Report for Assignment 1

Stock Volatility Analysis Using Hadoop

Harsh Harwani

1. Method and Implementation

- I used 3 mapreduce stages in order to find the top 10 stocks with minimum volatility and top 10 stocks with maximum volatility.
- In the first stage the filename which is the stock name and date fields are extracted from the file that is being processed, the stock name/month/year is set as the key and the particular line of data is stored as a value.The reducer receives the input from the mapper with the values for every stock name and every month combined.The reducer then extracts the first and the last date for every month of and calculates the Xi value and sets it back with the key.The output of the first stage is (stock name/month/year,Xi value)
- In the second stage the mapper receives the input from the first reducer,the mapper receives the key as stock name/month/year as key and Xi as value,the mapper removes the month and year from the stock name and puts stock name as the key which will result in combining of all the Xi values of a particular company for all the months.The reducer receives the input from the mapper with the key as stock name and all Xi values for that company.The reducer goes through all the Xi values and computes the stock volatility according to the formula.The output of the second reducer is(stock name,volatility). we set a counter in job 2 ‘s configuration and increment it every time when we calculate volatility of a stock and write it in the intermediate file.
- In the third stage the mapper receives input as (stock name,volatility) .As we need to display the top 10 stocks with maximum volatility and top 10 stocks with minimum volatility I sorted the data using map-reduce internal sorting the mapper always writes output sorted by the key,so the in this stage the mapper all swaps the key and value which is received in the input
- the output from the mapper will be (volatility,stock name).The reducer will receive the input in a sorted order so the top 10 stocks are the stocks with minimum volatility.we use the counter that we set during job2,which tells us the total number of stocks,using that the counter we display the top 10 stocks with maximum volatility as those would be the last 10 stocks in the intermediate file.

Here’s an example that how the implementation will work during different stages:

First stage:

The input to the mapper will be:	
Key	**Value**
AAIT/12/2014	2014-12-31,32.95,32.95,32.95,32.95,000,32.95
AAIT/12/2014	2014-12-30,32.86,32.95,32.86,32.95,1200,32.95
AAIT/12/2014	2014-12-29,33.18,33.22,33.18,33.20,1700,33.20
AAIT/12/2014	2014-12-01,34.20,34.20,34.20,34.20,100,33.84

The output from the mapper will be:	
Key	**Value**
AAIT/12/2014	{ (2014-12-31,32.95,32.95,32.95,32.95,000,32.95),(2014-12-30,32.86,32.95,32.86,32.95,1200,32.95).....(2014-12-01,34.20,34.20,34.20,34.20,100,33.84) }

The input to the reducer will be :	
Key	**Value**
AAIT/12/2014	{(2014-12-31,32.95,32.95,32.95,32.95,000,32.95),(2014-12-30,32.86,32.95,32.86,32.95,1200,32.95).....(2014-12-01,34.20,34.20,34.20,34.20,100,33.84) }

The output from the reducer will be:	
Key	**Value**
AAIT/12/2014	Xi value
AAIT/11/2014	Xi value
AAIT/10/2014	Xi value
AAIT/01/2014	Xi value

Second stage:

The input to the mapper will be:	
Key	**Value**
AAIT/12/2014	Xi value
AAIT/11/2014	Xi value
AAIT/10/2014	Xi value
AAIT/01/2014	Xi value

The output from the mapper will be:	
Key	**Value**
AAIT	{Xi value,Xi value,Xi value,Xi value,Xi value}
AAL	{Xi value,Xi value,Xi value,Xi value,Xi value}

The input to the reducer will be :	
Key	**Value**
AAIT	{Xi value,Xi value,Xi value,Xi value,Xi value}
AAL	{Xi value,Xi value,Xi value,Xi value,Xi value}

The output from the reducer will be:	
Key	**Value**
AAIT	Xi value
AAL	Xi value
AAME	Xi value

Third stage:

The input to the mapper will be:	
Key	**Value**
AAIT	volatility
AAL	volatility
AAME	volatility

The output from the mapper will be:	
Key	**Value**
volatility	AAIT
volatility	AAL
volatility	AAME

The input to the reducer will be:	
Key	**Value**
volatility	AAIT
volatility	AAL
volatility	AAME

The output from the reducer will be:	
Key	**Value**
AAIT	volatility
AAL	volatility
AAME	volatility

The output at the last stage will be sorted according to the volatility.The third reduce map stage is only used for sorting.By default the map phase sorts the output according to the key.So we sort the output according to the volatility and the reducer displays the top 10 stocks with minimum and maximum volatility.

