Submitting Spark Applications

Starting an interactive session:

- 1. Start an interactive session asking three nodes, using the **start_interactive** script . This script requests by default one node from the y15 reservation **for an hour**:
 - a. Modify ./start_interactive.sh. script with your reservation number and walltime.

```
\#qsub -IVl select=3:ncpus=36,walltime=01:00:00,place=scatter:excl -A y15 -q <reservation number> -j oe
```

This will give you an interactive session into a node (e.g. node r1i3n0) and you will see something like this:

```
[USERNAME@r1i3n0~]$./start_interactive.sh
qsub: waiting for job 399686.indy2-login0 to start
qsub: job 399686.indy2-login0 ready
[USERNAME@r1i3n0~]$
```

Starting a Spark cluster:

2. Start the spark cluster using the *start_spark* script. It will configure a spark cluster, with the master running in a node, and three workers (with 72 cores per worker) running in each reserved node.

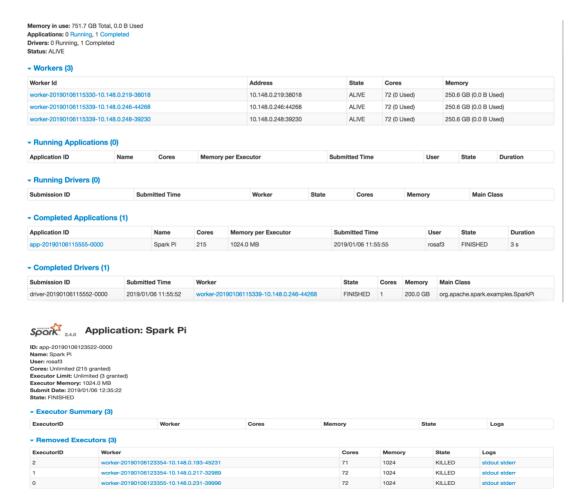
```
[USERNAME@r1i3n0~]$./start_spark.sh
3 node(s) assigned
Autoloading gcc/6.2.0
starting org.apache.spark.deploy.master.Master, logging to
/lustre/home/y15/rosaf3/spark-2.4.0-bin-hadoop2.7/logs/spark-
rosaf3-org.apache.spark.deploy.master.Master-1-r1i3n0.out
rli3n0: starting org.apache.spark.deploy.worker.Worker, logging to
/lustre/home/y15/rosaf3/spark-2.4.0-bin-hadoop2.7/logs/spark-
rosaf3-org.apache.spark.deploy.worker.Worker-1-r1i3n0.out
rli3n14: starting org.apache.spark.deploy.worker.Worker, logging
to /lustre/home/y15/rosaf3/spark-2.4.0-bin-hadoop2.7/logs/spark-
rosaf3-org.apache.spark.deploy.worker.Worker-1-r1i3n14.out
rli3n13: starting org.apache.spark.deploy.worker.Worker, logging
to /lustre/home/y15/rosaf3/spark-2.4.0-bin-hadoop2.7/logs/spark-
rosaf3-org.apache.spark.deploy.worker.Worker-1-r1i3n13.out
starting org.apache.spark.deploy.history.HistoryServer, logging to
/lustre/home/y15/rosaf3/spark-2.4.0-bin-hadoop2.7/logs/spark-
rosaf3-org.apache.spark.deploy.history.HistoryServer-1-r1i3n0.out
```

Submitting a Spark Application to the Spark Cluster – using cluster mode:

3. Submit the *SparkPi* application (which is an example that comes with spark) to your Spark Cluster that you have started by using *spark_submit_SparkPi* script.

Note: You will have to modify this script to replace the master node (—master spark://MASTER NODE:7077), with the node where you master is running.

- 4. Check the result inside the spark-2.4.0-bin-hadoop2.7/work/ directory. The result will be a driver directory corresponding with this application:
 - a. more spark-2.4.0-bin-hadoop2.7/work/driver-20190106121316-0001/stdout | grep Pi \rightarrow Pi is roughly 3.14035114035
- 5. To monitor the spark applications in your spark cluster, just launch another terminal session and run something like this:
 - a. ssh USER@login.cirrus.ac.uk -L8080:MASTER NODE:8080
 - b. Web browser → localhost:8080



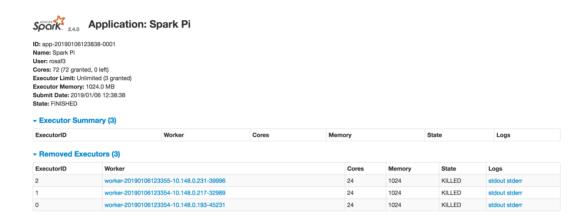
Tuning Resource Allocations:

6. We can also modify the number of cores to use per worker/executor, by using the —total-executor-cores in our spark-submission script.

For example, if we decide to use 72 cores for running this spark application:

>> spark-2.4.0-bin-hadoop2.7/bin/spark-submit --verbose --class org.apache.spark.examples.SparkPi --master spark://r1i2n5:7077 --deploy-mode cluster --total-executor-cores 72 spark-2.4.0-bin-hadoop2.7/examples/jars/spark-examples 2.11-2.4.0.jar 10

We can see that the spark cluster will run the application using the 3 workers/executors, but using only 24 cores per worker/executor.



Submitting a Spark Application to the Spark Cluster – using client mode:

7. We can also submit the wordcount.py application stored in *Spark_Applications* folder, using the **client deployment mode**.

<u>In cluster mode</u>, the Spark driver runs inside an application master process which is managed by cluster-manager on the cluster, and the client can go away after initiating the application. <u>In client mode</u>, the driver runs in the client process, and the application master is only used for requesting resources from spark cluster.

>> \$SPARK_HOME/bin/bin/spark-submit --verbose --master spark://r1i2n5:7077 --deploy-mode client wordcount.py <FILE.TXT>

replace <FILE.TXT> by a file

Important: Currently, the **standalone mode** does not support **cluster mode** for **Python applications**.