dataChild，并将其重命名为 dataA，具体命令如下。

```
$ hdfs dfs -cp /data/dataA_New /data/dataChild/dataA
```

上述命令执行完成后，执行“hdfs dfs -ls /data/dataChild”命令查看目录/data/dataChild 的信息，如图 3-16 所示。

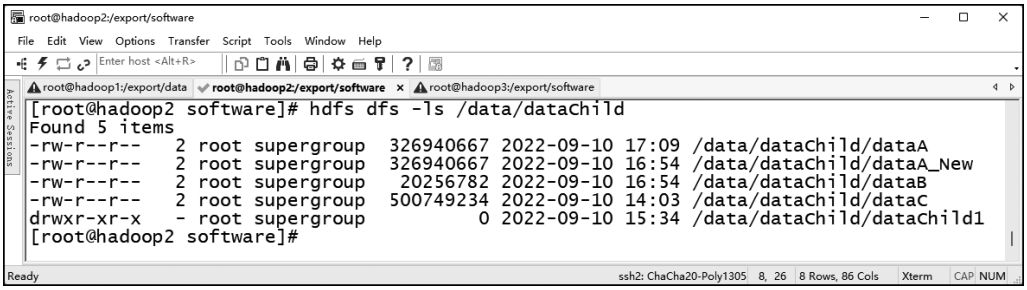


图 3-16　查看目录/data/dataChild 的信息

从图 3-16 可以看出，目录/data/dataChild 中存在文件 dataA。

### 6. -rm

-rm 用于删除 HDFS 指定目录或文件，语法格式如下。

```
hdfs dfs -rm [-f] [-r] [-skipTrash] [-safely] <src>
```

上述语法格式中,参数-f 为可选,用于判断删除的目录或文件是否存在;参数-r 为可选,用于递归删除指定目录中的所有子目录和文件;参数-skipTrash 为可选,表示删除的文件或目录不会放入回收站;参数-safely 为可选,用于启动安全确认,删除目录时会提示是否删除,以免误删。

接下来演示如何使用子命令选项-rm 删除目录/data 的子目录/dataChild,具体命令如下。

```
$ hdfs dfs -rm -r /data/dataChild
```

上述命令的执行效果如图 3-17 所示。


图 3-17 删除目录/data 的子目录/dataChild

从图 3-17 可以看出,目录/data 的子目录/dataChild 被删除并放入回收站了,同时显示了该目录在回收站的位置。

### 7. -put

-put 用于将本地文件系统中的指定文件上传到 HDFS 指定目录,语法格式如下。

```
hdfs dfs -put [-f] <localsrc> <dst>
```

上述语法格式中,参数-f 为可选,用于判断上传的文件在 HDFS 指定目录中是否存在,如果存在,则上传的文件会替换 HDFS 指定目录中已经存在的文件;参数 localsrc 用于指定本地文件系统中上传的文件,可以同时上传多个文件;参数 dst 用于指定上传到 HDFS 的目录,该目录必须存在。

假设本地文件系统的/export/data 目录下存在两个文件 a.txt 和 b.txt,其中,文件 a.txt 的内容为 Hello HDFS;文件 b.txt 的内容为 Hello Hadoop。接下来演示如何使用子命令选项-put 将本地文件系统中/export/data 目录下的文件 a.txt 和 b.txt 上传到 HDFS 的目录/data,具体命令如下。

```
$ hdfs dfs -put /export/data/a.txt /export/data/b.txt /data
```

上述命令执行完成后,执行“hdfs dfs -ls /data”命令查看目录/data 的信息,如图 3-18 所示。

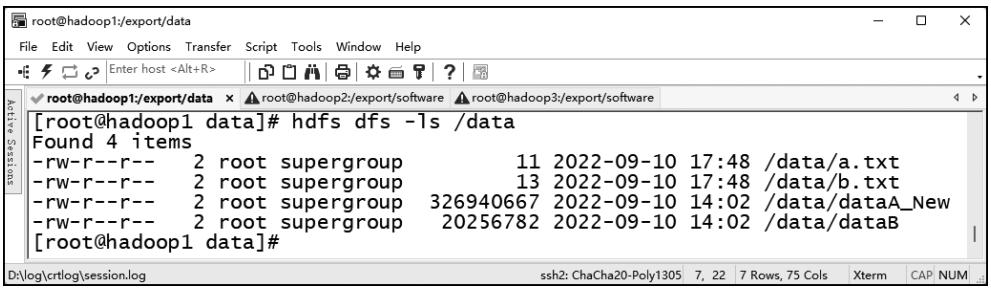


图 3-18 查看目录/data 的信息(8)

