

# **Scheduling Divisible Workloads from Multiple Sources in Regular and Torus Mesh**

A Dissertation Presented

by

**Junwei Zhang**

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

**Doctor of Philosophy**

in

**Applied Mathematics and Statistics**

Stony Brook University

**August 2018**

**Stony Brook University**

The Graduate School

**Junwei Zhang**

We, the dissertation committee for the above candidate for the  
Doctor of Philosophy degree, hereby recommend  
acceptance of this dissertation.

**Thomas G. Robertazzi - Dissertation Advisor**  
**Professor, Department of Electrical and Computer Engineering**

**Joseph S.B. Mitchell - Chairperson of Defense**  
**Professor, Department of Applied Mathematics and Statistics**

**Esther M. Arkin - Member**  
**Professor, Department of Applied Mathematics and Statistics**

**Yue Zhao - Member**  
**Professor, Department of Electrical and Computer Engineering**

This dissertation is accepted by the Graduate School.

Charles Taber  
Dean of the Graduate School

Abstract of the Dissertation

**Scheduling Divisible Workloads from Multiple  
Sources in Regular and Torus Mesh**

by

**Junwei Zhang**

**Doctor of Philosophy**

in

**Applied Mathematics and Statistics**

Stony Brook University

**2018**

Abstract Here

*Key Words:*

*To my Parents and all loving ones*

# Table of Contents

<b>List of Figures</b>	<b>ix</b>
<b>List of Tables</b>	<b>x</b>
<b>Acknowledgements</b>	<b>xi</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Closed-form Processor Equivalence</b>	<b>2</b>
2.1 Problem Description	3
2.2 With Front End Scenario	8
2.2.1 Regular Mesh	8
2.2.2 Torus mesh	32
2.2.3 General Case	37
2.3 Without Front End Scenario	38
2.3.1 Regular Mesh	38
2.3.2 Toroidal Mesh	51
2.3.3 General Case	53

2.4	Comparing Result Between Front End Processor and Without	
	Front End Processor . . . . .	54
2.4.1	Regular Mesh . . . . .	54
2.4.2	Toroidal . . . . .	58
<b>3</b>	<b>Sensitivity Analysis . . . . .</b>	<b>60</b>
<b>4</b>	<b>Multi-source Assignment Utilizing Voronoi Diagram . . . . .</b>	<b>61</b>
<b>5</b>	<b>Multi-source Re-assignment Using Optimal Mass Transporta-</b>	
	<b>tion . . . . .</b>	<b>62</b>
<b>6</b>	<b>Conclusions . . . . .</b>	<b>63</b>
	<b>Bibliography . . . . .</b>	<b>64</b>

## List of Figures

2.1	A $m \times n$ regular mesh( $m = 5, n = 5$ ) . . . . .	4
2.2	A toroidal mesh with grid unit cores . . . . .	4
2.3	The $2 \times 2$ regular mesh and the data injection is on $P_0$ . . . . .	9
2.4	The timing diagram for $2 \times 2$ regular mesh and the data injection position is $P_0$ . . . . .	10
2.5	$2 \times 2$ regular mesh. $\alpha_0, \alpha_1, \alpha_2, \alpha_3$ value . . . . .	11
2.6	The $2 \times 3$ regular mesh and the data injection happens on corner processor $P_0$ . . . . .	12
2.7	The timing diagram for a $2 \times 3$ regular mesh and the data injection happens on processor $P_0$ . . . . .	14
2.8	$2 \times 2$ regular mesh. $\alpha_0, \alpha_1, \alpha_3, \alpha_5$ data fraction value . . . . .	16
2.9	The $2 \times n$ ( $n = 10$ ) regular mesh and the workload happens on $P_0$	17
2.10	The timing diagram for $2 \times 10$ regular mesh and the data injection happens on $P_0$ . . . . .	19
2.11	$3 \times 8$ regular mesh.The data injection position is $P_0$ . . . . .	21
2.12	$5 \times 5$ regular mesh.The data injection position is processor $P_0$ . .	22
2.13	The $3 \times 3$ regular mesh and the data injection position is $P_0$ . . .	24

2.14	The timing diagram for 3*3 regular mesh and the data injection occurs on the boundary processor $P_0$ . . . . .	25
2.15	The data fraction simulation result of 3*3 regular mesh and the data injection happens on the boundary $P_0$ . . . . .	27
2.16	3*3 regular mesh.The data injection position is inner grid point $P_0$	28
2.17	The timing diagram for 3*3 regular mesh and the data injection is inner grid $P_0$ . . . . .	29
2.18	3*3 regular mesh.The data injection position is inner grid point $P_0$	31
2.19	The rectangular toroidal network . . . . .	32
2.20	The rectangular toroidal network . . . . .	33
2.21	The m*n toroidal network and the data injection is $P_{4,2}$ . . . . .	34
2.22	The data fraction deployed based on the radius value . . . . .	36
2.23	The timing diagram for 2*2 regular mesh without front end. . .	39
2.24	The data fraction deployed based on the radius value . . . . .	41
2.25	The timing diagram for 2*3 regular mesh without front end. . .	42
2.26	The data fraction deployed based on the radius value . . . . .	44
2.27	The timing diagram for 3*3 boundary data injection on $P_0$ . . .	47
2.28	The fraction curve for 3*3 boundary data injection on $P_0$ . . .	49
2.29	The timing diagram for 3*3 inner grid injection $P_0$ . . . . .	51
2.30	The data fraction deployed based on the radius value . . . . .	52
2.31	The comparing result between front-end processor with without front-end processor in 2*2 regular mesh . . . . .	55
2.32	The comparing result between front-end processor with without front-end processor in 2*4 regular mesh . . . . .	56



2.33	The comparing result between front-end processor with without front-end processor in 3*3 regular mesh injection on boundary processor . . . . .	57
2.34	The comparing result between front-end processor with without front-end processor in 3*3 regular mesh injection on inner grid processor . . . . .	58
2.35	The comparing result between front-end processor with without front-end processor in 6*6 regular mesh injection on inner grid processor . . . . .	59

## List of Tables

2.1	The processor number of various $D_i$ . . . . .	35
-----	---	----

## Acknowledgements

# Chapter 1

## Introduction

[?, ?]

## Chapter 2

### Closed-form Processor Equivalence

In large-scale data intensive problems with geographically distributed resources, load is generated from multiple sources[1].It is assumed that the problem representation can be divided amongst the processors.Thus the problem representation is said to be "divisible". This chapter we discuss the closed-form processor equivalence[2][3] problem in the grid network of regular mesh,toroidal rectangle mesh.

This chapter is organized as follows.A problem introduction,notation and definitions are shown in Section I.

Section II considers about the unit core with front-end scenario.In the front-end situation,the regular mesh and torus rectangle mesh closed-form processor equivalence formula is investigated.Considering some various closed-form formula,the data injection position occurs on the corner processor,the boundary unit core or the inner grid processor.

Section III presents the unit core without front-end situation.In addition, the regular mesh and toroidal are discussed,respectively.

In section IV we give a comprehensive comparison result considering the different factors, for example the ratio between the communication ability to the computation capacity, the data injection position and the number of processors.

## 2.1 Problem Description

The processing of massive amounts of data on distributed and parallel networks is becoming more and more common. The problem of minimizing the processing time of extensive loads originating from a multiplicity of sources and being processed on a multiplicity of nodes presents a challenge.

The research user case involves the regular mesh Fig. 2.12 or toroidal mesh Fig. 2.2 of  $m \times n$  processors. We refer to the  $m \times n$  processor as  $P_0, P_1, P_2, \dots, P_{m \times n - 1}$ .

The workloads are denoted as  $\mathbf{L}_1, \mathbf{L}_2, \dots, \mathbf{L}_k$ . We consider two situations, such as the processor contains the front-end and without front-end. In order to simply the question model, we have the following assumptions:

- We do not consider return communications.
- Communication delays are taken into consideration.
- The time costs of computation and communication are assumed to be linear function of the data size.
- The network environment is homogeneous, that is, all the processors have the same computation capacity.
- The link speeds between any two unit cores are identical.

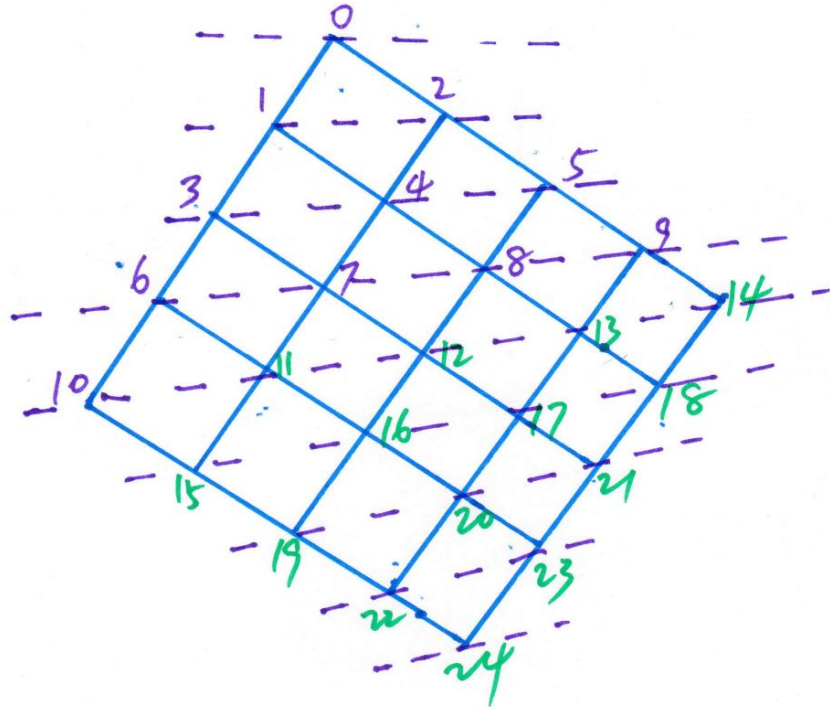


Figure 2.1: A  $m \times n$  regular mesh ( $m = 5, n = 5$ )

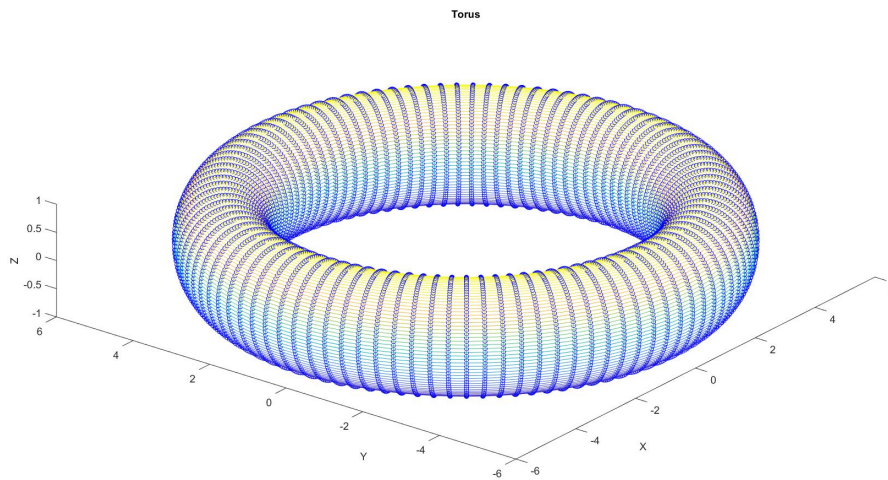


Figure 2.2: A toroidal mesh with grid unit cores

- The port of each processor is limited. In NOC (network on chip), the port number is 5.

In the front-end assumption, the communication and the computation can be executed simultaneously. That is, upon receiving their respective load fractions, the processors start processing their own workload and rely all the other fractions to the next processor at the same time.

In the without front-end scenario, unlike the previous protocol, the processors now simultaneously receive the data and only start to process it as soon as each processor receives its entire load assignment [4].

Thus, the problem becomes how to partition and schedule the workloads amongst the processors to get the minimum finish time. This study will focus on the speedup metric which is defined as the ratio of computation time on one processor to the computation time on the entire  $m * n$  processors.

To achieve the minimum solution is obtained by forcing the processors over a network to stop processing simultaneously. Intuitively, this is because the solution could be improved by transfer load from some busy processor to idle ones. The following notations and definitions are utilized:

### Notations and Definitions

- $D_i$ : The minimum number of hops from the node  $P_i$  to the nearest data load injection  $L$ .
- $level_i$ : The manhattan distance to the nearest data injection
- $P_i$ : The  $i$ th processor.  $i$  should be from 0 to  $m * n - 1$ .



- $L_i$ : The  $i$ th work load.  $i$  should be from 1 to  $k$ .
- $\alpha_0$ : The load fraction assigned to the root processor.
- $\alpha_i$ : The load fraction assigned to the  $i$ th processor.
- $\omega_i$ : The inverse computing speed on the  $i$ th processor.
- $\omega_{eq}$ : The inverse computing speed on an equivalent node collapsed from a regular mesh or toroidal.
- $z_i$ : The inverse link speed on the  $i$ th link.
- $T_{cp}$ : Computing intensity constant. The entire load can be processed in  $\omega_i T_{cp}$  on the  $i$ th processor.
- $T_{cm}$ : Communication intensity constant. The entire load can be transmitted in  $z_i T_{cm}$  seconds over the  $i$ th link.
- $T_{f,n}$ : The finish time of the whole regular network. Here  $T_{f,n}$  is equal to  $\omega_{eq} T_{cp}$ .
- $T_{f,0}$ : The finish time for the entire divisible load solved on the root processor. Here  $T_{f,0}$  is equal to  $1 \times \omega_0 T_{cp}$ , that is  $\omega_0 T_{cp}$ .
- $\sigma = \frac{z T_{cm}}{\omega T_{cp}}$ : The ratio between the communication speed to the computation speed.
- $\sum_{i=1}^k L_i = 1$
- $\sum_{i=0}^{m*n-1} \alpha_i = 1$

- $Speedup = \frac{T_{f,0}}{T_{f,n}} = \frac{\omega T_{cp}}{\alpha_0 \omega T_{cp}} = \frac{1}{\alpha_0}$

## 2.2 With Front End Scenario

The processors compute and transfer the data load simultaneously. First we consider about the  $2 \times 2$  regular mesh,  $2 \times n$  regular mesh. After, we analyze a more general case  $m \times n$  regular mesh and obtain a general closed-form matrix presentation. Finally, we give a key principle to address this type of question. In addition, different data injection position, such as the corner, boundary and inner grid point are also discussed.

### 2.2.1 Regular Mesh

#### Data Injection on The Corner Processor

##### $2 \times 2$ Regular Mesh

The  $L_1$  is assigned on the corner unit core  $P_0$  Fig. 2.3. The whole task are tackled by four processors  $P_0, P_1, P_2, P_3$  together.

The processor  $P_0, P_1$  and  $P_2$  start to process its respective fraction at the same time. The port number of  $P_0$  is limited, so the processor  $P_3$  handles until the  $\alpha_1$  and  $\alpha_2$  are assigned completed to  $P_1$  and  $P_2$ .

According to the divisible load theory[5], we can obtain the timing diagram Fig. 2.4.

