STONY BROOK UNIVERSITY

CEAS Technical Report CEAS-832

Simple Performance Bounds for Multicore and Parallel Channel Systems

Carlos Fernando Gamboa and Thomas G. Robertazzi

July 19, 2010

Abstract

A simple modification of existing divisible load scheduling algorithms, boosting link speed by M for M parallel channels per link, allows time optimal load scheduling and performance prediction for parallel channel systems. The situation for multicore models is more complex but can be handled by a substitution involving equivalent processor speed. These modifications yield upper bounds on such parallel systems' performance. This concept is illustrated for ideal single level (star) tree networks under a variety of scheduling policies. Less than ideal parallelism can also be modeled though mechanisms of inefficiency require further research.

1 Introduction

Recent technological trends have included the use of multiple cores on processor chips [1,33] and also the use of parallel communication channels [2] between two computing nodes. There is a large body of literature (more than one hundred journal papers from 1988 to the present) on the mathematics of scheduling networked computing, called divisible load theory (DLT). With a slight modification of existing mathematical DLT models, they can be applied to analyze these two technologies.

The modification for the case of parallel channels is to replace the inverse communication speed parameter of a single link, z, by z/M. where M is the number of channels per link. The case of modeling multicore systems is more complex and the inverse computing speed of a (virtual) processor can not be replaced by w/G where G is the number of cores per virtual processor, in general. However replacing inverse processing speed variables by equiavlent processing speeds allows the performance of ideal multicore systems to be calculated.

These modifications allow divisible load theory to provide performance bounds for the ideal situation: a processor node has G cores and a link has M channels per link. Elaborating on this point is the purpose of this paper. We note that future research can model multicore and parallel channels where less than an ideal speedup is achieved, leading to predications of less than ideal performance.

1.1 Divisible Load Theory Models

There has been a steady amount of research in divisible load theory since the original work of Cheng and Robertazzi [6] and Agrawal and Jagadish [31] in

1988. Most of these studies develop an efficient load distribution strategy and protocol in order to achieve optimal processing time in networks/grids with a single root processor. The optimal solution is obtained by forcing the processors over a network to all stop processing at the same time instant. Intuitively, a proof by contradiction holds that this is done because the solution could be improved by transferring load if some processors were idle while other are still busy [7]. Studies for network topologies including linear daisy chains, tree and bus networks using a set of recursive equations were presented in [6,8,9] respectively. There has been further work in terms of load distribution policies for hypercubes [10] and mesh networks [11]. The concept of equivalent networks [12] was developed for complex networks such as multilevel tree networks. Research has also considered scheduling policy with multi-installment [13], multi-round algorithms [14], independent task scheduling [15], fixed communication charges [16], detailed parameterizations and solution reporting time optimization [17] and combinatorial optimization [18]. Though divisible load theory is fundamentally a deterministic theory, it has been shown that there is some mathematical equivalence to Markov chain models [19].

There is a smaller amount of literature on divisible load modeling with multiple sources. A 2002 paper on multi-source load distribution attempts to integrate Markovian queueing theory and divisible load scheduling theory [20]. In 2003 the authors in [21] studied two source grid scheduling with memory capacity constraints. Also studied is two source grid scheduling in a 2009 journal paper [30]. Researchers in 2005 investigated [22] the use of linear programming to maximize throughput for large grids with multiple loads/sources. In 2005, other researchers [23] created a numerical solution for a linear daisy chain network with load originating at the ends of the chain. Mathematical programming solutions and flow structure in multi-source problems was studied in 2006 [24]. Recently some workers used a partitioning scheme mentioned in [24] to develop interesting multisource partitioning and load distribution algorithms in [29].

Surveys of divisible load theory appear in [3,4,5,27,28,32].

1.2 This Paper

In section II a review of the basic single level tree model analysis for the the three scheduling strategies to be used in this paper is presented. Section III discusses notation for parallel link systems speedup expressions and section IV contains the actual expressions. Section V presents results for single level trees under two scheduling strategies for multicore systems. Section VI

discusses performance results. The conclusion and an open question appears in section VII.

2 The model

2.1 Notation and overview

Due to its realistic and tractable nature, DLT models and analysis are a suitable tool to be able to model interactions among different cores located on a chip. In this section are presented different DLT scheduled policies that will be used in this study originally presented in [32]. Contrary to the methodology used initially in [32] to obtain optimal distribution of loads on a per processor basis, a different approach to handle the mathematical relationships among processors is proposed. The focus is to obtain the speedup expression for a network topology.

A single level tree network topology is shown in Fig. 1

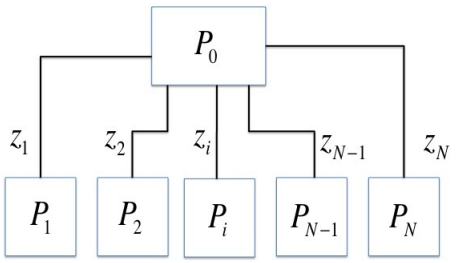

Figure 1: Single level tree network

This network topology considers only one channel of communication per root processor to children processor pair.

The variables or parameters used on this model are:

α_i: The load share fraction assigned to the i^{th} link-processor pair.

w_i: The inverse of the computing speed of the i^{th} processor.

z_i: The inverse of the link 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_i: Is the total time measured from the beginning of the scheduling process up to the end of the computation of the data by the ith processor.

