Spring 2019

E599/B649 High Performance Big Data Systems

Lab Tasks

SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING
COMPUTER SCIENCE

Lab 2 of FA18-BL-ENGR-E599-33796

Goal

The objectives and the estimated timing of this lab session are:

- Connection to Juliet Cluster (5 min)
- Allocation of node resource (5 min)
- Install and configure Hadoop (20 min)
- Install Harp (20 min)
- Run K-means example (10 min)

We assume that you have subscribed to a futuresystems account (with an username) in the last lab session and have some basic knowledge of the following items:

- Linux command line and bash
- SSH tools
- Git/Github
- Maven
- Command line editor: e.g., Vi/Vim

If you are not familiar with any Linux command line editor. You may edit files at your desktop/laptop environment and upload them to the server side by using `scp` command

```
$ scp file/at/local/path ${user.name}@juliet.futuresystems.org:/remote/path
```

Deliverables

Run the Harp K-means example. Submit the results output file (.txt) to Canvas folder

Evaluation

Lab participation: credit for 1 point based upon a successful completion of the lab tasks

SSH to Juliet cluster at Futuresystems

MacOS and Linux

Open Terminal

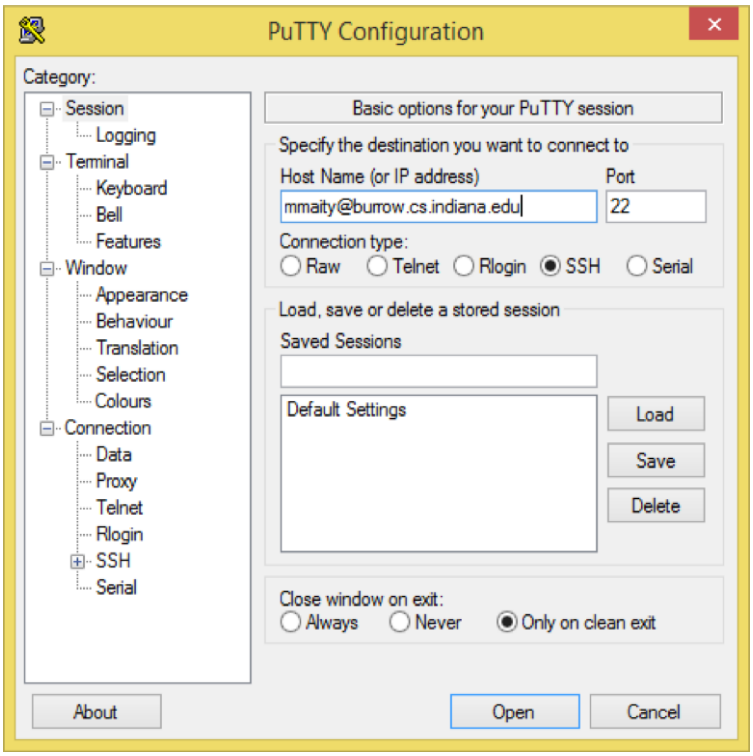
```
ssh username@juliet.futuresystems.org
```

Enter your passphrase if your public key has not been uploaded

Windows

Use *Putty*

- Enter Host Name as username@juliet.futuresystems.org
- Enter 22 for Port
- Choose *SSH* as connection type



Register one Juliet node

Use *slurm* commands to allocate node resource

Check the resource availability

```
[username@j-login1 ~]$ sinfo
```

It returns the list of nodes

```
[sakkas@j-login1 ~]$ sinfo
PARTITION  AVAIL  TIMELIMIT  NODES   GRES  STATE NODELIST
juliet        up   infinite      1 (null)  down* j-093
juliet        up   infinite      1 (null)  drain j-077
juliet        up   infinite     21 (null)   resv j-[012-027,078,087-089,115]
juliet        up   infinite     98 (null)  alloc j-[001-010,029-034,036-041,043-076,080-086,090-092,094,096
-114,116-124,126-128]
juliet        up   infinite      7 (null)   idle j-[011,028,035,042,079,095,125]
romeo         up   infinite      4  gpu:8    mix r-[001-004]
volta         up   infinite      2  gpu:8    mix r-[005-006]
KNL7250       up   infinite      1 (null)  down* t-018
KNL7250       up   infinite      3 (null)  alloc t-[001,005,012]
KNL7250       up   infinite     44 (null)   idle t-[002-004,006-010,015,021,023-042,044,047,050,053,055-064
]
KNL7290       up   infinite      4 (null)  alloc t-[013-014,016-017]
KNL7290       up   infinite     12 (null)   idle t-[011,019-020,022,043,045-046,048-049,051-052,054]
KNL           up   infinite      1 (null)  down* t-018
KNL           up   infinite      7 (null)  alloc t-[001,005,012-014,016-017]
KNL           up   infinite     56 (null)   idle t-[002-004,006-011,015,019-064]
aurora        up   infinite      1 (null)    mix aurora01
aurora        up   infinite      1 (null)   idle aurora02
[sakkas@j-login1 ~]$
```

