Optimality for Staggered Scheduling

Kwangil Ko*,
Thomas Robertazzi*
and Larry Wittie**
*Dept. of Electrical and Computer Engineering,
**Dept. of Computer Science,
Stony Brook University,
Stony Brook, NY 11794
e-mail: tom@ece.sunysb.edu

Abstract —

Staggered scheduling is compared to "arbitrary" scheduling using matrix based techniques. It is shown that staggered scheduling always has a smaller solution time compared to any other possible arbitrary schedule for the same sequence of load distribution and solution reporting. This work is significant in using the arbitrary schedule concept in much the same way as the general distribution is used in queueing theory.

I. Introduction

Over the past 15 years there has been a good amount of research [1],[2] on divisible load scheduling. A divisible load is a computation and communication load that can be arbitrarily partitioned among processors and links. Such loads arise in data intensive computing, grid computing and metacomputing. A generally linear theory has been developed that provides tractable analytic solutions for optimal load allocation for a given scheduling strategy and interconnection network.

In this paper staggered scheduling is compared to "arbitrary" scheduling using matrix based techniques. Here staggered scheduling means that a computational load is well scheduled so as to minimize communication delay (this is elaborated on in the paper). It is proved that staggered scheduling always has a smaller solution time compared to any other possible arbitrary schedule for the same sequence of load distribution and solution reporting. This work is novel and significant in using the arbitrary schedule concept for scheduling, in much the same way the general distribution is used in queueing theory. It is also novel in its use of a matrix based analytical methodology. A final novel feature of this work is the optimization of solution reporting sequence back to the load originating processor.

An earlier work on divisible load scheduling optimality is [3].

This work is done in the context of divisible load modeling in a single level tree network with sequential load distribution and sequential solution reporting.

II. Arbitrary Load Scheduling

In this section load scheduling which is not necessarily optimal, called arbitrary load scheduling, is discussed. The computation of finish time is obtained when arbitrary load is distributed to each processor. In arbitrary load scheduling, the load is not scheduled to each processor but allocated arbitrarily; that is, we allow any possible load fraction allocation to processors as long as a specified sequence of load distribution and solution reporting is kept. We use the word "arbitrarily" in the same sense as a M/G/1 queue has a "general" solution. That is, any scheduling policy may be substituted for an arbitrary scheduling and the results still hold. The following notation will be used throughout this paper:

- p_i: The i^{th} processor.
- α_i: The fraction of the entire processing load assigned to the i^{th} processor.
- w_i: A constant inversely proportional to the computation speed of the i^{th} processor.
- z_i: A constant inversely proportional to the channel speed of the i^{th} link
- T_{cp}: Computing intensity constant. The entire load is processed in $w_i T_{cp}$ seconds by 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_{cm}^{sol}: Solution reporting communication intensity constant. The entire solution report can be transmitted in $z_i T_{cm}^{sol}$ seconds over the i^{th} link.

Assume that there are $(M+1)$ processors named as follows:

$$\mathbf{P} = \{p_0, p_1, p_2, \cdots, p_i, \cdots, p_M\}$$

An arbitrary load distribution is defined as an $(M+1)$ tuple, $\widetilde{\boldsymbol{\alpha}}$, given by the fractions of the total load allocated to each processor, p_i for $i = 0, 1, 2, \cdots, M$ and selected arbitrarily.

$$\widetilde{\boldsymbol{\alpha}} = (\widetilde{\alpha}_0, \widetilde{\alpha}_1, \widetilde{\alpha}_2, \cdots, \widetilde{\alpha}_i, \cdots, \widetilde{\alpha}_M) \tag{1}$$

Here, the tilde is used to denote an arbitrary schedule. Further, the load is normalized.

$$\sum_{i=0}^{M} \widetilde{\alpha}_i = 1 \tag{2}$$

Fig. 1 shows a timing diagram for arbitrary load distribution. The computation takes place over a single level tree network where each processor is equipped with a front-end processor for communication off-loading. The use of front end processors for communication off-loading means that communication and computation can proceed simultaneously for each processor. In the diagram computation time appears above each axis and communication time appears below each axis.

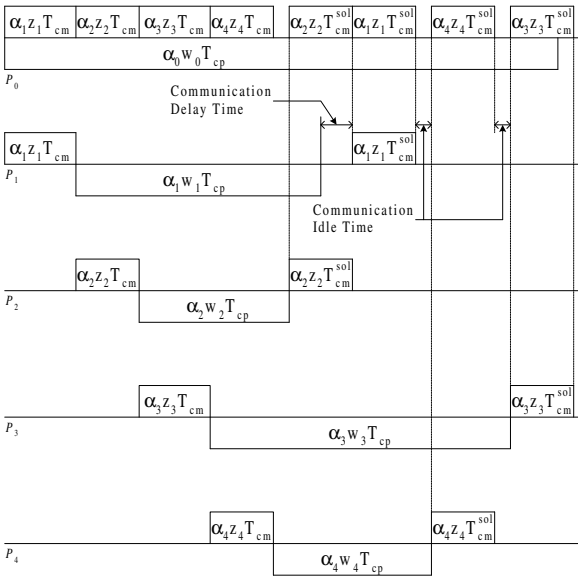

Figure 1: Timing Diagram for Arbtrary Scheduling of Homogeneous Single Level Tree Network.

