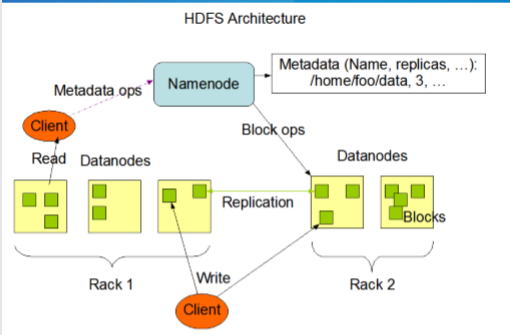
# 知识点回顾：

## HDFS架构：



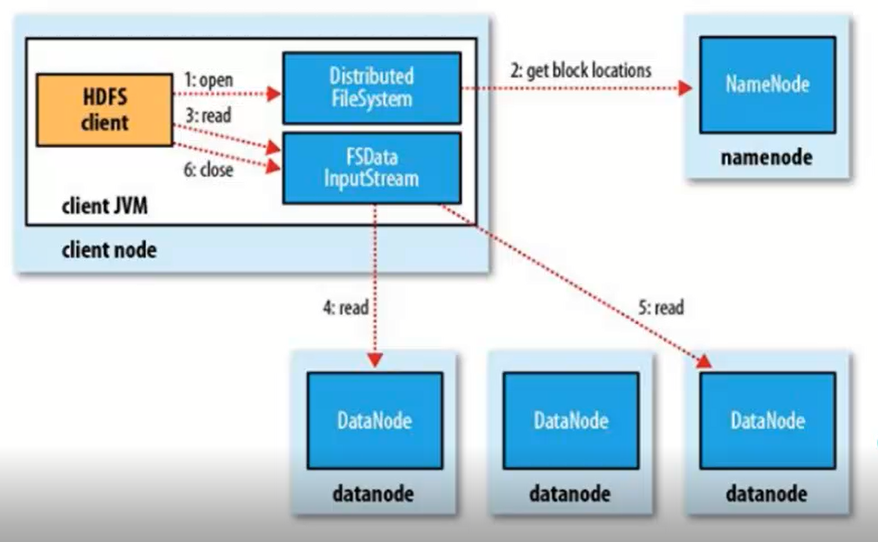
Client：文件切分。文件上传 HDFS 的时候，Client 将文件切分成 一个一个的Block，然后进行存储；与 NameNode 交互，获取文件的位置信息；与 DataNode 交互，读取或者写入数据；Client 提供一些命令来管理 HDFS，比如启动或者关闭HDFS；Client 可以通过一些命令来访问 HDFS。

NameNode：就是 master，它是一个主管、管理者。1、管理 HDFS 的名称空间。2、管理数据块（Block）映射信息 3、配置副本策略 4、处理客户端读写请求。

DataNode：就是Slave。NameNode 下达命令，DataNode 执行实际的操作。 存储实际的数据块。 执行数据块的读/写操作。

Secondary NameNode：并非 NameNode 的热备。当NameNode 挂掉的时候，它并不能马上替换 NameNode 并提供服务。 1、辅助 NameNode，分担其工作量。定期合并 fsimage和fsedits，并推送给NameNode。在紧急情况下，可辅助恢复 NameNode。

