第3章 HDFS分布式文件系统

学习目标

- 了解文件系统的基本分类，能够说出本地文件系统、网络文件系统和分布式文件系统的主要特点。
- 掌握HDFS的基本结构，能够描述HDFS架构中的NameNode、DataNode、数据块等核心概念及其作用。
- 了解HDFS的主要特点，能够说明HDFS在大文件存储、高容错性和可移植性等方面的优势。
- 了解HDFS的文件读写流程，能够说出从客户端向HDFS写入文件以及从HDFS读取文件的主要步骤。
- 了解HDFS的健壮性，能够解释HDFS中心跳机制、副本机制和负载均衡等策略的含义。
- 掌握HDFS Shell操作，能够熟练运用dfs子命令提供的子命令选项操作HDFS。
- 熟悉HDFS Java API操作，能够编写简单的应用程序，实现对HDFS的读写和管理操作。
- 了解Federation，能够说出Federation的基本概念及其使用方式。
- 了解Erasure Coding，能够说出Erasure Coding的基本概念及其使用方法。
- 通过HDFS健壮性的学习，掌握保障数据可靠性和故障恢复的基本方法，培养根据业务需求制定基础容灾方案的能力。

HDFS是Hadoop体系中的核心组件之一，主要用于解决海量数据的存储与管理问题。作为当前应用最广泛的分布式文件系统，HDFS以高可靠性、高吞吐量和可扩展性著称，为大数据的分布式计算提供了坚实的存储基础。本章将从文件系统（File System）的基本分类入手，逐步讲解HDFS的体系结构、工作原理与常见操作，帮助读者全面理解其在Hadoop架构中的作用与应用场景。

3.1 文件系统的分类

在计算机系统中，文件系统用于组织与管理文件命名空间（Namespace）、持久化存储及访问控制。常见的操作系统（如Windows、macOS、Linux）均内置了本地文件系统。随着网络与分布式技术的发展，文件系统也逐渐具备通过网络或分布式架构向多台主机提供统一访问接口的能力。从部署和访问方式来看，文件系统通常可分为本地（单机）文件系统、网络

文件系统和分布式文件系统,具体介绍如下。

1. 本地文件系统

本地文件系统是最基础、最常见的文件系统类型。它运行在单台计算机上,直接管理该主机的机械硬盘(Hard Disk Drive,HDD)和固态硬盘(Solid State Drive,SSD),由操作系统内核负责文件的存储、组织与访问控制。然而,随着数据规模的不断增长,本地文件系统的局限性逐渐凸显,主要体现在以下两方面。

(1) 存储空间完全依赖单机的硬件资源,受限于磁盘数量、接口带宽及主机性能,扩展性有限。

(2) 若未启用磁盘阵列(Redundant Arrays of Independent Disks,RAID)、快照或备份机制,一旦主机或存储介质发生故障,文件将面临不可恢复的风险。

2. 网络文件系统

网络文件系统(Network File System,NFS)是一种在网络环境下扩展文件访问能力的文件系统类型。它允许客户端像访问本地文件系统一样访问远程服务器上的文件,实现跨主机的数据共享与读写操作。网络文件系统的主要优势在于集中化管理和便捷的文件共享机制,能够有效提高资源利用率并简化管理工作,但同时存在以下不足。

(1) 在高并发访问或跨地域远程访问场景下,网络传输可能成为性能瓶颈。

(2) 若后端仅由单一存储节点组成,则系统仍可能存在单点故障风险,导致文件无法访问或服务中断。

3. 分布式文件系统

分布式文件系统(Distributed File System,DFS)是由多台服务器协同组成的统一文件系统类型。它通过整合集群中各服务器的存储资源,对外提供单一命名空间与一致的文件访问接口,使用户在使用时无须关心底层数据分布与存储位置。

在分布式文件系统中,为解决文件存储的容量瓶颈问题,可以通过增加服务器的方式来灵活扩展系统的总存储空间与数据处理能力,这种横向扩展机制使系统能够随着数据规模的增长而平滑扩容,从而满足大型数据文件的存储需求。

虽然分布式文件系统在容量扩展方面有效突破了单机存储的瓶颈,但如果无法提升面对大量文件请求时的读写性能,则依然难以弥补本地文件系统和网络文件系统在性能上的不足。因此,分布式文件系统在设计上通过数据分块与并行访问来提升读写效率。

具体而言,分布式文件系统在存储大文件时,会将文件切分成若干较小的数据块,并将这些数据块分布存储到集群中的不同服务器上。例如,一个 30GB 的文件 data.txt 可以被划分为 3 个 10GB 的数据块,分别存储在服务器 B、服务器 C 和服务器 D 上,如图 3-1 所示。

当客户端读取文件 data.txt 时,系统会并行地从这 3 台服务器中读取对应的数据块,并在客户端或上层系统中将其重新组合为完整文件。这种数据分块与并行访问的方式能够充分利用集群中多台服务器的存储与计算资源,显著提高文件的读写效率。

在分布式文件系统中,为记录文件被切分后的信息,需要引入一台专门的服务器,用于管理文件的元数据信息,如图 3-2 所示。

在图 3-2 中,服务器 A 负责保存文件的元数据信息,包括文件被划分的数据块数量及各数据块所在服务器的位置。当客户端请求访问文件 data.txt 时,系统会首先向服务器 A 发送查询请求。服务器 A 根据记录的元数据信息返回文件各数据块的位置信息。随后,客户

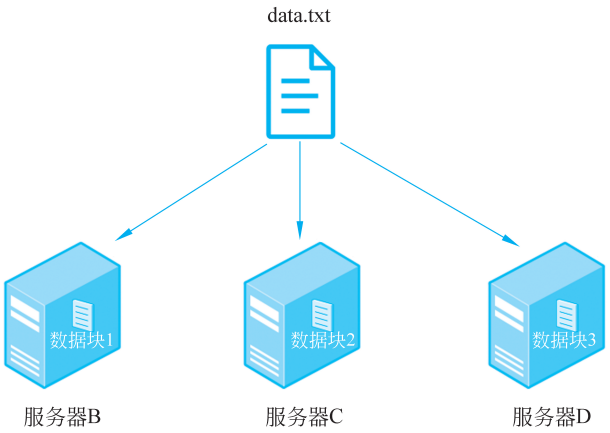

图 3-1　分布式文件系统(1)