Without loss of generality, it is assumed that load is distributed from originator (root) processor, p_0. As soon as each child processor completely receives its assigned segment of the load it begins processing that segment. After each processor finishes this computation it requires the use of its link for a period of time to broadcast the result of its computation back to the originator. It is assumed that the scheduling must be done in such a way that only one processor transmits at one time and processors have sufficient time to perform their computations before transmitting their results. Thus, in the diagram processor 1's solution reporting is delayed until processor 2 reports its solution. Also note the presence of idle time after processor 1 reports its solution as processors 3 and 4 are still computing.

The originator processor, p_0 distributes fractions to M processors from p_1 to p_M in sequence. For simplicity we do not consider multi-installment load distribution strategies as in [2]. Processors start to process their fractions as soon as processors receive their fraction. For example, the i^{th} processor waits until the previous processors receive their fractions, and then starts to receive its fraction, $\widetilde{\alpha}_i$. After this reception, the i^{th} processor can processes its fraction.

First, it is necessary to calculate the time that it takes for the i^{th} processor to receive the fraction of the arbitrarily scheduled load, $\widetilde{\alpha}_i$, and to process this load. That time is defined as $\widetilde{T}_{rp,p_i}$ (the "rp" stands for receive and process).

$$\widetilde{T}_{rp,p_i} = \sum_{j=1}^{i} \widetilde{\alpha}_j z_j T_{cm} + \widetilde{\alpha}_i w_i T_{cp} \tag{3}$$

Note that the above expressions do not include solution reporting time. The first term and the second term in equation (3) are the reception time and computation time for the i^{th} processor, respectively.

However, processors finish processing in some arbitrary order as shown in Fig. 1. The reporting order is not predetermined but depends on $\widetilde{T}_{rp,p_i}$ for $i = 1, 2, \cdots, M$. The solution for the i^{th} processor can be sent to the originator processor after $\widetilde{T}_{rp,p_i}$. The processors are assumed to send their solutions back to the root in the order of finishing their work under arbitrary scheduling. Thus, sorting $\left\{\widetilde{T}_{rp,p_i} : i = 1, 2, \cdots, M\right\}$ is needed to determine the solution reporting order in the case of arbitrary load scheduling. The result of ascending order sorting $\left\{\widetilde{T}_{rp,p_i} : i = 1, 2, \cdots, M\right\}$ can be represented by:

$$\widetilde{\mathbf{T}}_{rp}^{(sort)} = \left\{\widetilde{T}_{rp}^{(1)}, \cdots, \widetilde{T}_{rp}^{(i)}, \cdots, \widetilde{T}_{rp}^{(j)}, \cdots \widetilde{T}_{rp}^{(M)}\right\} \tag{4}$$

Here for $i < j$:

$$\widetilde{T}_{rp}^{(i)} \leq \widetilde{T}_{rp}^{(j)} \tag{5}$$

Further, $\widetilde{\boldsymbol{\alpha}}^{(sort)}$ is the version of the sorting $\{\widetilde{\alpha}_i : i = 1, 2, \cdots, N\}$ with respect to $\widetilde{\mathbf{T}}_{rp}^{(sort)}$.

$$\widetilde{\boldsymbol{\alpha}}^{(sort)} = \left\{\widetilde{\alpha}^{(1)}, \cdots, \widetilde{\alpha}^{(i)}, \cdots, \widetilde{\alpha}^{(j)}, \cdots, \widetilde{\alpha}^{(M)}\right\} \tag{6}$$

Consequently, $\widetilde{\alpha}^{(i)}$ is the load fraction of the entire processing load assigned to $p^{(i)}$. Here $p^{(i)}$ is defined as the i^{th} reporting processor. Also, $z^{(i)}$ is the inverse speed of the link connected to $p^{(i)}$. Then the following property holds:

$$\sum_{i=1}^{M} \widetilde{\alpha}^{(i)} z^{(i)} = \sum_{i=1}^{M} \widetilde{\alpha}_i z_i \tag{7}$$

The reporting order is that $p^{(i)}$ is followed by $p^{(i+1)}$ for $i = 1, 2, \cdots, M-1$. In Fig. 1, the reporting order of processors is p_2, p_1, p_4, and then p_3.

"Finish time" is the time for the system to complete processing its entire load. Naturally, in general finish time is equal to the time when the originator processor finishes processing or the last reporting processor completes processing and reporting.

$$\widetilde{T}_f(M) = \max\left[\ \widetilde{\alpha}_0 w_0 T_{cp} \quad \widetilde{T}_s^{(M)}\ \right] \tag{8}$$

The term, $\widetilde{\alpha}_0 w_0 T_{cp}$ is the processing time of the originator processor. Here, $\widetilde{\alpha}_0$ is arbitrary generated. The term, $\widetilde{T}_s^{(M)}$, is the time taken by the last reporting processor to complete reporting.

