

HDFS及其接口应用

0. Refs

1. HDFS简介

1.1. 命令行接口

1.2. JAVA API

2. 编写Hadoop应用程序

2.1. 编写合并文本的应用程序

2.2. 应用程序打包

0. Refs

<https://dblab.xmu.edu.cn/blog/2460/>

1. HDFS简介

Hadoop Distributed File System,Hadoop分布式文件系统。实际本质上也是个文件系统，但是它高度抽象，隐藏了众多细节，让用户感知不到TA正在使用一个分布式文件系统的同时，又能应用到DFS的分布式存储架构的优势。

1.1. 命令行接口

既然是个文件系统，必然有其对应的接口。hadoop提供了三种shell命令方式与dfs交互：

1. `hadoop fs`：适用于任何不同的文件系统，比如本地文件系统和HDFS文件系统
2. `hadoop dfs`：只能适用于HDFS文件系统
3. `hdfs dfs`：跟hadoop dfs的命令作用一样，也只能适用于HDFS文件系统

可以使用-help选项查看各个命令如何使用，例如：

```
./bin/hadoop fs -help put
```

其实光靠猜也能大概猜出来每条命令咋用的，比如put就是把本地的文件上传到hdfs里面去，get就是把hdfs里的某些文件拉取到本地文件系统中，mkdir就是在hdfs中创建一个文件夹等等...