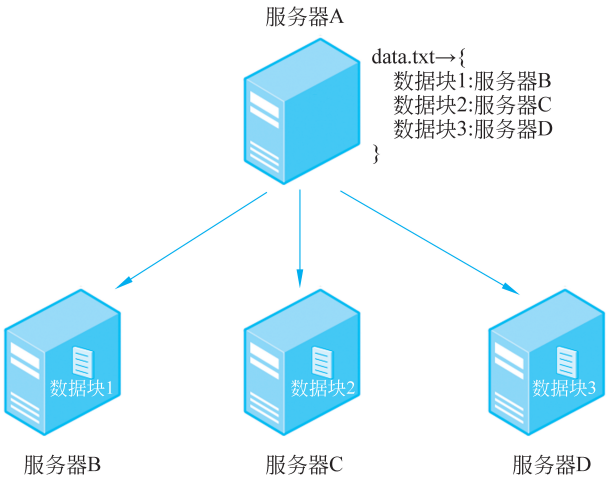

图 3-2　分布式文件系统(2)

端根据这些位置信息,从对应的服务器并行读取数据块并组装为完整文件。

在解决了分布式文件系统的存储容量问题以及提高文件的读写效率后,还必须进一步考虑文件存储的可靠性。由于文件在分布式环境中会被拆分成多个数据块并分布存储在不同服务器上,一旦某台服务器发生故障,其上保存的数据块就可能丢失,从而导致整个文件无法完整恢复。这与本地文件系统和网络文件系统所面临的可靠性问题在本质上相同。

为此,分布式文件系统通常采用副本机制来保障数据可靠性。该机制通过为每个数据块保存多份副本,并将它们存储在不同的服务器上,从而在部分服务器失效时仍能保证文件完整恢复。例如,当分布式文件系统设置副本数量为 2 时,文件被切分后的每个数据块都会有两份副本,并分别存放在不同的服务器中,如图 3-3 所示。

在图 3-3 中,服务器 B 存储数据块 1 和数据块 3,服务器 C 存储数据块 1 和数据块 2,服务器 D 存储数据块 2 和数据块 3。如果服务器 C 发生故障,则其所存储的数据块 1 与数据块 2 仍可分别从服务器 B 与服务器 D 中读取,从而保证文件的完整性。

这种副本机制显著提升了系统的容错能力,使分布式文件系统能够在节点宕机、网络波动或磁盘损坏等情况下依然保持数据可用。

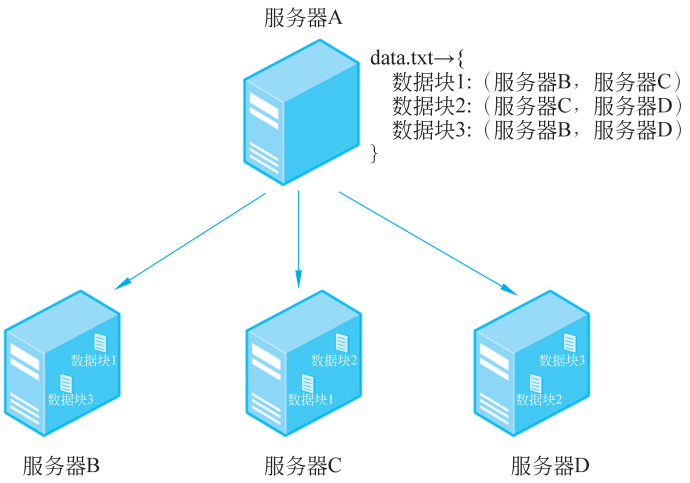

图 3-3 分布式文件系统(3)

3.2 HDFS 简介

在全面了解分布式文件系统的设计思想后,我们便可以深入认识 Hadoop 的核心组件之一 HDFS。HDFS 的设计灵感源自 Google 在 2003 年提出的 GFS 理念,并在开源社区中不断演进与完善。经过多年的发展,HDFS 已成为支撑大数据平台的基础设施之一,广泛应用于互联网、金融、科研、能源等领域的海量数据存储与访问场景。本节将详细介绍 HDFS 的架构组成与系统特点。

3.2.1 HDFS 架构

HDFS 采用主从(Master/Slave)架构,即一个 HDFS 集群通常由一个主节点(Master)和多个从节点(Slave)组成。在该架构中,NameNode 作为主节点,负责管理文件系统的命名空间与元数据信息;DataNode 作为从节点,负责存储实际的数据块并执行读写操作。此外,为减轻 NameNode 的负载并提升元数据的管理效率,HDFS 还引入了 SecondaryNameNode,用于定期协助 NameNode 合并元数据。

HDFS 的架构如图 3-4 所示。

下面针对图 3-4 中 HDFS 架构的核心概念进行介绍。

1. 数据块

在 HDFS 中,数据块是文件系统的最小存储单位。当文件写入 HDFS 时,系统会根据预设的数据块大小上限将文件切分为多个数据块,并按照放置策略分布到集群中的 DataNode 上,以实现数据的分布式存储与并行访问。

在 Hadoop 3.x 版本中,默认的数据块大小为 134217728 字节,即 128MB。当文件大小不足一个完整的数据块,或在文件切分后最后一个数据块未达到上限时,系统仅会为该数据块分配实际占用的字节数。例如,当数据块大小为 128MB 时,一个 300MB 的文件将被切分为 3 个数据块,大小分别为 128MB、128M 和 44MB。

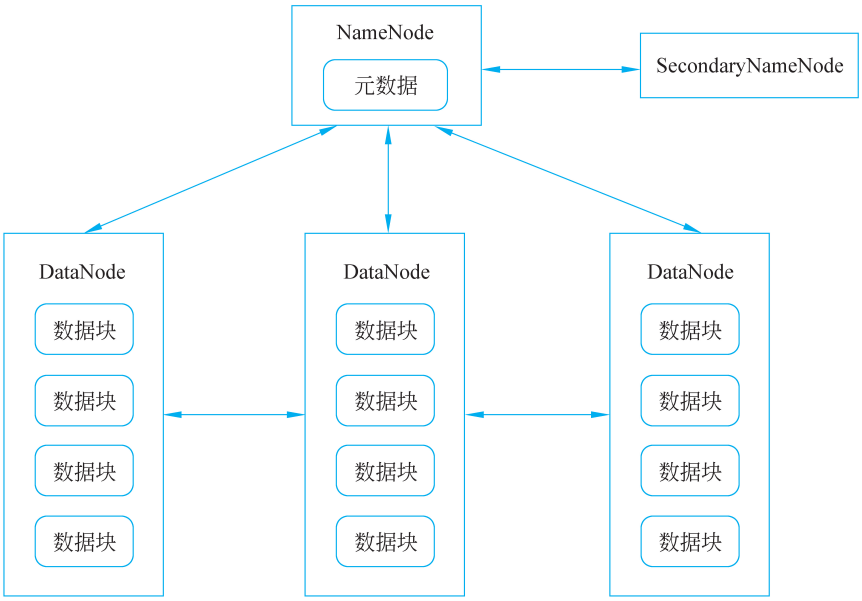

图 3-4　HDFS 的架构

2. 元数据

在 HDFS 中，元数据用于记录文件系统的结构与管理信息，包括目录树、文件名、文件大小、副本数量、文件与数据块的对应关系等。为提高访问效率，NameNode 会将元数据加载在内存中，以便快速响应客户端请求。同时，为防止因故障而导致元数据丢失，HDFS 会在 NameNode 的元数据目录下生成 FsImage 文件，用于持久化保存内存中的元数据。