T_f: Is the time when the last processor finishes reporting. T_f=max(T_1,T_2,...,T_N)

Three different scheduling protocols will be reviewed for this network topology. The mathematical representation obtained for this fundamental model will allow us to extrapolate it to two different network topologies that are envisioned that could be used for the design of M channels per link and G cores per virtual processor systems.

2.2 Sequential distribution, staggered start

As presented in different studies, in this paper Gantt chart-like timing diagrams for modeling the load distribution in the network are used. The horizontal axis represents time, communication time is presented above the axis and computation time is presented below the axis.

In order to find the optimal load distribution on each processor all processors need to finish at the same time [3,7].

$$T_f(P_0) = T_f(P_1) \tag{1}$$
$$T_f(P_2) = T_f(P_3) \tag{2}$$

$$T_f(P_{N-1}) = T_f(P_N) \tag{3}$$

The equations below state that the communication and processing time on a processor is equal to the processing time of the next processor.

$$\begin{aligned}
\alpha_0 w_0 T_{cp} &= \alpha_1 z_1 T_{cm} + \alpha_1 w_1 T_{cp} \\
\alpha_1 w_1 T_{cp} &= \alpha_2 z_2 T_{cm} + \alpha_2 w_2 T_{cp} \\
\alpha_i w_i T_{cp} &= \alpha_{i+1} z_{i+1} T_{cm} + \alpha_{i+1} w_{i+1} T_{cp} \\
\alpha_{N-1} w_{N-1} T_{cp} &= \alpha_N z_N T_{cm} + \alpha_N w_N T_{cp}
\end{aligned}$$

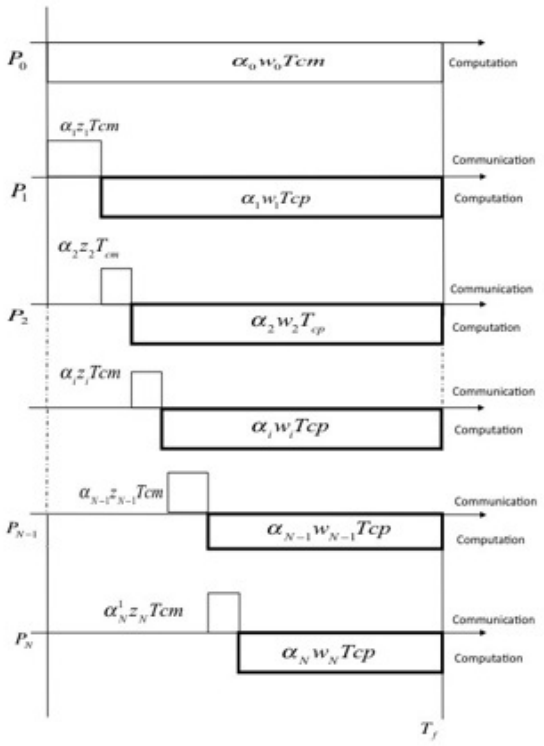

Figure 2: Timing diagram of single level tree with sequential distribution and staggered start

The normalization equation for N+1 processor is :

$$\alpha_0 + \alpha_1 + \alpha_2 + \alpha_3 + \cdot + \alpha_i + \cdot + \alpha_{N-1} + \alpha_N = 1 \tag{4}$$

Expressing this equation in terms S_i

$$\alpha_0 = \alpha_1 S_0 \tag{5}$$
$$\alpha_1 = \alpha_2 S_1 \tag{6}$$
$$\alpha_i = \alpha_{i+1} S_i \tag{7}$$
$$\alpha_{N-1} = \alpha_N S_{N-1} \tag{8}$$

where

$$S_i = \frac{(z_{i+1}T_{cm} + w_{i+1}T_{cp})}{w_i T_{cp}} \tag{9}$$

After solving the previous equation system for α_0 with the normalization equation the following expression is obtained:

$$\alpha_0 = \frac{1}{1 + \sum_{i=0}^{N-1} \prod_{j=0}^{i} \frac{1}{S_j}} \tag{10}$$

This study will be focus on the speedup metric which is defined as the ratio of computation time on one processor to the computation time on the entire N children network. Specifically the speedup will be studied for a homogeneous single level tree networks. So it is intended to measure the parallel processing advantage using the speedup relationship of conventional DLT theory as:

$$Speedup = \frac{T_{f0}}{T_{fN}} \tag{11}$$

Where T_{f0} represents the time processing the entire load on one processor so that α_0 equals to 1. Thus,

$$T_{f0} = \alpha_0 w_0 Tcp \tag{12}$$

$$T_{f0} = 1 \cdot w_0 Tcp \tag{13}$$

and

$$T_{fN} = \frac{1}{1 + \sum_{i=0}^{N-1} \prod_{j=0}^{i} \frac{1}{S_j}} w_0 Tcp \tag{14}$$

Here T_{fN} represents the finish time in an N children network as in the single level tree network presented in Fig. 1.

