Scheduling Divisible Loads with Nonlinear Communication Time

Kai Wang, and Thomas G. Robertazzi, *Fellow, IEEE*

Abstract—A scheduling model for single level tree networks with various distribution policies is studied where the communication time from the root to each node is nonlinear in the size of the load. An iterative method is implemented to solve the optimal load distribution. The difference between sub-linear and super-linear complexity is examined. Applications arise in the aerospace field.

Index Terms—Scheduling, Parallel and distributed systems, Divisible load, Nonlinear, Matrix computation

1 Introduction

For decades it has been realized that many algorithms of much practical interest have a computational time that is a nonlinear in the size of the algorithms input. This includes algorithms used in aerospace applications such as the fast Fourier transform, matrix operations, line detection using the Hough transform [1] and pattern recognition using 2D Hidden Markov models.

But a related question that has received much less attention is whether the transmission time of data moving over links between processing nodes can be nonlinear in the size of the data to be transmitted. Normally one would think this is not possible. If one doubles the amount of data to be transmitted one would think it should take twice as much time to transmit as half that amount of data. This intuition is based on the usual linear intuitive model of a channel. Naturally we are ignoring overhead such as packet headers in this consideration.

However there is another way of looking at things: indexing data transmission not by time but by data structural properties. For instance, if one transmits a square matrix and indexes data transmission by matrix (i.e. row/column) size, the transmission time is proportional to a square power of the size of the matrix. Alternately if one transmits a binary tree of data where each node holds x bytes and indexes data transmission by the size of the tree in levels, L, the transmission time is proportional to $2^L - 1$.

In this paper such nonlinear models of communication time operating either with linear or nonlinear models of computation is investigated. This is done in the context of divisible loads and divisible load scheduling. Divisible loads are perfectly partitionable loads that are a good model for parallel systems processing embarrassingly parallel loads consisting of voluminous amounts of data. That is, we assume that there are no precedence relationships in the processing. Divisible load scheduling techniques are used in this paper because of their tractability in order to make analytical progress. We pay particular attention to communication time that is a positive integer power law function of load size. However the approach can be generalized to communication time that is an arbitrary differentiable function of load size, as is shown below. The difference between sub-linear and super-linear complexity is examined. Finally, the equations presented in sections 2 and 3 are very generic. In reality for particular problems, methods of managing data yield specific variations on the mathematical model used.

Over a hundred and forty journal papers [2] describing the divisible load scheduling have been published since the original work of Cheng and Robertazzi in 1988 [3] and Agrawal and Jagadish [4] that year. The basic problem is to determine the optimal scheduling of load given the interconnection and processor network topology, scheduling policy, processor and link speeds, and computing and communication intensities. The aim is to finish processing in the shortest time by the optimal scheduling of the load taking into consideration the aspects of both computation and communication. This occurs if processors stop working at the same time. If not, the load can be transferred from busy to idle processors to improve the solution. This linear model is well suited for parallel and distributed computing because of its tractable and realistic characteristics [5].

Over the years, divisible load theory has been applied to various kinds of networks topologies including linear daisy chains, tree and bus networks using a set of recursive equations [3] [6]. Further studies have been made for hypercubes [7] and mesh networks [8]. The idea of equivalent networks [9] was developed for

• *K. Wang and T. G. Robertazzi are with the Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY, 11790.*
E-mail: kaiwang.sunysb@gmail.com

complex networks such as multilevel tree networks. Research has also examined scheduling policy with multi-installment [10], multi-round algorithms [11], independent task scheduling [12], fixed communication charges [13], detailed parameterizations and solution reporting time optimization [14], large matrix vector computations [15] and combinatorial optimization [16]. Recent new applications includes aligning biological sequences [21], aligning multiple protein sequences [22], optimizing parallel image registration [23], divisible MapReduce computations [24], multicore and parallel channel systems [25] and parallel video transcoding [26].

To the best of our knowledge, only nonlinear computation, not nonlinear communication, has been investigated to date using divisible load theory. The first to do so was Drozdowski and Wolniewicz [17] who demonstrated super-linear speedup by defining processing times as a piecewise linear (and thus nonlinear) function of the size of the input load for evaluating the memory hierarchy of a computer. These results were determined using mathematical programming. Later, Hung and Robertazzi [18] obtained analytically optimal load allocation and speedup for simultaneous (i.e. concurrent) load distribution for a quadratic nonlinearity. They also presented an iterative solution for sequential load distribution with a nonlinearity of arbitrary power. Suresh, Run, Kim et.al. [19] [20] used a mild assumption on the communication to computation speed ratio to present scheduling solutions for certain nonlinear divisible load problems including optimal sequencing and arrangement results. Beaumont, Larcheveque and Marchal examined when time savings are possible with nonlinear computation [28].

