Northeastern

EECE5698
Parallel Processing for Data Analytics

Running Spark on a Standalone Cluster

Local Mode

- **Jobs are run on one machine**
- **Parallelized over cores**
- **For example, from an interactive node:**

```
pyspark --master local[10]
```

Standalone Cluster

- One node reserved as master – runs driver
- Other nodes reserved as workers – RDD partitions
- More setup steps required
- Reminder: execute commands from an interactive node

Setup

- Copy spark files

```
cp -r /scratch/spark …
                  /scratch/$USER
ls /scratch/$USER
```

Setup

- Modify .bashrc

```
#Uncomment this line to run spark in standalone mode
#source /scratch/$USER/spark/conf/spark-env.sh
```

Setup

- Modify .bashrc

```
#Uncomment this line to run spark in standalone mode
source /scratch/$USER/spark/conf/spark-env.sh
```

Setup

- Modify .bashrc

```
#Uncomment this line to run spark in standalone mode
source /scratch/$USER/spark/conf/spark-env.sh
```

- We're ready to launch a cluster!

Step 1: Launch a master node

```
cd /scratch/$USER/spark

sbatch spark-master.slurm

#If all nodes on default partition are taken
sbatch -p PARTITION spark-master.slurm
```

Step 1: Launch a master node

```
#Confirm a master node has been allocated
squeue –u $USER #or sq

#Obtain node’s IP address
traceroute NODE #Node name from queue
```

