A Parallel Transferable Uniform Multi-Round Algorithm in Heterogeneous Distributed Computing Environment

Hiroshi Yamamoto*, Masato Tsuru, and Yuji Oie

Department of Computer Science and Electronics,
Kyushu Institute of Technology, Kawazu 680-4, Iizuka, 820-8502 Japan
yamamoto@infonet.cse.kyutech.ac.jp, {tsuru, oie}@cse.kyutech.ac.jp

Abstract. The performance of parallel computing systems using the master/worker model for distributed grid computing tends to be degraded when large data sets have to be dealt with, due to the impact of data transmission time. In our previous study, we proposed a parallel transferable uniform multi-round algorithm (PTUMR), which efficiently mitigated this impact by allowing chunks to be transmitted in parallel to workers in environments that were homogeneous in terms of workers' computation and communication capacities. The proposed algorithm outperformed the uniform multi-round algorithm (UMR) in terms of application turnaround time, but it could not be directly adapted to heterogeneous environments. In this paper, therefore, we propose an extended version of PTUMR suitable for heterogeneous environments. This algorithm divides workers into appropriate groups based on both computation and communication capacities of individual workers, and then treats each group of workers as one virtual worker. The new PTUMR algorithm is shown through performance evaluations to significantly mitigate the adverse effects of data transmission time between master and workers compared with UMR, achieving turnaround times close to the theoretical lower limits even in heterogeneous environments.

Keywords: Grid Computing, Master/Worker Model, Divisible Workload, Multi-Round Scheduling, UMR.

1 Introduction

Grid computing has recently increased in popularity for distributed applications [1,2]. The master/worker model is suited to grid computing environments involving a large number of computers that differ in resource capacities. In this model, a master with application tasks dispatches subtasks to several workers, which process the data allocated by the master. A typical instance of applications based on the master/worker model is a divisible workload application

* The author is now working in FUJITSU LABORATORIES LTD. The contact e-mail address is hiro-yamamoto@jp.fujitsu.com

M. Gerndt and D. Kranzlmüller (Eds.): HPCC 2006, LNCS 4208, pp. 51–60, 2006.

[3,4,5], where the master divides the application data into an arbitrary number of chunks and then dispatches them to multiple workers. For a given application, it is assumed that computation and transmission times for a chunk are roughly proportional to the size of the chunk.

The existing uniform multi-round algorithm (UMR), can handle an application having a large amount of data in a 'multiple-round' manner to overlap the time required for communication with that required for computation [3,6,7,8]. However, it utilizes sequential transmission model, i.e. the master transmits data to one worker at a time [9,10]. In actual networks where the master and workers are connected via a heterogeneous network, it is unable to minimize the adverse effects of data transmission on the application turnaround time.

Therefore, in our previous study, we proposed a new scheduling algorithm, parallel transferable uniform multi-round (PTUMR) adapted to the heterogeneous network [11]. The proposed algorithm fully utilizes the high-speed data transmission capacity of the heterogeneous network by allowing the master to transmit application data to multiple workers simultaneously. However it is not adaptive to a heterogeneous environment containing workers with varying resource capacities.

The contributions of this paper are two-fold. Firstly, we extend the PTUMR so that it can be applied in a heterogeneous environment. The master divides workers into appropriate groups based on both computation and communication capacities of individual workers, and then treats the set of workers in a group as one virtual worker. After that, the master optimally transmits chunks to the virtual workers sequentially as in UMR. Secondly, we evaluate the efficiency of the new PTUMR in various environments. The proposed algorithm reduces the adverse effects of data transmission time on application turnaround time to a greater extent than the conventional UMR by handling heterogeneity in terms of workers' capacities and the network model, allowing turnaround times close to the theoretical lower limit to be achieved.

This paper is organized as follows. In Section 2, the conceptual basis for multiple-round scheduling and the conventional UMR algorithm are introduced. The proposed PTUMR algorithm is presented in Section 3, and its performance is investigated in Section 4. The conclusion follows in Section 5.