从图 3-18 可以看出，HDFS 的目录/data 中存在文件 a.txt 和 b.txt，说明已成功将本地文件上传到 HDFS。

### 8. -cat

-cat 用于查看 HDFS 指定文件的内容，语法格式如下。

```
hdfs dfs -cat <src>
```

上述语法格式中，参数 src 用于指定查看的文件。接下来演示如何使用子命令选项-cat 查看目录/data 中的文件 a.txt 的内容，具体命令如下。

```
$ hdfs dfs -cat /data/a.txt
```

上述命令执行完成的效果如图 3-19 所示。

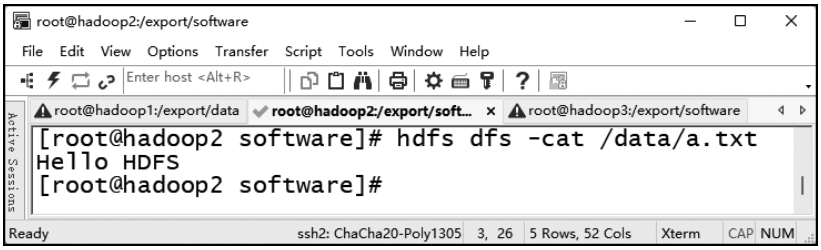


图 3-19　查看目录/data 中文件 a.txt 的内容

从图 3-19 可以看出，目录/data 中文件 a.txt 的内容为 Hello HDFS，说明已成功查看到 HDFS 中的 a.txt 文件。

### 9. -get

-get 用于将 HDFS 的指定文件下载到本地文件系统的指定目录，语法格式如下。

```
hdfs dfs -get [-f] <src> <localdst>
```

上述语法格式中，参数-f 为可选，用于判断下载的文件在本地文件系统的指定目录中是否存在，如果存在，则下载的文件会替换指定目录中已存在的文件；参数 src 用于指定 HDFS 中的文件，可以同时下载多个文件；参数 localdst 用于指定下载到本地文件系统的路径，该路径必须存在。

接下来演示如何使用子命令选项-get 将 HDFS 中目录/data 中的文件 a.txt 和 b.txt 下载到本地文件系统的/opt 目录下，具体命令如下。

```
$ hdfs dfs -get /data/a.txt /data/b.txt /opt
```

上述命令执行完成后，在本地文件系统执行“ll /opt”命令查看目录/opt 的内容，如图 3-20 所示。

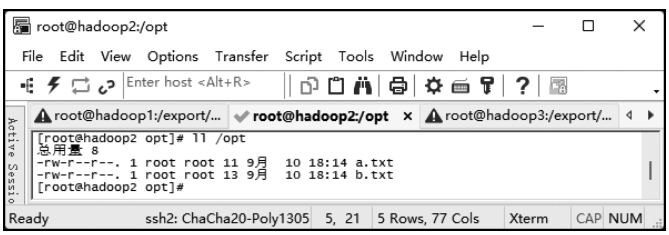


图 3-20　查看目录/opt 的内容

从图 3-20 可以看出,本地文件系统的目录/opt 中存在文件 a.txt 和 b.txt,说明已成功将 HDFS 上的文件下载到本地。

### 3.5.2 案例——通过 Shell 脚本定时采集数据到 HDFS

在实际的开发环境中,服务器每天都会产生大量的日志文件,这些日志文件会记录服务器的运行状态。当服务器宕机时,可以从日志文件中查找服务器宕机的原因,从而尽快让服务器恢复正常运行。为了方便地分析日志文件,通常采用 Shell 脚本周期性地将日志文件上传到 HDFS,如每隔 1 小时将日志文件上传到 HDFS 上。同样,合理的规划有助于帮助人们在个人发展中明确方向和目标,提高自我认知和决策能力,同时提高工作效率和质量。

接下来通过一个案例演示如何通过 Shell 脚本周期性地将 Hadoop 的日志文件上传到 HDFS,本案例的 Hadoop 日志文件数据来自第 2 章中完全分布式模式部署的 Hadoop 集群在运行时产生的数据。

#### 1. 创建 Shell 脚本

在虚拟机 Hadoop1 的/export/data 目录下执行“vi uploadHDFS.sh”命令,编辑 Shell 脚本文件 uploadHDFS.sh,在该文件中实现上传 Hadoop 日志文件到 HDFS 的代码,具体代码如文件 3-1 所示。

