YARN

Igor Yakushin
ivy2@uchicago.edu

YARN

- You have already seen how to execute programs written in Java, PySpark, Hive, HBase, Pig.
- When you execute any program in Hadoop cluster, it is queued by YARN resource manager
- You can examine what is running or queued on the Hadoop cluster from a command line with `yarn application --list`:

```
[ivy2@md01 ~]$ yarn application --list
WARNING: YARN_OPTS has been replaced by HADOOP_OPTS. Using value of YARN_OPTS.
Total number of applications (application-types: [], states: [SUBMITTED, ACCEPTED, RUNNING] and tags: []):5
Application-Id                  Name           Type   User       Queue                 State    Final-State  Progress Tracking-URL
application_1568600009828_2148  pyspark-shell  SPARK    zzhao9     root.users.zzhao9   RUNNING  UNDEFINED  10%  http://md01.rcc.local:4042
application_1568600009828_2146  pyspark-shell  SPARK    zzhao9     root.users.zzhao9   RUNNING  UNDEFINED  10%  http://md01.rcc.local:4044
application_1568600009828_2141  pyspark-shell  SPARK  shermanw   root.users.shermanw   RUNNING  UNDEFINED  10%  http://md01.rcc.local:4040
application_1568600009828_2145  pyspark-shell  SPARK  zwang151   root.users.zwang151   RUNNING  UNDEFINED  10%  http://md01.rcc.local:4043
application_1568600009828_2142  pyspark-shell  SPARK  shermanw   root.users.shermanw   RUNNING  UNDEFINED  10%  http://md01.rcc.local:4041
```

- If you ssh to hadoop with `-Y` option and your ssh client supports X-forwarding (under Mac you might have to install XQuartz for that), you can run firefox there and point it to the Tracking-URL associated with each job to see more details about it.

YARN

- `yarn top` allows to understand how busy the cluster is and who uses how much resources:

```
[ivy2@md01 ~]$ yarn top
YARN top - 13:40:33, up 84d, 17:27, 0 active users, queue(s): root
NodeManager(s): 5 total, 5 active, 0 unhealthy, 0 decommissioned, 1 lost, 0 rebooted
Queue(s) Applications: 5 running, 2129 submitted, 0 pending, 2118 completed, 6 killed, 0 failed
Queue(s) Mem(GB): 0 available, 768 allocated, 1251 pending, 22 reserved
Queue(s) VCores: 0 available, 176 allocated, 278 pending, 5 reserved
Queue(s) Containers: 176 allocated, 278 pending, 5 reserved
APPLICATIONID                  USER       TYPE  QUEUE PRIOR   #CONT  #RCONT  VCORES RVCORES     MEM    RMEM  VCORESECS    MEMSECS %PRO
application_1568600009828_2141 shermanw   spark root.users.shermanw     0     160       0     160       0    716G      0G     768515    341
application_1568600009828_2145 zwang151   spark root.users.zwang151     0       7       0       7       0     28G      0G      27933     11
application_1568600009828_2146 zzhao9     spark root.users.zzhao9       0       4       0       4       0      8G      0G      15901    338
application_1568600009828_2142 shermanw   spark root.users.shermanw     0       4       0       4       0     14G      0G      54486     19
application_1568600009828_2149 zzhao9     spark root.users.zzhao9       0       1       1       1       5      1G     22G      29829   1297
```

- This command works similar to the standard UNIX `top`, refreshing the screen every few seconds. To exit from it, press `Ctrl-C`
- The two most important lines from the above are:

```
Queue(s) Mem(GB): 0 available, 768 allocated, 1251 pending, 22 reserved
Queue(s) VCores: 0 available, 176 allocated, 278 pending, 5 reserved
```

- The first line says that 768 GB of memory is allocated and 0 are available
- The second line says that 176 CPU cores are allocated and 0 are available

YARN

- Therefore, if you start the job, it will not run for a while until enough memory and computing cores become available
- Notice, that if you use Spark interactively via Jupyter, you are holding cluster resources even if you are not running anything especially if you pin RDD or Data Frame to memory with cache()
- We have rather small Hadoop cluster and it is better to submit job to it in batch rather than lock resources using interactive Jupyter notebook
- There is a cronjob running on login node, killing jupyter notebooks and the corresponding pyspark jobs that spent in queue more than N hours. When this was written, N was 24 hours.
- It is very important to exit Jupyter notebook properly to release resources

YARN

- The cluster becomes particularly busy and unusable at the end of each semester when students are finishing their final projects
- To kill running job, use

```
yarn application -kill <Application ID>
```

- You can find application id using one of the previous commands
- To get logs for a particular application:

```
yarn logs -applicationId <Application ID>
```

- When you launch a batch job under YARN, it runs in shell foreground, the shell will not retun until the job is finished.
- To run it in a background, use standard UNIX way (see my Linux tutorial for details). For example:

```
nohup spark-submit myscript.py > out.log 2> error.log &
```