Based on the timing diagram, we can set the equations for the finish time of each processor to get the optimal fragment size of loads:

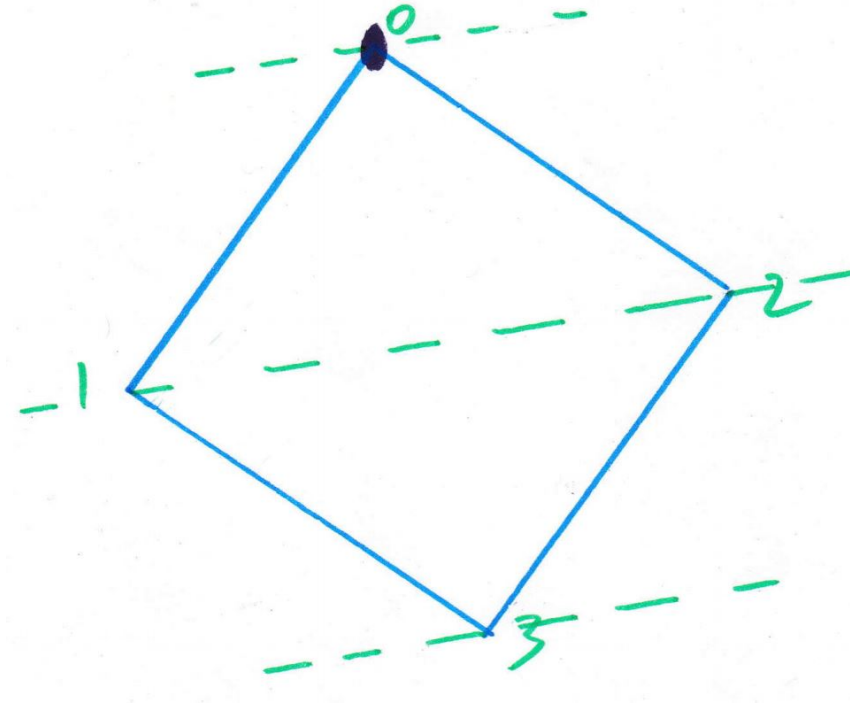


Figure 2.3: The 2\*2 regular mesh and the data injection is on  $P_0$

$$\left\{ \begin{array}{ll} \alpha_0 \omega T_{cp} = T_{f,m} & (2.1) \\ \alpha_0 \omega T_{cp} = T_{f,m} & (2.2) \\ \alpha_1 \omega T_{cp} = T_{f,m} & (2.3) \\ \alpha_2 \omega T_{cp} = T_{f,m} & (2.4) \\ \alpha_1 z T_{cm} + \alpha_3 \omega T_{cp} = T_{f,m} & (2.5) \\ \alpha_0 + \alpha_1 + \alpha_2 + \alpha_3 = 1 & (2.6) \\ \sigma = \frac{z T_{cm}}{\omega T_{cp}} & (2.7) \\ 0 < \sigma < 1 & (2.8) \\ 0 < \alpha_0, \alpha_1, \alpha_2, \alpha_3 < 1 & (2.9) \\ 9 & (2.10) \end{array} \right.$$

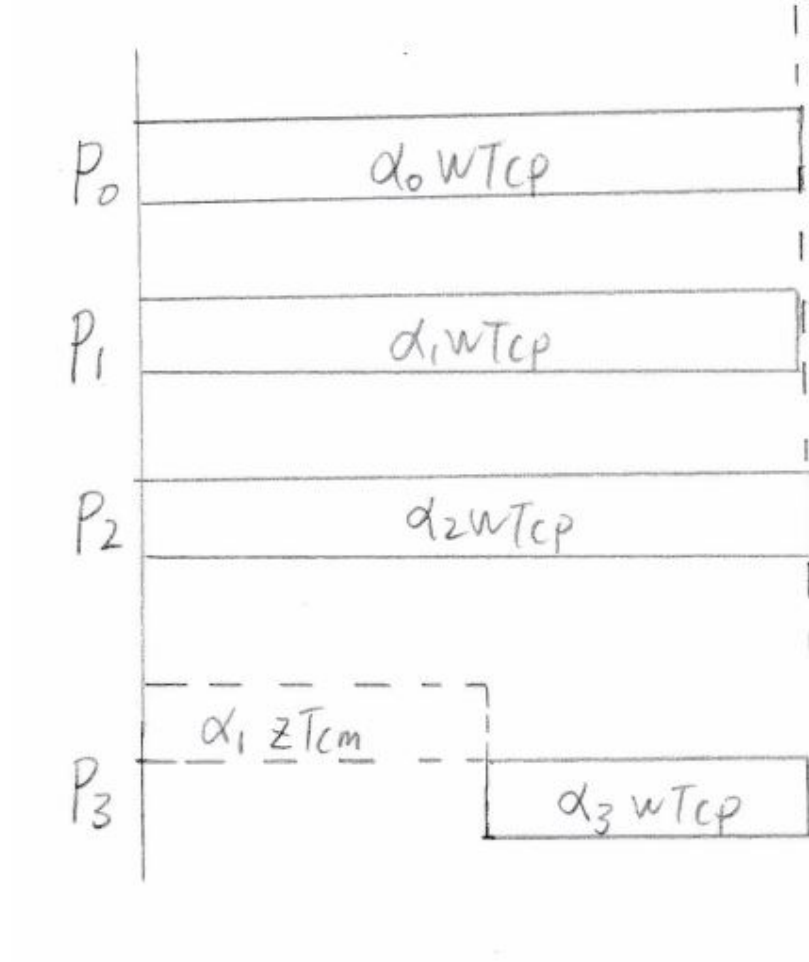


Figure 2.4: The timing diagram for 2\*2 regular mesh and the data injection position is  $P_0$

We can obtain the

$$\alpha_1 = \alpha_2$$

The group of equations are represented by the matrix form:

$$\begin{bmatrix} 1 & 2 & 1 \\ 1 & -1 & 0 \\ 0 & \sigma - 1 & 1 \end{bmatrix} \times \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad (2.11)$$

The simulation result is illustrated:

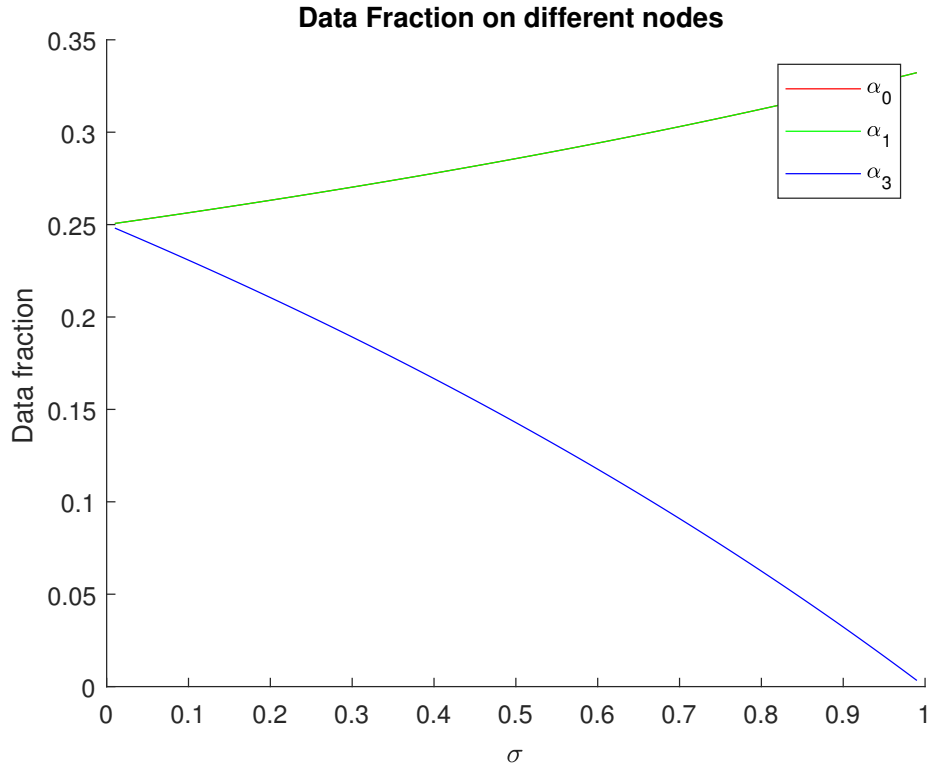


Figure 2.5: 2\*2 regular mesh.  $\alpha_0, \alpha_1, \alpha_2, \alpha_3$  value

In Fig. 2.5,  $P_0, P_1, P_2$  three processors have the same data fraction workload, so the curve of  $\alpha_0$  and  $\alpha_1$  coincide. The figure says that the  $\sigma$  value grows and the  $\alpha_3$  drops simultaneously. In other words, the communication capacity

drops down and there is less data workload assigned to  $P_3$ . Further, it means it will be economical to keep the load local nor distribute to other processor.

The speedup is:

$$Speedup = \frac{T_{f,0}}{T_{f,n}} = \frac{\omega T_{cp}}{\alpha_0 \omega T_{cp}} = \frac{1}{\alpha_0}$$

### 2\*3 Regular Mesh

In the 2 \* 3 Fig. 2.6 regular mesh,  $L_1$  happens on processor  $P_0$ . There are 6 processors to take the responsibility.

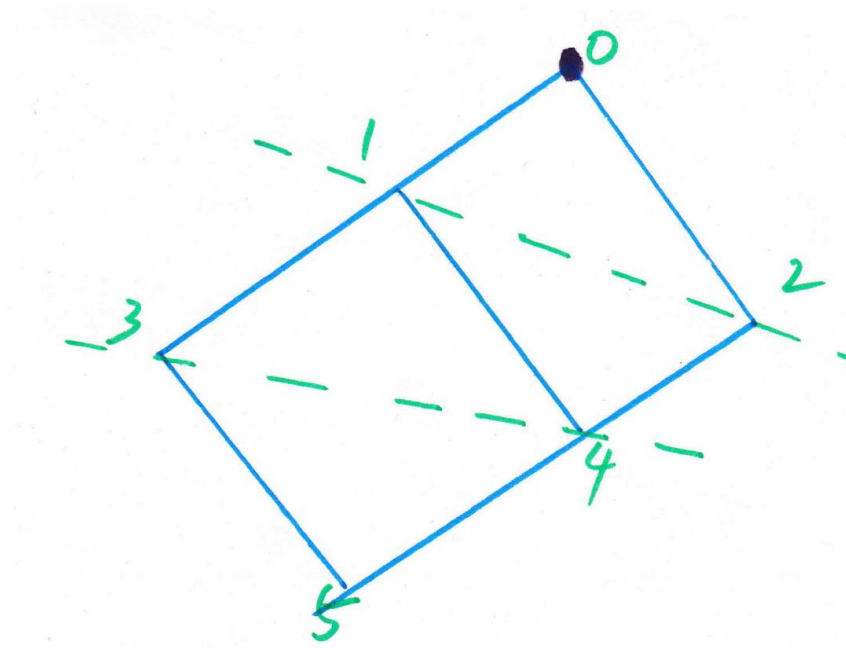


Figure 2.6: The 2\*3 regular mesh and the data injection happens on corner processor  $P_0$

$P_0, P_1$  and  $P_2$  start processing the load at the same time. According the processor  $P_3$  and  $P_4$  start to work until they get the data from their parent

processor  $P_1, P_2$ . Yet, the number of port of each processor is 4 or 5. So the fraction of  $\alpha_3, \alpha_4$  are queued in the  $L_1$ . So the  $P_3$  and  $P_4$  have to wait the fraction of  $\alpha_1$  and  $\alpha_2$  are transmitted completely and then they will obtain their respective workload task fraction.

The last unit core  $P_5$  starts to execute until the work load fraction  $\alpha_0, \alpha_1, \alpha_2, \alpha_3, \alpha_4$  are transmitted completed.

According to the divisible load theory[5], we can obtain the timing diagram Fig. 2.7.



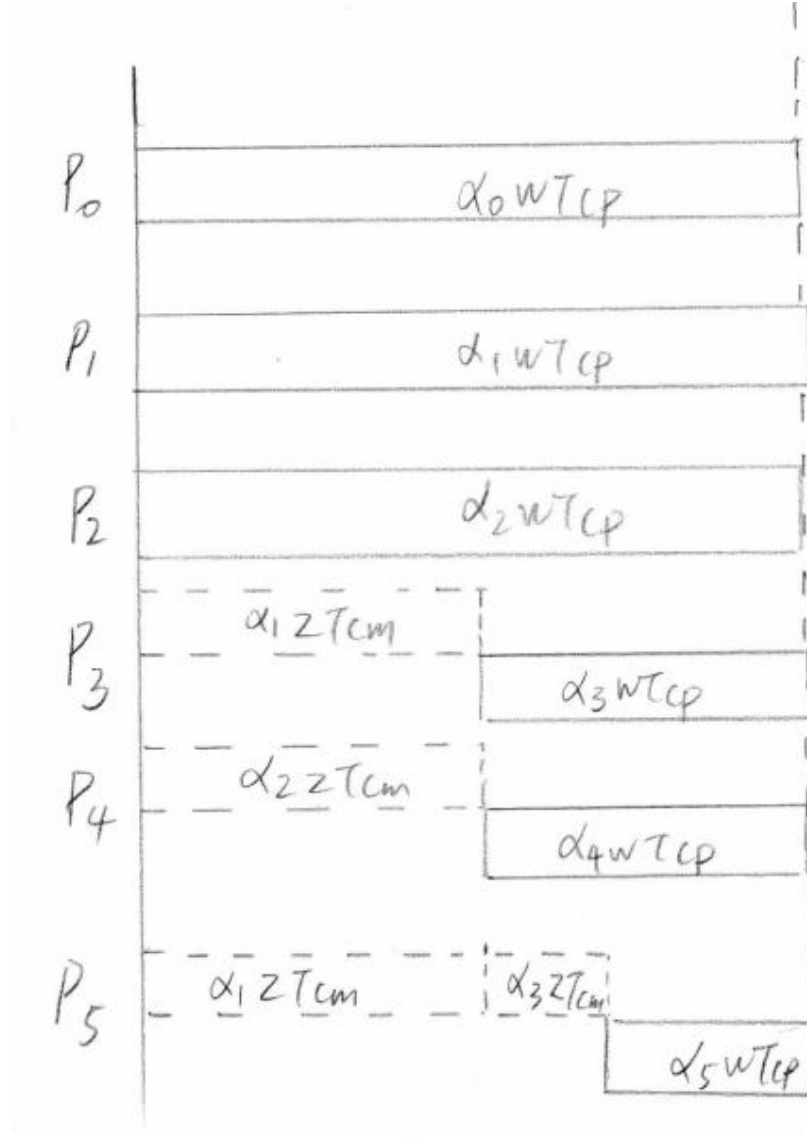


Figure 2.7: The timing diagram for a 2\*3 regular mesh and the data injection happens on processor  $P_0$

The equations as follows:

$$\left\{ \begin{array}{ll} \alpha_0 \omega T_{cp} = T_{f,m} & (2.12) \\ \alpha_1 \omega T_{cp} = T_{f,m} & (2.13) \\ \alpha_2 \omega T_{cp} = T_{f,m} & (2.14) \\ \alpha_1 z T_{cm} + \alpha_3 \omega T_{cp} = T_{f,m} & (2.15) \\ \alpha_2 z T_{cm} + \alpha_4 \omega T_{cp} = T_{f,m} & (2.16) \\ (\alpha_1 + \alpha_3) z T_{cm} + \alpha_5 \omega T_{cp} = T_{f,m} & (2.17) \\ \alpha_0 + \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + \alpha_5 = 1 & (2.18) \\ \sigma = \frac{z T_{cm}}{\omega T_{cp}} & (2.19) \\ 0 < \sigma < 1 & (2.20) \\ 0 < \alpha_0, \alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5 < 1 & (2.21) \end{array} \right. \quad (2.22)$$

After the simplification, we found:

$$\left\{ \begin{array}{l} \alpha_1 = \alpha_2 \\ \alpha_3 = \alpha_4 \end{array} \right. \quad (2.23)$$

$$\left\{ \begin{array}{l} \alpha_3 = \alpha_4 \end{array} \right. \quad (2.24)$$

The matrix closed-form formula is:

$$\begin{bmatrix} 1 & 2 & 2 & 1 \\ 1 & -1 & 0 & 0 \\ 0 & \sigma - 1 & 1 & 0 \\ 0 & \sigma - 1 & \sigma & 1 \end{bmatrix} \times \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_3 \\ \alpha_5 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (2.25)$$

The simulation result are shown in Fig. 2.8.  $P_0, P_1$  have the same fraction so their curves of  $\alpha_0$  and  $\alpha_1$  coincide.