在 HDFS 的运行过程中，用户对文件系统的增删改操作不会直接写入 FsImage 文件，而是以日志的方式记录在 EditLog 文件中，从而避免频繁的磁盘 I/O 操作，以减轻 NameNode 的负载。EditLog 文件同样生成于 NameNode 的元数据目录下，它采用追加写入的方式保存元数据的每次变更，并为每条记录分配唯一的事务 ID。

当 NameNode 启动时，会首先加载最新的 FsImage 文件，然后根据 EditLog 文件中的事务 ID 顺序回放变更操作，从而恢复系统的元数据状态。随后，NameNode 会等待各 DataNode 上报数据块信息，在内存中重建数据块与存储位置的映射关系。

需要说明的是，在高可用模式的 Hadoop 集群中，EditLog 文件不再保存在 NameNode 的元数据目录中，而是由一组 JournalNode 进程共同维护和管理。

3. NameNode

NameNode 是 HDFS 集群中的主节点，也称为名称节点，负责管理整个文件系统的命名空间与元数据信息，是 HDFS 的核心控制组件。由于 NameNode 负责协调所有客户端与 DataNode 的交互，因此一旦发生故障，用户将无法访问 HDFS，因此其稳定性对于系统的运行至关重要。

NameNode 的主要职责如下。

(1) 管理文件系统的命名空间，执行文件与目录的创建、删除、重命名等操作，并处理客户端发起的各类请求，包括文件读写与权限验证。

(2) 维护文件系统的元数据信息,包括文件与数据块之间的映射关系,以及各数据块的副本存放位置。

(3) 管理和监控所有 DataNode,定期接收它们的心跳与状态报告,并协调 DataNode 执行数据的读写、复制与恢复任务。

4. DataNode

DataNode 是 HDFS 集群中的从节点,也称为数据节点,负责数据块的存储与管理,是 HDFS 进行数据交互的实际执行者。DataNode 的主要职责如下。

(1) 根据 NameNode 的指令执行数据块的创建、复制与删除操作。

(2) 定期向 NameNode 汇报自身存储的数据块列表及运行状态。

(3) 直接与客户端进行数据传输。

5. SecondaryNameNode

SecondaryNameNode 是 HDFS 集群中的辅助节点,主要用于协助 NameNode 管理文件系统的元数据。它会周期性地从 NameNode 中获取 FsImage 文件和 EditLog 文件,在本地执行合并操作,并将生成的新 FsImage 文件回传给 NameNode。

3.2.2 HDFS 的特点

作为当前应用最广泛的分布式文件系统,HDFS 之所以能够支撑海量数据存储与高并发访问,得益于其在大文件存储、高容错性、可移植性等方面的特点,具体介绍如下。

1. 大文件存储

HDFS 支持 GB 级乃至 TB 级的大型文件存储。系统在写入文件时,会将文件切分为多个数据块,并按照放置策略分布到集群中的各个 DataNode 上。在读取文件时,客户端可对多个数据块进行并行访问,从而实现高效读取并提升系统的整体吞吐量。

2. 高容错性

HDFS 通过副本机制与机架感知策略提升系统的容错能力。当某个 DataNode 发生故障时,NameNode 会检测并定位存储在其他 DataNode 上的副本,从而保证文件仍可被正常访问。随后,HDFS 会根据副本放置策略在可用的 DataNode 上自动重新生成丢失的数据块副本,从而恢复副本数量。

3. 简单的一致性模型

HDFS 采用“一次写入,多次读取”的数据存储模型。文件创建后仅允许单个客户端进行写入,写入完成后只能在文件尾部追加数据,而不能对文件内部的任意位置进行修改。这种模型虽然在一定程度上降低了操作的灵活性,但显著简化了数据一致性的维护,并提高了系统的可靠性。

4. 计算尽量靠近数据

在分布式环境中,HDFS 通常与上层计算引擎协同工作,将任务优先调度到存放目标数据块的 DataNode 或同一机架内的节点上执行。这种数据本地化策略能够最大限度地减少网络数据传输开销,从而显著提升系统的整体吞吐量与计算效率。

5. 可移植性

HDFS 基于 Java 实现,具备良好的跨平台特性,可运行在多种硬件架构与操作系统环境中。为进一步提升性能,HDFS 通常会启用本地原生库,以充分利用底层系统的 I/O 优

化能力。凭借这种兼容性，HDFS 已成为构建大数据平台的通用存储基础。

3.3　HDFS 的文件读写流程

通过前文对 HDFS 架构的学习可知，HDFS 主要由 NameNode 与 DataNode 协同构成。那么，这两类节点如何配合完成文件的读写服务？为回答这一问题，下面将分别介绍 HDFS 中文件写入流程与文件读取流程的详细说明，系统梳理从客户端到 NameNode，再到各 DataNode 的整体工作机制。

1. 文件写入流程

客户端向 HDFS 写入文件 data.txt 的完整流程如图 3-5 所示。

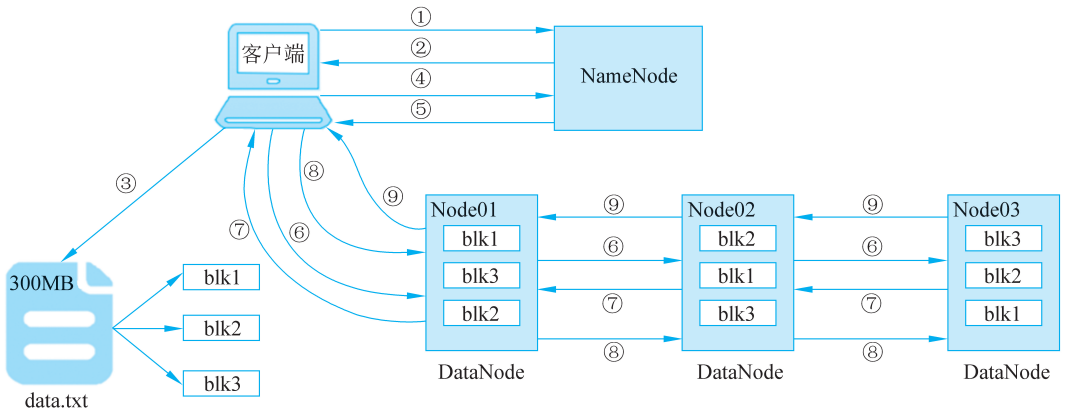

图 3-5　客户端向 HDFS 写入文件 data.txt 的完整流程

具体介绍如下。

（1）客户端与 NameNode 建立通信，并发起写入文件 data.txt 的请求。

（2）NameNode 在接收到请求后，会依次检查目标目录是否存在、用户权限是否符合要求、文件名是否已被占用等条件。若检查通过，NameNode 便在文件系统的命名空间中创建对应的文件条目，并返回创建成功的响应。

