Hadoop 分布式文件系统 HDFS

Java API 本地访问

侯在钱

目 录

1. 导入 Jar 包 2
2. HDFS 写文件 2
2.1. 程序代码 2
2.2. 打成 Jar 包 3
2.3. 上传 Jar 包 3
2.4. 启动 Hadoop 3
2.5. 运行程序 3
3. HDFS 读文件 4
3.1. 程序代码 4
3.2. 打成 Jar 包 5
3.3. 上传 Jar 包 5
3.4. 启动 Hadoop 5
3.5. 运行程序 5
4. 读写本地与 HDFS 文件的读写 5
4.1. 程序代码 6
4.2. 打成 Jar 包 7
4.3. 上传 Jar 包 7
4.4. 启动 Hadoop 7
4.5. 运行程序 7

利用 HDFS 给提供的 API，我们可以通过程序读写 HDFS 中的文件。在 Hadoop 中用于文件操作的主类位于 org.apache.hadoop.fs 软件包中。包括常见的 open、read、write、 close。我们首先使用的是 FileSystem 类，这是一个与文件系统交互的抽象类，我们通过调用 FileSystem.get(Configuration conf)来取得所需的 FileSystem 实例，如下我们可以获得与 HDFS 接口的 FileSystem 对象：

```
Configuration conf = new Configuration();
```

```
FileSystem hdfs = FileSystem.get(conf);//获得 HDFS 的 FileSystem 对象
```

如果我们要实现 HDFS 文件系统与本地文件系统的交互，我们还需要获取本地文件系统的 FileSystem 对象。本地是指 Linux 文件系统，因为 Hadoop 是构建在 Linux 之上的，我们称 Linux 为本地。获取本地文件系统的 FileSystem 对象的方法：

```
FileSystem local = FileSystem.getLocal(conf);//获得本地文件系统的 FileSystem 对象。
```

1. 导入 Jar 包

使用 HDFS Java API 的程序在编译时需要的 Jar 包有：hadoop-common-2.6.0.jar，此文件在 Hadoop 安装包的根目录下：/hadoop-2.6.0/share/hadoop/common/hadoop-common-2.6.0.jar。

把这个 Jar 包拷贝到 Eclipse 工程中，并加入到 Eclipse 的 Java Build Path 的 Libraries 中。

以下通过一个例子来讲解如何使用 Java API 来访问 HDFS。

2. HDFS 写文件

本例子实现在 HDFS 中创建文件"/user/root/test/test.txt"，并向文件中写入"Helo world!"

2.1. 程序代码

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

import java.io.IOException;

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

/**
 * 向Hadoop的HDFS中写文件，写入内容为“Helo world!”。
```

```
 * @author HouZaiqian
 */
public class HdfsWriteTest {

    public static void main(String[] args) throws IOException {
        Configuration conf = new Configuration();

        FileSystem fs = FileSystem.get(conf);

        Path f = new Path("/user/root/test/test.txt");
        FSDataOutputStream out = fs.create(f);

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

        out.close();

    }

}
```

2.2.打成 Jar 包

jar cvf hdfstest.jar com

2.3.上传 Jar 包

上传 **hdfstest.jar** 到 **Hadoop** 集群中的任意一台 **Linux** 上的任意目录下。

2.4.启动 Hadoop

在**/$HADOOP_HOME/sbin/**目录下执行：

./start_all.sh

注意：如果 **Hadoop** 已经启动了，则不用再启动。

2.5.运行程序

在 **hdfstest.jar** 文件所在的目录下执行如下命令：

hadoop jar hdfstest.jar com.hadoop.hdfs.test.HdfsWriteTest

备注：此命令是执行 **HdfsWriteTest** 程序。执行完毕后，通过 **HDFS** 的 **cat** 命令去查看此文件的内容，如果能查看到“**Hello World**!”。则说明此程序运行成功。

3. HDFS 读文件

从 **HDFS** 中读取一个文本文件中内容，并打印。首先确保 **HDFS** 中的这个文件是存在的。本节就以上节写入的文件**"/user/root/test/test.txt"**为例，说明如何读文件。

3.1. 程序代码

