Instructions for Assessed Exercises

Dr. Nikos Ntarmos <nikos.ntarmos@glasgow.ac.uk>

Introduction

The following describe how to connect to one of the computers in the SoCS network, and how to use said computers to access the BD4 clusters. Users logged on to one of the lab PCs in the Boyd Orr or Sir Alwyn Williams buildings may skip the next section and go directly to "Programmatic access to the BD4 clusters".

External access to the SoCS network

All computers in the SoCS network (including lab PCs and course servers) are hidden to the outside world by one or more firewalls, and the computing nodes comprising the cluster for the BD4 course are no exception. This holds for all computers not directly connected to the School's network, including computers connected to the *eduroam* wireless network. In order for students and staff members to be able to bypass these firewalls, the School is using a relatively low capacity server, called "sibu.dcs.gla.ac.uk" ("sibu" for short).

NOTE: Sibu is meant to be used **only** as an intermediate step and not as a general purpose server/host, as its processing capacity is rather limited.

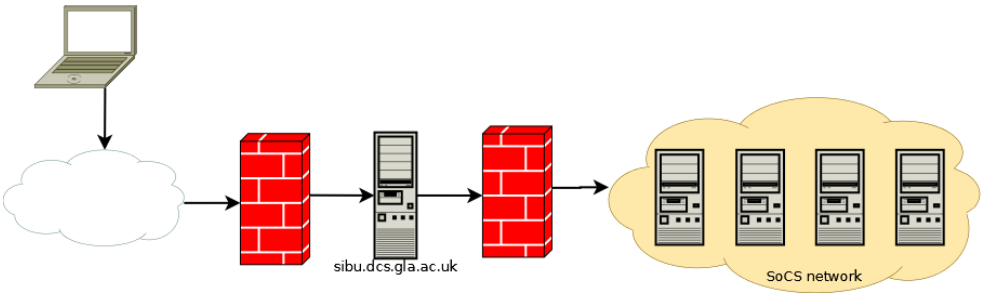

Figure 1 Sibu as an intermediate SSH hop to the SoCS network

Sibu is accessible from most networks outside the SoCS network block; however, there are several cases where connections are being dropped/rejected by Sibu. This may happen intermittently if, for example, you are connecting from a mobile network (i.e., over 3G/4G) or from a remote network yet unknown to Sibu's firewall. In these cases, you'd need to connect over VPN to the University, and then to ssh to Sibu. See [1] for instructions on how to configure your computer for VPN access.

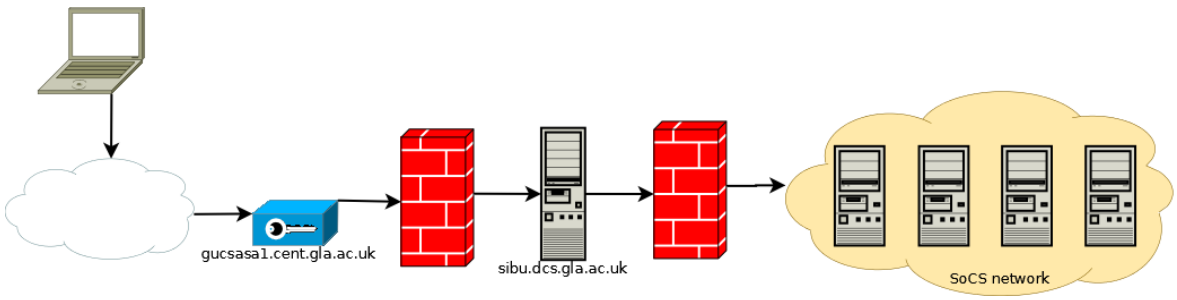

Figure 2 VPN+Sibu

[1] http://www.gla.ac.uk/services/it/flexaccess/vpn/offcampusaccess/

Sibu auto-mounts the users' home directories over the network (using NIS+/NFS); this means that all of your files are directly accessible from Sibu, and any file you upload to your home directory on Sibu is also directly accessible by any other Linux/*nix host in the SoCS network. This allows you to copy files to and from the SoCS network by just using Sibu as a mediator. You can use *scp* on Linux/*nix/MacOSX computers, or *WinSCP* on Windows hosts.

```
user@home:~$ scp myfiles.zip user@sibu.dcs.gla.ac.uk:
user@home:~$ scp -r myfiles/ user@sibu.dcs.gla.ac.uk:
```