（3）客户端根据配置文件中定义的数据块大小，将文件 data.txt 切分为多个数据块。例如，当数据块大小为 300MB 时，客户端会将文件 data.txt 切分为 blk1、blk2 和 blk3 3 个数据块。

（4）客户端向 NameNode 发送请求，申请为数据块 blk1 分配 DataNode。

（5）NameNode 会根据配置文件中定义的副本数量及机架感知策略，挑选出适合存放数据块 blk1 的 DataNode 列表，并返回给客户端。例如，当副本数量为 3 时，NameNode 会返回包含 3 个 DataNode 的列表。

（6）客户端按照就近原则，从 DataNode 列表中选择第一台 DataNode（Node01）建立连接。然后，第一台 DataNode 与第二台 DataNode（Node02）建立连接，第二台 DataNode 与第三台 DataNode（Node03）建立连接，形成一个单向的数据传输管道（Pipeline）。

（7）当管道中的各个 DataNode 依次建立连接后，最末端的 DataNode 会逐层向上汇报连接成功。最终，客户端收到来自第一台 DataNode 的确认信息，表示整个数据传输管道已

成功建立。

(8) 客户端首先将数据块 blk1 写入本地内存缓冲区,并以默认大小 64KB 的 packet(数据包)为单位进行发送。整个写入过程以流式传输的方式进行,具体流程如下。

① 客户端将第一个 packet 发送至管道中的第一台 DataNode。

② 第一台 DataNode 接收到 packet 后,将数据写入本地文件系统,并立即转发给第二台 DataNode。

③ 第二台 DataNode 接收到 packet 后,将数据写入本地文件系统,并立即转发给第三台 DataNode。

这种多节点流水线式写入在提高效率的同时,也强调了各节点之间的协同配合。只有每个环节都确保数据写入可靠、反馈信息准确,整个系统才能稳定运行,这与我们在团队协作中"各负其责,密切配合"的要求不谋而合。

(9) 当管道中的各个 DataNode 依次完成写入后,位于管道末端的 DataNode(Node03)会沿反方向逐层向上返回确认信息。最终,客户端收到来自第一台 DataNode 的确认信息,即认为当前 packet 已成功写入。

当数据块 blk1 的所有 packet 均成功写入并收到相应的确认信息后,客户端会通知 NameNode 数据块 blk1 已上传完成。随后,客户端按照相同的流程依次执行第(4)~(9)步,完成数据块 blk2 和 blk3 的上传。

当文件 data.txt 的所有数据块均上传完成后,客户端关闭输出流,并向 NameNode 发送最终确认。此时,NameNode 更新文件 data.txt 的元数据信息。至此,文件 data.txt 的写入过程全部完成。

2. 文件读取流程

客户端从 HDFS 中读取文件 data.txt 的完整流程如图 3-6 所示。

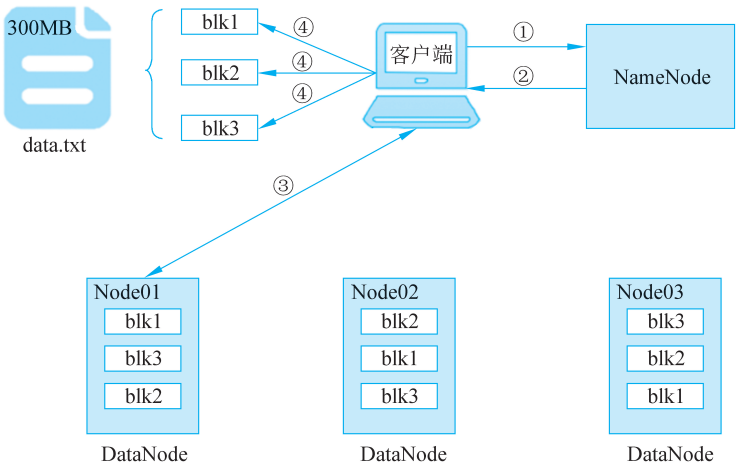

图 3-6 客户端从 HDFS 中读取文件 data.txt 的完整流程

具体介绍如下。

(1) 客户端与 NameNode 建立通信,并发起读取文件 data.txt 的请求。

(2) NameNode 在接收到请求后,会依次检查目标文件是否存在、用户权限是否符合要求等条件。若检查通过,则 NameNode 返回文件 data.txt 的数据块信息,包括各数据块的编

号及其副本所在的 DataNode 地址。

(3) 客户端根据就近原则,从数据块信息中选择距离自身最近的 DataNode 读取数据块。若该 DataNode 可提供全部数据块,则读取过程结束;若部分数据块不在该 DataNode 上,则客户端会自动从下一个合适的 DataNode 中读取剩余数据块,直到完整地读取文件的所有数据块。

(4) 客户端会对各数据块进行校验,确保内容正确无误。校验通过后,系统按照数据块编号的顺序将其依次拼接,恢复成完整的文件 data.txt。这种在读取阶段再次对数据内容进行校验的机制,本质上是一种对数据质量负责的体现。无论是在工程实践还是日常工作中,保持“不轻信,不想当然,对重要结果进行复核”的意识是防止错误扩散的有效方式。

需要说明的是,当文件体积过大时,NameNode 可能无法一次性地返回全部数据块的信息。此时,HDFS 会采用分批返回的方式逐步完成文件读取。

3.4　HDFS 的健壮性

HDFS 的主要目标是在出现故障的情况下仍能可靠地存储与访问数据。为此,系统采用心跳机制、副本机制、数据完整性校验、安全模式和快照 5 种策略共同保障数据可靠性与可恢复性,具体介绍如下。

1. 心跳机制

为实时掌握集群各节点的运行状态,DataNode 会按照固定时间间隔向 NameNode 发送心跳信息。当 NameNode 在固定时间间隔内持续接收到某个 DataNode 的心跳信息时,认为该节点处于存活状态;若在固定时间间隔内未接收到心跳信息,则根据缺失的持续时长依次将该节点标记为 stale 或 dead 状态。

