ACADEMIC PRESS

Available at www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

J. Parallel Distrib. Comput. 64 (2004) 29–35

Journal of Parallel and Distributed Computing

http://www.elsevier.com/locate/jpdc

Load sequencing for a parallel processing utility

Saravut Charcranoon, Thomas G. Robertazzi,* and Serge Luryi

Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY 11794, USA

Received 25 May 2000; revised 3 March 2003; accepted 23 May 2003

Abstract

The monetary cost optimization of a computer utility load distribution problem is examined. The problem is to find the sequence in which to distribute divisible computing load from a root processor to its children processors which achieves the lowest monetary distribution cost. The convergence performance of a heuristic greedy algorithm is studied. This problem is directly relevant to computer utilities which offer computing and software hosting to organizations for a monetary charge.

Keywords: Computer utility; Divisible load; Economics; Heuristic algorithm; Single-level tree network; Software leasing; Star network

1. Introduction

There is an increasing amount of ongoing work on the leasing, rather than purchase, of software. Application Service Providers (ASPs), which do such leasing, have received much attention in the past 3 years [2,10–14] as well as an increasing amount of market share. More recently, there have been corporate announcements on the creation of computer utilities: computer services providers using third-party machines. We envision a situation where future computer utilities provide researchers and developers access to high-performance parallel machines and/or proprietary algorithms at some charge. The machines involved may be based on new principles currently under research such as cryogenic peta flops technology [7] or quantum computing. These machines may be expensive to build and operate, thus leading to a necessity to lease their processing time in a cost and performance efficient manner. Accounting for leased processing time has been an open research problem for some time. As W. Wayt Gibbs [9] put it in a 1997 issue of *Scientific American*:

> So far not even the most ambitious metacomputing prototypes have tackled accounting: determining a fair price for idle processor cycles. It all depends on the risk, on the speed of the machine, on the cost of communication, on the importance of the problem—on a million variables, none of them well understood.

The question to be investigated in this paper is how load should be sequenced for distribution to processors, taking into account processor and link speeds and costs so that a parallel processor can solve a submitted problem at minimal monetary cost to both the service, and by implication, to the user.

A number of deliberate choices were made regarding the features of this problem:

- Single level tree (star) topology.
- Divisible load.
- Single installment sequential load distribution.
- Linear costs for communication and computation.

Generally, these choices were made for both realism and analytical tractability. It is natural for a first study to consider a star topology in a system whose load is distributed to a number of satellite processors. The divisible load model has over the years seen its tractability proven [3,8] and well models problems involving data parallelism. Such problems arise in scientific, engineering and business computing. In spite of these innocuous choices the related mathematics is algebraically substantive. A secondary reason for the choices is that they allow a comparison with earlier published work by some of the authors [15] considering computation costs only, in bus networks.

With these choices we seek in this paper to optimize the order in which a root processor should distribute

*Corresponding author. Fax: +1-631-632-8494.
E-mail address: tom@ece.sunysb.edu (T.G. Robertazzi).

0743-7315/$ - see front matter © 2003 Elsevier Inc. All rights reserved.
doi:10.1016/S0743-7315(03)00113-8

load to its children in order to complete the processing of a load in a minimal amount of time. This is a combinatorial optimization problem we call the "sequencing" problem. The thrust of this work is to develop a greedy swapping algorithm to solve this problem. Simple conditions are found for deciding when to make a particular swap. It is not guaranteed that the greedy algorithm will always converge to an optimal solution. However, the greedy algorithm converged to either optimal or best-known solutions with very high probability for networks containing small or moderate numbers of children processors. The greedy algorithm is also substantially faster than implementations of tabu search or simulated annealing for this problem.

One more choice regarding the problem discussed in this paper requires some explanation. In this work there are two objective functions to be optimized: the finish (solution) time, and the total monetary processing and transmission monetary cost. It is well known that there are several approaches to solve such multiple objective function optimization problems. The approach taken here is to find the minimal cost processor load distribution sequence such that for any load distribution sequence, finish time is minimized using the methodology of [3]. That is, for each possible load distribution sequence, load is allocated so that all processors stop computing at the same time instant and finish time is minimized for that specific load distribution sequence. While other approaches are certainly possible, we believe that the proposed approach is a natural one for a first study.