This paper is organized as follows. In section 1, the introduction was made. In section 2, we discuss the optimal scheduling for different distribution policies. In section 3, a specific example is presented. In section 4, we present the conclusion.

2 Optimal Scheduling under Different Distribution Policies

2.1 Sequential distribution, simultaneous start

We assume the time of writing to memory can be subsumed into the computation time. We consider single level trees in this paper as a basic starting point architecture. Note that if link speeds are homogenous, a single level tree under sequential distribution is equivalent to a bus. We also assume communication speed and computation speed are known though they can be estimated [27]. Consider now a single level tree network, sequential distribution (load is distributed from the root to each child in turn), simultaneous start policy (load reception and computation start at the same time). We also assume that the root does processing.

TABLE 1
Table of symbols

α_i	The load fraction assigned to the ith child processor from the root node.
w_i	The inverse of the computing speed of the ith child processor.
z_i	The inverse of the link speed of the link that connects ith processor to the root node.
T_{cp}	Computing intensity constant: The entire load is processed in w_iT_{cp} seconds by the ith child processor. We assume the time of writing to memory can be subsumed into the computation time.
T_{cm}	Communication intensity constant: The entire load can be transmitted in z_iT_{cm} seconds to the ith child processor from the root node.
χ	Communication integer power nonlinearity ($\chi > 1$)
y	Computation integer power nonlinearity ($y > 1$)

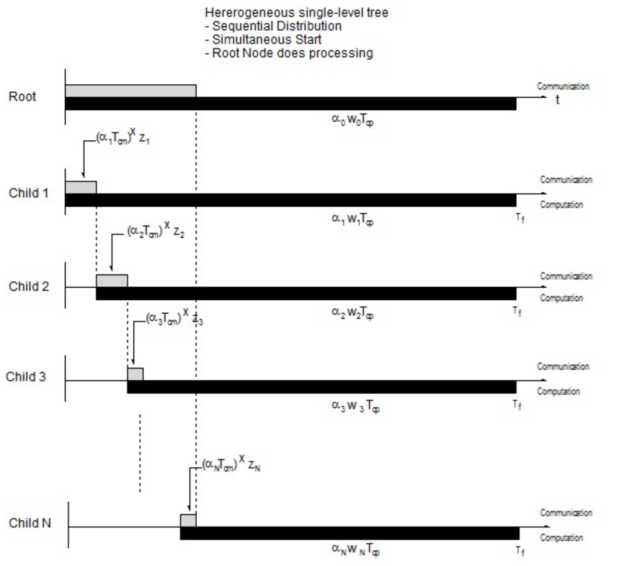


Fig. 1. Sequential distribution, simultaneous start

Certainly many nonlinear communication/computation functions are possible. In this paper, for purposes of demonstration we use a communication integer power nonlinearity χ ($\chi > 1$). Certainly also polynomial nonlinearities could be considered. From figure 1, one has the timing equations:

$$\alpha_0 w_0 T_{cp} = \alpha_1 w_1 T_{cp} \tag{1}$$

$$\alpha_1 w_1 T_{cp} = (\alpha_1 T_{cm})^{\chi} z_1 + \alpha_2 w_2 T_{cp} \tag{2}$$

$$\alpha_2 w_2 T_{cp} = (\alpha_2 T_{cm})^{\chi} z_2 + \alpha_3 w_3 T_{cp} \tag{3}$$

$$\cdots\cdots$$
$$\cdots\cdots$$
$$\cdots\cdots$$

$$\alpha_{N-1} w_{N-1} T_{cp} = (\alpha_{N-1} T_{cm})^{\chi} z_{N-1} + \alpha_N w_N T_{cp} \tag{4}$$

We assume in our timing diagram that communication time does not extend beyond computation time (ie. communication is generally faster than computation).

The normalization equation is:

$$\alpha_0 + \alpha_1 + \alpha_2 + \cdots + \alpha_N = 1 \tag{5}$$

These equations can be re-written as:

$$f_0 = \alpha_0 w_0 T_{cp} - \alpha_1 w_1 T_{cp} = 0 \tag{6}$$

$$f_1 = (\alpha_1 T_{cm})^{\chi} z_1 - \alpha_1 w_1 T_{cp} + \alpha_2 w_2 T_{cp} = 0 \tag{7}$$

$$f_2 = (\alpha_2 T_{cm})^{\chi} z_2 - \alpha_2 w_2 T_{cp} + \alpha_3 w_3 T_{cp} = 0 \tag{8}$$

