

**CEG-7380**

**Cloud Computing**

**Spring 2016**

**Project #1**

**Report File**

**DHRUVKUMAR NAVINCHANDRA PATEL**

**U00791652**

**Simple Moving Average**

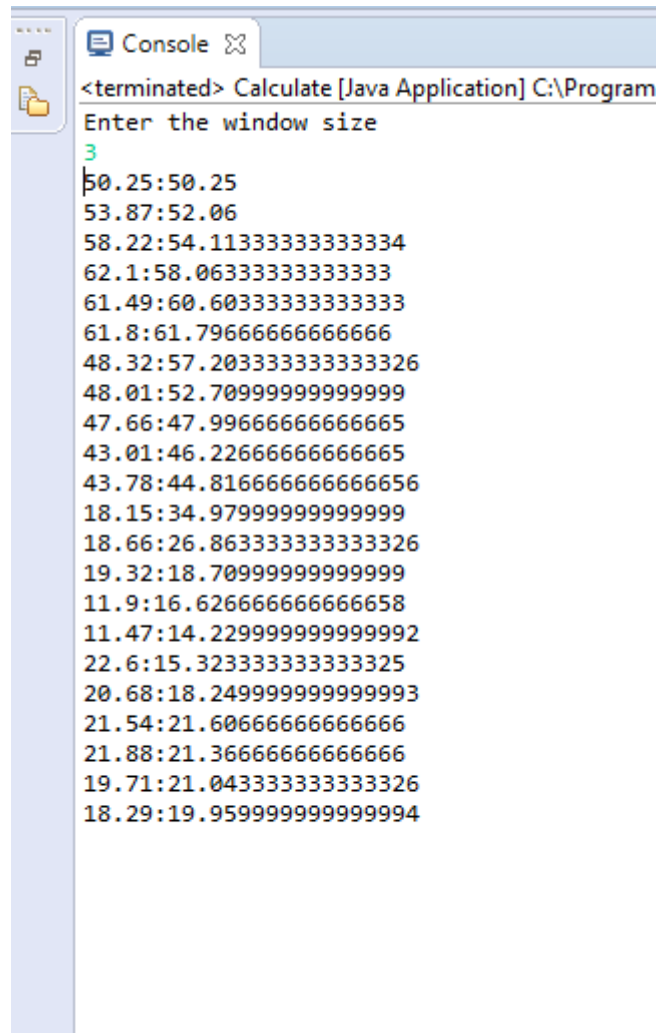
Following are the different steps which I followed during development of this project

- 1) **Analysis:** First I am calculating Simple Moving Average using Single Host Java Program and then implement that on Map-Reduce Hadoop framework.
  - 1) In this project implementing Simple Moving Average that is arithmetic mean of the values in particular time period. So, here I am processing company's Stock data.
  - 2) Here in input file I am taking Company symbol, Date and Closing Price and according to that calculating Simple Moving Average for particular Date Period according to Window size.
  - 3) I implemented whole program in which one can enter Window size dynamically and generate output according to that window size.
  - 4) For example, for first day simple moving average is the closing price itself and then for second =  $\text{first} + \text{second} / 2$  and then if window size is 3 then calculate according to window size.
- 2) **Implementation:** There are two implementation Single host and Hadoop Map-Reduce Environment.
  - 1) **Single Host in Java :**
    - i) For single host I developed simple java program which take following data as an input.

```
Calculate.java input.txt
1 50.25
2 53.87
3 58.22
4 62.10
5 61.49
6 61.80
7 48.32
8 48.01
9 47.66
10 43.01
11 43.78
12 18.15
13 18.66
14 19.32
15 11.90
16 11.47
17 22.60
18 20.68
19 21.54
20 21.88
21 19.71
22 18.29
```

- ii) First enter the window size from console. And then I am reading above inputs from file and store into array list. Then I am sending window size and each entry from array list to one return type method called returnmovingaverage.
- iii) For first one the simple moving average is closing price itself and for second one  $\text{first} + \text{second} / 2$  and when the number increase I am storing data in one array list according to window size.
- iv) Following is the Output of above input