This paper is organized as follows. The model and monetary cost concept are provided in Sections 2 and 3. The resulting cost efficient sequencing is then presented in Section 4. A performance evaluation based on experiments using that algorithm is discussed in Section 5. Finally, the conclusion is given in Section 6.

2. Model, notation and load distribution

2.1. Model description

In this paper, a single-level tree (star) network where the root processor is equipped with a front-end processor is considered. A single-level tree network consists of $(N+1)$ processors and (N) links. All the processors are connected to the root processor, p_0, via communication links. Associated with the links and processors are the associated linear cost coefficients, $c_1^l, c_2^l, \ldots, c_N^l$ and $c_0^p, c_1^p, c_2^p, \ldots, c_N^p$, respectively. The root processor, assumed to be the only processor at which the load arrives, partitions a total processing load into $(N+1)$ fractions, keeps its own fraction α_0, and distributes the other fractions $\alpha_1, \alpha_2, \ldots, \alpha_N$ to the children processors $p_1, p_2, \ldots, p_N$ respectively and sequentially. Each processor begins computing immediately after receiving its assigned fraction of load and continues without any interruption until all of its assigned load fraction has been processed. Let:

- α_i the load fraction assigned to the ith link-processor pair
- w_i the inverse of the computing speed of the ith processor
- z_i the inverse of the link speed of the ith link
- T_{cp} computing intensity constant: the entire load is processed in $w_i T_{cp}$ seconds by the ith processor
- T_{cm} communication intensity constant: the entire load can be transmitted in $z_i T_{cm}$ seconds over the ith link
- T_f the finish time. Time at which the last processor ceases computation.

Then $\alpha_i w_i T_{cp}$ is the time to process the fraction α_i of the entire load on the ith processor. Note that the units of $\alpha_i w_i T_{cp}$ are [load] $\times$ [sec/load] $\times$ [dimensionless quantity] = [seconds]. Likewise, $\alpha_i z_i T_{cm}$ is the time to transmit the fraction α_i of the entire load over the ith link. Note that the units of $\alpha_i z_i T_{cm}$ are [load] $\times$ [sec/load] $\times$ [dimensionless quantity] = [seconds].

Our goal is to develop means to determine the optimal sequence (order) in which load should be distributed by the root to its children.

2.2. Cost efficient load distribution

It is intuitive that to minimize the processing finish time only, the time efficient load distribution should be such that all processors finish computing at the same time. Otherwise, the processing finish time could be reduced by transferring some fractions of load from busy processors to idle processors. Formal proofs of this argument in the case of linear, bus, and tree networks appear in [3]. However, under certain sets of network parameters, in order to minimize the processing finish time, it is not necessary that all processors have to be utilized. In [3] conditions are found which determine which processors should be used to process the arriving load in the case of a single-level tree network. Still, the processors with non-zero assigned load have to finish computing at the same time. In this paper it is assumed that system parameters are such that all processors in the network are utilized in order to achieve a cost efficient processing of the arriving load.

2.3. Fundamental recursive equations

The process of load distribution can be represented by Gantt-chart-like timing diagrams. The communication time is shown above the time axis and the computation time is shown below the time axis, as illustrated in Fig. 1. Referring to the timing diagram and assuming that all processors cease computation at the same time,