## HDFS读写流程：

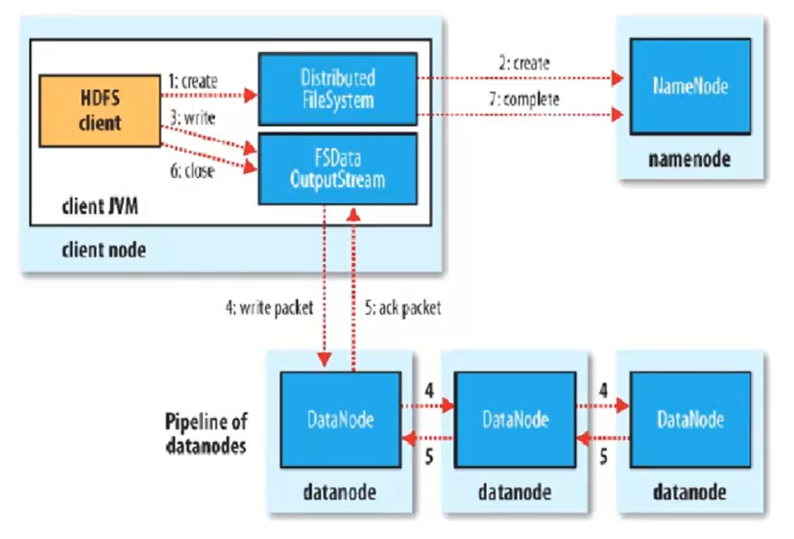
[](http://www.sukidesu.top/usr/uploads/2019/09/1268692364.png)

1、客户端发送请求，调用 DistributedFileSystem API 的 open 方法发送请求到 Namenode，获得 block 的位置信息，因为真正的 block 是存在 Datanode 节点上的，而 namenode 里存放了 block 位置信息的元数据。

2、Namenode 返回所有 block 的位置信息，并将这些信息返回给客户端。

3、客户端拿到 block 的位置信息后调用 FSDataInputStream API 的 read 方法并行的读取 block 信息，图中 4 和 5 流程是并发的，block 默认有 3 个副本，所以每一个 block 只需要从一个副本读取就可以。

4、datanode 返回给客户端。

[](http://www.sukidesu.top/usr/uploads/2019/09/4180750734.png)

1、客户端发送请求，调用 DistributedFileSystem API 的 create 方法去请求 namenode，并告诉 namenode 上传文件的文件名、文件大小、文件拥有者。

2、namenode 根据以上信息算出文件需要切成多少块 block，以及 block 要存放在哪个 datanode 上，并将这些信息返回给客户端。

3、客户端调用 FSDataInputStream API 的 write 方法首先将其中一个 block 写在 datanode 上，每一个 block 默认都有 3 个副本，并不是由客户端分别往 3 个 datanode 上写 3 份，而是由

已经上传了block的datanode产生新的线程，由这个namenode按照放置副本规则往其它datanode写副本，这样的优势就是快。

4、写完后返回给客户端一个信息，然后客户端在将信息反馈给 namenode。

5、需要注意的是上传文件的拥有者就是客户端上传文件的用户名，举个例子用 windows 客户端上传文件，那么这个文件的拥有者就是 administrator，和 linux 上的系统用户名不是一样的。

## 副本策略

HDFS的存放策略是将一个副本存放在本地机架节点上，一个副本存放在同一个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，提高了写操作的效率。