As mentioned before, for the speedup for a homogeneous single level tree network, in that particular case every $S_i = S$ for i from 1=N so link speeds are equal on the network and the processor speed as well. Thus equation (14) can be rewritten as:

$$T_{fN} = \frac{1}{1 + \frac{1}{S_0}(1 + \sum_{i=1}^{N-1} \frac{1}{S^i})} w_0 Tcp \tag{15}$$

And the corresponding speedup will be:

$$Speedup = 1 + \frac{1}{S_0}(1 + \sum_{i=1}^{N-1} \frac{1}{S^i}) \tag{16}$$

2.3 Simultaneous distribution, staggered start

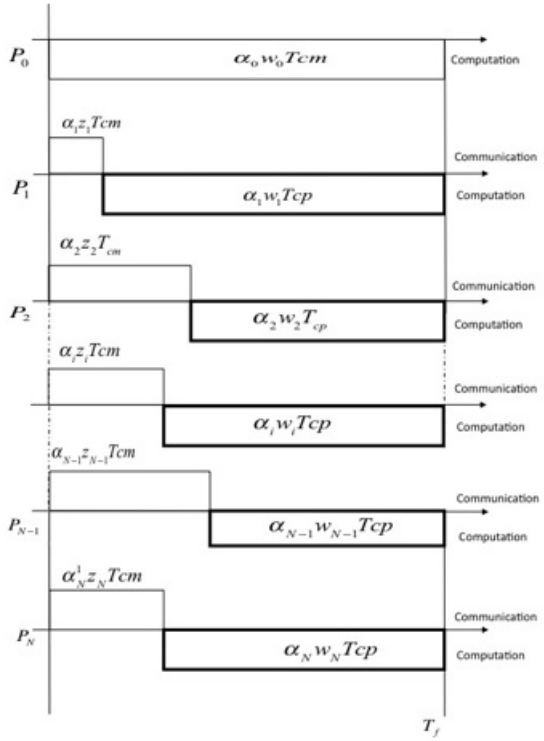

Figure 3: Timing diagram of single level tree with simultaneous distribution and staggered start

Different than the previous protocol, the processors now simultaneously receive the data and only start to process it as soon as each processor receives it entire load assignment Fig. 3.

Again, for a time optimal load allocation it is assumed that all of the processors stop computing at the same time. The equations that describe this model are:

$$\alpha_0 w_0 T_{cp} = \alpha_1 z_1 T_{cm} + \alpha_1 w_1 T_{cp}$$
$$\alpha_1 w_1 T_{cp} + \alpha_1 z_1 T_{cm} = \alpha_2 z_2 T_{cm} + \alpha_2 w_2 T_{cp}$$
$$\alpha_i w_i T_{cp} + \alpha_i z_i T_{cm} = \alpha_{i+1} z_{i+1} T_{cm} + \alpha_{i+1} w_{i+1} T_{cp}$$
$$\alpha_{N-1} w_{N-1} T_{cp} + \alpha_{N-1} z_{N-1} T_{cm} = \alpha_N z_N T_{cm} + \alpha_N w_N T_{cp}$$

Equation (4) can be used as this is the normalization equation.

Expressing the previous equation system in terms of g_1 and s_i for i from 1 to N-1.

$$\alpha_0 = \alpha_1 g_1 \tag{17}$$
$$\alpha_1 = \alpha_2 S_1 \tag{18}$$
$$\alpha_i = \alpha_{i+1} S_i \tag{19}$$
$$\alpha_{N-1} = \alpha_N S_{N-1} \tag{20}$$

where

$$S_i = \frac{(z_{i+1} T_{cm} + w_{i+1} T_{cp})}{z_i T_{cm} + w_i T_{cp}} \tag{21}$$

and

$$g_1 = \frac{(z_1 T_{cm} + w_1 T_{cp})}{w_0 T_{cp}} \tag{22}$$

The corresponding, solution time optimal, fraction of load for this particular schedule protocol can be found, in a manner similar to that in the previous section, as

$$\alpha_0 = \frac{1}{1 + \frac{1}{g_1}(1 + \sum_{i=1}^{N-1} \prod_{j=1}^{i} \frac{1}{S_j})} \tag{23}$$

The general expression for speedup will be,

$$Speedup = \frac{1}{\frac{1}{1+\frac{1}{g_1}(1+\sum_{i=1}^{N-1} \prod_{j=1}^{i} \frac{1}{S_j})}} \tag{24}$$

When a homogenous network is considered the speedup can be expressed as

$$Speedup = \frac{1}{\frac{1}{1+\frac{1}{g_1}(1+\sum_{i=1}^{N-1}\prod_{j=1}^{i}\frac{1}{1})}} \tag{25}$$

Simplifying the above equation,

$$Speedup = 1 + \frac{1}{g_1}(1 + N - 1) \tag{26}$$

The final expression will be

$$Speedup = 1 + \frac{1}{g_1}(N) \tag{27}$$

2.4 Simultaneous distribution, simultaneous start

Fig. 4 presents the last protocol considered here.

In this case the processors are able to process the load as soon as they receive the initial transmission. It is assumed that all finish at the same time to achieve a time optimal distribution of load.

$$\begin{aligned}
\alpha_0 w_0 T_{cp} &= \alpha_1 w_1 T_{cp} \\
\alpha_1 w_1 T_{cp} &= \alpha_2 w_2 T_{cp} \\
\alpha_i w_i T_{cp} &= \alpha_{i+1} w_{i+1} T_{cp} \\
\alpha_{N-1} w_{N-1} T_{cp} &= \alpha_N w_N T_{cp}
\end{aligned}$$

Note, that it is assumed that communication speed is faster than computation speed for each processor link pair so,

$$\alpha_i z_i T_{cm} < \alpha_i w_i T_{cp}$$

By expressing the previous equation system in terms of f_i the previous equation system can be written as,

