Tobi Bosede
550.700
9/7/16

Master's Research - Spark Documentation

Apache Spark (Spark) was originally designed to be used with the scala API. However Spark also has APIs for other languages, specifically java, python, and R. In my master's research I am using the python API, called pyspark, because I am very familiar with python and pyspark has a great deal of support from the open source community.

The purpose of this document is to provide guidance on potential problems that may arise in setting up and configuring Spark. I assume here that you have Spark 1.6.1 and its dependencies installed and will be using pyspark.

Some Basic Terminology

- **SparkContext** - main entry point for Spark functionality. A SparkContext represents the connection to a Spark cluster.

- **RDD (Resilient Distributed Dataset)** - a fault tolerance collection of elements that can be operated on in parallel; can be created via a SparkContext, typically a variable named *sc.*

Examples
sc.parallelize([1,"ABC", (3,4)]).count() creates a RDD from an array
sc.textFile("data.txt") creates an RDD from a text file

- **Log4j** (not to be confused with Py4J) - what Spark uses for logging, can be configured by editing the `log4j.properties` file which can be created from `log4j.properties.template` in the /conf directory

To quiet the Log4j logger so it only displays messages when there is an error, in the `log4j.properties` file change *log4j.rootCategory=INFO, console* to *log4j.rootCategory=ERROR, console.*

- **Py4J** - allows python programs to access objects in the Java Virtual Machine (JVM)

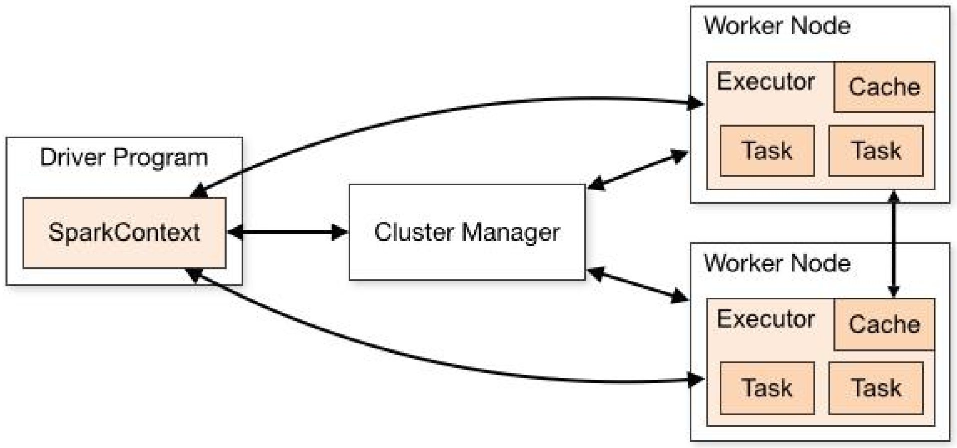
Since Java is in charge of distributing jobs to workers and we are using the python API for Spark, we need a way for the two languages to communicate.

Standalone Mode

In real-world scenarios, we are interested in harnessing the computing power of multiple machines to do analyses. So we create a cluster and this cluster needs a manager. I decided that Spark should be deployed as a standalone cluster because this avoids the need for a third party cluster manger such as Hadoop NextGen (YARN), Amazon EC2, or Apache Mesos. Spark can run without a cluster manager, but when we do this we are not making the best use of Spark. In standalone mode, even if we have just one machine we can create a pseudo-distributed system with a driver (master) and workers (slaves). They can be started as follows:

1)Navigate to the directory of the shell scripts
cd /<location of spark instance>/sbin
2) Start master:
./start-master.sh
3) Start worker:
./start-slave.sh <master-spark-URL>
4) Start history node:
./start-history-server.sh

There are similar scripts in the *sbin* directory to stop the cluster. The following diagram provides a visual of the relationships between the aforementioned.



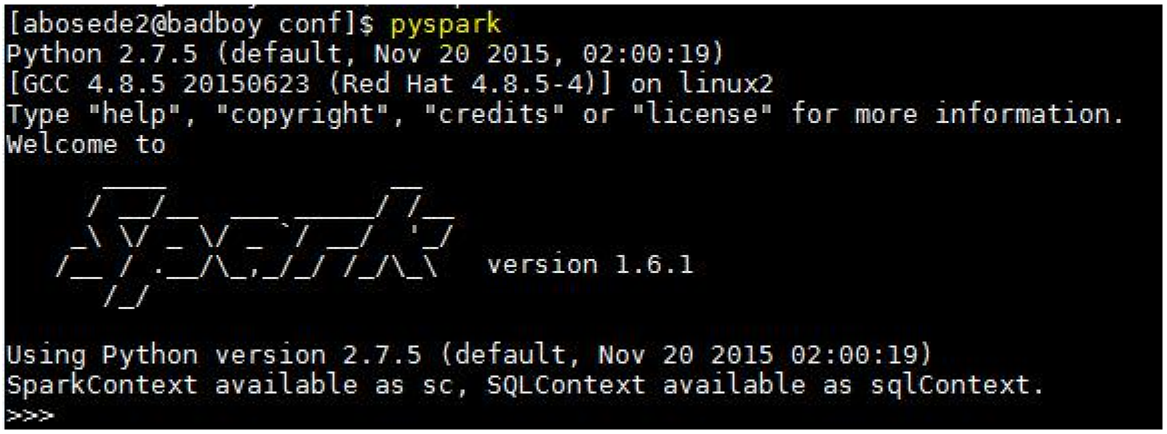
Ways to Submit Jobs/Applications

- **The Interactive Shell**

The interactive shell is for testing. There a spark context has already been created and configured for us. To exit the shell use Ctrl+D.