## HDFS Shell:

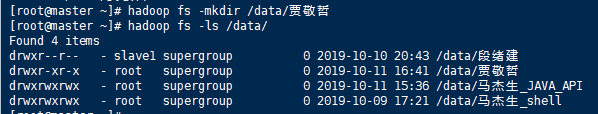
感受：Tab键无法使用 特别不好用

# HDFS操作练习：

1. 创建目录：

命令： hadoop fs -mkdir /data/贾敬哲

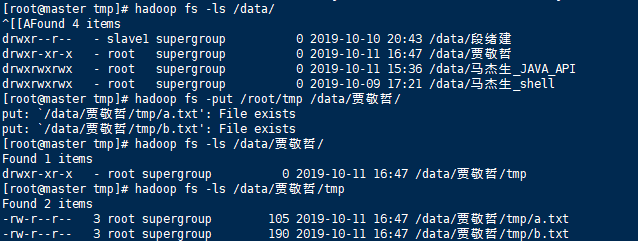
截图：



1. 上传目录：

命令：hadoop fs -put /root/tmp /data/贾敬哲/

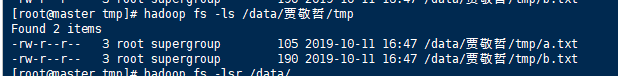
截图：原目录下的 tmp/a.txt tmp/b.txt 均已上传到hdfs



1. 查看文件列表：

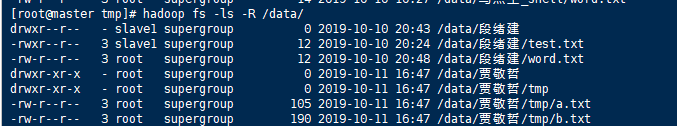
命令: hadoop fs -ls /data/贾敬哲/tmp

截图:



命令 递归查看目录树

截图：



1. 统计文件数:

命令：hadoop fs -count /data/贾敬哲/

截图：

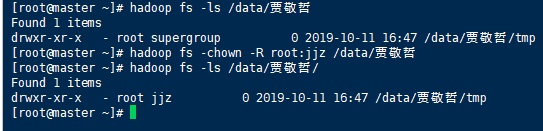


由图可知，/data/贾敬哲/ 目录下 存在两个目录 两个文件 文件总计大小 295

1. 修改用户

命令：hadoop fs -chown -R root:jjz /data/贾敬哲

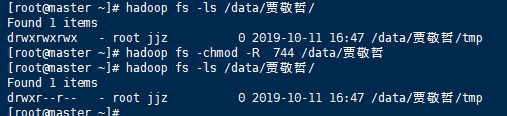
截图：



1. 修改用户权限

命令：hadoop fs -chmod -R 744 /data/贾敬哲

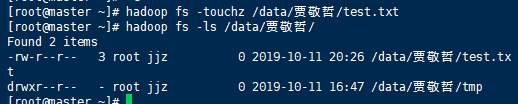
截图：



1. 创建txt文件

命令：hadoop fs -touchz /data/贾敬哲/word.txt

截图：



1. 追加文字

命令：hadoop fs -appendToFile /root/w.txt /data/贾敬哲/word.txt

截图：



1. 查看文字

命令：hadoop fs -cat /data/贾敬哲/word.txt

截图：

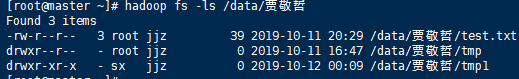


1. JAVA\_API
   1. 创建目录：

代码：

public void mkdir() {  
 Configuration conf = new Configuration();  
 FileSystem fs;  
 try {  
 fs = FileSystem.*get*(conf);  
 Path srcPath = new Path("/data/贾敬哲/tmp1");  
 boolean res = fs.mkdirs(srcPath);  
 if (res) {  
 System.*out*.println("创建文件夹成功");  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
}

结果：

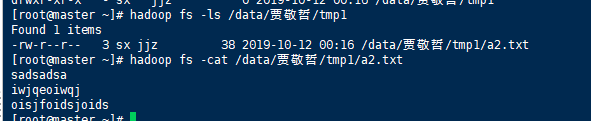


* 1. 上传目录：

代码：

public void put() {  
 Configuration conf = new Configuration();  
 FileSystem fs;  
 try {  
 fs = FileSystem.*get*(conf);  
 Path srcPath = new Path("H:\\testhdfs\\src\\main\\resources\\a2.txt");  
 Path dPath = new Path("/data/贾敬哲/tmp1");  
 fs.copyFromLocalFile(srcPath, dPath);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
}

