Big Data: Large Volume Data Analysis

First Team Work
Group 7

(master) Thomas Psaltikidis, 15173
Athanasios Farmakis, dai17102
George Zdoupas, 15336

Stylianos Gogas, dai17116

1. **The algorithm used to solve the problem**

Map-reduce was used in a distributed configuration for both cases, where first the map is completed and then the reduce is assigned to the required number of reducers (1,2,4,8) with the steps being map -> emit -> group by key -> reduce.

2. **Problems we encountered: how were they addressed?**

To everyone's surprise, several problems were encountered during the preparation of the work.

- problems with the cluster (a machine did not connect to the cluster and when it did, it created an hdfs directory instead of myhdfs)
    - solving by video conference with the teacher, editing slaves file
- problems with the implementation of the code (initially we tried to run it with java, but it seemed that the packages brought to us by Maven (org.apache ...) were missing and we could not solve it. We had to turn to python where things were very more easily, we did not even need an editor to write the code beyond the vi of virtual machines)
    - search the internet for what might have gone wrong with Maven, tried some solutions suggested on sites like stackoverflow and got nowhere
    - Similarly, searches were made for code syntax in python, where no problems arose other than some minor syntax errors, which were resolved fairly easily. The code was written on the virtual machine with the vi editor.
- problems with the structure of the whole system (where do we save the code? where do we save the data we will map-reduce? how do we run map-reduce in a cluster and not locally?)
    - Internet search where it did not help much
    - very good reading of the commands and commands_2020 files in order to understand 100% of the system structure, and to understand the steps that need to be done to run the map-reduce in a cluster
    - finding the command running in a cluster map-reduce with python implementation was a hassle, as the majority of searches implemented code in java
- again problems with the cluster (the same machine as the first bullet because for some reason it did not exist, as if destroyed, as a master I had a problem with passwordless ssh and did not

- had become a new machine again with the result that it displays
  “@@
  @ WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED! @
  @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@.
  IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
  Someone could be eavesdropping on you right now (man-in-the-middle attack)!
  It is also possible that the RSA host key ..... ”. Solution by searching the Internet

Somewhere here to note that the experience of team members with Linux systems is almost zero (or rather, it was).

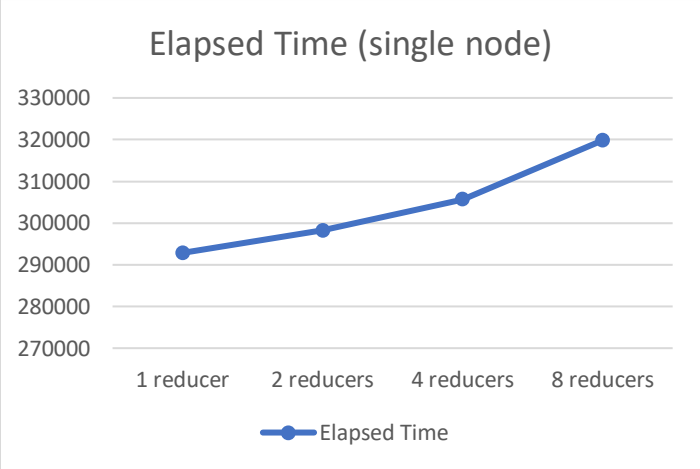
- problems with yarn and map-reduce configuration
  - the configuration seemed to have an issue so the reduce does not run in a cluster (while the map was completed normally) and had to be tampered with by the master. The problem he had was "Container is running beyond memory limits". We had to configure the maximum memory allocations and fill in the following lines in mapred-site.xml
  - ```
    <property>
        <name> mapreduce.map.memory.mb </name>
        <value> 6144 </value>
    </property>
    <property>
        <name> mapreduce.reduce.memory.mb </name>
        <value> 6144 </value>
    </property>
    ```
    then it ran ok. Afterwards we were informed by a colleague who asked in the course that had the respective topic, that we could approach the problem differently, but after we had completed all the runs for all possible provisions we left it as it is.