Alternative Implementation:
As the third MapReduce phase is only used for sorting we can avoid that stage and sort the output in the cleanup method of the second reducer.I tried the alternate implementation as well but I could only test the implementation on small data for 48 cores and there was a marginal improvement in performance, but it is possible that the performance might have improved considerably in case of large dataset.The time taken by small data set on 48 cores was about 8.75 minutes and in case of implementation where three stages were used the time was 9.1 minutes.So we can see there was a improvement in time,but I was not able to check the implementation on medium and large datasets,so below I am showing the results of implemetation using 3 map-reduce stages.

2. Experiment and Discussion

The experiments are conducted on CCR in UB. Each of the compute node for the experiment has 12 processors, 48GB memory and is connected with Infiniband Network. Each process is an Intel Xeon CPU with 2.4GHz and 6 cores.
The program was first developed the eclipse IDE in one single machine, then the jar file was exported and delivered to the CCR and executed.
The below results show the parallel execution of small,medium and large data sets on each of 12,24 and 48 cores in CCR.

Parallel Execution:

Problem Size	Execution Time(mins): 1 node (12 cores)	Execution Time(mins): 2 node (24 cores)	Execution Time(mins): 4 node (48 cores)
Small	40.73333333	17.9	9.016666667
Medium	120.8	52.18333333	25.61666667
Large	408.7666667	172.4166667	83.2333

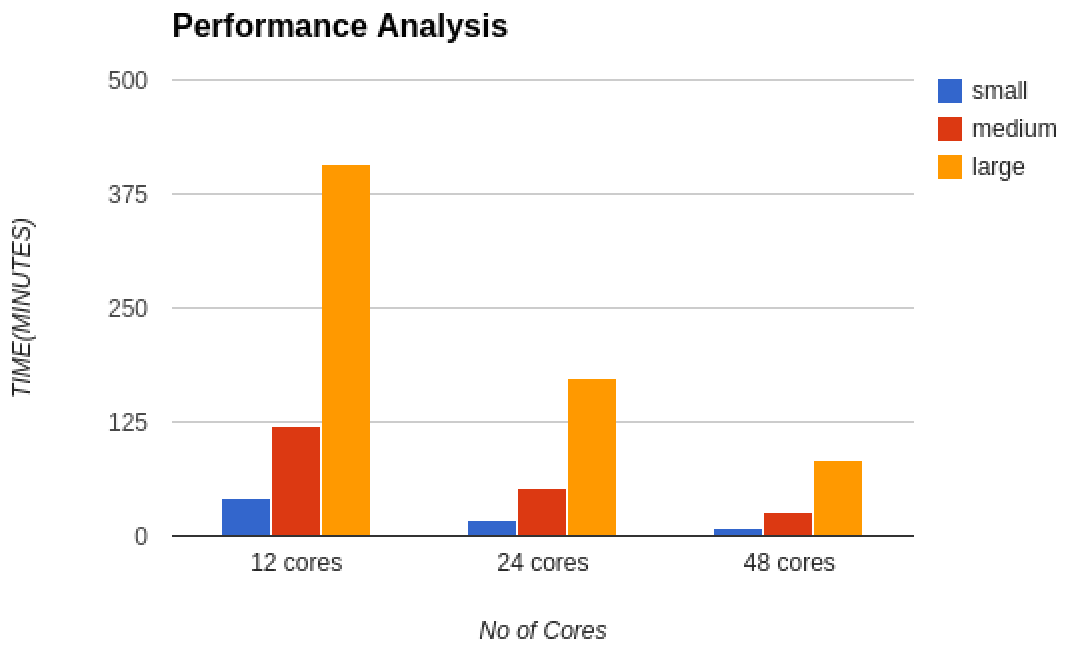

Figure 1: Performance analysis of different datasets in case of different number of cores:

The above graph shows the time taken by small,medium and large datasets on machines with different number of cores. As we can see that as the number of cores increases the time taken for the job to complete reduces.The performance is scalable as the time taken on 48 cores is almost 1/4th of the time taken on 12 cores.I ran the experiments twice on different times of the day to make sure other factors such as load on the machine do not come into account.So we can conclude that the implementation is scalable and will perform as expected as the number of cores and data increases.

Serial Execution:

No of nodes: (1 core)	Small	Medium	Large
1	45.75	127.3666668	415.3333333

Figure 2: SpeedUp graph:

Problem Size	Speed-Up: 1 node (12 cores)	Speed-Up): 2 node (24 cores)	Speed-Up: 4 node (48 cores)
Small	1.097122302	2.497725159	5.14719863
Medium	0.9514524957	2.374029204	4.949481935
Large	1.015484841	2.306552952	4.90290554

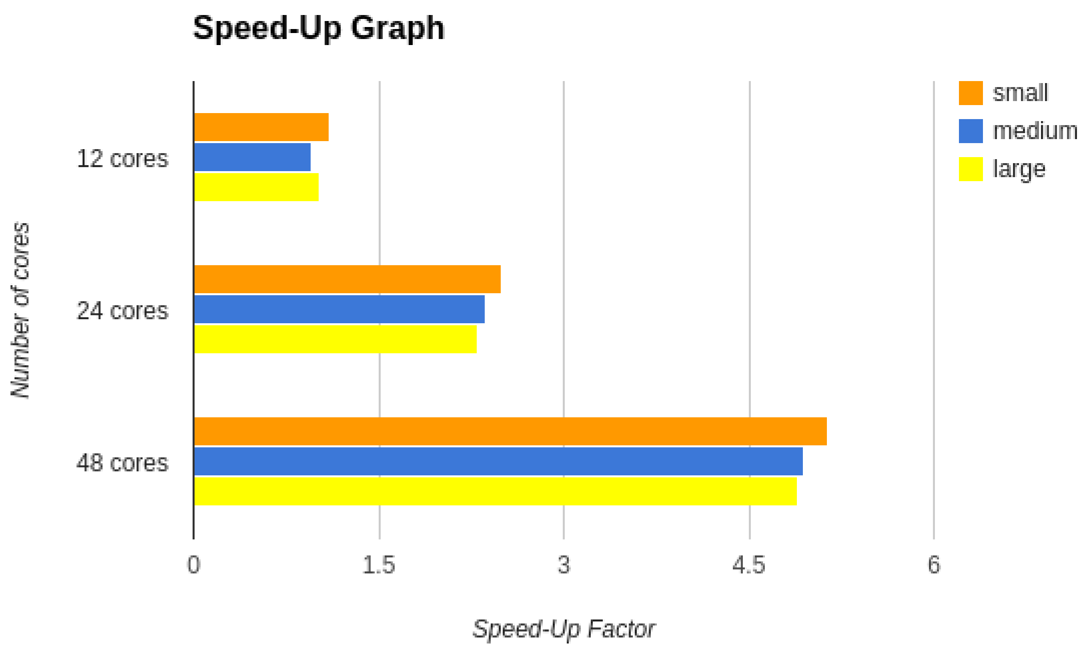

Figure 2: Speed-Up Analysis of different datasets in case of different number of cores:

As we can see when the number of cores is small the speedup factor is almost 1 which means the time taken in case of serial and parallel execution is the same,but as we increase the number of cores the speedup factor rises and in case of 4 nodes, the speedup factor is almost 5 which means the execution runs 5 times faster as compared to the serial execution.

So we conclude that by using Mapreduce efficiently,we can take advantage of parallel execution and can process very large data sets quickly and accurately.