Allocate one node

21 juliet nodes j-[012-027,078,087-089,115] are reserved with reservation name "sakkas_4" for this lab session.

```
[username@j-login1 ~]$ salloc --no-shell --no-kill -p juliet --reservation=sakkas_4 -
w $Node bash
```

Here *$Node* is the name of available node such as *j-012*. The file *Juliet-Node-Allocation.csv* assigns each of you a distinct Juliet node. Check whether your allocation is successful by

```
[username@j-login1 ~]$ squeue
```

```
[sakkas@j-login1 ~]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES   GRES NODELIST(REASON)
                26    juliet     bash    pengb  R 157-03:33:59      1 (null) j-030
               195   KNL7290     bash    pengb  R 148-04:07:24      1 (null) t-013
               246    juliet     bash     lc37  R 145-05:11:20      1 (null) j-006
               281       KNL     bash sabraosn  R 143-22:59:19      2 (null) t-[012,014]
               471     romeo     bash ajpiergi  R 136-23:11:35      1 (null) r-002
               473    juliet     bash   sakkas  R 136-20:24:44      1 (null) j-029
               534   KNL7250     bash     lc37  R 134-05:24:12      1 (null) t-001
               541     romeo   (null) miajiang  R 134-03:54:15      1 (null) r-002
               542     romeo   (null) miajiang  R 134-03:50:28      1 (null) r-002
               596    juliet     bash hrakholi  R 131-09:20:48      1 (null) j-031
               768     romeo     bash   lijguo  R 123-21:33:02      1  gpu:1 r-002
               772    aurora     bash     lc37  R 123-08:55:59      1 (null) aurora01
               810   KNL7250     bash     lc37  R 120-08:10:43      1 (null) t-005
               865     romeo     bash miajiang  R 115-04:16:45      1 (null) r-002
               874       KNL     bash sabraosn  R 114-08:51:48      1 (null) t-016
               875       KNL     bash sabraosn  R 114-08:40:32      1 (null) t-017
               927    juliet     bash fg474adm  R 106-23:11:31     16 (null) j-[110-114,116-124,126-127]
               951    juliet     bash vlabeyko  R 105-10:28:39      7 (null) j-[103-109]
              1001    juliet     bash fg474adm  R 101-05:53:21      2 (null) j-[101-102]
              1425     volta     bash  kimsunw  R 73-06:55:51      1 (null) r-006
              1493    juliet     bash fg474adm  R 64-03:14:58      1 (null) j-128
              1495    juliet     bash vlabeyko  R 62-09:55:57      4 (null) j-[097-100]
              1496    juliet     bash vlabeyko  R 62-02:42:24      5 (null) j-[001-005]
              1767     volta     bash  scwager  R 53-20:51:51      1 (null) r-006
              1818     volta     bash  kimsunw  R 45-05:56:44      1 (null) r-005
              1893     romeo     bash pmorpari  R 42-04:09:39      1  gpu:1 r-002
              1911     romeo     bash pmorpari  R 40-06:50:29      1  gpu:1 r-002
              1963    juliet   (null)   styagi  R 30-02:52:38      1 (null) j-094
              1974     volta     bash   lijguo  R 28-01:46:39      1  gpu:1 r-006
              2036     romeo     bash    pengb  R 22-19:51:29      1 (null) r-002
              2045    juliet     bash   styagi  R 22-06:27:22      1 (null) j-096
              2047    juliet     bash   styagi  R 22-06:26:10      1 (null) j-090
              2048    juliet     bash   styagi  R 22-06:26:06      1 (null) j-091
              2087    juliet     bash   styagi  R 16-03:50:01      1 (null) j-072
              2106     romeo     bash emhassan  R 8-21:37:10      1  gpu:1 r-002
              2107     romeo     bash emhassan  R 8-21:34:24      1  gpu:1 r-002
              2112     romeo     bash emhassan  R 8-21:29:17      1  gpu:1 r-002
              2182    juliet     bash pulasthi  R 5-05:28:48     50 (null) j-[032-034,036-041,043-071,073-076,080-086,092]
              2184     romeo     bash   shujon  R 5-05:28:15      1  gpu:1 r-003
              2218     romeo     bash   shujon  R 3-17:56:30      1  gpu:2 r-004
              2219     romeo     bash   shujon  R 4-02:13:18      1  gpu:1 r-001
              2222     romeo     bash   lijguo  R 3-23:17:41      1  gpu:1 r-001
              2223     romeo     bash   lijguo  R 3-23:13:34      1  gpu:1 r-003
              2224     romeo     bash emhassan  R 3-22:37:56      1  gpu:1 r-001
              2225     romeo     bash emhassan  R 3-22:32:55      1  gpu:1 r-001
              2226     romeo     bash pmorpari  R 3-22:24:13      1  gpu:1 r-001
              2227     romeo     bash pmorpari  R 3-22:22:52      1   gpu: r-002
              2228     romeo     bash   lijguo  R 3-17:56:30      1  gpu:1 r-004
              2248    juliet     bash     lc37  R 2-04:32:04      4 (null) j-[007-010]
              2257    juliet     bash   sakkas  R       0:06      1 (null) j-012
```

