第3章

HDFS的介绍和简单操作

HDFS是Hadoop原生的文件系统，主要用于存储数据，本章将对HDFS的基础内容进行介绍。本章内容安排如下。

3.1　Hadoop分布式文件系统(HDFS)

介绍HDFS的概念。

3.2　HDFS的原理

介绍HDFS的工作原理。

3.3　HDFS写操作

介绍HDFS的写操作。

3.4　HDFS读操作

介绍HDFS的读操作。

3.5　HDFS删除操作

介绍HDFS的删除操作。

3.6　HDFS常用命令

介绍HDFS的常用命令。

3.7　实验

对HDFS的简单实验进行介绍。

通过本章的学习，读者将对HDFS的概念、工作原理有初步的了解，学习到HDFS一些常用的命令，通过简单的实验，加深读者对HDFS的掌握情况。

3.1　Hadoop 分布式文件系统(HDFS)

HDFS(Hadoop Distributed File System)是一个分布式文件系统,具有高容错性,适合部署在低成本的机器上,并提供了高吞吐量的数据访问,非常适合大规模数据集上的应用。

HDFS 具有以下三个特点。

(1) 高吞吐量性: HDFS 的每个块分布在不同的节点上,在用户访问时,HDFS 会通过计算最近使用和访问量最小的服务器,将其提供给用户。由于 Block 在不同的节点上都有备份,所以速度和效率是比较快的,另外,在数据节点上会对区块进行缓存,提高了整体性能。

(2) 高容错性: 系统故障是不可避免的,如何做到故障之后的数据恢复和容错处理是至关重要的。HDFS 将数据进行多份复制并且分布到物理位置的不同服务器上,进行数据校验功能,后台的连续自检数据功能,使得高容错性得以实现。

(3) 可扩展性: 支持新的节点向外扩展,增加集群的容量,并且不需要人为干预。

3.2　HDFS 的原理

从整体上看,HDFS 主要由三个组件构成: NameNode, SecondaryNameNode, DataNode。HDFS 是以 Master/Slave 模式运作的,其中,上述的 NameNode 和 SecondaryNameNode 运行在 Master 节点上,DataNode 运行在 Slave 节点上。

NameNode 和 DataNode 架构如图 3.1 所示。

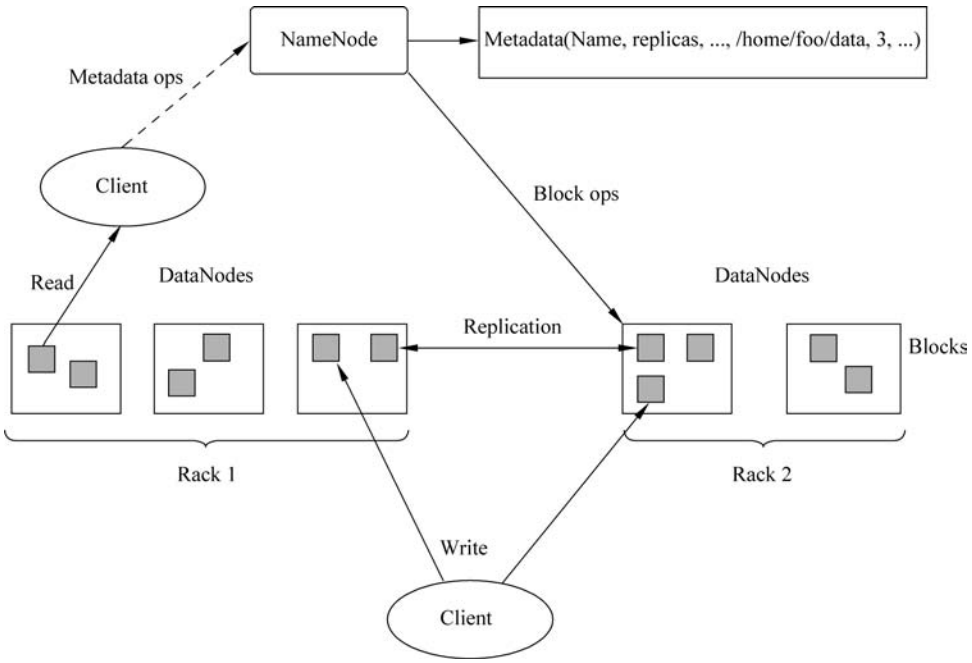

图 3.1　NameNode 和 DataNode 架构

1. NameNode

当一个客户端请求一个文件或者存储一个文件时，首先需要知道具体是到哪个DataNode上进行存取，获得这些信息后，客户端再直接和这个DataNode进行交互，而这些信息的维护者就是NameNode。