    ```

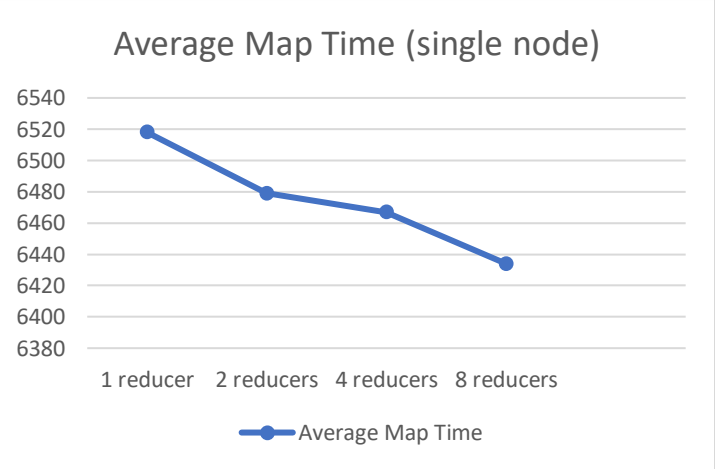
3. Execution of the code and calculation of times

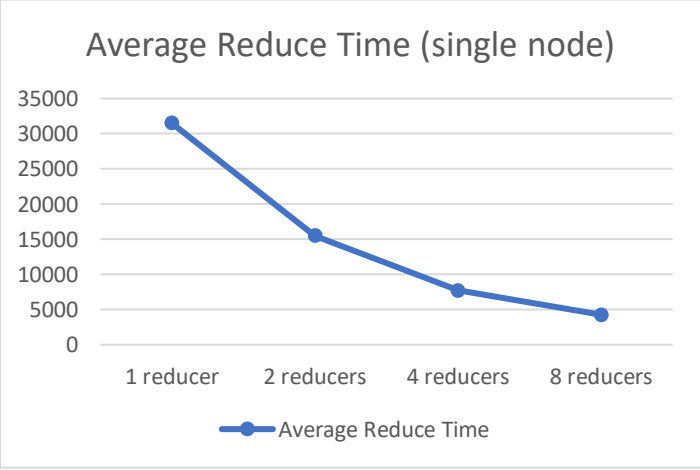
a. Let's first look at the execution for 1 node in distributed configuration. The node used was the master. If you want to see in detail the execution times you can see the xlsx file that exists in the submitted file (with max: ..., min: ... the longest and the shortest time are marked respectively, which is not taken into account in the calculation of the average). The times given in the tables and diagrams are in ms.

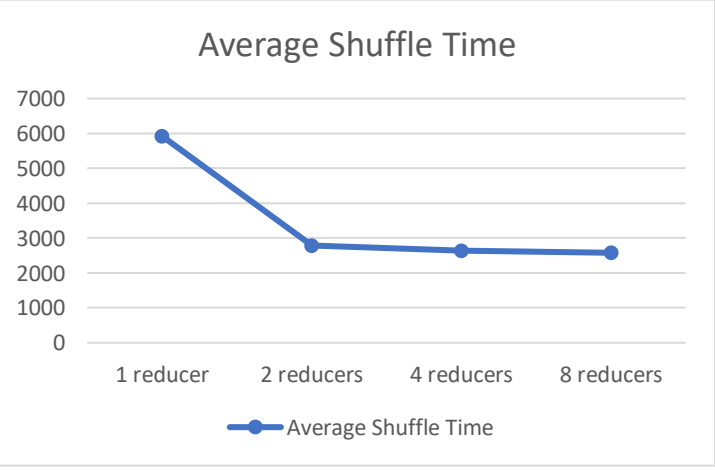
Run on a node					
No. of reducers	**Elapsed Time**	**Average Map Time**	**Average Reduce Time**	**Average Shuffle Time**	**Average Merge Time**
1	292849	6518	31486	5924	2666
2	298205	6479	15514	2782	3258
4	305681	6467	7734	2633	1867
8	319832	6434	4271	2579	984

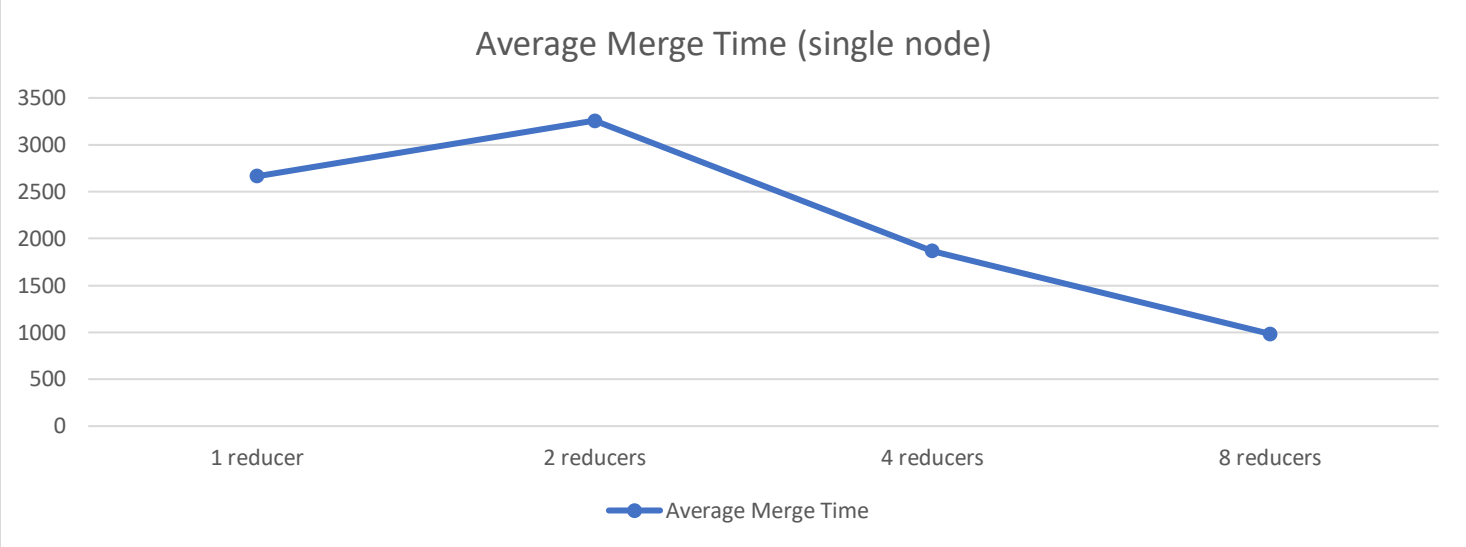
Diagrams for single node executions