默认情况下,当 NameNode 超过 30s 未接收到 DataNode 的心跳信息时,会将该节点标记为 stale 状态,表示其状态信息可能已过期。当 NameNode 连续 10min 未接收到 DataNode 的心跳信息时,会将该节点标记为 dead 状态,表示其不可用。

当某个 DataNode 被标记为 dead 状态后,系统会将其从调度列表中移除,该节点不再参与数据读写操作。与此同时,NameNode 会检测其上存储的数据块副本,并在其他存活的 DataNode 上启动副本复制操作,以恢复副本数量至预设目标值,从而保证数据的完整性与可靠性。

2. 副本机制

副本机制用于确保存储在 DataNode 上的每个数据块都具有多个副本。默认情况下,副本数量为 3,系统会在 HDFS 中为每个数据块保存 3 个副本,并尽可能地将它们分布在不同的 DataNode 上,以提高容错能力。

当因某个 DataNode 不可用而导致某些数据块的副本数量低于预设值时,NameNode 会检测到这一情况,并调度其他处于存活状态的 DataNode 自动创建新的副本,直至副本数量恢复到预设值。

3. 数据完整性校验

数据完整性校验用于防止由于存储设备故障或网络传输错误而导致的数据损坏。当客户端从 HDFS 读取文件时,系统会对文件的每个数据块进行校验和(Checksum)验证。具

体而言,当客户端向 HDFS 写入文件时,DataNode 会按照固定大小的 Chunk(默认为 512Byte)计算校验和,并将校验和与数据块文件一同存储在本地文件系统中。当客户端读取文件时,会从 DataNode 获取数据块及其对应的校验和,并在本地重新计算校验和进行比对。若检测到数据块的校验和不匹配,则说明该副本已损坏,客户端将自动从其他 DataNode 读取该数据块的副本,以确保获取正确的数据内容。

4. 安全模式

安全模式是 HDFS 的一种特殊运行状态,在该状态下,文件系统对客户端仅提供只读访问,不允许执行写入或删除等修改操作。安全模式主要用于 HDFS 启动阶段,以确保文件系统的元数据与数据块信息之间的一致性。

当 HDFS 启动时,NameNode 会自动进入安全模式,此时各 DataNode 会向其上报所保存的数据块及其副本状态。NameNode 根据这些信息统计当前满足副本要求的数据块比例,当该比例达到设定的安全阈值(默认为 99.9%)后,系统即认为数据块分布完整,自动退出安全模式并进入正常运行状态。若长时间未达到安全阈值,NameNode 将持续停留在安全模式,导致客户端无法执行修改操作。

5. 快照

快照指 HDFS 对整个文件系统或指定目录在某一时间点所生成的只读镜像。快照能够记录当时的数据状态,用于在后续出现误删、误改等操作时恢复数据。当用户因操作错误或系统异常而导致数据丢失时,可通过快照将文件系统恢复至指定的历史时间点,从而有效保障数据的可恢复性和系统的安全性。

3.5 HDFS Shell 操作

3.5.1 HDFS Shell 介绍

在学习了 HDFS 的基本概念与架构之后,下一步需要通过命令行工具对分布式文件系统进行实际操作。HDFS 提供了一组类 UNIX 风格的命令行工具,通常称之为 HDFS Shell。借助 HDFS Shell 的命令,用户可以像使用本地文件系统一样执行目录创建、文件上传与下载、权限设置、空间统计等常见操作。

保持与 Linux Shell 使用体验的一致性是 HDFS Shell 的重要设计目标之一。诸如 ls、mkdir、cat、rm、du 等命令的语义在 HDFS Shell 中相同,但它们实际作用于 HDFS 命名空间,而非 Linux 操作系统的本地文件系统。

HDFS Shell 的语法格式如下。

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

上述语法格式中,hdfs 是 HDFS Shell 的入口。OPTIONS 为可选项,是 HDFS Shell 的选项,用于指定命令的行为。例如,--config 选项用于指定 Hadoop 配置文件的所在目录;--daemon 选项用于操作 HDFS 的守护进程;--help 选项用于显示 HDFS Shell 的用法说明与可用的选项和子命令列表。

SUBCOMMAND 是 HDFS Shell 的子命令,用于指定要执行的具体操作。按照功能类型,子命令可分为 3 种,分别是 Admin Commands(管理员命令)、Client Commands(客户端

命令)和 Daemon Commands(进程命令),其中 Admin Commands 主要用于 HDFS 的管理与维护操作,例如 fsck 子命令用于检查文件系统的一致性。Client Commands 主要用于文件系统的日常操作,例如 dfs 子命令用于管理文件系统中的目录与文件。Daemon Commands 主要用于管理 HDFS 的守护进程,例如 datanode 子命令用于运行 DataNode 进程。

SUBCOMMAND OPTIONS 为可选项,是 HDFS Shell 子命令选项,用于指定子命令执行时所需的具体选项。例如,dfs 子命令可以通过子命令选项-mkdir 在文件系统中创建目录。

本书将重点介绍 HDFS 日常操作中常用的子命令 dfs,并说明其常见的子命令选项及功能。子命令 dfs 常用的子命令选项如表 3-1 所示。

表 3-1　子命令 dfs 常用的子命令选项

子命令选项	功 能 描 述
-ls	输出指定目录下的文件和子目录信息
-du	统计文件或目录的大小,以及文件的副本在 HDFS 中实际占用的存储空间
-mv	移动文件或目录
-cp	复制文件或目录
-rm	删除文件或目录
-head	查看文件开头 1KB 的内容
-put	将本地文件系统中的文件上传到 HDFS
-cat	将文件内容显示在标准输出上
-help	查看子命令的用法与子命令选项说明
-mkdir	创建目录
-get	将 HDFS 中的文件下载到本地文件系统

下面对表 3-1 中子命令 dfs 常用的子命令选项的语法格式与基本用法进行说明,具体内容如下。

1. -mkdir

-mkdir 的语法格式如下。

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

上述语法格式中,参数-p 为可选项,用于递归创建多级目录。在未指定参数-p 且上级目录不存在时,命令将失败并返回错误。在指定参数-p 时,如果目录已存在,则命令不会重复创建且不会报错。path 用于指定一个或多个目录,每个目录通过空格分隔。

接下来,演示如何在 HDFS 中创建目录/data 及多级目录/data/dataChild/dataChild1,具体命令如下。

```
hdfs dfs -mkdir -p /data /data/dataChild/dataChild1
```

