Hadoop 分布式文件系统 HDFS

Java API 远程访问

侯在钱

目 录

上一节介绍的是在 Hadoop 环境内运行 Java 程序，访问 HDFS 系统。所以在运行 Java 程序时，必须把 Java 程序上传到 Hadoop 主机上才能运行。

本文介绍在 Eclipse 中编写客户端程序访问远程的 Hadoop HDFS 文件。可以在 Hadoop 系统之外的设备上运行 Java 程序，并能访问 HDFS。

1. 准备工作

在编写代码之前，需要准备开发环境。

在 Eclipse 中创建 Java Project，把 Java 需要调用的 Hadoop API 包拷贝到项目工程中，并且加入到 Java Build Path 中。这些 Jar 包有：

JAR 文件	所在目录
hadoop-common-2.6.0.jar	/hadoop-2.6.0/share/hadoop/common/

hadoop-hdfs-2.6.0.jar	/hadoop-2.6.0/share/hadoop/hdfs/
guava-11.0.2.jar	/hadoop-2.6.0/share/hadoop/common/lib/
hadoop-auth-2.6.0.jar	/hadoop-2.6.0/share/hadoop/common/lib/
slf4j-api-1.7.5.jar	/hadoop-2.6.0/share/hadoop/common/lib/
slf4j-log4j12-1.7.5.jar	/hadoop-2.6.0/share/hadoop/common/lib/
log4j-1.2.17.jar	/hadoop-2.6.0/share/hadoop/common/lib/
htrace-core-3.0.4.jar	/hadoop-2.6.0/share/hadoop/common/lib/
protobuf-java-2.5.0.jar	/hadoop-2.6.0/share/hadoop/common/lib/
所有以“commons-”打头的那些 jar 包	/hadoop-2.6.0/share/hadoop/common/lib/

注意：如果导入其它 Jar 包，有可能程序运行时会出错。

2. 远程写文件

编写客户端程序远程写 Hadoop HDFS 中的文件，参见如下。

运行此程序会报“Permission denied”，使用“hdfs dfs -chmod 777 /user”给指定目录设置读写权限，777 是把所有权限都放开。设置目录的读写权限后再运行 HDFS 的文件读写程序。除此方法外，还可以参照文本后面的第 13 节解决。

```
Configuration conf = new Configuration();
//此处的设置必须和core-site.xml配置文件中对应的参数值一致
conf.set("fs.defaultFS","hdfs://192.168.1.200:9000");

FileSystem fs = FileSystem.get(conf);

Path f = new Path("/user/test.txt");//文件路径
FSDataOutputStream out = fs.create(f);//创建输入流对象

String s = "Helo world! Remote!";
byte[] b = s.getBytes();
out.write( b );

out.close();
```

3. 远程读文件

编写客户端程序远程读取 Hadoop HDFS 中的文件，参见如下。

运行此程序会报“Permission denied”，使用“hdfs dfs -chmod 777 /user”给指定目录设置读写权限，777 是把所有权限都放开。设置目录的读写权限后再运行 HDFS 的文件读写程序。除此方法外，还可以参照文本后面的第 13 节解决。

```
package com.hadoop.hdfs.test;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FSDataInputStream;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;

public class HdfsRemoteRead {

    public static void main(String[] args) throws IOException {
        Configuration conf = new Configuration();
        //此处的设置必须和core-site.xml配置文件中对应的参数值一致
        conf.set("fs.defaultFS","hdfs://192.168.1.200:9000");

        FileSystem fs = FileSystem.get(conf);

        Path f = new Path("/user/test.txt");
        FSDataInputStream hdfsInputStream = fs.open(f);

        byte[] data = new byte[1024];
        int len = hdfsInputStream.read(data);
        while (len>0) {
            String s = new String(data, 0, len);
            System.out.println(s);
            len = hdfsInputStream.read(data);
        }

        hdfsInputStream.close();
    }
}
```

4. 创建空文件

```
Configuration conf = new Configuration();
//此处的设置必须和core-site.xml配置文件中对应的参数值一致
conf.set("fs.defaultFS","hdfs://192.168.1.200:9000");

FileSystem fs = FileSystem.get(conf);
Path p = new Path("/user/abc.txt");

fs.createNewFile(p);
fs.close();// 释放资源
```

5. 修改文件名

```
Configuration conf = new Configuration();
//此处的设置必须和core-site.xml配置文件中对应的参数值一致
conf.set("fs.defaultFS","hdfs://192.168.1.200:9000");

FileSystem fs = FileSystem.get(conf);
Path p1 = new Path("/user/test.txt");
Path p2 = new Path("/user/test1.txt");

fs.rename(p1, p2);


fs.close();// 释放资源
```