Figure 3 Copying files/directories with scp

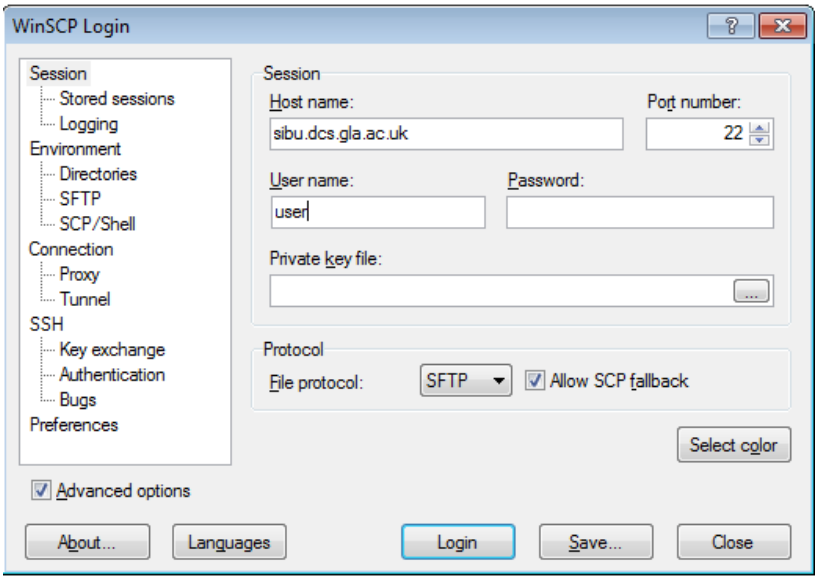

Figure 4 WinSCP

For your assessed exercises, you will need to be able to (compile and) run Java 7 code which will be connecting to the BD4 clusters in order to submit MapReduce jobs and access data stored on HDFS and/or HBase. Access to the cluster nodes is only allowed for computers within the SoCS network, as these nodes are behind the firewalls mentioned earlier. Thus, in order for your code to access these clusters, it will have to be executed from a host within the School's network (e.g., any of the lab PCs in Boyd Orr). Currently the L4 lab PCs are configured to dual-boot Windows and Linux. However, for the following to work you will need to locate a PC booted into Linux, as Windows doesn't provide SSH and/or remote shell sessions by default.

```
1. user@home:~$ ssh user@sibu.dcs.gla.ac.uk

2. user@sibu:~$ ssh bo620-XXu
   (where: XX ∈ [01, 02, 03, …, 30])

3. user@bo620-XXu:~$ java …
```

Figure 5 SSH hop through Sibu to L4 lab PCs

Finding such a PC may be an exercise in patience, as other users may reboot the PCs to Windows without first asking for permission. To this end, the School has provided us with a separate server, named "karkar.dcs.gla.ac.uk" (or "karkar" for short). This is a Linux host with a much higher processing capacity than Sibu, thus you may use it to compile and run your code accessing the BD4

clusters. Note that you will still have to hop through Sibu (and/or the VPN concentrator), as Karkar is not accessible from outside the School's network. In any case, Glasgow University (GU) students are urged to use lab PCs whenever possible, as Karkar will also be used by fellow students from the Singapore Institute of Technology (SIT) enrolled in this course.