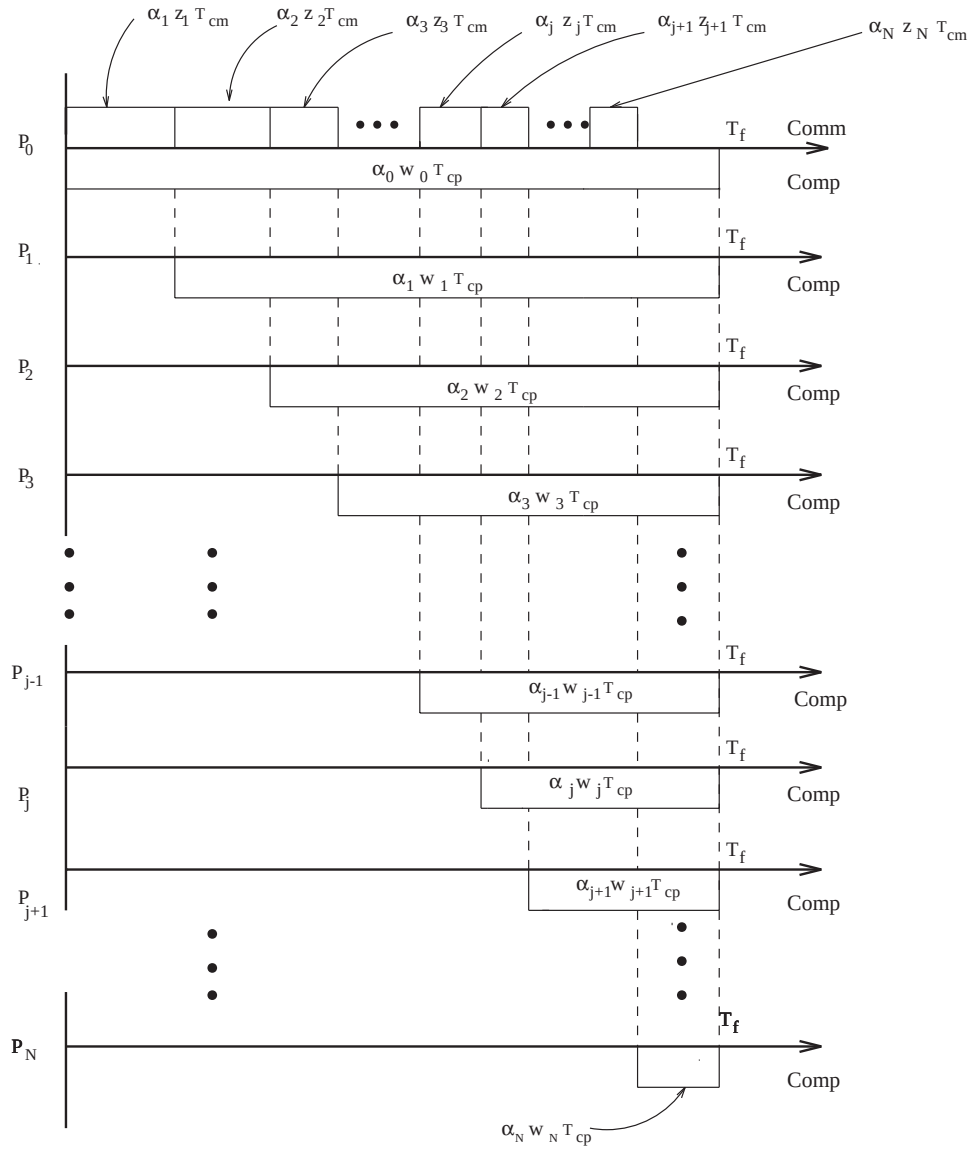

Fig. 1. Timing diagram: normal case.

one can derive fundamental recursive equations as

$$\alpha_i w_i T_{cp} = \alpha_{i+1} z_{i+1} T_{cm} + \alpha_{i+1} w_{i+1} T_{cp}, \quad i = 0, \ldots, N-1, \tag{1}$$

$$\alpha_{i+1} = k_i \alpha_i = \left(\prod_{j=0}^{i} k_j \right) \alpha_0, \quad i = 0, \ldots, N-1, \tag{2}$$

where

$$k_i = \frac{w_i T_{cp}}{(z_{i+1} T_{cm} + w_{i+1} T_{cp})}, \quad i = 0, \ldots, N-1.$$

