TP1: Getting Started with *Hadoop*

Shadi Ibrahim
January 26th, 2017

MapReduce has emerged as a leading programming model for data-intensive computing. It was originally proposed by Google to simplify development of web search applications on a large number of machines.

Hadoop is a java open source implementation of MapReduce sponsored by Yahoo! The Hadoop project is a collection of various subprojects for reliable, scalable distributed computing. The two fundamental subprojects are the Hadoop MapReduce framework and the HDFS. HDFS is a distributed file system that provides high throughput access to application data. It is inspired by the GFS. HDFS has master/slave architecture. The master server, called NameNode, splits files into blocks and distributes them across the cluster with replications for fault tolerance. It holds all metadata information about stored files. The HDFS slaves, the actual store of the data blocks called DataNodes, serve read/write requests from clients and propagate replication tasks as directed by the NameNode.

The Hadoop MapReduce is a software framework for distributed processing of large data sets on compute clusters. It runs on the top of the HDFS. Thus data processing is collocated with data storage. It also has master/slave architecture. The master, called Job Tracker (JT), is responsible of : (a) Querying the NameNode for the block locations, (b) considering the information retrieved by the NameNode, JT schedule the tasks on the slaves, called Task Trackers (TT), and (c) monitoring the success and failures of the tasks.

The goal of this TP is to study the implementation and the operation of the Hadoop Platform. We will see how to deploy the platform and how to send and to retrieve the data to/from the HDFS. Finally we will run simple examples using the MapReduce paradigm.

Exercise 1: Installing Hadoop platform on your local machine

The goal of this exercise is to learn how to set up and configure a single-node Hadoop installation so that you can quickly perform simple operations using Hadoop MapReduce and the Hadoop Distributed File System (HDFS).

Question 1.1

Required software for Hadoop on Linux include:

- Java must be installed.
- ssh must be installed and sshd must be running to use the Hadoop scripts that manage remote Hadoop daemons.

You can check if the ssh daemon runs using the `ps` command:

```
$ps aux | grep sshd
```

If needed, you can install ssh as follows:

```
$sudo apt-get install ssh
```

To avoid typing your password each time you start the Hadoop daemons, you should create a couple of `ssh` keys thanks to the `ssh-keygen` command (if not already done so), then add the public key to the authorized keys.

```
ssh-keygen -t rsa -P ""
OR cat $HOME/.ssh/id_rsa.pub ≫ $HOME/.ssh/authorized_keys
```

Check the success of this step by running:
`ssh localhost`.
When runningthis command, you should not be prompted to type any password.

Question 1.2

Download the Hadoop platform from (http://people.irisa.fr/Shadi.Ibrahim/Nantes-2016.html). Extract the contents of hadoop-1.2.1.tar.gz in your home. To run Hadoop needs a `JAVA_HOME` environment variable specifies the directory of your `JVM`. Add the following lines at the end of your `$HOME/.bashrc` file:

```
export JAVA_HOME=<java path>
export HADOOP_PREFIX=<your Hadoop directory path>
export PATH=$PATH:$HADOOP_PREFIX/bin
```

To make take into account these changes in your open terminal type:
`. ~/.bashrc`

Alternatively, you may open a new terminal and these commands will automatically be executed.

Question 1.3

Before starting the Hadoop platform, edit its configuration files located in the `conf/` folder. As you are using your own machine, you must configure Hadoop as follows:

- In masters and slaves files (used to automate `ssh` connections) specify the name of your machine: `localhost`
- In `conf/hadoop-env.sh` set the JAVA_HOME variable consistently with your environment, for instance:

  Change
  `# export JAVA_HOME=/usr/lib/j2sdk1.5-sun`
  To
  `export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64`

- In `conf/core-site.xml` add:

```
<configuration>
  <property>
    <name>fs.default.name</name>
      <value>hdfs://localhost:9000</value>
    </property>
</configuration>
```

  This indicates the name of the machine and the associated network port on which the NameNode process will run. You should check that the 9000 port on your machine is not already in use:
  `netstat -a | grep tcp | grep LISTEN | grep 9000`

- In `conf/hdfs-site.xml` add:

```
<configuration>
   <property>
      <name>dfs.replication</name>
         <value>1</value>
      </property>
</configuration>
```

