**广 州 商 学 院**

**实验报告（第4次）**

实验名称 HDFS编程 实验时间 第 周

同组同学 小组分工

**一、实验目的**

1. 理解HDFS在Hadoop体系结构中的角色；
2. 熟练使用HDFS操作常用的Shell命令；
3. 熟悉HDFS操作常用的Java API。

**二、实验仪器设备或材料**

1. Centos7或其他可用版本；
2. Hadoop3.3.5或其他可用版本；
3. JDK1.8或其他可用版本。

**三、实验原理**

1. 关于HDFS的shell命令
2. 关于HDFS的常用Java API
3. HDFS读取/写入数据过程

**四、实验内容与步骤**

1. JAVA编程配置，并实现数据读写过程
2. Eclipse配置

上传eclipse-4.7.0-linux.gtk.x86\_64.tar至/home/hadoop/下载

接着在终端解压

cd /home/hadoop/下载

tar -zxvf ./eclipse-4.7.0-linux.gtk.x86\_64.tar.gz -C /opt/module/

启动：

cd /opt/module/eclipse/

./eclipse

启动后配置过程查看PPT

1. IntelliJ IDEA配置

查看IntelliJ IDEA配置连接Hadoop

先修改目录/user/root的权限为777

hadoop fs -chmod 777 /user/root

读：

import java.io.BufferedReader;

import java.io.InputStreamReader;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.FileSystem;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.fs.FSDataInputStream;

public class Chapter3 {

public static void main(String[] args) {

try {

Configuration conf = new Configuration();

conf.set("fs.defaultFS","hdfs://192.168.182.100:9000");

conf.set("fs.hdfs.impl","org.apache.hadoop.hdfs.DistributedFileSystem");

FileSystem fs = FileSystem.get(conf);

Path file = new Path("读取文件路径");

FSDataInputStream getIt = fs.open(file);

BufferedReader d = new BufferedReader(new InputStreamReader(getIt));

String content = d.readLine(); //读取文件一行

System.out.println(content);

d.close(); //关闭文件

fs.close(); //关闭hdfs

} catch (Exception e) {

e.printStackTrace();

} } }

写：

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.FileSystem;

import org.apache.hadoop.fs.FSDataOutputStream;

import org.apache.hadoop.fs.Path;

public class Chapter4 {

public static void main(String[] args) {

try {

Configuration conf = new Configuration();

conf.set("fs.defaultFS","hdfs://192.168.182.100:9000");

conf.set("fs.hdfs.impl","org.apache.hadoop.hdfs.DistributedFileSystem");

FileSystem fs = FileSystem.get(conf);

byte[] buff = "Hello world".getBytes(); // 要写入的内容

String filename = "/user/root/test.txt"; //要写入的文件名

FSDataOutputStream os = fs.create(new Path(filename));

os.write(buff,0,buff.length);

System.out.println("Create:"+ filename);

os.close();

fs.close();

} catch (Exception e) {

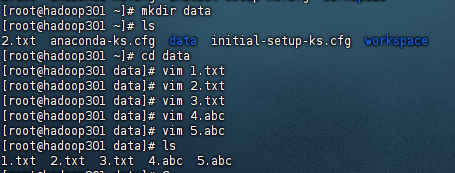
e.printStackTrace();

} } }

1. 利用JAVA编程实现以下功能（Eclipse与IntelliJ IDEA选一）
2. 将HDFS中指定文件的内容输出到终端中

import org.apache.hadoop.conf.Configuration;  
import org.apache.hadoop.fs.\*;  
import java.io.\*;  
  
public class CatFile {  
 */\*\*  
 \* 读取文件内容  
 \*/* public static void cat(Configuration conf, String remoteFilePath) throws IOException {  
 FileSystem fs = FileSystem.*get*(conf);  
 Path remotePath = new Path(remoteFilePath);  
 FSDataInputStream in = fs.open(remotePath);  
 BufferedReader d = new BufferedReader(new InputStreamReader(in));  
 String line = null;  
 while ((line = d.readLine()) != null)  
 System.*out*.println(line);  
 d.close();  
 in.close();  
 fs.close();  
 }  
 */\*\*  
 \* 主函数  
 \*/* public static void main(String[] args) {  
 Configuration conf = new Configuration();  
 conf.set("fs.default.name", "hdfs://192.168.182.100:9000");  
 String remoteFilePath = "/user/root/1.txt"; // HDFS路径  
 try {  
 System.*out*.println("读取文件: " + remoteFilePath);  
 CatFile.*cat*(conf, remoteFilePath);  
 System.*out*.println("\n读取完成");  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
}

1. 从指定目录中过滤出所有后缀名不为“.abc”的文件，对过滤之后的文件进行读取，并将这些文件的内容合并到新文件中。



文件1内容：11111

文件2内容：2222

以此类推



import java.io.IOException;

import java.io.PrintStream;

import java.net.URI;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.\*;

/\*\*

\* 过滤掉文件名满足特定条件的文件

\*/

class MyPathFilter implements PathFilter {

String reg = null;

MyPathFilter(String reg) {

this.reg = reg;

}

public boolean accept(Path path) {

if (!(path.toString().matches(reg)))

return true;

return false;

}

}

/\*\*\*

\* 利用FSDataOutputStream和FSDataInputStream合并HDFS中的文件

\*/

public class MergeFile {

Path inputPath = null; //待合并的文件所在的目录的路径

Path outputPath = null; //输出文件的路径

public MergeFile(String input, String output) {

this.inputPath = new Path(input);

this.outputPath = new Path(output);

}

public void doMerge() throws IOException {

Configuration conf = new Configuration();

conf.set("fs.defaultFS"," hdfs://192.168.182.100:9000");

conf.set("fs.hdfs.impl","org.apache.hadoop.hdfs.DistributedFileSystem");

FileSystem fsSource = FileSystem.get(URI.create(inputPath.toString()), conf);

FileSystem fsDst = FileSystem.get(URI.create(outputPath.toString()), conf);

//下面过滤掉输入目录中后缀为.abc的文件

FileStatus[] sourceStatus = fsSource.listStatus(inputPath,

new MyPathFilter(".\*\\.abc"));

FSDataOutputStream fsdos = fsDst.create(outputPath);

PrintStream ps = new PrintStream(System.out);

//下面分别读取过滤之后的每个文件的内容，并输出到同一个文件中

for (FileStatus sta : sourceStatus) {

//下面打印后缀不为.abc的文件的路径、文件大小

System.out.print("路径：" + sta.getPath() + " 文件大小：" + sta.getLen()

+ " 权限：" + sta.getPermission() + " 内容：");

FSDataInputStream fsdis = fsSource.open(sta.getPath());

byte[] data = new byte[1024];

int read = -1;

while ((read = fsdis.read(data)) > 0) {

ps.write(data, 0, read);

fsdos.write(data, 0, read);

}

fsdis.close();

}

ps.close();

fsdos.close();

}

public static void main(String[] args) throws IOException {

MergeFile merge = new MergeFile(

"补充",

"补充");

merge 补充;

}

}

**五、实验结果与分析**

（截取主要运行结果界面即可）

**六、结论与体会**

**七、教师评语**