Clearly, from Eqs. (1) and (2), there are N equations and $N+1$ unknowns. An additional equation, the normalization equation, is needed in order to solve this system of equations. The normalization equation is,

$$\alpha_0 + \alpha_1 + \cdots + \alpha_N = 1. \tag{3}$$

With the normalization equation, one then resolves the recursive equations, Eq. (1), to obtain the closed-form expression of α_0, the fraction of load assigned to the root processor. Once α_0 is known, the other processor load fractions can be obtained by substituting α_0 into Eq. (2) and solving recursively as follows:

$$\alpha_0 = \left[1 + \sum_{i=1}^{N} \left[\prod_{j=0}^{i-1} k_j \right] \right]^{-1}, \tag{4}$$

$$= \frac{1}{D} \prod_{i=1}^{N} (z_i T_{cm} + w_i T_{cp}), \tag{5}$$

$$\alpha_1 = \frac{1}{D} (w_0 T_{cp}) \prod_{i=2}^{N} (z_i T_{cm} + w_i T_{cp}), \tag{6}$$

$$\alpha_n = \frac{1}{D} \prod_{i=0}^{n-1} (w_i T_{cp}) \prod_{i=n+1}^{N} (z_i T_{cm} + w_i T_{cp}), \tag{7}$$

$$\alpha_N = \frac{1}{D} \prod_{i=0}^{N-1} (w_i T_{cp}), \tag{8}$$

where

$$D = \prod_{i=1}^{N} (z_i T_{cm} + w_i T_{cp}) + \sum_{n=1}^{N} \left(\left(\prod_{i=0}^{n-1} (w_i T_{cp}) \right) \times \left(\prod_{i=n+1}^{N} (z_i T_{cm} + w_i T_{cp}) \right) \right). \tag{9}$$

3. Sequencing and monetary cost

3.1. Sequencing

Consider a single level tree network with the following ordered set:

$$\Theta = \{p_0, (l_1, p_1), \ldots, (l_j, p_j), (l_{j+1}, p_{j+1}), \ldots, (l_N, p_N)\}.$$

This indicates that processor p_1 is the first processor that is assigned a fraction of load from p_0 followed by processor p_2 and so on. Link-processor pairs (l_j, p_j) and (l_{j+1}, p_{j+1}) in the ordered set Θ above may not necessarily be physically adjacent. Any change in the ordered set above is equivalent to a corresponding change in the sequence of load distribution. Therefore, sequencing is a mechanism that changes one ordered set to another ordered set. In this paper we seek to do this sequencing in a cost efficient manner. Note that load distribution in this context involves software control. No hardware changes are made to the network.

3.2. Link-processor monetary cost

The link-processor monetary cost for processing a fraction of load at any processor is defined as the cost incurred from utilizing the processor and its corresponding link in order to process the underlying fraction of load. We assume that the cost coefficients associated with links and processors are static. They do not change with either the level of load in progress or a period of time when the job arrives. This cost is defined only in terms of accounting for the duration during which the resource is busy serving the assigned divisible load. A linear cost model is assumed. Let:

c_n^p	the computing cost per second of utilizing the nth processor
c_n^l	the communication cost per second of utilizing the nth link
$c_n^p w_n$	the computing cost per unit load of utilizing the nth processor
$c_n^l z_n$	the communication cost per unit load of utilizing the nth link
$(c_n^p w_n + c_n^l z_n)$	the processing cost per unit load of the nth link-processor pair.

3.3. Total monetary cost

Total monetary cost is a cost incurred for a network to process an entire load. It is a linear addition of all individual link-processor costs incurred by utilizing individual link-processor pairs. This individual cost depends on the assigned fraction of load, which in turn is determined by an order of load distribution. Therefore, this total cost depends on the sequence of load distribution. In this paper, the total cost of processing an entire of load is the prime performance metric to be optimized. Define:

$$C_0 = c_0^p w_0 T_{cp}, \tag{10}$$

$$C_n = c_n^l z_n T_{cm} + c_n^p w_n T_{cp}, \quad n = 1, \ldots, N. \tag{11}$$

C_n	the cost of processing the entire of load on the nth processor
$\alpha_n C_n$	the cost of processing the assigned fraction of load (α_n) on the nth processor.