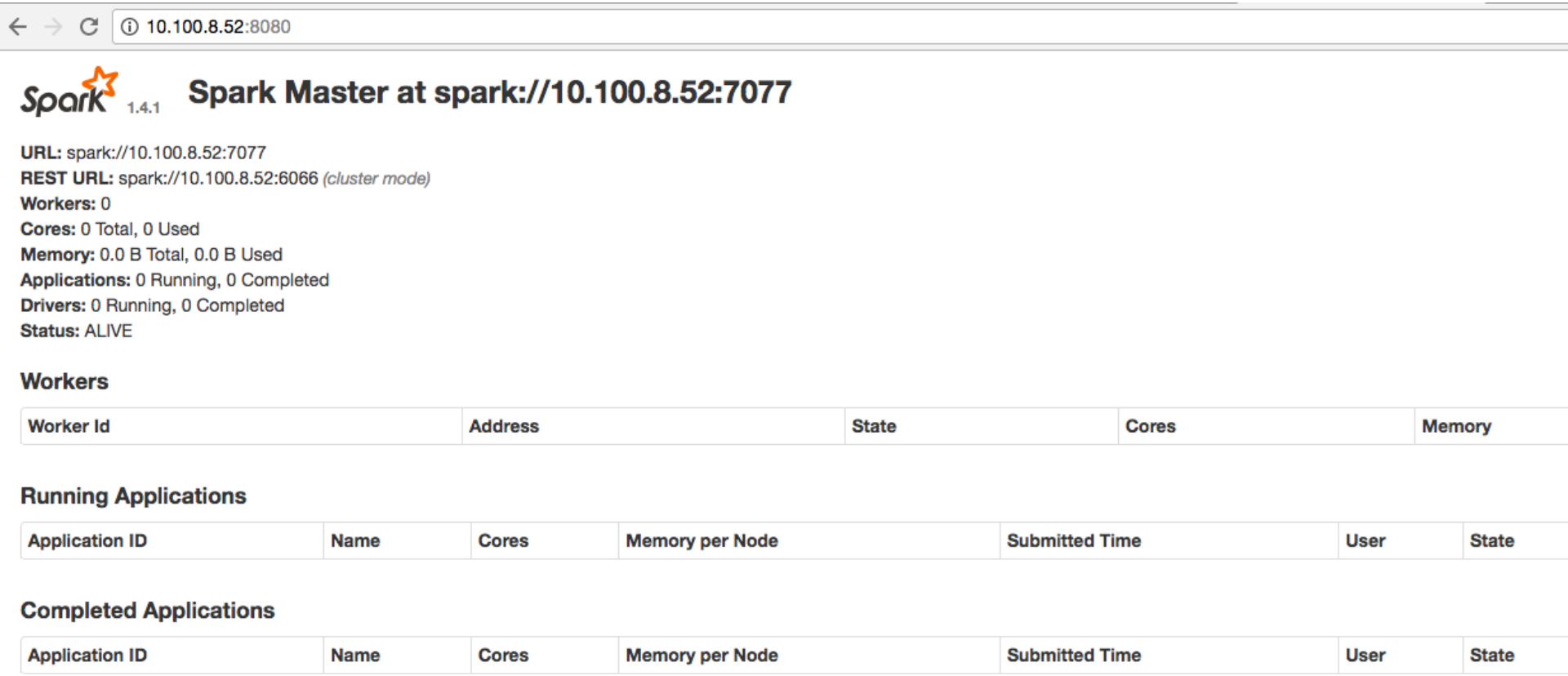
Open Cluster GUI

- From within Northeastern network, in a browser enter:

```
IPADDRESS:8080
```

where IPADDRESS is the IP address obtained from the traceroute command

Open Cluster GUI



Step 2: Add workers

```
cd /scratch/$USER/spark

#NUM: Number of workers to add
#PARTITION: Discovery partition from which to add nodes
#IPADDRESS: IP address of master node (from traceroute)
sbatch –N NUM --partition PARTITION …
       --time=01:00:00
      spark-workers.slurm spark://IPADDRESS:7077

#Check that workers are in your queue
squeue –u $USER
```

Step 3: Submitting Jobs

- Three ways to submit Spark jobs (local or cluster mode)
  - `pyspark` shell
  - `spark-submit`
  - Python – explicit specification

```
sc = SparkContext(master='spark://MASTER_IP:7077', \
    appName='MyApp')
```

Example: pyspark shell

```
pyspark --master spark://MASTER_IP:7077

rdd = sc.parallelize(range(1000))

total,count =rdd.map(lambda x: (x,1))\
                .reduce(lambda x,y:
(x[0]+y[0],x[1]+y[1]) )

print 1.*total/count
```

Example: WordCounter

```
import sys
from pyspark import SparkContext

if __name__ == '__main__':
    sc = SparkContext(master='local[10]', appName='WordCount')
    lines = sc.textFile(sys.argv[1])

    lines.flatMap(lambda s: s.split()) \
         .map(lambda word: (word, 1)) \
         .reduceByKey(lambda x, y: x + y) \
                .sortBy(lambda (x,y):y, ascending=False) \
         .saveAsTextFile(sys.argv[2])
```

Example: WordCounter - Direct Specification in Python

```
import sys
from pyspark import SparkContext

if __name__ == '__main__':
    sc = SparkContext(master='spark://MASTER_IP:7077', \

       appName='WordCount')
    lines = sc.textFile(sys.argv[1])

    lines.flatMap(lambda s: s.split()) \
         .map(lambda word: (word, 1)) \
         .reduceByKey(lambda x, y: x + y) \
                .sortBy(lambda (x,y):y, ascending=False) \
         .saveAsTextFile(sys.argv[2])
```

```
python WordCounter.py INPUT OUTPUT
```

Example: WordCounter – Using SparkSubmit

```
import sys
from pyspark import SparkContext

if __name__ == '__main__':
    sc = SparkContext(appName='WordCount')
    lines = sc.textFile(sys.argv[1])

    lines.flatMap(lambda s: s.split()) \
         .map(lambda word: (word, 1)) \
         .reduceByKey(lambda x, y: x + y) \
                .sortBy(lambda (x,y):y, ascending=False) \
         .saveAsTextFile(sys.argv[2])
```

```
spark-submit --master spark://MASTER_IP:7077
       WordCounter.py INPUT OUTPUT
```