# Data Intensive Grid Scheduling: Multiple Sources with Capacity Constraints

Han Min Wong
Dept. of Electrical and Computer Engineering
The National University of Singapore
4 Engineering Drive 3, Singapore 117576
engp0963@nus.edu.sg

Dantong Yu
Dept. of Physics
Brookhaven National Laboratory
Upton, NY 11973, USA
dtyu@bnl.gov

Bharadwaj Veeravalli
Dept. of Electrical and Computer Eng.
The National University of Singapore
4 Engineering Drive 3, Singapore 117576
elebv@nus.edu.sg

Thomas G. Robertazzi
Dept. of Electrical and Computer Eng.
Stony Brook University
Stony Brook, NY 11794, USA
tom@ece.sunysb.edu

**ABSTRACT**
In this paper, we apply divisible load theory to model the Grid scheduling problem involving multiple sources to multiple sinks, and present an optimized scheduling technique for this scenario. This scheduling technique can be easily extended to schedule resources with buffer space constraints. We provide a step-wise scheduling algorithm for these constraints. Two example calculations will show the practical utility and efficiency of DLT.

**KEY WORDS**
divisible load scheduling theory(DLT), Grid computing, heterogeneous computing, parallel computing, resource constraints

## 1 Introduction

Geographically distributed, heterogeneous collections of computers form a new concept of computing infrastructure, known as a computing "Grid". Grid computing and its applications are receiving an increasing amount of attention. One open problem is devising analytical performance models that well represent networked and integrated computing and communications. Existing tools for this purpose include stochastic queueing theory, which has been extensively developed over the years. A more recent approach [5, 11, 16] is what is referred to as divisible load theory.

Divisible load theory is designed to solve the challenging problem of allocating and scheduling computing resources on a Grid for thousands of independent tasks from large number of users. Divisible load theory involves the optimization of distributed computing problems where both communication and computation load is partitionable. Load may be divisible in fact or as an approximation (for instance a large number of small, independent tasks). Most divisible load theory uses a continuous mathematics, linear model which admits solution time optimization through linear equations or recursions, allows equivalent elements and other linear model features. Divisible load theory is fundamentally a deterministic theory, though some equivalencies to Markov chain modeling have been demonstrated [17]. Solutions generated from divisible load theory are surprisingly tractable.

A limitation in applying divisible load theory, as it has been developed to date, to Grid computing is that most of the literature involves a single source node distributing computing load over a network to multiple sink nodes. An appropriate modeling tool would need to be able to model multiple sources and sinks. Some existing work on multiple sources is [12] where tasks arrive according to a basic stochastic process to multiple nodes. However an analytical approach to Grid modeling that is exact and deterministic would be useful. To this end, in this paper we present a technique for scheduling divisible loads from multiple sources to multiple sinks, with and without buffer capacity constraints. We note that the results in this paper are a first step in applying divisible load theory to Grid scheduling. There is much that can be done beyond this.

This paper is organized as follows. In section 2, the problem is formulated. Optimal load distribution without buffer constraints appears in Section 3 and with buffer constraints appears in Section 4. The conclusion appears in Section 5.

### 1.1 Related Work

Since the 1988 origin of divisible load theory [6], scheduling in linear daisy chains, trees [9], hypercubes [8] and meshes [10] has been studied. Work on scheduling has considered multi-installment scheduling [7], fixed communication charges [18], finite buffers [4], Markov chain models [17], multiple rounds [13], detailed parameterizations and solution reporting optimization [19] and combinatorial optimization [14]. Almost all work to date has assumed load originates at a single node.

## 2 Grid Topology and Problem Formulation

In this section, we will consider a Grid architecture and formally define the problem we address. We consider a tightly coupled, bipartite multiprocessor system. In the Grid system, we assume that there are $N$ sources denoted as $S_1, S_2, ..., S_N$ and $M$ sinks denoted as $K_1, K_2, ..., K_M$. For each source, there is a direct link to all the sinks and we denote the link between $S_i$ and $K_j$ as $l_{i,j}$, $i = 1, .., N$, $j = 1, .., M$, respectively. Each source $S_i$ has a load, denoted by $L_i$ to process. Without loss of generality, we assume that all sources can send their loads to all the sinks simultaneously. Similarly, we also assume that all the sinks can receive load portions from all sources at the same time instant.

