Project Spark and scalability

Daniel Hagimont
hagimont@enseeiht.fr

The goal of this project is to experiment with Spark in cluster mode and to evaluate its scalability. The cluster mode will be simulated on your laptop with Docker. Docker allows you to create containers which run an operating system (Ubuntu). Such containers can be seen as virtual machines, so that your laptop simulates a cluster of machines.

1. Docker

You can find many tutorials about Docker on the net. The installation method depends on your operating system. I provide below the method I used for an Ubuntu system and a little tutorial. You don't have to be experts with Docker. Just manage to have it installed on your computer and work properly.

Install Docker

```
wget -qO- https://get.docker.com/ | sh
sudo adduser $USER docker
```

Verify that Docker is well installed

```
docker run hello-world
```

Download a Ubuntu image from the hub

```
docker pull ubuntu
```

List the images you have locally, each image has a name, tag and ID

```
docker images
```

Remove an image

```
docker rmi -f hello-world
```

Start a container (it = interactive mode, the container runs a bash)

```
docker run -it --name mycontainer ubuntu /bin/bash
```

List your running containers (from another terminal)

```
docker ps -a
```

You can customize your container (e.g. install apache2) and then, save an image of the container (from another terminal)

```
docker commit mycontainer myimage
```

You can stop your container (from another terminal)

```
docker stop mycontainer
```

You can remove your container (from another terminal)

```
docker rm mycontainer
```

For all these commands, you can use names or IDs as well

2. A Docker-based Spark Cluster

You are given a big archive which includes everything to run the Docker-based Spark Cluster. I assume that you have a shell (such as bash) to run script commands. Untar the archive and you will find the following directories :

hadoop-2.7.1

This is the installation of Hadoop. We are only using the HDFS part (the distributed file system). We are only interested in the etc/hadoop/slaves file which describes the machines where a DataNode is running (so machines where blocks can be stored). For the moment, we have 2 nodes (slave1 and slave2) because we will create an architecture with 3 docker containers (master, slave1 and slave2).

This installation is visible and used in each container instance.

spark-2.4.3-bin-hadoop2.7

This is the installation of Spark. We are only interested in the conf/slaves file which describes the machines where a Worker is running (so machines where computations can be performed). For the moment, we have 2 nodes (slave1 and slave2) for the same reason as above and because computations will take place where data are stored.

This installation is visible and used in each container instance.

containers

This is where you have everything for managing your container architecture. The scripts have to be launched from your computer (not in a container).

- server is the directory which includes everything to build a container. You should not have to modify anything there.
- build-image.sh is the script you must execute to build the image of the containers we will use during the experiment. You should execute this script once (or each time you modify the content of the container image, which should not happen during this project).
- clean-image.sh is used to delete the container image.
- kill-containers.sh is used to kill the containers running on your computer. Execute this script when you finished your experiments.
- start-containers.sh is used to start the architecture. It creates a master container (CONTNAME=master) which executes on core 3 (CONTCPU=3) and a set of slaves (2 slaves for the moment). You have to modifiy this script if you want more slaves. Just copy/past the 3 lines which create a slave and modify the CONTNAME and CONTCPU variables. You can see the available cores on your computer (for a Linux system) with "grep processor /proc/cpuinfo".

appli

This directory includes the applications (Python programs) to execute, and a set of scripts which allow starting and stopping Spark, and launching applications. This directory is visible in the container instances.

- WordCount.py is the WordCount application in Python. Notice that WordCount.py includes references to files in HDFS.
- generate.sh in a script which generates a big file (data.txt) by replicating the filesample.txt file. "./generated.sh 19" generates a file of size 2^19 = 512 Mb (as filesample.txt is 1 Kb). You can launch this script from your computer (not in a container).
- start.sh is used to start Spark in the cluster of Docker instances. It should be launched in

the master container. It will start Spark in all the container instances. Notice that if you want to add slaves, you have to modify this script (add an ssh command to each slave node).

- stop.sh is used to stop Spark in the cluster of Docker instances.
- copy.sh is used to copy data.txt into HDFS. It creates a directory /input in HDFS and copy data.txt into that directory. It also create a /output directory for the results. The WordCount application reads its input data from /input/data.txt in HDFS and produces results in /output in HDFS. You have to modify this script if you need to copy other data to HDFS.
- run.sh is used to run the application (Wordcount.py for the moment). You must modify this script to launch another application.

3. Demonstration with WordCount

If Docker is properly installled, you should be able to execute the WordCount application in the Docker-Based Spark Cluster as follows.

In appli

Generate a big file (data.txt)

./generate.sh 19 (512 Mb)

you should see a data.txt file of 512Mb

In containers

Build the container image

./build-image.sh

Start the 3 containers (master, slave1, slave2)

./start-containers.sh

you should see your 3 Docker instances running :

docker ps -a

Then, even if the master is running in background, you can log in the master :

docker exec -it master bash

You have then a command shell in the master container instance.

In the master

cd /root/appli

Start Spark

./start.sh

You can check that Spark started well on the master :

jps

You should see the following processes : NameNode, SecondaryNameNode, Master

You can check that Spark started well on slave1/slave2 :

ssh slave1

jps

You should see the following processes : DataNode, Worker

Get back to the master :

exit

From a web browser (firefox), you can visit

http://localhost:50070 and see the DataNode (and see after the copy step the blocks)

http://localhost:8080 and see the Workers (and later the executed applications)

Copy data in HDFS

./copy.sh

If /input and /output don’t exist or already exist, this is not an error.

Visit in the web browser the DataNodes to see that the blocks were created.

Launch the application

./run.sh

After execution for the WordCount, you can see the results :

hdfs dfs cat /output/*

4. Performance evaluations

Each Docker container instance is allocated only one core. You must evaluate the performance of WordCount with 1,2,3,4 slaves.

Integrate in this framework the application developped in the first part of the project and evaluate its performance (1,2.3.4 slaves).

To change the number of slaves/containers, you have to edit the following files :

- hadoop-2.7.1/etc/hadoop/slaves
- spark-2.4.3-bin-hadoop2.7/conf/slaves
- containers/start-containers.sh
- appli/start.sh

You may have to change the block size.

This can be done in the hadoop-2.7.1/etc/hadoop/hdfs-site.xml

The current size is 1048576 (16Mb). For the application developped in the first part of the project, you can experiment with 1Mb blocks.