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.

Spark on the cscluster

- Spark is already installed and configured on the master and all of the worker nodes.
- Once you tunneel in via SSH (for port 8080), the spark master's web UI is at `http://localhost:8080`, where we can also find the master's URL to use for your spark programs.
- The spark master's URL is `spark://cscluster00.boisestate.edu:7077`. You can pass it as the "master" argument to `SparkContext` or to `spark-submit`.
  Now we can run a spark program on the cluster as follows (we are redirecting standard stream 1) – otherwise all of the words will come out on the console!). Note that we are assuming a shared local filesystem so all Spark nodes can access the data.

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

Spark on the cscluster (2)

- 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.

```
spark-submit --class "WordCount" --master spark://cscluster00.boisestate
    .edu:7077 wordcount-hdfs-save.jar hdfs://cscluster00:9000/user/amit
    /input
```

- Note that the `wordcount-hdfs-save.jar` is a modified version of the word count program that gets the input from HDFS using the following:

```
sc.textFile("hdfs://cscluster00.boisestate.edu:9000/user/amit/input");
```

- Note that the `wordcount-hdfs-save.jar` is a modified version of the word count program that saves the output to HDFS using the following:

```
counts.saveAsTextFile("hdfs://cscluster00.boisestate.edu:9000/user/amit/
    output");
```

Using Jupyter Lab remotely on `cscluster`

- Use SSH to tunnel to `cscluster00` (use your correct Boise Stateuser name if it is different on your machine) as follows. I chose the port 9088 arbitrarily to not conflict with other users. Choose a different port if you get a conflict.

  ```
  ssh -L 9088:localhost:9088 cscluster00
  ```

- Then on `cscluster00`, start `jupyter lab` as follows to route the interface back to your machine:

  ```
  jupyter lab --no-browser --port=9088
  ```

- Then open up a tab in your browser on your local machine and use the following URL (where the token is shown in the output from jupyter lab when you start it as above.

  ```
  localhost:9088/lab?token=...
  ```

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