$$\cdots\cdots$$
$$\cdots\cdots$$
$$\cdots\cdots$$

$$\begin{aligned} f_{N-1} = (\alpha_{N-1} T_{cm})^{\chi} z_{N-1} - \alpha_{N-1} w_{N-1} T_{cp} \\ + \alpha_N w_N T_{cp} = 0 \end{aligned} \tag{9}$$

$$f_N = \alpha_0 + \alpha_1 + \alpha_2 + \cdots + \alpha_N - 1 = 0 \tag{10}$$

and

$$\vec{f} = \begin{bmatrix} f_0(\alpha_0, \alpha_1, \alpha_2, \cdots, \alpha_N) \\ f_1(\alpha_0, \alpha_1, \alpha_2, \cdots, \alpha_N) \\ f_2(\alpha_0, \alpha_1, \alpha_2, \cdots, \alpha_N) \\ \vdots \\ f_{N-1}(\alpha_0, \alpha_1, \alpha_2, \cdots, \alpha_N) \\ f_N(\alpha_0, \alpha_1, \alpha_2, \cdots, \alpha_N) \end{bmatrix} = 0 \tag{11}$$

One can use the multivariate Newton's method to solve this set of timing equations. For the above equations, the Taylor expansion of f_i in the neighborhood of α:

$$\begin{aligned} f_i(\vec{\alpha} + \delta\vec{\alpha}) = f_i(\vec{\alpha}) + \delta\alpha_0 \frac{\partial f_i}{\partial \alpha_0}(\vec{\alpha}) + \cdots + \delta\alpha_N \frac{\partial f_i}{\partial \alpha_N}(\vec{\alpha}) \\ + O(|\delta\vec{\alpha}|^2) \approx f_i(\vec{\alpha}) + \nabla f_i(\vec{\alpha}) \cdot \delta\vec{\alpha} \end{aligned} \tag{12}$$

This can be rewritten as

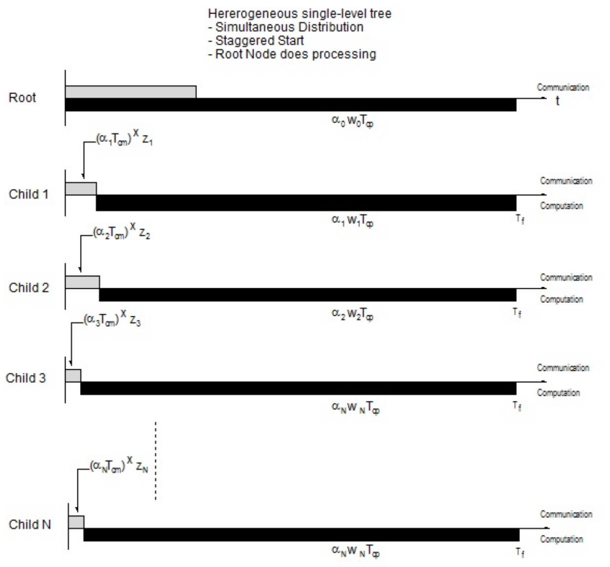
$$\vec{f}(\vec{\alpha} + \delta\vec{\alpha}) \approx \vec{f}(\vec{\alpha}) + \mathcal{J}_{\vec{f}}(\vec{\alpha}) \delta\vec{\alpha} = 0 \tag{13}$$

where $\mathcal{J}_{\vec{f}}(\vec{\alpha})$ is the Jacobian of $\vec{f} = (f_1, \cdots, f_N)^T$.

$$\mathcal{J}_{\vec{f}}(\vec{\alpha}) = \begin{bmatrix} \nabla f_0^T(\vec{\alpha}) \\ \nabla f_1^T(\vec{\alpha}) \\ \nabla f_2^T(\vec{\alpha}) \\ \vdots \\ \nabla f_N^T(\vec{\alpha}) \end{bmatrix} = \begin{bmatrix} \frac{\partial f_0}{\partial \alpha_0}(\vec{\alpha}) & \cdots & \frac{\partial f_0}{\partial \alpha_N}(\vec{\alpha}) \\ \frac{\partial f_1}{\partial \alpha_0}(\vec{\alpha}) & \cdots & \frac{\partial f_1}{\partial \alpha_N}(\vec{\alpha}) \\ \frac{\partial f_2}{\partial \alpha_0}(\vec{\alpha}) & \cdots & \frac{\partial f_2}{\partial \alpha_N}(\vec{\alpha}) \\ \vdots & \vdots & \vdots \\ \frac{\partial f_N}{\partial \alpha_0}(\vec{\alpha}) & \cdots & \frac{\partial f_N}{\partial \alpha_N}(\vec{\alpha}) \end{bmatrix} \tag{14}$$

One can first make a guess of the solution for the αs and then iterate the relation below until it converges to a solution:

$$\vec{\alpha}^{k+1} = \vec{\alpha}^k - \mathcal{J}^{-1}(\vec{\alpha}^k) \vec{f}(\vec{\alpha}^k) \tag{15}$$