结果：

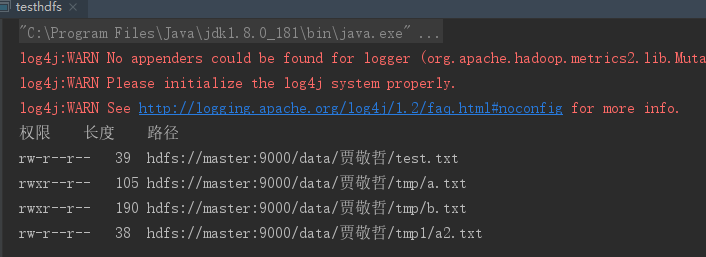


* 1. 查看文件列表

代码：

public void ls(String path) {  
 Configuration conf = new Configuration();  
 FileSystem fs;  
 try {  
 fs = FileSystem.*get*(conf);  
 Path srcPath = new Path(path);  
 RemoteIterator<LocatedFileStatus> res = fs.listFiles(srcPath, true);  
 System.*out*.println("权限\t长度\t路径");  
 while (res.hasNext()) {  
 LocatedFileStatus file = res.next();  
 System.*out*.println(file.getPermission() + "\t" + file.getLen() + "\t" + file.getPath());  
 }  
  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
}

结果：

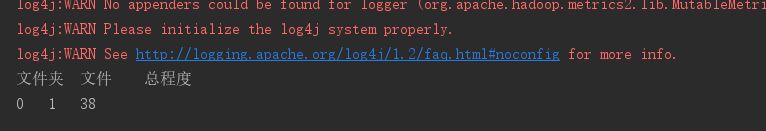


* 1. 统计文件数

代码：

public void count() {  
 Configuration conf = new Configuration();  
 FileSystem fs;  
 try {  
 fs = FileSystem.*get*(conf);  
 Path srcPath = new Path("/data/贾敬哲/tmp1");  
 RemoteIterator<LocatedFileStatus> res = fs.listFiles(srcPath, true);  
 int sumLen = 0;  
 int num = 0;  
 int folder = 0;  
 while (res.hasNext()) {  
 num++;  
 LocatedFileStatus file = res.next();  
 sumLen += file.getLen();  
 if (file.isDirectory()) {  
 folder++;  
 }  
  
 }  
 System.*out*.println("文件夹\t文件\t总程度");  
 System.*out*.println(String.*valueOf*(folder) + "\t" + String.*valueOf*(num) + "\t" + String.*valueOf*(sumLen));  
  
  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
}

结果：

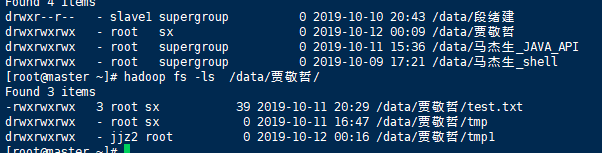


* 1. 修改用户

代码：

public void chown() {  
 Properties properties = System.*getProperties*();  
 properties.setProperty("HADOOP\_USER\_NAME", "root");  
 Configuration conf = new Configuration();  
 FileSystem fs;  
 try {  
 fs = FileSystem.*get*(conf);  
 Path srcPath = new Path("/data/贾敬哲/tmp1");  
 fs.setOwner(srcPath, "jjz2", "root");  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
}

结果：tmp1文件夹所有者被修改了

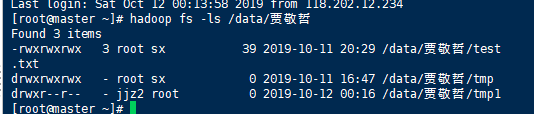


* 1. 修改用户权限

代码：

public void chmod() {  
 Properties properties = System.*getProperties*();  
 properties.setProperty("HADOOP\_USER\_NAME", "root");  
  
 Configuration conf = new Configuration();  
 conf.set("user","root");  
 FileSystem fs;  
 try {  
 fs = FileSystem.*get*(conf);  
  
 Path srcPath = new Path("/data/贾敬哲/tmp1");  
 FsPermission fp = new FsPermission("744");  
 fs.setPermission(srcPath, fp);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
}