Window Size = 3;



```
<terminated> Calculate [Java Application] C:\Program
Enter the window size
3
50.25:50.25
53.87:52.06
58.22:54.11333333333334
62.1:58.06333333333333
61.49:60.60333333333333
61.8:61.79666666666666
48.32:57.20333333333326
48.01:52.70999999999999
47.66:47.99666666666665
43.01:46.22666666666665
43.78:44.81666666666656
18.15:34.97999999999999
18.66:26.86333333333326
19.32:18.70999999999999
11.9:16.62666666666658
11.47:14.22999999999992
22.6:15.32333333333325
20.68:18.24999999999993
21.54:21.60666666666666
21.88:21.36666666666666
19.71:21.04333333333326
18.29:19.95999999999994
```

Here output is like

Closing price: simple moving average

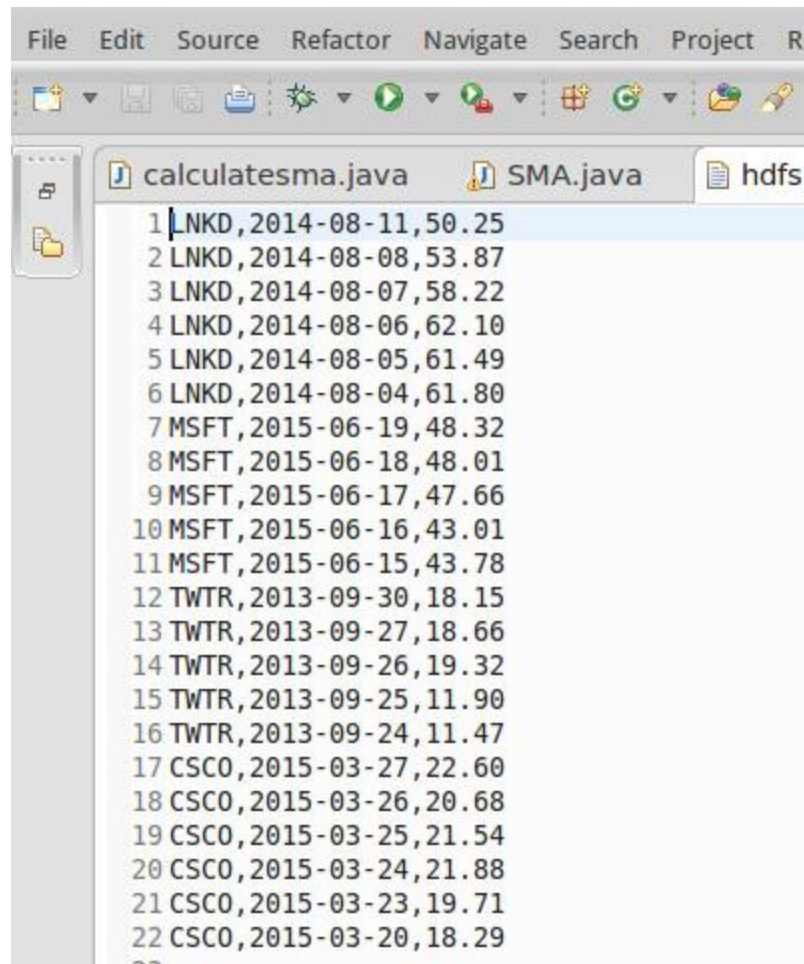
For 58.22 the SMA =  $50.25 + 53.87 + 58.22 / 3$

For 62.1 the SMA =  $53.87 + 58.22 + 62.1 / 3$

And so on.