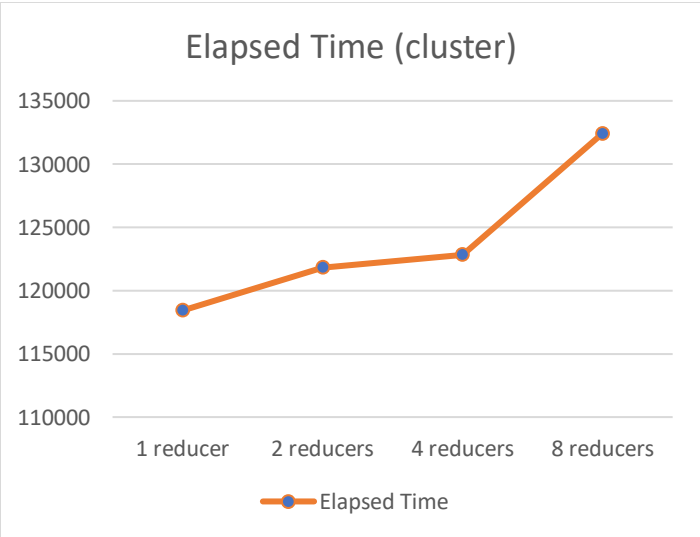


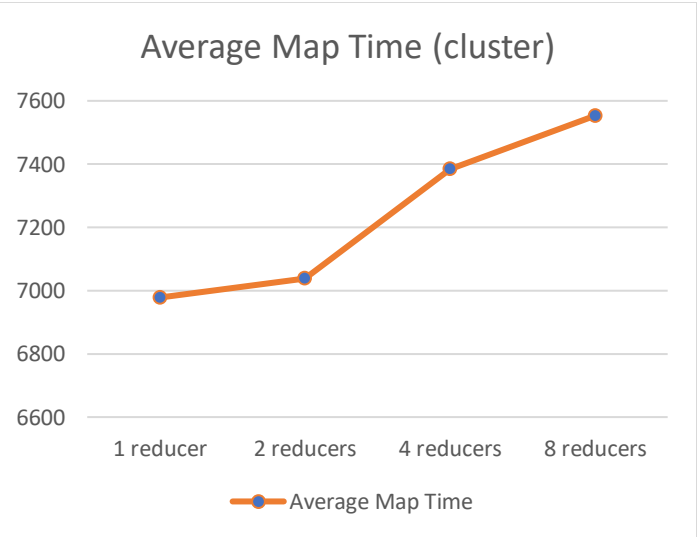


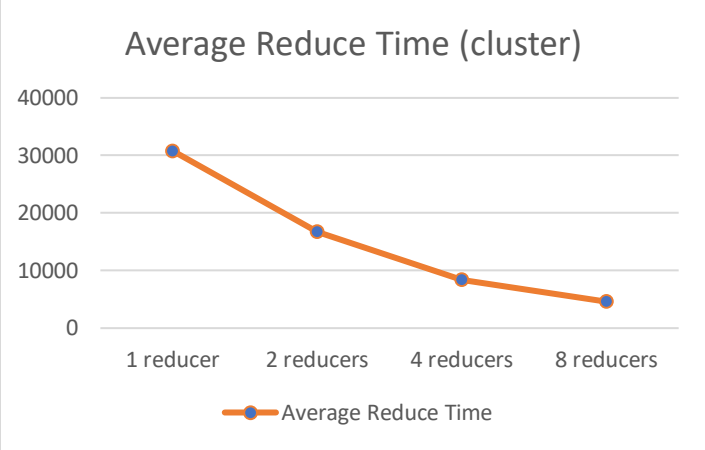
b. Then we will see the execution in the cluster. The cluster consists of 4 nodes, a master and three slaves. Similarly the times are detailed in the xlsx file.

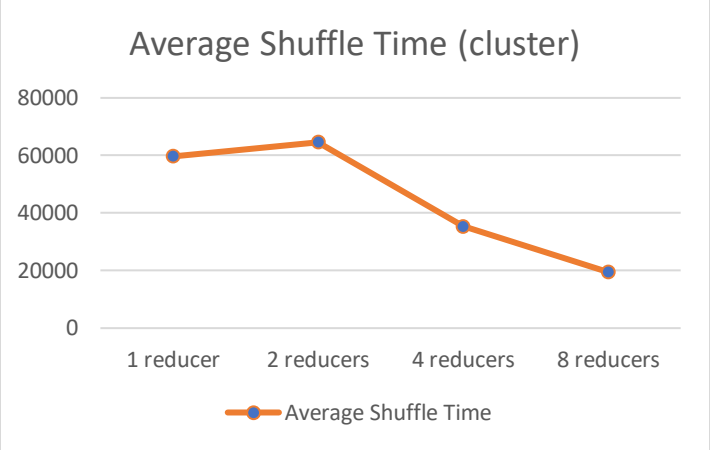
Execution in cluster					
No. of reducers	**Elapsed Time**	**Average Map Time**	**Average Reduce Time**	**Average Shuffle Time**	**Average Merge Time**
1	118425	6978	30796	59766	671
2	121832	7039	16724	64566	438
4	122825	7385	8400	35396	1735
8	132432	7553	4598	19480	984

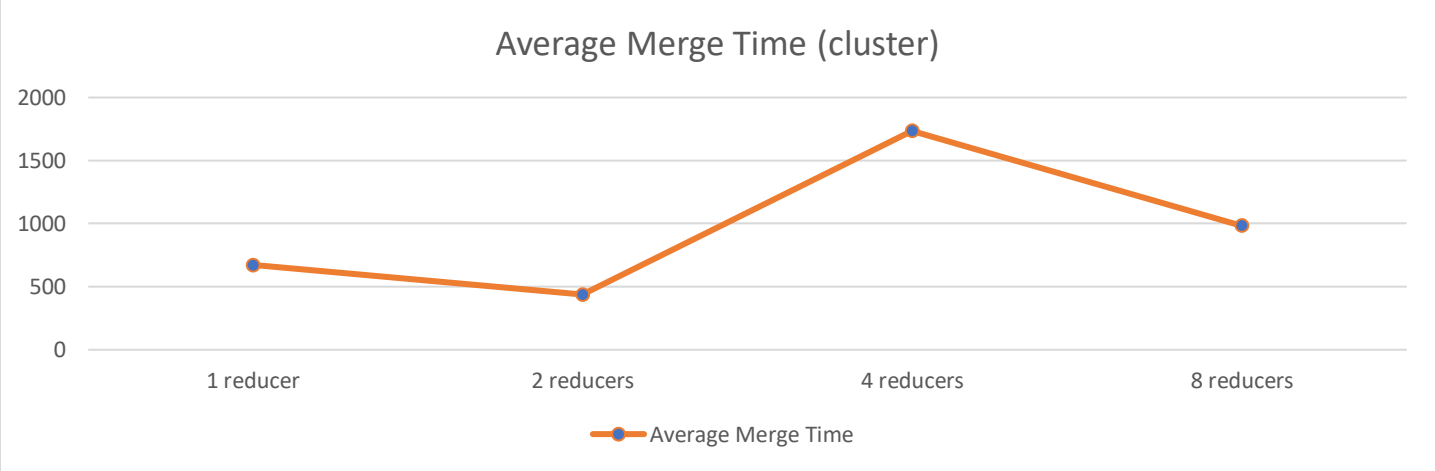
Diagrams for cluster executions







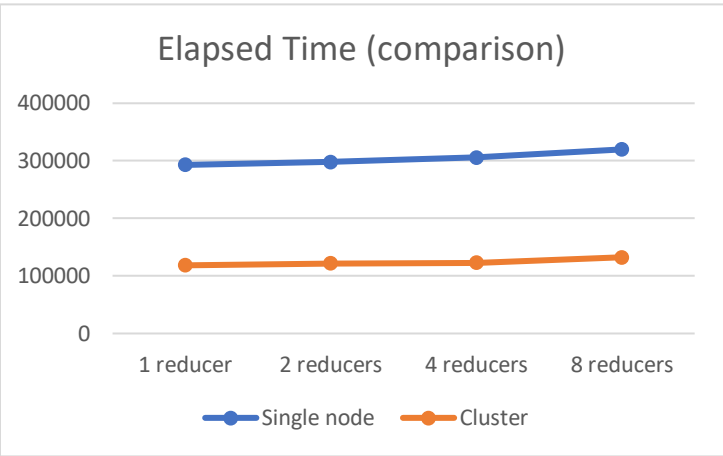


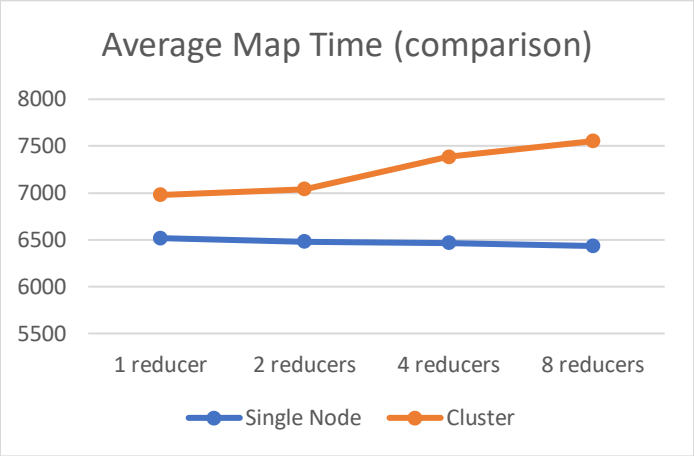


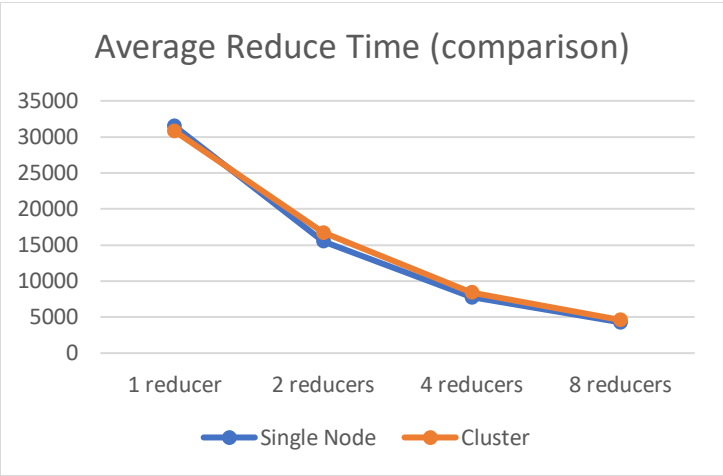
c. Let's see comparatively the execution times, for single node and for cluster

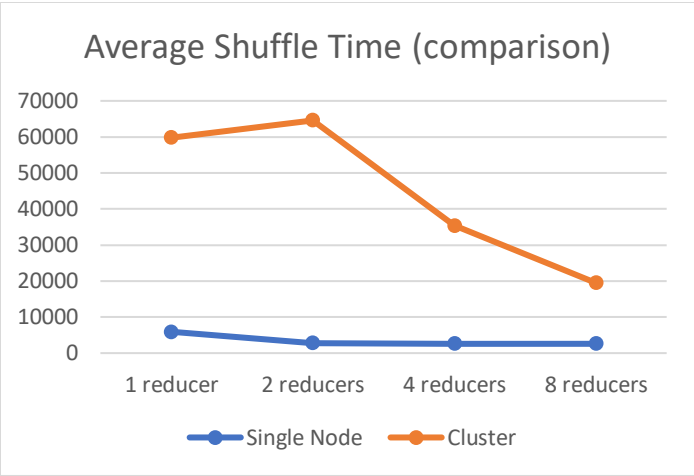
Execution in cluster						
