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 5 hours:
 - a. Modify ./start_interactive.sh. script with your reservation number and walltime.

```
\#qsub -IVl select=3:ncpus=36,walltime=00:30:00,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:

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. Remember to use source for exporting the environment variables into your
PATH.

```
[USERNAME@r1i3n0~]$ source./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
```

Note: In this example, master node is r1i3n0

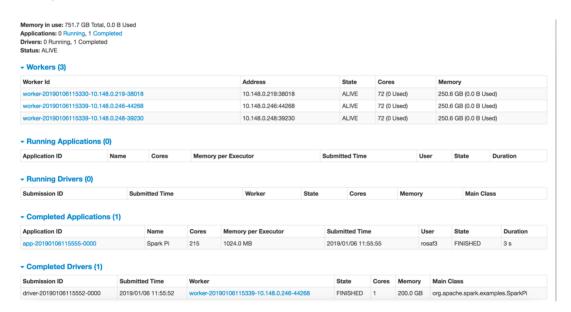
Submitting a Spark Application to the Spark Cluster – using cluster mode:

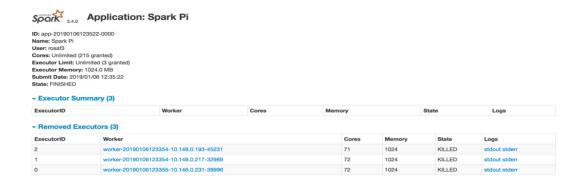
3. Submit the SparkPi application (which is an example that comes with spark) to your Spark Cluster that you have started by using spark_submit_SparkPi script, indicating your master node as the first argument (e.g. r1i3n0), which corresponds to the node assigned in your interactive session.

./spark_submit_SparkPi.sh r1i3n0

- 4. Check the result inside the \$HOME/spark-2.4.0-bin-hadoop2.7/work/ directory. The result will be a driver directory corresponding with this application:
 - a. more \$HOME/spark-2.4.0-bin-hadoop2.7/work/driver-20190106121316-0001/stdout | grep Pi → Pi is roughly 3.1403511404
- 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

Note: In our example, MASTER NODE is r1i3n0. This node could be different each time you start an interactive session.





Tuning Resource Allocations:

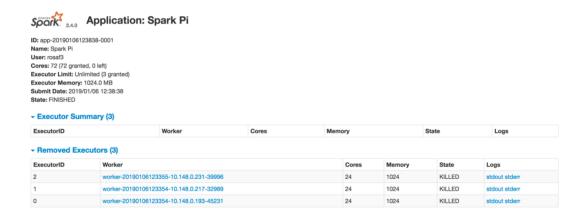
6. We can also modify the number of cores to use per worker/executor, by using the —total-executor-cores in our spark-submission script.

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:// *r1i3n0*: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

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.

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 <u>client deployment mode</u>.

<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.

>> export PYSPARK_PYTHON=/lustre/sw/anaconda/anaconda3-5.1.0/bin/python3

>> \$SPARK_HOME/bin/spark-submit --verbose --master spark:// r1i3n0:7077 --deploy-mode client wordcount.py \$HOME/spark-2.4.0-bin-hadoop2.7/README.md

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.

You can replace \$\frac{\\$HOME/spark-2.4.0-bin-hadoop2.7/README.md}\$, by other FILE that you will like to use as an input for counting the words in it.

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