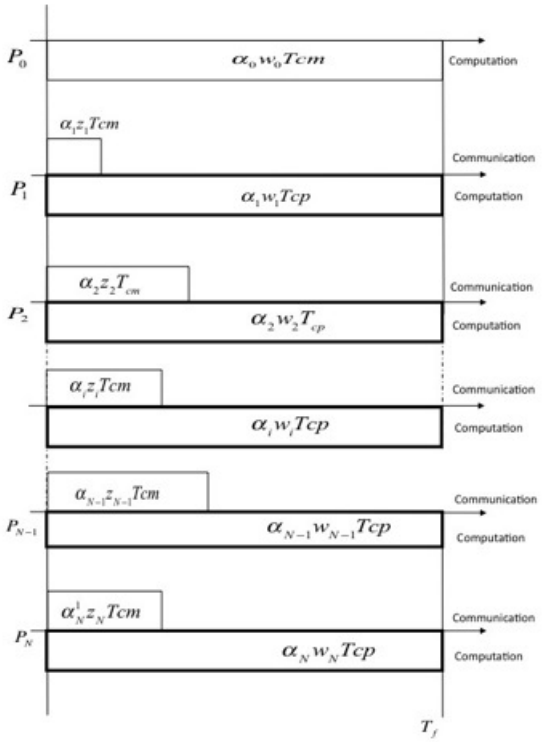

Figure 4: Timing diagram of single level tree with simultaneous distribution and simultaneous start

$$\alpha_0 = \alpha_1 f_0 \tag{28}$$

$$\alpha_1 = \alpha_2 f_1 \tag{29}$$

$$\alpha_i = \alpha_{i+1} f_i \tag{30}$$

$$\alpha_{N-1} = \alpha_N f_{N-1} \tag{31}$$

where

$$f_i = \frac{(w_{i+1} T_{cp})}{w_i T_{cp}} \tag{32}$$

The optimal load fraction assigned to the root processor is

$$\alpha_0 = \frac{1}{1 + \sum_{i=0}^{N-1} \prod_{j=0}^{i} \frac{1}{f_j}} \tag{33}$$

Here the speedup will be,

$$Speedup = (1 + \sum_{i=0}^{N-1} \prod_{j=0}^{i} \frac{1}{f_j}) \tag{34}$$

Considering a homogeneous network, the equation below represents the speedup for the simultaneous distribution, simultaneous start protocol. In this case it is assumed that all of the processors on the network are similar with the exception of the root processor which has inverse computing speed w_0. Thus the speedup is,

$$Speedup = 1 + \frac{1}{f_0}(1 + \sum_{i=1}^{N-1} \prod_{j=1}^{i} 1) \tag{35}$$

After simplifying the above equation,

$$Speedup = 1 + \frac{1}{f_0}(N) \tag{36}$$

3 M Parallel Channel per Link: Notation

In this section the specific notation used in developing speedup expressions for M parallel channels per link in single level tree topologies is presented. The actual speedup expressions appear in the next section.

Fig. 5 shows a single level tree architecture consistent with N+1 processor and M channels per link (χ) per processor pair. In this model the root core will be able to distribute the load assigned to every core in parallel fashion using M available channels (χ). Every connection pair is similar, thus there is always the same amount of channels available for every root core child core pair in the network.

The variables used in this model need to be presented at this time;

z_j: The inverse of the link speed of the j^{th} link.

w_i: The inverse of the computing speed of the i^{th} processor.

$\chi_{i,j}$: The z_j / M inverse of the link speed with M channels per the j^{th} link and i^{th} processor.

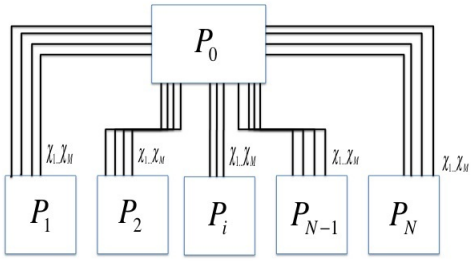

Figure 5: Single level tree network M parallel interconnection channels

j: Represents the link number between root processor and i^{th} children processor.

A homogeneous network topology is considered during this study. Thus all of the communication channels on the network are the same. In addition all of the cores used on the system are identical as well. Thus

$$\chi_{i,j} = \frac{z_{ij}}{M} \tag{37}$$

where

i=$1 \cdots N$ number of processors .

j=$1 \cdots M$ parallel channels per processor pair.

For this particular case the mathematical expressions obtained to described the optimal load assignment on the past section can be extrapolated for the network architecture shown in Fig. 5 by adjusting the proper parameter, in this case the link speed parameter.

4 Speedup for M parallel channels per link

In the past sections different speedup expressions were reviewed for different schedule protocols for the network architecture presented in Fig. 1. These expressions were defined in terms of dummy variables which involved the link speed and the processor speed. By redefining the dummy parameter according to the new network topology with M parallel channels per link,

new speedup expressions can be found. The following sections will show the speedup for the network topology presented on Fig. 5.