**文件 3-1 uploadHDFS.sh**

```
#! /bin/bash
export HADOOP_HOME=/export/servers/hadoop-3.3.0
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
hadoop_log_dir=/export/servers/hadoop-3.3.0/logs/
log_toupload_dir=/export/data/logs/toupload/
date=`date +%Y_%m_%d_%H_%M`
hdfs_dir=/hadoop_log/$date/
if [ -d $log_toupload_dir ];
then
        echo "$log_toupload_dir exits"
else
        mkdir -p $log_toupload_dir
if
ls $hadoop_log_dir | while read fileName
do
if [[ $fileName == *.log ]];
then
    echo "moving hadoop log to $log_toupload_dir"
    cp $hadoop_log_dir/*.log $log_toupload_dir
    scp root@hadoop2:$hadoop_log_dir/*.log $log_toupload_dir
    scp root@hadoop3:$hadoop_log_dir/*.log $log_toupload_dir
    echo "moving hadoop log willDoing"
    break
if
done
echo "create $hdfs_dir"
hdfs dfs -mkdir -p $hdfs_dir
ls $log_toupload_dir | while read fileName
do
    echo "upload hadoop log $fileName to $hdfs_dir"
    hdfs dfs -put $log_toupload_dir$fileName $hdfs_dir
    echo "upload hadoop log $fileName willDoing"
done
echo "delete $log_toupload_dir log"
rm -fr $log_toupload_dir
```

上述代码中，第 2～3 行代码添加了 Hadoop 的环境变量，以免脚本执行时无法使用 HDFS 的 Shell 命令。

第 4 行代码定义变量 hadoop_log_dir，指定 Hadoop 的日志文件存放到目录/export/servers/hadoop-3.3.0/logs/。

第 5 行代码定义变量 log_toupload_dir，指定收集不同虚拟机中 Hadoop 日志文件的目录/export/data/logs/toupload/。

第 6 行代码定义变量 date，指定获取当前系统时间的年(%Y)、月(%m)、日(%d)、时(%H)和分(%M)，并通过字符“_”拼接为字符串。

第 7 行代码定义变量 hdfs_dir，指定 Hadoop 日志文件上传到 HDFS 的目录/hadoop_log/$date/，其中，目录中的 $date 表示使用变量 date 代替。

第 8～13 行代码用于判断存放不同服务器中 Hadoop 日志文件的目录是否存在，如果不存在，则创建该目录。

第 14～25 行代码用于将不同虚拟机中 Hadoop 的日志文件收集到目录/export/data/logs/toupload/，其中，第 19 行代码用于收集当前虚拟机 Hadoop1 中的 Hadoop 日志文件；第 20 行代码用于收集虚拟机 Hadoop2 中的 Hadoop 日志文件。第 21 行代码用于收集虚拟机 Hadoop3 中的 Hadoop 日志文件。

第 27 行代码用于在 HDFS 中创建存放 Hadoop 日志文件的目录。

第 28～33 行代码用于遍历目录/export/data/logs/toupload/中的 Hadoop 日志文件，并将这些日志文件逐个上传至 HDFS 的指定目录。

第 35 行代码用于删除目录/export/data/logs/toupload 中的 Hadoop 日志文件，以免后续再次执行 Shell 脚本 uploadHDFS.sh 上传 Hadoop 日志文件时，上传重复的日志文件。

需要注意的是，由于脚本文件 uploadHDFS.sh 中涉及使用 scp 命令从其他虚拟机获取文件的操作，所以为了脚本文件 uploadHDFS.sh 运行过程的连续性，避免出现输入密码的操作，这里需要在配置了单向免密钥功能的虚拟机 Hadoop1 上执行脚本文件 uploadHDFS.sh。

### 2. 执行 Shell 脚本

进入虚拟机 Hadoop1 的/export/data/目录，确保 Hadoop 集群处于启动状态，执行“sh uploadHDFS.sh”命令运行 Shell 脚本文件 uploadHDFS.sh，脚本文件运行完成后的效果如图 3-21 所示。