To start the shell navigate to the location of your Spark instance. Then type:
./bin/pyspark

You can avoid all this typing be creating an alias. All you would have to type as in the below image is:
pyspark.



```
[abosede2@badboy conf]$ pyspark
Python 2.7.5 (default, Nov 20 2015, 02:00:19)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-4)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 1.6.1
      /_/

Using Python version 2.7.5 (default, Nov 20 2015 02:00:19)
SparkContext available as sc, SQLContext available as sqlContext.
>>>
```

- **spark-submit script**

For programs in which SparkContext is defined use:
spark-submit <location of spark instance>/examples/src/main/python/<program name>.py <arguments to be passed into program>

Example using pi.py
    spark-submit /usr/local/spark-1.6.1/examples/src/main/python/pi.py 10

Make sure to add *sc.stop()* to your python program, since only one spark context can be in use at any moment.

- **Interfacing via Juptyer Notebook**

A notebook is useful in sharing Spark applications and visualizing their results. I have chosen the Jupyter notebook because of its predecessor IPython and my comfort with IPython.

We allow Jupyter to import the SparkContext from pyspark by adding the following lines to our bashrc file.

```
export SPARK_HOME="<location of spark instance>"
export PYTHONPATH=$SPARK_HOME/python/:$PYTHONPATH
```

We should now get no errors when we type:

```
from pyspark import SparkContext
```

Make sure to add *sc.stop()* to your notebook, since only one spark context can be in use at any moment.

Monitoring

We can look at two different UIs to monitor jobs or applications. One is at the 4040 port and the other is at 8080.

The UI at the 4040 port is temporary and is only up when a spark context exists. Additional jobs will bind to successive ports (i.e. 4041, 4042, etc).

The UI at the 8080 port is for the cluster. Therefore it not only shows the status of the master, workers, and every job submitted, but also allows information to persist after the SparkContext has been stopped. However, each time the cluster is stopped existing information disappears from 8080 UI.

- **Specifying the Master**

When monitoring an application, it is very important to specify the master for the job or be aware of what the default master is. For instance if the default master is local, but the 8080 UI is for a master that is not local, you will not see your job on the 8080 UI if you did not specify the master URL for the UI.

Examples (we specify the master in the highlighted, 16 is the number of threads)

1) For spark-submit

```
spark-submit --master local[16] /usr/local/spark-1.6.1/examples/src/main/python/pi.py 10
spark-submit --master spark://10.160.5.48:7077 /usr/local/spark-1.6.1/examples/src/main/python/pi.py 10
```

2) For the shell

```
pyspark --master local[16]
pyspark --master spark://10.160.5.48:7077
```

3) For Jupyter notebook

```
from pyspark import SparkConf
sconf = SparkConf().setMaster("local")
sconf = SparkConf().setMaster("spark://10.160.5.48:7077")
```

We can set the default master in the configuration file, `spark-defaults.conf`, which can be created from `spark-defaults.conf.template`. For example we can add:

```
spark.master                    spark://10.160.5.48:7077
```

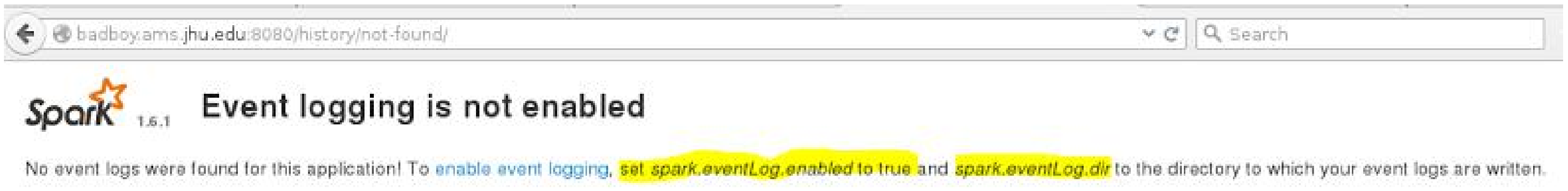
- **Enabling Event Logging**

Another important aspect of using the 8080 UI is enabling the logs. For example, we can add the below to `spark-defaults.conf`:

```
spark.eventLog.enabled          true
spark.eventLog.dir              file:///usr/local/spark-1.6.1/logs
```

If this is done as root, make sure that permissions are set such that all non-root users can still submit jobs.

By enabling event logging, we avoid getting the bellow message in the 8080 UI.



Event logging is not enabled

No event logs were found for this application! To enable event logging, set spark.eventLog.enabled to true and spark.eventLog.dir to the directory to which your event logs are written.

Packages

Packages are available to provide functionality not built into Spark. They are hosted at spark-packages.org. This website is similar to R's CRAN.

- **spark-csv**

I have found the spark-csv package to be a useful addition which allows for CSV files can be read as DataFrame for use in SparkSQL. Spark-csv parses data as well as infers schema. To include version: 1.4.0-s_2.11 of this package in your Spark applications type:

```
pyspark --packages com.databricks:spark-csv_2.11:1.4.0
spark-submit --packages com.databricks:spark-csv_2.11:1.4.0
```

To add this package to your default application configurations either:

1) add the line below to `spark-defaults.conf`

```
spark.jars.packages     com.databricks:spark-csv_2.11:1.3.0
```

OR

2) add the line below to your bashrc file

```
export PACKAGES="com.databricks:spark-csv_2.11:1.4.0"
export PYSPARK_SUBMIT_ARGS="--packages $PACKAGES pyspark-shell"
```

For more details on any of these topics and Spark in general, go to http://spark.apache.org/docs/latest/.