2 The Conventional UMR Scheduling Algorithm

Recently, a number of scheduling methods have been proposed in which the master dispatches data to workers in a multiple-round manner in order to overlap communication with computation and thereby reduce the application turnaround time. Figure 1 shows a simple example of this scenario where the master dispatches a workload of the application to a worker. In this figure, a black rectangle represents the fixed-length overhead for one round of computation and a gray rectangle represents the fixed-length overhead in one round of data transmission. In multiple-round scheduling the entire application data set W [units] is divided into multiple chunks of arbitrary size and processed in M rounds, which can reduce

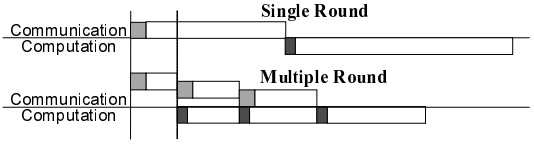

Fig. 1. Multiple-round scheduling

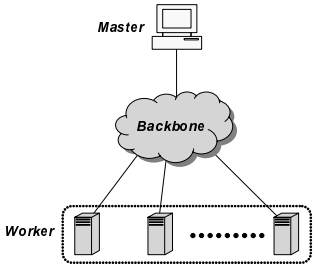

Fig. 2. Distributed computing model

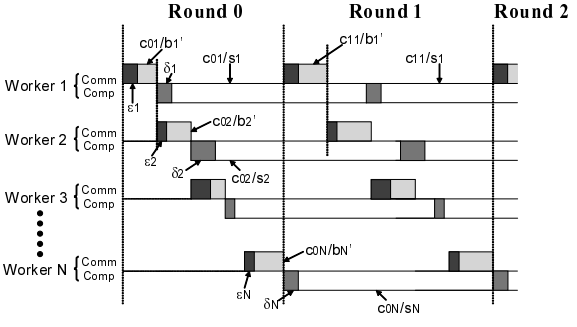

Fig. 3. Timing chart of data transmission and worker processing under UMR

the adverse effects of data transmission time on the application turnaround time. However, the use of a large number of rounds results in an increase in the total overhead. Thus, optimizing the number of rounds so as to minimize the application turnaround time is a key issue in multiple-round scheduling.

UMR is an example of a multiple-round scheduling algorithm [6,7]. The distributed computing model for UMR is shown in Fig. 2. The master and N workers are connected to a high-speed network that is free of bottlenecks. This model has heterogeneity in terms of the computation and communication capacities of workers: each worker i has associated with its computation speed s_i [units/s], data transmission capacity b_i [units/s] of the link attached to the worker, and overheads δ_i [s], ϵ_i [s] added to the computation time and data transmission time, respectively. Furthermore, the data transmission capacity of the link attached to the master is denoted by b_0 [units/s].

UMR adopts the sequential transmission model whereby the master transmits a chunk to one worker at a time. Therefore, the actual data transmission rate b'_i between the master and worker i has to be bounded above by $\min\{b_i, b_0\}$. Figure 3 illustrates how the data is transmitted to workers and then processed under UMR, where the size of the chunk allocated to the worker i in Round j is denoted by c_{ji} [units]. The master determines the amount of chunks allocated to each worker in such a way that the computation time becomes identical for all workers during a round. To reduce data transmission time in the first round, relatively small chunks are transmitted to workers in this round, and the size of chunks then grows exponentially in subsequent rounds.

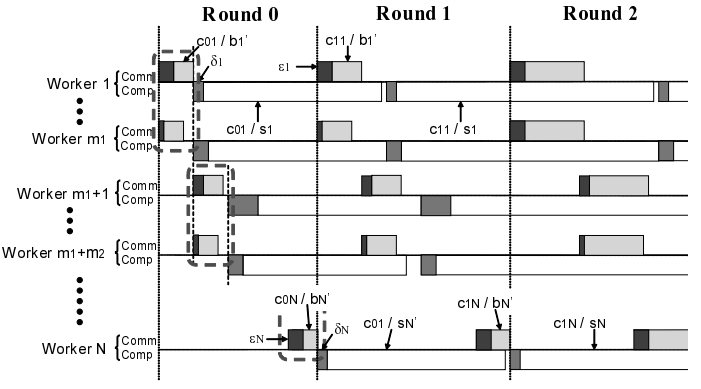

Fig. 4. Timing chart of data transmission and worker processing under PTUMR

3 The PTUMR Scheduling Algorithm

The PTUMR algorithm determines how the application data should be divided and when the data should be transmitted to workers in a network environment that allows the master to transmit data to multiple workers in parallel (Fig. 4), assuming that ϵ_i can be overlapped among concurrent transmissions. More precisely, the PTUMR divides workers into appropriate groups, and treats the set of workers in each group as a single virtual worker. Then the master transmits chunks to virtual workers sequentially, as in UMR.