如果丢失了管理节点，存储在数据节点上的区块就会失去作用，因为没有办识别区块内的文件。因此管理节点的高可用性和元数据的备份在任何Hadoop集群中都十分重要。

NameNode启动后会进入一个称为安全模式的特殊状态。处于安全模式的NameNode是不会进行数据块的复制的。NameNode从所有的DataNode接收心跳信号和块状态报告。块状态报告包括某个DataNode所有的数据块列表。每个数据块都有一个指定的最小副本数。当NameNode检测确认某个数据块的副本数目达到这个最小值时，那么该数据块就会被认为是副本安全的；在一定百分比(这个参数可配置)的数据块被NameNode检测确认是安全之后(加上一个额外的30s等待时间)，NameNode将退出安全模式状态。接下来它会确定还有哪些数据块的副本没有达到指定数目，并将这些数据块复制到其他DataNode上。

2. DataNode

DataNode是HDFS中的Worker节点，它负责存储数据块。HDFS中文件不会直接进行存储，而是会将一个大文件划分为多个大区块分布在集群中。每个区块大小为128MB(也可以自定义每块的大小)。每个块会被复制3份，以便应对故障和数据的遗失情况。通过这种方式，文件等于有了多个备份，所以实现了高容错性，对数据的保护很有帮助。

3. SecondaryNameNode

SecondaryNameNode是为了解决NameNode重启速度而存在的。需要注意的是，SecondaryNameNode并不是NameNode的备份。所有HDFS文件的元信息都保存在NameNode的内存中。在NameNode启动时，它首先会加载fsimage文件(保存了上一个检查点的HDFS的元信息)到内存中，在系统运行期间，所有对NameNode的操作也都保存在了内存中，同时为了防止数据丢失，这些操作又会不断被持久化到本地edits文件(保存从上一个检查点开始发生的HDFS元信息的改变信息)中。显然，在NameNode重启的过程中，edits文件需要和fsimage文件合并到一起，以便在下次使用时直接加载fsimage文件到内存。但是在这个过程中，合并又会影响到Hadoop重启的速度。所以出现了SecondaryNameNode。

这种合并过程的步骤如下。

(1) 合并之前告知NameNode把所有的操作写到新的edites文件并将其命名为edits.new。

(2) SecondaryNameNode从NameNode请求fsimage文件和edits文件。

(3) SecondaryNameNode把fsimage文件和edits文件合并成新的fsimage文件。

(4) NameNode从SecondaryNameNode获取合并好的新的fsimage文件并将旧的文件替换掉，并把edits文件用第一步创建的edits.new文件替换掉。

(5) 更新 fstime 文件中的检查点。

如图 3.2 所示是一个 HDFS 存储文件的例子。当存储 File 1 时，因为不足 128MB(这里对于块的划分用默认设置)，所以直接被存储在单独区块里，因为需要备份，区块中的数据会被在节点 1,2,3 中进行复制。对于 File 2，因为大于 128MB，需要分成两个块，然后进行复制，分别备份到三个不同的节点上。

