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: Rigid Body

To my Parents and all loving ones

Table of Contents

List of Figures

List of Tables

Acknowledgements

Chapter 1

Introduction

[?, ?]

Chapter 2

Closed-form Processor Equivalence

In this chapter,we mainly discuss the closed-form processor equivalence[1] problem in the grid network of regular mesh ,torus mesh and further I will extend the technique to more general mesh.This chapter is organized as follows.Section 1, I will give a problem introduction, notation and definitions.Section 2, I will consider about the unit core with front-end. In this kind of situation, we will investigate regular mesh and torus mesh closed-form processor equivalence formula.Also,I will give various closed-form formula, if the data injection position is on different type of position,such as on the corner unit core, the boundary unit core and inner grid unit processor.Section 3, I will consider the similar scenarios if the unit core doesn't contain the front-end.In section 4, I will 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 unit core.

2.1 Problem Description

The situation we are considering involves a regular mesh or torus mesh of m*n processors as shown in Fig. 2.12 Fig. 2.2. We refer to the $m * n$ processor as P_0,P_1,P_2,$\cdots$,P_{m*n-1}.

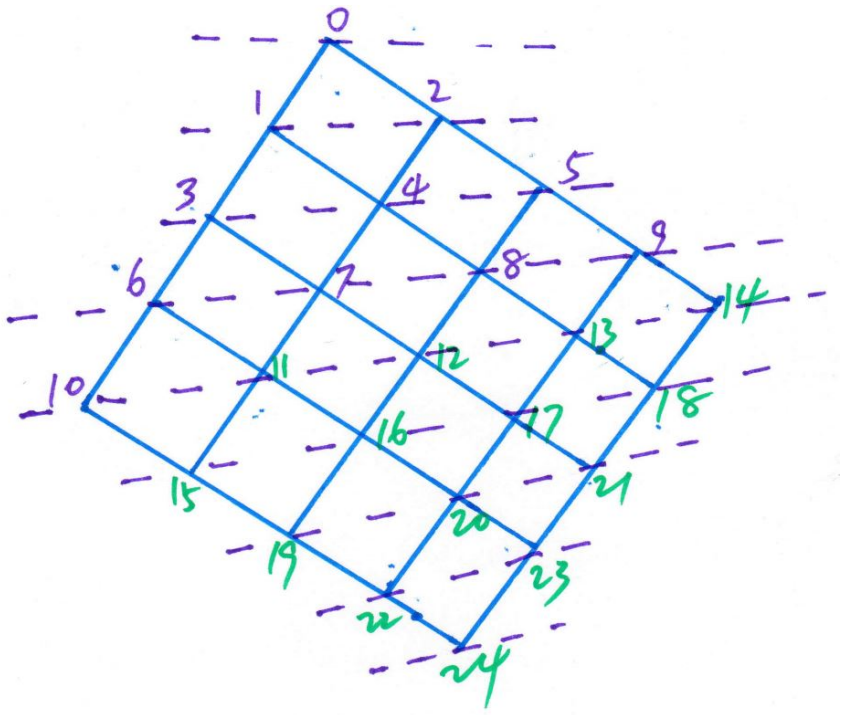

Figure 2.1: m*n regular mesh(m = 5, n = 5)

The workloads are denoted as $\mathbf{L_1}$,$\mathbf{L_2}$,$\cdots$,$\mathbf{L_k}$. We consider two situations, such as all the processors contain the front-end and without front-end.
In the front-end assumption, the communication and the computation can be executed simultaneously.In addition, we can assume that the network environment is homogeneous, in other words,all the processors contains the same computation capacity, and the link speeds between any two unit cores are

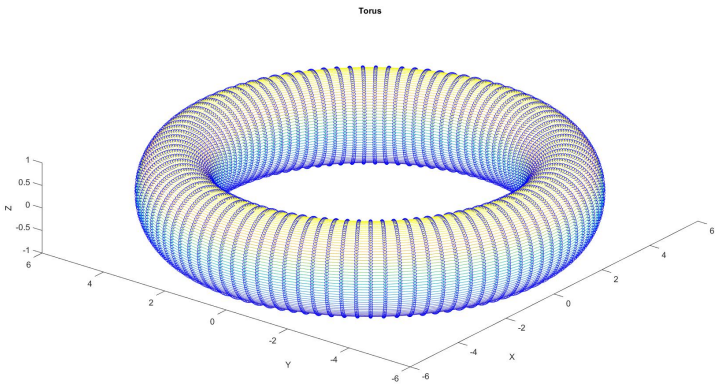

Figure 2.2: torus mesh with grid unit cores

identical.

In the without front-end scenario,each processor has to wait for obtaining the appointed data fraction first and then it can start to transmit data.

Thus, the problem becomes how to partition and schedule the workloads amongst the processors to get the minimum finish time.

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.

In this section,I will try to utilize the group of recursive equations to show the closed form relationship with.

The following notations and definitions are utilized:

- P_i: the ith processor. i should be from 0 to $m * n - 1$.
- L_i: the ith work load. i should be from 1 to k.

- α_0: The load fraction assigned to the root processor.
- α_i: The load fraction assigned to the ith processor.
- ω_i: The inverse computing speed on the ith processor.
- ω_{eq}: The inverse computing speed on an equivalent node collapsed from a regular mesh.
- z_i: The inverse link speed on the ith link.
- T_{cp}: Computing intensity constant.The entire load can be processed in $\omega_i T_{cp}$ on the ith processor.
- T_{cm}: Communication intensity constant.The entire load can be transmitted in $z_i T_{cm}$ seconds over the ith 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 $\omega_0 T_{cp}$.
- $\sum_{i=1}^{k} L_i = 1$
- $\sum_{i=0}^{m*n-1} \alpha_i = 1$

2.2 With Front End Scenario

In the front end unit core processors which will compute and transfer the data load simultaneously.Considering about the $2*2$,$2*n$,$3*n$ regular mesh as

example to deduct the general case $m*n$ closed form for a single load injection situation on corner,boundary and inner grid point.

2.2.1 Regular Mesh

2.2.2 Data Injection on Corner

2*2 regular mesh

The L_1 position on the corner unit core P_0 Fig. 2.3. The whole workload are processed by four processors P_0,P_1,P_2,P_3.