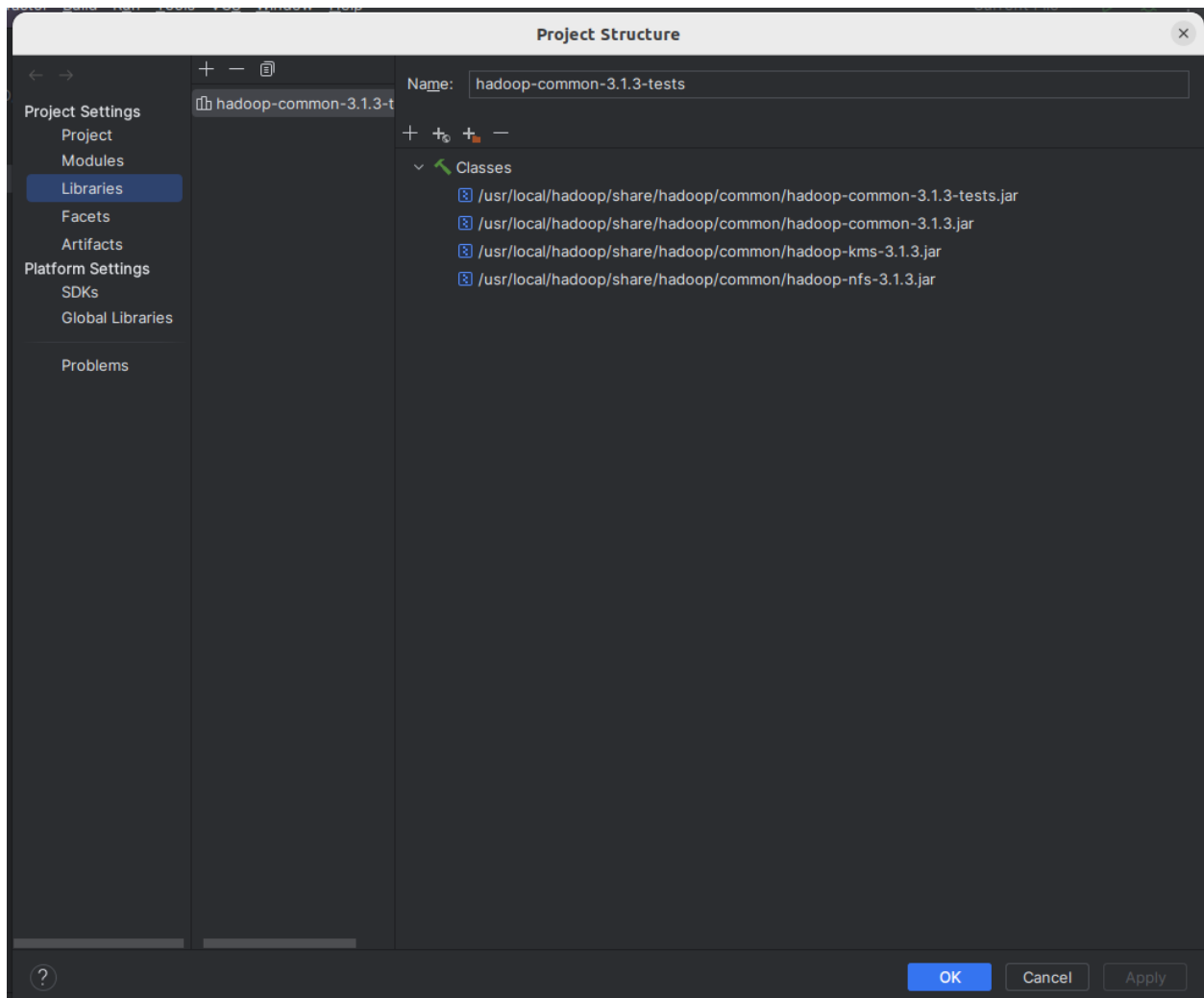
1.2. JAVA API

当我们遇到一个非常复杂的任务时，就不能仅靠命令行和hdfs交互了（得输入非常多的命令），这个时候最好是以代码脚本的方式和hdfs交互。Apache他们的生态就是Java系的，因此java社区中支持了一套完整且丰富的API，以供开发者和用户与hdfs交互。

java的开发环境我并没有像ref中的教程一样选择Eclipse，我个人偏爱jetbrain旗下的idea。因此我在我的ubuntu虚拟机里配了一个idea的集成开发环境。

首先我们hadoop的接口是以jar包形式提供的，所以要在项目的lib中集成这些jar包。这些jar包包括：

- (1) `"/usr/local/hadoop/share/hadoop/common"`目录下的所有JAR包，包括hadoop-common-3.1.3.jar、hadoop-common-3.1.3-tests.jar、hadoop-nfs-3.1.3.jar和hadoop-kms-3.1.3.jar，注意，不包括目录jdiff、lib、sources和webapps；
- (2) `"/usr/local/hadoop/share/hadoop/common/lib"`目录下的所有JAR包；
- (3) `"/usr/local/hadoop/share/hadoop/hdfs"`目录下的所有JAR包，注意，不包括目录jdiff、lib、sources和webapps；
- (4) `"/usr/local/hadoop/share/hadoop/hdfs/lib"`目录下的所有JAR包。



之后就可以编写代码，使用hdfs的API了。

2. 编写Hadoop应用程序

2.1. 编写合并文本的应用程序

既然我们要使用hadoop与hdfs，那势必要启动hadoop服务器，这个是之前一项作业提到过的，配置好之后脚本启动就ok。

对于应用的程序，本次的任务是把某些特定文件的内容合并到一个文件merge.txt中，需要根据特殊的规则过滤掉不符合内容的文件，例如source3.abc。

然后为了批量上传文件到hdfs，我写了个上传文件的脚本：

```

package Merge_File_Demo.UploadFiles;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IOUtils;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.net.URI;

public class Upload {
    public static void main(String[] args) {
        // HDFS配置
        Configuration conf = new Configuration();
        conf.set("fs.defaultFS", "hdfs://localhost:9000");
        conf.set("fs.hdfs.impl", "org.apache.hadoop.hdfs.DistributedFileSystem");

        // 本地文件目录
        String localDirPath = args[0];

        // HDFS目标路径
        String hdfsTargetPath = args[1];

        try {
            // 获取HDFS文件系统对象
            FileSystem fs = FileSystem.get(URI.create(conf.get("fs.defaultFS")), conf);

            // 遍历本地目录并上传文件
            File localDir = new File(localDirPath);
            File[] files = localDir.listFiles();
            if (files != null) {
                for (File file : files) {