上述命令执行完成后,通过 HDFS Web UI 查看 HDFS 的目录结构,如图 3-7 所示。

从图 3-7 可以看出,在 HDFS 中存在/data 目录,在/data 目录下存在 dataChild 子目录,在/data/dataChild 目录下存在/data/dataChild/dataChild1 子目录,说明已成功在 HDFS 中创建目录/data 及多级目录/data/dataChild/dataChild1。

从图 3-7 可以看出,在 HDFS 的目录结构中,每个对象(文件或目录)都包含

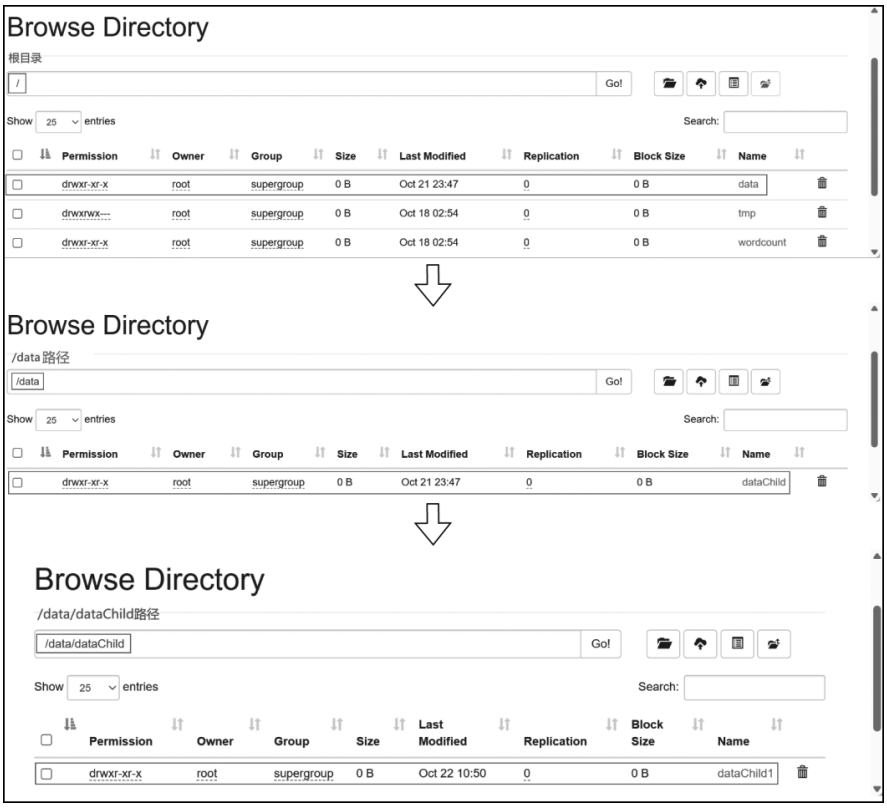

图 3-7　通过 HDFS Web UI 查看 HDFS 的目录结构

Permission、Owner、Group、Size、Last Modified、Replication、Block Size 和 Name 信息，分别表示权限、所属用户、所属用户组、大小、最近修改时间、副本数量、数据块大小和名称。

在 HDFS 中，权限由 4 部分内容组成，用于标识对象的访问控制信息和类型。下面以图 3-7 中/data 目录的权限为例进行介绍，如图 3-8 所示。

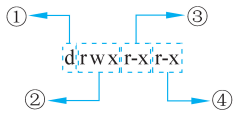

图 3-8　权限的组成部分

关于图 3-8 中权限每部分的介绍如下。

① 用于标识对象类型，d 表示目录，若该位置为-则表示普通文件。

② 用于标识所属用户的操作权限，位置固定为 rwx 3 位，其中 r 表示读取权限，w 表示写入权限，x 表示执行权限。缺失某项权限时用-占位。图 3-8 显示的权限表示所属用户具有读取、写入和执行 3 项权限。

③ 用于标识所属用户组的操作权限，位置固定为 rwx 3 位。图 3-8 显示的权限表示所属用户组具有读取和执行两项权限。

④ 用于标识不在所属用户组内的其他用户的操作权限，位置固定为 rwx 3 位。图 3-8 显示的权限表示其他用户具有读取和执行两项权限。

需要说明的是，由于目录在 HDFS 中不存在副本，也不会被切分为数据块，所以其副本数量和数据块大小的值均为 0。

2. -put

-put 的基础语法格式如下。

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

上述语法格式中，localsrc 用于指定本地文件系统中源文件的路径，可一次指定一个或多个文件的路径，各路径之间用空格分隔。dst 用于指定 HDFS 上已存在的目录或文件路径。若指定为文件路径，则上传后的文件将以该路径中的文件名命名。参数-f 为可选项，用于覆盖指定目录下已存在的同名文件；若未指定参数-f 且指定目录中已存在同名文件，命令将提示文件已存在。

需要注意的是，当 localsrc 指定多个路径时，dst 必须为已存在的目录。

接下来，演示如何将本地文件系统中的文件 app_log_20251021.log 和 app_log_20251022.log 上传到 HDFS 的/data 目录下，具体操作步骤如下。

（1）参考上传 JDK 安装包的方式将文件 app_log_20251021.log 和 app_log_20251022.log 上传到虚拟机 Node01 的/export/data 目录中。

（2）将/export/data 目录中的文件 app_log_20251021.log 和 app_log_20251022.log 上传 HDFS 的/data 目录下。在虚拟机 Node01 中执行如下命令。

```
hdfs dfs -put /export/data/app_log_20251021.log \
/export/data/app_log_20251022.log /data
```

上述命令执行完成后，通过 HDFS Web UI 查看/data 目录下的内容，如图 3-9 所示。

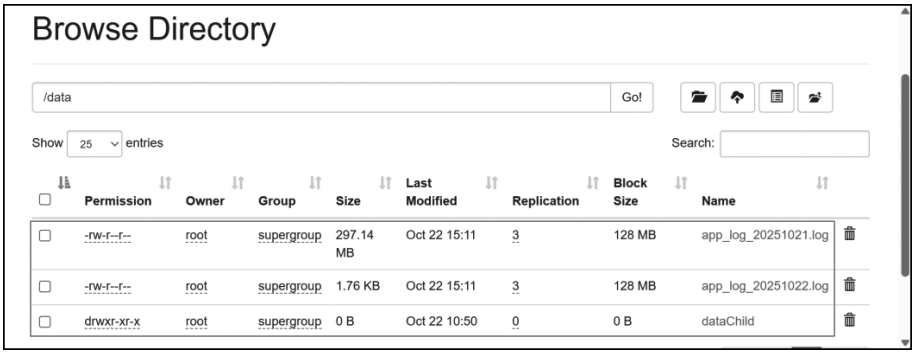

图 3-9　通过 HDFS Web UI 查看/data 目录下的内容

从图 3-9 可以看出，/data 目录下存在文件 app_log_20251021.log 和 app_log_20251022.log，它们的副本数量均为 3 且数据块大小均为 128MB，说明已成功将本地文件系统中的文件上传到 HDFS 中的指定目录下。