Hererogeneous single-level tree
- Simultaneous Distribution
- Staggered Start
- Root Node does processing
Root
Communication
t
$\alpha_0 w_0 T_{cp}$
$(\alpha_1 T_{cm})^X z_1$
Child 1
Communication
Computation
$\alpha_1 w_1 T_{cp}$
T_f
$(\alpha_2 T_{cm})^X z_2$
Child 2
$\alpha_2 w_2 T_{cp}$
$(\alpha_3 T_{cm})^X z_3$
Child 3
$\alpha_N w_N T_{cp}$
$(\alpha_N T_{cm})^X z_N$
Child N

Fig. 2. Simultaneous distribution, staggered start

One can set the number of runs for this iteration and stop if it converges to a solution with acceptable error. Here we ran 5000 iterations for 20 processors and it converged to a solution with negligible error after several thousand runs.

2.2 Simultaneous distribution, staggered start

A simultaneous distribution, staggered start policy involves the root distributing loads to its children nodes simultaneously and the children nodes starting computation after they have received the entire fractions of the loads. In figure 2:

$$\alpha_0 w_0 T_{cp} = (\alpha_1 T_{cm})^{\chi} z_1 + \alpha_1 w_1 T_{cp} \tag{16}$$

$$(\alpha_1 T_{cm})^{\chi} z_1 + \alpha_1 w_1 T_{cp} = (\alpha_2 T_{cm})^{\chi} z_2 + \alpha_2 w_2 T_{cp} \tag{17}$$

$$(\alpha_2 T_{cm})^{\chi} z_2 + \alpha_2 w_2 T_{cp} = (\alpha_3 T_{cm})^{\chi} z_3 + \alpha_3 w_3 T_{cp} \tag{18}$$

$$\cdots\cdots$$
$$\cdots\cdots$$
$$\cdots\cdots$$

$$\begin{aligned} (\alpha_{N-1} T_{cm})^{\chi} z_{N-1} + \alpha_{N-1} w_{N-1} T_{cp} \\ = (\alpha_N T_{cm})^{\chi} z_N + \alpha_N w_N T_{cp} \end{aligned} \tag{19}$$

Manipulating the recursive equations and normalization equation one can obtain

$$f_0 = \alpha_0 w_0 T_{cp} - (\alpha_1 T_{cm})^{\chi} z_1 - \alpha_1 w_1 T_{cp} = 0 \tag{20}$$

$$f_1 = (\alpha_1 T_{cm})^{\chi} z_1 + \alpha_1 w_1 T_{cp} - (\alpha_2 T_{cm})^{\chi} z_2 - \alpha_2 w_2 T_{cp} = 0 \tag{21}$$

$$f_2 = (\alpha_2 T_{cm})^{\chi} z_2 + \alpha_2 w_2 T_{cp} - (\alpha_3 T_{cm})^{\chi} z_3 - \alpha_3 w_3 T_{cp} = 0 \tag{22}$$

$$\cdots\cdots$$

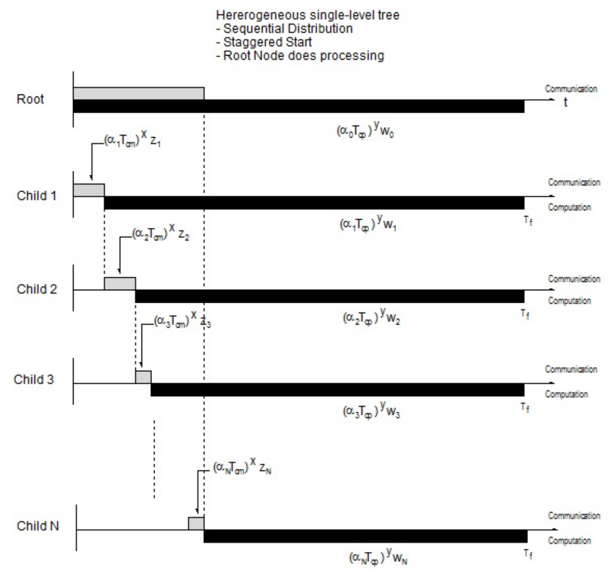

Hererogeneous single-level tree
- Sequential Distribution
- Staggered Start
- Root Node does processing
Root
Communication
t
$(\alpha_0 T_{cp})^y w_0$
$(\alpha_1 T_{cm})^\chi z_1$
Child 1
Communication
Computation
$(\alpha_1 T_{cp})^y w_1$
T_f
$(\alpha_2 T_{cm})^\chi z_2$
Child 2
Communication
Computation
$(\alpha_2 T_{cp})^y w_2$
T_f
$(\alpha_3 T_{cm})^\chi z_3$
Child 3
Communication
Computation
$(\alpha_3 T_{cp})^y w_3$
T_f
$(\alpha_N T_{cm})^\chi z_N$
Child N
Communication
Computation
$(\alpha_N T_{cp})^y w_N$
T_f