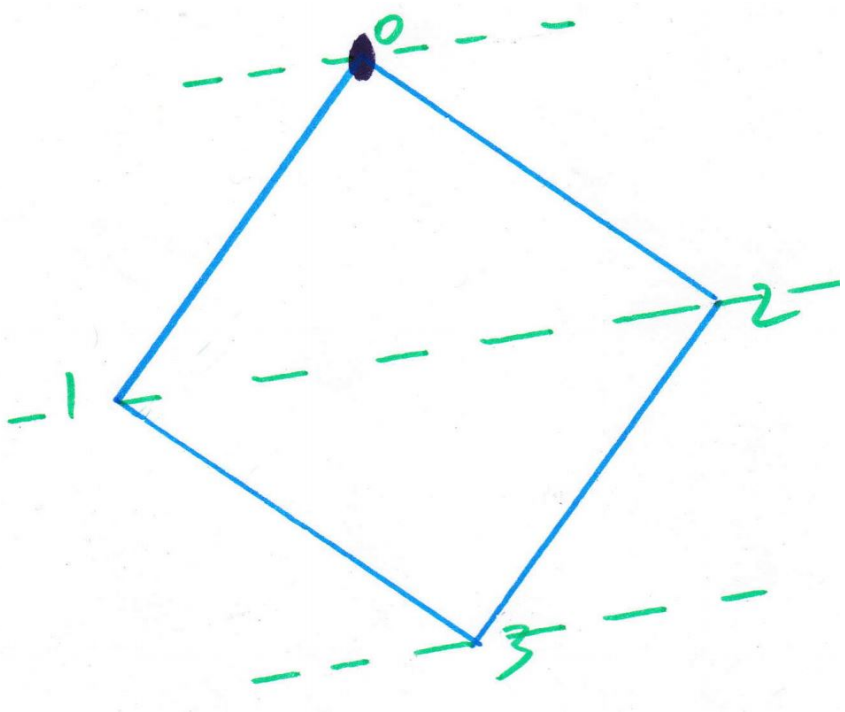

Figure 2.3: 2*2 regular mesh and the data injection is on the corner unit core P_0

According to the divisible load theory[2],we can obtain the time diagram Fig. 2.4 and the group of linear equations based on the processor's communication and computations situation.

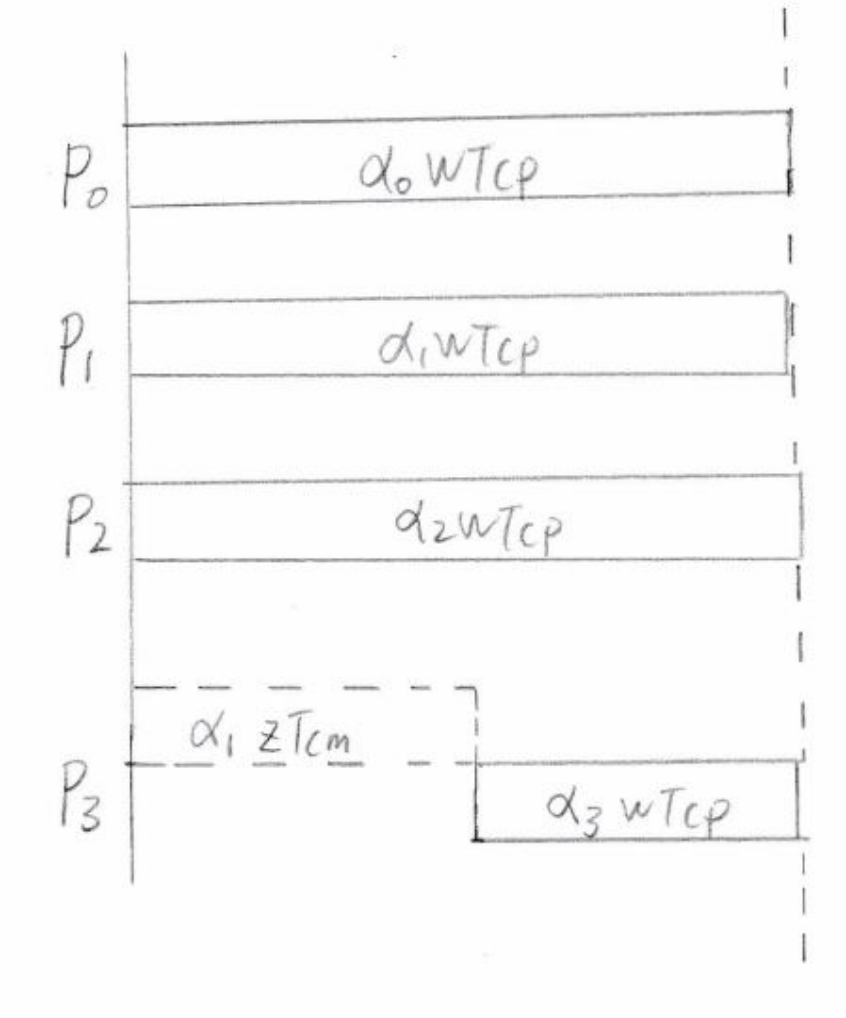

Figure 2.4: The time diagram for 2*2 regular mesh and the data injection position is on processor P_0

In Fig. 2.3 the equations are illustrated as follows:

$$
\begin{cases}
\alpha_0 \omega T_{cp} = T_{f,m} & (2.1) \\
\alpha_0 \omega T_{cp} = T_{f,m} & (2.2) \\
\alpha_1 \omega T_{cp} = T_{f,m} & (2.3) \\
\alpha_2 \omega T_{cp} = T_{f,m} & (2.4) \\
\alpha_1 z T_{cm} + \alpha_3 \omega T_{cp} = T_{f,m} & (2.5) \\
\alpha_0 + \alpha_1 + \alpha_2 + \alpha_3 = 1 & (2.6) \\
\sigma = \dfrac{z T_{cm}}{\omega T_{cp}} & (2.7) \\
 & (2.8)
\end{cases}
$$

We can obtain the

$$\alpha_1 = \alpha_2$$

Further,the group of equations can be transferred into the matrix form:

$$
\begin{bmatrix} 1 & 2 & 1 \\ 1 & -1 & 0 \\ 0 & \sigma - 1 & 1 \end{bmatrix} \times \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{2.9}
$$

$$0 < \sigma < 1$$

$$0 < \alpha_0, \alpha_1, \alpha_2, \alpha_3 < 1$$

We can get the solution simulation as follows:

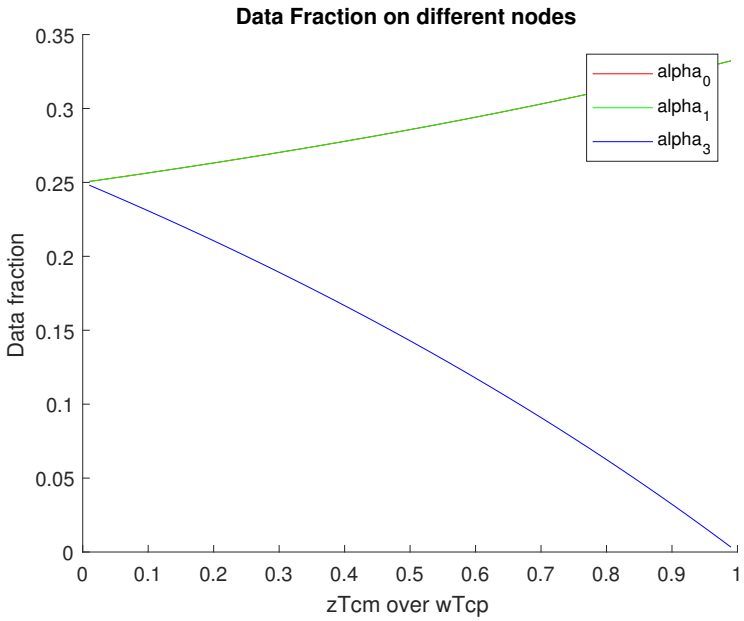

Figure 2.5: 2*2 regular mesh.α_0,α_1,α_2,α_3 value

As the figure illustrates, P_0,P_1,P_2 three unit cores have the same data fraction workload,so there are only two cures appearing in the figure. As the σ value grows,the α_3 drops simultaneously,which means the communication capacity drops down and there is less data workload transfered to the further processors such as P_3.

