Report - *The Scala of a Python*

Hara Kumar (harak@kth.se) | Enrique Perez Soler (eps@kth.se)

Introduction

This project compares the performance of following configurations: Spark+Scala on one machine, Spark+Scala on 3 machine (2 executor) cluster, and Spark+PySpark on one machine. This project aims to answer 2 questions:

1) When working with large datasets, one would expect the Spark+Scala on 3 machine cluster to have the best performance, but constructing RDDs and coordinating between different machines has overhead costs, so how would performance be affected when working with relatively small data sets? The answer to this question can be found by comparing the results of the Spark+Scala on one machine configuration with the Spark+Scala on cluster configuration.
2) The scripts will all take advantage of Spark RDDs, for which the Scala language was designed for. This is opposed to PySpark which contains an additional layer to be python friendly. So does Spark+PySpark perform worse than Spark+Scala? The answer to this question can be found by comparing the results Spark+Scala one machine experiment with the Spark+PySpark one machine experiment.

2. Data

The data we will be using is the server log data from Lab 1. We have extracted the code used in the section “Unique Hosts per day” (the advanced step after the Unique Hosts script). The code was modified slightly to read the log file from an HDFS (necessary for the clustered experiment) and to measure execution time, which is the measure of comparison in these experiments. The code was also rewritten in PySpark for the Spark+PySpark on one machine experiment.

3. Methodology

Google Cloud Compute Engine was used to run the different experiments. A Ubuntu VM was initiated and Spark, Scala and Python were installed in it (mainly, other libraries were added like *sbt* to run the code). After these were configured correctly, new instances where created based on that original one to transfer with it the configuration as well.

The hardware for the single node machines are 8 vCPUs, 30 GB memory, while the hardware for the cluster is 4 vCPUs, 15 GB memory per node. The cluster has a total of 3 machines (1 coordinator, and 2 executors). This means the hardware of the two executors is equal to the

hardware capacity of the single node machines. However, the cluster system has greater hardware capacity compared to the single node machines when including the coordinator node. This hardware configuration was chosen as preliminary experiments showed that the hardware capacity of the coordinator was less influential than the capacity of the nodes.

4. Results

	Trial 1	Trial 2	Trial 3	Average
Spark+Scala (one machine)	20.4 seconds	13.4 seconds	14.7 seconds	16.2 seconds
Spark+Scala (cluster)	23.3 seconds	15.2 seconds	17.3 seconds	18.6 seconds
Spark+PySpark (one machine)				

As was expected, the Spark+Scala on a single machine outperforms the cluster. Vertical scaling is a better if the option to do so is available, as it saves on performance overhead due to coordination and ethernet communication.

5. How to replicate results?

Spark + Scala (one machine OR cluster)
Build clusters using Google VMs using the hardware configurations mentioned above. Install java 8, spark, hdfs, and sbt in all machines (or install it one machine and clone the drives). Then upload the apache.log file into the hdfs drive. In the UniqueHosts.scala file, change line 43 to point to the hdfs location of the apache log file. Then using the load command, execute the scala file in a spark-shell terminal. Ensure that the spark shell is properly configured and is able to utilize all available nodes, when applicable.

PySpark
For this section, python, pyspark, and all dependencies needs to be installed. To access pyspark and executed a sample program to test its correct configuration. The script used as a sample was named *examplepython.py* and it did the following:

```
from pyspark import SparkContext
from pyspark.sql import HiveContext

sc = SparkContext()
SQLContext = HiveContext(sc)
```

```
SQLContext.setConf("spark.sql.hive.convertMetastoreOrc", "false")
txt = SQLContext.sql( "SELECT 1")
txt.show(2000000, False)
```

We proceeded to run the script with the command.

```
$ spark-submit examplepython.py
```

For the PySpark script we were planning on translating the same scala code with the same transformations and actions to python. Unfortunately we couldn't manage to fully convert all the desired transformations and actions as well as we would have wanted. The fact that the scala code made use of RDDs and these gave some issues in PySpark when managing the log files and other semantic issues prevented us to fully test and run the experiment to measure the performance comparison between both. The code is provided in the zip file to showcase our efforts in the matter. If only we could have more time and more availability we could have found the small issues that jeopardized our last experiment.