Fig. 3. Nonlinear communication and nonlinear computation

$$\cdots\cdots$$
$$\cdots\cdots$$

$$f_{N-1} = (\alpha_{N-1}T_{cm})^\chi z_{N-1} + \alpha_{N-1} w_{N-1} T_{cp} \\ -(\alpha_N T_{cm})^\chi z_N - \alpha_N w_N T_{cp} = 0 \tag{23}$$

$$f_N = \alpha_0 + \alpha_1 + \alpha_2 + \cdots + \alpha_N - 1 = 0 \tag{24}$$

Then one can use the Newton's method to solve it for the unknown α's.

2.3 Nonlinear communication, nonlinear computation, sequential distribution, staggered start

Here, both the computation and communication time is nonlinear to the size of the load. We have integer power χ ($\chi > 1$) for the communication nonlinearity and integer power y ($y > 1$) for the computation nonlinearity.

A sequential distribution, staggered start policy involves the root node distributing loads to its children nodes in a sequential way and the children nodes starting computation after they have received the entire fractions of the loads.

One can have timing equations (figure 3):

$$(\alpha_0 T_{cp})^y w_0 = (\alpha_1 T_{cm})^\chi z_1 + (\alpha_1 T_{cp})^y w_1 \tag{25}$$

$$(\alpha_1 T_{cp})^y w_1 = (\alpha_2 T_{cm})^\chi z_2 + (\alpha_2 T_{cp})^y w_2 \tag{26}$$

$$(\alpha_2 T_{cp})^y w_2 = (\alpha_3 T_{cm})^\chi z_3 + (\alpha_3 T_{cp})^y w_3 \tag{27}$$

$$\cdots\cdots$$
$$\cdots\cdots$$
$$\cdots\cdots$$

$$(\alpha_{N-1} T_{cp})^y w_{N-1} = (\alpha_N T_{cm})^\chi z_N + (\alpha_N T_{cp})^y w_N \tag{28}$$

$$\alpha_0 + \alpha_1 + \alpha_2 + \cdots + \alpha_N = 1 \tag{29}$$

Manipulating the recursive equations and normalization equation one can obtain

$$f_0 = (\alpha_0 T_{cp})^y w_0 - (\alpha_1 T_{cm})^\chi z_1 - (\alpha_1 T_{cp})^y w_1 = 0 \tag{30}$$

$$f_1 = (\alpha_1 T_{cp})^y w_1 - (\alpha_2 T_{cm})^\chi z_2 - (\alpha_2 T_{cp})^y w_2 = 0 \tag{31}$$

$$f_2 = (\alpha_2 T_{cp})^y w_2 - (\alpha_3 T_{cm})^\chi z_3 - (\alpha_3 T_{cp})^y w_3 = 0 \tag{32}$$

$$\cdots\cdots$$
$$\cdots\cdots$$
$$\cdots\cdots$$

$$f_{N-1} = (\alpha_{N-1} T_{cp})^y w_{N-1} - (\alpha_N T_{cm})^\chi z_N - (\alpha_N T_{cp})^y w_N = 0 \tag{33}$$

$$f_N = \alpha_0 + \alpha_1 + \alpha_2 + \cdots + \alpha_N - 1 = 0 \tag{34}$$

One can calculate the Jacobian of the $\vec{f}$ and use Newton's method to solve for the unknown α's.

Whether the complexity is sub-linear or super-linear can be calculated from the exponents y (computation) and χ (communication). One usually thinks of algorithm complexity in terms of data size (as in $O(N)$ where N is the data size). We have an alternative description of data size which needs to be normalized with respect to how long it takes to communicate data. To communicate data takes time $T_{com} = N^\chi$ and to process (compute) data takes time $T_{proc} = N^y$. Normally we say the algorithm has $O(N^y)$ complexity. In terms of T_{com} though $N = {T_{com}}^{1/\chi}$ and substituting for N, $T_{proc} = {T_{com}}^{y/\chi}$ so the actual complexity is $O(N^{y/\chi})$ instead of $O(N^y)$. If $y > x$ the overall complexity is super-linear and if $y < x$ the overall complexity is sub-linear.