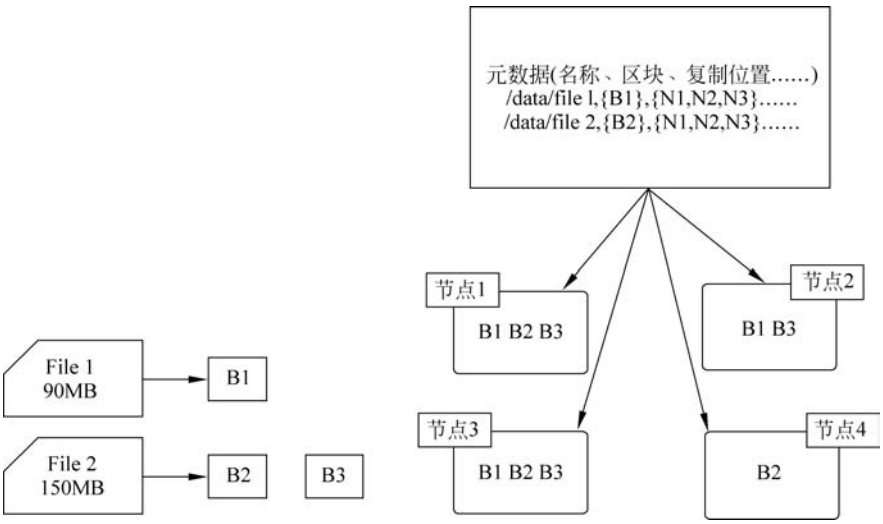

图 3.2　HDFS 存储文件例子

数据块是磁盘读写的基本单位，与普通文件系统类似，HDFS 默认数据块大小为 128MB，但是反观磁盘块的大小一般为 512B。相较而言，HDFS 块显得很大，这是因为增大块可以减少寻址时间与文件传输时间的比例，所以 HDFS 更适合于大数据的吞吐。但是这样的话少量数据的处理就会较慢。当然，磁盘块太大也不好，因为 MapReduce 通常以块为单位作为输入，块过大会导致整体任务数量过小，降低作业处理速度。

3.3　HDFS 写操作

客户端要向 HDFS 写数据，首先要跟 NameNode 通信以确认可以写文件并获得接收文件 Block 的 DataNode，然后客户端按顺序将文件的 Block 逐个传递给相应的 DataNode，并由接收到 Block 的 DataNode 负责向其他 DataNode 复制 Block 的副本。具体流程(如图 3.3 所示)描述如下。

(1) 客户端将文件写入本地磁盘的临时文件中。

(2) 当临时文件大小达到一个 Block 大小时，HDFS Client 通知 NameNode，申请写入文件。

(3) NameNode 在 HDFS 的文件系统中创建一个文件，并把该 Block ID 和要写入的 DataNode 的列表返回给客户端。

(4) 客户端收到这些信息后，将临时文件写入 DataNodes。

(5) 文件写完后(客户端关闭)，NameNode 提交文件。

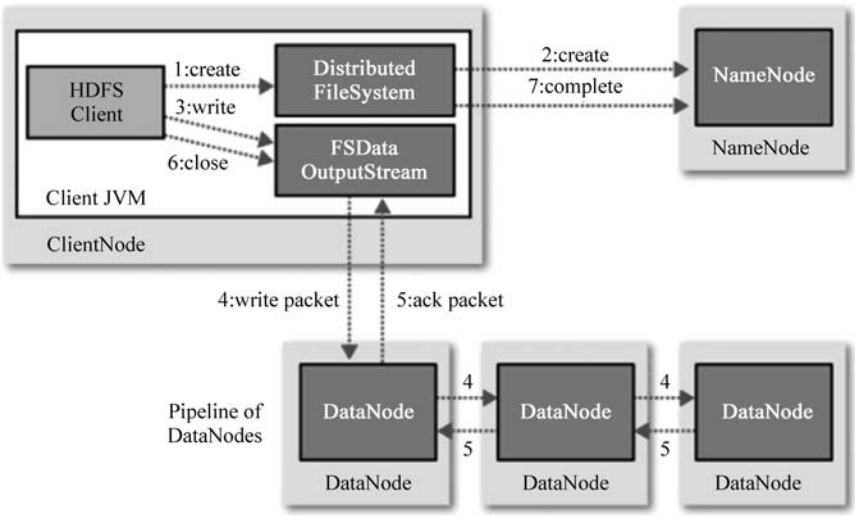

图 3.3 HDFS 写操作流程

在客户端写入 DataNode 时，通过 Pipeline(流水线)的方式进行复制，首先，客户端将文件内容写入第一个 DataNode(一般以 4kb 为单位进行传输)，第一个 DataNode 接收后，将数据写入本地磁盘，同时也传输给第二个 DataNode，后面的 DataNode 接收完数据后，都会发送一个确认给前一个 DataNode，最终第一个 DataNode 返回确认给客户端。客户端接收到整个 Block 的确认后，会向 NameNode 发送一个最终的确认信息。每个 Block 都会有一个校验码，并存放到独立的文件中，以便读的时候来验证其完整性。如果写入某个 DataNode 失败，数据会继续写入其他的 DataNode。然后 NameNode 会找另外一个好的 DataNode 继续复制，以保证冗余性。