The total cost, C_{total}, is defined as a summation of the individual processing costs incurred at each link-processor pair. That is,

$$C_{total} = \alpha_0 C_0 + \sum_{n=1}^{N} \alpha_n C_n. \tag{12}$$

By substituting α_0 and all α_n from the previous section into Eq. (12) one obtains the total monetary cost expression that explicitly shows the jth and $(j+1)$st link-processor pairs as

$$\begin{aligned} C_{total} = \frac{1}{D} \Bigg\{ & \prod_{i=1}^{N} (z_i T_{cm} + w_i T_{cp})(c_0^p w_0 T_{cp}) \\ & + \sum_{n=1}^{j-1} \left[\left(\prod_{i=0}^{n-1} (w_i T_{cp}) \right) \times \left(\prod_{i=n+1}^{N} (z_i T_{cm} + w_i T_{cp})(c_n^l z_n T_{cm} + c_n^p w_n T_{cp}) \right) \right] \\ & + \prod_{i=0}^{j-1} (w_i T_{cp}) \prod_{i=j+1}^{N} (z_i T_{cm} + w_i T_{cp}) \times (c_j^l z_j T_{cm} + c_j^p w_j T_{cp}) \\ & + \prod_{i=0}^{j} (w_i T_{cp}) \prod_{i=j+2}^{N} (z_i T_{cm} + w_i T_{cp}) \times (c_{j+1}^l z_{j+1} T_{cm} + c_{j+1}^p w_{j+1} T_{cp}) \end{aligned}$$

$$+\sum_{n=j+2}^{N}\left[\prod_{i=0}^{n-1}(w_iT_{cp})\prod_{i=n+1}^{N}(z_iT_{cm}+w_iT_{cp})\times(c_n^lz_nT_{cm}+c_n^pw_nT_{cp})\right]\Bigg\}. \quad (13)$$

Since the total cost can be put in a simple form as

$$C_{total}=\frac{N}{D}. \quad (14)$$

Thus, the corresponding numerator, N, is the collection of terms within the curly braces. The corresponding denominator, D, with the terms due to the jth and the $(j+1)$st link-processor pairs explicitly shown, is

$$\begin{aligned}D=&\prod_{i=1}^{N}(z_iT_{cm}+w_iT_{cp})\\&+\sum_{n=1}^{j-1}\left(\prod_{i=0}^{n-1}(w_iT_{cp})\prod_{i=n+1}^{N}(z_iT_{cm}+w_iT_{cp})\right)\\&+\prod_{i=0}^{j-1}(w_iT_{cp})(z_{j+1}T_{cm}+w_{j+1}T_{cp})\\&\times\prod_{i=j+2}^{N}(z_iT_{cm}+w_iT_{cp})\\&+\prod_{i=0}^{j-1}(w_iT_{cp})(w_jT_{cp})\prod_{i=j+2}^{N}(z_iT_{cm}+w_iT_{cp})\\&+\sum_{n=j+2}^{N}\left(\left(\prod_{i=0}^{n-1}(w_iT_{cp})\right)\times\left(\prod_{i=n+1}^{N}(z_iT_{cm}+w_iT_{cp})\right)\right). \quad (15)\end{aligned}$$

4. Cost efficient sequencing

The heuristic algorithm will, at each iteration, swap the logical position of two processors which are logically adjacent in the load distribution sequence if it leads to a cost improvement. One can also find expressions for the cost of a swapped sequence as a ratio $C'_{total}=N'/D'$. Taking the difference $C_{total}-C'_{total}$ one can, with some algebra, find conditions under which swapping leads to an cost improvement. These conditions are summarized below.