Finally, login into the allocated node (e.g., j-012)

```
[username@j-login1 ~]$ ssh j-012
```

Install and setup Hadoop

Install JDK

Juliet cluster has already installed a JDK at the path `bash /opt/jdk1.8.0_101`

Install Hadoop 2.6.0

Download and extract the hadoop-2.6.0 binary into your machine.

It's available at hadoop-2.6.0.tar.gz.

```
$ mkdir ~/Hadoop
$ cd ~/Hadoop
$ wget https://archive.apache.org/dist/hadoop/core/hadoop-2.6.0/hadoop-2.6.0.tar.gz
$ tar -xvzf hadoop-2.6.0.tar.gz
```

Set the environment variables in file `~/.bashrc` .

Open the *.bashrc* script at your home directory.

```
$ vim ~/.bashrc
```

Export the environment variables pointed to your installation path of Java and Hadoop

```
export JAVA_HOME=/opt/jdk1.8.0_101
export HADOOP_HOME=$HOME/Hadoop/hadoop-2.6.0
export YARN_HOME=$HADOOP_HOME
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export PATH=$HADOOP_HOME/bin:$JAVA_HOME/bin:$PATH
```

Run the following command to make sure the changes are applied.

```
$ source ~/.bashrc
```

Check if environment variables are set correctly by running the following command.

```
$ hadoop
```

The results should look similar to the example below.

```
Usage: hadoop [--config confdir] COMMAND
where COMMAND is one of:
fs                   run a generic filesystem user client
version              print the version
jar <jar>            run a jar file
checknative [-a|-h]  check native hadoop and compression libraries availability
distcp <srcurl> <desturl> copy file or directories recursively
archive -archiveName NAME -p <parent path> <src>* <dest> create a hadoop archive
classpath            prints the class path needed to get the
credential           interact with credential providers
Hadoop jar and the required libraries
daemonlog            get/set the log level for each daemon
trace                view and modify Hadoop tracing settings
or
CLASSNAME            run the class named CLASSNAME
Most commands print help when invoked w/o parameters.
```

Modify the following files in the Apache Hadoop distribution.