机架感知：HDFS 采用一种称为机架感知的策略来改进数据的可靠性、可用性和网络带宽的利用率。大型 HDFS 实例一般运行在跨越多个机架的计算机组成的集群上，不同机架上的两台机器之间的通信需要经过交换机。在大多数情况下，同一个机架内的两台机器间的带宽会比不同机架的两台机器间的带宽大。通过一个机架感知的过程，NameNode 可以确定每个 DataNode 所属的机架 ID。一个简单但没有优化的策略就是将副本存放在不同的机架上。这样可以有效防止当整个机架失效时数据的丢失，并且允许读数据的时候充分利用多个机架的带宽。这种策略设置可以将副本均匀分布在集群中，有利于当组件失效情况下的负载均衡。但是，因为这种策略的一个写操作需要传输数据块到多个机架，增加了写的代价。在大多数情况下，副本系数是 3，HDFS 的存放策略是将一个副本存放在本地机架的节点上，一个副本放在同一机架的另一个节点上，最后一个副本放在不同机架的节点上。这种策略减少了机架间的数据传输，这就提高了写操作的效率。机架的错误远远比节点的错误少，所以这个策略不会影响到数据的可靠性和可用性。与此同时，因为数据块只放在两个(不是三个)不同的机架上，所以此策略减少了读取数据时需要的网络传输总带宽。在这种策略下，副本并不是均匀分布在不同的机架上。三分之一的副本在一个节点上，三分之二的副本在一个机架上，其他副本均匀分布在剩下的机架中，这一策略在不损害数据可靠性和读取性能的情况下改进了写的性能。

3.4　HDFS 读操作

客户端将要读取的文件路径发送给 NameNode，NameNode 获取文件的元信息（主要是 Block 的存放位置信息）返回给客户端，客户端根据返回的信息找到相应 DataNode 逐个获取文件的 Block 并在客户端本地进行数据追加合并从而获得整个文件。具体过程（如图 3.4 所示）描述如下。

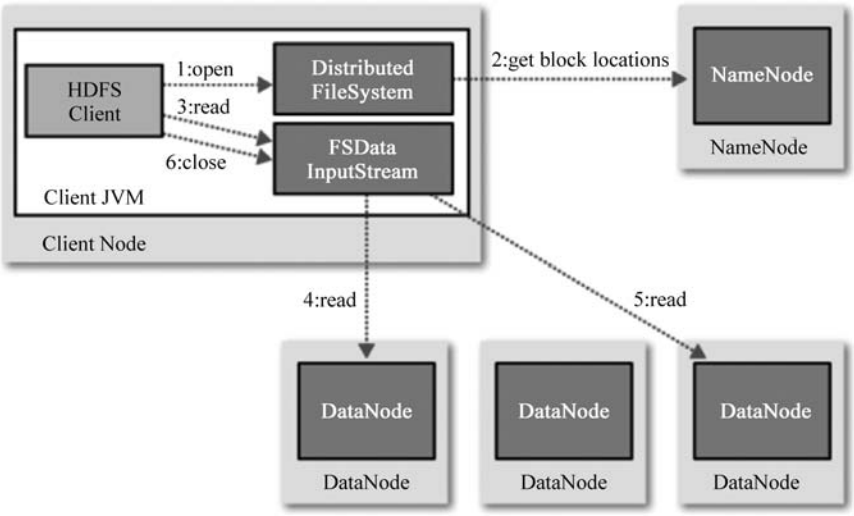

图 3.4　HDFS 读操作流程

（1）客户端向 NameNode 发起 RPC 调用，请求读取文件数据。

（2）NameNode 检查文件是否存在，如果存在则获取文件的元信息（Block ID 以及对应的 DataNode 列表）。

（3）客户端收到元信息后选取一个网络距离最近的 DataNode，依次请求读取每个数据块；客户端首先要校检文件是否损坏，如果损坏，客户端会选取另外的 DataNode 请求。

（4）DataNode 与客户端建立 Socket 连接，传输对应的数据块，客户端收到数据缓存到本地，之后写入文件。

（5）依次传输剩下的数据块，直到整个文件合并完成。

3.5　HDFS 删除操作

HDFS 删除数据流程相对简单，客户端向 NameNode 发起 RPC 调用，请求删除文件。NameNode 检查合法性。NameNode 查询文件相关元信息，向存储文件数据块的 DataNode 发出删除请求。DataNode 删除相关数据块。NameNode 返回结果给客户端。

需要注意的是，当用户或应用程序删除某个文件时，这个文件并没有立刻从 HDFS 中删除。实际上，HDFS 会将这个文件重命名后转移到/trash 目录。只要文件还在/trash 目录中，该文件就可以被迅速地恢复。文件在/trash 中保存的时间是可配置的，当超过这个时间时，NameNode 就会将该文件从名字空间中删除。删除文件会使得该文件