Theorem 1. *In a single-level tree network if one of the following conditions is satisfied, then the total cost of the adjacent pairwise swapped sequence, $C'_{total}(\theta'_{(1,2,\ldots,\underline{j+1,j},\ldots,N)})$, is less than the total cost of the current sequence, $C_{total}(\theta_{(1,2,\ldots,\underline{j,j+1},\ldots,N)})$, for $1\leqslant j<N$. Otherwise $C'_{total}(\theta'_{(1,2,\ldots,\underline{j+1,j},\ldots,N)})$ is greater than or equal to $C_{total}(\theta_{(1,2,\ldots,\underline{j,j+1},\ldots,N)})$.*

(1) $z_j<z_{j+1}$ *and* $\frac{z_j}{C_j}<\frac{z_{j+1}}{C_{j+1}}$ *and* $C'_{total}<C_{total}$.
(2) $z_j=z_{j+1}$ *and* $\frac{z_j}{C_j}<\frac{z_{j+1}}{C_{j+1}}$.
(3) $z_j>z_{j+1}$ *and* $\frac{z_j}{C_j}<\frac{z_{j+1}}{C_{j+1}}$.
(4) $z_j>z_{j+1}$ *and* $\frac{z_j}{C_j}=\frac{z_{j+1}}{C_{j+1}}$.
(5) $z_j>z_{j+1}$ *and* $\frac{z_j}{C_j}>\frac{z_{j+1}}{C_{j+1}}$ *and* $C'_{total}<C_{total}$.

Corollary 1. *In a homogeneous single-level tree network where all link speeds are identical (i.e. a bus network) and all processor speeds are identical, the total cost, C_{total}, is minimized over all load distribution sequence if and only if the sequence of load distribution is arranged to satisfy the following condition*:

$$C_1\leqslant C_2\leqslant\cdots\leqslant C_N,$$

where

$$\begin{aligned}(c_1^lzT_{cp}+c_1^pwT_{cp})&\leqslant(c_2^lzT_{cp}+c_2^pwT_{cp})\leqslant\cdots\\&\leqslant(c_N^lzT_{cp}+c_N^pwT_{cp}).\end{aligned}$$

That is, the optimal load distribution sequence for a homogeneous bus network is one where load distribution is in non-decreasing order of the sum of the link and processor costs.

5. Performance results

5.1. Convergence study

One can develop an algorithm based on a greedy strategy, i.e., it makes the best choice at each iteration. It is also based on local search. A local search attempts to find a solution or a sequence better than the current one through a search in the neighborhood of the current sequence. In this paper two sequences are neighbors if one can be obtained from the other by one logically adjacent pairwise swap. There are thus $(N-1)$ sequences in the neighborhood of any sequence of N processors.

Based on the concept of greedy strategy and local search [1], at each step the algorithm searches its neighbors and finds the best one in terms of the lowest total cost and adopts it as the next sequence. It continues until no further improvement in C_{total} can be made, then it stops. The total cost is guaranteed to be improved in each step by the conditions in the theorems if the algorithm moves to a better sequence indicated by the theorems. By the aid of the theorem for adjacent pairwise swapping, the number of neighbors for potential candidates in a local search can be reduced. The sequence found using the greedy algorithm was compared to the best sequence found via an exhaustive search algorithm.

To evaluate the effectiveness of the proposed algorithm, a number of experiments were conducted on a

single-level tree network. A program was designed to generate a prescribed number of sets of the parameters w_i, z_i, c_i^l, c_i^p, T_{cm}, and T_{cp} randomly and uniformly in the interval $[0, 10]$ for w_i, z_i, $[0, 20]$ for c_i^l, c_i^p and $[0, 5]$ for T_{cm}, and T_{cp}. These parameters serve as the starting point of the greedy cost-efficient sequencing routine.

The first experiment in running the greedy cost efficient sequencing program on a single-level tree network, is performed for one root processor and five children processors. The total number of runs is 10 and for each run 10,000 random sets of network parameters were generated. The results from this experiment are that the greedy algorithm always converged to an optimal sequence, and also indicate that the optimal sequence is not necessarily unique.