4.1 Speedup for sequential distribution staggered start with M parallel channels per link

Equation (16) shows the speedup for a network topology of one link to one processor. The dummy parameter defined in there is S_i,

$$S_i = \frac{(z_{i+1}T_{cm} + w_{i+1}T_{cp})}{w_i T_{cp}} \tag{38}$$

When considering this parameter for M parallel channels per link the expression needs to be adjusted. The parameter adjusted is the speed of the link z_i. Using equation (37), B will be the new name for the dummy variable for this protocol on the new architecture.

$$B_i = \frac{(\frac{z_{ji}}{M}T_{cm} + w_{i+1}T_{cp})}{w_i T_{cp}} \tag{39}$$

In the case of using a homogeneous network topology where the M channel speeds are equal and the speed of processors is the same on all the cores in the network, equation (40) can be used. Thus,

$$B_i = \frac{(\frac{z}{M}T_{cm} + wT_{cp})}{wT_{cp}} \tag{40}$$

and

$$B_0 = \frac{(\frac{z}{M}T_{cm} + wT_{cp})}{w_0 T_{cp}} \tag{41}$$

The speedup found for this protocol and this architecture after equation (16) is:

$$Speedup = 1 + \frac{1}{B_0}(1 + \sum_{i=1}^{N-1} \frac{1}{B^i}) \tag{42}$$

4.2 Speedup for simultaneous distribution staggered start with M parallel channels per link

Following the same methodology as in section (II) the speedup expression for this protocol will be obtained from equation (24). The dummy variables used to relate link speed and processor speed for this protocol were:

$$B_i = \frac{(\chi_{(i+1),j} T_{cm} + w_{i+1} T_{cp})}{\chi_{i,j} T_{cm} + w_i T_{cp}} \tag{43}$$

and

$$G_1 = \frac{(\chi_{1,j} T_{cm} + w_1 T_{cp})}{w_0 T_{cp}} \tag{44}$$

Equation (24) yields equation (45) representing the speedup equation using the new dummy variables according to this network topology,

$$Speedup = \frac{1}{\frac{1}{1+\frac{1}{G_1}(1+\sum_{i=1}^{N-1}\prod_{j=1}^{i}\frac{1}{B_j})}} \tag{45}$$

For a homogeneous network case from equation (27) and substituting the parameters,

$$Speedup = 1 + \frac{1}{G_1}(N) \tag{46}$$

4.3 Speedup for simultaneous distribution simultaneous start with M parallel channels per link

The last scheduling protocol considered is for the network topology presented on Fig. 5. Following the same methodology used before, the dummy parameters represent the relationship among processors and links in the case of M=1, the conventional case considered in section (2.4). As can be seen the dummy variable does not depend on speed of the link. Thus, the speedup for this network architecture remains the same.

5 Speedup for G parallel cores per virtual processor

The case of G parallel cores per processor is somewhat more involved than the case of M channels per link.

There are two considerations. As an example, consider the single level tree (star) topology of Fig. 6. While virtual processor nodes (children nodes) can be made faster and faster by adding more cores, eventually if this process continues the single link speed would be slower than than the increased virtual processor speed. Thus load distribution to achieve a parallel processing advantage would make no sense [3]. For there to be an actual system speedup the link speed must eventually be increased, perhaps by adding parallel channels, as well as by simply increasing the number of cores.

The second consideration involves levels of load distribution. Again, as an example consider the network of Fig. 6. Load must be distributed both from the root node to the virtual processor children nodes and within the virtual processor nodes. What are the possibilities for load distribution? There are two large families of load distribution policies: sequential and simultaneous.

If the internal virtual processor load distribution is sequential (i.e. load is distributed to one core at a time) then an G core virtual processor is not G times faster than a single core. This is because it well known [3] that under sequential load distribution speedup saturates with an increasing number of cores. On the other hand, if the load distribution is simultaneous (i.e. load is distributed to each core simultaneously (in parallel)) then it is well known that the system speedup is of the form 1+kG where k is a constant depending on the system parameters of the specific version of simultaneous load distribution used (see equations (27) and (46) and also reference [32]). This is a linear, not a proportional, relationship. Thus in both cases the single core speed can not be simply replaced by G times the core speed for G cores per virtual processor. Similar arguments can be made at each level of tree distribution.

However by substituting the equivalent processing speed [3,12,32] of the ith subnetwork for w_i, ideal performance for scheduling policies can be evaluated. Here we have equivalent elements in a linear system sense: an "equivalent" element exactly mimicing the behavior of a larger sub-network. Two examples of the use of equivalent elements are given below.

The network topology considered in Fig. 6 consists of a root processor P_0 that distributes or assigns load to N virtual equivalent processors which

are composed of G different cores each. The load assignment among root processor and virtual processor can be done using the previous schedule protocols from the classical DLT sections (2.2), (2.3) and (2.4). Within the virtual processor the cores are organized in a single level tree fashion, DLT is used to distribute the load among cores on every virtual processor. It is assumed that every load assignment is arbitrarily partitionable.

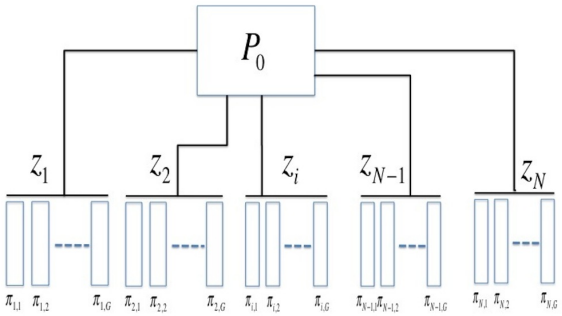

Figure 6: Single level tree network G parallel cores

Where,

P_0: Is the root processor with inverse computing speed w_0.

g: Represents the core number that belongs to the i^{th} virtual processor.

G: Represents total number of cores per virtual processor.