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

One more example is the $2*3$ Fig. 2.6 regular mesh and the data injection happens on processor P_0.Also, there are 6 processors to take the responsibility

to process the workload.

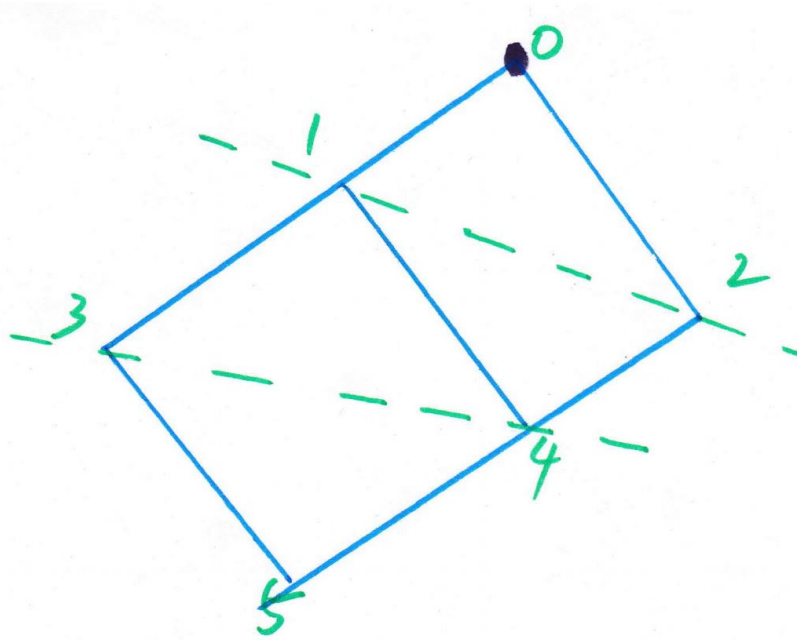

Figure 2.6: 2*3 regular mesh and the data injection happens on the corner processor P_0

According to the time diagram analysis:

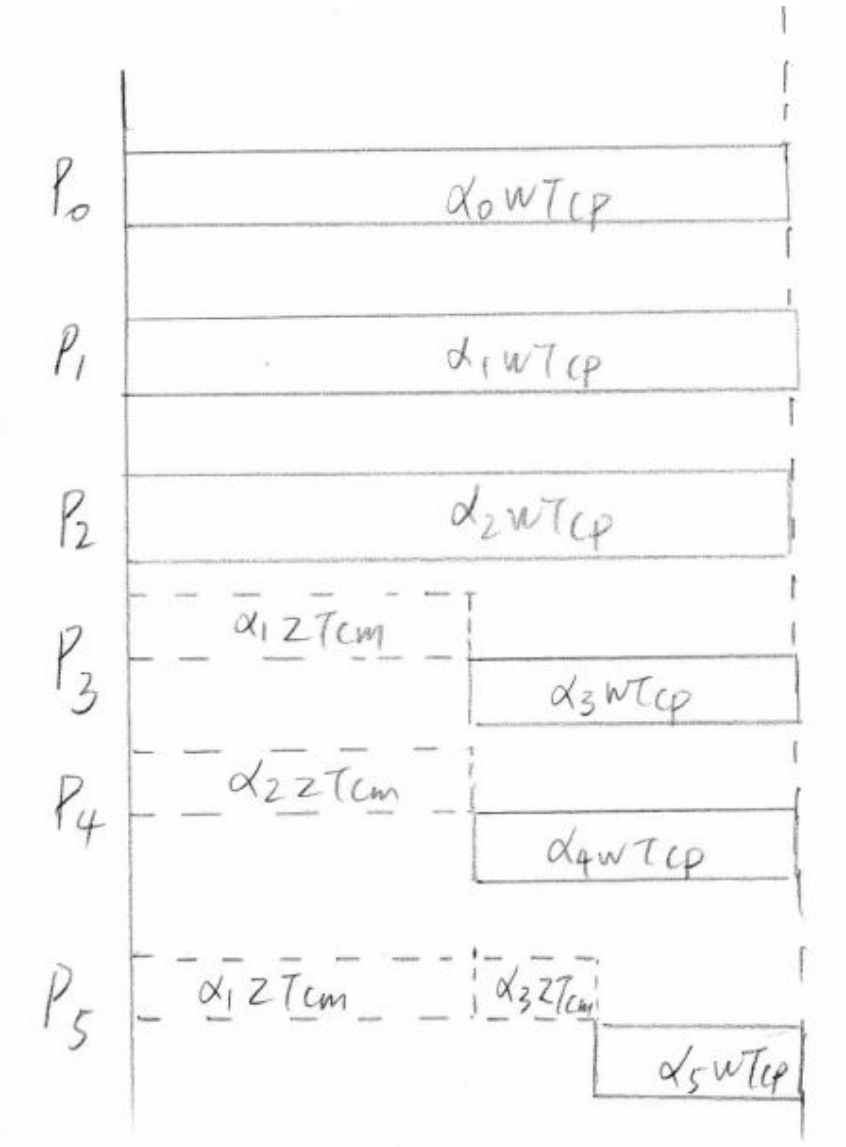

Figure 2.7: The time diagram for a 2*3 regular mesh and the data injection happens on processor P_0

The equations as follows:

$$
\begin{cases}
\alpha_0 \omega T_{cp} = T_{f,m} & (2.10) \\
\alpha_1 \omega T_{cp} = T_{f,m} & (2.11) \\
\alpha_2 \omega T_{cp} = T_{f,m} & (2.12) \\
\alpha_1 z T_{cm} + \alpha_3 \omega T_{cp} = T_{f,m} & (2.13) \\
\alpha_2 z T_{cm} + \alpha_4 \omega T_{cp} = T_{f,m} & (2.14) \\
(\alpha_1 + \alpha_3) z T_{cm} + \alpha_5 \omega T_{cp} = T_{f,m} & (2.15) \\
\alpha_0 + \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + \alpha_5 = 1 & (2.16) \\
\sigma = \dfrac{z T_{cm}}{\omega T_{cp}} & (2.17) \\
& (2.18)
\end{cases}
$$