In this section, we develop an expression for the finish time of arbitrary scheduling. Now, $\widetilde{T}_s^{(0)}$ is the time taken by the originator processor to distribute load fractions to all of the processors.

$$\widetilde{T}_s^{(0)} = \sum_{j=1}^{M} \widetilde{\alpha}_j z_j T_{cm} \tag{9}$$

Also, $\widetilde{T}_s^{(i)}$ is the time taken by the i^{th} reporting order processor to complete sending its solution according to this reporting order. Here $p^{(i)}$, the i^{th} reporting order processor, tries to report its solution after $\widetilde{T}_{rp}^{(i)}$. The i^{th} reporting order processor can immediately send its solution if $p^{(i-1)}$ already finished reporting. However, if $p^{(i-1)}$ didn't finish reporting its solution yet, $p^{(i)}$ should wait until $p^{(i-1)}$ finishes. Consequently, $p^{(i)}$ can report its solution after a time equal to the maximum of $\widetilde{T}_s^{(i-1)}$ and $\widetilde{T}_{rp}^{(i)}$. For $i = 1, 2, \cdots, M$:

$$\widetilde{T}_s^{(i)} = \max\left[\widetilde{T}_s^{(i-1)}, \widetilde{T}_{rp}^{(i)}\right] + \widetilde{\alpha}^{(i)} z^{(i)} T_{cm}^{sol} \tag{10}$$

The last term in the above equation is the time taken by $p^{(i)}$ to report. Note that the first reporting processor can't send

its solution while the originator processor distributes fractions to children processors.

Let G_1 be the time difference between the instant of the start of the transmission of the solution of the first reporting processor, $\widetilde{T}_{rp}^{(1)}$ and the instant that the originator processor finishes distributing load fractions, $\widetilde{T}_s^{(0)}$. Also let G_i be the time difference between $\widetilde{T}_{rp}^{(i)}$, the instant of the start of transmission of the solution of the i^{th} reporting processor, and $\widetilde{T}_s^{(i-1)}$, the instant when the transmission of the solution of the $(i-1)^{th}$ reporting processor is complete. For $i = 2, 3, \cdots, M$:

$$G_i = \widetilde{T}_{rp}^{(i)} - \widetilde{T}_s^{(i-1)} \tag{11}$$

It is assumed that the root can transmit or receive with only one child processor at a time. In the case that $p^{(i)}$ were to begin to report its solution while $p^{(i-1)}$ is transmitting its solution, $\left(\widetilde{T}_{rp}^{(i)} < \widetilde{T}_s^{(i-1)}\right)$, a conflict would occur. Also all links are idle when $p^{(i)}$ cannot immediately send its solution, after $p^{(i-1)}$ has already transmitted $\widetilde{\alpha}^{(i-1)}$, $\left(\widetilde{T}_{rp}^{(i)} > \widetilde{T}_s^{(i-1)}\right)$. A negative G_i indicates a conflict and a positive G_i indicates a time when all of the links are idle. In the case of a conflict $\left(G_i < 0 \text{ or } \widetilde{T}_{rp}^{(i)} < \widetilde{T}_s^{(i-1)}\right)$, $p^{(i)}$ can immediately report its solution as soon as $p^{(i-1)}$ finishes transmitting its solution. Thus equation (10) is represented as follows:

$$\widetilde{T}_s^{(i)} = \widetilde{T}_s^{(i-1)} + \widetilde{\alpha}^{(i)} z^{(i)} T_{cm}^{sol} \tag{12}$$

In the case of idleness $\left(G_i > 0 \text{ or } \widetilde{T}_{rp}^{(i)} > \widetilde{T}_s^{(i-1)}\right)$, equation (10) is represented as follow:

$$\widetilde{T}_s^{(i)} = \widetilde{T}_{rp}^{(i)} + \widetilde{\alpha}^{(i)} z^{(i)} T_{cm}^{sol} \tag{13}$$

Using equation (11) the last two equations can be combined as follows:

$$\widetilde{T}_s^{(1)} = \widetilde{T}_s^{(0)} + \widetilde{\alpha}^{(1)} z^{(1)} T_{cm}^{sol} + G_1 sign\,(G_1) \tag{14}$$

$$\vdots$$

$$\widetilde{T}_s^{(i)} = \widetilde{T}_s^{(i-1)} + \widetilde{\alpha}^{(i)} z^{(i)} T_{cm}^{sol} + G_i sign\,(G_i) \tag{15}$$

$$\widetilde{T}_s^{(i+1)} = \widetilde{T}_s^{(i)} + \widetilde{\alpha}^{(i+1)} z^{(i+1)} T_{cm}^{sol} + G_{i+1} sign\,(G_{i+1})$$

$$\vdots \tag{16}$$

$$\widetilde{T}_s^{(M)} = \widetilde{T}_s^{(M-1)} + \widetilde{\alpha}^{(M)} z^{(M)} T_{cm}^{sol} + G_M sign\,(G_M) \tag{17}$$

Here $sign(x)$ is defined as follows:

$$sign(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \tag{18}$$

In equation (15), the time for the i^{th} reporting processor to finish reporting its solution includes three terms. One is the time for the $(i-1)^{th}$ reporting processor to finish reporting. The second term is the time for the i^{th} reporting processor to report the solution. The last term is the potential idle time of reporting between $p^{(i)\prime}$s and $p^{(i-1)\prime}$s reports.