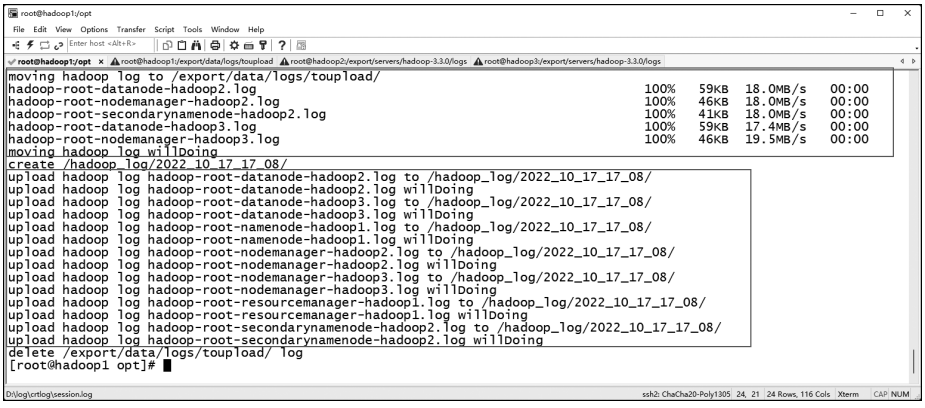


图 3-21　脚本文件运行完成后的效果

图 3-21 展示的脚本文件 uploadHDFS.sh 的输出信息中,首先将 Hadoop 的日志文件收集到目录/export/data/logs/toupload,然后在 HDFS 中创建目录/hadoop_log/2022_10_17_17_08/,并将目录/export/data/logs/toupload 收集的 Hadoop 日志文件上传到目录/hadoop_log/2022_10_17_17_08/,最后删除目录/export/data/logs/toupload 中的 Hadoop 日志文件。

### 3. 验证 Hadoop 日志文件是否上传成功

在本地计算机的浏览器中打开 HDFS 的 Web UI,查看 HDFS 的/hadoop_log/2022_10_17_17_08/目录中是否存在 Hadoop 日志文件,如图 3-22 所示。

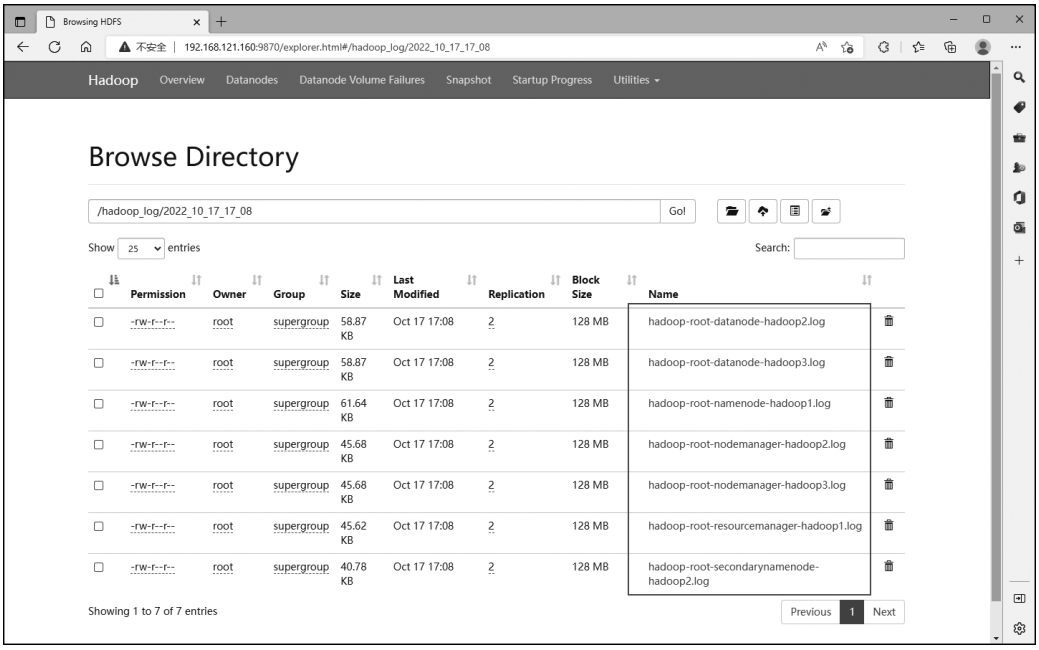


图 3-22 验证 Hadoop 日志文件是否上传成功

从图 3-22 可以看出,HDFS 的/hadoop_log/2022_10_17_17_08/目录中存在 Hadoop 日志文件,如日志文件 hadoop-root-datanode-hadoop2.log 用于记录虚拟机 Hadoop2 中的 DataNode 服务在运行过程中产生的日志信息,说明已成功将 Hadoop 日志文件上传到 HDFS 的指定目录。