After the simplification, we found:

$$\alpha_1 = \alpha_2$$

$$\alpha_3 = \alpha_4$$

So the group of equations can be reduced to a matrix form as follows:

$$
\begin{bmatrix}
1 & 2 & 2 & 1 \\
1 & -1 & 0 & 0 \\
0 & \sigma - 1 & 1 & 0 \\
0 & \sigma - 1 & \sigma & 1
\end{bmatrix}
\times
\begin{bmatrix}
\alpha_0 \\ \alpha_1 \\ \alpha_3 \\ \alpha_5
\end{bmatrix}
=
\begin{bmatrix}
1 \\ 0 \\ 0 \\ 0
\end{bmatrix}
\tag{2.19}
$$

The workload fraction is calculated in Fig. 2.8 the curve of α_0 and α_1 coincide:

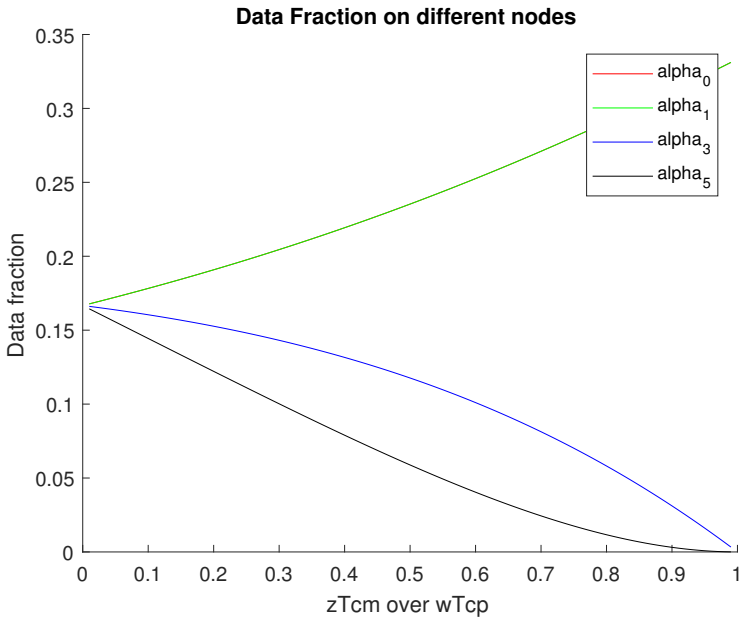

Figure 2.8: 2*2 regular mesh.α_0,α_1,α_3,α_5 data fraction value

2*n regular mesh

In this situation,there are $2*n$ Fig. 2.9 homogeneous processors to address the whole workload at the same time and the unique data injection happens on the corner processor P_0.

The diagram for Fig. 2.9 is Fig. 2.10:

The equations can be presented as:

$$\alpha_0 \omega T_{cp} = T_{f,m}$$

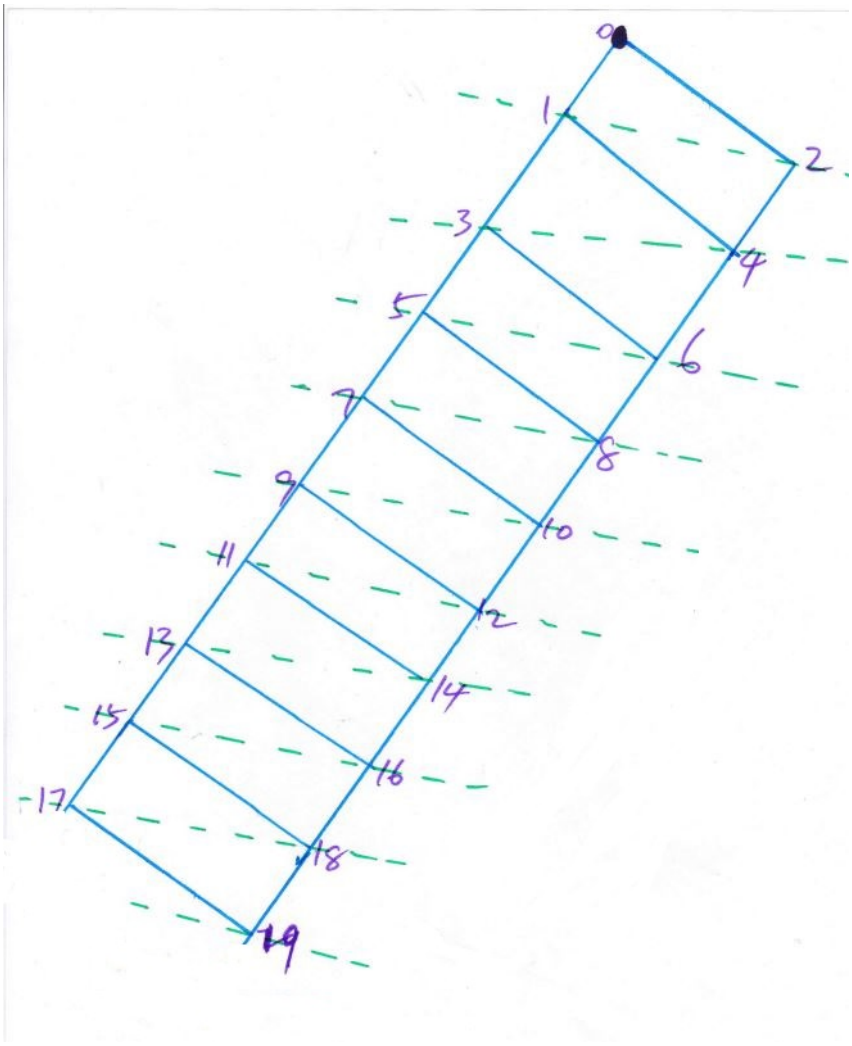

Figure 2.9: 2*n (n = 10) regular mesh.The workload happens on P_0

$$\alpha_1 \omega T_{cp} = T_{f,m}$$

$$\alpha_2 \omega T_{cp} = T_{f,m}$$

$$\alpha_1 z T_{cm} + \alpha_3 \omega T_{cp} = T_{f,m}$$

$$\alpha_2 z T_{cm} + \alpha_4 \omega T_{cp} = T_{f,m}$$

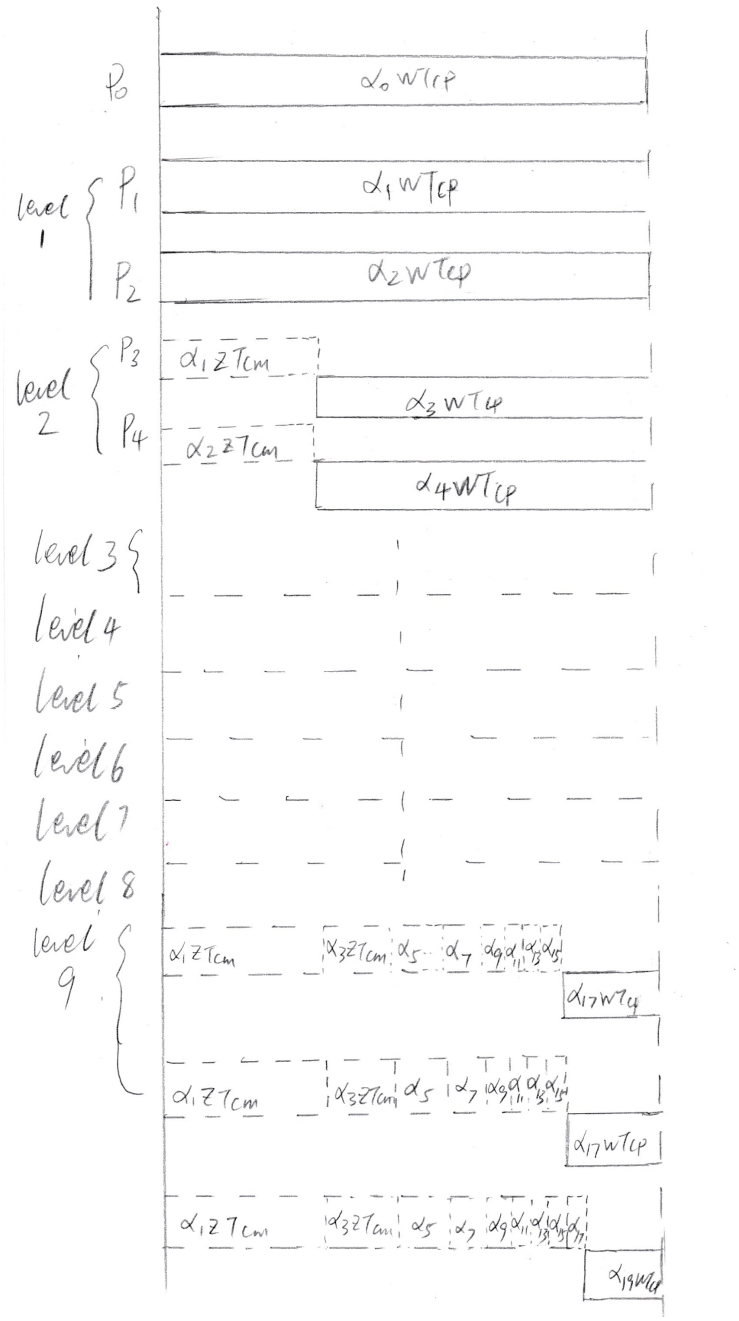

Figure 2.10: The time diagram for 2*10 regular mesh and the data injection happens on P_0

$$(\alpha_1 + \alpha_3) z T_{cm} + \alpha_5 \omega T_{cp} = T_{f,m}$$

$$\vdots$$

$$(\alpha_1 + \alpha_3 + \cdots + \alpha_{2\times n-1}) z T_{cm} + \alpha_{2\times n-1} \omega T_{cp} = T_{f,m}$$

$$\alpha_0 + \cdots + \alpha_{2\times n-1} = 1$$

$$\sigma = \frac{z T_{cm}}{\omega T_{cp}}$$

$$0 < \sigma < 1$$

$$0 < \alpha_0 \quad \alpha_1 \quad \alpha_3 \quad \cdots \quad \alpha_{2\times n-1} < 1$$

Transferring the groups of linear equations we can obtain:

$$\begin{bmatrix} 1 & 2 & 2 & \cdots & 2 & 2 & 1 \\ 1 & -1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & \sigma-1 & 1 & \cdots & 0 & 0 & 0 \\ 0 & \sigma-1 & \sigma & 1 & 0 & \cdots & 0 \\ 0 & \sigma-1 & \sigma & \sigma & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \\ 0 & \sigma-1 & \sigma & \cdots & \sigma & \sigma & 1 \end{bmatrix} \times \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_3 \\ \alpha_5 \\ \vdots \\ \alpha_{2\times n-3} \\ \alpha_{2\times n-1} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{2.20}$$

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 the investigation, we find that an import rule:

The processors which has the same hops distance with the data load have the same work load fraction.

The following regular mesh,torus cases and more general topology cases are analyzed by this technique.

m*n regular mesh

Considering a more general $m * n$ regular case mesh, such as Fig. 2.11 Fig. 2.12.

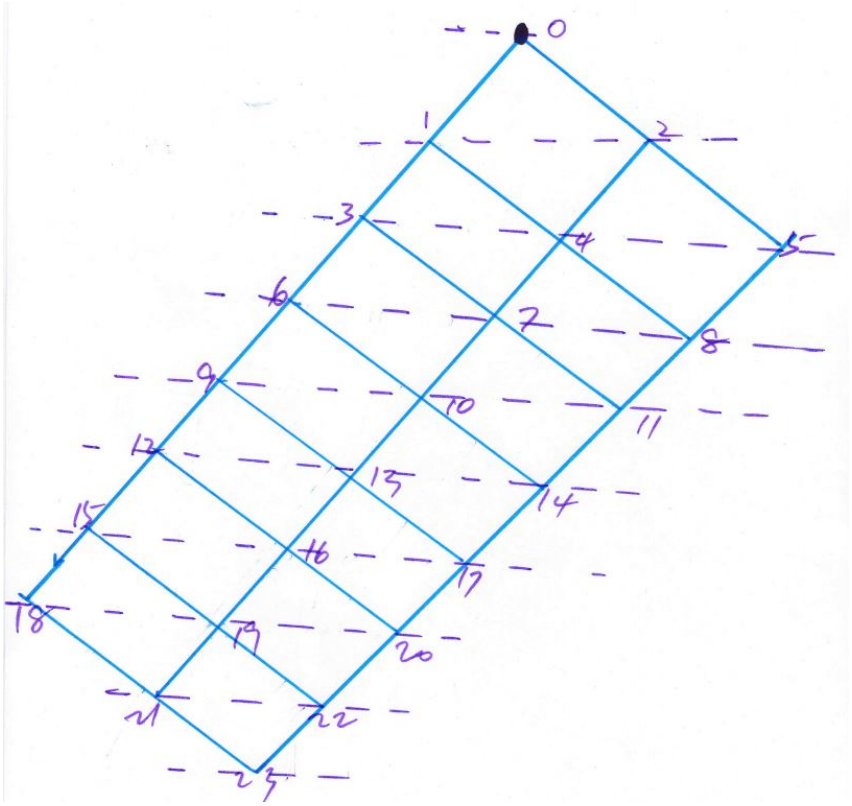

Figure 2.11: 3*8 regular mesh.The data injection position is P_0

The matrix form solution for Fig. 2.11 is shown:

$$\begin{bmatrix} 1 & 2 & 3 & 3 & 3 & 3 & 3 & 3 & 2 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma-1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma-1 & \sigma & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma-1 & \sigma & \sigma & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma-1 & \sigma & \sigma & \sigma & 1 & 0 & 0 & 0 & 0 \\ 0 & \sigma-1 & \sigma & \sigma & \sigma & \sigma & 1 & 0 & 0 & 0 \\ 0 & \sigma-1 & \sigma & \sigma & \sigma & \sigma & \sigma & 1 & 0 & 0 \\ 0 & \sigma-1 & \sigma & \sigma & \sigma & \sigma & \sigma & \sigma & 1 & 0 \\ 0 & \sigma-1 & \sigma & \sigma & \sigma & \sigma & \sigma & \sigma & \sigma^{18} & 1 \end{bmatrix} \times \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_3 \\ \alpha_6 \\ \alpha_9 \\ \alpha_{12} \\ \alpha_{15} \\ \alpha_{18} \\ \alpha_{21} \\ \alpha_{23} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{2.21}$$