The objective in this study is to schedule all the $N$ loads among $M$ sink nodes such that the *processing time*, defined as the time instant when all loads have finished being processed by all the $M$ sinks, is minimal. The scheduling strategy is such that the scheduler (assumed to be resident in $S_1$) will first obtain the information about the size of the loads that other sources have in their local memory. The scheduler will then calculate and notify each source of the optimum amount of load that each source has to give to each sink. This information can be easily communicated via any means of standard or customized communication protocol and it would not incur any significant communication overhead.

The sources, upon knowing the amount of load that they should give to each sink respectively, will send the loads to all sinks simultaneously. The sinks will then start computing the loads immediately after they receive their respective loads. Following Kim's model [1], the sinks immediately start computing the load fractions as they start receiving them. It may be noted that we assume that each sink has adequate memory/buffer space to accommodate and process all the loads it receives from all the sources. We also assume that communication time is faster than computation time so no processor starves for load. In Section 4, we shall relax this assumption and study the impact of finite buffer space. We describe the actual load distribution strategy in the next section.

## 3 Design and Analysis of Load Distribution Strategy

In all the literature within the divisible load scheduling domain so far, an optimality criterion [3] that is used to derive an optimal solution is as follows. It states that in order to obtain an optimal processing time, it is necessary and sufficient that all the sinks that participate in the computation must stop at the same time instant. Otherwise, load could be redistributed to improve the processing time. We use this optimality principle in the design of our load distribution strategy.

We shall now introduce an index of definitions and notations that are used throughout this paper.

| | |
|---|---|
| $N$ | The total number of sources in the system |
| $M$ | The total number of sinks in the system |
| $S_i$ | The $i$-th source |
| $L_i$ | The loads in $S_i$ |
| $L$ | The sum of loads in the system, where $L = \sum_{i=1}^{N} L_i$ |
| $K_j$ | The $j$-th sink |
| $\alpha_{i,j}$ | The amount of load that $K_j$ will receive from $S_i$ |
| $\alpha_j$ | The fraction of $L$ that $K_j$ will receive from all sources, where $\sum_{j=1}^{M} \alpha_j = 1$ |
| $w_j$ | The inverse of the computing speed of $K_j$ |
| $l_{i,j}$ | The link between $S_i$ and $K_j$ |
| $z_{i,j}$ | The inverse of the link speed of $l_{i,j}$ |
| $T_{cp}$ | The computing intensity constant. A unit load can be processed in $w_j T_{cp}$ time by $K_j$ |
| $T_{cm}$ | The communication intensity constant. A unit load can be communicated in $z_{i,j} T_{cm}$ by $l_{i,j}$ |
| $t_j$ | The *finish time*, defined as the time instant when the computation ends, of $K_j$, where $t_j = \alpha_j L w_j T_{cp}$ |
| $T(M)$ | The *processing time*, defined as the time instant when all loads $L_i$, $i = 1, ..., N$ are processed by $M$ sinks, where $T(M) = max\{t_j,\ j = 1, ..., M\}$ |

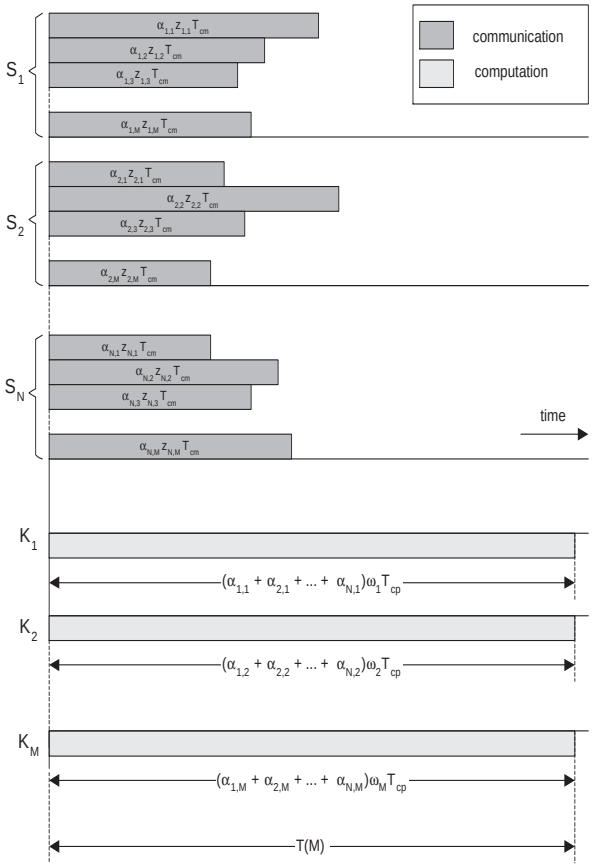


Figure 1. Timing diagram of the distribution strategy with $N$ sources and $M$ sinks