```
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;

/**
 * 从Hadoop的HDFS读取文件，并显示。
 * @author HouZaiqian
 */
public class HdfsReadTest {

    public static void main(String[] args) throws IOException {
        Configuration conf = new Configuration();

        FileSystem fs = FileSystem.get(conf);

        Path f = new Path("/user/root/test/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();

    }

}
```

3.2.打成 Jar 包

jar cvf hdfstest.jar com

3.3.上传 Jar 包

上传 **hdfstest.jar** 到 **Hadoop** 集群中的任意一台 **Linux** 上的任意目录下。

3.4.启动 Hadoop

在**/HADOOP_HOME/sbin/**目录下执行：

./start_all.sh

备注：如果 **Hadoop** 已经启动了，则不用再启动。

3.5.运行程序

在 **hdfstest.jar** 文件所在的目录下执行如下命令：

hadoop jar hdfstest.jar com.hadoop.hdfs.test.HdfsReadTest

备注：此命令是执行 **HdfsReadTest** 程序。执行完毕后，会在控制台打印从文件中读出的内容。

4. 读写本地与 HDFS 文件的读写

下面的 MergeTest 程序，用于合并本地多个文件，并把合并后的新文件放入 HDFS，这个程序在实际工作中会经常用到。

开发这个程序的步骤如下：

4.1.程序代码

```
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.FSDataOutputStream;
import org.apache.hadoop.fs.FileStatus;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;

public class MergeTest {

    /**
     * 功能：把第1个参数指定的目录下的所有文件都合并写到HDFS系统中指定的文件中。
     * 第1个参数为Linux本地的目录。
     * 第2个参数为HDFS系统中的文件。
     * @param args
     * @throws IOException
     */
    public static void main(String[] args) throws IOException {
        Configuration conf = new Configuration();
        //获得HDFS文件系统的对象
        FileSystem hdfsFileSystem = FileSystem.get(conf);
        //获得本地文件系统的对象
        FileSystem localFileSystem = FileSystem.getLocal(conf);
        //设定输入目录
        Path inputDir = new Path(args[0]);
        //设定输出目录
        Path hdfsFile = new Path(args[1]);
        try {
            //FileStatus的listStatus()方法获得一个目录中的文件列表
            FileStatus[] inputFiles = localFileSystem.listStatus(inputDir);
            //生成HDFS输出流
            FSDataOutputStream out = hdfsFileSystem.create(hdfsFile);
            for (int i=0; i<inputFiles.length; i++) {
                System.out.println(inputFiles[i].getPath().getName());
                //打开本地输入流
                FSDataInputStream in = localFileSystem.open(inputFiles[i].getPath());
                byte[] buffer = new byte[256];
```

```
                int bytesRead = 0;
                while ((bytesRead = in.read(buffer))>0) {
                    //通过一个循环来写入
                    out.write(buffer, 0, bytesRead);
                }
                in.close();
            }
            out.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

4.2.打成 Jar 包

jar cvf MergeTest.jar com

4.3.上传 Jar 包

上传**MergeTest.jar**到**Hadoop**集群中的任意一台**Linux**上的任意目录下。

4.4.启动 Hadoop

在**/HADOOP_HOME/sbin/**目录下执行：

./start_all.sh

备注：如果 **Hadoop** 已经启动了，则不用再启动。

4.5.运行程序

在 **hdfstest.jar** 文件所在的目录下执行如下命令：

hadoop jar PutMerge.jar com.hadoop.hdfs.test.MergeTest /tmp/test/ /root/hadoop/tmp/myfile.txt

备注：此命令的功能是把本地**/tmp/test/**目录下的所有文件内容合并写到 **HDFS** 文件系统中的**/root/hadoop/tmp /myfile.txt** 文件里。