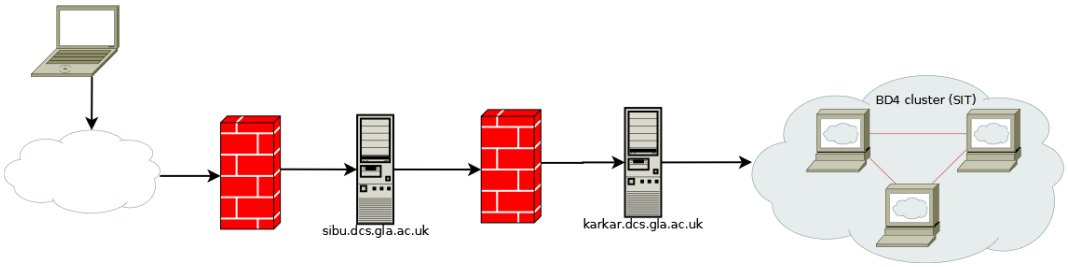

Figure 6 Sibu+Karkar

Last, you may want to access the web UIs of the BD4 clusters (e.g., to get statistics and logs of your jobs, list files on HDFS, examine your HBase tables, etc.). Since the servers for these UIs are running on the cluster nodes, you will have to tunnel your HTTP connections through your (VPN+) SSH connection. The easiest way to do this is via SSH's built-in support for SOCKS proxying. The first step in using this is enabling dynamic port forwarding on the SSH connection. This is accomplished via the "*-D*" command line option, or via the corresponding UI knobs (e.g., in Putty for Windows hosts).

```
1. user@home:~$ ssh -CD 1080 user@sibu.dcs.gla.ac.uk
2. user@sibu:~$ ssh karkar
3. user@karkar:~$ java …
```

Figure 7 SSH hop through Sibu to Karkar, with SOCKS proxying

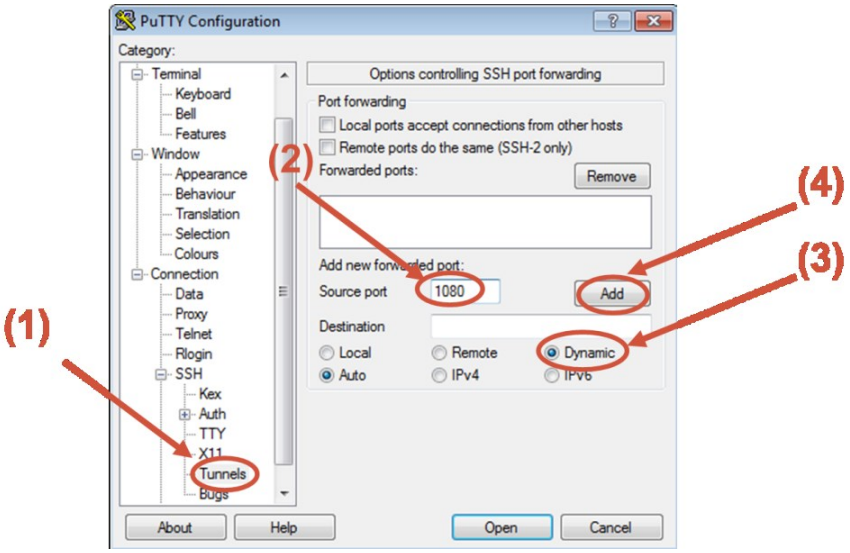

Figure 8 SOCKS proxying in Putty

Once the SOCKS proxy is in place, you then need to configure your web browser of choice to connect through a SOCKS v5 proxy, listening at IP 127.0.0.1 and port 1080. Please refer to your browser's documentation about how to best accomplish this, or contact us for further instructions. The URLs for the NameNode, Resource Manager, History Server, and HBase Master UIs will then be:

- For UG students:
  - NameNode: http://dsenode0.dcs.gla.ac.uk:50070/
  - Resource Manager: http://dsenode0.dcs.gla.ac.uk:8088/
  - Job History Server: http://dsenode0.dcs.gla.ac.uk:19888/jobhistory
  - HBase Master: http://dsenode0.dcs.gla.ac.uk:60010/
- For SIT students:
  - NameNode: http://bigdata-10.dcs.gla.ac.uk:50070/
  - Resource Manager: http://bigdata-10.dcs.gla.ac.uk:8088/
  - Job History Server: http://bigdata-10.dcs.gla.ac.uk:19888/jobhistory
  - HBase Master: http://bigdata-10.dcs.gla.ac.uk:60010/

Last, keep in mind that network connections can go down at any time point. To avoid losing your progress (or losing track of your running MR jobs etc.), please consider using *screen* or *tmux*. This program is already installed on karkar and it should also be there in the lab PCs. Google around for instructions on how to use, detach, and reattach *screen* and *tmux* sessions, as this can be a "life saver" when working remotely (e.g., over ssh).

Programmatic and shell access to the BD4 clusters

Once logged on to a computer within the SoCS network, you will be able to access the BD4 clusters programmatically by using the appropriate configuration files, jar files and scripts provided for you under /local/bd4/bd4-hadoop-ug/ for UG students or under /local/bd4/bd4-hadoop-sit/ for SIT students. Said directory constitutes a mini Hadoop environment with client libraries and scripts allowing you to access the clusters remotely. To use this mini environment, you will need to "source" the appropriate setup script file from your Shell's RC file.