The solution of the recursion of equation (14) to equation (17), $\widetilde{T}_s^{(M)}$, can be written as follows:

$$\widetilde{T}_s^{(M)} = \widetilde{T}_s^{(0)} + \sum_{i=1}^{M} \widetilde{\alpha}^{(i)} z^{(i)} T_{cm}^{sol} + \sum_{i=1}^{M} G_i sign(G_i) \tag{19}$$

Substituting equation (7) and (9) into (19), $\widetilde{T}_f^{(M)}$ can be rewritten as follows:

$$\widetilde{T}_s^{(M)} = \sum_{i=1}^{M} \widetilde{\alpha}_i z_i T_{cm} + \sum_{i=1}^{M} \widetilde{\alpha}_i z_i T_{cm}^{sol} + \sum_{i=1}^{M} G_i sign(G_i) \tag{20}$$

The above equation expresses the time at which all processors finish reporting their solution to the originator processor. Note that $\widetilde{T}_f(M)$, defined as the finish time for arbitrary load scheduling, is the maximum of the processing time of the originator processor and the completion time of reporting all solutions. The processing time of the originator processor is determined by $\widetilde{\alpha}_0$ generated arbitrary and the completion time of reporting all solutions is obtained from equation (20). Thus, $\widetilde{T}_f(M)$ from equation (8) can be calculated.

Next, our attention turns to developing the concept of staggered scheduling.

III. Staggered Scheduling

The basic idea of staggered scheduling is that $p^{(i)}$, the i^{th} reporting processor, begins to transmit its solution immediately after the transmission of the solution of $p^{(i-1)}$, the $(i-1)^{th}$ reporting processor for $i = 2, 3, \cdots, M$ while the originator (root) processor finishes its processing as soon as receiving the solution of the last reporting order processor, $p^{(M)}$.

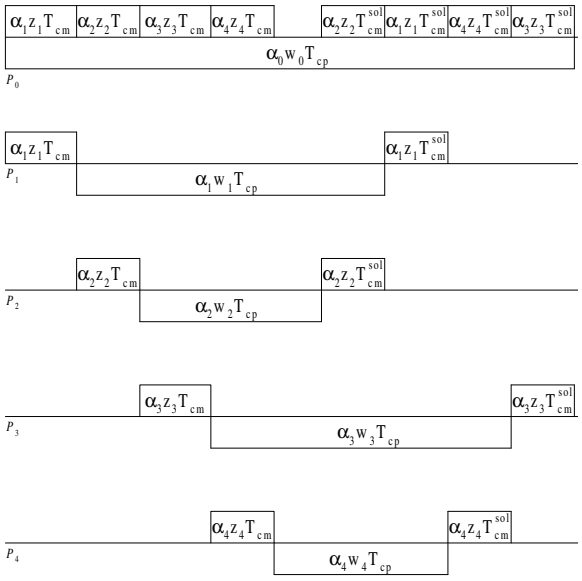

Figure 2: Timing Diagram for Staggered Scheduling of Homogeneous Single Level Tree Network.

Fig. 2 is a timing diagram of staggered scheduling. From equation (19) the first term and $G_1 sign(G_1)$ can be represented as follows:

$$\begin{aligned} T_s^{(0)} &+ G_1 sign(G_1) \\ &= \widetilde{T}_s^{(0)} + \left(T_{rp}^{(1)} - \widetilde{T}_s^{(0)}\right) \cdot sign\left(T_{rp}^{(1)} - \widetilde{T}_s^{(0)}\right) \end{aligned} \tag{21}$$

Again, $T_s^{(0)}$ is the time taken by the originator processor to distribute load fractions to all of the processors so as to create a well posed problem. The originator processor can distribute fractions before the first solution is received $\left(T_{rp}^{(1)} \geq \widetilde{T}_s^{(0)}\right)$.

The above equation can then be reduced to $T_{rp}^{(1)}$. This can be done since $T_{rp}^{(1)}$ is greater than $\widetilde{T}_s^{(0)}$ for staggered scheduling. Thus equation (19) can be rewritten as follows using equation (7):

$$\widetilde{T}_s^{(M)} = \widetilde{T}_{rp}^{(1)} + \sum_{j=1}^{M} \widetilde{\alpha}_j z_j T_{cm}^{sol} + \sum_{j=2}^{M} G_j sign(G_j) \tag{22}$$

If reporting solutions back to the originator processor is staggered so that there is no idle time, then the last term of the above equation is zero.

$$T_s^{(M)} = T_{rp}^{(1)} + \sum_{j=1}^{M} \alpha_j z_j T_{cm}^{sol} \tag{23}$$

Note we use α_i (no tilde) to denote a staggered scheduled load fraction.

In addition, the originator processor finishes its processing as soon as it receives the solution of $p^{(M)}$, the last reporting processor.

$$\begin{aligned} T_s^{(M)} &= T_f(M) \\ &= \alpha_0 w_0 T_{cp} \end{aligned} \tag{24}$$

Here, $\widetilde{\alpha}_i$ for $i = 0, 1, \cdots, M$ should be adjusted for staggering.

$$\alpha_i = \widetilde{\alpha}_i - \alpha_i^{'} \tag{25}$$

Again, $\widetilde{\alpha}_i$ is the arbitrary scheduling load fraction for the i^{th} processor and now α_i is the staggered scheduling load fraction for the i^{th} processor. The normalization equation of load fractions for staggered scheduling is applied.

$$\sum_{j=0}^{M} \alpha_j = 1 \tag{26}$$

From the two normalization equation, equation (2) and equation (26), the following equation is obtained.

$$\sum_{j=0}^{M} \alpha_j^{'} = 0 \tag{27}$$

The problem is to find α_i for $i = 0, 1, \cdots, M$ so that G_i for $i = 2, 3, 4, \cdots, M$ equals zero, which means that all processors are staggered. This problem is addressed in the next section.

