A Product Form Solution for Tree Networks with Divisible Loads

Thomas G. Robertazzi

Department of Electrical and Computer Engineering
Stony Brook University
Stony Brook, NY 11794, USA
tom@ece.sunysb.edu

Abstract

A product form solution for the optimal fractions of divisible load to distribute to processors in a multi-level tree network is described. Here optimality involves parallel processing the load in a minimal amount of time. This tractable solution is similar to the product form solution for equilibrium state probabilities arising in Markovian queueing networks. The existence of this product form solution answers a long standing open question for divisible load scheduling.

Keywords: divisible load scheduling; product from solution; queueing networks

1 Introduction

It is well known in queueing theory that there are product form solutions (solutions that are a product of terms) for the equilibrium state probabilities of both open and closed queueing networks with Markovian statistics [1,2,3]. Here open networks allow customers to enter and leave a network of queues and closed networks are sealed systems. Under Markovian statistics the input arrival processes for open networks are independent Poisson processes and the service times of customers at queues are independent and follow negative exponential distributions. Product form solutions in queueing networks are considered a simplifying feature. Moreover efficient numerical algorithms for solving such product form networks have been developed such as Mean Value Analysis [4] and the Convolution algorithm [5].

If $p_i(n_i)$ is the marginal probability that there are n_i customers in the ith queue and $p(\underline{n})$ is the joint equilibrium probability that there are n_1 customers in the first queue, n_2 customers in the 2nd queue and n_M customers in the Mth queue then under the product form solution for open networks:

$$p(\underline{n}) = p_1(n_1)p_2(n_2)p_3(n_3)...p_M(n_M) \tag{1}$$

For closed queueing networks the product form solution is of the same format with the inclusion of a normalization constant factor.

A "divisible" load is a computational/communication load that is for all purposes perfectly partitionable among processors and links in a network. One can think of a divisible load as a massive data file for which we want to assign fractions of it to a number of processors tied together by an interconnection network to achieve the benefits of parallel processing. Moreover we seek a time optimal solution: the load must be distributed from a source (or sources) to processors over the network in a scheduled fashion so that all load is processed in a minimal amount of time. The study of the scheduling of divisible loads began with the work of Aggrawal and Jagadish [6] and Cheng and Robertazzi [7] published in 1988. This has been an active research area with over a hundred journal papers [8], tutorials [9,10], a monograph [11] and a number of recent book chapters [12,13,14] on the topic appearing over the years.

Linear divisible load theory is a theory of proportions. As an example, consider two processors connected by a link of infinite data rate. Suppose that one processor is twice as fast as the other. Then irregardless of which processor the load originates on, two thirds of the load should be assigned to the faster processor and one third of the load should be assigned to the slower processor. Generalizing this example of proportions, divisible load theory allows the amount of load as well as when the load should be placed on processors and links to be calculated. For linear divisible load theory this is done by mathematical programming, or more relevant to this paper, by linear equation solution or in many cases by algebraic recursions.

Such solutions are based on an "optimality principle" [11]. For a given interconnection network and load distribution policy, the minimal time optimal solution results when all of the processors stop processing at the same instant. Analytical proofs of this principle are available [11,15]. However the simple intuition behind the proofs is that if the processors didn't stop processing at the same instant, load could be transferred from busy to idle processors to improve the solution so that the solution where the processors do not stop at the same time is not optimal.

For many years solutions for optimal load allocation in single layer tree networks that are essentially a one dimensional product form solution have been known [15]. However it has been an open problem, explicitly recognized in [21], as to what type of network an M dimensional product form solution corresponds to. This paper demonstrates that there is an M dimensional product form solution for load distribution from a single source (root) node in a multi-level tree type interconnection network. This is significant as tree type networks are often used in an embedded fashion in other interconnection networks for load distribution [16,17,18,19,23].

In fact it has been known since 1990 [20] that optimal load distribution to processors from a root node in a multiple level tree network could be solved recursively by collapsing single layer subtrees, working from the bottom of the tree to the root, into equivalent processors and tying the solutions together. But this has not been done in explicit analytical form to date as is done in this letter.