The matrix form solution for Fig. 2.12 is shown:

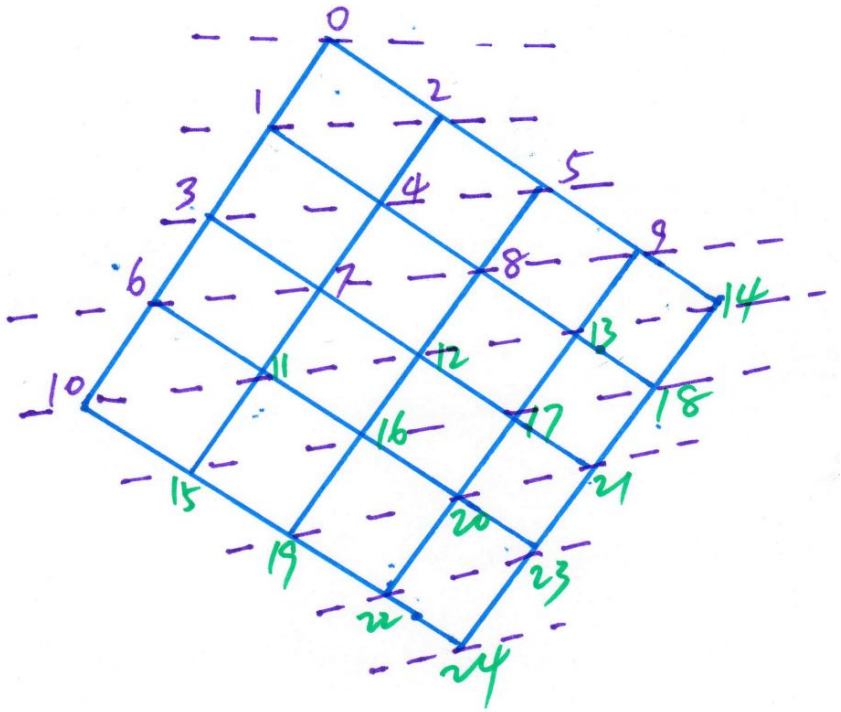

Figure 2.12: 5*5 regular mesh.The data injection position is processor P_0

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 4 & 3 & 2 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma-1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma-1 & \sigma & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma-1 & \sigma & \sigma & 1 & 0 & 0 & 0 & 0 \\ 0 & \sigma-1 & \sigma & \sigma & \sigma & 1 & 0 & 0 & 0 \\ 0 & \sigma-1 & \sigma & \sigma & \sigma & \sigma & 1 & 0 & 0 \\ 0 & \sigma-1 & \sigma & \sigma & \sigma & \sigma & \sigma & 1 & 0 \\ 0 & \sigma-1 & \sigma & \sigma & \sigma & \sigma & \sigma & \sigma & 1 \end{bmatrix} \times \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_3 \\ \alpha_6 \\ \alpha_{10} \\ \alpha_{15} \\ \alpha_{19} \\ \alpha_{22} \\ \alpha_{24} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{2.22}$$

2.2.3 Data injection on the boundary processor

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

The time diagram is Fig. 2.14:

The equations are following:

$$\alpha_0 \omega T_{cp} = T_{f,m}$$

$$\alpha_1 \omega T_{cp} = T_{f,m}$$

$$\alpha_2 \omega T_{cp} = T_{f,m}$$

$$\alpha_3 \omega T_{cp} = T_{f,m}$$

$$\alpha_1 z T_{cm} + \alpha_4 \omega T_{cp} = T_{f,m}$$

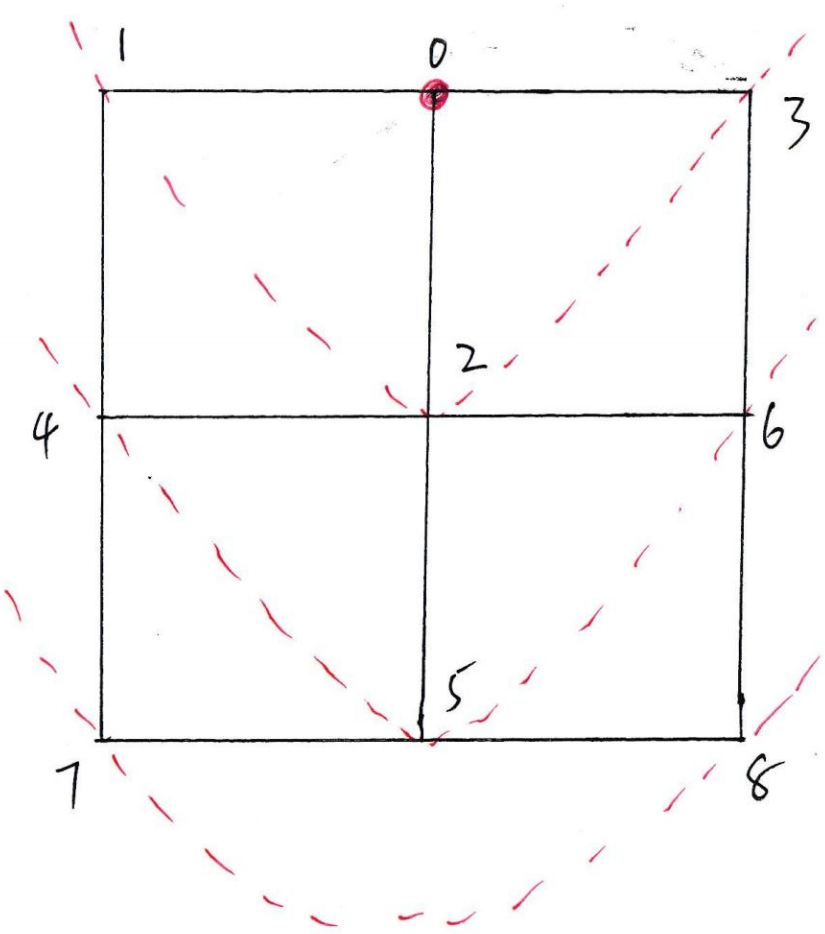

Figure 2.13: 3*3 regular mesh.The data injection position is P_0

$$\alpha_2 z T_{cm} + \alpha_5 \omega T_{cp} = T_{f,m}$$

$$\alpha_3 z T_{cm} + \alpha_6 \omega T_{cp} = T_{f,m}$$

$$(\alpha_1 + \alpha_4) z T_{cm} + \alpha_7 \omega T_{cp} = T_{f,m}$$

$$(\alpha_2 + \alpha_5) z T_{cm} + \alpha_8 \omega T_{cp} = T_{f,m}$$

$$\alpha_0 + \cdots + \alpha_8 = 1$$

$$\sigma = \frac{zT_{cm}}{\omega T_{cp}}$$

From the group of

And the matrix form is :

$$\begin{bmatrix} 1 & 3 & 3 & 2 \\ 1 & -1 & 0 & 0 \\ 0 & \sigma - 1 & 1 & 0 \\ 0 & \sigma - 1 & \sigma & 1 \end{bmatrix} \times \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_4 \\ \alpha_7 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.23}$$