2.4 Differentiable functions of communication time and computation time

The power law examined above are of much practical interest but in fact one can account for communication time that is a differentiable, increasing and nonlinear function of $\alpha_i T_{cm}$. Here, a set of differentiable functions for sequential distribution, simultaneous start is introduced as an example:

$$\alpha_0 w_0 T_{cp} = \alpha_1 w_1 T_{cp} \tag{35}$$

$$\alpha_1 w_1 T_{cp} = g(\alpha_1 T_{cm}) z_1 + \alpha_2 w_2 T_{cp} \tag{36}$$

$$\alpha_2 w_2 T_{cp} = g(\alpha_2 T_{cm}) z_2 + \alpha_3 w_3 T_{cp} \tag{37}$$

$$\cdots\cdots$$

$$\cdots\cdots$$
$$\cdots\cdots$$

$$\alpha_{N-1}w_{N-1}T_{cp} = g(\alpha_{N-1}T_{cm})z_{N-1} + \alpha_N w_N T_{cp} \quad (38)$$

where $g(\alpha_i T_{cm})$ is a differentiable increasing nonlinear function of $\alpha_i T_{cm}$, such as:

$$g(\alpha_i T_{cm}) = (\alpha_i T_{cm})^{\chi} \quad (39)$$

$$g(\alpha_i T_{cm}) = e^{\alpha_i T_{cm}} \quad (40)$$

$$g(\alpha_i T_{cm}) = \frac{\alpha_i T_{cm}}{1+\alpha_i T_{cm}} \quad (41)$$

3 An Example

Here, we take z, T_{cp} and T_{cm} all as 1 and leave w as a variable. Note that z (inverse communication speed) appears linearly in the equations. Below are two performance measurements we use:

Makespan: the time period between when the root processor starts to send loads and the last processor finishes computing.

Speedup: the ratio of processing time on one processor to processing time on the entire network.

3.1 Second order communication, third order computation, sequential distribution, staggered start

In this case, the communication time is the square of the size of the load and the computation time is the cube of the size of the load. Thus the complexity is super-linear. One has the timing equations:

$$\vec{f} = \begin{bmatrix} (\alpha_0 T_{cp})^3 w_0 - (\alpha_1 T_{cm})^2 z_1 - (\alpha_1 T_{cp})^3 w_1 \\ (\alpha_1 T_{cp})^3 w_1 - (\alpha_2 T_{cm})^2 z_2 - (\alpha_2 T_{cp})^3 w_2 \\ (\alpha_2 T_{cp})^3 w_2 - (\alpha_3 T_{cm})^2 z_3 - (\alpha_3 T_{cp})^3 w_3 \\ \vdots \\ (\alpha_{N-1} T_{cp})^3 w_{N-1} - (\alpha_N T_{cm})^2 z_N \\ -(\alpha_N T_{cp})^3 w_N \\ \alpha_0 + \alpha_1 + \alpha_2 + \cdots + \alpha_N - 1 \end{bmatrix} = 0 \quad (42)$$

Let

$$\epsilon_i = 3\alpha_i^2 T_{cp}^3 w_i \quad (43)$$

and

$$\theta_i = 2\alpha_i T_{cm}^2 z_i \quad (44)$$

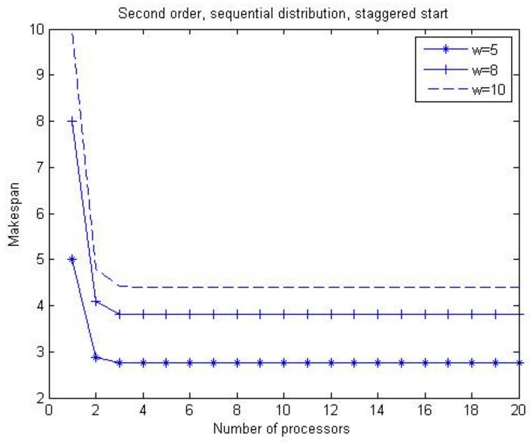


Fig. 4. Makespan - Nonlinear communication (second order) and nonlinear computation (third order), sequential distribution staggered start

The Jacobian of $\vec{f}$ is:

$$\mathcal{J}_{\vec{f}}(\vec{\alpha}) = \begin{bmatrix} \frac{\partial f_0}{\partial \alpha_0}(\vec{\alpha}) & \cdots & \frac{\partial f_0}{\partial \alpha_N}(\vec{\alpha}) \\ \frac{\partial f_1}{\partial \alpha_0}(\vec{\alpha}) & \cdots & \frac{\partial f_1}{\partial \alpha_N}(\vec{\alpha}) \\ \frac{\partial f_2}{\partial \alpha_0}(\vec{\alpha}) & \cdots & \frac{\partial f_2}{\partial \alpha_N}(\vec{\alpha}) \\ \vdots & \vdots & \vdots \\ \frac{\partial f_N}{\partial \alpha_0}(\vec{\alpha}) & \cdots & \frac{\partial f_N}{\partial \alpha_N}(\vec{\alpha}) \end{bmatrix}$$

$$= \begin{bmatrix} \epsilon_0 & -\epsilon_1-\theta_1 & 0 & \cdots & 0 & 0 \\ 0 & \epsilon_1 & -\epsilon_2-\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & & \\ 0 & 0 & 0 & \cdots & \theta_{N-1} & -\epsilon_N-\theta_N \\ 1 & 1 & 1 & \cdots & 1 & 1 \end{bmatrix} \quad (45)$$