Here, $M!$ staggered scheduling cases exist because of the reporting sequence. In this section, the staggered finish time for all cases will be obtained. The smallest one of the $M!$ staggered finish times is the optimal schedule. Each child processor has four time phases; receiving, processing, reporting the solution, and resting after reporting solution until the originator processor receives all solutions. The processor which sends its solution last has no rest time. Thus, the sum of the four time phases is equal to the finish time for any $i = 1, 2, \cdots, M$ and is represented as follows:

$$\begin{aligned} T_f(M) &= \alpha_0 w_0 T_{cp} \\ &= T_s^{(M)} \end{aligned} \tag{28}$$

For any $i = 0, 1, \cdots, M$:

$$\begin{aligned} T_s^{(M)} &= \sum_{j=1}^{i} \alpha_j z_j T_{cm} + \alpha_i w_i T_{cp} \\ &\quad + r_{i,i} \alpha_i z_i T_{cm}^{sol} + \sum_{j=1, j\neq i}^{M} r_{i,j} \alpha_j z_j T_{cm}^{sol} \end{aligned} \tag{29}$$

Here, we suppress that the dependence of $T_s^{(M)}$ on i as $T_s^{(M)}$ is the same for all i. The first two terms of equation (29) are the time taken by the i^{th} processor to receive its fraction and then to process its fraction, respectively. The processor reports its solution back to the originator (root) immediately after processing (third term). The last term is the resting time until all processors complete processing and reporting their solutions to the root. The $r_{i,j}$ is defined as a resting coefficient. Here, $r_{i,j}$ is one if the i^{th} processor is resting when the j^{th} processor reports its solution. Further $r_{i,j}$ is zero if the i^{th} processor is processing. The coefficient $r_{i,j}$ can be determined from the solution reporting order of processors and, again, could be zero or one. If the i^{th} processor reports its solution before the j^{th} processor, $r_{i,j}$ is one and $r_{j,i}$ is zero. If the i^{th} processor reports its solution after the j^{th} processor, $r_{i,j}$ is zero and $r_{j,i}$ is one. Thus, one of the $r_{i,j}$ and $r_{j,i}$ is one, and the other zero. Also the i^{th} processor should wait until it sends its own solution so $r_{i,i} = 1$ in the third term of equation (29). If the i^{th} processor is the k^{th} solution reporting processor, then the following equation holds:

$$\begin{aligned} \sum_{j=1}^{M} r_{j,i} &= k \qquad (30) \\ &= S_i \qquad (31) \end{aligned}$$

Here, S_i is the reporting order of the i^{th} processor. The properties of the resting coefficient are summarized as follows:

1. $r_{i,j} = 1$ or 0
2. $r_{i,j} + r_{j,i} = 1$
3. $r_{i,j} \cdot r_{j,i} = 0$
4. $(r_{i,j})^2 = r_{i,j}$
5. $r_{i,j} = 1$ if $j = i$

Equation (28) can be rewritten as follows for $i = 1, 2, \cdots, M$.

$$\begin{aligned} &-\alpha_0 w_0 T_{cp} + \sum_{j=1}^{i} \alpha_j z_j T_{cm} + \alpha_i w_i T_{cp} \\ &+\alpha_i z_i T_{cm}^{sol} + \sum_{j=1, j\neq i}^{M} r_{i,j} \alpha_j z_j T_{cm}^{sol} \\ = \;& 0 \end{aligned} \tag{32}$$

Equation (32) for $i = 1, 2, \cdots, M$ and (26) form $(M+1)$ linear equations.

$$\left[\mathbf{D} + \mathbf{W} + \mathbf{R}^{(k)}\right] \boldsymbol{\alpha} = \mathbf{E} \tag{33}$$

Here,

$$\boldsymbol{\alpha} = \begin{bmatrix} \alpha_0 & \alpha_1 & \cdots & \alpha_{M-1} & \alpha_M \end{bmatrix}' \tag{34}$$

$$\mathbf{E} = \begin{bmatrix} T_{cp} & 0 & 0 & \cdots & 0 \end{bmatrix}' \tag{35}$$

$$\mathbf{D} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & z_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & z_1 & \cdots & z_M \end{bmatrix} T_{cm} \tag{36}$$

$$\mathbf{W} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ -w_0 & w_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -w_0 & 0 & \cdots & w_M \end{bmatrix} T_{cp} \tag{37}$$

$$\mathbf{R}^{(k)} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & z_1 & \cdots & r_{1,M} z_M \\ 0 & \vdots & \ddots & \vdots \\ 0 & r_{M,1} z_1 & \cdots & z_M \end{bmatrix} T_{cm}^{sol} \quad (38)$$

Here, $r_{i,j}$ is for $i,j = 0,1,2,\cdots,M$, and the reporting sequence matrix, $\mathbf{R}^{(k)}$ for $k = 1,2,\cdots,M!$ reporting sequences are specified. Now α_i for $i = 0,1,2,\cdots,M$ can be obtained using Cramer's rule.

$$\alpha_i = \frac{\det(\mathbf{E}_{i+1})}{\det\left[\mathbf{D}+\mathbf{W}+\mathbf{R}^{(k)}\right]} \quad (39)$$