$\pi_{i,g}$: The inverse of the computing speed of the g^{th} core on the i^{th} virtual processor.

z_i: The inverse of the link speed of the i^{th} link per virtual processor.

$z_{i,g}$: The inverse of the link speed of the g^{th} link between the g^{th} core and the root distributor in a virtual processor.

5.1 Speedup for sequential distribution and staggered start with M parallel cores

Using similar methodology as the one presented in previous section and referring to Equation (16) which shows the speedup for a network topology

one link to one processor. The dummy parameter defined in there is S_i,

$$S_i = \frac{(z_{i+1}T_{cm} + w_{i+1}T_{cp})}{w_i T_{cp}} \tag{47}$$

Let BP be the dummy parameter for G parallel cores distributed in N virtual processors for this protocol on the new architecture. As shown in equation (14)

$$Speedup = 1 + \sum_{i=0}^{N-1} \prod_{j=0}^{i} \frac{1}{BP_j} \tag{48}$$

where

$$BP_0 = \frac{(z_1 T_{cm} + weq_1 T_{cp})}{w_0 T_{cp}} \tag{49}$$

and for $i = 1$ to N virtual processors

$$BP_i = \frac{(z_{i+1}T_{cm} + weq_{i+1}T_{cp})}{weq_i T_{cp}} \tag{50}$$

The single level tree core network for the the virtual processor can be collapsed into one single node that will allow us to find the equivalent computation speed weq,

$$weq_i = \frac{1}{1 + \sum_{i=0}^{G-1} \prod_{j=0}^{i} \frac{1}{SC_j}} \tag{51}$$

Here G is the number of cores in every virtual processor and SC is the constant of the related interconnection links and cores on every virtual processor.

$$SC_g = \frac{(z_{i,g+1}T_{cm} + \pi_{i,g+1}T_{cp})}{\pi_{i,g} T_{cp}} \tag{52}$$

The sequential distribution staggered start policy is used also within the virtual processor to distribute load among cores processors.

5.2 Speedup for simultaneous distribution and simultaneous start with M parallel cores

Another scheduling protocol to assign load to the rest of the virtual processors is simultaneous distribution, simultaneous start. Equations (34) and (36) represent the speedup expressions for this protocol. When the incoming load on every virtual processor arrives the load is partitioned and assigned using the simultaneous distribution and simultaneous start presented on section (2.4).

The new dummy parameter fc_g is:

$$fc_g = \frac{(\pi_{i,(g+1)} T_{cp})}{\pi_{i,g} T_{cp}} \tag{53}$$

For this case the weq will be expressed as:

$$weq_i = \frac{1}{1 + \sum_{i=0}^{G-1} \prod_{j=0}^{i} \frac{1}{fc_j}} \tag{54}$$

Considering a homogeneous network the equation below represents equivalent computation speed weq for the simultaneous distribution simultaneous start protocol. In this case it is assumed that all of the cores in the network are similar with the exception of the root processor which is π_0. Thus the dummy parameter fc_g evaluated for values different than $g = 0$ will be one, while fc_0 with $g = 0$ will be,

$$fc_0 = \frac{\pi T_{cp}}{\pi_0 T_{cp}} \tag{55}$$

Thus by simplifying the weq_i using (55) it can be expressed as,

$$weq_i = \frac{1}{1 + \frac{1}{f_{co}}(1 + \sum_{i=1}^{G-1} \prod_{j=1}^{i} 1)} \tag{56}$$

After simplifying the above equation,

$$weq_i = \frac{1}{1 + \frac{1}{f_{co}} G} \tag{57}$$

To find the general speedup of the entire network topology for this protocol the equation (34) (which represents the standard speedup of this protocol) will need to be modified to take into account the virtual processors comprised of the different G cores. Thus,

$$Speedup = (1 + \sum_{i=0}^{N-1} \prod_{j=0}^{i} \frac{1}{CP_j}) \tag{58}$$

where CP is :

$$CP_0 = \frac{weq_1 T_{cp}}{w_0 T_{cp}} \tag{59}$$

and for $i = 1$ to N virtual processors

$$CP_i = \frac{(weq_{i+1} T_{cp})}{weq_i T_{cp}} \tag{60}$$

In the same way for the homogenous case equation (36) can be used to obtain the general expression for speedup for this protocol when all the virtual processors have the same equivalent computation speed weq leading to equation (61).

$$Speedup = 1 + \frac{1}{CP_0}(N) \tag{61}$$

Note that $weq_{i+1} = weq$ in the homogeneous case, by substituting equation (59) into the equation (61) the speedup for this network topology can be obtained as:

$$Speedup = 1 + \frac{w_0 T_{cp}}{weq T_{cp}}(N) \tag{62}$$

Equations (57) and (62) are used to denote the speedup in terms of the G cores and N virtual processors, resulting in equation (63)

$$Speedup = 1 + w_0 \frac{(f_{co} + G)}{f_{co}} N \tag{63}$$

6 Performance Results

The scheduling protocols, Sequential Distribution Staggered Start and Simultaneous Distribution Staggered Start were simulated for a singe level tree network with M parallel channels per link with the following parameters:

$w_i = 100$ for i=1 to 10, 20 40.

$w_0 = 90$.

$z_i = 100$ for i=1 to 20.

$\pi_0 = 95$.

$\pi_i = 100$ for i=1 to 10.