One can set the initial $\vec{\alpha}$ as all zeros and insert it into the right hand side of the iterative function below to get a newer set of the $\vec{\alpha}$ on the left hand side. Substitute the newer $\vec{\alpha}$ into the right hand side again and so on.

$$\vec{\alpha}^{k+1} = \vec{\alpha}^k - \mathcal{J}^{-1}(\vec{\alpha}^k)\vec{f}(\vec{\alpha}^k) \quad (46)$$

After the $\vec{\alpha}$ is obtained, one can calculate the speedup and the makespan (finish time).

$$Makespan = \alpha_0^3 w_0 T_{cp}^3 \quad (47)$$

$$Speedup = \frac{w_0 T_{cp}^3}{\alpha_0^3 w_0 T_{cp}^3} = \frac{1}{\alpha_0^3} \quad (48)$$

Figure 4 and 5 show how the makespan and speedup change as the number of processors increases with sequential distribution, staggered start, second order communication and third order computation.

The nature of the curves in this section are a function of the power of the nonlinearity and also the scheduling protocols that is used.

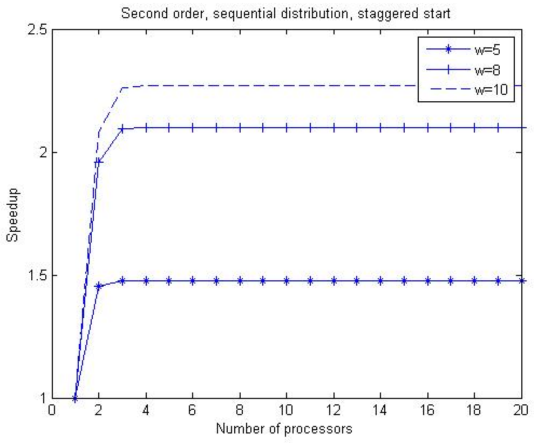


Fig. 5. Speedup - Nonlinear communication (second order) and nonlinear computation (third order), sequential distribution staggered start

4 Conclusion

Scheduling divisible loads with nonlinear communication time has many aerospace applications including fast Fourier transform, matrix operations, line detection using the Hough transform and pattern recognition using 2D hidden Markov models. This paper proposes an iterative method to find the optimal scheduling for single level tree networks under different distribution policies. A quadratic and cubic nonlinearity example is used to demonstrate the proposed algorithm. The testing results show that this scheduling algorithm can provide an optimal solution for parallel and distributed systems with divisible and nonlinear communication time loads. Such optimal solutions can maximize the responsiveness of critical data processing where time is of the essence.

5 Acknowledgements

The authors would like to acknowledge useful discussions with Joseph Toledano at the start of this research.
The authors thank the reviewers and the EIC for substantive suggestions that added to the paper.