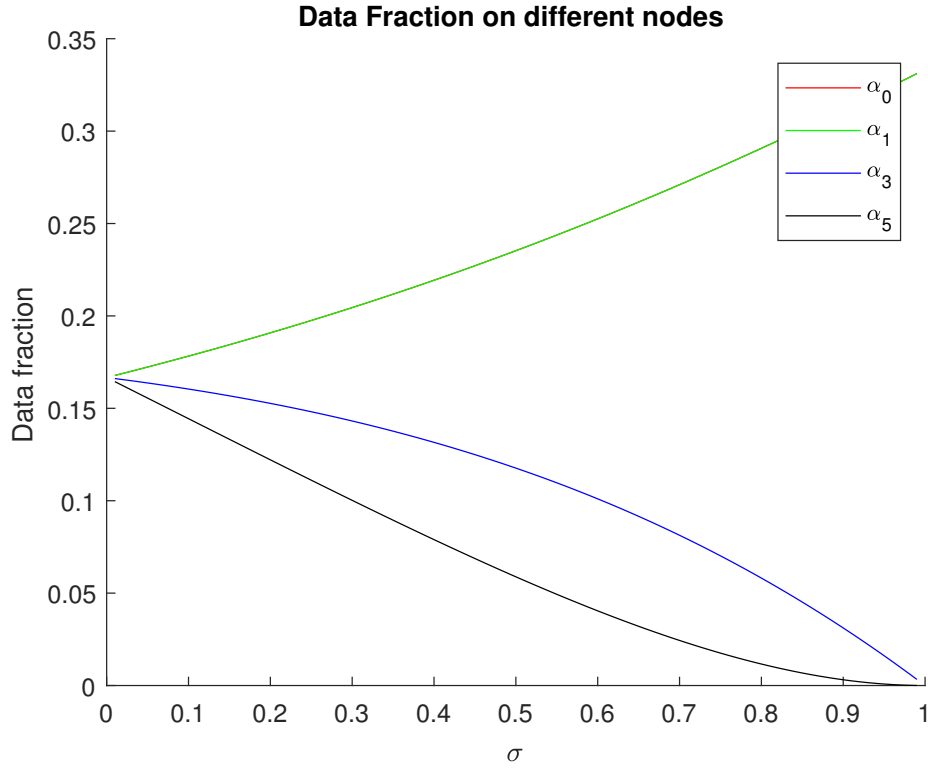


Figure 2.8: 2\*2 regular mesh.  $\alpha_0, \alpha_1, \alpha_3, \alpha_5$  data fraction value

## 2\*n Regular Mesh

The 2\*n Fig. 2.9 homogeneous regular mesh address  $L_1$  at the same time and  $L_1$  happens on  $P_0$ .

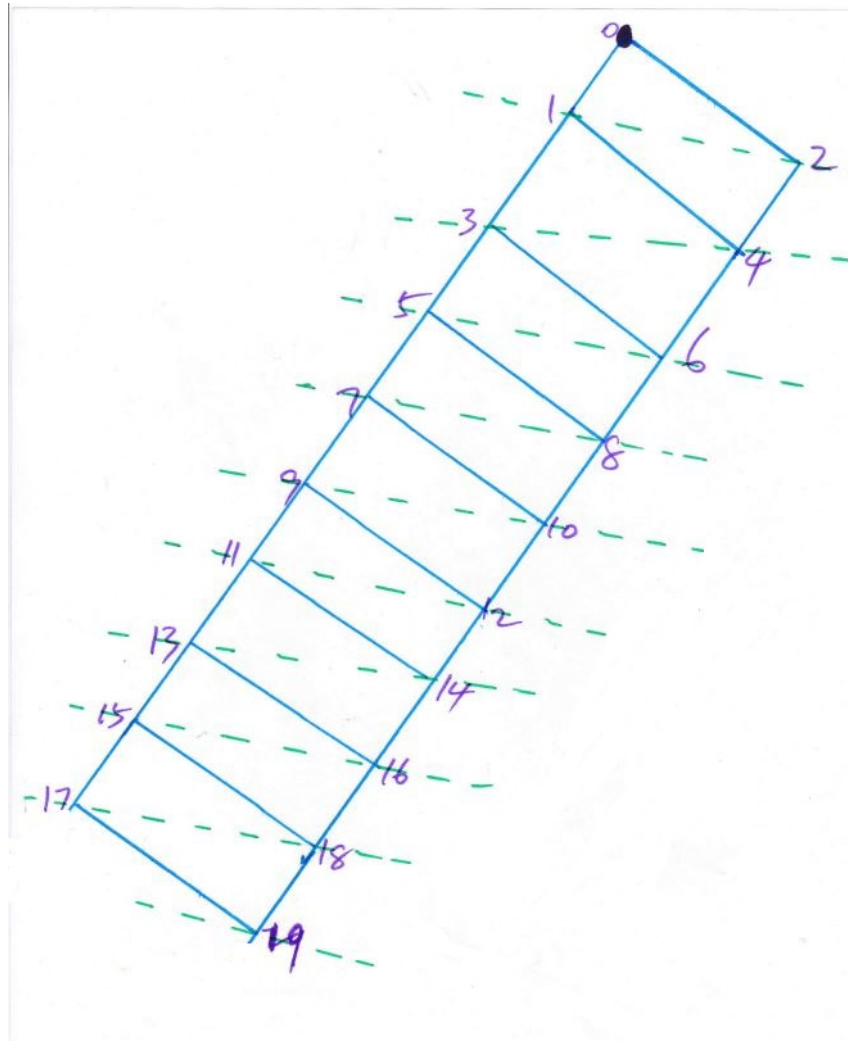


Figure 2.9: The 2\*n (n = 10) regular mesh and the workload happens on  $P_0$

Similarly to the analysis of Fig. 2.4 and Fig. 2.7, the timing diagram for Fig. 2.9 is shown in Fig. 2.10

The equations are presented as:

$$\left\{ \begin{array}{ll} \alpha_0 \omega T_{cp} = T_{f,m} & (2.26) \\ \alpha_1 \omega T_{cp} = T_{f,m} & (2.27) \\ \alpha_2 \omega T_{cp} = T_{f,m} & (2.28) \\ \alpha_1 z T_{cm} + \alpha_3 \omega T_{cp} = T_{f,m} & (2.29) \\ \alpha_2 z T_{cm} + \alpha_4 \omega T_{cp} = T_{f,m} & (2.30) \\ (\alpha_1 + \alpha_3) z T_{cm} + \alpha_5 \omega T_{cp} = T_{f,m} & (2.31) \\ \vdots & (2.32) \\ (\alpha_1 + \alpha_3 + \dots + \alpha_{2 \times n-1}) z T_{cm} + \alpha_{2 \times n-1} \omega T_{cp} = T_{f,m} & (2.33) \\ \alpha_0 + \dots + \alpha_{2 \times n-1} = 1 & (2.34) \\ \sigma = \frac{z T_{cm}}{\omega T_{cp}} & (2.35) \\ 0 < \sigma < 1 & (2.36) \\ 0 < \alpha_0 \quad \alpha_1 \quad \alpha_3 \quad \dots \quad \alpha_{2 \times n-1} < 1 & (2.37) \end{array} \right. \quad (2.38)$$

The matrix closed-form is shown:

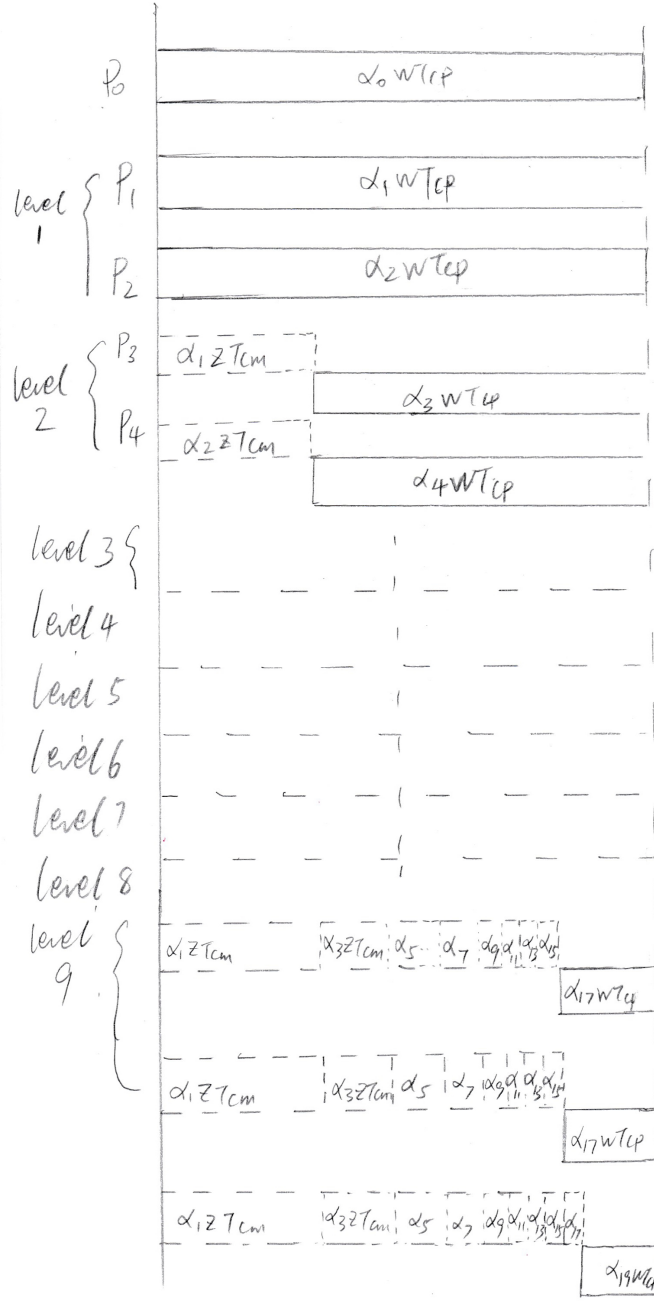


Figure 2.10: The timing diagram for 2\*10 regular mesh and the data injection happens on  $P_0$

$$\begin{bmatrix} 1 & 2 & 2 & \cdots & 2 & 2 & 1 \\ 1 & -1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & \sigma - 1 & 1 & \cdots & 0 & 0 & 0 \\ 0 & \sigma - 1 & \sigma & 1 & 0 & \cdots & 0 \\ 0 & \sigma - 1 & \sigma & \sigma & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \\ 0 & \sigma - 1 & \sigma & \cdots & \sigma & \sigma & 1 \end{bmatrix} \times \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_3 \\ \alpha_5 \\ \vdots \\ \alpha_{2 \times n - 3} \\ \alpha_{2 \times n - 1} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad (2.39)$$

Finally, the speedup is:

$$Speedup = \frac{T_{f,0}}{T_{f,n}} = \frac{\omega T_{cp}}{\alpha_0 \omega T_{cp}} = \frac{1}{\alpha_0}$$

After three user cases' investigation, we find a crucial rule:

$$\forall D_i = D_j, \quad \text{then} \quad \alpha_i = \alpha_j, \quad 0 \leq i, j \leq m * n - 1$$

## $m \times n$ Regular Mesh

Considering a general  $m \times n$  regular mesh, such as Fig. 2.11 Fig. 2.12.

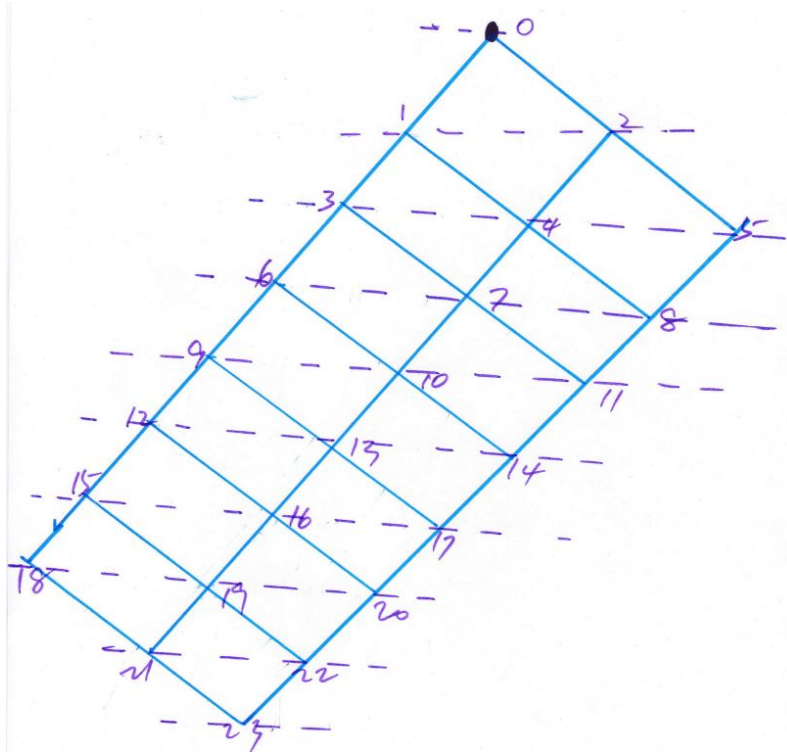


Figure 2.11:  $3 \times 8$  regular mesh. The data injection position is  $P_0$



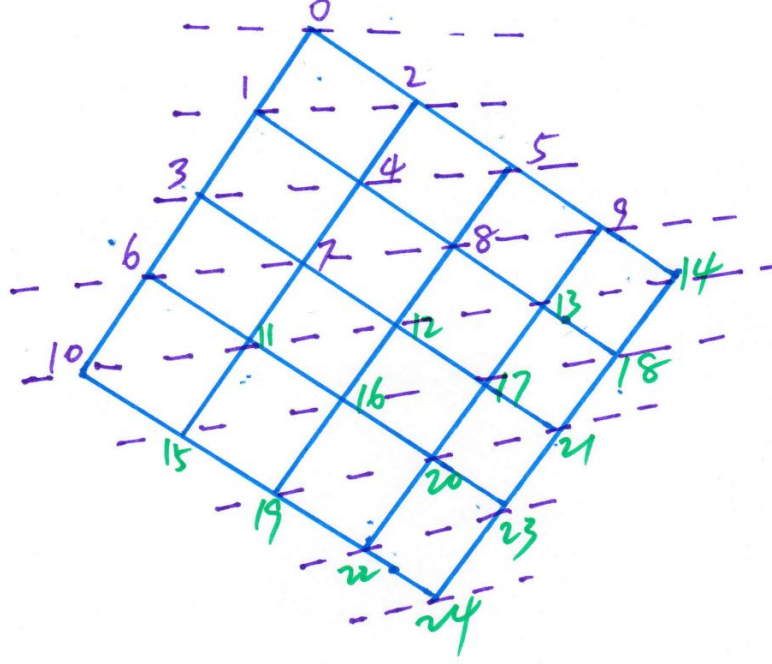


Figure 2.12: 5\*5 regular mesh. The data injection position is processor  $P_0$

Utilizing the rule, we obtain the closed-form matrix equations for Fig. 2.11:

$$\begin{bmatrix}
 1 & 2 & 3 & 3 & 3 & 3 & 3 & 2 & 1 \\
 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & \sigma - 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & \sigma - 1 & \sigma & 1 & 0 & 0 & 0 & 0 & 0 \\
 0 & \sigma - 1 & \sigma & \sigma & 1 & 0 & 0 & 0 & 0 \\
 0 & \sigma - 1 & \sigma & \sigma & \sigma & 1 & 0 & 0 & 0 \\
 0 & \sigma - 1 & \sigma & \sigma & \sigma & \sigma & 1 & 0 & 0 \\
 0 & \sigma - 1 & \sigma & \sigma & \sigma & \sigma & \sigma & 1 & 0 \\
 0 & \sigma - 1 & \sigma & \sigma & \sigma & \sigma & \sigma & \sigma & 1
 \end{bmatrix} \times \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_3 \\ \alpha_6 \\ \alpha_9 \\ \alpha_{12} \\ \alpha_{15} \\ \alpha_{18} \\ \alpha_{21} \\ \alpha_{23} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad (2.40)$$

Also, the matrix equation for Fig. 2.12:

$$\begin{bmatrix}
 1 & 2 & 3 & 4 & 5 & 4 & 3 & 2 & 1 \\
 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & \sigma - 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & \sigma - 1 & \sigma & 1 & 0 & 0 & 0 & 0 & 0 \\
 0 & \sigma - 1 & \sigma & \sigma & 1 & 0 & 0 & 0 & 0 \\
 0 & \sigma - 1 & \sigma & \sigma & \sigma & 1 & 0 & 0 & 0 \\
 0 & \sigma - 1 & \sigma & \sigma & \sigma & \sigma & 1 & 0 & 0 \\
 0 & \sigma - 1 & \sigma & \sigma & \sigma & \sigma & \sigma & 1 & 0 \\
 0 & \sigma - 1 & \sigma & \sigma & \sigma & \sigma & \sigma & \sigma & 1
 \end{bmatrix} \times \begin{bmatrix}
 \alpha_0 \\
 \alpha_1 \\
 \alpha_3 \\
 \alpha_6 \\
 \alpha_{10} \\
 \alpha_{15} \\
 \alpha_{19} \\
 \alpha_{22} \\
 \alpha_{24}
 \end{bmatrix} = \begin{bmatrix}
 1 \\
 0 \\
 0 \\
 0 \\
 0 \\
 0 \\
 0 \\
 \vdots \\
 0
 \end{bmatrix} \quad (2.41)$$

## Data Injection On the Boundary Processor

After explain the corner scenario, we extend the rule to boundary processor condition.

