ESIR-ABD Lab: Introduction to Spark

François Taiani ({`ftaiani.ouvaton.org`})[*]

1 Objective of the lab

This lab aims to provide a first introduction to the distributed computing framework Spark, starting from a single machine installation, and progressing to a two machine deployment, optionally with HDFS (hadoop's distributed file system, which Spark can use to manage data).

2 Preliminaries

2.1 VM installation

The lab uses the VM service of the ESIR / ISTIC platform: `http://vm.istic.univ-rennes1.fr/`.

- Create a new VM on the service.
  - Name your machine as "ESIR-ABD-FamilyName1-FamilyName2" where "FamilyNameX" are the two family names of your pair.
  - Indicate me as course convenor (*Enseignant responsable*).
  - Select a VM duration of 4 months.
  - Select Ubuntu 16 as your target system.
- Using an `ssh` shell, change the root password, create two new accounts for each of you, and add each of them to the sudoer group (`sudo`). From now on, use the two newly created accounts rather than the root account to connect to the VM. You are also strongly encouraged to create an RSA key to log into your VM with `ssh`, and to disable password based authentication (`PasswordAuthentication` and `ChallengeResponseAuthentication`) in `/etc/ssh/sshd_config` (see `man sshd_config` for more detail).
- If you plan to access your VM from outside the university network, you should request it to be accessible externally by raising a ticket on the university helpdesk (see the documentation attached to the VM).
- Using `apt-get update` (once) and `apt-get -f install` (until all error messages disappear), update your installation to the latest version of your packages.

2.2 Installing Scala

- The version of Spark we will use needs Scala 2.11, which should by now be installed on your system:
  - Check scala is properly installed with: `scala -version`

2.3 Installing sbt

`sbt` is the build automation program used by default by Spark.

- Use the `.deb` package available from `http://www.scala-sbt.org/release/docs/Installing-sbt-on-Linux.html` to install `sbt` as you have installed Scala.

[*]francois.taiani@irisa.fr

2.4 Installing Spark

Install Spark 2.0.0 by downloading the install archive from the course's moodle page (using either `wget` to download it directly from inside your VM, or `scp` to transfer it from your local machine to your VM). and following the instructions from `http://spark.apache.org/downloads.html`. (You are welcome to pick a more recent version, yet be aware that you might have to use a different scala version in this case, and that some of the detailed instructions of this brief might need to be adapted.) Choose a pre–built version with Hadoop 2.7. Check that your installation works by running some of the examples (for instance `./bin/run-example SparkPi`).

2.5 Reading about Spark, Scala and sbt

Before starting the lab, get some high level understanding of the technologies we will be using from the following sources:

- `https://spark.apache.org/docs/latest/programming-guide.html`
- Scala
  - `http://docs.scala-lang.org/tutorials/scala-for-java-programmers.html` (except for the last 3 topics: *Case Classes and Pattern Matching*, *Traits*, and *Genericity*)
  - `http://docs.scala-lang.org/overviews/collections/arrays.html`
- `http://www.scala-sbt.org/0.13/tutorial/Hello.html`

2.6 Executing your first Spark Self-Contained Application

Follow the instruction from `http://spark.apache.org/docs/2.0.0/quick-start.html#self-contained-applications` to compile and execute your first Spark application.

By default Spark log messages are printed on the console. These logs are useful to understand what is happening, but if you find them too verbose, you can set Spark's default log level output to WARN by creating a log4j configuration file (`log4j.rootCategory=WARN, console`, see `http://spark.apache.org/docs/latest/configuration.html#configuring-logging`). You can also decide to redirect Spark's log output to a file (usual setting in production).

Notes

- You might find `cat > somefile` and `mkdir -p /some/path/` useful.
- Change `simple.sbt` to the `scala` and `spark` versions you are using.
- `sbt` will probably download several jars to resolve your dependencies. That should only happen once, and is normal.

2.7 Some technical notes

- You should manage your source code using a versioning system, either `git` or `svn`, on a public forge, and provide me access to it. **Beware: Your code should not be publicly accessible. Make sure to make it private, and to invite me to it.**
- You might decide to edit your code directly on the remote machine (using `nano` or `vi`), or to edit it locally on your machine (with your preferred editor) and push the new versions to the server (using `scp` or `rsync` for instance). I suggest you start with the first method, and move to the second when your development environment is properly set up.