## 2) Hadoop MapReduce implementation :

- i) In Hadoop MapReduce I am taking Following Data as an input. I developed this program in our virtual Cloud which has a version of Hadoop 1.2.1.



```
1 LNKD, 2014-08-11, 50.25
2 LNKD, 2014-08-08, 53.87
3 LNKD, 2014-08-07, 58.22
4 LNKD, 2014-08-06, 62.10
5 LNKD, 2014-08-05, 61.49
6 LNKD, 2014-08-04, 61.80
7 MSFT, 2015-06-19, 48.32
8 MSFT, 2015-06-18, 48.01
9 MSFT, 2015-06-17, 47.66
10 MSFT, 2015-06-16, 43.01
11 MSFT, 2015-06-15, 43.78
12 TWTR, 2013-09-30, 18.15
13 TWTR, 2013-09-27, 18.66
14 TWTR, 2013-09-26, 19.32
15 TWTR, 2013-09-25, 11.90
16 TWTR, 2013-09-24, 11.47
17 CSC0, 2015-03-27, 22.60
18 CSC0, 2015-03-26, 20.68
19 CSC0, 2015-03-25, 21.54
20 CSC0, 2015-03-24, 21.88
21 CSC0, 2015-03-23, 19.71
22 CSC0, 2015-03-20, 18.29
```

- ii) **Mapper Implementation:** In Mapper I am taking data from above inputs and make Company Symbol as Key and Remaining Date, Closing price as value.
  - a) In mapper Text and Dateclosingprice as type of Output key and output value.
  - b) Here I converted Date from String to Long and then I set Date and closing price into intermediate class which is Dateclosingprice.
  - c) So Output from mapper is Company symbol as a key and object of Dateclosingprice as value.
- iii) **Dateclosingprice implementation:** This class is intermediate class which implements WritableComparable interface and has some inbuilt implemented methods which is used for Sorting Company Data according To Date. This class executed after mapper and before reducer.

- a) Here my assumption is in this class there is one compareTo method which is for sorting. This is automatically called before reducer in Hadoop 2.6.3 version. But, in our virtual cloud has a Hadoop version 1.2.1.
- b) So, in that case I store data in array list which is coming from the Dateclosingprice class and I called Collection.sort() method to enable compareTo.
- c) This class is basically for Serialization and Deserialization.

iv) **Reducer implementation:** In reducer I am getting sorted Date and according to closing price for each and every key. Reducer automatically sort key in Alphabetical order. Here I am sending window size and closing price into calculatesma class which return simple moving average for that time period.

- a) At last, I am make company symbol as key format and Date, simple moving average as value format and write into output file.
- b) In Hadoop 1.2.1 version there is no TextOutputFormat.SEPERATOR so in my output there is a space between key and value.