The timing diagram shown in Figure 1, represents the communication and computation (load distribution and processing) of the sources and sinks within the system, with the x-axis representing the time. From the timing diagram, we can see that,

$$\sum_{i=1}^{N} \alpha_{i,j} w_i T_{cp} = \sum_{i=1}^{N} \alpha_{i,j+1} w_{j+1} T_{cp},\ j = 1, ..., M-1 \quad (1)$$

As our objective is to determine the above optimal fractions $\alpha_{i,j}$, we impose the following condition in our strategy. Let $\alpha_{i,j} = \alpha_j L_i,\ i = 1, ..., N$ and $j = 1, .., M$, respectively. This condition essentially assumes that each sink receives a load that is proportional to the size of the load from the source. Moreover, each sink receives the same load fraction (percentage of total load) from each source. Without this condition, the system of equation is underconstrainted, and additional constraints need to be added for a unique solution. With this condition, the above equation simplifies to,

$$\sum_{i=1}^{N} \alpha_j L_i w_j = \sum_{i=1}^{N} \alpha_{j+1} L_i w_{j+1}\ ,\quad j = 1, ..., M-1 \quad (2)$$

Using the above equation together with the fact that $\sum_{i=1}^{M} \alpha_i = 1$, we have

$$\alpha_j = \frac{1}{w_j (\sum_{x=1}^{M} \frac{1}{w_x})}\ ,\quad j = 1, ..., M \quad (3)$$

Hence, the fraction of load that should be given by $S_i$ to $K_j$ is

$$\alpha_{i,j} = \frac{1}{w_j (\sum_{x=1}^{M} \frac{1}{w_x})} L_i \quad (4)$$

Thus the processing time is given by,

$$T(M) = \sum_{i=1}^{N} \alpha_{i,M} w_M T_{cp} \quad (5)$$

The following example clarifies the above computations. The system values in this example are directly taken from a recently reported study on Solenoidal Tracker At RHIC (STAR) simulation [2] used in large scale physics experiments. The STAR collaboration is a large international collaboration of about 400 high energy and nuclear physicists located at 40 institutions in the United States, France, Russia, Germany, Israel, Poland, and so on. After the Relativistic Heavy-Ion Collider at Brookhaven National Laboratory came on-line in 1999, STAR began data taking and concurrent data analysis that will last about ten years. STAR needs to perform data acquisition and analyzes over approximately 250 tera-bytes of raw data, 1 peta-bytes of derived and reconstructed data per year. Details on data acquisition and hardware can be found in the above paper.

**Example 1** : Consider a system with 3 sources and 4 sinks, with parameters $w_1 = 1.11 \times 10^{-9}$, $w_2 = 6.25 \times 10^{-10}$, $w_3 = 5.00 \times 10^{-10}$, $w_4 = 3.57 \times 10^{-10}$, and $T_{cp} = 6.52 \times 10^{14}$sec/load. We let the 3 sources have loads $L_1 = 5$, $L_2 = 2$, and $L_3 = 3$ unit loads, respectively. Using (4), we have the following values for $\alpha_{i,j}$ and the processing time is $T(4) = 8.93 \times 10^5$secs.

| | $S_1$ | $S_2$ | $S_3$ | $\sum \alpha_{i,j}$ |
|---|---|---|---|---|
| $K_1$ | 0.62 | 0.25 | 0.37 | 1.24 |
| $K_2$ | 1.09 | 0.44 | 0.66 | 2.19 |
| $K_3$ | 1.37 | 0.55 | 0.82 | 2.74 |
| $K_4$ | 1.92 | 0.76 | 1.15 | 3.83 |
| $L_i$ | 5.00 | 2.00 | 3.00 | 10.00 |