```

```

        if (file.isFile()) {
            // 创建HDFS路径对象
            Path hdfsPath = new Path(hdfsTargetPath

            // 打开本地文件输入流
            InputStream in = new FileInputStream(fi

            // 使用Hadoop的API将文件写入HDFS
            fs.copyFromLocalFile(new Path(file.getAl

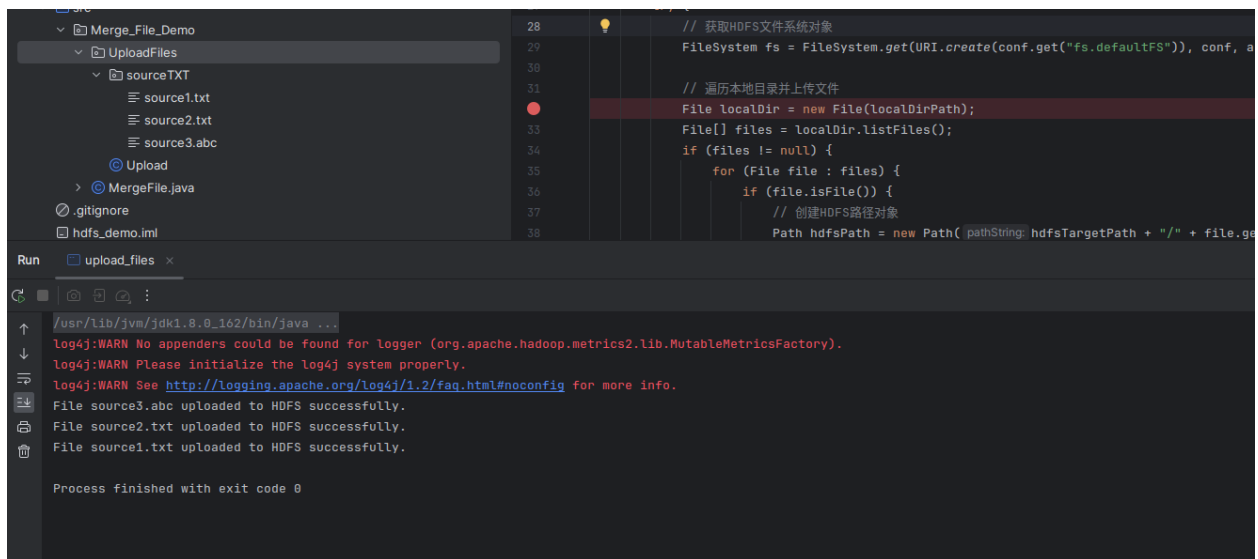
            // 关闭输入流
            IOUtils.closeStream(in);
            System.out.println("File " + file.getNar

        }
    }
} else {
    System.out.println("No files found in the local
}

// 关闭文件系统连接
fs.close();
} catch (IOException | InterruptedException e) {
    e.printStackTrace();
}
}
}
}

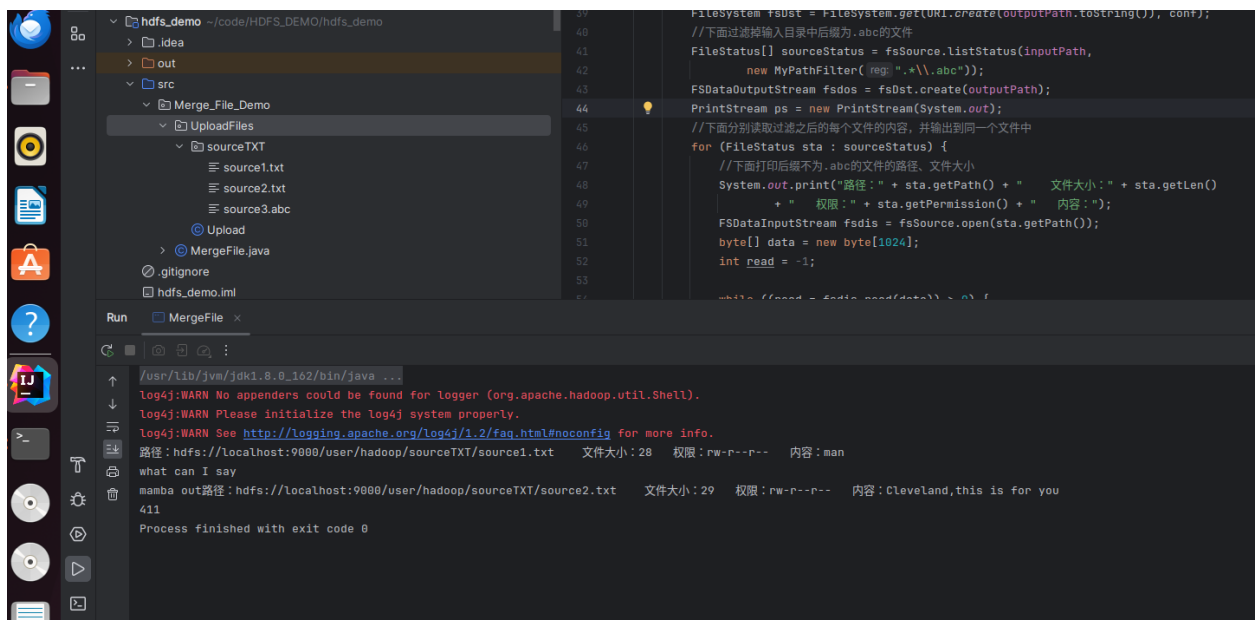
```

