SLR207 Project Report

Introduction

This report will give an overview of the project made as well as the different steps involved in creating the mapreduce from scratch and the results reached.

Implementation: Overview of The project

The Project is composed of 2 main classes:

-Master: Take as an input a text file and convert it to a byte array. Then splits the array into many parts (equal to the number of machines that will process the text), and sends them to each machine (Slave) via TCP socket in parallel (via the ExecutorService that will create many threads).

-Slave: The slave class will run in the machines that will process the text and do the MapReduce. They will be able to communicate with each other through sockets in order to divide correspondingly the words of each text chunk with the help of a built-in hashing function in Java; this helps limit communications between every slave and the master and rely less on the master to divide the tasks.

Phase 1 : Mapping phase

Once the master has sent the part of the text corresponding to each machine, the Slave will receive a byte array, convert it to String, and store all the words of the String in an array. The text chunks are sent in a parallel way to the slaves to make it more efficient and faster as it does not make sense sending them sequentially.

Phase 2 : Hashing Phase

The resulting array from the previous phase is then used to hash each word and associate it to a number which will represent the id of a machine. The words corresponding to every machine are stored in an ArrayList.

Phase 3.1 : Synchronization

The synchronization between the different machines is done with the help of the Master. Before starting with the shuffling phase which consists of sending each resulting arrayList from hashing phase via TCP socket to the other Slaves, the Master receives a message “finish”, sent by each machine that has finished the hashing. When the Master receives the message from all of the Slaves, it will send a “start” message to each of them, so that they can start with the shuffling.

Phase 3.2: Shuffling Phase

Once the hashing is done , the Slave will create a ServerSocket that will wait to receive the words associated with it by the other machines (via the method receiveWordArray). On the other Hand, each Slave will start sending the corresponding arrayList to the other slaves ONLY after receiving the “Start” message from the Master provided by the Synchronization phase elaborated in the previous paragraph.

Phase 4 : Reduce Phase

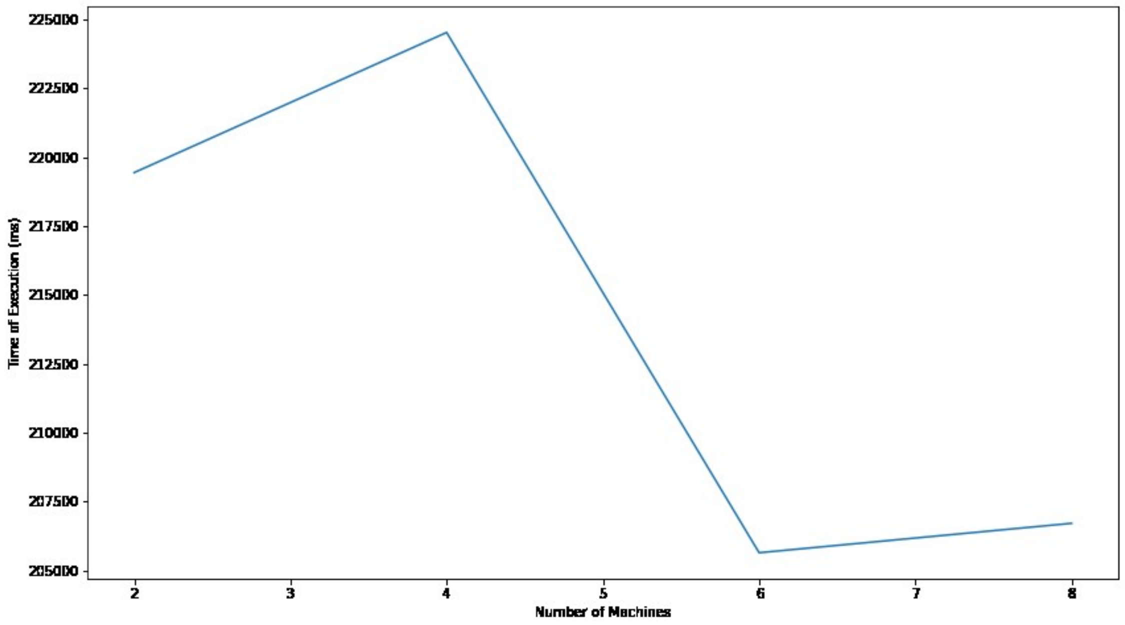
After each machine has received the corresponding ArrayList words from all other slaves (NB : the Reducing Phase starts Only after making sure that all arrays were received by all the slaves), it will call the processReceivedWordArray method, that will count the occurrence of each word and store it in a HashMap.

Phase 5 : Gathering Results

Finally, the Master will receive all the results communicated by the slaves via socket and will display them in the console.

Results

The code was tested with a text file of 17688 KB (sante_publique.txt). The following graph represents the time of execution of the code in function of the number of machines (slaves) used.



2 machines: 219437 ms
4 machines: 224523 ms
6 machines: 205640 ms
8 machines: 206708 ms

Note that the execution took more time than expected because the test was performed outside Telecom Paris, so connecting to the machines took a lot of time.