Parallel Implementation of Moving Averages on RMESH

John JENQ
Computer Science Department
Montclair State University
Montclair, New Jersey, USA

ABSTRACT

Moving averages are important financial indicators. There are short term moving average and long term moving averages. The interaction of the short term moving averages and long term averages give analyst the good clues about the direction of future assets performance. In this paper, the computation of two popular moving averages are discussed. The n-day simple moving average treats all past n days' closing prices equally while the n-day exponential moving average assigns more weight to most recent day and least weight to lease recent day closing price when form the computation formula. Both methods can be done in O(logN) time on the Reconfigurable Mesh.

Keywords: moving average, parallel processing, high performance computing. Reconfigurable mesh

1. INTRODUCTION

Moving averages can be used as financial indicators. The indicators are essential for companies and financial firms to predict and forecast the future performance of certain companies and stocks. See [4] for example. There are various types of moving averages. The two most popular moving averages are simple moving average and exponential moving average.

Assuming the daily closing price for day *t* is C_t. The *n*-day simple moving average can be defined as $SMA(t) = \frac{\sum_{i=t-1}^{i=t-n} C_i}{n}$, where C_i is the closing price at day *i*. The simple moving average can be computed by taking the average closing price of a stock, over the last *N* periods. Some of the popular simple moving averages are 5, 10, 20, 40, and 200. The 5-day and 10-day averages are considered as short term average while the 200-day average will be considered as long term average. The crossing of the long term and short term averages can be used by some investor as market indicators to buy or sell stocks. Let's assume the last ten periods for a stock are 1, 2, 3, 4, 5, 6, 7, 8, 9,and 10 then the 10 day simple moving average can be computed as (1+2+3+4+5+6+7+8+9+10)/10 = 5.5. While simple moving average giving all past *n*-day closing price equally weight, the *n*-day exponential moving average assign more weight to recent price.

Exponential moving average can be defined as $EMA(t) = (P_t - EMA(t-1)) * \alpha + EMA(t-1)$, where P_t is closing price at day *t*. The α is weighting factor and can be defined as $\alpha = 2/(n+1)$, where n is the number of time periods involving in the computation. For example, for 10-day EMA, $\alpha = 2/(10+1) = 0.181818$. While 20-day EMA, $\alpha = 2/(20+1) = 0.095238$.

In [5], Zhou and Zhang used financial indicators such as moving averages, volumes, Relative strength index, etc. on neural network to predict future stock price. In [2], Jenq proposed computation of SMA and EMA in order to be used in stock market prediction run on a neural network using GPU.

The rest of the paper was organized as the following. Section 2 discusses the Reconfigurable Mesh architecture, Section 3 discussed the basic operations that will be used as building blocks to construct our algorithms to compute simple moving average and exponential moving average. Section 4 discuss the

implementation of both simple moving average and exponential moving average on RMESH. The time complexity for both algorithms are also discussed. Section 5 concludes this report.

2. PRELIMINARIES ON RECONFIGURABLE MESH WITH BUSES (RMESH)

The particular reconfigurable mesh architecture that we use in this paper is called RMESH[3]. It employs a reconfigurable bus to connect together all processors. Figure 1 shows a $4\times4\times2$ RMESH. By opening some of the switches, the bus may be reconfigured into smaller buses that connect only a subset of the processors.
The important features of an RMESH are:

a. An $N\times M\times L$ RMESH is a 3-dimensional mesh-connected array of processing elements (PEs). Each PE in the RMESH is connected to a broadcast bus, which is itself constructed as an $N\times M\times L$ grid. The PEs are connected to the bus at the intersection of the grids. Each PE manages up to six bus switches (Figure 3) that are software controlled and can be used to reconfigure the bus into sub buses. The ID of each PE is a triple (*i , j, k*) where *i* is the row index, *j* is the column index and *k* is the plane index. The ID of the upper left corner PE on plane zero is *(0,0,0)* and that of the lower right one is (*N-1,M-1,0*).

b. There are up to six switches associated with a PE and are labeled as E (east), W (west), S (south), N (north), B(back), and F(front). Notice that the east (west, north, south, back, front) switch of a PE is also the west (east, south, north, front, back) switch of the PE (if any) on its right (left, top, bottom, back, front). Two PEs can simultaneously set (connect, close) or unset (disconnect, open) a particular switch as long as the settings do not conflict. The broadcast bus can be subdivided into subbuses by opening (disconnecting) some of the switches.

c. Only one processor can put data onto a given sub bus at any time

d. In unit time, data put on a subbus can be read by every PE connected to it. If a PE is to broadcast a value in register I to all of the PEs on its subbus, then it uses the command broadcast(I).

e. To read the content of the broadcast bus into a register R, the statement R = content(bus) is used.

f. Row buses are formed if each processor disconnects (opens) its S switch, B switch, and connects (closes) its E switch. Column buses are formed by disconnecting the E, and B switches and connecting the S switches. Similarly, Z buses can be formed by connecting F(or B) switches and disconnecting E and S switches. While the plane buses can be formed by disconnecting B switch only.