- `$HADOOP_HOME/etc/hadoop/hadoop-env.sh`

Add the $JAVA_HOME path

```
export JAVA_HOME=/opt/jdk1.8.0_101
```

- `$HADOOP_HOME/etc/hadoop/core-site.xml` :

```
$ vim $HADOOP_HOME/etc/hadoop/core-site.xml
```

Copy the following text into the file and replace *${user.name}* with your user name and *${namenode}* with the current juliet node name (e.g., j-012)

```
<configuration>
<property>
<name>fs.default.name</name>
<value>hdfs://${namenode}:9010</value>
</property>
<property>
<name>hadoop.tmp.dir</name>
<value>/tmp/hadoop-${user.name}</value>
<description>A base for other temporary directories.</description>
</property>
</configuration>
```

- `$HADOOP_HOME/etc/hadoop/hdfs-site.xml` :

```
$ vim $HADOOP_HOME/etc/hadoop/hdfs-site.xml
```

Copy the following text into the file, where *${hadoop_home}* refers to the installation directory of your hadoop (e.g. $HADOOP_HOME)

```
<configuration>
<property>
<name>dfs.hosts</name>
<value>${hadoop_home}/etc/hadoop/slaves</value>
</property>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property>
<name>dfs.namenode.http-address</name>
<value>${namenode}:50271</value>
</property>
<property>
<name>dfs.namenode.secondary.http-address</name>
<value>${namenode}:50291</value>
</property>
</configuration>
```

- `$HADOOP_HOME/etc/hadoop/mapred-site.xml` :

You will be creating this file. It does not exist in the original package.

```
$ vim $HADOOP_HOME/etc/hadoop/mapred-site.xml
```

Copy the following text into the file.

```
<configuration>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
<property>
<name>mapreduce.map.collective.memory.mb</name>
<value>100000</value>
</property>
<property>
<name>mapreduce.map.collective.java.opts</name>
<value>-Xmx90000m -Xms90000m</value>
</property>
</configuration>
```

- `$HADOOP_HOME/etc/hadoop/yarn-site.xml` :

```
$ vim $HADOOP_HOME/etc/hadoop/yarn-site.xml
```

Copy the following text into the file.

```
<configuration>
<property>
<name>yarn.resourcemanager.hostname</name>
<value>${namenode}</value>
</property>
<property>
<name>yarn.resourcemanager.address</name>
<value>${namenode}:8132</value>
</property>
<property>
<name>yarn.resourcemanager.scheduler.address</name>
<value>${namenode}:8230</value>
</property>
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
<property>
<name>yarn.nodemanager.log-dirs</name>
<value>/tmp/hadoop-${user.name}</value>
</property>
<property>
<name>yarn.scheduler.maximum-allocation-mb</name>
<value>128000</value>
</property>
<property>
<name>yarn.nodemanager.resource.memory-mb</name>
<value>120000</value>
</property>
<property>
<name>yarn.nodemanager.delete.debug-delay-sec</name>
<value>10000000</value>
</property>
</configuration>
```

- `$HADOOP_HOME/etc/hadoop/slaves`:

```
$ vim $HADOOP_HOME/etc/hadoop/slaves
```

Update the `slaves` by adding your ${namenode} (e.g., j-012). If you do have other node, also add their names in `slaves`

```
${namenode}
${other node 1}
${other node 2}
...
```

Start the Hadoop service

Format the file system using the following code.

```
$ hdfs namenode -format
```

You should be able to see it exit with status 0 as follows.

```
...
...
xx/xx/xx xx:xx:xx INFO util.ExitUtil: Exiting with status 0
xx/xx/xx xx:xx:xx INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at xxx.xxx.xxx.xxx
```

Launch NameNode, SecondaryNameNode and DataNode daemons.

```
$ $HADOOP_HOME/sbin/start-dfs.sh
```

Launch ResourceManager and NodeManager Daemons.

```
$ $HADOOP_HOME/sbin/start-yarn.sh
```

Check if the daemons started successfully by running the following command.

```
$ jps
```

The output should look similar to the following text with `xxxxx` replaced by the process ids for "NameNode", "SecondaryNameNode", etc.

```
xxxxx NameNode
xxxxx SecondaryNameNode
xxxxx DataNode
xxxxx NodeManager
xxxxx Jps
xxxxx ResourceManager
```

If all the processes listed above aren't in your output recheck your configurations and rerun the commands in this section after executing the following commands.

```
$ $HADOOP_HOME/sbin/stop-dfs.sh
$ $HADOOP_HOME/sbin/stop-yarn.sh
$ rm -r /tmp/hadoop-${user.name}
```

Install Harp

Clone Harp repository

It is available at DSC-SPIDAL/harp.