The second experiment was conducted on a single-level tree network with a number of children processors varied from 4 to 7. For each size of network, 10,000 random sets of network parameters were generated. In the case of the number of children processors being 8, 1000 sets of parameters were generated. The results also show that the program always converged to a minimum total cost sequence and the optimal sequence was not necessarily unique. In the experiments run the greedy algorithm always converged to the minimum total cost sequence of load distribution regardless of the sequence it was initialized with.

The average number of iterations per number of children is 6.0 iterations for a 5 child tree, 23.8 iterations for a 10 child tree and 54.5 iterations for a 15 child tree. One numerical difficulty encountered was an inability of the greedy algorithm to generate cost differences large enough to allow convergence for more than about 22 children.

It should be noted that the solution quality appears to be problem dependent. For instance, a related work [6] considers the problem of arranging processor location in a star network (shifting processors from link to link) as part of a configuration design problem. For this problem it was found, that for small N, sub-optimal solutions were much more likely to occur using a straight forward implementation of a greedy algorithm than for the problem of this paper.

5.2. Convergence remarks

Two definitions that will be used below are now presented.

Definition 1. A compliant sequence is a sequence such that no adjacent pair of link-processors meets with any of the conditions of Theorem 1.

Definition 2. An optimal sequence is a sequence such that the associated total cost is less than or equal to that of any other load distribution sequence.

Based on earlier work published in [4] we conjectured that a sequence of load distribution is a compliant sequence if and only if it is an optimal total cost sequence. If this was true our greedy algorithm, which produces compliant solutions, would always produce optimal solutions. More recently in a companion paper [5] tabu search and simulated annealing were used to find solutions when the number of children pairs is higher than 8. For N greater than 20, network instances were found where the proposed greedy algorithm gives suboptimal solutions. That is, it provides (locally optimal) solutions with a higher total cost than the solutions obtained by tabu search and simulated annealing. This finding thus voids the validity of the only if part of above conjecture. We note also that the final sequences given by tabu search and simulated annealing were also found to be compliant sequences.

Nonetheless, the results from the companion paper indicates that the proposed greedy algorithm appears to work well for problems of small to medium size although it shows some suboptimal solutions for a large size network. Particularly, it is very attractive in terms of a running time, i.e., its running time is lower than those of the other two algorithms. For certain parameters the greedy algorithm is three times faster than implementations of tabu search and simulated annealing. This is even more pronounced when N is large.

6. Conclusions

It has been found that monetary cost based optimization of computer utility load distribution sequencing, even using generic assumptions of load divisibility, linear monetary costs and a star topology, leads to a fairly complex exercise in algebra. No simple analytic determination of an optimal load distribution sequence appears possible for this problem though analytic conditions for best greedy algorithm decisions were found. The determination of efficient (and often optimal) load distribution sequences is possible using a number of combinatorial algorithms. Open research problems this research leads to include sequencing for large and alternate network topologies based on monetary cost and alternative monetary cost modeling.

The growth of software leasing makes viable the leasing of processing time and proprietary algorithms on third-party machines. This could be a cost-effective means for the research community to maximize the impact of research computing budgets. If so, problems of the type discussed in this note will be of increasing interest.

Acknowledgments

The support of NSF Grant CCR-99-12331 is acknowledged.

References

[1] E. Aarts, J.K. Lenstra (Eds.), Local Search in Combinatorial Optimization, John Wiley, Chichester, England, 1997.

[2] C. Bennett, G.T. Timbrell, Application service providers: will they succeeed?, Inform. Sys. Frontiers 2 (2000) 195–211.

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

[4] S. Charcranoon, T.G. Robertazzi, S. Luryi, Cost efficient load sequencing in single-level tree networks, in: Proceedings of the 1998 Conference on Information Sciences and Systems, Princeton University, Princeton, NJ., March, 1998, also related: US Patent 6,370,560, S. Charcranoon, T.G. Robertazzi, S. Luryi, Load sharing controller for optimizing resource utilization, April 9, 2002.