To find out which shell you are using, execute "*echo $0*" on a terminal window. If this command prints out "tcsh" then you are using a CSH-compatible shell and you will be creating/updating ~/.cshrc; if you get "-bash" then you are using a SH-compatible shell and you should be creating/updating ~/.bashrc. If you look under /local/bd4 (i.e., "ls /local/bd4") you should see there are 4 aptly named setup scripts: bd4-setup-{ug,sit}.{sh,csh}. Choose the appropriate script for your institution and shell and then add a "source" line to your ~/.basrc file. For example, UG students using SH-compatible shells should add:

```
source /local/bd4/bd4-setup-ug.sh >/dev/null
```

SIT students using a CSH-compatible shell should add:

```
source /local/bd4/bd4-setup-sit.csh >/dev/null
```

and so on.

The next time you log on or open a new terminal, you should have the appropriate mini environment configured. To test it out, type:

```
hdfs-dfs.sh -ls -d
```

You should get a single line of output (ignore the warning printed) similar to:

```
drwxr-x---   - username hadoop          0 2017-01-30 13:57 .
```

NOTE: There will be two such mini-environments: one for GU students ("*bd4-*-ug*") and one for SIT students ("*bd4-*-sit*"). Please **take care to use the correct one**! If you now look at the files under *conf/* in the appropriate mini-environment directory, you'll find a file named "***core-site.xml***". This configuration file should be parsed and added to your Java client code configuration object, by using the *addResource()* method of the *org.apache.hadoop.conf.Configuration* class (or one of its appropriate subclasses). Keep in mind that you'll need to use the absolute path to the file.

CAUTION: Make sure that the configuration initialised thusly is passed on to your job object! To achieve this, you can either initialise the conf object and pass it as an argument like so:

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;
[…]

public class MyMRDriver extends Configured implements Tool {
   […]
   public static void main(String[] args) {
      […]
      Configuration conf = new Configuration();
      conf.addResource(new Path("/local/bd4/bd4-hadoop-.../conf/core-site.xml"));
      conf.set("mapred.jar", "file:///path/to/my.jar");
      ToolRunner.run(conf, new MyMRDriver(), args);
      […]
   }
   […]
}
```

or add the extra configuration parameters in the job's *run()* method, like so:

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.util.Tool;
[…]

public class MyMRDriver extends Configured implements Tool {
   […]
   @Override
   public int run(String[] args) {
      Configuration conf = new Configuration(getConf());
      conf.addResource(new Path("/local/bd4/bd4-hadoop-.../conf/core-site.xml"));
      conf.set("mapred.jar", "file:///path/to/my.jar");
      Job job = Job.getInstance(conf);
      […]
   }
   […]
}
```

Similarly, for jobs accessing HBase you will have to "import" these configuration parameters in your HBaseConfiguration object like so:

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;
[…]

public class MyMRDriver extends Configured implements Tool {
   […]
   public static void main(String[] args) {
      […]
      Configuration conf = HBaseConfiguration.create();
      conf.addResource(new Path("/local/bd4/bd4-hadoop-.../conf/core-site.xml"));
      conf.set("mapred.jar", "file:///path/to/my.jar");
      ToolRunner.run(conf, new MyMRDriver(), args);
      […]
   }
   […]
}
```

or equivalently:

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.util.Tool;
[…]

public class MyMRDriver extends Configured implements Tool {
   […]
   @Override
   public int run(String[] args) {
      Configuration conf = HBaseConfiguration.create(getConf());
      conf.addResource(new Path("/local/bd4/bd4-hadoop-.../conf/core-site.xml"));
      conf.set("mapred.jar", "file:///path/to/my.jar");
      Job job = Job.getInstance(conf);
      […]
   }
   […]
}
```

In order for you to compile and run your code at home (or in the lab), you will need a basic set of Jar files. You can find them under the lib/ directory of your mini-environment. Alongside these Jar files there will also be a set of shell scripts which you can use to execute the "*mapred*", "*hdfs*" and "*hbase*" commands mentioned in the tutorials. The "*hdfs-dfs.sh*" script replicates the functionality of "*hdfs dfs …*". Similarly, "*mapred-job.sh*" replicates the functionality of "*mapred job …*". That is, for example, instead of executing "*hdfs dfs -ls /user/bd4-ae1*" you should now execute "*hdfs-dfs.sh -ls /user/bd4-ae1*", instead of "*mapred job -list*" you should use "*mapred-job.sh -list*", etc. Last, "hbase.sh" provides a subset of the functionality of the "hbase" command – namely, "*hbase shell*", "*hbase classpath*" and "*hbase CLASSNAME*". Additionally there is a shell script, "*javac-run.sh*", to allow you to compile your code directly from the console, as well as a shell script, "*java-run.sh*" to allow you to submit your jobs from the console.