And the simulation result is shown:

2.2.4 Data injection on the inner grid processor

Looks at the special case about the data injection happens on the inner grid,

The time diagram for this user case can be illustrated as Fig. 2.17:

The group of equations are:

$$\alpha_0 \omega T_{cp} = T_{f,m}$$

$$\alpha_1 \omega T_{cp} = T_{f,m}$$

$$\alpha_2 \omega T_{cp} = T_{f,m}$$

$$\alpha_3 \omega T_{cp} = T_{f,m}$$

$$\alpha_4 \omega T_{cp} = T_{f,m}$$

$$\alpha_1 z T_{cm} + \alpha_5 \omega T_{cp} = T_{f,m}$$

$$\alpha_1 z T_{cm} + \alpha_6 \omega T_{cp} = T_{f,m}$$

$$\alpha_1 z T_{cm} + \alpha_7 \omega T_{cp} = T_{f,m}$$

$$\alpha_1 z T_{cm} + \alpha_8 \omega T_{cp} = T_{f,m}$$

The equation for the boundary condition as follows:

$$\sigma = \frac{z T_{cm}}{\omega T_{cp}}$$

And the matrix form is :

$$\begin{bmatrix} 1 & 4 & 4 \\ 1 & -1 & 0 \\ 0 & \sigma - 1 & 1 \end{bmatrix} \times \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_5 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{2.24}$$

The simulation result for the inner grid is Fig. 2.18:

2.2.5 Torus mesh

2.3 Without Front End Scenario

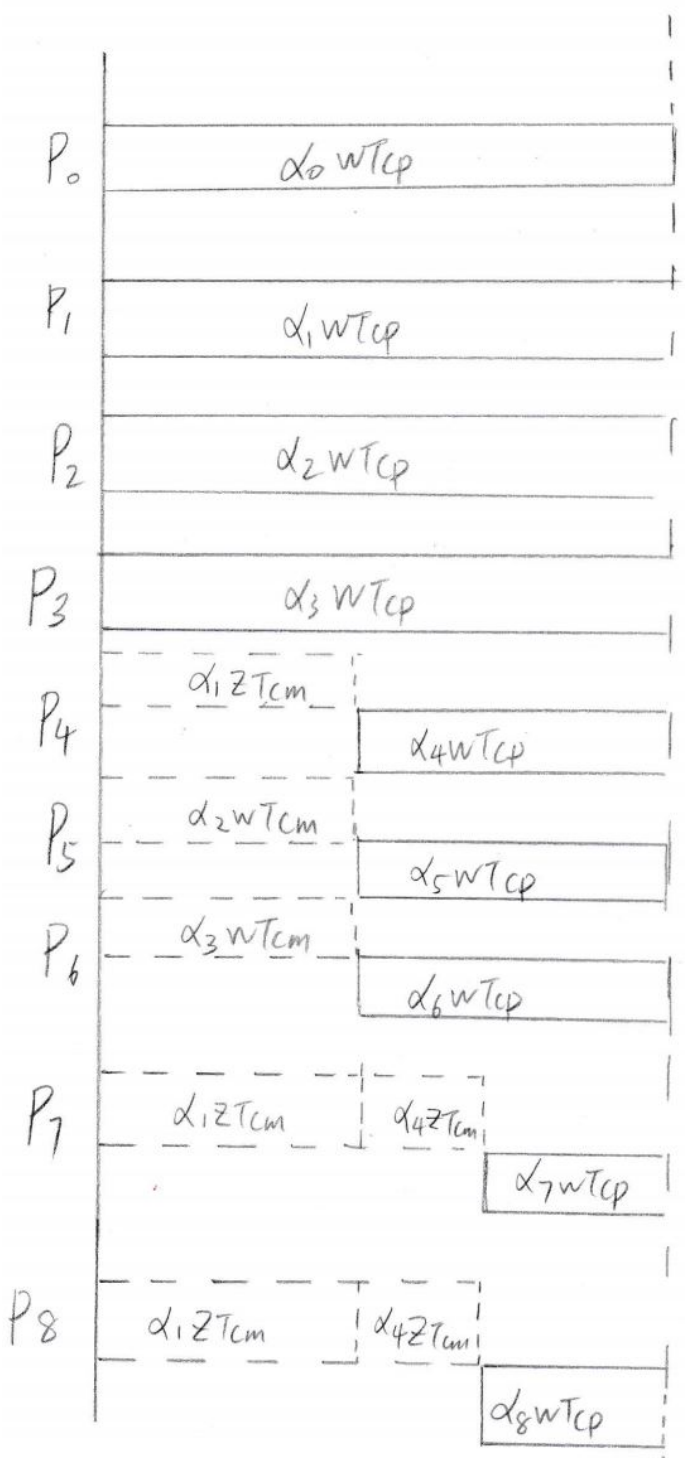

Figure 2.14: The time diagram for 3*3 regular mesh and the data injection occurs on the boundary processor P_0

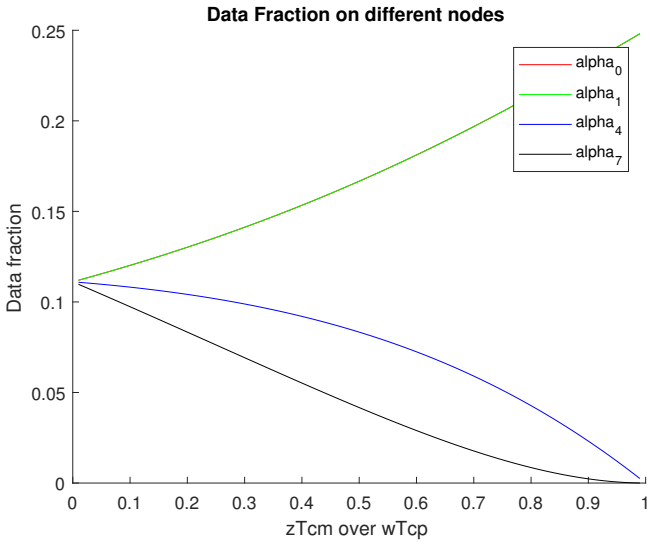

Figure 2.15: The data fraction simulation result of 3*3 regular mesh and the data injection happens on the boundary P_0

