Spark Setup instructions for Linux environment

Reference links:

<https://github.com/amplab/shark/wiki/Running-Shark-Locally>

<https://github.com/amplab/shark/wiki/Running-Shark-on-a-Cluster>

Running Shark Locally

This guide describes how to get Shark up and running on a cluster. If you are interested in using Shark on Amazon EC2, see page [[Running Shark on EC2]] to use the set of EC2 scripts to launch a pre-configured cluster in a few mins.

## Dependencies ##

\*\*NOTE: Shark is a \_drop-in\_ tool that can be used on top of existing Hive warehouses. It does not require that you change your existing Hive deployment \_in any way\_.\*\*

Running Shark on a cluster requires the following external components:

\* [Scala 2.9.3](http://www.scala-lang.org/)

\* [Spark 0.8.0](http://www.spark-project.org)

\* A compatible Java Runtime: OpenJDK 7, Oracle HotSpot JDK 7, or Oracle HotSpot JDK 6u23+

\* The Shark-specific Hive JAR (based on [Hive 0.9](http://hive.apache.org/)), included in the Shark binary distribution

\* A HDFS cluster: setup not included in this guide.

Note that unlike earlier version of Spark and Shark, running the latest version on a cluster NO LONGER requires Apache Mesos.

### Scala

If you don't have Scala 2.9.3 installed on your system, you can download it by:

$ wget http://www.scala-lang.org/files/archive/scala-2.9.3.tgz

$ tar xvfz scala-2.9.3.tgz

### Spark

We are using Spark's standalone deployment mode to run Shark on a cluster. You can [click on this click to find more information](http://spark-project.org/docs/latest/spark-standalone.html).

Download Spark:

$ wget http://spark-project.org/download/spark-0.8.0-incubating-bin-hadoop1.tgz # Hadoop 1/CDH3 - or -

$ wget http://spark-project.org/download/spark-0.8.0-incubating-bin-cdh4.tgz # Hadoop 2/CDH4

$ tar xvfz spark-\*-bin-\*.tgz

$ mv spark-\*-bin-\*/ spark-0.8.0

Edit `spark-0.8.0/conf/slaves` to add the hostname of each slave, one per line.

Edit `spark-0.8.0/conf/spark-env.sh` to set SCALA\_HOME and SPARK\_WORKER\_MEMORY

export SCALA\_HOME=/path/to/scala-2.9.3

export SPARK\_WORKER\_MEMORY=16g

SPARK\_WORKER\_MEMORY is the maximum amount of memory that Spark can use on each node. Increasing this allows more data to be cached, but be sure to leave memory (e.g. 1 GB) for the OS and any other services that the node may be running.

## Shark

Download the binary distribution of Shark 0.8. The package contains two folders, `shark-0.8.0` and `hive-0.9.0-shark-0.8.0-bin`.

$ wget https://github.com/amplab/shark/releases/download/v0.8.0/shark-0.8.0-bin-hadoop1.tgz # Hadoop 1/CDH3 - or -

$ wget https://github.com/amplab/shark/releases/download/v0.8.0/shark-0.8.0-bin-cdh4.tgz # Hadoop 2/CDH4

$ tar xvfz shark-\*-bin-\*.tgz

$ cd shark-\*-bin-\*

Now edit `shark-0.8.0/conf/shark-env.sh` to set the HIVE\_HOME, SCALA\_HOME and MASTER environmental variables. The master URI must \_exactly match\_ the `spark://` URI shown at port 8080 of the standalone master.

export HADOOP\_HOME=/path/to/hadoop

export HIVE\_HOME=/path/to/hive-0.9.0-shark-0.8.0-bin

export MASTER=<Master URI>

export SPARK\_HOME=/path/to/spark

export SPARK\_MEM=16g

source $SPARK\_HOME/conf/spark-env.sh

The last line is there to avoid setting SCALA\_HOME in two places. Make sure SPARK\_MEM is not larger than SPARK\_WORKER\_MEMORY set in the previous section.

If you are using Shark on an existing Hive installation, be sure to set HIVE\_CONF\_DIR (in shark-env.sh) to a folder containing your configuration files. Alternatively, copy your Hive XML configuration files into Shark's `hive-0.9.0-bin/conf`. For example:

cp /etc/hive/conf/\*.xml /path/to/hive-0.9.0-bin/conf/

Copy the Spark and Shark directories to slaves. We assume that the user on the master can SSH to the slaves. For example:

$ while read slave\_host; do

$ rsync -Pav spark-0.8.0 shark-0.8.0 $slave\_host

$ done < /path/to/spark/conf/slaves

Launch the cluster by running the Spark cluster launch scripts:

$ cd spark-0.8.0

$ ./bin/start-all.sh

## Configuring with Hadoop2/CDH4

The newest versions of Hadoop require additional configuration options. You may need to set the following values inside of Hive's configuration file (hive-site.xml):

\* `fs.default.name`: Should point to the URI of your HDFS namenode. E.g. hdfs://myNameNode:8020/

\* `fs.defaultFS`: Should be equal to `fs.default.name`

\* `mapred.job.tracker`: Should list the host:port of your JobTracker or be set to "NONE" if you are only using Spark. Note that this needs to be explicitly set even if you aren't using a JobTracker.

\* `mapreduce.framework.name`: Should be set to a non-empty string, e.g. "NONE".

## Testing

You can now launch Shark with the command

$ ./bin/shark-withinfo

More detailed information on Spark standalone scripts and options is [also available](https://github.com/mesos/spark/wiki/Running-A-Spark-Standalone-Cluster).

To verify that Shark is running, you can try the following example, which creates a table with sample data:

CREATE TABLE src(key INT, value STRING);

LOAD DATA LOCAL INPATH '${env:HIVE\_HOME}/examples/files/kv1.txt' INTO TABLE src;

SELECT COUNT(1) FROM src;

CREATE TABLE src\_cached AS SELECT \* FROM SRC;

SELECT COUNT(1) FROM src\_cached;

See the [Shark User Guide](https://github.com/amplab/shark/wiki/Shark-User-Guide) for more details on using Shark.