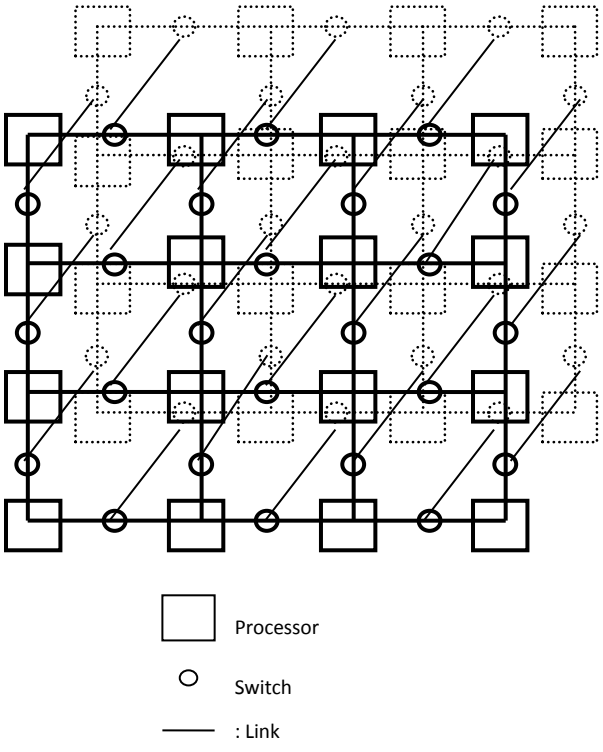

Figure 1 a 4 x 4 x 2 RMESH

3. BASIC DATA MANIPULATION OPERATIONS

The following are operations which will serve as the building blocks to form our algorithms to solve the moving average problem. You can find the more detailed discussion in [3], [1].

3.1. Broadcast

In a data broadcast operation, data originated in one PE are sent to the remaining *N -1* PEs, where *N* is the total number of PEs in the RMESH network. This operation takes *O*(*1*) time.

3.2 Diagonalization

This operation will *diagonalize* a row (column) of elements, by which we mean moving a specific row (column) elements to diagonal positions with respect to that row (column). With the RMESH bus, this operation can be done in *O(1)* time.

3.3 DataSum

Let's consider a special case as in [1] when the data to be summed are in the same row. By applying the similar operation as prefix sum, we can get the sum of data in *O(logN)* time.

3.4 PrefixSum

Each PE holds a number in its A variable.The prefix sum operation computes the partial sum of each PE(i) according to the following equation:

$$PrefixSum(i) = \sum_{j=0}^{i} A(j)$$

This operation can be done in *O(logN)* time.

4. COMPUTATION OF MOVING AVERAGE ON RMESH

In this section, we will discuss both the computation of simple moving average and exponential moving average.

4.1 Simple moving average on RMESH

The main operation in computation of simple moving average is to calculate the prefix sum. By using the prefix sum operation, the *N*-day Simple moving average of day *"i"* can be calculate as

SMA[i] = (prefixSum[i] – prefixSum[i-N])/N

The following is the step by step detailed algorithm that compute SMA in RMESH

SMA (*n*)

//compute *n* day simple average on RMESH

//input: *PE(0,j*) contains the closing price *Svalue* for day *j*

//output: *PE(0,j*) receive SMA for day *j*

// for simplicity we omit the detail of handle day 0 to day *j-1*

Step1 Computer prefix sum on row *0*

Step2 Diagonlization (Savlue)

Step3 Setup row bus

Step4 broadcast Svalue

Step5 *P[i,j]* disconnect East switch, where *i==j*

Step6 *P[i,j]* broadcast Svalue, where *i ==(j-n)*

Step7 Tvalue = BusContent for *P[i,j]* where *i==j*

Step8 *SMA = Svalue – Tvalue*

Step9 set up column bus

Step10 *P[i,j]* broadcast *SMA*, where *i==j*

Step11 *PE[0,j]* receive SMA

Figure 2. Computing Simple Moving Average on RMESH

All steps except prefix sum operation takes constant time. Because prefix sum takes *O(logN)* time, so the total time complexity is *O(logN).*