$T_{cp} = 1$

T_{cm}= 1

In Fig. 7 are shown results for the sequential distribution and staggered start for single level tree M parallel links network topology. Three different networks size were simulated composed of 10, 20 and 40 processors. On the vertical axis the value of speedup is presented. On the other hand the horizontal axis presents the number of parallel links.

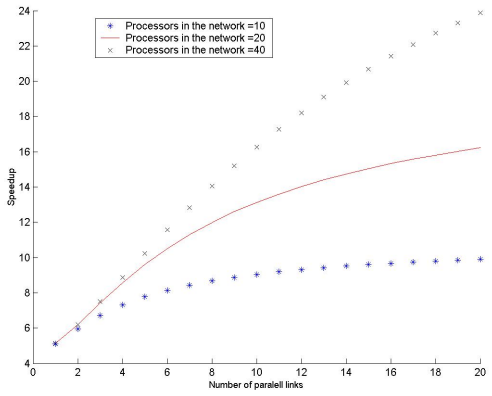

Figure 7: Speedup for a single level tree M parallel links with sequential distribution and sequential start

It can be seen that the speedup saturates as the number of links is increased. That is, the speedup is limited by computing power as the number

of links increases and the latency goes to zero. Also, the speedup increases with increasing numbers of processors.

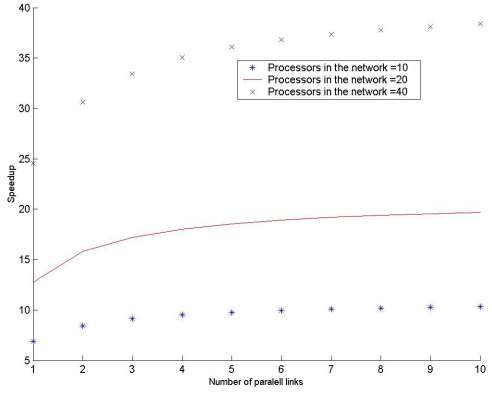

Figure 8: Speedup for a single level tree with M parallel links simultaneous distribution and staggered start

Fig. 8 represents the simulations results for the protocol with simultaneous distribution and staggered start. The speedup plot for the network of 10 processors, for instance, rapidly saturates to 10 after 5 parallel links. After this point adding extra links will not result in a gain as one approaches zero transmission latency and one is limited solely by processing speed. In addition the speedup is constrained by the network size, as shown on different plots for the network simulated.

Fig. 9 shows the simulation for a single level tree network with G parallel cores. The network topology simulated consisted on a single level tree network that uses the simultaneous distribution and simultaneous start to assign load from root to virtual nodes. This protocol is used to distribute load among cores on every virtual processor. On the vertical axis is shown the metric for speedup and on the horizontal axis are the number of cores simulated per virtual processor basis. The linear growth exhibited is in line with previous studies [32]. Notice the high values of the speedup for different network sizes.

7 Conclusion and an Open Question

The single level tree examples of this paper demonstrate the ease with which idealized (bounding) multicore and parallel channel performance can be cal-

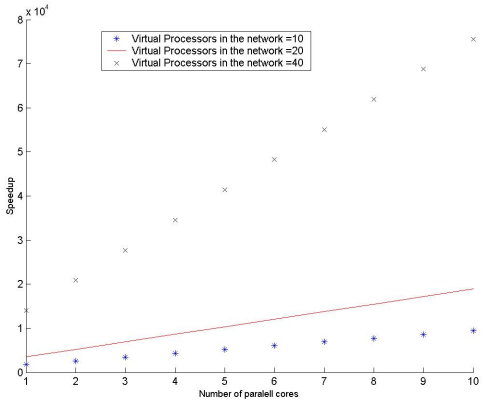

Figure 9: Speedup for a single level tree with G parallel cores simultaneous distribution and simultaneous start

culated for a wide variety of load distribution strategies and interconnection networks.

To obtain results for parallel systems with less than ideal performance, processing/link speeds in divisible load theory models could be replaced by values that are less than ideal speedups. The outstanding open research question though is creating accurate models of system features and behavior that lead to less than ideal performance. This is probably highly system and feature dependent. This problem should lead to a fair amount of interesting and useful future work.

References

[1] J. D.Ownes, W. J. Daily, R. Ho, D.N. Jayasimha, S. W. Keckler, L. Peh, *Research Challenges for ON-CHIP Interconnections Networks* IEEE Computer Society, pp. 272-1732, 2007.

[2] M. Moges and T.G. Robertazzi, "Wireless Sensor Networks: Scheduling for Measurement and Data Reporting," *IEEE Trans. on Aerospace and Electronic Systems*, vol. 42, no. 1, pp. 327-340, Jan. 2006.

[3] V. Bharadwaj, D. Ghose, V. Mani, and T.G. Robertazzi: *Scheduling Divisible Loads in Parallel and Distributed Systems.* IEEE Computer Society Press, Los Alamitos, CA, 1996.

[4] V. Bharadwaj, D. Ghose, T.G. Robertazzi, "Divisible Load Theory: A New Paradigm for Load Scheduling in Distributed Systems," *Cluster Computing*, vol. 6, pp. 7-18, 2003.

[5] T.G. Robertazzi, "Ten Reasons to Use Divisible Load Theory," *Computer*, vol. 36, pp. 63-68, 2003.

[6] Y.C. Cheng and T.G. Robertazzi, "Distributed Computation with Communication Delays," *IEEE Trans. on Aerospace and Electronic Systems*, vol. 22, pp. 60-79, 1988.