大致意思就是命令行给参数，然后遍历指定的本地目录，把文件全部上传到hdfs的指定目录里面去。



运行结果大概这样。

然后运行合并文件的应用程序，结果如下：



Browse Directory

Show 25 entries

Search:

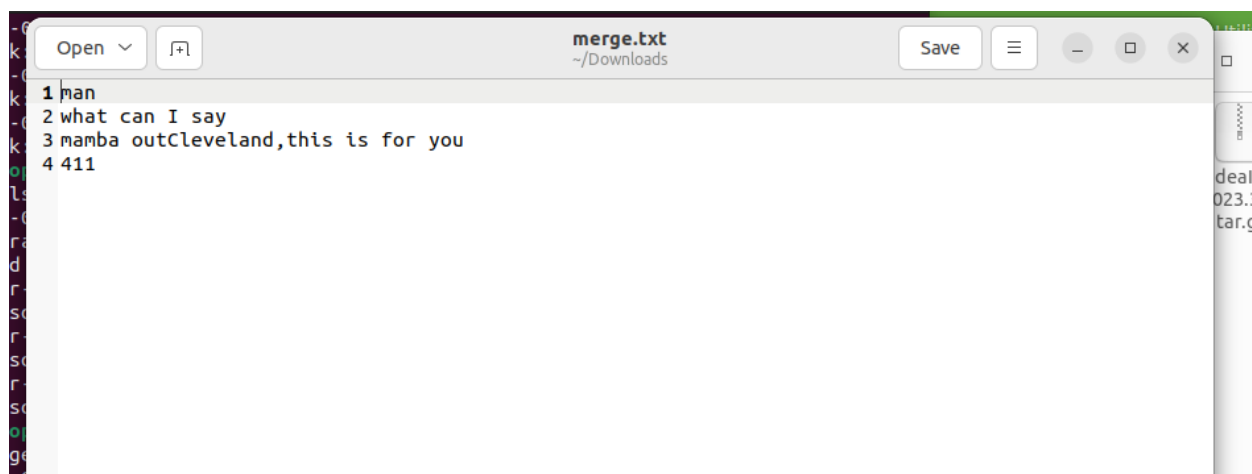
<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	<input type="checkbox"/>
<input type="checkbox"/>	-rw-r--r--	hadoop	supergroup	57 B	Mar 19 08:44	3	128 MB	merge.txt	<input type="checkbox"/>
<input type="checkbox"/>	-rw-r--r--	hadoop	supergroup	28 B	Mar 19 08:42	3	128 MB	source1.txt	<input type="checkbox"/>
<input type="checkbox"/>	-rw-r--r--	hadoop	supergroup	29 B	Mar 19 08:42	3	128 MB	source2.txt	<input type="checkbox"/>
<input type="checkbox"/>	-rw-r--r--	hadoop	supergroup	111 B	Mar 19 08:42	3	128 MB	source3.abc	<input type="checkbox"/>

Showing 1 to 4 of 4 entries

Hadoop, 2019.