After appropriately grouping the workers, the PTUMR algorithm analytically determines the appropriate number M^+ of rounds so that the application turnaround time for the total amount W of application data is minimized.

We assume that the values b_0 and $b_i (i = 1, 2, \ldots, N)$ are known to the master, and also that it can control the rate of data transmission b'_i to worker i to be within the range $[0, \min(b_0, b_i)]$. Note that such control can be achieved under the TCP by constraining the sending socket buffer size.

3.1 Computation and Data Transmission Capacities of a Virtual Worker

This subsection shows how to derive the resource capacity of a virtual worker in terms of the resource capacities of its members. The set of workers composing the virtual worker k is denoted by L_k and the number of workers in L_k by m_k.

In order to minimize the computation time of the virtual worker, the size c_{ji} of chunk allocated to worker i in Round $j (= 0, \ldots, M-1)$ should be proportional to its computation speed s_i [3]. In addition, we take the overhead Δ_k of virtual worker k to be the largest overhead δ_i among all workers in L_k. Then, the computation time of the virtual worker k in Round j is given by

$$T_{comp_{jk}} = \frac{C_{jk}}{\sum_{i \in L_k} s_i} + \max_{i \in L_k}\{\delta_i\} = \frac{C_{jk}}{S_k} + \Delta_k. \quad \left(C_{jk} = \sum_{i \in L_k} c_{ji} \right) \tag{1}$$

where C_{jk} is defined as the total size of chunks allocated to all workers in L_k in Round j, and S_k denotes the computation speed of virtual worker k.

Next, we assume that the data transmission time in each round is identical for all workers in L_k by limiting the data transmission rate b'_i ($\leq b_i$) to each worker i. In addition, we define that the overhead E_k of virtual worker k is the largest overhead ϵ_i among all workers in L_k. Then, the data transmission time of the virtual worker k in Round j is given by

$$T_{comm_{jk}} = \frac{C_{jk}}{\sum_{i \in L_k} b'_i} + \max_{i \in L_k}\{\epsilon_i\} = \frac{C_{jk}}{B_k} + E_k. \tag{2}$$

3.2 The Grouping Method

Since the resource capacity of the virtual worker depends on those of its members, the grouping method strongly affects the performance of PTUMR. The grouping method of PTUMR consists of two steps.

1. All workers are sorted and given serial numbers in ascending order of $r_i = s_i / \min\{b_0, b_i\}$. The workers are then divided into several groups according to the following equation.

$$m_k = \max\left\{ m \;\middle|\; \sum_{l=l_k}^{l_k+m-1} b_l \leq b_0 \right\} + x, \quad l_{k+1} = l_k + m_k. \tag{3}$$

   where l_k indicates the serial number of the first worker composing the virtual worker k and x indicates the number of additional workers added to the group after the number of worker has been chosen in a way to make full use of the master-network bandwidth. Note that the optimal value of x cannot be derived analytically. However, in our extensive performance evaluation, we found ten or more additional workers improved the performance to a nearly optimal point in various conditions.

1'. The master groups workers whose resource capacities are close to each other according to the following equation.

$$m_k = \min\{m_k \text{ in (3)},\ m'_k\},\ m'_k = \max\left\{ m \middle| r_{l_k+m-1} \leq \frac{1.5 \times \sum_{l=l_k}^{l_k+m-2} r_l}{m-1} \right\}. \tag{4}$$

   If r_i of the next worker is 1.5 times larger than the average r_i of all workers already selected for the group, this next worker will not be included.

2. The virtual workers are sorted in ascending order of $R_k = S_k / B_k$, and the number N_v of virtual workers utilized for application processing is chosen according to the following equation.

$$N_v = \max\left\{ n \;\middle|\; \sum_{k=1}^{n} R_k < 1 \right\}. \tag{5}$$

Step 1 presents a basic grouping method which attempts to preferentially select workers with larger r_i, as happens in UMR [7], and to fully utilize the bandwidth of the master-network link. Furthermore, the additional number x of workers aims at reducing the number of steps required to transmit data to all workers, which allows overlapping of the overhead for more workers.

However, in more heterogeneous environments, a virtual worker determined by Step 1 may include some workers with much lower r_i than others, which leads to critical degradation of the resource capacity of the virtual worker. Therefore, when heterogeneity is high, the basic grouping method does not result in efficient execution of the application (shown later in Section 4.1). Step 1' proposes a modified PTUMR, PTUMR with Grouping Threshold (GT), which is restricted to make a group of workers with similar resource capacities.