[7] J. Sohn and T.G. Robertazzi, "Optimal Divisible Load Sharing for Bus Networks," *IEEE Trans. on Aerospace and Electronic Systems*, vol. 32, pp. 34-40, 1996.

[8] Y.C. Cheng and T.G. Robertazzi, "Distributed Computation for a Tree Network with Communication Delays," *IEEE Trans. on Aerospace and Electronic Systems*, vol. 26, pp. 511-516, 1990.

[9] S. Bataineh and T.G. Robertazzi, "Bus Oriented Load Sharing for a Network of Sensor Driven Processors," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 21 pp. 1202-1205, 1991.

[10] J. Blazewicz and M. Drozdowski, "Scheduling Divisible Jobs on Hypercubes," *Parallel computing*, vol. 21, pp. 1945-1956, 1996.

[11] J. Blazewicz and M. Drozdowski, "The Performance Limits of a Two Dimensional Network of Load Sharing Processors," *Foundations of Computing and Decision Sciences*, vol. 21, pp. 3-15, 1996.

[12] T.G. Robertazzi, "Processor Equivalence for a Linear Daisy Chain of Load Sharing Processors," *IEEE Trans. on Aerospace and Electronic Systems*, vol.29, pp. 1216-1221, 1993.

[13] V. Bharadwaj, D. Ghose, V. Mani, "Multi-installment Load Distribution in Tree Networks with Delays," *IEEE Trans. on Aerospace and Electronic Systems*, vol. 31, pp. 555-567, 1995.

[14] Y. Yang, H. Casanova, "UMR: A Multi-Round Algorithm for Scheduling Divisible Workloads," *Proc. Int'l Parallel and Distributed Processing Symposium (IPDPS'03)*, 2003.

[15] O. Beaumont, A. Legrand, and Y. Robert, "Optimal Algorithms for Scheduling Divisible Workloads on Heterogeneous Systems," *12^{th} Heterogeneous Computing Workshops HCW'2003*, 2003.

[16] J. Blazewicz and M. Drozdowski, "Distributed Processing of Distributed Jobs with Communication Startup Costs," *Discrete Applied Mathematics*, vol. 76, pp. 21-41, 1997.

[17] A.L. Rosenberg, "Sharing Partitionable Workloads in Heterogeneous NOWs: Greedier is Not Better," *Proc. IEEE Int'l Conf. on Cluster Computing*, pp. 124-131, 2001.

[18] P.F. Dutot, "Divisible Load on Heterogeneous Linear Array," *Proc. of the Int'l Parallel and Distributed Processing Symposium (IPDPS'03)*, 2003.

[19] M. Moges and T. Robertazzi, "Optimal Divisible Load Scheduling and Markov Chain Models," *Proc. of the 2003 Conference on Information Sciences and Systems*, Baltimore, MD, 2003.

[20] K. Ko, and T. Robertazzi, "Scheduling in an Environment of Multiple Job Submissions," *Proc. of the 2002 Conf. on Information Sciences and Systems*, 2002.

[21] H. Wong, B. Veeravalli, D. Yu and T. Robertazzi, "Data Intensive Grid Scheduling: Multiple Sources with Capacity Constraint," *IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 2003)*, 2003.

[22] L. Marchal, Y. Yang, H. Casanova and Y. Robert, "A Realistic Network/Application Model for Scheduling Loads on Large-Scale Platforms," *Proc. of the Int'l Parallel and Distributed Processing Symposium*, 2005.

[23] T. Lammie and T. Robertazzi, "A Linear Daisy Chain with Two Divisible Load Sources," *Proc. of 2005 Conf. on Information Sciences and Systems*, 2005.

[24] D. Yu and T. Robertazzi, "Multi-Source Grid Scheduling for Divisible Loads," *Proc. of 2006 Conf. on Information Sciences and Systems*, 2006.

[25] D. Piriyakumar and C. Murthy, "Distributed Computation for a Hypercube Network of Sensor-Driven Processors with Communication Delays Including Setup Time," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 28, pp. 245-251, 1998.

[26] J. Hung and T. Robertazzi, "Scalable Scheduling for Clusters and Grids using Cut Through Switching," *International Journal of Computers and Applications*, vol. 26, pp. 147-156, 2004.

[27] M. Drozdowski, *Scheduling for Parallel Processing*, Springer, New York, 2009.

[28] H. Casanova, A. Legrand and Y. Robert, *Parallel Algorithms*, CRC Press, Boca Raton, Florida, 2009.

[29] J. Jingxi, B. Veeravalli and J. Weissman, "Scheduling Multi-Source Divisible Loads on Arbitrary Networks," *to appear IEEE Trans. on Parallel and Distributed Systems.*

[30] M.A. Moges, D. Yu and T.G. Robertazzi, "Grid Scheduling Divisible Load from Two Sources," *Computers and Mathematics with Applications*, pp. 1081-1092, 2009.

[31] R. Agrawal and H.V. Jgadish, "Partitioning Techniques for Large Grain Parallelism," *IEEE Transactions on Computers*, vol. 37, no. 12, pp. 1627-1634, 1988.

[32] T.G. Robertazzi, *Networks and Grids: Technology and Theory* Springer, NY, 2007.

[33] N.R. Tallent and J.M. Mellor-Crummey, "Identifying Performance Bottlenecks in Work Stealing Computations," *Computer* pp. 44-50, Dec. 2009.