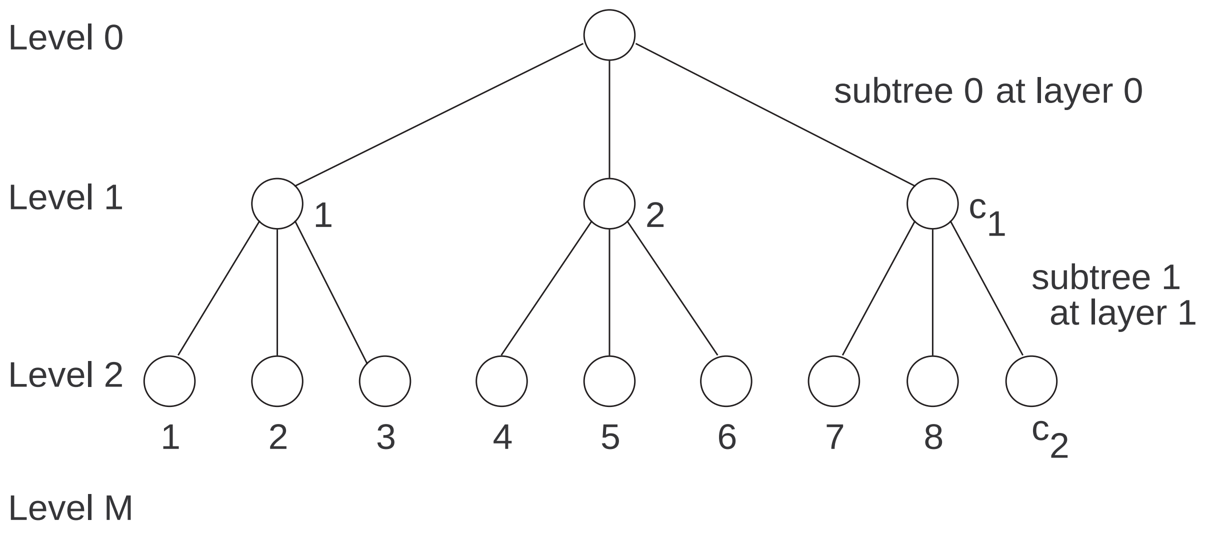

Figure 1: A multi-level tree.

Section 2 describes the model and notation used in this letter. The product form solution itself is presented in section 3. The conclusion appears in section 4.

2 Model and Notation

Consider the multilevel tree of Figure 1. The root is at “level” 0, the root’s children are at level 1, the root’s grandchildren are at level 2 and the lowest level of the finite tree is level M. Also defined are single layer ”subtrees” consisting of a node and its children. The root and its children form subtree 0 at “layer” 0. A child of the root and its children are at the subtree 1 layer. Going down the tree the last subtrees are at the subtree M-1 layer. The difference between “level” and “layer” is illustrated in the figure. The nodes at the ith level, across all subtrees, are numbered from left to right as 1,2,3 ... c_i.

It is assumed that a divisible load of volume V originates at the root at time 0. The load is distributed throughout the tree to the other nodes. each one of which can process load. There are several load distribution policies that can be used [14,22] which are discussed below.

In terms of the processing and transmission of divisible load the following parameters are defined:

w_r: the inverse of the constant computing speed of the rth node.

z_t: the inverse of the constant transmission speed of the tth link.

T_{cp}: the computation intensity constant. The entire load can be processed on the rth processor in time $w_r T_{cp}$.

T_{cm}: the communication intensity constant. The entire load can be transmitted over the tth link in time $z_t T_{cm}$.

For the allocation of load the following variables are defined:

$\beta_{i,j}$: the fraction of load received at the root of an ith layer subtree that is sent to its jth child.

$\alpha_{i,j}$: the fraction of the total load that is processed at the ith level by the jth processor.

For the α if the normalized total load is unity ($V = 1$):

$$\sum_{i=0}^{M} \sum_{j=1}^{c_i} \alpha_{i,j} = 1 \tag{2}$$

If the total amount of load is V:

$$\sum_{i=0}^{M} \sum_{j=1}^{c_i} \alpha_{i,j} V = V \tag{3}$$

3 The M Level Subtree Product Form Solution

A little thought will show that the fraction of load received at a leaf (i.e. bottom most) node in an M level tree can be written as a product of the fractions of load passed to each node above it in a path back to the root for $i = M$ and $j = 1, 2, 3 ... c_i$:

$$\alpha_{i,j} = \beta_{0,d} \beta_{1,e} \beta_{2,f} ... \beta_{i-2,h} \beta_{i-1,j} V \tag{4}$$

In this equation d is the dth child in subtree 0 along the path from the root to the node of interest. Then e is the eth child in subtree 1 along the same path and so on.

This equation expresses the optimal load assigned to each (bottom level) leaf processor as a product of terms of the fractions of load received at each node along a path from the root to the leaf node of interest. These β fractions are relative. As an example, consider a three level tree. If $\beta_{0,1} = 0.5$ and $\beta_{1,1} = 0.2$, then the absolute amount of load received by the leftmost grandchild of the root is 0.10. That is, twenty percent of the load received by the leftmost child of the root (which is fifty percent of the total load) or ten percent of the total load is sent to the leftmost grandchild of the root.