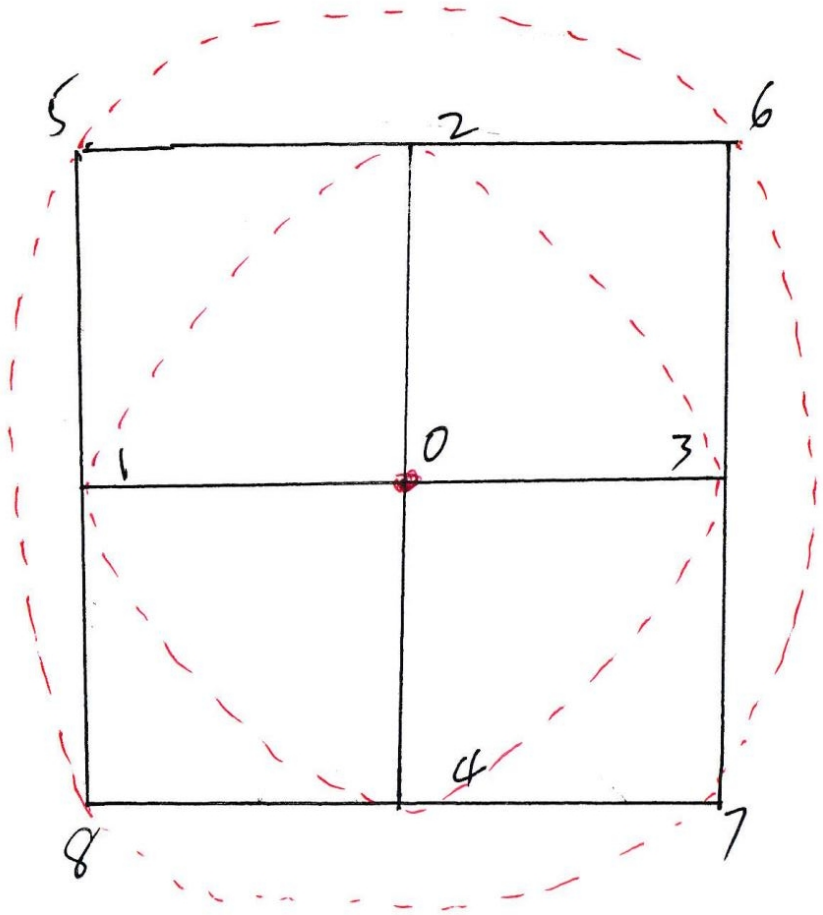

Figure 2.16: 3*3 regular mesh.The data injection position is inner grid point P_0

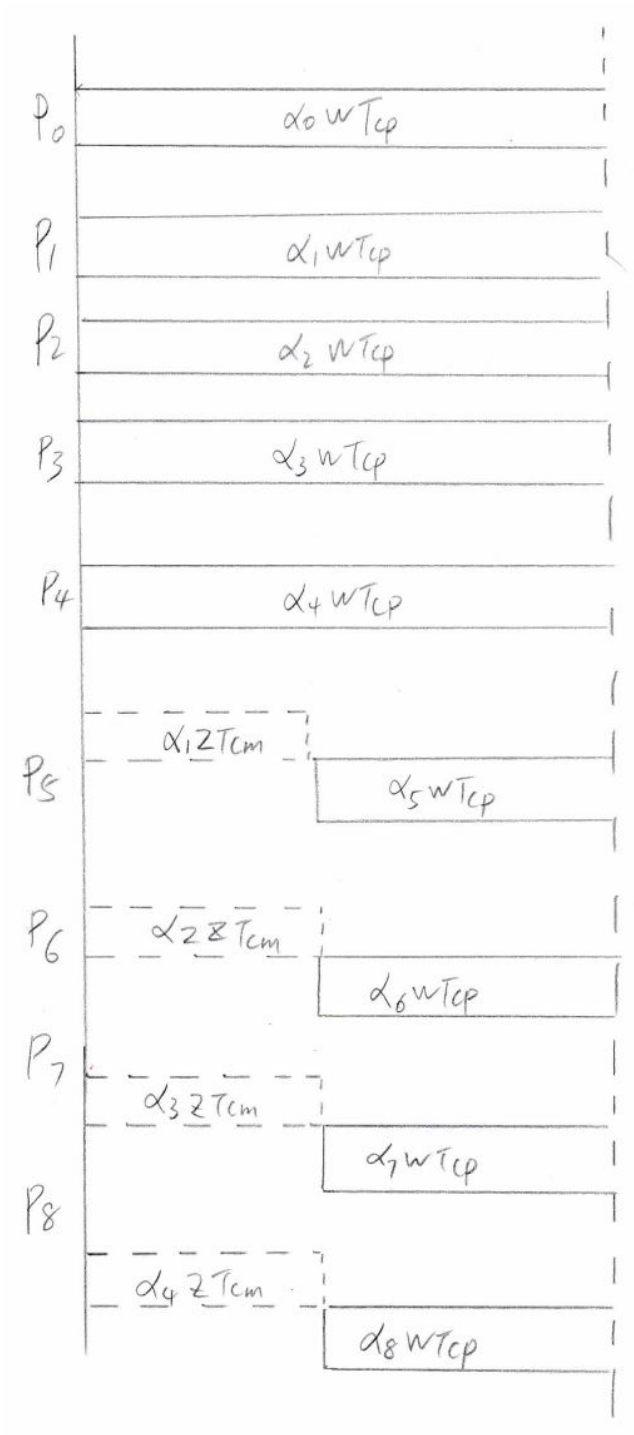

Figure 2.17: The time diagram for 3*3 regular mesh and the data injection is inner grid P_0

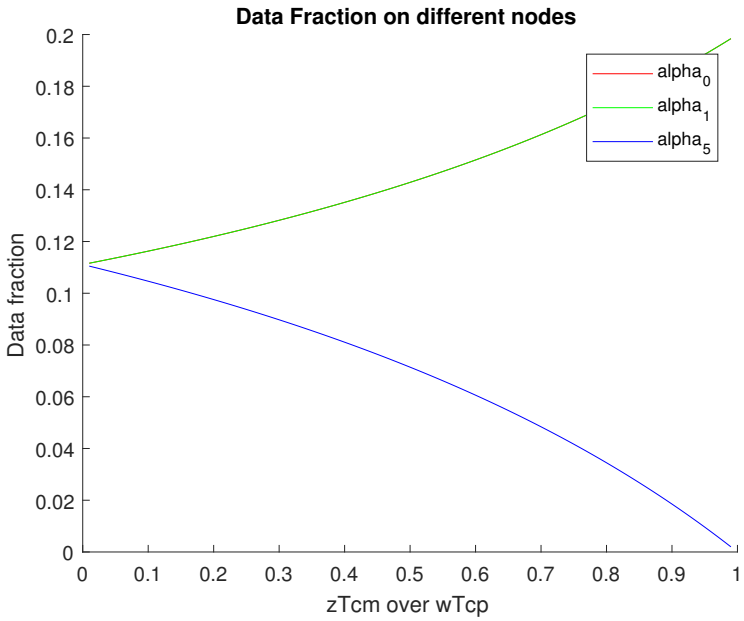

Figure 2.18: 3*3 regular mesh.The data injection position is inner grid point P_0

Chapter 3

Numerical Implementations

Chapter 4

Numerical Results

Chapter 5

Parallelization

Chapter 6

Conclusions

Bibliography

[1] 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.

[2] 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.