一、实现远程执行

此程序的目的是执行远程机器上的Shell脚本。

【环境参数】

远程机器IP：192.168.243.21

用户名：user

密码：password

命令：python /data/applogs/bd-job/jobhandler/gluesource/employee.py

【具体步骤】

1、导入需要依赖的jar包。

```
<dependency>
   <groupId>ch.ethz.ganymed</groupId>
   <artifactId>ganymed-ssh2</artifactId>
   <version>262</version>
</dependency>
```

```
<dependency>
   <groupId>commons-io</groupId>
   <artifactId>commons-io</artifactId>
   <version>2.5</version>
</dependency>
```

2、编写RemoteShellExecutor工具类。

```
package com.vicente.vicenteboot.test;

import java.io.*;
import java.nio.charset.Charset;
import ch.ethz.ssh2.ChannelCondition;
import ch.ethz.ssh2.Connection;
import ch.ethz.ssh2.Session;
import ch.ethz.ssh2.StreamGobbler;
import org.apache.commons.io.IOUtils;

public class RemoteShellExecutor {

    private Connection conn;
    /** 远程机器IP */
    private String ip;
    /** 用户名 */
    private String osUsername;
    /** 密码 */
    private String password;
    private String charset = Charset.defaultCharset().toString();

    private static final int TIME_OUT = 1000 * 5 * 60;
```

```
public RemoteShellExecutor(String ip, String usr, String pasword) {
    this.ip = ip;
    this.osUsername = usr;
    this.password = pasword;
}


/**
 * 登录
 * @return
 * @throws IOException
 */
private boolean login() throws IOException {
    conn = new Connection(ip);
    conn.connect();
    return conn.authenticateWithPassword(osUsername, password);
}

/**
 * 执行脚本
 *
 * @param cmds
 * @return
 * @throws Exception
 */
public int exec(String cmds) throws Exception {
    InputStream stdOut = null;
    InputStream stdErr = null;
    String outStr = "";
    String outErr = "";
    int ret = -1;
    try {
        if (login()) {
            // Open a new {@link Session} on this connection
            Session session = conn.openSession();
            // Execute a command on the remote machine.
            session.execCommand(cmds);
            stdOut = new StreamGobbler(session.getStdout());
            outStr = processStream(stdOut, charset);

            stdErr = new StreamGobbler(session.getStderr());
            outErr = processStream(stdErr, charset);

            session.waitForCondition(ChannelCondition.EXIT_STATUS, TIME_OUT);
```

```
                System.out.println("outStr=" + outStr);
                System.out.println("outErr=" + outErr);

                ret = session.getExitStatus();
            } else {
                throw new Exception("登录远程机器失败" + ip); // 自定义异常类 实现略
            }
        } finally {
            if (conn != null) {
                conn.close();
            }
            IOUtils.closeQuietly(stdOut);
            IOUtils.closeQuietly(stdErr);
        }
        return ret;
    }

    private String processStream(InputStream in, String charset) throws Exception {
        byte[] buf = new byte[1024];
        StringBuilder sb = new StringBuilder();
        while (in.read(buf) != -1) {
            sb.append(new String(buf, charset));
        }
        return sb.toString();
    }

    public static void main(String args[]) throws Exception {
        RemoteShellExecutor executor = new RemoteShellExecutor("192.168.243.21", "domp", "bluemoon2016#");
        // 执行myTest.sh 参数为java Know dummy
        System.out.println(executor.exec2("python /data/bluemoon/kettle/runScript/ods/fact_org_employee.py
    }
}
```

3、运行结果

备份数据成功。

4、说明：

0 // getExitStatus方法的返回值

注：一般情况下shell脚本正常执行完毕，getExitStatus方法返回0。

此方法通过远程命令取得Exit Code/status。但并不是每个server设计时都会返回这个值，如果没有则会返回null。

二、ganymed-ssh讲解：

1. Jar包：ganymed-ssh2-build210.jar

2. 步骤：

a) 连接：

Connection conn = new Connection(ipAddr);

conn.connect();

b)认证：

boolean authenticateVal = conn.authenticateWithPassword(userName, password);

c) 打开一个Session:

if(authenticateVal)

Session session = conn.openSession();

d) 执行Shell命令：

1）若是执行简单的Shell命令：（如 jps 、last 这样的命令 ）

session.execCommand(cmd);

2) 遇到问题：

用方法execCommand执行Shell命令的时候，会遇到获取不全环境变量的问题，

比如执行 hadoop fs -ls 可能会报找不到hadoop 命令的异常

试着用execCommand执行打印环境变量信息的时候，输出的环境变量不完整

与Linux主机建立连接的时候会默认读取环境变量等信息

可能是因为session刚刚建立还没有读取完默认信息的时候，execCommand就执行了Shell命令

解决：

所以换了另外一种方式来执行Shell命令：

```
// 建立虚拟终端
session.requestPTY("bash");
// 打开一个Shell
session.startShell();
// 准备输入命令
PrintWriter out = new PrintWriter(session.getStdin());

// 输入待执行命令
out.println(cmd);
out.println("exit")
// 6. 关闭输入流
out.close();
// 7. 等待，除非1.连接关闭；2.输出数据传送完毕；3.进程状态为退出；4.超时
session.waitForCondition(ChannelCondition.CLOSED | ChannelCondition.EOF |
ChannelCondition.EXIT_STATUS , 30000);
```

用这种方式执行Shell命令，会避免环境变量读取不全的问题，第7步里有许多标识可以用，比如当exit命令执行后或者超过了timeout时间，则session关闭

这里需要注意，当一个Shell命令执行时间过长时，会遇到ssh连接超时的问题，

解决办法：

1. 之前通过把Linux主机的sshd_config的参数ClientAliveInterval设为60，同时将第7步中timeout时间设置很大，来保证命令执行完毕，

