Running Spark

Spark Local: your computer

From the [Spark download page](https://spark.apache.org/downloads.html), get Spark version 2.3.1 (the version we'll be using on the cluster), “Pre-built for Hadoop 2.7 and later”, and click the “download Spark” link. Unpack that somewhere you like. Set an environment variable so you can find it easily later:

export SPARK\_HOME=/home/you/spark-2.3.1-bin-hadoop2.7/

export PYSPARK\_PYTHON=python3

If you have [iPython](https://ipython.org/) installed, you can also export PYSPARK\_DRIVER\_PYTHON=ipython to use it in the pyspark shell.

Then you can start the pyspark shell or a standalone job like this:

${SPARK\_HOME}/bin/pyspark

${SPARK\_HOME}/bin/spark-submit sparkcode.py

While the job is running, you can access the web frontend at <http://localhost:4040/>.

Spark Local: lab computer

Spark is already available on the machines. Set an environment variable so you can get to it easily later:

export SPARK\_HOME=/usr/shared/CMPT/big-data/spark-2.3.1-bin-hadoop2.7

export PYSPARK\_PYTHON=python3

Then you can start the pyspark shell or a standalone job like this:

${SPARK\_HOME}/bin/pyspark

${SPARK\_HOME}/bin/spark-submit sparkcode.py

While the job is running, you can access the web frontend at <http://localhost:4040/>.

Cluster

You will need a newer version of Spark than the default, which you can enable with (each time you log in):

module load spark

Spark will be in your path and you can get started:

pyspark

spark-submit sparkcode.py

Debugging & Exceptions

On the cluster, you won't be able to see any exceptions your code throws (since they are happening on some node out there in the cluster). There are two ways to deal with this…

After a run, you can (as always) see each process' output:

yarn logs -applicationId application\_1234567890\_1234 | less

Or you can run a job in yarn-client mode. This moves the driver (your code) to the login node, so any exceptions thrown there are visible. Please don't do this often, since it shifts work from the cluster back to the login node (which can run out of memory if everybody is doing this and running pyspark):

spark-submit --master yarn-client sparkcode.py

Monitoring Jobs

In the YARN web front end ([http://localhost:8088](http://localhost:8088/) if you have your ports forwarded as in the [Cluster](https://coursys.sfu.ca/2018fa-cmpt-732-g1/pages/Cluster) instructions), you can click your app while it's running, then the “ApplicationMaster” link.

If you're on campus, the link will work. If not, you can replace “nml-cloud-199.cs.sfu.ca” with “localhost” in the URL and it should load. (Or if you really want, in your OS' /etc/hosts file, add 127.0.0.1 nml-cloud-199.cs.sfu.ca and the links will work.)

After the job has finished, you can also use the yarn logs command to get the stdout and stderr from your jobs, as described in the [Cluster](https://coursys.sfu.ca/2018fa-cmpt-732-g1/pages/Cluster)instructions.

Spark and PyPy

[PyPy](http://pypy.org/) is a Python implementation that includes a [Just-In-Time compiler](https://en.wikipedia.org/wiki/Just-in-time_compilation) that can be astonishingly fast. It can be used with Spark to speed up the Python code execution. (In Python Spark, your logic is split between the Scala/JVM implementation of the core logic and the Python implementation of your logic and parts of the PySpark API.)

In general for Spark, you need to set the PYSPARK\_PYTHON variable to the command to start the pypy executable.

On the cluster, you can do this:

module load spark

module load pypy3

This sets the PYSPARK\_PYTHON variable to point to a recent PyPy version and SPARK\_YARN\_USER\_ENV so the PYSPARK\_PYTHON is set on the executors as well.