6. 上传文件或目录

```
Configuration conf = new Configuration();
//此处的设置必须和core-site.xml配置文件中对应的参数值一致
conf.set("fs.defaultFS","hdfs://192.168.1.200:9000");

// 加载默认配置

FileSystem fs = FileSystem.get(conf);

// 本地文件或目录

Path src = new Path("D:\\tmp\\abc");
```

```
// HDFS文件或目录
Path dst = new Path("/user/");

try {
   fs.copyFromLocalFile(src, dst);
} catch (IOException e) {
   e.printStackTrace();
}


fs.close();// 释放资源
```

7. 下载文件或目录

```
Configuration conf = new Configuration();
//此处的设置必须和core-site.xml配置文件中对应的参数值一致
conf.set("fs.defaultFS","hdfs://192.168.1.200:9000");

FileSystem fs = FileSystem.get(conf);
//HDFS上的目录或文件
Path p1 = new Path("/user/");
//本地的目录或文件。p1是目录，则p2也设置为目录。p1是文件，则p2也设置为文件。
Path p2 = new Path("D://tmp//hdfstest");
//fs.copyToLocalFile( p1, p2 ); //这种写法有错，需要4个参数，第1和第4都是布尔值
fs.copyToLocalFile(false, p1, p2, true);
fs.close();// 释放资源
```

8. 删除文件

```
Configuration conf = new Configuration();
//此处的设置必须和core-site.xml配置文件中对应的参数值一致
conf.set("fs.defaultFS","hdfs://192.168.1.200:9000");
FileSystem fs = FileSystem.get(conf);
Path p = new Path("/user/abc/1.txt");
fs.deleteOnExit(p);

fs.close();// 释放资源
```

9. 获得目录中的文件列表

```
package com.hadoop.hdfs.test;

import java.io.IOException;

import org.apache.hadoop.conf.*;
import org.apache.hadoop.fs.*;
import org.apache.hadoop.security.AccessControlException;

/**
 * 打印"/"根目录下的目录和文件，以及2级目录或文件。
 * @author HouZaiqian
 *
 */
public class PrintDirectory {

    /**
     * @param args
     * @throws IOException
     */
    public static void main(String[] args) throws IOException {
        //获取HDFS的conf对象
        Configuration conf = new Configuration();
        //此处的设置必须和core-site.xml配置文件中对应的参数值一致，并且主机名必须改成IP地址。
        conf.set("fs.defaultFS","hdfs://192.168.20.4:9000");

        //获取HDFS上的文件系统对象
        FileSystem hdfs = FileSystem.get(conf);

        //获取根目录
        Path rootPath = new Path("/");
        System.out.println("/");

        //获取根目录下的所有1级子目录或文件
        FileStatus[]  fileStatus = null;
        try {
            fileStatus = hdfs.listStatus(rootPath);
        }
        catch (AccessControlException e) {
            System.out.println("无权限访问这个目录："+rootPath.toString());
        }
        //循环打印子目录或文件
```

```
        for (int i = 0; i < fileStatus.length; i++) {
            //获取当前目录或文件的路径对象
            Path childPath = fileStatus[i].getPath();
            //打印子目录或文件名
            System.out.println("  "+childPath.getName());
            //打印子目录或文件全路径
            //System.out.println("  "+childPath.toString());
            FileStatus[] childrenFileStatus = null;
            try {
                //获取子目录的子目录或文件
                childrenFileStatus = hdfs.listStatus(childPath);
                //打印这个目录下的子目录或文件
                for (int k = 0; k < childrenFileStatus.length; k++) {
                    System.out.println("    " +
childrenFileStatus[k].getPath().getName());
                    //打印全路径
                    //System.out.println("    " +
childrenFileStatus[k].getPath().toString());
                }
            }
            catch (AccessControlException e) {
                System.out.println("无权限访问这个目录："+childPath.toString());
            }
        }

        hdfs.close();

    }

}
```

10. 创建目录

```
Configuration conf = new Configuration();
//此处的设置必须和core-site.xml配置文件中对应的参数值一致
conf.set("fs.defaultFS","hdfs://192.168.1.200:9000");

FileSystem fs = FileSystem.get(conf);
Path p = new Path("/user/newdir/");

fs.mkdirs(p);
fs.close();// 释放资源
```

11. 删除目录

```
Configuration conf = new Configuration();
//此处的设置必须和core-site.xml配置文件中对应的参数值一致
conf.set("fs.defaultFS","hdfs://192.168.1.200:9000");

FileSystem fs = FileSystem.get(conf);
Path p = new Path("/user/newdir/");

fs.deleteOnExit(p);
fs.close();// 释放资源
```

12. 常见问题 Permission denied

Permission denied:

解决方法：

到服务器上修改 NameNode 节点的 hadoop 的配置文件：conf/hdfs-site.xml，增加 dfs.permissions 的配置项，将 value 设置为 false。

<property>

<name>dfs.permissions</name>

<value>false</value>

</property>

修改完后要重启 Hadoop 服务才能生效。