3. -ls

-ls 的基础语法格式如下。

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

上述语法格式中，参数-C 为可选项，用于仅列出指定目录下文件和子目录的完整路径，不显示权限、所属用户、大小等详细信息。默认情况下，列出指定目录下文件和子目录的完整路径和详细信息。

参数-h 用于以易读格式显示文件的大小，例如以 KB、MB 为单位。默认情况下，HDFS Shell 输出的大小以字节为单位，不像 HDFS Web UI 那样自动格式化为易读格式。若文件的大小小于 1KB，则仍然以字节为单位。

参数-R 用于递归地列出指定目录及其子目录中的所有内容。

参数-t 用于按照最近修改时间对输出结果进行降序排序。默认按路径的字典序输出结果。

参数-S 用于按照文件大小对输出结果进行降序排序。

参数-r 用于对输出结果进行反向排序。可与参数-t 和-S 联合使用,从而实现升序排序的效果。

接下来,演示如何查看 HDFS 指定目录下的内容,具体内容如下。

(1) 查看/data 目录的内容,并按照最近修改时间对输出结果进行升序排序,具体命令如下。

```
hdfs dfs -ls -t -r /data
```

上述命令执行完成后的效果如图 3-10 所示。

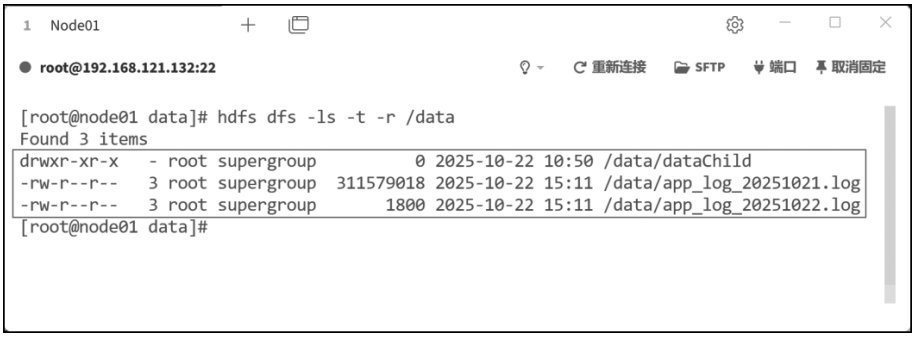

图 3-10 查看/data 目录的内容(1)

在图 3-10 中,输出的每行内容对应/data 目录中的一个对象,包含该对象的详细信息。每行从左到右依次表示权限、副本数量、所属用户、所属用户组、大小、最近修改日期和完整路径。

从图 3-10 可以看出,/data 目录中包含一个子目录和两个文件,它们按照最近修改时间进行了升序排序。

(2) 查看/data 目录的内容,以易读格式显示文件的大小,并按照文件和目录的大小对输出结果进行降序排序,具体命令如下。

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

上述命令执行完成后的效果如图 3-11 所示。

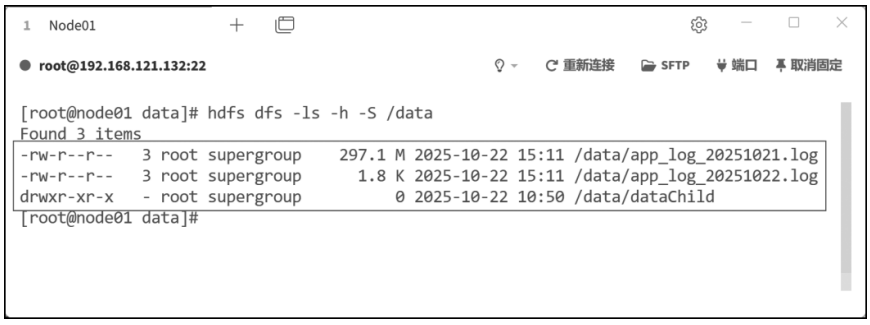

图 3-11 查看/data 目录的内容(2)

从图 3-11 可以看出，/data 目录中的内容按照文件和目录的大小进行了降序排序，并且输出结果中的大小信息已经过格式化处理，以 KB(千字节)和 MB(兆字节)为单位显示。

(3) 查看/data 目录及其子目录中的所有内容，但不显示详细信息，具体命令如下。

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

上述命令执行完成后的效果如图 3-12 所示。

图 3-12　查看/data 目录的内容(3)

从图 3-12 可以看出，输出结果不仅包含/data 目录下所有文件和子目录的完整路径，还列出了其子目录 dataChild 中的所有文件和子目录的完整路径。

4. -du

-du 的基础语法格式如下。

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

上述语法格式中，参数-s 为可选项，主要用于统计指定目录中所有文件的总大小，以及这些文件副本的总大小。在未指定参数-s 且 path 为目录时，会逐条列出该目录下的每个文件与子目录，并分别显示其大小与副本占用的存储空间。目录的大小以及副本占用的存储空间是其包含的所有文件大小及其副本占用空间的汇总。

接下来，演示如何逐条列出/data 目录下的每个文件与子目录，并以易读格式分别显示其大小与副本占用的存储空间，具体命令如下。

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

上述命令执行完成后的效果如图 3-13 所示。

图 3-13　逐条列出/data 目录下每个文件与子目录的大小

从图 3-13 可以看出,文件 app_log_20251021.log 的大小为 297.1MB,其副本占用的存储空间为 891.4MB。文件 app_log_20251022.log 的大小为 1.8KB,其副本占用的存储空间为 5.3KB。由于子目录 dataChild 为空,不包含任何文件,所以其大小及副本占用的存储空间均为 0。

5. -mv

-mv 的语法格式如下。

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

上述语法格式中,src 用于指定要移动的目录或文件路径,可一次指定一个或多个目录或文件路径,它们之间用空格分隔。dst 用于指定目标目录。若目标目录中已存在同名文件或目录,则命令将提示对象已存在并终止操作。

在移动文件或目录时,可能出现以下几种情况。

(1) 当目标目录已存在时,系统会将指定的文件或目录移动到该目录内,并保留其原有名称。

(2) 当移动的对象为单个目录时,目标目录可以尚不存在,但其父目录必须已存在。此时,系统会将源目录移动到目标目录的父目录下,并根据目标目录中的最后一级目录名对其进行重命名。