□

## 4 Scheduling Under Resource Constraints: Buffer Capacity Constraints

In our analysis so far, we assume that the buffer capacity of the sinks are infinite, i.e., a sink can hold any amount of load rendered by the sources. However, in reality, each sink always has a limit to the amount of buffer space that can be used. Further, in a multi-processor environment, sinks may be running multiple tasks such that it is required to share the available resources, hence there may be only a limited amount of buffer space that is allocated for processing particular loads. As a result, we are naturally confronted with the problem of scheduling divisible loads under buffer capacity constraints. In this section, we revisit the problem of scheduling $N$ loads among $M$ sinks, with buffer capacity constraints. We tune the IBS algorithm [4] proposed in the literature to solve the buffer space constraint problem. The IBS algorithm produces a minimum time solution given prespecified buffer constraints. The iterative IBS algorithm exhibits finite convergence and is discussed in [4]. We shall first introduce an index of notations that are used.

| | |
|---|---|
| $L_i^{(q)}$ | The remaining loads in $S_i$ on the $q$-th iteration |
| $L^{(q)}$ | The sum of remaining loads on the $q$-th iteration, where $L^{(0)} = L$. Hence $L^{(q)} = L^{(0)} - \sum_{x=1}^{q-1} Y^{(x)} L^{(x)}$ |
| $Y^{(q)}$ | Fraction of the load $L^{(q)}$ that should be taken into consideration on the $q$-th installment. Where $Y^{(q)} \leq 1$. |
| $\alpha_{i,j}^{(q)}$ | The amount of load given from $S_i$ to $K_j$ in the $q$-th iteration. |
| $\alpha_j^{(q)}$ | The fraction of load from $L^{(q)}$ that $K_j$ should take in the $q$-th iteration |
| $B_j$ | The total amount of buffer space in $K_j$ |
| $B_j^{(q)}$ | The available buffer space in $K_j$ in the $q$-th iteration |
| $P_{all}$ | Set of all the sinks in the system |
| $P_{full}^{(q)}$ | Set of sinks with no available buffer space in the $q$-th iteration |
| $P_{vac}^{(q)}$ | Set of sinks with available buffer space in the $q$-th iteration |

**Initial state:**
$L_i^{(0)} = L_i,\ L^{(0)} = \sum_{i=1}^{N} L_i^{(0)}, P_{vac}^{(0)} = P_{all},$
$P_{full}^{(0)} = \emptyset,\ B_j^{(0)} = B_j$
$q = 0$

Do {
$q = q + 1$
**Phase 1**: Determine $\alpha_{i,j}^{(q)}$
$\alpha_j^{(q)} = 1/(w_j \sum_{x=1}^{M} \frac{1}{w_x})\ ,\ \ \forall K_j \in P_{vac}^{(q-1)}$
$\alpha_j^{(q)} = 0\ ,\ \ \forall K_j \in P_{full}^{(q-1)}$
$Y^{(q)} = \min\{B_j^{(q-1)}/(\alpha_j^{(q)} L^{(q-1)})\ ,\ \forall K_j \in P_{vac}^{(q-1)}\}$
If $(Y^{(q)} > 1)$ $\{Y^{(q)} = 1\}$
$\alpha_{i,j}^{(q)} = Y^{(q)} \alpha_j^{(q)} L_i^{(q-1)}\ ,\ \forall K_j \in P_{all},\ \forall S_i$
**Phase 2**: Update
$L_i^{(q)} = L_i^{(q-1)} - Y^{(q)} L_i^{(q-1)}$
$L^{(q)} = \sum_{i=1}^{N} L_i^{(q)}$
$B_j^{(q)} = B_j^{(q-1)} - Y^{(q)} \alpha_j^{(q)} L^{(q-1)}\ ,\ \forall K_j \in P_{vac}^{(q)}$
$P_{full}^{(q)} = P_{full}^{(q-1)}$
If $(B_j^{(q)} = 0)$ $\{P_{full}^{(q)} = P_{full}^{(q)} \cup K_j\}$
$P_{vac}^{(q)} = P_{all} - P_{full}^{(q)}$
} while $L^{(q)} > 0$

$\alpha_{i,j} = \sum_{x=1}^{q} \alpha_{i,j}^{(x)}\ ,\ \forall K_j \in P_{all},\ \forall S_i$

Table 1. Pseudocode for the modified IBS algorithm

The algorithm we propose is basically a recursive algorithm that attempts to fill up one or more sinks' buffer space at every iteration by following the load distribution suggested in Section 3. When a sink's buffer is completely filled up, it will no longer be considered for scheduling in the next iteration. On the other hand, in any iteration, if the remaining load is not enough to completely consume any buffer, then the suggested distribution by (4) will be used. The algorithm is presented in its pseudocode in Table 1.