因为是执行Mahout中一个聚类算法，耗时最少7、8分钟，数据量大的话，需要几个小时。

2. 后来将命令改成了nohup的方式执行，nohup hadoop jar .... >> XXX.log && touch XXX.log.end &

这种方式是提交到后台执行，即使当前连接断开也会继续执行，把命令的输出结果写入日志，如果

hadoop命令执行成功，则生成.end文件

获取文件的方法 ganymed-ssh2-build210.jar 也提供了，如下

SCPClient scpClient = con.createSCPClient();

scpClient.get("remoteFiles","localDirectory"); //从远程获取文件

e) 获取Shell命令执行结果：

InputStream stderr = new StreamGobbler(session.getStderr());

InputStream in = new StreamGobbler(session.getStdout());

获取流中的数据：

```
private String processStdErr(InputStream in, String charset)
            throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(in, charset));
        StringBuffer sb = new StringBuffer();

        if (in.available() != 0) {
            while (true) {
                String line = br.readLine();
                if (line == null)
                    break;
                sb.append(line).append(System.getProperty("line.separator"));
            }
        }
        return sb.toString();
    }
```

三、使用实例

1、使用session.execCommand(cmds);来执行命令会出现一个问题，就是还没有加载完成服务器的环境变量就开始执行命令了

这导致了很多情况是找不到命令，解决的办法就是：

```
/**
 * 执行脚本
 *
 * @param cmds
 * @return
 * @throws Exception
 */
public int exec2(String cmds) throws Exception {
    InputStream stdOut = null;
    InputStream stdErr = null;
    String outStr = "";
    String outErr = "";
    int ret = -1;
    try {
        if (login()) {
            Session session = conn.openSession();
            // 建立虚拟终端
            session.requestPTY("bash");
            // 打开一个Shell
            session.startShell();
            stdOut = new StreamGobbler(session.getStdout());
```

```
            stdErr = new StreamGobbler(session.getStderr());
            BufferedReader stdoutReader = new BufferedReader(new InputStreamReader(stdOut));
            BufferedReader stderrReader = new BufferedReader(new InputStreamReader(stdErr));

            // 准备输入命令
            PrintWriter out = new PrintWriter(session.getStdin());
            // 输入待执行命令
            out.println(cmds);
            out.println("exit");
            // 6. 关闭输入流
            out.close();
            // 7. 等待，除非1.连接关闭；2.输出数据传送完毕；3.进程状态为退出；4.超时
            session.waitForCondition(ChannelCondition.CLOSED | ChannelCondition.EOF | ChannelCondition.EXI
            System.out.println("Here is the output from stdout:");
            while (true)
            {
                String line = stdoutReader.readLine();
                if (line == null)
                    break;
                System.out.println(line);
            }
            System.out.println("Here is the output from stderr:");
            while (true)
            {
                String line = stderrReader.readLine();
                if (line == null)
                    break;
                System.out.println(line);
            }
            /* Show exit status, if available (otherwise "null") */
            System.out.println("ExitCode: " + session.getExitStatus());
            ret = session.getExitStatus();
            session.close();/* Close this session */
            conn.close();/* Close the connection */

        } else {
            throw new Exception("登录远程机器失败" + ip); // 自定义异常类 实现略
        }
    } finally {
        if (conn != null) {
            conn.close();
        }
        IOUtils.closeQuietly(stdOut);
        IOUtils.closeQuietly(stdErr);
    }
    return ret;
```

```
}
```

2、scp复制一个文件到服务器上

```
/**
 * 远程传输单个文件
 *
 * @param localFile
 * @param remoteTargetDirectory
 * @throws IOException
 */

public void transferFile(String localFile, String remoteTargetDirectory) throws IOException {
    File file = new File(localFile);
    if (file.isDirectory()) {
        throw new RuntimeException(localFile + "  is not a file");
    }
    String fileName = file.getName();
    execCommand("mkdir -p " + remoteTargetDirectory);

    SCPClient sCPClient = connection.createSCPClient();
    SCPOutputStream scpOutputStream = sCPClient.put(fileName, file.length(), remoteTargetDirectory, "0600"

    String content = IOUtils.toString(new FileInputStream(file),StandardCharsets.UTF_8);
    scpOutputStream.write(content.getBytes());
    scpOutputStream.flush();
    scpOutputStream.close();
}

/**
 * 传输整个目录
 *
 * @param localDirectory
 * @param remoteTargetDirectory
 * @throws IOException
 */
public void transferDirectory(String localDirectory, String remoteTargetDirectory) throws IOException {
    File dir = new File(localDirectory);
    if (!dir.isDirectory()) {
        throw new RuntimeException(localDirectory + " is not directory");
    }

    String[] files = dir.list();
    for (String file : files) {
        if (file.startsWith(".")) {
            continue;
```

```
        }
        String fullName = localDirectory + "/" + file;
        if (new File(fullName).isDirectory()) {
            String rdir = remoteTargetDirectory + "/" + file;
            execCommand("mkdir -p " + remoteTargetDirectory + "/" + file);
            transferDirectory(fullName, rdir);
        } else {
            transferFile(fullName, remoteTargetDirectory);
        }
    }

}
```

使用:

```
public static void main(String[] args) throws IOException {
    SSHAgent sshAgent = new SSHAgent();
    sshAgent.initSession("192.168.243.21", "user", "password#");
    sshAgent.transferFile("C:\\data\\applogs\\bd-job\\jobhandler\\2020-03-07\\483577870267060231.log","/da
    sshAgent.close();
}
```

参考：https://www.cnblogs.com/-wangjiannan/p/3751330.html