If the single data injection roots on the boundary processor,for example Fig. 2.13.

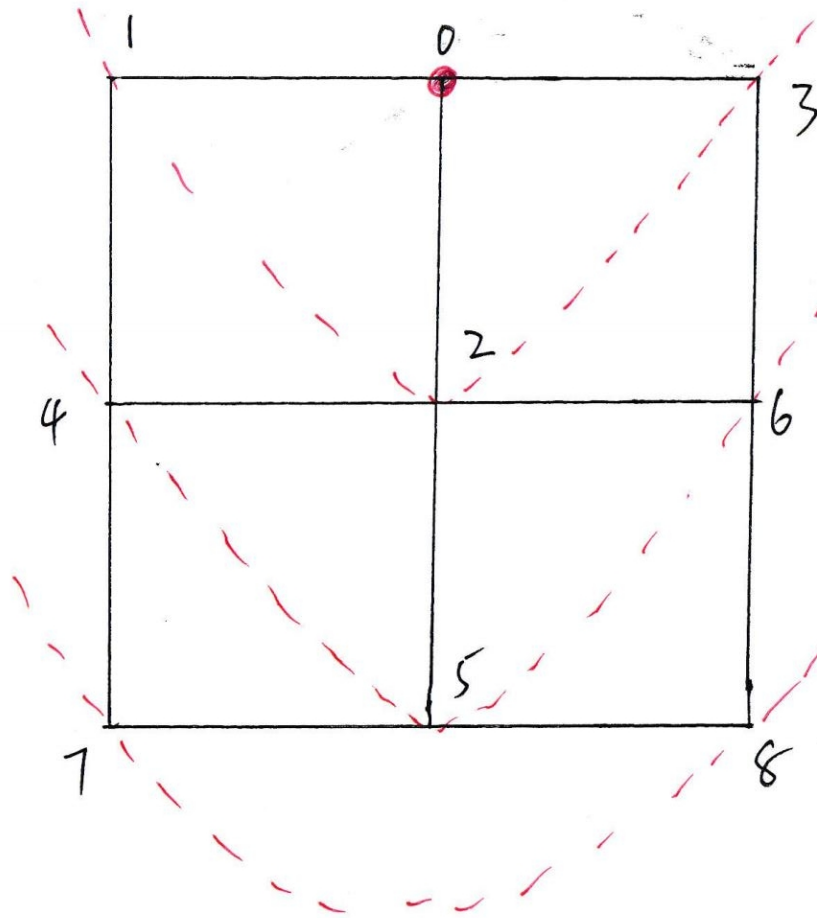


Figure 2.13: The 3\*3 regular mesh and the data injection position is  $P_0$

The timing diagram is Fig. 2.14:

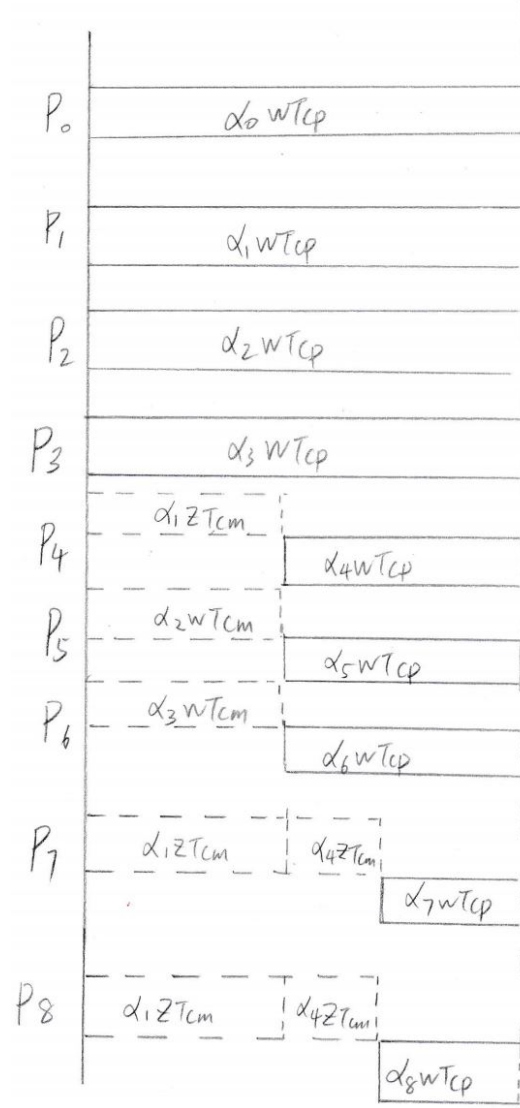


Figure 2.14: The timing diagram for 3\*3 regular mesh and the data injection occurs on the boundary processor  $P_0$

The equations are:

$$\left\{ \begin{array}{ll} \alpha_0 \omega T_{cp} = T_{f,m} & (2.42) \\ \alpha_1 \omega T_{cp} = T_{f,m} & (2.43) \\ \alpha_2 \omega T_{cp} = T_{f,m} & (2.44) \\ \alpha_3 \omega T_{cp} = T_{f,m} & (2.45) \\ \alpha_1 z T_{cm} + \alpha_4 \omega T_{cp} = T_{f,m} & (2.46) \\ \alpha_2 z T_{cm} + \alpha_5 \omega T_{cp} = T_{f,m} & (2.47) \\ \alpha_3 z T_{cm} + \alpha_6 \omega T_{cp} = T_{f,m} & (2.48) \\ (\alpha_1 + \alpha_4) z T_{cm} + \alpha_7 \omega T_{cp} = T_{f,m} & (2.49) \\ (\alpha_2 + \alpha_5) z T_{cm} + \alpha_8 \omega T_{cp} = T_{f,m} & (2.50) \\ \alpha_0 + \dots + \alpha_8 = 1 & (2.51) \\ \sigma = \frac{z T_{cm}}{\omega T_{cp}} & (2.52) \\ 0 < \sigma < 1 & (2.53) \\ 0 < \alpha_0 \quad \alpha_1 \quad \alpha_2 \quad \dots \quad \alpha_8 < 1 & (2.54) \end{array} \right. \quad (2.55)$$

And the matrix form is :

$$\begin{bmatrix} 1 & 3 & 3 & 2 \\ 1 & -1 & 0 & 0 \\ 0 & \sigma - 1 & 1 & 0 \\ 0 & \sigma - 1 & \sigma & 1 \end{bmatrix} \times \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_4 \\ \alpha_7 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (2.56)$$

And the simulation result is shown:  $P_0$  and  $P_1$  have the same  $\alpha$ , so the curve of  $\alpha_0$  and  $\alpha_1$  coincide.

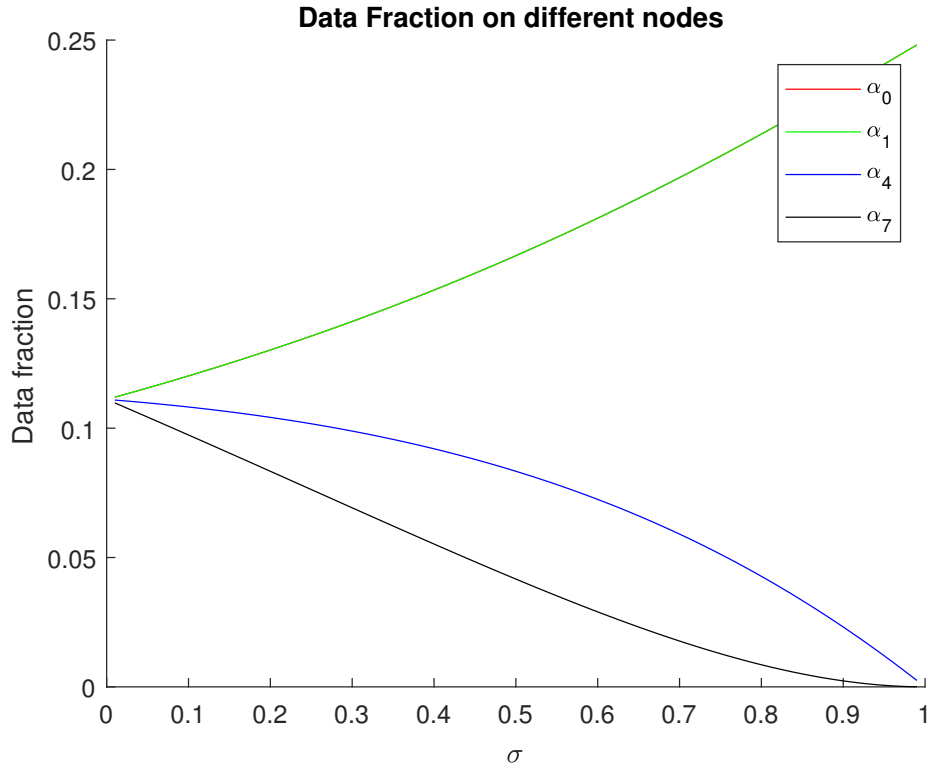


Figure 2.15: The data fraction simulation result of 3\*3 regular mesh and the data injection happens on the boundary  $P_0$

## Data Injection On The Inner Grid Processor

$L_1$  is on the inner grid processor  $P_0$  Fig. 2.16,

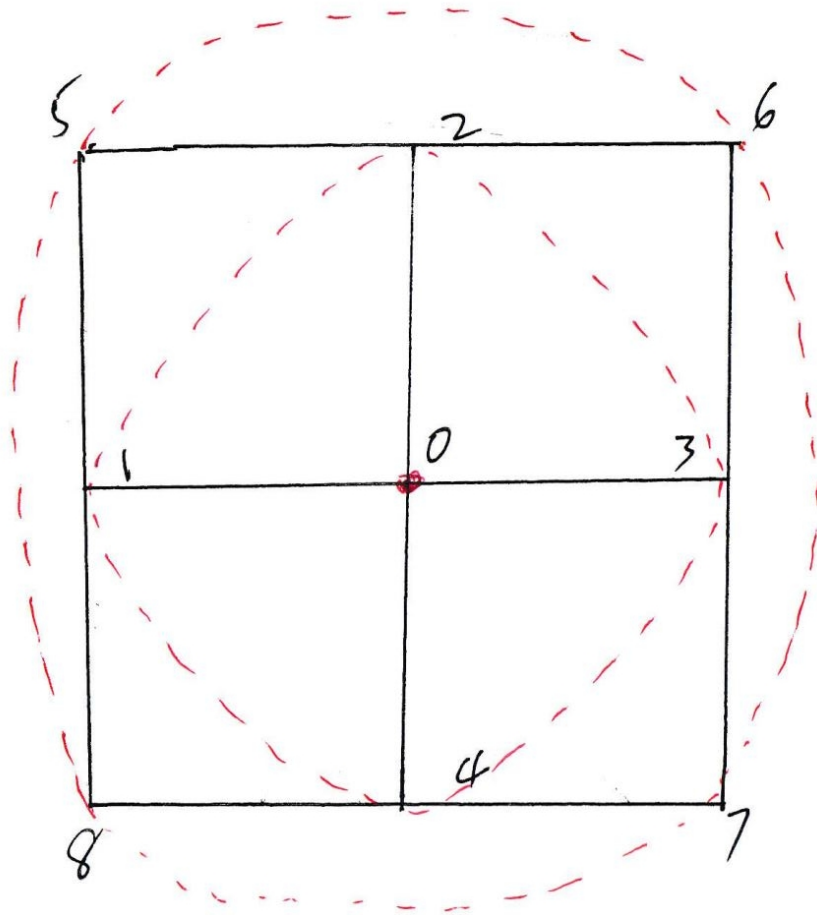


Figure 2.16: 3\*3 regular mesh. The data injection position is inner grid point  $P_0$

The timing diagram for this user case is illustrated as Fig. 2.17:

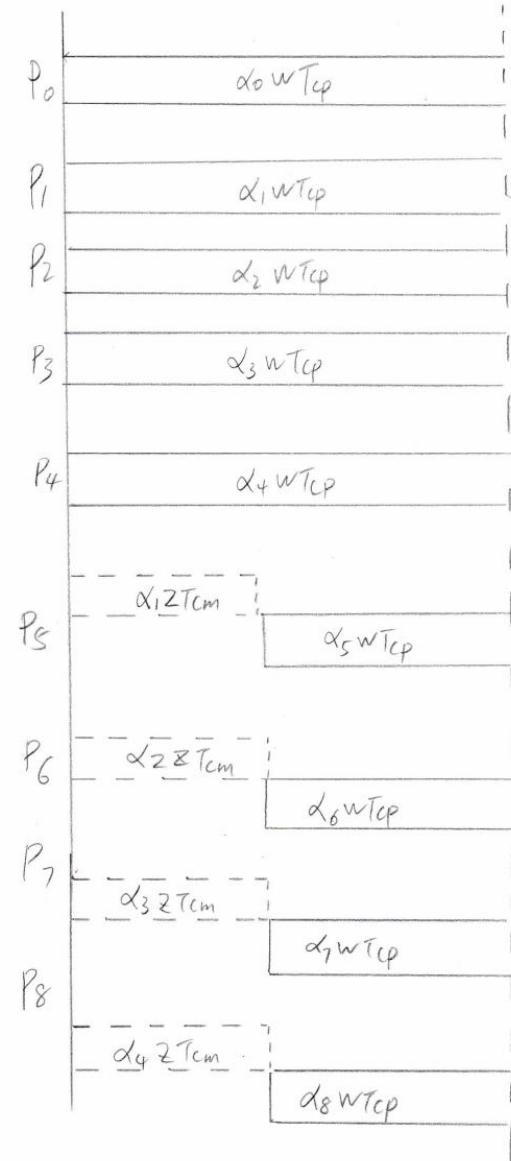


Figure 2.17: The timing diagram for 3\*3 regular mesh and the data injection is inner grid  $P_0$



The group of equations are:

$$\left\{ \begin{array}{ll} \alpha_0 \omega T_{cp} = T_{f,m} & (2.57) \\ \alpha_1 \omega T_{cp} = T_{f,m} & (2.58) \\ \alpha_2 \omega T_{cp} = T_{f,m} & (2.59) \\ \alpha_3 \omega T_{cp} = T_{f,m} & (2.60) \\ \alpha_4 \omega T_{cp} = T_{f,m} & (2.61) \\ \alpha_1 z T_{cm} + \alpha_5 \omega T_{cp} = T_{f,m} & (2.62) \\ \alpha_1 z T_{cm} + \alpha_6 \omega T_{cp} = T_{f,m} & (2.63) \\ \alpha_1 z T_{cm} + \alpha_7 \omega T_{cp} = T_{f,m} & (2.64) \\ \alpha_1 z T_{cm} + \alpha_8 \omega T_{cp} = T_{f,m} & (2.65) \\ \sigma = \frac{z T_{cm}}{\omega T_{cp}} & (2.66) \\ 0 < \sigma < 1 & (2.67) \\ 0 < \alpha_0 \quad \alpha_1 \quad \alpha_2 \quad \cdots \quad \alpha_8 < 1 & (2.68) \end{array} \right.$$

The matrix form is :

$$\begin{bmatrix} 1 & 4 & 4 \\ 1 & -1 & 0 \\ 0 & \sigma - 1 & 1 \end{bmatrix} \times \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_5 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad (2.69)$$

The simulation result for the inner grid is Fig. 2.18:  $P_0$  and  $P_1$  have the same  $\alpha$  value,so the curve of  $\alpha_0$  and  $\alpha_1$  coincide.