Here $\mathbf{E}_i$ is obtained from $[\mathbf{D}+\mathbf{W}+\mathbf{R}]$ by replacing the i^{th} column of $[\mathbf{A}+\mathbf{B}+\mathbf{C}]$ by $\mathbf{E}$ for $i = 0,1,\cdots,M$. Then α_0 is:

$$\alpha_0 = \frac{\det(\mathbf{E}_1)}{\det\left[\mathbf{D}+\mathbf{W}+\mathbf{R}^{(k)}\right]} \quad (40)$$

Now the finish time for computing a unit load with M children processor is obtained as follows for any $k = 1,2,\cdots,M!$:

$$T_f(M) = \frac{\det(\mathbf{E}_1)}{\det\left[\mathbf{D}+\mathbf{W}+\mathbf{R}^{(k)}\right]} w_0 T_{cp} \quad (41)$$

There are $M!$ possible cases depending on the solution reporting sequence. In this section the finish time of each possible order has been obtained. It should be noted that for a specific reporting sequence recursive solutions for the staggered schedule load fractions can be developed as in [2].

IV. Optimality Proof

In this section, both arbitrary scheduling and staggered scheduling are assumed to follow the same reporting sequence. We will show that if all of the processors are staggered for a *given* reporting sequence, the finish time of staggered scheduling is always less than the finish time of arbitrary scheduling with the same reporting sequence.

Let the hyperplane H_0 in $(M+1)$-dimensions be a set of the form $\{\mathbf{x} : \mathbf{n}\cdot\mathbf{x} = 1\}$ where $\mathbf{n}$ is a 1 by $(M+1)$ ones vector.

Definition 1 *Any point on the hyperplane H_0 can be defined as $\widetilde{\boldsymbol{\alpha}}$ which is an arbitrary load distribution.*

$$\{\widetilde{\boldsymbol{\alpha}}:\ \mathbf{n}\cdot\widetilde{\boldsymbol{\alpha}} = 1\} \quad (42)$$

Here:

$$\widetilde{\boldsymbol{\alpha}} = (\widetilde{\alpha}_0, \widetilde{\alpha}_1, \widetilde{\alpha}_2, \cdots, \widetilde{\alpha}_i, \cdots, \widetilde{\alpha}_M) \quad (43)$$

The set of $\{\widetilde{\boldsymbol{\alpha}}:\ \mathbf{n}\cdot\widetilde{\boldsymbol{\alpha}} = 1\}$ is a normalization equation.

Consider staggered scheduling. There exist $M!$ cases depending on reporting sequences. Let $\mathbf{Q}^{(k)}$ for $k = 1,2,\cdots,M!$ be $\left[\mathbf{D}+\mathbf{W}+\mathbf{R}^{(k)}\right]$ in the linear equation (33). There exists a unique solution (through normalization) on the hyperplane H_0 given reporting sequence, $\mathbf{R}^{(k)}$, for any $k = 1,2,\cdots,M!$ satisfying equation (33).

Definition 2 *In the $(M+1)$-dimensional space, the unique solution of the linear equation (33) is defined as $\boldsymbol{\alpha}^k$ for any $k = 1,2,\cdots,M!$.*

$$\mathbf{Q}^{(k)}\cdot\boldsymbol{\alpha}^k = \mathbf{E} \quad (44)$$

Here:

$$\boldsymbol{\alpha}^k = \left(\alpha_0^k, \alpha_1^k, \cdots, \alpha_M^k\right) \quad (45)$$

Specifically, $\boldsymbol{\alpha}^k$ is staggered scheduling for a given reporting sequence, $\mathbf{R}^{(k)}$. Thus $\boldsymbol{\alpha}^k$ is the fraction of load for the staggered scheduling of the $\mathbf{Q}^{(k)}$ matrix.

Definition 3 *Let $\overline{\boldsymbol{\alpha}}$ be any point in a set of the form:*

$$\left\{\overline{\boldsymbol{\alpha}}:\ \overline{\alpha}_0 \neq \alpha_0^k \text{ and } \overline{\alpha}_i^k = m\alpha_i^k \text{ for all } i = 1,2,\cdots,M\right\} \quad (46)$$

Thus $\overline{\boldsymbol{\alpha}}$ can be defined as follows:

$$\overline{\boldsymbol{\alpha}} = \left(\overline{\alpha}_0, m\alpha_1^k, \cdots, m\alpha_i^k, \cdots, m\alpha_M^k\right) \quad (47)$$

Here, m is any arbitrary real number and $0 < m < \frac{1}{1-\alpha_0^k}$.

The upper limit of m is chosen so that the $\overline{\boldsymbol{\alpha}}$ is on a hyperplane. Each load fraction of the children processors in the load distribution $\overline{\boldsymbol{\alpha}}$, is proportional to the load fraction of the children processors in the load distribution $\boldsymbol{\alpha}^k$. Thus, the reporting time of all of the children processors with load distribution $\overline{\boldsymbol{\alpha}}$, is also staggered. In the load distribution, $\overline{\boldsymbol{\alpha}}$, it is not necessary that the computation of the originator processor and reporting of the last reporting processor finish at the same time.