相关的数据块被释放。从用户删除文件到 HDFS 空闲空间的增加之间会有一定时间的延迟。只要被删除的文件还在“/trash”目录中，用户就可以恢复这个文件。如果用户想恢复被删除的文件，可以浏览“/trash”目录找回该文件。“/trash”目录仅保存被删除文件的最后副本。

3.6 HDFS 常用命令

HDFS 一些基本的命令格式如下。

```
hadoop fs -cmd args
```

说明：cmd 为具体命令，args 为参数。

```
hadoop fs -ls /                          #查看“/”目录下的文件
hadoop fs -mkdir /user/hadoop            #新建文件夹
hadoop fs -put a.txt /user/hadoop/       #将“a.txt”上传到“/user/hadoop/”
hadoop fs -get /user/hadoop/a.txt /      #获取“a.txt”文件
hadoop fs -cat /user/hadoop/a.txt        #查看“/user/hadoop/a.txt”的内容
hadoop fs -rm /user/hadoop/a.txt         #删除“/user/hadoop/a.txt”
hadoop fs -tail /user/hadoop/a.txt       #查看“/user/hadoop/a.txt”最后 1000 行
```

3.7 实验

为了帮助读者更好地掌握 HDFS，本节将对 HDFS 的简单操作进行介绍。

如图 3.5 所示，在浏览器中输入主机的地址和端口号“50070”，在出现的界面中选择 Browse the file system 菜单项，可以浏览 HDFS 的文件结构信息。HDFS 的文件结构信息如图 3.6 所示。

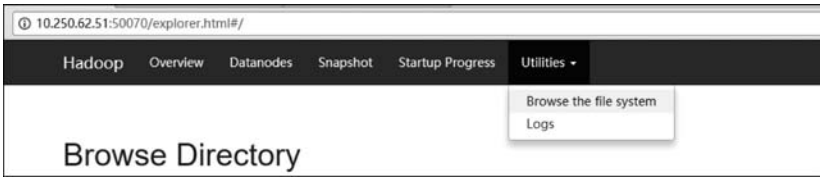

图 3.5 浏览 HDFS

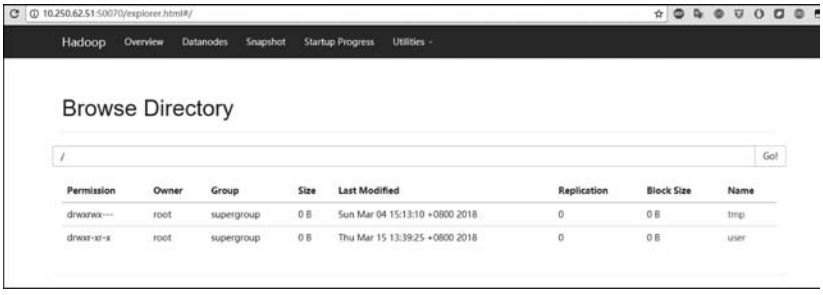

图 3.6 HDFS 中的文件结构信息

3.7.1　创建目录

1. 创建目录的基本命令

```
用法：./hdfs  dfs －mkdir 目录名
举例：./hdfs  dfs －mkdir  /user
```

2. 循环创建多级菜单

```
用法：./hdfs  dfs － mkdir －p /目录名/二级目录名
举例：[root@hadoop01 bin]# ./hdfs dfs －mkdir －p /usr/data
```

3.7.2　上传文件命令

使用 HDFS 的上传文件命令，可以将单个文件或多个文件从本地文件系统复制到目标文件系统；也可以从标准输入读取输入并写入目标文件系统。

```
用法：hdfs dfs －put <localsrc>…<dst>
举例：[root@hadoop01 bin]# ./hdfs dfs －put /etc/profile /usr/data
```