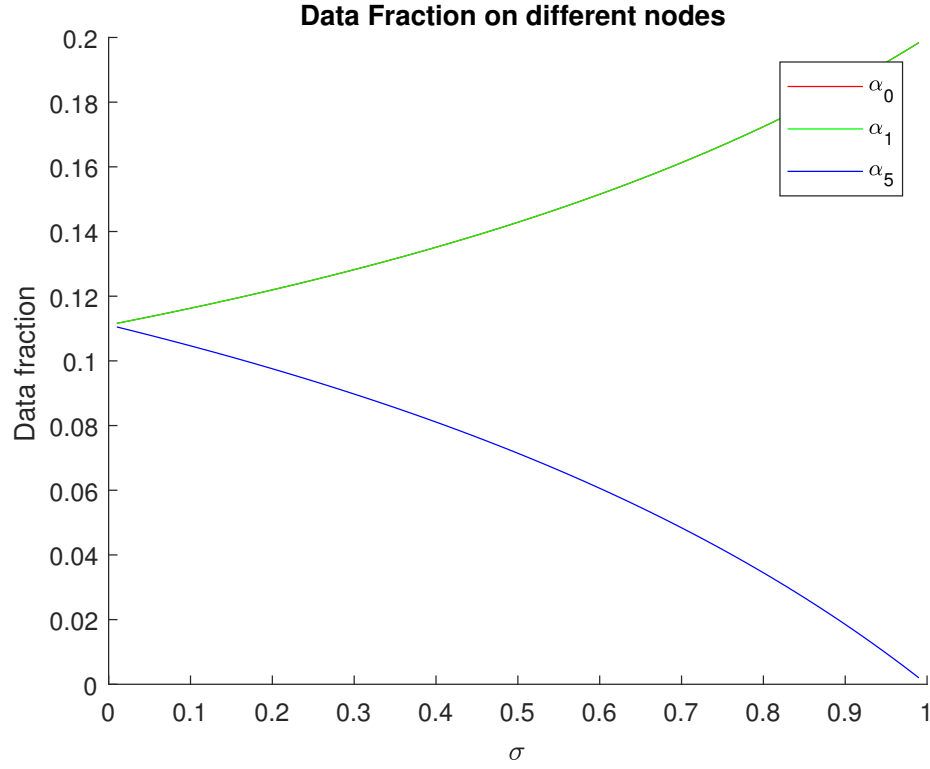


Figure 2.18: 3\*3 regular mesh. The data injection position is inner grid point  $P_0$

### 2.2.2 Torus mesh

According to the paper[6], there are three different torus mesh. In this paper, our intent is not to propose one model to "fit all" problems but rather to indicate one normal case. Other situation can be extended use the same rule. The toroidal network is also a regular complex in that each polygonal face has the same number of links and each node is connected to the same number of links.

We consider the toroidal network is Fig. 2.19 and Fig. 2.20.

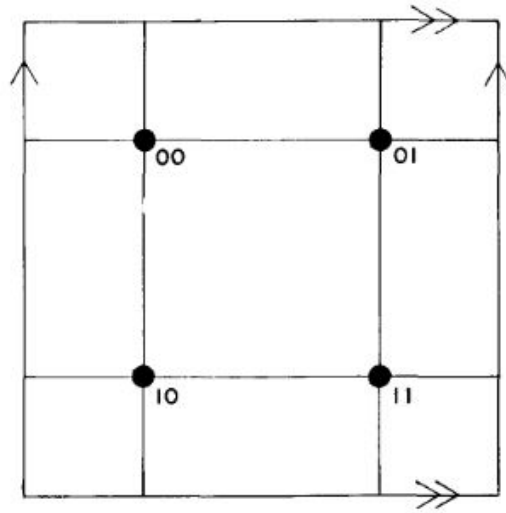


Figure 2.19: The rectangular toroidal network

Based on the rule, this question is transferred to that given a  $D_i$  and I find the number of instance.

- We assume the toroidal processor number is  $m * n$ , in other words, the width is  $m$  and the height is  $n$ .
- $L$  to present the load injection.

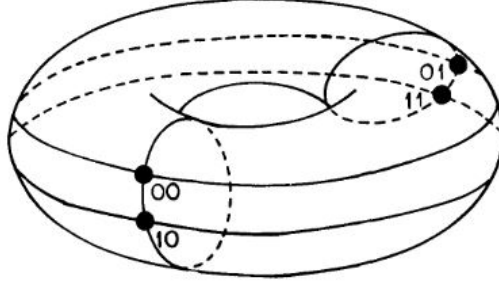


Figure 2.20: The rectangular toroidal network

- $L_x$  is the  $L$ 's  $X$  coordinate.
- $L_y$  is the  $L$ 's  $Y$  coordinate.
- $D_i$  to present the shortest Manhattan distance between the grid node  $i$  and  $L$ .
- $D_x$  is the  $X$  coordinate shortest Manhattan distance.
- $D_y$  is the  $Y$  coordinate shortest Manhattan distance.
- $N_x$  is the node  $i$ 's  $X$  coordinate.
- $N_y$  is the node  $i$ 's  $Y$  coordinate.

$$\left\{ \begin{array}{l} D = D_x + D_y \\ D_x = \min\{\|N_x - L_x\|, m - \|N_x - L_x\|\} \\ D_y = \min\{\|N_y - L_y\|, n - \|N_y - L_y\|\} \end{array} \right. \quad \begin{array}{l} (2.70) \\ (2.71) \\ (2.72) \end{array}$$

In the  $m * n$  ( $m = 6, n = 6$ ) toroidal networks,  $L$  happens on grid position  $(4, 2)$ . We calculate the  $D_i$  matrix Table 2.1 by breadth first search (**BFS**) algorithm.

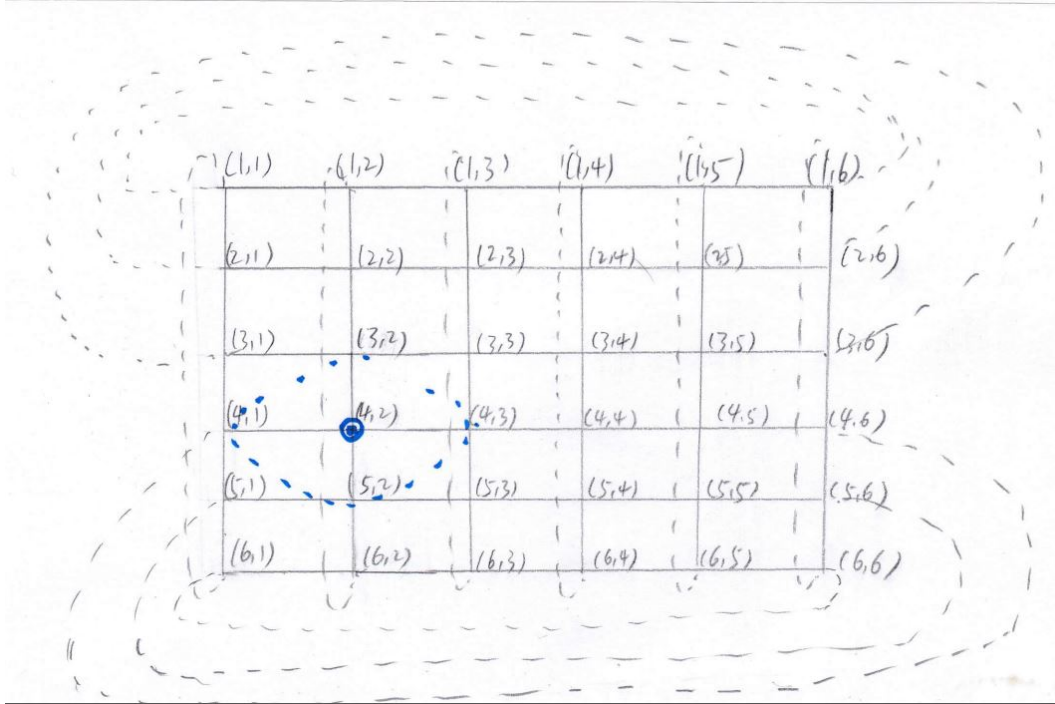


Figure 2.21: The  $m * n$  toroidal network and the data injection is  $P_{4,2}$

$D_i$	Number
0	1
1	4
2	8
3	10
4	8
5	4
6	1

Table 2.1: The processor number of various  $D_i$

And the  $D_i$  matrix as follow in table Table 2.1

The matrix closed-form is

$$\begin{bmatrix} 1 & 4 & 8 & 10 & 8 & 4 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma - 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & \sigma - 1 & \sigma & 1 & 0 & 0 & 0 \\ 0 & \sigma - 1 & \sigma & \sigma & 1 & 0 & 0 \\ 0 & \sigma - 1 & \sigma & \sigma & \sigma & 1 & 0 \\ 0 & \sigma - 1 & \sigma & \sigma & \sigma & \sigma & 1 \end{bmatrix} \times \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \\ \alpha_6 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (2.73)$$

The simulation result is :

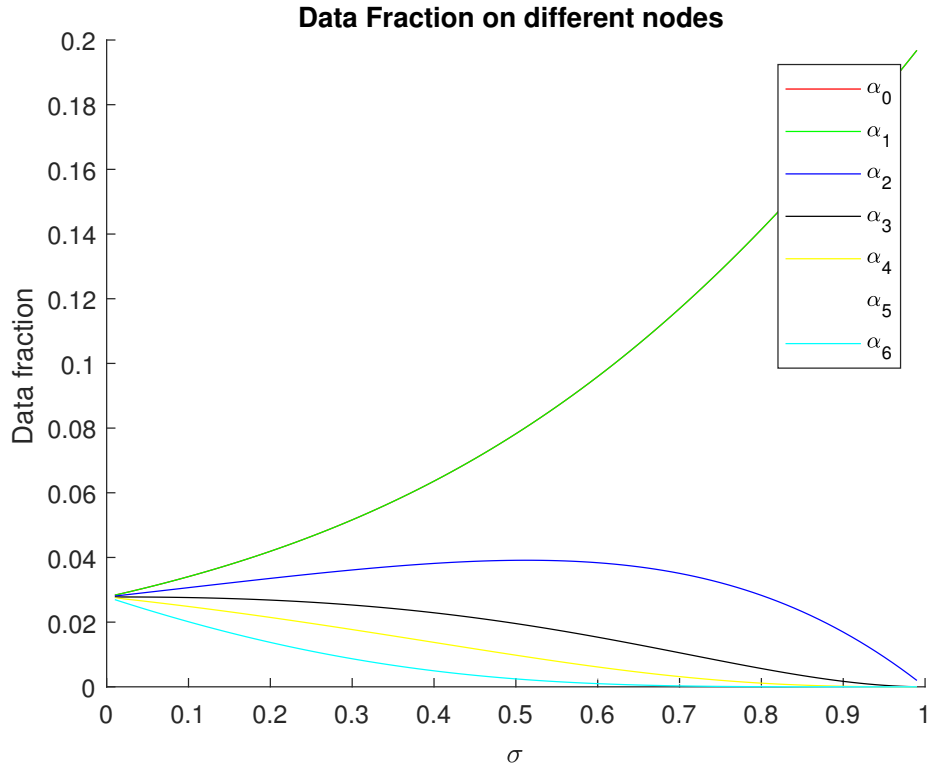


Figure 2.22: The data fraction deployed based on the radius value

From Fig. 2.22, we see as the  $\sigma$  value grows, more and more workload are assigned to the  $P_{4,2}$  and its one hop neighbors. That is, as the communication ability goes down, the economical method is to process the data locally.

### 2.2.3 General Case

$$\begin{bmatrix} 1 & m_1 & m_2 & \cdots & m_{n-2} & m_{n-1} & m_n \\ 1 & -1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & \sigma - 1 & 1 & \cdots & 0 & 0 & 0 \\ 0 & \sigma - 1 & \sigma & 1 & 0 & \cdots & 0 \\ 0 & \sigma - 1 & \sigma & \sigma & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \\ 0 & \sigma - 1 & \sigma & \cdots & \sigma & \sigma & 1 \end{bmatrix} \times \begin{bmatrix} \alpha_{l_0} \\ \alpha_{l_1} \\ \alpha_{l_2} \\ \alpha_{l_3} \\ \vdots \\ \alpha_{l_{n-1}} \\ \alpha_{l_n} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad (2.74)$$

The  $m_1, m_2, \dots, m_n$  are the number of processors on the  $level_1, level_2, \dots, level_n$ . Also, the  $\alpha_{l_0}, \alpha_{l_1}, \dots, \alpha_{l_n}$  are corresponding workload fraction.



## 2.3 Without Front End Scenario

### 2.3.1 Regular Mesh

#### Data Injection on Corner Processor

##### 2\*2 Regular Mesh

This subsection concerns the processors without front-end. Because of without front-end, the processors simultaneously receive the data and only start to process it as soon as each processor receives its entire load assignment.

We consider the timing diagram for Fig. 2.3, Fig. 2.6, Fig. 2.9 and so on. In addition, we also give the new closed-form matrix equations for the previous user cases.

Also, the rule also plays a dominate role in establish the mathematics model.

The timing diagram of Fig. 2.3 is shown:

$P_0$  starts to process the assigned workload and it starts to transfer the  $\alpha_1, \alpha_2$  and  $\alpha_3$  fraction workload after it totally receive its  $\alpha_0$  task. That is,  $P_1$  and  $P_2$  are idle until the  $L_1$  finish its data injection to  $P_0$ . The similar situation happens to  $P_1$  and  $P_2$  and they both starts to transmit the  $\alpha_3$  after they totally receive the appropriate workload. In other words,  $P_3$  has to wait until the previous two level processors obtains their own data.