It may be noted that, although the algorithm will suggest a load portion to be given to all sinks at every iteration, the distribution strategy remains the same as presented in Section 2, where $\alpha_{i,j}$ is the sum of all suggested loads given by $S_i$ to $K_j$ in all iterations. The following example illustrates the working of the algorithm.

**Example 2:** In this example, we consider the system and loads in Example 1 with sinks having buffer capacities as follows: $B_1 = 6$, $B_2 = 5$, $B_3 = 2$, and $B_4 = 2$, respectively. Using the algorithm, we have the following values for $\alpha_{i,j}^{(q)}$

| $q = 1$ | $S_1$ | $S_2$ | $S_3$ | $\sum \alpha_{i,j}^{(1)}$ | $B_j^{(1)}$ |
|---|---|---|---|---|---|
| $K_1$ | 0.32 | 0.13 | 0.19 | 0.64 | 5.36 |
| $K_2$ | 0.57 | 0.23 | 0.34 | 1.14 | 3.86 |
| $K_3$ | 0.71 | 0.29 | 0.43 | 1.43 | 0.57 |
| $K_4$ | 1.00 | 0.40 | 0.60 | 2.00 | 0.00 |
| $q = 2$ | $S_1$ | $S_2$ | $S_3$ | $\sum \alpha_{i,j}^{(2)}$ | $B_j^{(2)}$ |
| $K_1$ | 0.13 | 0.05 | 0.08 | 0.26 | 5.10 |
| $K_2$ | 0.23 | 0.09 | 0.14 | 0.46 | 3.40 |
| $K_3$ | 0.29 | 0.11 | 0.17 | 0.57 | 0.00 |
| $K_4$ | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| $q = 3$ | $S_1$ | $S_2$ | $S_3$ | $\sum \alpha_{i,j}^{(3)}$ | $B_j^{(3)}$ |
| $K_1$ | 0.63 | 0.25 | 0.38 | 1.26 | 3.84 |
| $K_2$ | 1.12 | 0.45 | 0.67 | 2.24 | 1.16 |
| $K_3$ | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| $K_4$ | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |

From, the above results, we observe that the buffer of $K_4$ and $K_3$ are fully consumed at the first and second iteration, respectively. At the final iteration, the remaining load is insufficient to completely fill up either the buffer of $K_1$ or $K_2$, hence the distribution suggested by (4) is used. The values for $\alpha_{i,j}$ are

| | $S_1$ | $S_2$ | $S_3$ | $\sum \alpha_{i,j}$ |
|---|---|---|---|---|
| $K_1$ | 1.08 | 0.43 | 0.65 | 2.16 |
| $K_2$ | 1.92 | 0.77 | 1.15 | 3.84 |
| $K_3$ | 1.00 | 0.40 | 0.60 | 2.00 |
| $K_4$ | 1.00 | 0.40 | 0.60 | 2.00 |
| $L_i$ | 5.00 | 2.00 | 3.00 | 10.00 |

Using the above values, the processing time is $1.56 \times 10^6$ seconds. As expected, the processing time is longer than that of Example 1. Also, note that the algorithm ensures that buffers of the faster sinks are fully consumed before the slower ones. This is to guarantee efficiency of the distribution under such capacity constraints. Further, at the final installment, when the remaining load is insufficient to completely fill up the available buffers, the load distribution suggested by (4) is used. As a result, the sinks with available buffers, will stop processing at the same time, hence ensuring an optimal solution. □

## 5 Conclusions

The problem of scheduling computationally intensive loads on Grid platforms is addressed in this paper. We used the divisible load paradigm approach to derive closed-form solutions for processing time and also addressed the problem under resource (buffer) constraints. We first considered the case where the buffers of the sinks are assumed to be infinite. In designing our strategy, the optimality principle was utilized to ensure an optimum solution. We later relaxed the assumption and considered the case where the buffer capacities at the sinks are limited. A modified IBS algorithm was proposed to solve the problem. The theoretical findings are demonstrated via examples using a practical Grid system [2].

Our study has demonstrated the relevance of the DLT approach to handle actual problems on Grid architectures. Again, this paper is an initial study on the multiple source problem in grid scheduling using divisible load theory. Much more can be done, such as an attempt to implement the strategies in this paper and to extend these results to more general Grid topologies.