输入上述命令后，在网页中可以看到文件已经成功上传，如图 3.7 所示。

图 3.7　文件上传成功

单击相应的文件名可以查看文件信息，如图 3.8 所示。

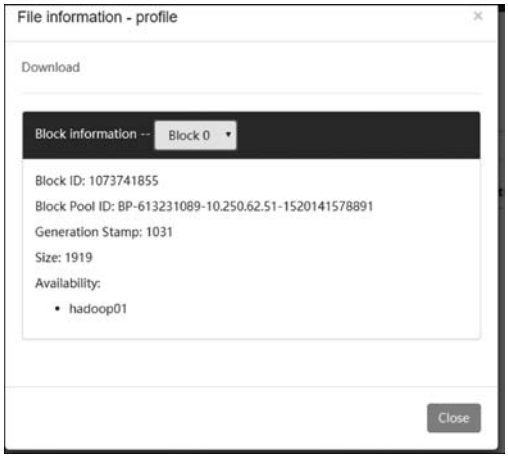

图 3.8　文件信息

3.7.3 罗列 HDFS 上的文件

```
用法：./hdfs dfs -ls [-R] <args>
说明：-R 选项将通过目录结构递归地返回 stat.
举例：./hdfs dfs -ls /usr/data
```

使用上述命令，将得到在目录/usr/data 下的所有文件名，如图 3.9 所示。

```
[root@hadoop01 bin]# ./hdfs dfs -ls /usr/data
18/03/14 22:49:20 WARN util.NativeCodeLoader: Unable to load native-hadoop library for
ere applicable
Found 1 items
-rw-r--r--   3 root supergroup       1919 2018-03-14 22:46 /usr/data/profile
[root@hadoop01 bin]#
```

图 3.9 罗列目录/usr/data 下的文件

而使用其他选项，如：

```
./hdfs dfs -ls -R /usr
```

将会得到以下结果：

```
[root@hadoop01 bin]# ./hdfs dfs -ls -R /usr
18/03/14 22:51:45 WARN util.NativeCodeLoader: Unable to load native-hadoop library for
your platform... using builtin-java classes where applicable
drwxr-xr-x   - root supergroup          0 2018-03-14 22:46 /usr/data
-rw-r--r--   3 root supergroup       1919 2018-03-14 22:46 /usr/data/profile
[root@hadoop01 bin]#
```

3.7.4 查看 HDFS 里某一个文件

```
用法：hdfs dfs -cat URI [URI …]
```

使用该命令，可以将源路径复制到标准输出。

```
举例：[root@hadoop01 bin]# ./hdfs dfs -cat /usr/data/profile
```

3.7.5 将 HDFS 中的文件复制到本地

```
用法：hdfs dfs -get [-ignorecrc] [-crc] <src> <localdst>
```

使用该命令可以将文件复制到本地文件系统。未通过 CRC 检查的文件可能会使用-ignorecrc 选项复制。文件和 CRC 可以使用-crc 选项复制。

```
举例：[root@hadoop01 bin]# ./hdfs dfs -get /usr/data/profile /usr/software/
```

使用上述命令后，在目录中罗列文件可以得到如图 3.10 所示的结果。

```
[root@hadoop01 software]# ls
hadoop  hive  jdk  kafka  profile  spark
[root@hadoop01 software]#
```

图 3.10　将文件 profile 复制到本地

3.7.6　递归删除 HDFS 下的文档

递归删除 HDFS 下的文档命令如下。

```
用法：./hdfs dfs -rmr /user
举例：
[root@hadoop01 bin]# ./hdfs dfs -rmr /user
rmr:DEPRECATED:Please use 'rm -r' instead.
18/03/14 22:37:52 WARN util.NativeCodeLoader:Unable to load native-hadoop library for
your platform... using builtin-java classes where applicable
Deleted /user
```

示例的命令结果如图 3.11 所示。

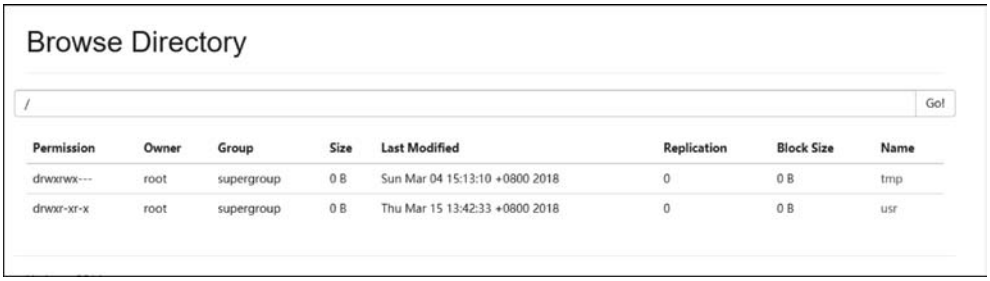

图 3.11　递归删除目录/user 下的文档