结果：

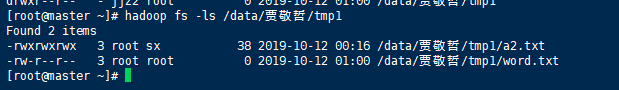


* 1. 创建txt文件

代码：

public void touchz() {  
 Properties properties = System.*getProperties*();  
 properties.setProperty("HADOOP\_USER\_NAME", "root");  
 Configuration conf = new Configuration();  
 FileSystem fs;  
 try {  
 fs = FileSystem.*get*(conf);  
  
 Path srcPath = new Path("/data/贾敬哲/tmp1/word.txt");  
 fs.create(srcPath);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
}

结果：word.txt添加了进去



* 1. 追加文字

代码：

public void add() {  
 Properties properties = System.*getProperties*();  
 properties.setProperty("HADOOP\_USER\_NAME", "root");  
 Configuration conf = new Configuration();  
 FileSystem fs;  
 BufferedWriter writer = null;  
 try {  
 fs = FileSystem.*get*(conf);  
 Path srcPath = new Path("/data/贾敬哲/tmp1/word.txt");  
 FSDataOutputStream out = fs.append(srcPath);  
 writer = new BufferedWriter(new OutputStreamWriter(out));  
 writer.write("newData");  
 writer.newLine();  
 writer.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
}

结果：newData加入到了文件中

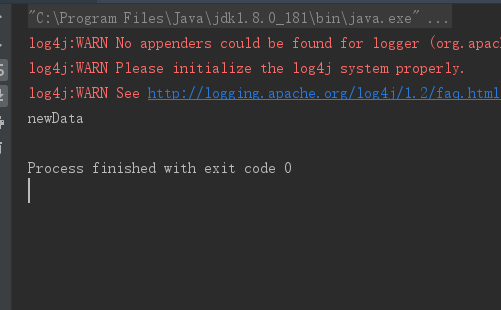


* 1. 查看文字

代码：

public void cat() {  
 Configuration conf = new Configuration();  
 BufferedReader reader = null;  
 String line = null;  
 try {  
 FileSystem HDFS = FileSystem.*get*(conf);  
 FSDataInputStream in = HDFS.open(new Path("/data/贾敬哲/tmp1/word.txt"));  
 reader = new BufferedReader(new InputStreamReader(in));  
 while ((line = reader.readLine()) != null) {  
 System.*out*.println(line);  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 } finally {  
 if (reader != null) {  
 try {  
 reader.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
}

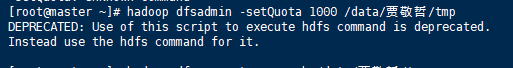
结果：

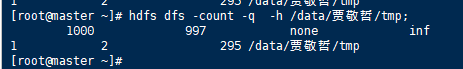


1. 配额设置

命令：hadoop dfsadmin -setQuota 1000 /data/贾敬哲/tmp

截图:





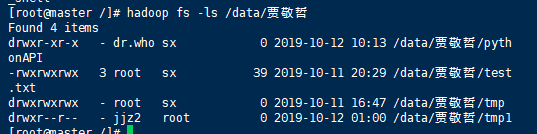
1. 创新：

PythonAPI操作hdfs

创建目录：

def mkdir():  
 client.makedirs('/data/贾敬哲/pythonAPI')

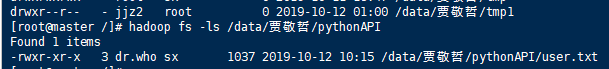
截图：



上传文件：

def put():  
 client.upload("/data/贾敬哲/pythonAPI/",r"C:\Users\sx\Desktop\nike-account-creator-master\user.txt")