- In `conf/mapred-site.xml` add:

```
<configuration>
   <property>
      <name>mapred.job.tracker</name>
         <value>localhost:9001</value>
      </property>
</configuration>
```

This indicates the name of the machine and the associated network port on which the JobTracker process will run. You can check that the 9001 port on your machine is not already in use:
`netstat -a | grep tcp | grep LISTEN | grep 9001`

Question 1.4

We will now start the platform, and start all daemons:

- Format a new distributed-filesystem:
  $ `bin/hadoop namenode -format`
- Start the hadoop daemons:
  $ `bin/start-all.sh`
- The hadoop daemon log output is written to the `$HADOOP_PREFIX/logs`. Check them
- A nice command for checking if the expected Hadoop processes are running is `jps`. Try it when the Hadoop processes are running!

Question 1.5

To Browse the web interface for the NameNode and the JobTracker; by default they are available at:

- NameNode - http://localhost:50070/

- JobTracker - http://localhost:50030/

Question 1.6

To stop all the daemons running on your machine:
$ `bin/stop-all.sh`

Exercise 2: Using the Hadoop distributed file System (HDFS)

The goal of this exercise is to learn how to perform simple operations using the Hadoop Distributed File System (HDFS).

The FileSystem (FS) shell is invoked by `bin/hadoop fs <args>`. All the FS shell commands take path `URIs` as arguments. The `URI` format is `scheme://autority/path`. For HDFS the scheme is `hdfs`, and for the local filesystem the scheme is `file`. The scheme and authority are optional: *an HDFS file or directory such as* `/parent/child` *can be specified as*)

- `hdfs://namenodehost/parent/child` OR as
- `/parent/child`

FS shell:

```
hadoop fs -cat URI [URI ...] –Copies source paths to stdout.
hadoop fs -copyFromLocal <localsrc> URI
hadoop fs -copyToLocal [-ignorecrc] [-crc] URI <localdst>
hadoop fs -get [-ignorecrc] [-crc] <src> <localdst>
hadoop fs -ls <args>
hadoop fs -mkdir <paths>
hadoop fs -put <localsrc> ...  <dst>
hadoop fs -rmr URI [URI ...]
```

Question 2.1

During start up the NameNode loads the file system state from the fsimage and the edits log file. It then waits for DataNodes to report their blocks so that it does not prematurely start replicating the blocks though enough replicas already exist in the cluster. During this time NameNode stays in Safemode. Safemode for the NameNode is essentially a read-only mode for the HDFS cluster, where it does not allow any modifications to file system or blocks. Normally the NameNode leaves Safemode automatically after the DataNodes have reported that most file system blocks are available.

```
bin/hadoop dfsadmin -safemode get/enter/leave
```

Question 2.2

We will now generate (in the local file system) three files loremIpsum-100, loremIpsum-75 and loremIpsum-25. Their sizes are 100MB, 75MB and 25MB respectively. For this you use three times `loremIpsum` and `generator.sh` script file provided (also available here: http://people.irisa.fr/Shadi.Ibrahim/cbd-ens):

- `./generator.sh loremIpsum 100`
- `./generator.sh loremIpsum 75`
- `./generator.sh loremIpsum 25`

Now copy the three files to HDFS (`put`) and then show their content (`cat`) and finally remove them (`rmr`).

Exercise 3: Running your first MapReduce program

The goal of this exercise is to execute two MapReduce examples, typically used for benchmarking, which come with the default Hadoop distribution. At this point we will simply run the compiled programs (although it is not necessary for this exercise, if you are curious to see how the map and reduce functions are programmed, the code of this programs is available in the `$HADOOP_PREFIX/src/examples/org/apache/hadoop/examples/` directory).

Question 3.1

Run the wordcount example:

- `hadoop jar $HADOOP_PREFIX/hadoop-examples-*.jar wordcount input output`

where input is the directory containing the input files (e.g., loremIpsum-75) while the output is the directory that will be created to store the results.

Use the 100 MB, 75 MB datasets and check the outputs. Then, inspect the log files generated by this job.

Question 3.2

Run the Pi estimator:

- `hadoop jar $HADOOP_PREFIX/hadoop-examples-*.jar pi maps samples_per_map`

where maps represents the number of map tasks used and *samples_per_map* the number of points processed by each map task (i.e. the number of reducers).

Run the Pi estimator with 20 maps and 10000 *samples_per_map*. Check the log files.