No. of reducers	**Mode**	**Elapsed Time**	**Average Map Time**	**Average Reduce Time**	**Average Shuffle Time**	**Average Merge Time**
1 **1**	**Single** **Cluster**	292849 118425	6518 6978	31486 30796	5924 59766	2666 671
2 **2**	**Single** **Cluster**	298205 121832	6479 7039	15514 16724	2782 64566	3258 438
4 **4**	**Single** **Cluster**	305681 122825	6467 7385	7734 8400	2633 35396	1867 1735
8 **8**	**Single** **Cluster**	319832 132432	6434 7553	4271 4598	2579 19480	984 984

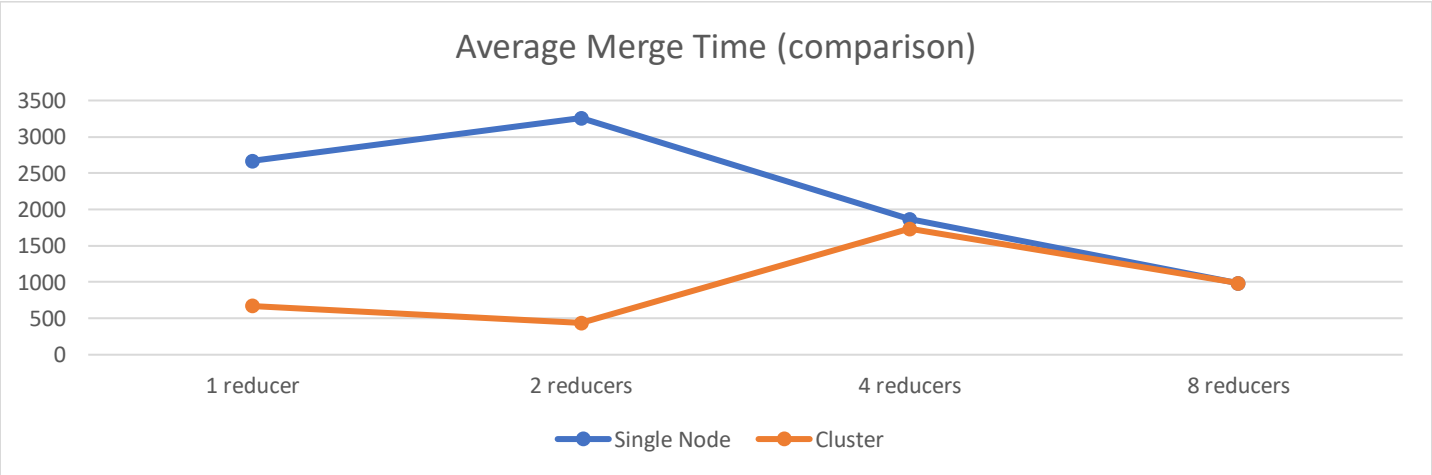
Modes time comparison charts











4. **Remarks**
    a. **Elapsed time:**The total time required to run a Map Reduce process. We notice that the time is dramatically reduced when we execute the process in the cluster with four slaves, than in the single node of the master. Determining factor in this fact, is the peculiarity that in the cluster, mapping and reducing processes can "run" at the same time, while in the single node the mapping process must end for the reducer to