### 4. 定时执行 Shell 脚本文件

下面通过 Linux 提供的定时任务工具 Crontab 定时运行 Shell 脚本文件 uploadHDFS.sh,实现周期性地将 Hadoop 的日志文件上传到 HDFS。

(1) 在虚拟机 Hadoop1 上执行“rpm -qa | grep crontab”命令,检查是否安装了 Crontab,如图 3-23 所示。

图 3-23 中出现了 Crontab 的版本号为 1.11,因此说明虚拟机 Hadoop1 已经安装了 Crontab。若没有显示 Crontab 的版本信息,则说明没有安装 Crontab,此时可以先安装 Crontab,具体命令如下。

```
$ yum -y install vixie-cron
$ yum -y install crontabs
```

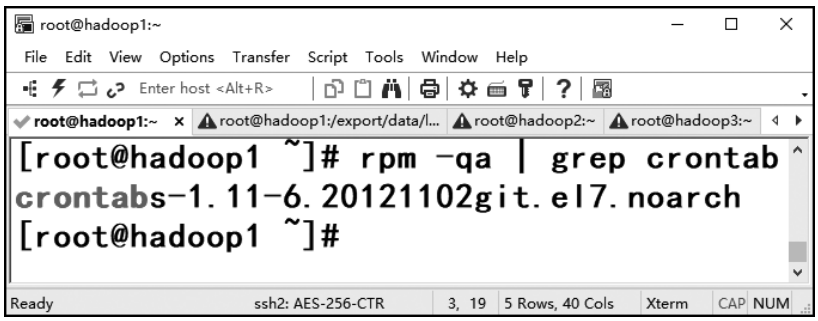


图 3-23　检查虚拟机 Hadoop1 是否安装了 Crontab

（2）使用 Crontab 时，必须保证 Crontab 服务处于运行状态，启动 Crontab 服务的命令如下。

```
$ service crond start
```

上述命令执行完成后，执行“service crond status”命令验证 Crontab 服务是否启动成功，如图 3-24 所示。

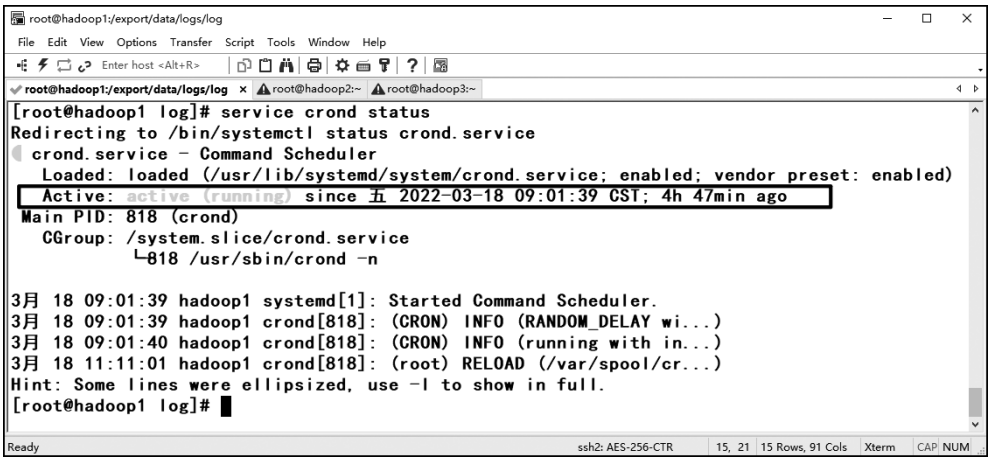


图 3-24　验证 Crontab 服务是否启动成功

图 3-24 中出现了“Active：active (running)”的提示信息，说明 Crontab 服务已启动成功。若 Crontab 服务启动失败，则会出现“Active：inactive (dead)”的提示信息。

（3）进入虚拟机 Hadoop1 的/export/data/目录，为 Shell 脚本文件 uploadHDFS.sh 添加可执行权限，这样做的目的是通过 Crontab 的定时任务运行 Shell 脚本文件 uploadHDFS.sh，具体命令如下。

```
$ chmod 777 uploadHDFS.sh
```

（4）在虚拟机 Hadoop1 上执行“crontab -e”命令编辑 Crontab 文件，配置定时任务，在 Crontab 文件中添加如下内容。

```
*/10 * * * * /export/data/uploadHDFS.sh
```

上述内容配置的定时任务表示每 10 分钟运行一次 Shell 脚本文件 uploadHDFS.sh。由于定时任务工具 Crontab 的使用并非本书的重点，所以上述配置定时任务的内容读者了解即