Lemma 1 *For any reporting sequence $\mathbf{R}^{(k)}$ for $k = 1,2,\cdots,M!$, staggered scheduling has a smaller finish time than scheduling with load distribution, $\overline{\boldsymbol{\alpha}}$.*

Lemma 1 can be written as follow equation: The finish time of scheduling with $\overline{\boldsymbol{\alpha}}$ can be written as follows:

$$\begin{aligned} \overline{T}_f(M) &\geq m T_s^{(M)} \\ &> T_s^{(M)} = T_f(M) \end{aligned} \quad (48)$$

Here, $\overline{T}_f(M)$ is the finish time of scheduling with $\overline{\boldsymbol{\alpha}}$,and $T_f(M)$ is the finish time of staggered scheduling.

Any point on the hyperplane H_0 can be denoted as $\widetilde{\boldsymbol{\alpha}}$ which is an arbitrary load distribution.

$$\widetilde{\boldsymbol{\alpha}} = (\widetilde{\alpha}_0, \widetilde{\alpha}_1, \widetilde{\alpha}_2, \cdots, \widetilde{\alpha}_i, \cdots, \widetilde{\alpha}_M) \quad (49)$$

Let $\widetilde{\mathbf{E}}$ be the result from multiplying $\mathbf{Q}^{(k)}$ by $\widetilde{\boldsymbol{\alpha}}$

$$\mathbf{Q}^{(k)}\cdot\widetilde{\boldsymbol{\alpha}} = \widetilde{\mathbf{E}} \quad (50)$$

Here $\widetilde{\mathbf{E}}$ can be expressed as follows:

$$\widetilde{\mathbf{E}} = \begin{bmatrix} T_{cp} & \widetilde{e}_1 & \widetilde{e}_2 & \cdots & \widetilde{e}_M \end{bmatrix}' \quad (51)$$

Subtracting both side of equation (44) from (50)

$$\mathbf{Q}^{(k)}\cdot(\widetilde{\boldsymbol{\alpha}} - \overline{\boldsymbol{\alpha}}) = \widetilde{\mathbf{E}} - \overline{\mathbf{E}} \quad (52)$$

or

$$\mathbf{Q}^{(k)}\cdot\widehat{\boldsymbol{\alpha}} = \widehat{\mathbf{E}} \quad (53)$$

Here:

$$\widehat{\boldsymbol{\alpha}} = \widetilde{\boldsymbol{\alpha}} - \overline{\boldsymbol{\alpha}} \quad (54)$$

and

$$\widehat{\mathbf{E}} = \widetilde{\mathbf{E}} - \overline{\mathbf{E}} \quad (55)$$

$$= \begin{bmatrix} 0 & \widehat{e}_1 & \widehat{e}_2 & \cdots & \widehat{e}_M \end{bmatrix}' \quad (56)$$

Lemma 2 *If two distributions, $\widetilde{\boldsymbol{\alpha}}$ and $\overline{\boldsymbol{\alpha}}$ are on the hyperplane then at least, one of $\widehat{e}_i$ for $i = 1,2,\cdots,M$ is greater than or equal to zero.*

Lemma 3 *In a given sequence, if $\overline{\alpha}_0$ equals to $\widetilde{\alpha}_0$ then the load distribution with $\overline{\boldsymbol{\alpha}}$, has a smaller finish time than arbitrary load distribution, $\widetilde{\boldsymbol{\alpha}}$.*

Proof.

From Lemma 2, assume that $\widehat{e}_i$ for any $i = 1, 2, \cdots, M$ is greater than or equal to zero. Also from equation (53), $\widehat{e}_i$ can be expressed as multiplying $(\widetilde{\boldsymbol{\alpha}} - \overline{\boldsymbol{\alpha}})$ by the $(i+1)^{th}$ row of $\mathbf{Q}^{(k)}$.

$$\begin{aligned} \widetilde{e}_i &= \mathbf{q}_{i+1}^{(k)} \cdot \widetilde{\boldsymbol{\alpha}} - \mathbf{q}_{i+1}^{(k)} \cdot \overline{\boldsymbol{\alpha}} \\ &\geq 0 \end{aligned} \tag{57}$$

Here, $\mathbf{q}_{i+1}^{(k)}$ is defined as the $(i+1)^{th}$ row of $\mathbf{Q}^{(k)}$. The second term on the right side of the above equation can be expressed as equation (33).

$$\begin{aligned} \mathbf{q}_{i+1}^{(k)} \cdot \overline{\boldsymbol{\alpha}} &= -\overline{\alpha}_0 w_0 T_{cp} + m \sum_{j=1}^{i} \alpha_j^k z_j T_{cm} \\ &\quad + m\alpha_i^k w_i T_{cp} + m\alpha_i^k z_i T_{cm}^{sol} \\ &\quad + m \sum_{j=1, j\neq i}^{M} r_{i,j} \alpha_j^k z_j T_{cm}^{sol} \end{aligned} \tag{58}$$

Similarly, $\mathbf{q}_i^{(k)} \cdot \widetilde{\boldsymbol{\alpha}}$ can be expressed as follows:

$$\begin{aligned} \mathbf{q}_{i+1}^{(k)} \cdot \widetilde{\boldsymbol{\alpha}} &= -\widetilde{\alpha}_0 w_0 T_{cp} + \sum_{j=1}^{i} \widetilde{\alpha}_j z_j T_{cm} \\ &\quad + \widetilde{\alpha}_i w_i T_{cp} + \widetilde{\alpha}_i z_i T_{cm}^{sol} \\ &\quad + \sum_{j=1, j\neq i}^{M} r_{i,j} \widetilde{\alpha}_j z_j T_{cm}^{sol} \end{aligned} \tag{59}$$