3 Part I of the lab (14 points): single machine experiment

3.1 Computing an integral

Taking inspiration from the `SparkPi` example code we want to compute the following integral using Spark:

$$\int_1^{10} \frac{1}{x} dx$$

- Mathematically, what is the value of the above integral?
- Write a Spark program that computes the above integral using the rectangle method (see `https://en.wikipedia.org/wiki/Rectangle_method` for more details).
- Evaluate the effect of the following parameters on the running time of your program:
  - the number of rectangles ;
  - the number of worker threads (`--master` option of `spark-submit`, see `https://spark.apache.org/docs/latest/submitting-applications.html#master-urls`) ;
  - the number of slices (see `https://spark.apache.org/docs/latest/programming-guide.html#parallelized-collections`) ;

When studying one parameter, you should keep the others constant to a default value. Your results should be averaged over a minimum of 10 experiments, and should only measure the execution of the main method. You should plot your results, and ideally report your error intervals.

3.2 Analyzing a (small) geolocated social dataset

The file `onlyBayLocsUsersWithFreq.txt` contains a list of 16240 Foursquare[1] users from the San Francisco area with the locations they have visited, and for each location the frequencies of their visits.

Each line (i.e. ending in a newline character) of the file corresponds to one user and is of the form[2] (using the BNF syntax[3]):

```
<user_Id> { <location_Id>","<frequency> }*
```

For instance, the third line of the file starts with

```
1devo 4c89630d9ef0224bd1b5517b,6 4827b8b1f964a520bb4f1fe3,1 3fd66200f964a5205ded1ee3,5
```

which means that user `1devo` has visited location `4c89630d9ef0224bd1b5517b` 6 times, location `4827b8b1f964a520bb4f1fe3` once, and location `3fd66200f964a5205ded1ee3` 5 times. (The location IDs can be used to directly query Foursquare by prefixing them with `http://foursquare.com/v/`. For instance `http://foursquare.com/v/4c89630d9ef0224bd1b5517b` tells us that the first location is a food truck located close to the Golden Gate Park in San Francisco.)

- Taking inspiration from the word count example presented in `http://spark.apache.org/examples.html`, write a Spark program that outputs the total number of visits of each location, sorted in reverse order of number of visits (i.e. the most popular location first).
  - **Notes:** By default the method `textFile(..)` of the `SparkContext` class splits a file on newlines (`\n`). So you will obtain a RDD of strings, with each string corresponding to one user. You will need to use the Scala method `split` to get down to location information.
- Write now a Spark program that outputs the total number of users who visited each location, also sorted in reverse order.
- Finally write a Spark program that outputs the average number of visits per user, per location, also in reverse order. (I.e, for each location `<location_Id>`, compute the average number of visits made by the users who visited this location.) There are several ways to do this, some more likely to be more efficient than others.
- Evaluate the same parameters as in the previous task on this last program.

3.3 Deliverable

- Source code, provided as a `git` or `svn` repository
- Report of maximum 1000 words (one collective report for the group): Your report should describe the overall working of your code, your results, and comment these results (why you think you obtained such results). Your report should also highlight any technical challenge or limitation you have encountered.

[1] `http://foursquare.com/`

[2] In a production environment, this data would probably be available in JSON format. But relatively straightforward to handle, this would add additional complexities to the lab, so we are using text instead.

[3] `https://en.wikipedia.org/wiki/Backus%E2%80%93Naur_form`

3.4 Marking scheme

- Using Scala: 2 points
- log(10): 3 points
  - Working example: 1 point
  - Experimental rigor: 1 point
  - Report (clarity, presentation): 1 point
- Foursquare Analysis: 9 points
  - Working example: 5 point
  - Experimental rigor: 2 point
  - Report (clarity, presentation): 2 point

4 Part II of the lab (3 points): cluster experiment

Repeat the above experiments using two VMs to run Spark instead of one. Use a simple `standalone` cluster manager.

- **Resource:**
  - `https://spark.apache.org/docs/latest/cluster-overview.html`

4.1 Deliverable

- Source code, provided as a `git` or `svn` repository
- Report of maximum 1000 words (one collective report for the group): As above, your report should describe the overall working of your code and set-up, your results, and comment these results (why you think you obtained such results). Your report should also highlight any technical challenge or limitation you have encountered.

5 Part III of the lab (optional, 3 points): cluster experiment with hdfs

Repeat the social network analysis experiment of Part II with an hdfs cluster.

- **Resources:**
  - `http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/`

5.1 Deliverable

- Source code, provided as a `git` or `svn` repository
- Report of maximum 1000 words (one collective report for the group): See above for what your report should contain.

6 Marking

You need to demonstrate your solution to obtain a mark. **A submitted solution that is not demonstrated will receive a grade of zero.** As soon as you have completed a part, you can ask for it to be marked. You should submit your code and report using moodle. The last lab session will be used for marking only. Be sure to submit your submissions the day before at the latest.