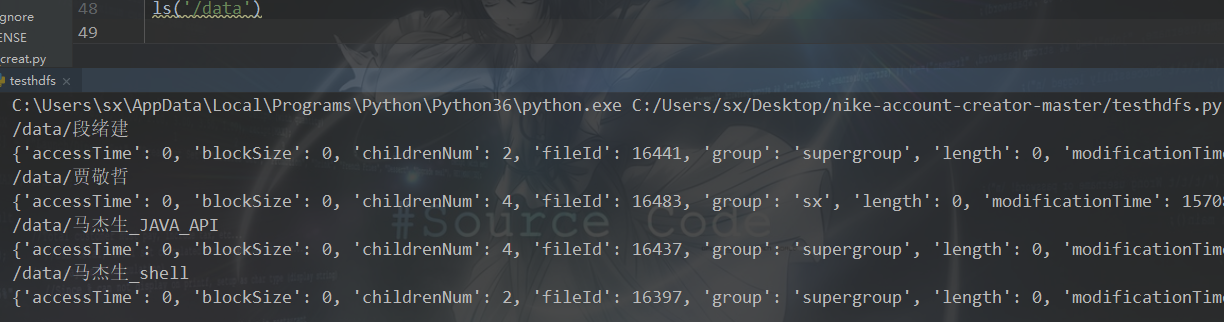
截图：



列出目录：

def ls(path,R = False):  
 *""" 递归 """* res = client.list(path,True)  
 if R==False:  
 for i in res:  
 print(path+"/"+i[0])  
 print(i[1])  
  
 else:  
 for i in res:  
 if i[1]['type'] == "FILE":  
 print(path+"/"+i[0])  
 print(i[1])  
 if i[1]['type'] == "DIRECTORY":  
 ls(path+"/"+i[0])

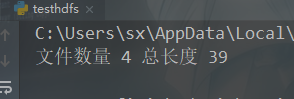
截图：



统计文件数：

def count():  
 res = client.list("/data/贾敬哲",True)  
 filenum = 0  
 lennum = 0  
 for i in res:  
 filenum+=1  
 lennum += int(i[1]['length'])  
 print("文件数量",filenum,"总长度",lennum)

截图：

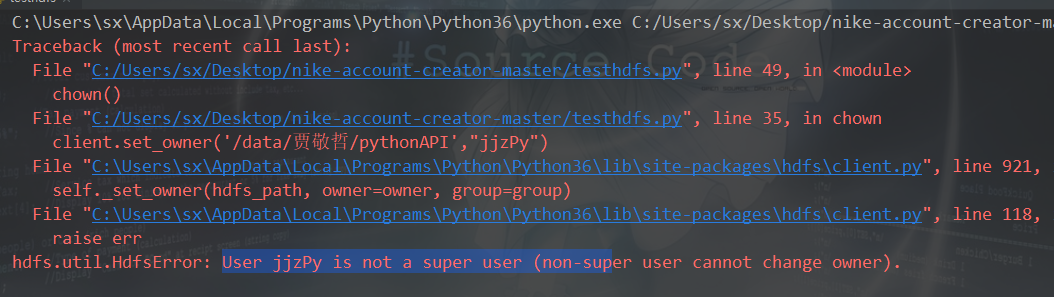


修改文件所有者：

因为权限不够，仍未解决

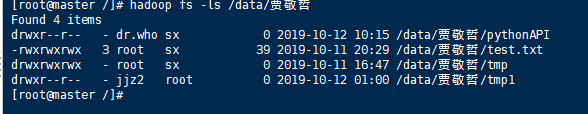
def chown():  
 client.set\_owner('/data/贾敬哲/pythonAPI',"jjzPy")

截图：



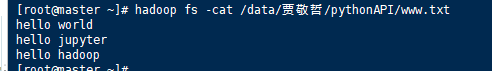
修改文件权限：

def chmod():  
 client.set\_permission('/data/贾敬哲/pythonAPI',"744")



增加文件内容：

def add():  
 client.write('data/贾敬哲/pythonAPI/user.txt', "newData", overwrite=False, append=True)



查看文件

def cat():  
 lines = []  
 with client.read('data/贾敬哲/pythonAPI/user.txt', encoding='utf-8', delimiter="\n") as reader:  
 for line in reader:  
 # pass  
 # print line.strip()  
 lines.append(line.strip())  
 print("".join(lines))

截图

