Khushboo Tekchandani

CSE587 Homework 1

Solution:

The problem required us to determine the volatility of stocks in NASDAQ using the mapreduce framework and find top ten stocks having the maximum and minimum volatilities.

To implement this, I used three map-reduce modules.

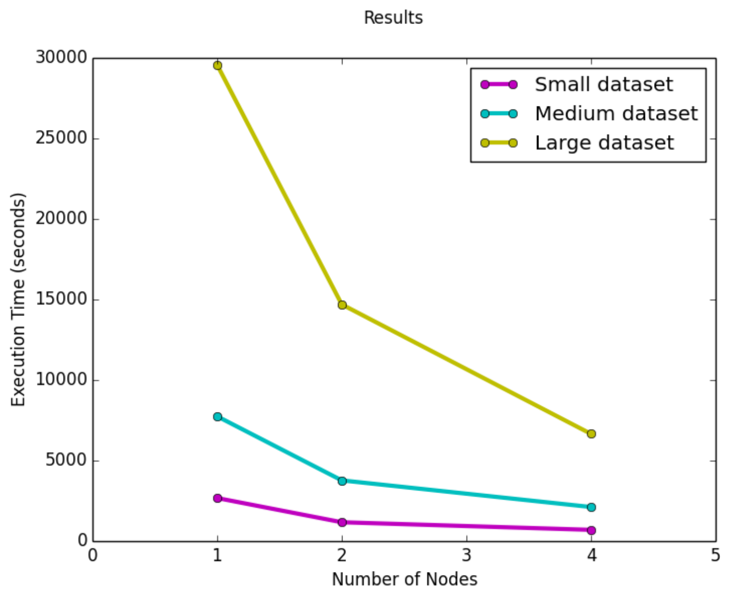
1. In the first module, the mapper takes the input data, reads the filename and associates it with the data of the stock stored in the file. The output of this module is the key value pair that has the **company name and date (MM/YYYY)** as the key and **closing adjusted values and date (DD)** as the value.
   The reducer manipulates the values received and calculates monthly rate of return. The output of the reducer is **company name and date (MM/YYYY)** as key and the corresponding **monthly rate of return** as the value.
2. The mapper of the second module receives the output produced by the first reducer. It discards the date associated in the key in the input. It gives the **company name** as key and the **monthly rates of return** associated with each company as output.
   The reducer of this module, processes the values received and calculates the average monthly rate of return for each company. It further calculates the volatility for each company. The output of this reducer is the **company name** as key and **volatility** as value.
3. The third and final module is used to get the desired output. The mapper here, sends a key "Final Value" and keys containing specific **company** and its **volatility** as the output.
   The reducer finally sorts all these volatilities and writes the top ten maximum and top ten minimum volatilities and their companies.

Evaluation:

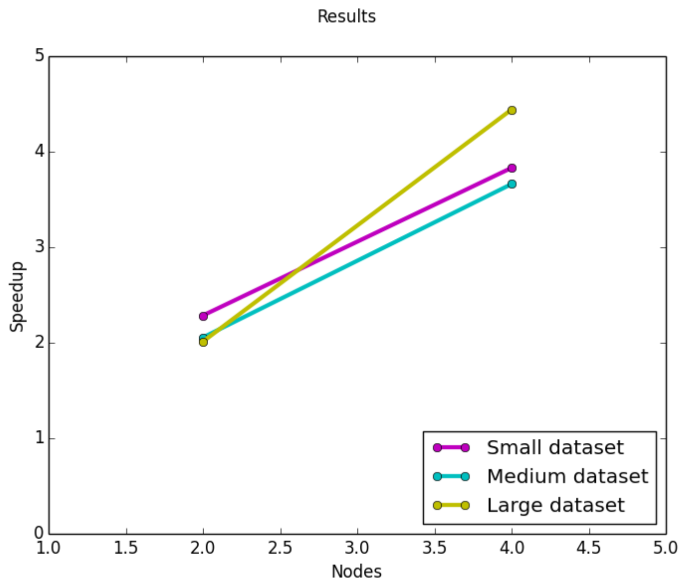
Using this implementation, I executed my code on CCR to understand the scalability and performance of my solution. The experiment was done using different data sets and different number of nodes. Following data was obtained by carrying out these experiments:

	Execution Time in Seconds		
Problem Size	**1 node (12 cores)**	**2 nodes (24 cores)**	**4 nodes (48 cores)**
Small	2674	1170	698
Medium	7748	3770	2116
Large	29569	14696	6657

Table 1: Execution time of the program on different number of nodes, using multiple data sets.



Graph1: Execution time for different problem sizes on different number of compute nodes



Graph2: Speedup of the result of increasing the compute nodes at different data sets

Comment on performance:

It is evident from the data received from the experiments that the execution time of the program almost halves by increasing the compute nodes. However, in this scenario, the hardware used also doubles. So, the performance is a trade of with the economic viability of the solution.

The speedup graph suggests that the gain in time that is decrease in execution time is much higher for the large data set when the number of compute nodes are increased. So, it can be concluded that when the problem size is greater, it is more beneficial to increase the hardware as compared to a smaller problem size.