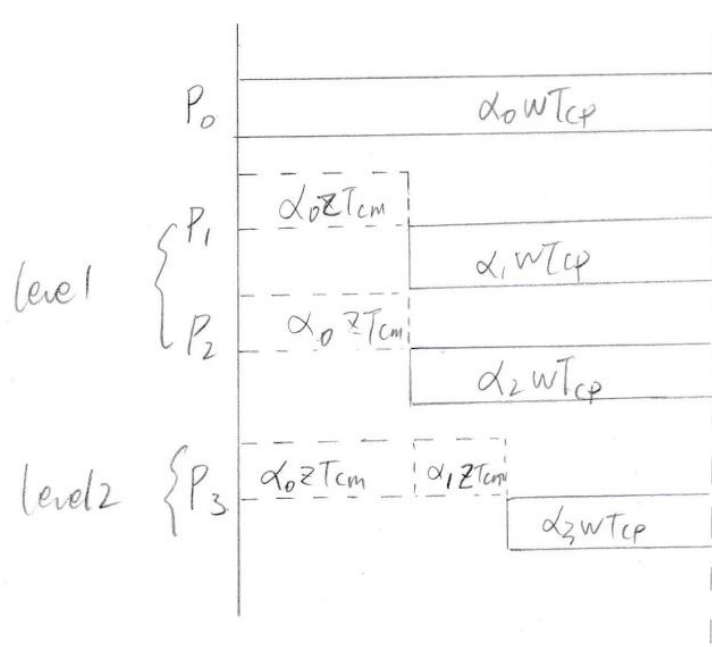


Figure 2.23: The timing diagram for 2\*2 regular mesh without front end.

The corresponding group of equations are as follows:

$$\left\{ \begin{array}{ll} \alpha_0 \omega T_{cp} = T_{f,m} & (2.75) \\ \alpha_1 z T_{cm} + \alpha_1 \omega T_{cp} = T_{f,m} & (2.76) \\ \alpha_2 z T_{cm} + \alpha_2 \omega T_{cp} = T_{f,m} & (2.77) \\ (\alpha_1 + \alpha_3) z T_{cm} + \alpha_3 \omega T_{cp} = T_{f,m} & (2.78) \\ \sigma = \frac{z T_{cm}}{\omega T_{cp}} & (2.79) \\ \alpha_0 + \alpha_1 + \alpha_2 + \alpha_3 = 1 & (2.80) \\ 0 < \sigma < 1 & (2.81) \\ 0 < \alpha_0, \alpha_1, \alpha_2, \alpha_3 < 1 & (2.82) \end{array} \right.$$

The group of equations imply that

$$\alpha_1 = \alpha_2$$

Further,the equations test and verify the rule again.

The matrix closed-form is presented as:

$$\begin{bmatrix} 1 & 2 & 1 \\ 1 & -(\sigma + 1) & 0 \\ 1 & -\sigma & -(\sigma + 1) \end{bmatrix} \times \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad (2.83)$$

The simulation result for Fig. 2.3 is provided in Fig. 2.24:

Fig. 2.24 explains that as the  $\sigma$  value grows up, the fraction is assigned to  $P_0$  increases and the fraction are distributed to  $level_1$  and  $level_2$  reduce. In other words, if the communication capability decreases, there are more data processed locally, which is reasonable. If the ability of the link decrease asymptotically equal to the processor computation capacity, there is only 10% data is deployed to the  $level_2$ . In addition, if the  $\sigma > 1$ , it means the link transmitting power is less than the processor's processing ability. In this scenario, keeping the data locally is more economical than transmitting it.

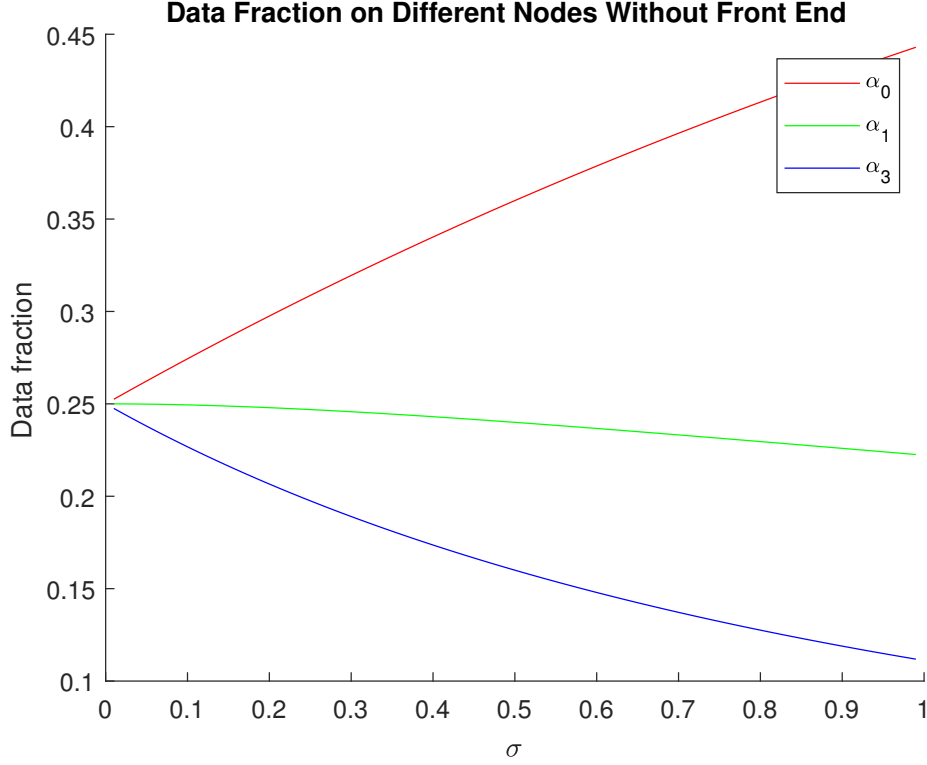


Figure 2.24: The data fraction deployed based on the radius value

### 2\*3 Regular Mesh

$P_0$  starts to process the assigned workload and it starts to transfer the  $\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5$  fraction workload after it totally receive its  $\alpha_0$  task. That is,  $P_1$  and  $P_2$  are idle until the  $L_1$  finish its data injection to  $P_0$ . According to the level 1, the similar situation happens to  $P_1$  and  $P_2$  and they both start to transmit the  $\alpha_3$  after they totally receive the appropriate workload. In other words,  $P_3$  has to wait until the previous three levels,  $level_0$ ,  $level_1$  and  $level_2$  processors obtain their own data.

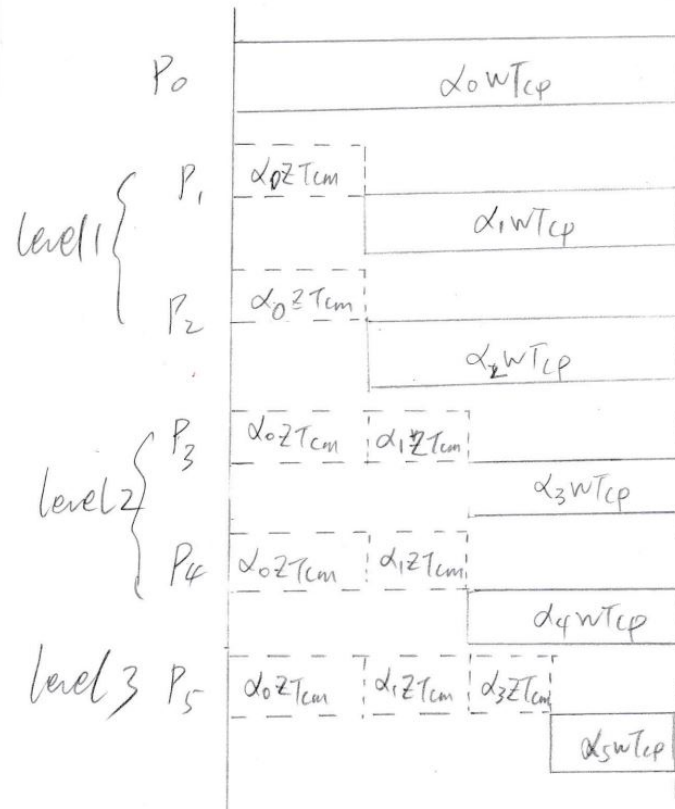


Figure 2.25: The timing diagram for 2\*3 regular mesh without front end.

In addition, the group of equations are as follows:

$$\left\{ \begin{array}{ll} \alpha_0 \omega T_{cp} = T_{f,m} & (2.84) \\ \alpha_1 z T_{cm} + \alpha_1 \omega T_{cp} = T_{f,m} & (2.85) \\ \alpha_2 z T_{cm} + \alpha_2 \omega T_{cp} = T_{f,m} & (2.86) \\ (\alpha_1 + \alpha_3) z T_{cm} + \alpha_3 \omega T_{cp} = T_{f,m} & (2.87) \\ (\alpha_1 + \alpha_4) z T_{cm} + \alpha_4 \omega T_{cp} = T_{f,m} & (2.88) \\ (\alpha_1 + \alpha_3 + \alpha_5) z T_{cm} + \alpha_5 \omega T_{cp} = T_{f,m} & (2.89) \\ \sigma = \frac{z T_{cm}}{\omega T_{cp}} & (2.90) \\ \alpha_0 + \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + \alpha_5 = 1 & (2.91) \end{array} \right.$$

$$\begin{bmatrix} 1 & 2 & 2 & 1 \\ 1 & -(\sigma + 1) & 0 & 0 \\ 1 & -\sigma & -(\sigma + 1) & 0 \\ 1 & -\sigma & -\sigma & -(\sigma + 1) \end{bmatrix} \times \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_3 \\ \alpha_5 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (2.92)$$

The speedup is

$$Speedup = \frac{T_{f,0}}{T_{f,n}} = \frac{\omega T_{cp}}{\alpha_0 \omega T_{cp}} = \frac{1}{\alpha_0}$$

The simulation result is:

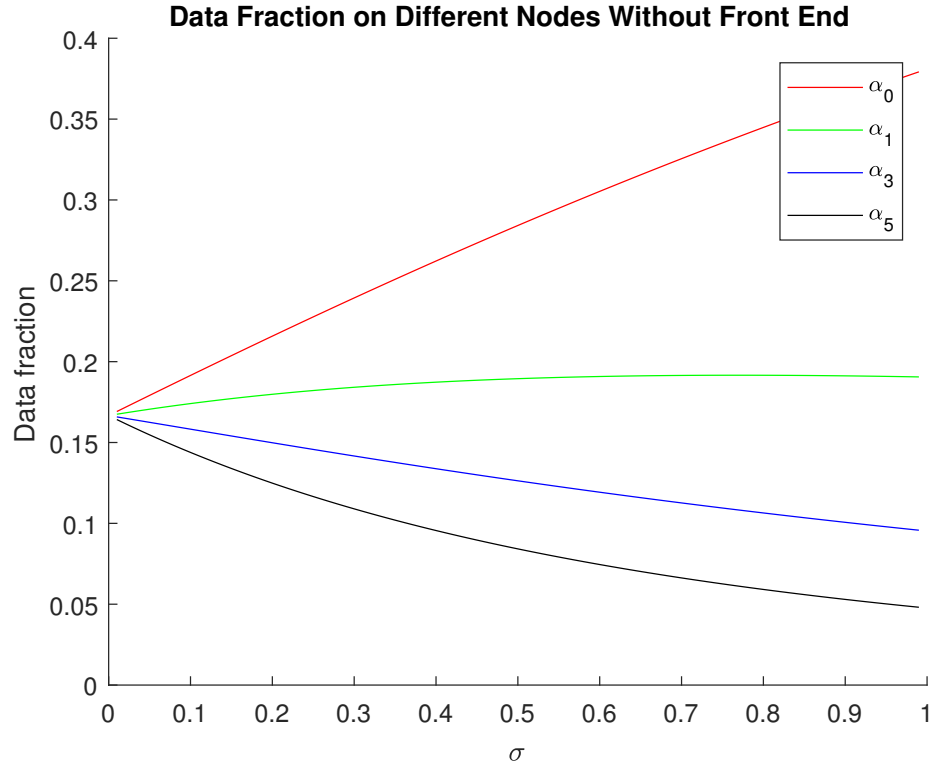


Figure 2.26: The data fraction deployed based on the radius value

## 2\*n Regular Mesh

Considering Fig. 2.9, the equations are demonstrated as follows:

$$\left\{ \begin{array}{ll}
\alpha_1 z T_{cm} + \alpha_1 \omega T_{cp} = T_{f,m} & (2.93) \\
\alpha_2 z T_{cm} + \alpha_2 \omega T_{cp} = T_{f,m} & (2.94) \\
(\alpha_1 + \alpha_3) z T_{cm} + \alpha_3 \omega T_{cp} = T_{f,m} & (2.95) \\
(\alpha_1 + \alpha_4) z T_{cm} + \alpha_4 \omega T_{cp} = T_{f,m} & (2.96) \\
(\alpha_1 + \alpha_3 + \alpha_5) z T_{cm} + \alpha_5 \omega T_{cp} = T_{f,m} & (2.97) \\
\vdots & (2.98) \\
(\alpha_1 + \alpha_3 + \cdots + \alpha_{2 \times n + 1}) z T_{cm} + \alpha_{2 \times n - 1} \omega T_{cp} = T_{f,m} & (2.99) \\
\sigma = \frac{z T_{cm}}{\omega T_{cp}} & (2.100) \\
0 < \sigma < 1 & (2.101) \\
0 < \alpha_0 \quad \alpha_1 \quad \alpha_3 \quad \cdots \quad \alpha_{2 \times n - 1} < 1 & (2.102)
\end{array} \right.$$

The matrix form for the group of equations are:



$$\begin{bmatrix}
1 & 2 & 2 & \cdots & 2 & 2 & 1 \\
1 & -(\sigma+1) & 0 & \cdots & 0 & 0 & 0 \\
1 & -\sigma & -(\sigma+1) & \cdots & 0 & 0 & 0 \\
1 & -\sigma & -\sigma & -(\sigma+1) & 0 & \cdots & 0 \\
1 & -\sigma & -\sigma & -\sigma & -(\sigma+1) & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \\
1 & -\sigma & -\sigma & \cdots & -\sigma & -\sigma & -(\sigma+1)
\end{bmatrix} \times \begin{bmatrix}
\alpha_0 \\
\alpha_1 \\
\alpha_3 \\
\alpha_5 \\
\vdots \\
\alpha_{2 \times n-3} \\
\alpha_{2 \times n-1}
\end{bmatrix} = \begin{bmatrix}
1 \\
0 \\
0 \\
0 \\
0 \\
\vdots \\
0
\end{bmatrix} \quad (2.103)$$

### m\*n Regular Mesh

Referring to Fig. 2.12, we utilize  $\sigma^*$  to present the  $-(\sigma+1)$ . The matrix closed-form is:

$$\begin{bmatrix}
1 & 2 & 3 & 4 & 5 & 4 & 3 & 2 & 1 \\
1 & \sigma^* & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & -\sigma & \sigma^* & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & -\sigma & -\sigma & \sigma^* & 0 & 0 & 0 & 0 & 0 \\
1 & -\sigma & -\sigma & -\sigma & \sigma^* & 0 & 0 & 0 & 0 \\
1 & -\sigma & -\sigma & -\sigma & -\sigma & \sigma^* & 0 & 0 & 0 \\
1 & -\sigma & -\sigma & -\sigma & -\sigma & -\sigma & \sigma^* & 0 & 0 \\
1 & -\sigma & -\sigma & -\sigma & -\sigma & -\sigma & -\sigma & \sigma^* & 0 \\
1 & -\sigma & -\sigma & -\sigma & -\sigma & -\sigma & -\sigma & -\sigma & \sigma^*
\end{bmatrix} \times \begin{bmatrix}
\alpha_0 \\
\alpha_1 \\
\alpha_3 \\
\alpha_6 \\
\alpha_{10} \\
\alpha_{15} \\
\alpha_{19} \\
\alpha_{22} \\
\alpha_{24}
\end{bmatrix} = \begin{bmatrix}
1 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0
\end{bmatrix} \quad (2.104)$$

## Data Injection on Boundary Processor

Fig. 2.13 shows an example of boundary processor  $P_0$  receiving the  $L_1$ . The timing diagram for Fig. 2.13 is Fig. 2.27.

