Spark on Clusters

Spark Cluster Options

- Standalone – a simple cluster manager included with Spark that makes it easy to set up a cluster.
- Apache Mesos – a general cluster manager that can also run Hadoop MapReduce and service applications.
- Hadoop YARN – the resource manager in Hadoop.
- Kubernetes – an open-source system for automating deployment, scaling, and management of containerized applications.

Standalone Cluster (1)

- Spark needs to be installed on the master and all of the nodes. For the lab, we installed Spark on cscluster00. Since all nodes share the home file system, there is no need to install Spark on the nodes.
- Start the master (typically, we would use cscluster00.boisestate.edu)

```
[amit@cscluster00 ~]$ start-master.sh
starting org.apache.spark.deploy.master.Master, logging to
/home/amit/spark-install/spark/logs/spark-amit-org.apache.
    spark.deploy.master.Master-1-cscluster00.boisestate.edu.
    out
```

- Once started, the master will print out a spark://HOST:PORT URL for itself, which you can use to connect workers to it, or pass as the "master" argument to SparkContext or to spark-submit.

```
[amit@cscluster00 ~]$ cat /home/amit/spark-install/spark/
    logs/spark-amit-org.apache.spark.deploy.master.Master-1-
    cscluster00.boisestate.edu.out | grep spark:
19/11/13 23:48:56 INFO Master: Starting Spark master at
    spark://cscluster00.boisestate.edu:7077
```

- The master's web UI is at http://localhost:8080, where we can also find the master's URL.

Standalone Cluster (2)

- Start one or more workers on multiple nodes by passing in the Spark URL of the master. We need password-less ssh access.

```
[amit@cscluster00 ~]$ ssh cscluster01
Last login: Tue Nov 12 14:27:47 2019 from cscluster00.
    boisestate.edu

[amit@cscluster01 ~]$ start-worker.sh spark://cscluster00.
    boisestate.edu:7077

starting org.apache.spark.deploy.worker.Worker, logging to /
    home/amit/spark-install/spark/logs/spark-amit-org.apache
    .spark.deploy.worker.Worker-1-cscluster01.boisestate.edu
    .out
```

- Once you have started a worker, look at the master's web UI. You should see the new node listed there, along with its number of CPUs and memory (minus one gigabyte left for the OS).
- We can now run Spark jobs on the cluster. For example:

```
spark-submit --class "WordCount" --master spark://
    cscluster00.boisestate.edu:7077  wordcount-basic.jar
    input  2> log 1> output
```

- Finally, we can stop the master and worker daemons.

```
[amit@cscluster00 ~]$ ssh cscluster01 stop-worker.sh
stopping org.apache.spark.deploy.worker.Worker
[amit@cscluster00 ~]$ stop-master.sh
stopping org.apache.spark.deploy.master.Master
```

Standalone Cluster (3)

- We can automate the start/stop of a standalone cluster. Add the names of the worker nodes to the file ~/spark-install/spark/conf/workers. Then use the start-all.sh and stop-all.sh scripts.

```
[amit@cscluster00 ~]$ start-all.sh
starting org.apache.spark.deploy.master.Master, logging to /
    home/amit/spark-install/spark/logs/spark-amit-org.apache
    .spark.deploy.master.Master-1-cscluster00.boisestate.edu
    .out
cscluster03.boisestate.edu: starting org.apache.spark.deploy
    .worker.Worker, logging to /home/amit/spark-install/
    spark/logs/spark-amit-org.apache.spark.deploy.worker.
    Worker-1-cscluster03.boisestate.edu.out
cscluster01.boisestate.edu: starting org.apache.spark.deploy
    .worker.Worker, logging to /home/amit/spark-install/
    spark/logs/spark-amit-org.apache.spark.deploy.worker.
    Worker-1-cscluster01.boisestate.edu.out

[amit@cscluster00 ~]$ stop-all.sh
cscluster03.boisestate.edu: stopping org.apache.spark.deploy
    .worker.Worker
cscluster01.boisestate.edu: stopping org.apache.spark.deploy
    .worker.Worker
stopping org.apache.spark.deploy.master.Master
```

Standalone Cluster with HDFS

- A spark program running on local host can connect with Hadoop HDFS running in pseudo-distributed mode on the same local host.
- A spark program running on a standalone cluster needs to connect with Hadoop HDFS running on a cluster, typically the same cluster.
- Here is a simple setup to get Hadoop working on a cluster of fixed nodes. First we need to configure Hadoop to run in real distrbuted mode.

```
cd CS535-resources
git pull --rebase
cd examples/Hadoop/local-scripts
setmode.sh distributed
```

- Edit the ~/hadoop-install/hadoop/etc/hadoop/workers file to add the names of the nodes in the standalone cluster. Edit the file ~/hadoop-install/hadoop/etc/hadoop/master to add cscluster00.boisestate.edu as the master.
- Then we create the HDFS filesystem and start the HDFS servers and load the input files. We don't need to start YARN (unless we want to submit spark jobs to YARN, which is possible).

```
hdfs namenode -format
start-dfs.sh
hdfs dfsadmin -report
hdfs dfs -mkdir -p /user/amit
hdfs dfs -put input input
```

- Now run the Spark program that accesses the HDFS. The submit command will stay the same.

Running Spark on Existing Cluster

- The `cscluster00` cluster already has Hadoop running on 8 nodes. Now it also has a Spark running on the same 8 nodes. To use it, simply load up your HDFS files and submit a spark job as follows:

```
spark-submit --class "WordCount" --master spark://
    cscluster00.boisestate.edu:7077  wordcount-basic.jar
    input  2> log 1> output
```