Thus:

$$\begin{aligned} \widehat{e}_i &= -\widetilde{\alpha}_0 w_0 T_{cp} + \overline{\alpha}_0 w_0 T_{cp} \\ &\quad + \left[\sum_{j=1}^{i} \widetilde{\alpha}_j z_j T_{cm} + \widetilde{\alpha}_i w_i T_{cp} + \widetilde{\alpha}_i z_i T_{cm}^{sol} \right. \\ &\quad \left. + \sum_{j=1, j\neq i}^{M} r_{i,j} \widetilde{\alpha}_j z_j T_{cm}^{sol} \right] \\ &\quad - m \cdot \left[\sum_{j=1}^{i} \alpha_j^k z_j T_{cm} + \alpha_i^k w_i T_{cp} + \alpha_i^k z_i T_{cm}^{sol} \right. \\ &\quad \left. + \sum_{j=1, j\neq i}^{M} r_{i,j} \alpha_j^k z_j T_{cm}^{sol} \right] \end{aligned} \tag{60}$$

Note that $\widehat{e}_i$ is greater than or equal to zero and $\overline{\alpha}_0$ equals to $\widetilde{\alpha}_0$ by assumption. Thus:

$$\begin{aligned} &\left[\sum_{j=1}^{i} \widetilde{\alpha}_j z_j T_{cm} + \widetilde{\alpha}_i w_i T_{cp} + \widetilde{\alpha}_i z_i T_{cm}^{sol} \right. \\ &\quad \left. + \sum_{j=1, j\neq i}^{M} r_{i,j} \widetilde{\alpha}_j z_j T_{cm}^{sol} \right] \end{aligned} \tag{61}$$

$$\begin{aligned} \geq\ & m \cdot \left[\sum_{j=1}^{i} \alpha_j^k z_j T_{cm} + \alpha_i^k w_i T_{cp} + \alpha_i^k z_i T_{cm}^{sol} \right. \\ &\quad \left. + \sum_{j=1, j\neq i}^{M} r_{i,j} \alpha_j^k z_j T_{cm}^{sol} \right] \end{aligned} \tag{62}$$

The above inequality may be summarized as follows:

$$\widetilde{T}_s^{(M)} > \overline{T}_s^{(M)} \tag{63}$$

Recall that the finish time of arbitrary load scheduling:

$$\widetilde{T}_f(M) = \max\left[\widetilde{\alpha}_0 w_0 T_{cp},\ \widetilde{T}_s^{(M)} \right] \tag{64}$$

If the finish time with distribution $\overline{\boldsymbol{\alpha}}$ is determined by $\overline{T}_s^{(M)}$:

$$\overline{T}_f(M) = \overline{T}_s^{(M)} \tag{65}$$

Then, the following result can be obtained from equation (63) and (64).

$$\widetilde{T}_f(M) \geq \widetilde{T}_s^{(M)} \geq \overline{T}_s^{(M)} = \overline{T}_f(M) \tag{66}$$

If the finish time with distribution $\overline{\boldsymbol{\alpha}}$ is determined by $\overline{\alpha}_0 w_0 T_{cp}$:

$$\overline{T}_f(M) = \overline{\alpha}_0 w_0 T_{cp} \tag{67}$$

Then, from equation (64),

$$\widetilde{T}_f(M) \geq \widetilde{\alpha}_0 w_0 T_{cp} = \overline{\alpha}_0 w_0 T_{cp} = \overline{T}_f(M) \tag{68}$$

Therefore, the finish time, $\overline{T}_f(M)$, is less than that of arbitrary scheduling.

$$\widetilde{T}_f(M) \geq \overline{T}_f(M) \tag{69}$$

■

Theorem 4 *Staggered load scheduling has a smaller finish time than arbitrary load scheduling.*

Proof. From inequality (48) and (69) the following result is obtained:

$$T_f(M) \leq \overline{T}_f(M) \leq \widetilde{T}_f(M) \tag{70}$$

■

V. Conclusion

This work is significant in using the arbitrary schedule concept for scheduling in much the same way that the general distribution is used in queueing theory. The proof methodology used here can be applied to other divisible load optimality problems.

References

[1] V. Bharawaj, D. Ghose and T.G. Robertazzi, "Divisible Load Theory: A New Paradigm for Load Scheduling in Distributed Systems," in special issue of *Cluster Computing* on Divisible Load Scheduling (D. Ghose and T. Robertazzi, editors), vol. 6, no. 1, Jan. 2003, pp. 7-17.

[2] V. Bharadwaj, D. Ghose, V. Mani and T. G. Robertazzi, *Scheduling Divisible Loads in Parallel and Distributed Systems*, IEEE Computer Society Press (now distributed by Wiley), 1996.

[3] J. Sohn and T.G. Robertazzi, "Optimal Load Sharing for a Divisible Job on a Bus Network," *IEEE Transactions on Aerospace and Electronic System*, vol. 32, no. 1, Jan. 1996, pp. 34-40.