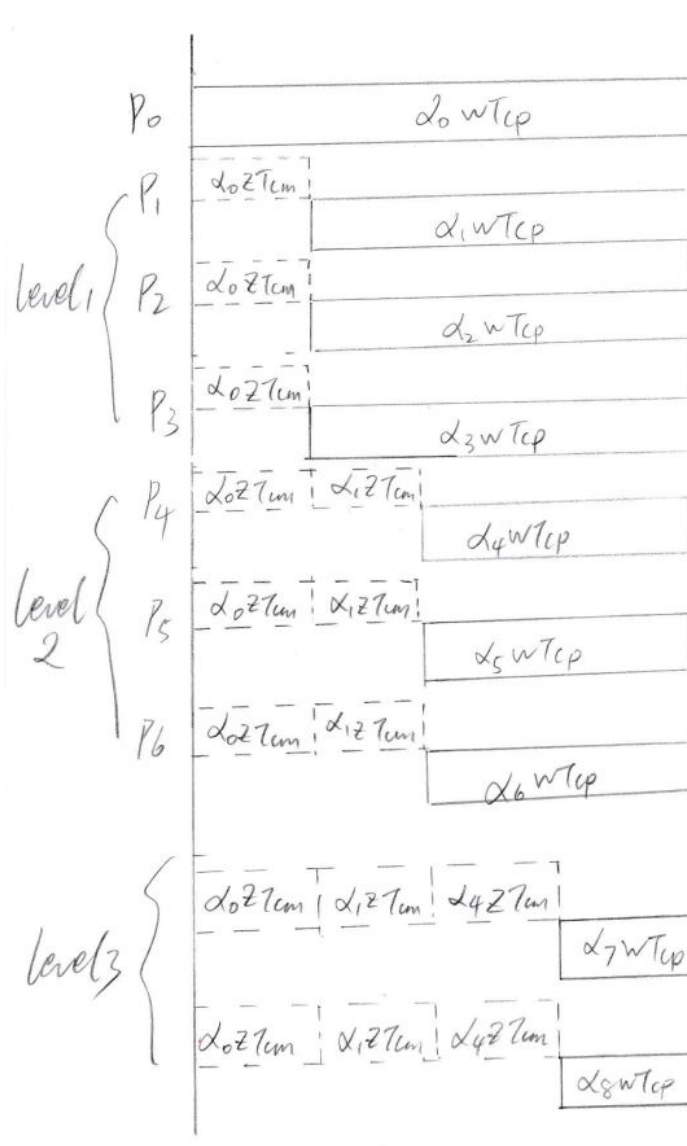


Figure 2.27: The timing diagram for 3\*3 boundary data injection on  $P_0$

$$\left\{ \begin{array}{ll} \alpha_0 \omega T_{cp} = T_{f,m} & (2.105) \\ \alpha_1 z T_{cm} + \alpha_1 \omega T_{cp} = T_{f,m} & (2.106) \\ \alpha_2 z T_{cm} + \alpha_2 \omega T_{cp} = T_{f,m} & (2.107) \\ \alpha_3 z T_{cm} + \alpha_3 \omega T_{cp} = T_{f,m} & (2.108) \\ (\alpha_1 + \alpha_4) z T_{cm} + \alpha_4 \omega T_{cp} = T_{f,m} & (2.109) \\ (\alpha_2 + \alpha_5) z T_{cm} + \alpha_5 \omega T_{cp} = T_{f,m} & (2.110) \\ (\alpha_3 + \alpha_6) z T_{cm} + \alpha_6 \omega T_{cp} = T_{f,m} & (2.111) \\ (\alpha_1 + \alpha_4 + \alpha_7) z T_{cm} + \alpha_7 \omega T_{cp} = T_{f,m} & (2.112) \\ (\alpha_1 + \alpha_4 + \alpha_8) z T_{cm} + \alpha_8 \omega T_{cp} = T_{f,m} & (2.113) \\ \sigma = \frac{z T_{cm}}{\omega T_{cp}} & (2.114) \\ 0 < \sigma < 1 & (2.115) \\ 0 < \alpha_0 \quad \alpha_1 \quad \alpha_3 \quad \alpha_4 \quad \alpha_5 \quad \alpha_6 \quad \alpha_7 \quad \alpha_8 < 1 & (2.116) \end{array} \right.$$

$$\begin{bmatrix} 1 & 3 & 3 & 2 \\ 1 & -(\sigma + 1) & 0 & 0 \\ 1 & -\sigma & -(\sigma + 1) & 0 \\ 1 & -\sigma & -\sigma & -(\sigma + 1) \end{bmatrix} \times \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_4 \\ \alpha_7 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (2.117)$$

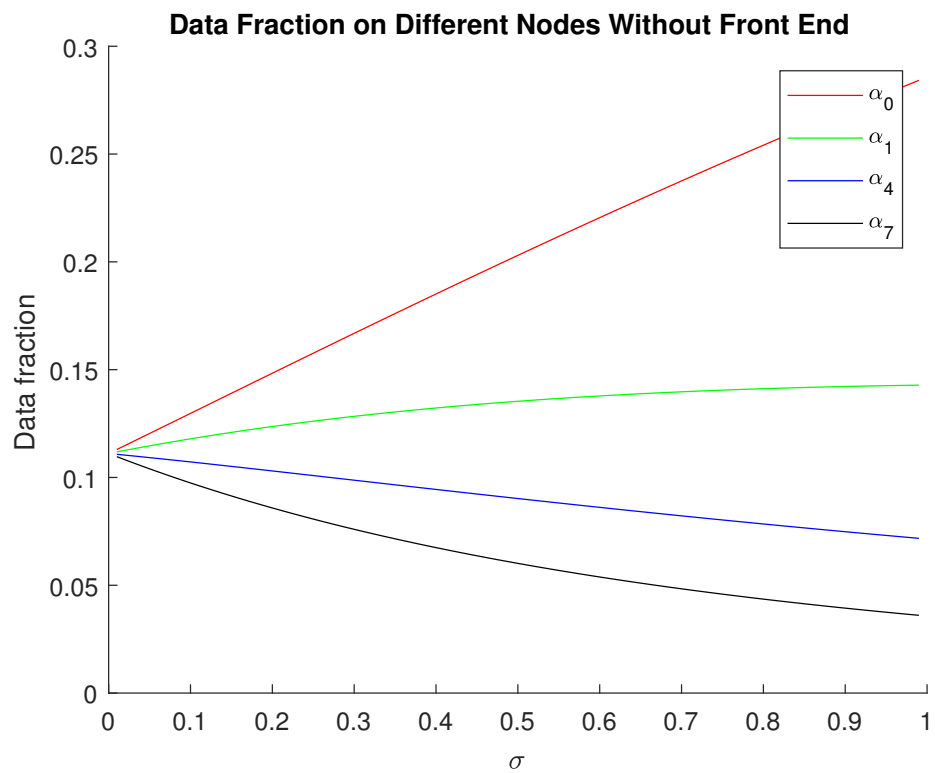


Figure 2.28: The fraction curve for 3\*3 boundary data injection on  $P_0$

## Data Injection on The Inner Grid Processor

The equations are:

$$\left\{ \begin{array}{ll} \alpha_0 \omega T_{cp} = T_{f,m} & (2.118) \\ \alpha_1 z T_{cm} + \alpha_1 \omega T_{cp} = T_{f,m} & (2.119) \\ \alpha_2 z T_{cm} + \alpha_2 \omega T_{cp} = T_{f,m} & (2.120) \\ \alpha_3 z T_{cm} + \alpha_3 \omega T_{cp} = T_{f,m} & (2.121) \\ \alpha_4 z T_{cm} + \alpha_4 \omega T_{cp} = T_{f,m} & (2.122) \\ (\alpha_1 + \alpha_5) z T_{cm} + \alpha_5 \omega T_{cp} = T_{f,m} & (2.123) \\ (\alpha_2 + \alpha_6) z T_{cm} + \alpha_6 \omega T_{cp} = T_{f,m} & (2.124) \\ (\alpha_3 + \alpha_7) z T_{cm} + \alpha_7 \omega T_{cp} = T_{f,m} & (2.125) \\ (\alpha_4 + \alpha_8) z T_{cm} + \alpha_8 \omega T_{cp} = T_{f,m} & (2.126) \\ \sigma = \frac{z T_{cm}}{\omega T_{cp}} & (2.127) \\ 0 < \sigma < 1 & (2.128) \\ 0 < \alpha_0 \quad \alpha_1 \quad \alpha_3 \quad \alpha_4 \quad \alpha_5 \quad \alpha_6 \quad \alpha_7 \quad \alpha_8 < 1 & (2.129) \end{array} \right.$$

The matrix closed-form is:

$$\begin{bmatrix} 1 & 4 & 4 \\ 1 & -(\sigma + 1) & 0 \\ 1 & -\sigma & -(\sigma + 1) \end{bmatrix} \times \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_5 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad (2.130)$$

The simulation result shows:

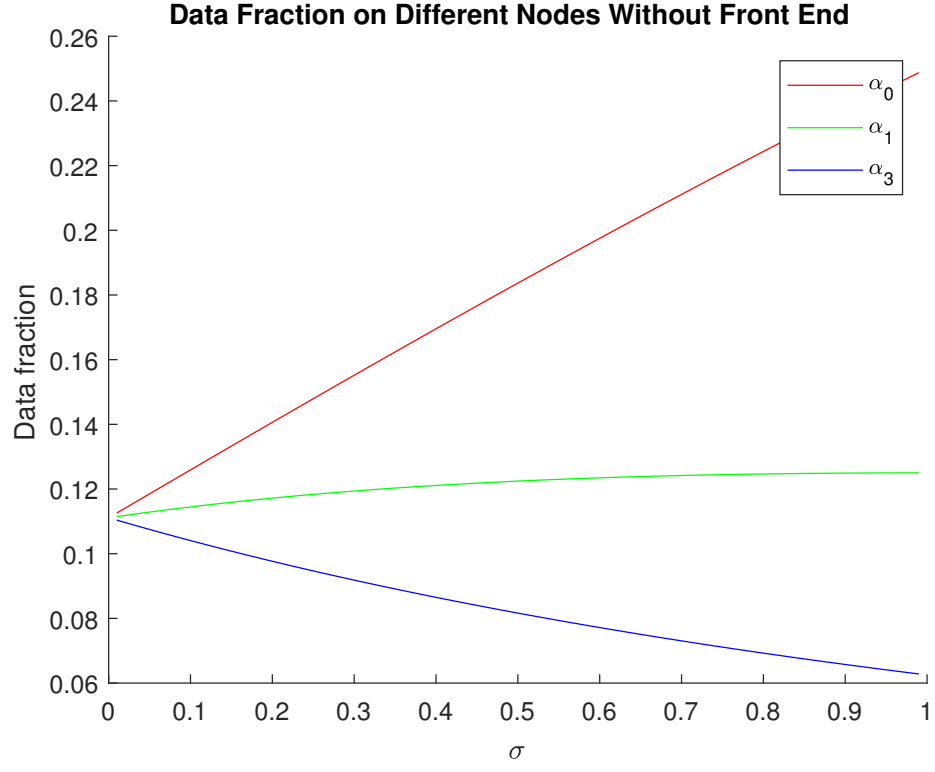


Figure 2.29: The timing diagram for 3\*3 inner grid injection  $P_0$

### 2.3.2 Toroidal Mesh

We utilize the  $\sigma^*$  to present  $-(\sigma + 1)$ . The matrix closed-form is:

$$\begin{bmatrix} 1 & 4 & 8 & 10 & 8 & 4 & 1 \\ 1 & \sigma^* & 0 & 0 & 0 & 0 & 0 \\ 1 & -\sigma & \sigma^* & 0 & 0 & 0 & 0 \\ 1 & -\sigma & -\sigma & -\sigma^* & 0 & 0 & 0 \\ 1 & -\sigma & -\sigma & -\sigma & \sigma^* & 0 & 0 \\ 1 & -\sigma & -\sigma & -\sigma & -\sigma & \sigma^* & 0 \\ 1 & -\sigma & -\sigma & -\sigma & -\sigma & \sigma & \sigma^* \end{bmatrix} \times \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \\ \alpha_6 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (2.131)$$

The simulation result is :

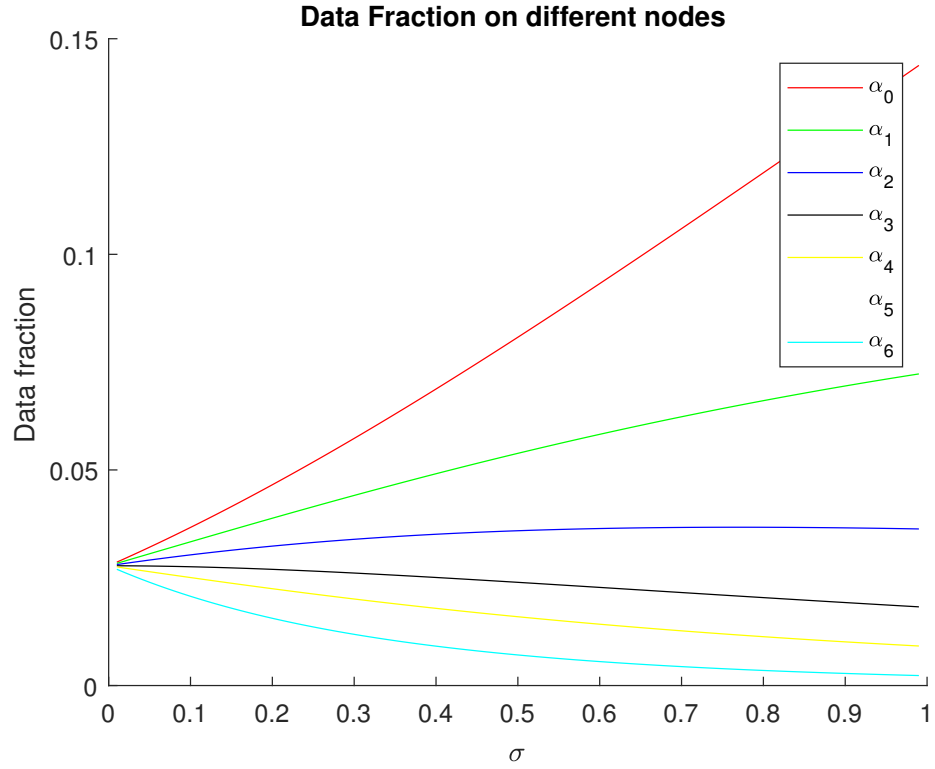


Figure 2.30: The data fraction deployed based on the radius value

### 2.3.3 General Case

$$\begin{bmatrix}
 1 & m_1 & m_2 & \cdots & m_{n-2} & m_{n-1} & m_n \\
 1 & -(\sigma+1) & 0 & \cdots & 0 & 0 & 0 \\
 1 & -\sigma & -(\sigma+1) & \cdots & 0 & 0 & 0 \\
 1 & -\sigma & -\sigma & -(\sigma+1) & 0 & \cdots & 0 \\
 1 & -\sigma & -\sigma & -\sigma & -(\sigma+1) & 0 & 0 \\
 \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \\
 1 & -\sigma & -\sigma & \cdots & -\sigma & -\sigma & -(\sigma+1)
 \end{bmatrix} \times \begin{bmatrix} \alpha_{l_0} \\ \alpha_{l_1} \\ \alpha_{l_2} \\ \alpha_{l_3} \\ \vdots \\ \alpha_{l_{n-1}} \\ \alpha_{l_n} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

(2.132)

The  $m_1, m_2, \dots, m_n$  are the number of processors on the  $level_1, level_2, \dots, level_n$ .

Also, the  $\alpha_{l_0}, \alpha_{l_1}, \dots, \alpha_{l_n}$  are corresponding workload fraction.



## 2.4 Comparing Result Between Front End Processor and Without Front End Processor

In the legend of figures, we use  $F$  to present the front-end situation and use  $NF$  to presents the without front-end processors setting,for example  $F\alpha_0$  means the  $\alpha_0$  data fraction deployed to  $P_0$ ,if the processor has front-end. $NF\alpha_0$  means the  $\alpha_0$  data fraction deployed to  $P_0$ ,if the processor is without front-end setting.