## 6 Acknowledgments

Thomas Robertazzi's research was funded by NSF grant CCR-99-12331. Bharadwaji Veeravalli's research is supported by Broadband21 SingAREN project # R-263-000-198-305.

## References

[1] Kim, H.-J., A Novel Optimal Load Distribution Algorithm for Divisible Loads", *Special Issue on Divisible Load Scheduling in Cluster Computing*, Kluwer Academic Publishers, 2003.

[2] Yu, D., and Robertazzi T.G., Divisible Load Scheduling for Grid Computing, *15th IASTED International Conference Parallel and Distributed Computing and Systems*, Marina del Rey, CA, USA, 2003.

[3] Bharadwaj, V., D. Ghose, and T. G. Robertazzi, Divisible Load Theory: A New Paradigm for Load Scheduling in Distributed Systems, *Special Issue on Divisible Load Scheduling in Cluster Computing*, Kluwer Academic Publishers, 2003.

[4] Li, X., V. Bharadwaj, and C. C. Ko, Divisible Load Scheduling on Single-level Tree Networks with Buffers Constraints, *IEEE Transactions on Aerospace and Electronic Systems*, 36(4), 2000, 1298-1308.

[5] Bharadwaj, V., Ghose, D., Mani, V. and Robertazzi, T.G., *Scheduling Divisible Loads in Parallel and Distributed Systems* ( IEEE Computer Society Press, Los Alamitos CA, 1996, 292 pages).

[6] Cheng, Y.C. and Robertazzi, T.G., Distributed Computation with Communication Delays, *IEEE Transactions on Aerospace and Electronic Systems*, 24(6), 1988, 700-712.

[7] Bharadwaj, V., Ghose, D. and Mani, V., Multi-installment Load Distribution in Tree Networks with Delays, *IEEE Transactions on Aerospace and Electronic Systems*, 31(2), 1995, 555-567.

[8] Blazewicz, J. and Drozdowski, M., Scheduling Divisible Jobs on Hypercubes, *Parallel Computing*, Vol. 21, 1995, 1945-1956.

[9] Kim, H.J., Jee, G.-I. and Lee, J.G., Optimal Load Distribution for Tree Network Processors, *IEEE Transactions on Aerospace and Electronic Systems*, 32(2), 1996, 607-612.

[10] Drozdowski, M. and Glazek, W., Scheduling Divisible Loads in a Three-Dimensional Mesh of Processors, *Parallel Computing*, 25(4), 1999, 381-404.

[11] Bharadwaj, V., Ghose, D., and Robertazzi, T.G., Divisible Load Theory: A New Paradigm for Load Scheduling in Distributed Systems, in Ghose, D. and Robertazzi, editors, special issue of *Cluster Computing* on Divisible Load Scheduling, 6(1), 2003, 7-18.

[12] Ko K. and Robertazzi, T.G., Scheduling in an Environment of Multiple Job Submissions, *Proceedings of the 2002 Conference on Information Sciences and Systems*, Princeton University, Princeton NJ, USA, 2002, six pages.

[13] Yang. Y. and Casanova, H., UMR: A Multi-Round Algorithm for Scheduling Divisible Workloads, *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'03)*, Nice, France, 2003.

[14] Dutot, P.-F., Divisible Load on Heterogeneous Linear Array, *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'03)*, Nice, France, 2003.

[15] Adler, M., Gong, Y., and Rosenberg, A.L., Optimal Sharing of Bags of Tasks in Heterogeneous Clusters, *Proceeding of SPAA03*, 2003.

[16] Robertazzi, T., Ten Reasons to Use Divisible Load Theory, *Computer*, May, 2003.

[17] Moges, M. and Robertazzi, T.G., Optimal Divisible Load Scheduling and Markov Chain Models, *Proceedings of the 2003 Conference on Information Sciences and Systems*, The Johns Hopkins University, Baltimore, MD, USA, 2003.

[18] Blazewicz, J. and Drozdowski, M., Distributed Processing of Divisible Jobs with Communication Startup Costs, *Discrete Applied Mathematics*, 76(1-3), 1997, 21-41.

[19] Rosenberg, A.L., Sharing Partitionable Workload in Heterogeneous NOWs: Greedier is Not Better, *Proc. of IEEE International Conference on Cluster Computing*, Newport Beach, CA, USA, 2001, 124-131.