Note also that all processors participate in load processing. This equation allows the optimal fraction of load at any leaf node to be calculated.

One may be interested in the optimal load allocation to a node in the interior of the multi-level tree. In this case the optimal load allocated to processor i,j (i.e. $\alpha_{i,j}$) can be found from a partial product form equation similar to equation (4) for the total load delivered to node i,j minus the load node i,j gives to its children (also computed from partial product form equations similar to equation (4)). Here a "partial" product form equation is similar to (4) but treats an interior node as a leaf node of a partial tree, computing the load delivered to the interior node (which includes load for itself and for its descendants). However even in the case of interior node load allocation the β's must be calculated from the bottom most layer working upwards. That is, with the node of interest being a root node in a single layer subtree, the children nodes in the subtree have inverse processing speeds equivalent to the (possibly multi-level) subtrees below them that they replace. Thus one starts with the single layer subtree at the bottom of the overall multi-level tree network, collapsing them into equivalent processors and continuing this process working upwards in the multi-level tree until the β's for nodes along a path from the overall multi-level tree root to the node of interest can be solved for [11,14].

How does one calculate the β? There is standard theory [14,22] available in the divisible load literature for doing this. Specifically with some reuse of the i, j notation from the previous equation, for the root of a single layer subtree:

$$\beta_{i,0} = \frac{\frac{1}{k_1}}{\left[\frac{1}{k_1} + 1 + \sum_{j=2}^{m}(\prod_{l=2}^{j} q_l)\right]} \tag{5}$$

For the leftmost child of a single layer subtree ($j = 1$):

$$\beta_{i,1} = \frac{1}{\left[\frac{1}{k_1} + 1 + \sum_{j=2}^{m}(\prod_{l=2}^{j} q_l)\right]} \tag{6}$$

For the other children of a single level subtree ($j = 2, 3 ... m$):

$$\beta_{i,j} = \frac{(\prod_{l=2}^{j} q_l)}{\left[\frac{1}{k_1} + 1 + \sum_{j=2}^{m}(\prod_{l=2}^{j} q_l)\right]} \tag{7}$$

Here we have written expressions for the fraction of the load that each node receives of the load given to the node's subtree's root. In this equation the j numbers the children in a specific subtree as $1, 2, 3...m$.

The k_1 and q_l are functions specific to a given load distribution policy in a single level tree network where load originates at the root node. They are usually a function of w, z, T_{cp}, and T_{cm}. They can also be functions of intermediate equivalent processing speeds of equivalent processors replacing a subnetwork consisting of a node and all the children/grandchildren... below it.

Several k and q functions appear in [14,22]. For instance consider a sequential load distribution where load is distributed in turn to each child once. Assume each child commences processing as soon as load begins to be received and that the root also does load processing starting at time 0. Then:

$$k_1 = \frac{w_0}{w_1} \tag{8}$$

$$q_l = \frac{w_{l-1}T_{cp} - z_{l-1}T_{cm}}{w_l T_{cp}} \quad l = 2, 3...m \tag{9}$$

In a multi-level tree for instance, the w_{l-1} and w_l are inverse equivalent processing speeds of the subtree networks that these single layer subtree children nodes replace.

As a second example, consider a policy with the simultaneous distribution of load to all children from the root. Let processing at a child start when all load for that child has been received and let the root process some of the load starting at time 0. Then:

$$k_1 = \frac{w_0 T_{cp}}{z_1 T_{cm} + w_1 T_{cp}} \tag{10}$$

$$q_l = \frac{w_{l-1}T_{cp} + z_{l-1}T_{cm}}{w_l T_{cp} + z_l T_{cm}} \tag{11}$$

Many policies can be substituted into the main equations. These policies differ in whether load distribution is sequential or simultaneous, whether the root does processing and whether processing at each child starts as load begins to be received or when load is completely received at a child.

If one substitutes equation (7) into equation (4) for $j = 2, ...m$ one has a product of terms where each term has a $\prod_{l=2}^{j} q_l$ component in its numerator. This product of products is akin to the product form solution of queueing networks. Note that the node 0 and node 1 numerator terms for β in a subtree have somewhat different values. This is because of the manner in which the β are determined in a single level tree here (i.e. using the leftmost child, node 1, as a reference processor with unnormalized allocation value of 1).

A normalization constant like term can be found from the product of the denominators of the individual β.

Thus the product form solution for the optimal load to assign to processors in a multilevel tree with divisible load distribution from the root has a finite "state" space and thus it also has a normalization constant (also called a partition function). Here the processors play a role analogous to the states of a closed queueing network Markov chain.