### 2.4.1 Regular Mesh

#### On the Corner Processor

Fig. 2.31 says that  $P_0$  takes more assigned task in without front-end scenario than front-end processor situation.As the  $\sigma$  value goes up,the fractions are deployed to the deeper levels dropping down.In the limit condition,for example, $\sigma = 1$ , there is no data transmitted to  $P_3$  in the front-end assumption, yet in the without front-end situation, there is still about 10% data fraction are communicated to  $P_3$ .

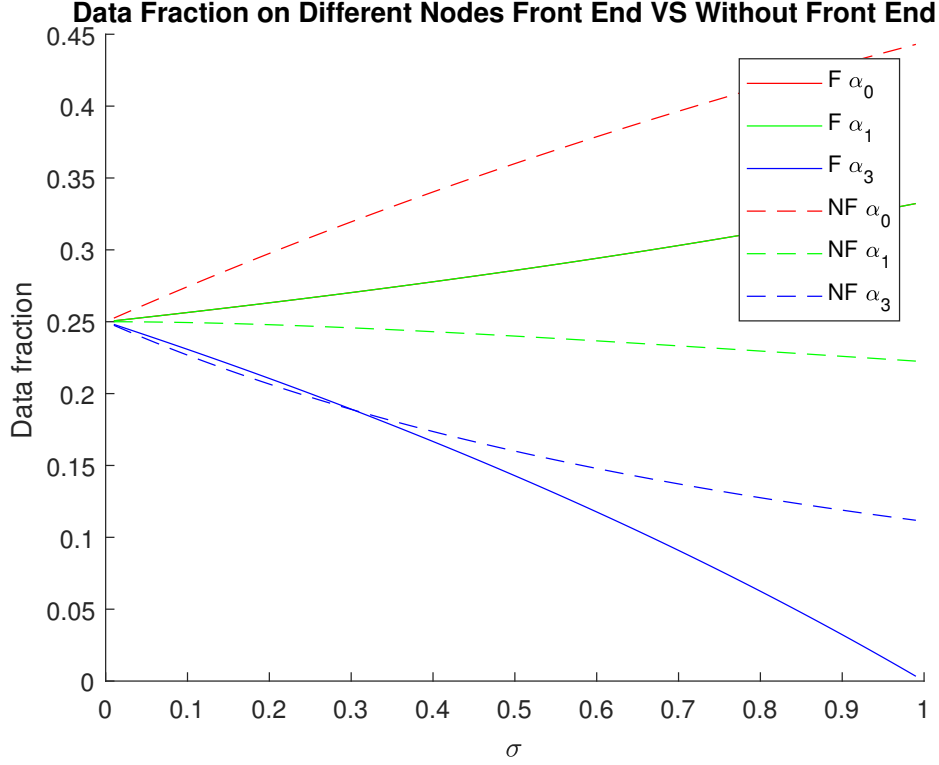


Figure 2.31: The comparing result between front-end processor with without front-end processor in 2\*2 regular mesh

Fig. 2.32 says that  $P_0$  takes more assigned task in without front-end scenario than front-end processor situation. As the  $\sigma$  value goes up, the fractions are deployed to the deeper levels dropping down. In the limit condition, for example,  $\sigma = 1$ , there is no data transmitted to  $level_3$ , that is,  $P_5$  in the front-end assumption. Yet in the without front-end situation, there is still about 5% data fraction is communicated to  $P_5$ .

Fig. 2.33 says that  $P_0$  takes more assigned task in without front-end scenario than front-end processor situation. As the  $\sigma$  value goes up, the fractions

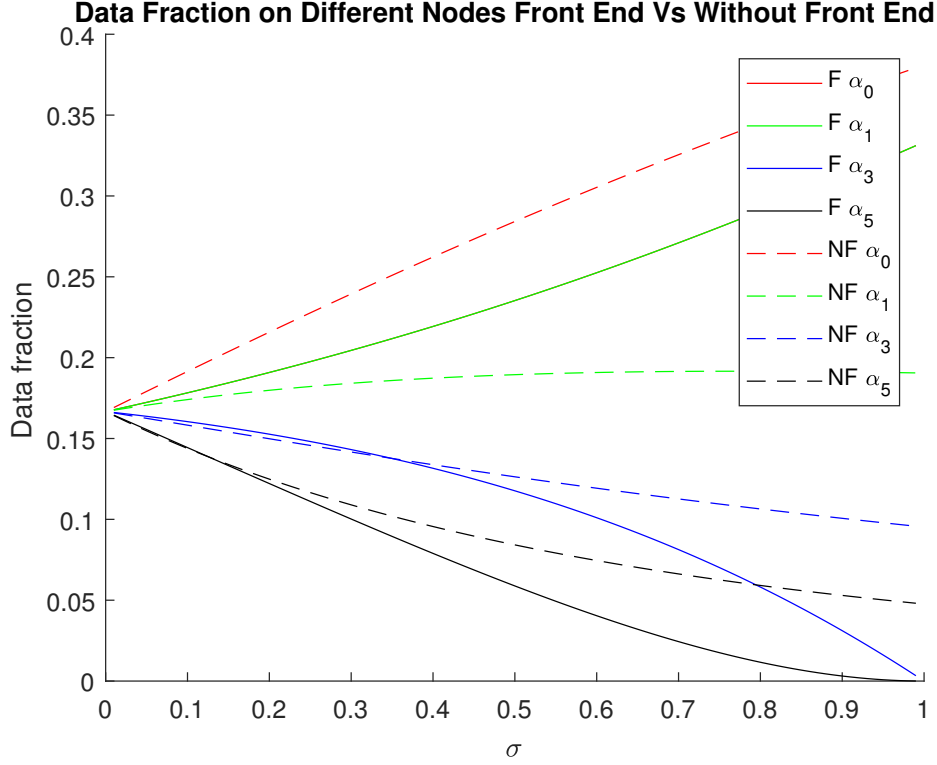


Figure 2.32: The comparing result between front-end processor with without front-end processor in 2\*4 regular mesh

are deployed to the deeper levels dropping down. In the limit condition, for example,  $\sigma = 1$ , there is no data transmitted to *level*<sub>3</sub>, that is,  $P_7$  and  $P_8$  in the front-end assumption. Yet in the without front-end situation, there is still about 5% data fraction is communicated to  $P_7$  and  $P_8$ .

Fig. 2.34 says that  $P_0$  takes more assigned task in without front-end scenario than front-end processor situation. As the  $\sigma$  value goes up, the fractions are deployed to the deeper levels dropping down. In the limit condition, for example,  $\sigma = 1$ , there is no data transmitted to *level*<sub>2</sub>, that is,  $P_5, P_6, P_7$  and  $P_8$

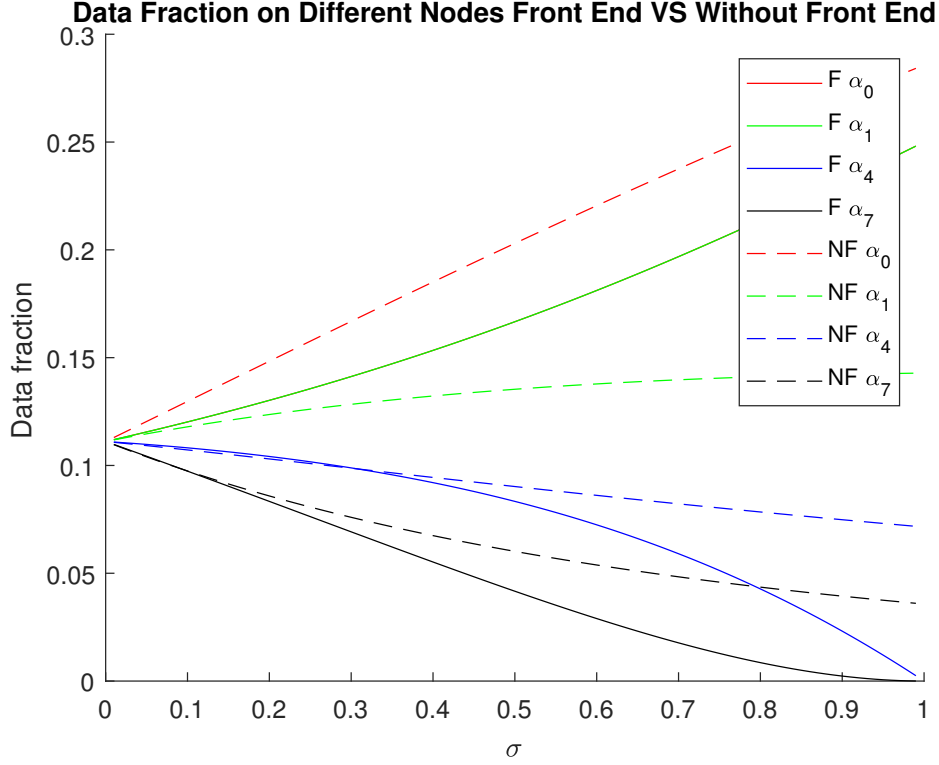


Figure 2.33: The comparing result between front-end processor with without front-end processor in 3\*3 regular mesh injection on boundary processor

in the front-end assumption. Yet in the without front-end situation, there is still about 5% data fraction is communicated to  $P_5, P_6, P_7$  and  $P_8$ .

Comparing with Fig. 2.33,  $P_0$  takes less workload in inner grid position than boundary data injection. The reason is there are 4 neighbor processors on the  $level_1$ , yet there is only three processors on  $level_1$  in boundary plan. In other words, the  $level_1$  plays a critical role in data deployed.

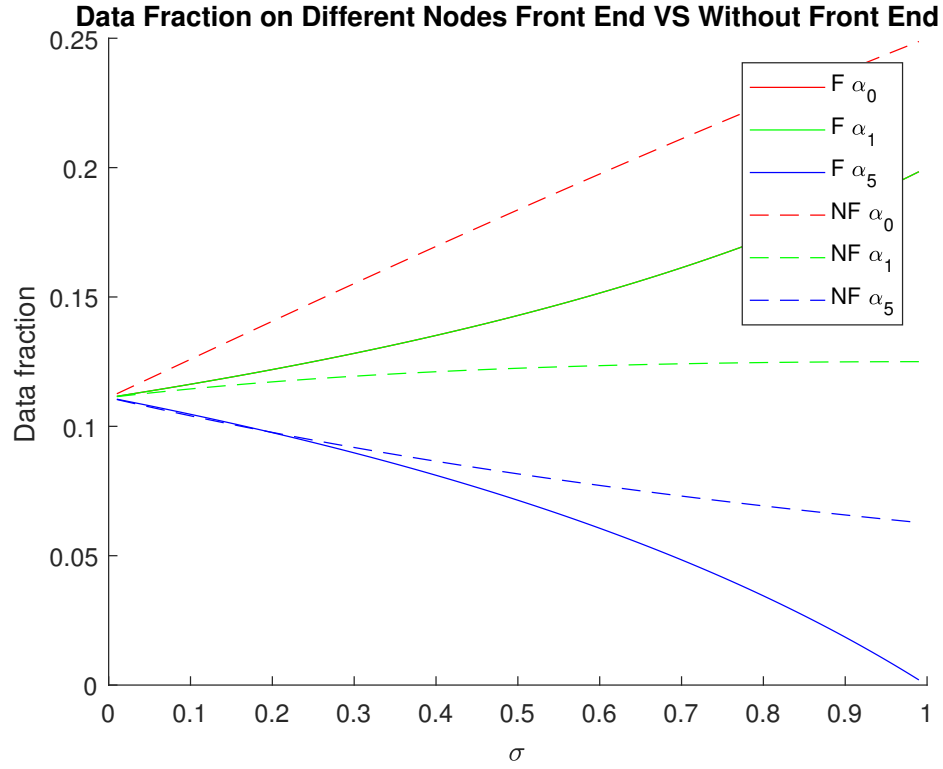


Figure 2.34: The comparing result between front-end processor with without front-end processor in 3\*3 regular mesh injection on inner grid processor

## 2.4.2 Toroidal

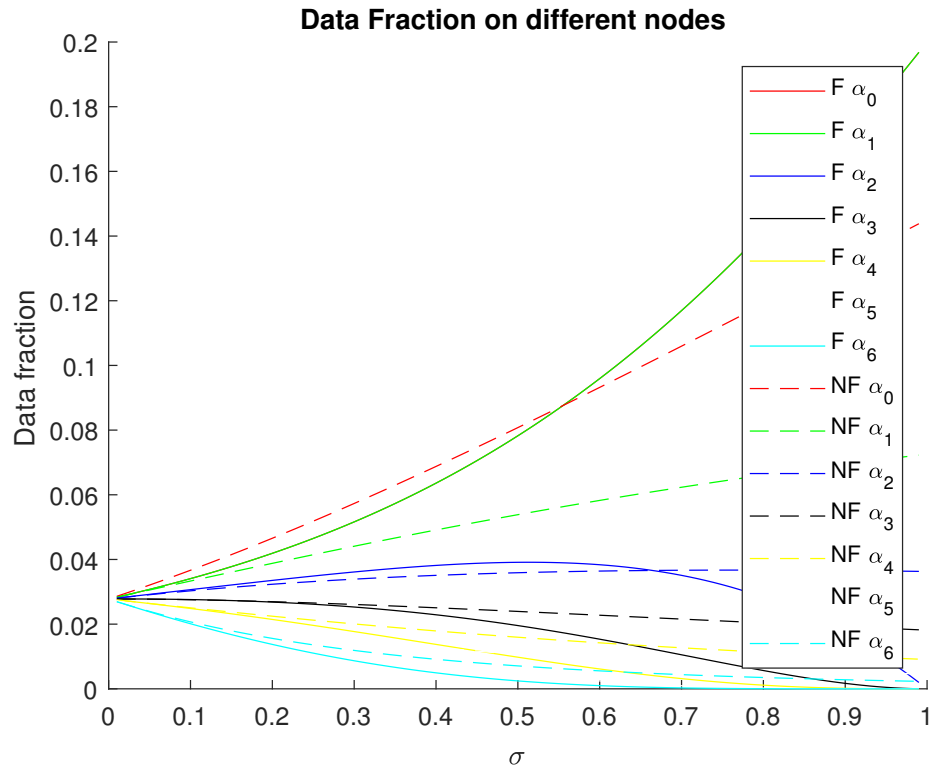


Figure 2.35: The comparing result between front-end processor with without front-end processor in 6\*6 regular mesh injection on inner grid processor

## **Chapter 3**

### **Sensitivity Analysis**

## Chapter 4

# Multi-source Assignment Utilizing Voronoi Diagram



## Chapter 5

# Multi-source Re-assignment Using Optimal Mass Transportation

## Chapter 6

## Conclusions

## Bibliography

- [1] M. Moges, D. Yu, and T. G. Robertazzi, “Grid scheduling divisible loads from two sources,” *Computers & Mathematics with Applications*, vol. 58, no. 6, pp. 1081–1092, 2009.
- [2] T. G. Robertazzi, “Processor equivalence for daisy chain load sharing processors,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 29, no. 4, pp. 1216–1221, 1993.
- [3] X. Liu, H. Zhao, and X. Li, “1scheduling divisible workloads from multiple sources in linear daisy chain networks.”
- [4] C. F. Gamboa and T. Robertazzi, “Simple performance bounds for multicore and parallel channel systems,” *Parallel Processing Letters*, vol. 21, no. 04, pp. 439–460, 2011.
- [5] V. Bharadwaj, D. Ghose, and T. G. Robertazzi, “Divisible load theory: A new paradigm for load scheduling in distributed systems,” *Cluster Computing*, vol. 6, no. 1, pp. 7–17, 2003.
- [6] T. G. Robertazzi, “Toroidal networks,” *IEEE Communications Magazine*, vol. 26, no. 6, pp. 45–50, 1988.