[5] S. Charcranoon, T.G. Robertazzi, S. Luryi, Heuristic methods for optimizing computing and communication costs for networked computer utilities, submitted for publication.

[6] S. Charcranoon, T.G. Robertazzi, S. Luryi, Parallel processor configuration design with processing/transmission costs, IEEE Trans. Comput. 49 (2000) 987–991.

[7] M. Dorojevets, COOL approach to petaflops computing, in Parallel Computing Technologies, Lecture Notes on Computer Science, Vol. 1662, Springer-Verlag, Berlin, 1999, pp. 351–364.

[8] D. Ghose, T.G. Robertazzi (Eds.), Special issue of Cluster Comput. on Divisible Load Scheduling 6 (2003) 5–86.

[9] W.W. Gibbs, World wide widgets, Sci. Amer. 276 (1997) 48.

[10] H.-A. Jacobsen, O. Gunther, Middleware for software leasing over the internet, in: Proceedings of ACM E-Commerce 99, 1999, pp. 87–95.

[11] C. Kenyon, G. Cheliotis, Architecture requirements for commercializing grid resources, in: Proceedings of IEEE International Symposium on High Performance Distributed Computing, 2002, pp. 215–224.

[12] N. Marchand, H.-A. Jacobsen, An economic model to study dependencies between independent software vendors and application service providers, Electron. Commerce Res. 1 (2001) 315–334.

[13] R. Patnayakuni, N. Seth, Why license when you can rent? Risks and rewards of the application service provider model, in: Proceedings of ACM SIGCPR 2001, 2001, pp. 182–187.

[14] A. Plepys, Software renting—better business, better environment: the case of application service providing (ASP), in: Proceedings of 2002 IEEE International Symposium on Electronics and the Environment, 2002, pp. 53–58.

[15] J. Sohn, T.G. Robertazzi, S. Luryi, Optimizing computing costs using divisible load analysis, IEEE Trans. Parallel Distrib. Systems 9 (1998) 225–234, also related: US Patent 5,889,989, J. Sohn, T.G. Robertazzi, S. Luryi, Load sharing controller for optimizing monetary cost, March 30, 1999.

Saravut Charcranoon received the B.Eng in Electrical Engineering from Chulalongkorn University, Bangkok, Thailand in 1989, the M.S. in Electrical Engineering from Washington University in St.Louis, Missouri in 1993 and the Ph.D in Electrical Engineering from Stony Brook University in 1998. During 1989-1991 and 1993-1995, he worked as an engineer at the data communication section, the Communication Authority of Thailand, Bangkok, Thailand. Since 1998, he has been working as a research scientist in the network strategy department, Alcatel USA, Plano, Texas. His research interest is primarily in the control and management of networks and distributed systems, and network performance analysis.

Serge Luryi received his Ph.D. degree in physics in 1978 from the University of Toronto. In 1980 he joined Bell Laboratories in Murray Hill, NJ, and became interested in semiconductor devices. In 1994 Dr. Luryi joined Stony Brook University, where he is currently a Distinguished Professor and Chair of Electrical and Computer Engineering. He is also Director of the NY State Center for Advanced Sensor Technology.

During 1986-1990 Dr. Luryi served as the Editor of IEEE Transactions on Electron Devices. He was elected Fellow of the IEEE in 1989 and Fellow of the American Physical Society in 1993 for contributions to the theory of electron transport in low-dimensional systems and invention of novel electron devices. He is also a Fellow of the IEEE.

Thomas G. Robertazzi has been a member of the Dept. of Electrical and Computer Engineering at Stony Brook University since 1983, where he is currently a Professor. He received the Ph.D from Princeton University and the B.E.E. from The Cooper Union. His research interests include parallel system scheduling, grid computing, metacomputing, performance evaluation and wireless communications. Prof. Robertazzi has authored texts on performance evaluation and telecommunications network planning. He is a Senior Member of the IEEE.