start. At the same time, there is a tendency for an infinitesimal increase in duration depending on the number of reducers, but we do not have enough data to draw a safe conclusion.
    b. **Average Map Time:**The time consumed by the mapping process, ie the production of key-value pairs for each line of input data. It is unaffected by any factor (number of nodes, reducers, etc.). That is, his times in all the execution scenarios we ran have very little difference between them.
    c. **Average Reduce Time:**The time it takes the reducer to make results for pairs of the same keys. There are no major differences between the times in the execution of the single-node and the cluster as the difference of the times has to do with the number of reducers. That is, when we double the number of reducers, the time required for the total reduction will be doubled.
    d. **Average Shuffle Time:**Shuffle consists of two parts: the "download" of the mapping outputs and their conversion into inputs for the reducer. The time it takes to complete is many times longer in the cluster than in the single node. Specifically, it is observed that in the cluster the times are multiple in relation to the single-node. It was also observed that during the execution there was quite a visible delay at this time and in relation to the rest, we believe due to some lag of the network or the overloaded use of okeanos servers.
    e. **Average Merge Time:**It is the process of grouping the reducer outputs. Unlike Shuffling, Merging consumes less time in the cluster than in the single node, ie the single node is slower than the cluster.

5. **Results and comparison of results**
    a. The results of the executions were compared with the diff command on the machine and with the help of the -s argument which gives an answer only if the files being compared are identical. In the end, the outputs are identical. Indicatively, some

screenshots are listed (in the directory local are stored the results that came out of the single node executions and in the cluster are corresponding to the cluster).

b. These outputs can be found at the following link:

```
user@snf-11954:~/project1$ diff -s local/reducer1-take1/part-00000 cluster/reducer1-take1/par
t-00000
Files local/reducer1-take1/part-00000 and cluster/reducer1-take1/part-00000 are identical
user@snf-11954:~/project1$ diff -s local/reducer2-take1/part-00000 cluster/reducer2-take1/par
t-00000
Files local/reducer2-take1/part-00000 and cluster/reducer2-take1/part-00000 are identical
user@snf-11954:~/project1$ diff -s local/reducer4-take1/part-00000 cluster/reducer4-take1/par
t-00000
Files local/reducer4-take1/part-00000 and cluster/reducer4-take1/part-00000 are identical
user@snf-11954:~/project1$ diff -s local/reducer8-take1/part-00000 cluster/reducer8-take1/par
t-00000
Files local/reducer8-take1/part-00000 and cluster/reducer8-take1/part-00000 are identical
```