Run the following code on the login node. (We need git-lfs and it is already installed on the login node).

```
$ logout
[username@j-login1 ~] $ git clone https://github.com/DSC-SPIDAL/harp.git
```

Login into the allocated node again. (e.g., j-012)

```
[username@j-login1 ~]$ ssh j-012
```

Set the environment variables in file `~/.bashrc`.

```
$ vim ~/.bashrc
```

Add the following text into the file. Replace `<where Harp locates>` with the path of where Harp is located in your system.

```
export HARP_ROOT_DIR=<where Harp locates>
#e.g. ~/harp
export HARP_HOME=$HARP_ROOT_DIR/core/
```

Run the following command to make sure the changes are applied.

```
$ source ~/.bashrc
```

Install Maven

Harp uses *Maven* to compile the source code.

```
## download maven binary
cd ~
wget http://apache.spinellicreations.com/maven/maven-3/3.5.4/binaries/apache-maven-3.5.4-bin.tar.gz
## unzip maven
tar xzvf apache-maven-3.5.4-bin.tar.gz
```

Add maven bin path to bashrc script, `vim ~/.bashrc`

```
export PATH=$HOME/apache-maven-3.5.4/bin:$PATH
```

Apply the changes

```
source ~/.bashrc
```

Check the installation of maven by

```
mvn -v
```

Should have something like

```
Apache Maven 3.5.4 (1edded0938998edf8bf061f1ceb3cfdeccf443fe; 2018-06-17T14:33:14-04:00)
Maven home: /N/u/sakkas/apache-maven-3.5.4
Java version: 1.8.0_101, vendor: Oracle Corporation, runtime: /opt/jdk1.8.0_101/jre
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "3.10.0-862.9.1.el7.x86_64", arch: "amd64", family: "unix"
```

Compile harp

Select the profile related to your hadoop version (For ex: hadoop-2.6.0) and compile using maven. Supported hadoop versions are 2.6.0, 2.7.5 and 2.9.0.

```
$ cd $HARP_ROOT_DIR
$ mvn clean package -Phadoop-2.6.0
```

Install harp plugin to hadoop as demonstrated below.

```
$ cp core/harp-collective/target/harp-collective-0.1.0.jar $HADOOP_HOME/share/hadoop/
mapreduce/
$ cp core/harp-hadoop/target/harp-hadoop-0.1.0.jar $HADOOP_HOME/share/hadoop/mapreduc
e/
$ cp third_party/fastutil-7.0.13.jar $HADOOP_HOME/share/hadoop/mapreduce/
```

You have completed the Harp installation.

Run Harp Example

Copy harp examples to `$HADOOP_HOME` using the following code.

```
$ cp $HARP_ROOT_DIR/ml/java/target/harp-java-0.1.0.jar $HADOOP_HOME
```

Make sure you are in the `$HADOOP_HOME` folder.

```
cd $HADOOP_HOME
```

Make sure that the namenode is not int the safe mode

```
hdfs dfsadmin -safemode leave
```

```
$ hadoop jar harp-java-0.1.0.jar edu.iu.kmeans.regroupallgather.KMeansLauncher <num o
f points> <num of centroids>
<vector size> <num of point files per worker> <number of map tasks> <num threads> <nu
mber of iteration> <work dir> <local points dir>
```

- `<num of points>` --- the number of data points you want to generate randomly
- `<num of centriods>` --- the number of centroids you want to clustering the data to
- `<vector size>` --- the number of dimension of the data
- `<num of point files per worker>` --- how many files which contain data points in each worker
- `<number of map tasks>` --- number of map tasks
- `<num threads>` --- how many threads to launch in each worker
- `<number of iteration>` --- the number of iterations to run
- `<work dir>` --- the root directory for this running in HDFS
- `<local points dir>` --- the harp kmeans will firstly generate files which contain data points to local directory. Set this argument to determine the local directory.

For example:

```
$ hadoop jar harp-java-0.1.0.jar edu.iu.kmeans.regroupallgather.KMeansLauncher 1000 1
0 100 5 1 2 10 /kmeans /tmp/kmeans
```

To fetch the results, use the following command:

```
$ hdfs dfs –get <work dir> <local dir>
#e.g. hdfs dfs -get /kmeans ~/Document
```

To see the results:

```
$ cat ~/Document/kmeans/centroids/out/output
```

You can copy the values and submit the results to the Canvas.