To summarize, to evaluate the α's for every node in a tree (a) working from the bottom to the top of the tree find the β's (b) calculate the α's for the leaf nodes using equation (4) and finally (c) working from the bottom to the top of the tree find the remaining interior node α's as described above.

4 Conclusion

This letter presents novel and compact analytical results for optimal load distribution in multilevel tree networks. As mentioned such trees are very general, being used as distribution networks in other interconnection topologies.

While it might be surprising that both queueing networks (a stochastic model) and divisible load scheduling (a deterministic model) admit product form solutions, the underlying reason is that, in their basic form, both are linear models. Certain patterned labelings of the elements of a linear system lead to recursive and product form solutions.

The results in this letter will be of interest theoretically, for devising simple code for optimal load distribution and also as one starting point for research on very large grid and cloud networks.

5 Acknowledgments

The author acknowledges the support of DOE grant DE-SC0003361. While several Ph.D students worked with the author on multi-level trees, he would like to particularly acknowledge the work of Y.C. Cheng in 1990 and more recently J.-T. Hung for making this letter possible. The comments of the reviewers were particularly valuable and very much appreciated.

References

[1] J.R. Jackson. Networks of waiting lines. *Operations Research*, 5:518-521, 1957.

[2] W.J. Gordon and G.F. Newell, Closed queueing systems with exponential servers. *Operations Research*, 15:254-265, 1967.

[3] F. Basket, K.M. Chandy, R.R. Muntz and F. Palacios. Open, closed and mixed networks of queues with different classes of customers. *Journal of the ACM*, 22:248-260, 1975.

[4] M. Reiser and S.S. Lavenberg. Mean-value analysis of closed multichain queueing networks. *Journal of the ACM*, 27:313-322, 1980.

[5] J.P. Buzen. Computational algorithms for closed queueing networks with exponential servers. *Communications of the ACM*, 16:527-531, 1973.

[6] R. Agrawal and H.V. Jagadish. Partitioning techniques for large-grained parallelism. *IEEE Trans. Computers*, 37:1627-1634, 1988

[7] Y.-C. Cheng and T.G. Robertazzi. Distributed computation with communication delays. *IEEE Trans. Aerospace and Electronic Systems*, 24:700-712, 1988.

[8] Divisible Load Theory publication list at the author's web page at www.ece.sunysb.edu

[9] B. Veeravalli, D. Ghose and T.G. Robertazzi. Divisible load theory: a new paradigm for load scheduling in distributed systems. *Cluster Computing*, 6:7-18, 2003.

[10] T.G. Robertazzi. Ten reasons to use divisible load theory. *Computer*, 36:63-68, 2003.

[11] B. Veeravalli, D. Ghose, V. Mani and T.G. Robertazzi. *Scheduling Divisible Loads in Parallel and Distributed Systems*. IEEE Computer Society Press (now distributed by Wiley), Los Alamitos CA USA, 1996.

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

[13] H. Casanova, A. Legrand and Y. Robert. *Parallel Algorithms*. CRC Press, FL USA, 2009.

[14] T.G. Robertazzi. *Networks and Grids: Technology and Theory*. Springer, New York, 2007.

[15] J. Sohn and T.G. Robertazzi, Optimal load sharing for a divisible job on a bus network. *IEEE Transactions on Aerospace and Electronic Systems*, 32:34-40, 1996.

[16] 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.

[17] J. Blazewicz, M. Drozdowski, F. Guinand and D. Trystram. Scheduling a divisible task in a two-dimensional toroidal mesh. *Discerte Applied Mathematics*, 94:35-50, 1999.

[18] M. Drozdowski and W. Glazek. Scheduling divisible loads in a three-dimensional mesh of processors. *Parallel Computing*, 25:381-404, 1999.

[19] W. Glazek. A multistage load distribution strategy for three-dimensional meshes. *Cluster Computing*, 6:31-39, 2003.

[20] Y.C. Cheng and T.G. Robertazzi, Distributed computation for a tree network with communication delays. *IEEE Transactions on Aerospace and Electronic Systems*, 26:511-516, 1990.

[21] M. Moges and T.G. Robertazzi, Divisible load scheduling and Markov chain models. *Computers and Mathematics with Applications*, 52:1529-1542, 2006.

[22] J.-T. Hung, *Scalable Scheduling in Parallel, Distributed and Grid Systems*, Ph.D Thesis, Stony Brook University, 2003.

[23] D. England, B. Veeravalli and J.B. Weissman, A robust spanning tree topology for data collection and dissemination in distributed environments. *IEEE Transactions on Parallel and Distributed Systems*, 18:608-620, 2007.