NOTE: Please do **not** include these scripts or the jar files in your submissions on Moodle; as a matter of fact, do **not submit any binary** (.class, .jar) **files**, as these are of no use to us and only bloat the size of your submissions.

CAUTION: Compile your code with a Java8 compiler, or with the source level compatibility set to Java 8 (or 1.8). Failing to do so may result in errors caused by incompatibility with the JDK version used in the clusters.

When your code is ready for execution, you should bundle it up into a jar file (say, "my.jar"). You can do that through your favourite IDE or by using command-line tools. It is this file that will be submitted to the cluster by the Job instance. In order for Hadoop's client-side code to locate said jar file, you should include the full path to it in your job's *Configuration* instance, under the key "*mapred.jar*" (see code snippets above). Please note that "*mapred.jar*" is not the name of your jar file but the predefined name of the relevant configuration variable; you should only change the second argument to the respective *conf.set(…)* method invocation to point to your jar file.

To run your code, execute it through your IDE of preference, making sure to add all jar files under "*bd4-hadoop/lib*", as well as the directory "*bd4-hadoop/conf*", to your classpath. Alternatively, use the "bd4-hadoop/java-run.sh" script; if you are using a Bourne-type shell (sh, bash, zsh), then type:

```
$ export HADOOP_CLASSPATH=/path/to/my.jar
$ java-run.sh MyMRDriver arg1 arg2 arg3 …
```

otherwise, if you are using a CSH-type shell (csh, tcsh), then type:

```
% setenv HADOOP_CLASSPATH /path/to/my.jar
% java-run.sh MyMRDriver arg1 arg2 arg3 …
```

(and consider changing your default shell to something more modern…)

Putting everything together, your workflow should follow the steps below:

1. Source the appropriate setup script from your shell's RC file.
2. Write the code using your favourite tools.
    - Make sure to include the *conf.addResource(…)* and *conf.set("mapred.jar", …)* lines in your code. Also make sure the paths used in these two lines correspond to the locations of the core-site.xml and jar file on the host from where you will execute the code (i.e., karkar or lab PC).
    - Make sure the configuration object you've populated with the above two lines is actually the one used by your job; look at the code samples above for proper ways of setting these parameters and passing the Configuration object to your job.
    - In eclipse or other IDEs, make sure to add all the files under *bd4-hadoop-…/lib/* and *bd4-hadoop-…/conf/* to your classpath or buildpath. On the command line, change directory to your src/ folder and execute:
      $ *javac-run.sh $(find . –name '*.java')*
3. Create a jar file with all of your classes.
    - In eclipse, right-click on your src/ folder and select Export->JAR. From the command line, change directory to your *src/* folder and execute:
      $ *jar -cvf /path/to/my.jar $(find . -name '*.class')*.
4. Copy your jar file to a location on sibu/karkar/Linux lab PC, making sure it will be at the exact same path as that defined in the *conf.set("mapred.jar", …)* line.
5. Set the HADOOP_CLASSPATH environment variable and execute your code via *java-run.sh* as mentioned above.

Last, for your 1st AE, you will be working with a dump of the Wikipedia edit history, downloaded and processed in 2008. The format of the file is defined in the AE1 spec sheet. Said file is already stored on HDFS as "/user/bd4-ae1/enwiki-20080103-full.txt". Under the same directory you will also find several cut-down versions of this file, including (in increasing file size) "/user/bd4-ae1/enwiki-20080103-sample.txt", "/user/bd4-ae1/enwiki-20080103-largersample.txt" and "/user/bd4-ae1/enwiki-20080103-perftest.txt". You can use the first two as toy datasets to test your code against. A similar setup exists for the 2nd AE; more details to be given at the hand-out.