Romain SAVIDAN
INF 727
26/11/2014

SSHadoop

- What is SSHadoop?

SSHadoop is a distributed wordcount application inspired by a 2004 Google research paper and the MapReduce environment but using SSH to communicate between nodes.

- How does it work?

SSHadoop takes as input a set of words and return, for each word, the number of appearances in the set.
There are a few phases to get to the result:
1-SPLIT: we split the input file (one file per node)
2-MAP: each word is assigned the value 1 per appearance. After this step, two dictionnaries are produced:
'UMXmachines' dictionary: shows for each input split file, the corresponding node
'keysUMX' dictionary: shows for each key, the corresponding nodes
3-SHUFFLE: for each word, we group the values (the 1's)
4-REDUCE: for each word, we sum all the 1's to get the number of appearances of each word
SSHadoop works in two ways: local or distributed. On local mode, all calculations are made on the local machine. On the distributed one, we split the calculations among the machines on the local network in order to have improved performances.

Local mode: everything is run on the local machine, using parallel threads to take advantage of multi-threading (if supported by local machine). Each split of the input file is assigned to a thread (in this mode, 1 node = 1 thread). In this way, there is no use choosing a number of nodes that is greater than the number of cores of the local machine.
All steps are performed in parallel on local machine, given the computer can use multi-threading.

Distributed mode: the Map calculation is distributed on the local network. Indeed, only the Map phase is distributed. Yet, we see in practice that this is by far the longest one. The steps for the Map phase to be distributed are the following:
- the input file is split on the master (1 split per node)
- the split files are distributed on the local network along with a .jar file that is used to run the Map process. All files are sent using SSH protocol (using JSCH library). The .jar file is executed on the slave nodes (but its execution is launched from the master through JSCH) and an output map file is produced on the remote nodes.
Remote machines that correctly received the files are considered as working. Once files are sent and execution of the map phase is launched, the master will watch for the mapped files to appear on the remote machines and transfer them as soon as they are created. There is no network error handling: we make the assumption that that machines will not fail between the time they received the files and the time they finished mapping. The rest of the process is similar to the local mode where threads are used.

- How to run a job?

Some parameters are needed to run a SSHadoop job properly. Those parameters must be written directly in the java code, in the MapReduce class:
- input file: the absolute path to the input file (it can have any name)
- working directory: the directory were all intermediary files (the splitted files) are stored for the distributed mode
- the mode: 0 for local mode, 1 for distributed
- the number of nodes: please note that there must not be more keys in the input file than nodes (it is always the case in practice)
- the maximum waiting time: the time we will wait for the map process to run before aborting. Since there is no network error handling, the prevent us to staying into an infinite loop in case a failure occur during the map phase.

In addition to those parameters, some files/folders are needed:
- folder 'parameters': must be in the same directory as the input file. It must contains:
    - auth.txt: a file with login on first line, password on second line for SSH connection
    - listMachines.txt: a file with all machines on the local network (1 machine per line). Can be obtained in linux with the command 'arp -a' for example
    - map.jar: the jar file that contains the code for the distributed Map phase

The output will appear in the same folder as the input file.

- Performances:

SSHadoop has been tested on:
- the basic file (3 lines text file with 'deer bear ...'). It runs almost instantaneously on both local and distributed modes.
- the bible (6MB file). It runs almost instantaneously on both local and distributed modes.
- the basic file repeated up to 40MB (we repeat the pattern into a single file untill it reaches 40MB). It runs in 7 seconds on local mode and 18/23 seconds in distributed mode (4/23 nodes). We see that here the file is too small for the distributed mode to be interesting.
- the basic file repeated up to 100MB. Here only the distributed phase works, the local one shows a 'Java heap space' error. The distributed mode runs in 58/48/46 seconds (4/8/12 nodes).
It is not possible to test it on very large files due to network limitations.

- Future possible improvements:

- distribute all the phases
- use multi-threading on remote server (in the map.jar file in distributed mode)
- extends the application to other types of calculations, not only wordcount