4.2 Exponential moving average on RMESH

Although the exponential moving average can be computed by using the formula

$$EMA(t) = (P_t - EMA(t-1)) * \alpha + EMA(t-1)$$

We can rewrite the formula as the follows.

$$EMA(t) = \alpha P_t + \alpha(1-\alpha)P_{t-1} + \alpha(1-\alpha)^2 P_{t-2} \ldots + \alpha(1-\alpha)^{n-1} P_{t-n+1}$$

Let's denote$(1-\alpha)\ as\ \beta$. The above formula will then become

$$EMA(t) = \alpha P_t + \alpha\beta P_{t-1} + \alpha\beta^2 P_{t-2} \ldots + \alpha\beta^{n-1} P_{t-n+1}$$

The algorithm that computes the EMA is listed as in Figure 3

EMA (*n*)

//compute *n* day Exponent moving average on

// RMESH

//input: *PE(0,j*) contains the closing price *Pvalue*

//for day *j*

//output: *PE(0,j*) receive EMA for day *j*

Step1 PE[0,0] broadcast alpha

Step2 PE[0,0] broadcast beta

Step3 Setup column bus

Step4 Broadcast Pvalue[0,j]

Step5 *P[i,j]* compute($\alpha * \beta^i * Pvalue$), where *i<=j*

Step6 *if (i==j) PE[I,j]* disconnect E-switch

Step7 Setup row bus

Step8 Perform DataSum from PE[I,0] to PE[I,j], where *i== j,* put result in DS

Step9 set up column bus

Step10 *P[i,j]* broadcast DS, where *i==j*

Step11 *PE[0,j]* receive DS and put it in *EMA (EMA[0,j] = DS)*

Figure 3 Computing Exponential Moving Average on RMESH

In Step5, each individual term in the EMA formula is calculated as shown in Figure4.

α					Row 0
$\alpha\beta^1$	α				Row 1
$\alpha\beta^2$	$\alpha\beta^1$	α			Row 2
$\alpha\beta^3$	$\alpha\beta^2$	$\alpha\beta^1$	α		Row 3
$\alpha\beta^4$	$\alpha\beta^3$	$\alpha\beta^2$	$\alpha\beta^1$	α	Row 4

(a) Populate α and β

Figure 4(a). Computing of $\alpha * \beta^i * Pvalue$

α					Row 0
$P_0\alpha\beta^1$	$P_1\alpha$				Row 1
$P_0\alpha\beta^2$	$P_1\alpha\beta^1$	$P_2\alpha$			Row 2
$P_0\alpha\beta^3$	$P_1\alpha\beta^2$	$P_2\alpha\beta^1$	$P_3\alpha$		Row 3
$P_0\alpha\beta^4$	$P_1\alpha\beta^3$	$P_2\alpha\beta^2$	$P_3\alpha\beta^1$	$P_4\alpha$	Row 4

(b) Prepare the term $\alpha * \beta^i * Pvalue$ for step8 DataSum

Figure 4(b). Computing of $\alpha * \beta^i * Pvalue$

It is easy to verify that all steps take constant time except Step8 which takes *O(logN)* time. Therefore the total time complexity for computing exponential moving average on RMSEH is *O(logN)* time.

5. CONCLUSION REMARKS

In this paper, the computation of two moving averages on RMESH was developed. The Simple moving average which treats all past n-day stock closing prices equally important in calculating today's moving average. The Exponential moving average assigns more weights for most recent day and least weight for least recent day. Both of these two algorithms perform the same time complexity which are both in *O(logn).* Question is can we do better than *O(logN)* time?

6. REFERENCES

[1] J. Jenq and S. Sahni, "Reconfigurable Mesh Algorithms for Fundamental Data Manipulation Operations", *Computing on Distributed Memory Multiprocessors*, *NATO Series F*, Ed. F. Ozguner, *Spring Verlag*, 1993, pp 27-46

[2] John Jenq, "Parallel Implementation of Moving Averages and Stock Market Prediction", by John Jenq, 2012 International Conference on Parallel and Distributed Processing Techniques and Applications, pp 833-837

[3] R. Miller, V.K. Prasanna Kumar, D. Reisis, and Q.F. Stout, Parallel computations on reconfigurable meshes, *IEEE Transactions on Computers*, vol. 42, no. 6, June 1993, pp. 678-692.

[4] Edward Tirados and John Jenq, "Analysis of Leading Economic Indicator Data and Gross Domestic Product Data Using Neural Network Methods", Journal of Systemics, Cybernetics and Informatics, vol 7, no 4, 2009, pp 51-56

[5] Yixin Zhou, and Jie Zhang, "Stock data analysis based on BP neural network", 2010 Second International Conference on Communication Software and Networks, pp 396 - 399