#### Output from Virtual Cloud:

```

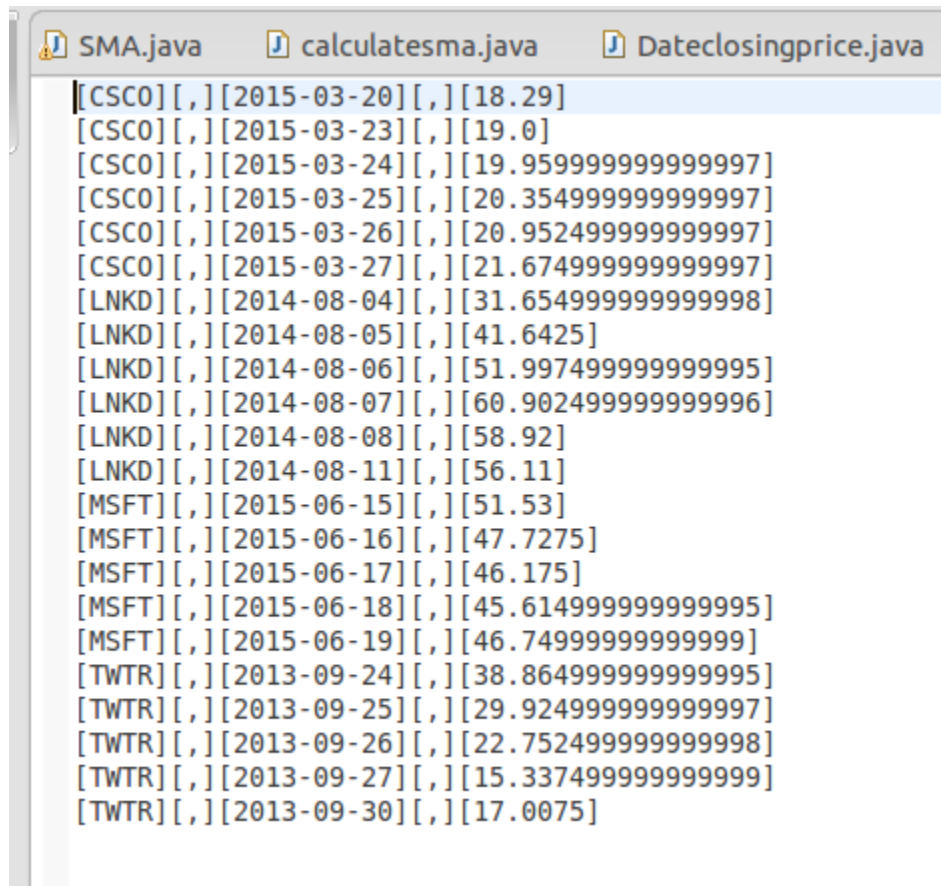
1 [CSC0] [2015-03-20][,][18.29]
2 [CSC0] [2015-03-23][,][19.0]
3 [CSC0] [2015-03-24][,][19.959999999999997]
4 [CSC0] [2015-03-25][,][21.043333333333333]
5 [CSC0] [2015-03-26][,][21.366666666666664]
6 [CSC0] [2015-03-27][,][21.606666666666666]
7 [LNKD] [2014-08-04][,][35.026666666666664]
8 [LNKD] [2014-08-05][,][48.629999999999995]
9 [LNKD] [2014-08-06][,][61.796666666666666]
10 [LNKD] [2014-08-07][,][60.60333333333333]
11 [LNKD] [2014-08-08][,][58.06333333333333]
12 [LNKD] [2014-08-11][,][54.113333333333334]
13 [MSFT] [2015-06-15][,][49.300000000000004]
14 [MSFT] [2015-06-16][,][45.68]
15 [MSFT] [2015-06-17][,][44.816666666666666]
16 [MSFT] [2015-06-18][,][46.226666666666666]
17 [MSFT] [2015-06-19][,][47.996666666666666]
18 [TWTR] [2013-09-24][,][35.93333333333333]
19 [TWTR] [2013-09-25][,][23.896666666666666]
20 [TWTR] [2013-09-26][,][14.229999999999995]
21 [TWTR] [2013-09-27][,][16.626666666666666]
22 [TWTR] [2013-09-30][,][18.709999999999994]

```

Output is following format:

[COMPANYSYMBOL] [Date][,][Simplemovingaverage]

- c) In Hadoop 2.6.0 version there is `TextOutputFormat.SEPERATOR` so I removed space between key and value and I am getting output exactly what you want.



```
[CSCO][,][2015-03-20][,][18.29]
[CSCO][,][2015-03-23][,][19.0]
[CSCO][,][2015-03-24][,][19.959999999999997]
[CSCO][,][2015-03-25][,][20.354999999999997]
[CSCO][,][2015-03-26][,][20.952499999999997]
[CSCO][,][2015-03-27][,][21.674999999999997]
[LNKD][,][2014-08-04][,][31.654999999999998]
[LNKD][,][2014-08-05][,][41.6425]
[LNKD][,][2014-08-06][,][51.997499999999995]
[LNKD][,][2014-08-07][,][60.902499999999996]
[LNKD][,][2014-08-08][,][58.92]
[LNKD][,][2014-08-11][,][56.11]
[MSFT][,][2015-06-15][,][51.53]
[MSFT][,][2015-06-16][,][47.7275]
[MSFT][,][2015-06-17][,][46.175]
[MSFT][,][2015-06-18][,][45.614999999999995]
[MSFT][,][2015-06-19][,][46.74999999999999]
[TWTR][,][2013-09-24][,][38.864999999999995]
[TWTR][,][2013-09-25][,][29.924999999999997]
[TWTR][,][2013-09-26][,][22.752499999999998]
[TWTR][,][2013-09-27][,][15.337499999999999]
[TWTR][,][2013-09-30][,][17.0075]
```

**Output is following format:**

`[COMPANYSYMBOL][,][Date][,][Simplemovingaverage]`

- v) **calculatesma** – This class contain all the logic for calculating simple moving average. Just take window size and closing price as input and return simple moving average as output.