从web页面可以看到merge.txt已经生成了。

随后我们利用hdfs的get命令把merge.txt拉到本地来。结果如下：



可以看到确实只有source1和source2的内容，source3.abc被过滤了。

2.2. 应用程序打包

通过idea提供的build artifacts，将两个应用程序（一个用来上传文件，一个用来合并文件）打包放置到/usr/local/hadoop/myapp

随后清除一下之前在idea中执行的记录，把

hadoop:localhost:9000/user/hadoop/sourceTXT下的文件全部清除，之后测试两个jar包。

```
mergeFile.jar uploadFile.jar
hadoop@hadoop: /usr/local/hadoop/myapp$ hadoop jar ./uploadFile.jar ~/code/HDFS_DEMO/hdfs_demo/src/Merge_File_Demo/UploadFiles/sourceTXT /user/hadoop/sourceTXT/
hadoop
2024-03-19 09:02:48,534 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
2024-03-19 09:04:02,326 INFO sasl.SaslDataTransferClient: SASL encryption trust
check: localhostTrusted = false, remoteHostTrusted = false
File source3.abc uploaded to HDFS successfully.
2024-03-19 09:04:04,772 INFO sasl.SaslDataTransferClient: SASL encryption trust
check: localhostTrusted = false, remoteHostTrusted = false
File source2.txt uploaded to HDFS successfully.
2024-03-19 09:04:05,374 INFO sasl.SaslDataTransferClient: SASL encryption trust
check: localhostTrusted = false, remoteHostTrusted = false
File source1.txt uploaded to HDFS successfully.
hadoop@hadoop: /usr/local/hadoop/myapp$
```

上传程序成功。

随后运行合并程序。

```
hadoop@hadoop: /usr/local/hadoop/myapp
library for your platform... using builtin-java classes where applicable
2024-03-19 09:04:02,326 INFO sasl.SaslDataTransferClient: SASL encryption trust
check: localhostTrusted = false, remoteHostTrusted = false
File source3.abc uploaded to HDFS successfully.
2024-03-19 09:04:04,772 INFO sasl.SaslDataTransferClient: SASL encryption trust
check: localhostTrusted = false, remoteHostTrusted = false
File source2.txt uploaded to HDFS successfully.
2024-03-19 09:04:05,374 INFO sasl.SaslDataTransferClient: SASL encryption trust
check: localhostTrusted = false, remoteHostTrusted = false
File source1.txt uploaded to HDFS successfully.
hadoop@hadoop: /usr/local/hadoop/myapp$ hadoop jar ./MergeFile.jar
2024-03-19 09:05:43,885 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
路径 : hdfs://localhost:9000/user/hadoop/sourceTXT/source1.txt 文件大小 : 28
权限 : rw-r--r-- 内容 : 2024-03-19 09:05:48,187 INFO sasl.SaslDataTransferClient
: SASL encryption trust check: localhostTrusted = false, remoteHostTrusted = fal
se
man
to what can I say
mamba out路径 : hdfs://localhost:9000/user/hadoop/sourceTXT/source2.txt 文件大
小 : 29 权限 : rw-r--r-- 内容 : Cleveland,this is for you
4112024-03-19 09:05:48,422 INFO sasl.SaslDataTransferClient: SASL encryption tru
st check: localhostTrusted = false, remoteHostTrusted = false
hadoop@hadoop: /usr/local/hadoop/myapp$
```

合并程序成功。

随后我们用hdfs dfs -cat捕获一下新出现的merge.txt


```
-rw-rw-r-- 1 root root 4096 2024-03-19 09:07:18,922 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
128
128
to 2024-03-19 09:07:19,975 INFO sasl.SaslDataTransferClient: SASL encryption trust
check: localhostTrusted = false, remoteHostTrusted = false
man
what can I say
01 mamba outCleveland.this is for you
411 hadoop@hadoop:/usr/local/hadoop/myapp$
```

内容正确，两个应用程序均成功。