References

[1] R.O. Duda and P.E. Hart, "Use of the Hough Transform to Detect Lines and Curves in Pictures, Communications of the ACM, vol. 15, 1972, pp. 11-15.
[2] http://www.ece.sunysb.edu/~tom/.
[3] Cheng, Y.C. and Robertazzi, T.G., "Distributed Computation with Communication Delays", IEEE Transactions on Aerospace and Electronic Systems, Vol. 24, No. 6, Nov.1988, pp. 700-712.
[4] Agrawal, R. and Jagadish, H.V., "Partitioning Techniques for Large Grained Paralleism", IEEE Transactions on Computers, Vol. 37, No. 12, Dec. 1988, pp. 1627-1634.
[5] Robertazzi, T.G, "Ten Reasons to Use Divisible Load Theory", IEEE Computer, vol, 36, no. 5, pp. 63-68, 2003.
[6] S. Bataineh and T.G. Robertazzi, "Bus Oriented Load Sharing for a Network of Sensor Driven Processors". IEEE Transactions on Systems, Man and Cybernetics, 21 1202-1205, 1991.
[7] J. Blazewicz and M. Drozdowski, "Scheduling Divisible Jobs on Hypercubes". Parallel computing, 21 1945-1956, 1996.
[8] J. Blazewicz and M. Drozdowski, "The Performance Limits of a Two Dimensional Network of Load Sharing Processors". Foundations of Computing and Decision Sciences, 21 3-15,1996.
[9] T.G. Robertazzi. "Processor Equivalence for a Linear Daisy Chain of Load Sharing Processors". IEEE Transactions on Aerospace and Electronic Systems, 29:1216-1221, 1993.
[10] V. Bharadwaj, D. Ghose, V. Mani, "Multi-installment Load Distribution in Tree Networks with Delays". IEEE Transactions on Aerospace and Electronic Systems, 31 555-567 (1995).
[11] Y. Yang, H. Casanova, "UMR: A Multi-Round Algorithm for Scheduling Divisible Workloads". Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'03), Nice, France, 2003.
[12] O. Beaumont, A. Legrand, and Y. Robert, "Optimal Algorithms for Scheduling Divisible Workloads on Heterogeneous Systems". 12th Heterogeneous Computing Workshops HCW'2003, 2003.
[13] J. Blazewicz and M. Drozdowski, "Distributed Processing of Distributed Jobs with Communication Startup Costs. Discrete Applied Mathematics", 76 21-41, 1997.
[14] A.L. Rosenberg, "Sharing Partitionable Workloads in Heterogeneous NOWs: greedier is not better". In D.S. Katz, T. Sterling, M. Baker, L. Bergman, M. Paprzycki, and R. Buyya, editors. Cluster Computing 2001 pp. 124-131, 2001.
[15] D. Ghose and H. J. Kim, "Load Partitioning and Trade-Off Study for Large Matrix Vector Computations in Multicast Bus Networks with Communication Delays", Journal of Parallel and Distributed Computing, vol. 54, 1998.
[16] P.F. Dutot, "Divisible Load on Heterogeneous Linear Array". Proceeding of the International Parallel and Distributed Processing Symposium (IPDPS'03), Nice, France, 2003.
[17] M. Drozdowski and P. Wolniewicz, "Out-of-core Divisible Load Processing", IEEE Transactions on Parallel and Distributed Systems, vol. 14, 2003, pp. 1048-1056.
[18] J.T. Hung and T.G. Robertazzi, "Scheduling Nonlinear Computational Loads", IEEE Transactions on Aerospace and Electronic Systems, vol. 44, no. 3, July 2008, pp. 1169-1182.
[19] S. Suresh, C. Run, H.J. Kim, T.G. Robertazzi and Y.-I. Kim, "Scheduling Second Order Computational Load in Master-Slave Paradigm", IEEE Transactions on Aerospace and Electronic Systems, vol. 48, no. 1, Jan 2012, pp. 780-793.
[20] S. Suresh, H.J. Kim, C. Run, and T.G. Robertazzi, "Scheduling Nonlinear Divisible Loads in a Single Level Tree Network", The Journal of Supercomputing, vol. 61, no. 3, Sept. 2012, pp. 1069-1088.
[21] Wong H.M. and B. Veeravalli, "Aligning Biological Sequences on Distributed Bus Networks: A Divisible Load Scheduling Approach," IEEE Transactiosn on Information Technology in BioMedicine, vol. 9, no. 4, Dec. 2005, pp. 489-501.
[22] Ping, D.L.H., Veeravalli, B. and Bader, D., "On the Design of High-Performance Algorithms for Aligning Multiple Protein Sequences in Mesh-Based Multiprocessor Architectures," Journal of Parallel and Distributed Computing, vol. 67, no. 9, 2007, pp. 1007-1017.
[23] Barlas, G.D., "An Analytical Approach to Optimizing Parallel Image Registration," IEEE Transactions on Parallel and Distributed Systems, vol. 21, no. 8, Aug. 2010, pp. 1074-1088.
[24] Berlinska, J. and Drozdowski, M., "Scheduling Divisible MapReduce Computations," Journal of Parallel and Distributed Computing, vol. 71, no. 3, March 2011, pp. 450-459.
[25] Gamboa, C.F. and Robertazzi, T.G., "Simple Performance Bounds for Multicore and Parallel Channel Systems," Parallel Processing Letters, vol. 21, no. 4, Dec 2011, pp. 439-459.
[26] Barlas, G., "Cluster-Based Optimized Parallel Video Transcoding," Parallel Computing, vol. 38, March 2012, pp. 226-244.
[27] D. Ghose,"A Feedback Strategy for Load Allocation in Workstation Clusters with Unknown Network Resource Capabilities Using the DLT Paradigm", Proc. 2002 Int'l Conf. Parallel and Distributed Processing Techniques and Applications (PDPTA 02), vol. 1, CSREA Press, 2002, pp. 425-428
[28] Beaumont O., Larcheveque H. and Marchal L. "Non Linear Divisible Loads: There Is No Free Lunch", 2013 IEEE 27th International Symposium on Parallel and Distributed Processing, pp. 863-873.