(3) 当移动的对象为单个文件时,dst 也可以指定为文件路径,该文件路径中指定的目标文件必须不存在,但其所在的目录必须已存在。此时,系统会将源文件移动到目标文件所在的目录下,并将其重命名为目标文件的名称。

接下来,演示如何移动 HDFS 的文件和目录,具体内容如下。

(1) 将/data 目录下的文件 app_log_20251021.log 和 app_log_20251022.log 移动到/data/dataChild 目录下,具体命令如下。

```
hdfs dfs -mv /data/app_log_20251021.log /data/app_log_20251022.log \
/data/dataChild
```

上述命令执行完成后,查看/data 目录及其子目录中的所有内容,如图 3-14 所示。

图 3-14 移动文件

从图 3-14 可以看出,/data 目录下的文件 app_log_20251021.log 和 app_log_20251022.log 已移动到/data/dataChild 目录下。

(2) 将/data/dataChild 目录下的文件 app_log_20251022.log 重命名为 app_log_

20251022.log.tmp，具体命令如下。

```
hdfs dfs -mv /data/dataChild/app_log_20251022.log \
/data/dataChild/app_log_20251022.log.tmp
```

上述命令执行完成后，查看/data 目录及其子目录中的所有内容，如图 3-15 所示。

图 3-15　重命名文件

从图 3-15 可以看出，/data/dataChild 目录下的文件 app_log_20251022.log 已重命名为 app_log_20251022.log.tmp。

(3) 将/data 目录下的子目录 dataChild 移动到根目录，具体命令如下。

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

上述命令执行完成后，查看根目录的内容，如图 3-16 所示。

图 3-16　移动目录

从图 3-16 可以看出，在根目录下存在 dataChild 子目录。

6. -cp

-cp 的基础语法格式如下。

```
hdfs dfs -cp [-f] <src> ... <dst>
```

上述语法格式中，参数-f 为可选项，用于覆盖指定目录下已存在的同名文件；若未指定参数-f 且指定目录中已存在同名文件，则命令将提示对象已存在并终止操作。src 用于指定要复制的目录或文件路径，可一次指定一个或多个目录或文件路径，它们之间用空格分隔。dst 用于指定目标目录。

在复制文件或目录时，可能出现以下几种情况。

(1) 当目标目录已存在时,系统会将指定的文件或目录复制到该目录内,并保留其原有名称。

(2) 当复制的对象为单个目录时,目标目录可以尚不存在,但其父目录必须已存在。此时,系统会将源目录复制到目标目录的父目录下,并根据目标目录中的最后一级目录名对其进行重命名。

(3) 当复制的对象为单个文件时,dst 也可以指定为文件路径,该文件路径中指定的目标文件必须不存在,但其所在的目录必须已存在。此时,系统会将源文件复制到目标文件所在的目录下,并将其重命名为目标文件的名称。

接下来,演示如何复制 HDFS 的文件和目录,具体内容如下。

(1) 将/dataChild 目录下的文件 app_log_20251022.log.tmp 复制到/data 目录下,并重命名为 app_log_20251022.log.tmp.copy,具体命令如下。

```
hdfs dfs -cp /dataChild/app_log_20251022.log.tmp \
/data/app_log_20251022.log.tmp.copy
```

上述命令执行完成后,查看/data 目录的内容,如图 3-17 所示。

图 3-17 复制文件

从图 3-17 可以看出,/data 目录中已存在文件 app_log_20251022.log.tmp.copy。

(2) 将/dataChild 目录下的文件 app_log_20251021.log 和子目录 dataChild1 复制到/data 目录下,具体命令如下。

```
hdfs dfs -cp /dataChild/app_log_20251021.log /dataChild/dataChild1 /data
```

上述命令执行完成后,查看/data 目录的内容,如图 3-18 所示。

图 3-18 复制文件和目录

从图 3-18 可以看出，/data 目录中已存在文件 app_log_20251021.log 和子目录 dataChild1。

7. -rm

-rm 的基础语法格式如下。

```
hdfs dfs -rm [-f] [-r|-R] [-skipTrash] <src> ...
```

上述语法格式中，参数-f 为可选项，用于忽略不存在的文件或目录且不提示错误；当未指定参数-f 时，若指定的文件或目录不存在，则提示对象不存在并终止操作。

参数-r 和-R 的含义相同，均为可选项，用于递归地删除目录及其全部内容。在未指定参数-r 或-R 时，只能删除文件，不能删除目录。

参数-skipTrash 为可选项，用于跳过回收站，直接永久删除指定文件或目录。

src 用于指定要删除的目录或文件路径，可一次指定一个或多个目录或文件路径，它们之间用空格分隔。

接下来，演示如何删除/dataChild 目录，具体命令如下。

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

上述命令执行完成后的效果如图 3-19 所示。

图 3-19　删除目录

从图 3-19 可以看出，删除/dataChild 目录后，系统提示已将其移入回收站，目录为/user/root/.Trash/Current。

8. -cat

-cat 的基础语法格式如下。

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

上述语法格式中，src 用于指定文件路径，可一次指定一个或多个文件路径，它们之间用空格分隔。

接下来，演示如何查看/data 目录中文件 app_log_20251022.log.tmp.copy 的内容，具体命令如下。

```
hdfs dfs -cat /data/app_log_20251022.log.tmp.copy
```

上述命令执行完成后的效果如图 3-20 所示。

图 3-20 查看文件的内容(1)

图 3-20 展示了文件 app_log_20251022.log.tmp.copy 的部分内容。

9. -get

-get 的基础语法格式如下。

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

上述语法格式中,参数-f 为可选项,用于覆盖指定目录下已存在的同名文件;若未指定参数-f 且指定目录中已存在同名文件,则提示对象已存在并终止操作。src 用于指定文件路径,可一次指定一个或多个文件路径,它们之间用空格分隔。localdst 用于指定本地文件系统的目录。

接下来,演示如何将/data 目录下的文件 app_log_20251021.log 和 app_log_20251022.log.tmp.copy 下载到本地文件系统的/opt 目录中,具体命令如下。

```
hdfs dfs -get /data/app_log_20251021.log \
/data/app_log_20251022.log.tmp.copy /opt
```

上述命令执行完成后,查看本地文件系统中/opt 目录的内容,如图 3-21 所示。

从图 3-21 可以看出,本地文件系统的/opt 目录中已存在文件 app_log_20251021.log 和 app_log_20251022.log.tmp.copy。

10. -head

-head 的语法格式如下。

```
hdfs dfs -head <file>
```

上述语法格式中,file 用于指定文件路径。