Step 2 then preferentially selects virtual workers with larger R_k, and limits the number of virtual workers so as to prevent the allocation of application tasks to a virtual worker having low capacity.

3.3 Derivation of Parameters That Result in Almost Minimal Turnaround Time

The new scheduling algorithm, PTUMR, determines the number M^+ of rounds that is nearly optimal in terms of minimizing application turnaround time. The application turnaround time T_{real} is determined by given parameters (the number M of rounds and the size C_{jk} of chunk allocated to virtual worker k in Round j). However, since T_{real} is difficult to express analytically, we instead derive the ideal application turnaround time T_{ideal} under the (ideal) assumption that no virtual worker ever enters the idle computation state once it has received its first chunk of data. In addition, we also assume that the time required to compute chunks received in each round is identical for all virtual workers.

We denote by $w_j (= \sum_{k=1}^{N_v} C_{jk})$ the total amount of chunk to be allocated to virtual workers in Round j, and from Eq. (1), the relation between w_j and the size C_{jk} of chunk allocated to the virtual worker k is given by

$$C_{jk} = \alpha_k \times w_j + \beta_k, \tag{6}$$
$$\left(\alpha_k = \frac{S_k}{\sum_{k=1}^{N_v} S_k}, \quad \beta_k = \frac{S_k \times \sum_{k=1}^{N_v} \{S_k \times \Delta_k\}}{\sum_{k=1}^{N_v} S_k} - S_k \times \Delta_k. \right)$$

In addition, from Eqs. (1) and (2), the total amount of chunk w_j for Round j can be determined by the total chunk size w_0 in the first round as follows.

$$w_j = \theta^j (w_0 - \gamma) + \gamma, \quad \left(\theta = \frac{1/\sum_{k=1}^{N_v} S_k}{\sum_{k=1}^{N_v} \{\alpha_k / B_k\}}, \right. \tag{7}$$
$$\left. \gamma = \frac{\sum_{k=1}^{N_v} \{S_k \times \Delta_k\} / \sum_{k=1}^{N_v} S_k - \sum_{k=1}^{N_v} \{\beta_k / B_k\} - \sum_{k=1}^{N_v} E_k}{\sum_{k=1}^{N_v} \{\alpha_k / B_k\} - 1/\sum_{k=1}^{N_v} S_k}. \right)$$

The application turnaround time T_{ideal} under the ideal assumption can be derived as a function of the number M of rounds, as follows.

$$T_{ideal} = \frac{1}{\sum_{k=1}^{N_v} S_k} \left\{ W + M \times \sum_{k=1}^{N_v} (S_k \times \Delta_k) + \sum_{k=1}^{N_v} \left[S_k \times \sum_{t=1}^{k} \left(\frac{\alpha_t \times \left(\frac{1-\theta}{1-\theta^M} \times (W - M\gamma) + \gamma \right) + \beta_t}{B_t} + E_t \right) \right] \right\}. \quad (8)$$

Due to space limitation, derivation of the application turnaround time in detail is omitted.

Let M^* denote the real value minimizing T_{ideal} in Eq. (8), which can be obtained by solving $\frac{\partial T_{ideal}}{\partial M} = 0$. Then, it is necessary to determine an appropriate number M^+ of rounds as an integer expected to nearly minimize T_{real} if M^* is not an integer. There are four possible integers to consider: $\lfloor M^* \rfloor - 1$, $\lfloor M^* \rfloor$, $\lceil M^* \rceil$ and $\lceil M^* \rceil + 1$. We can choose one among them in such a way as to minimize T_{real}.

4 Performance Evaluation

In this study, we assume, as was the case in the study proposing UMR [7], that the computation speed s_i, the worker-network link capacity b_i, and the latency parameters ϵ_i and δ_i corresponding to the related overheads of workers, are distributed uniformly within the following range.

$$\left((1 - \sqrt{3} \times het) \times mean, (1 + \sqrt{3} \times het) \times mean \right). \quad (9)$$

where *het* represents the heterogeneity of each resource capacity in the environment. We employ a coefficient of variation of each resource capacity as *het*. In addition, *mean* can be set to the average capacity over all workers, namely $\bar{s}$, $\bar{b}$, $\bar{\delta}$, and $\bar{\epsilon}$ listed in Tab 1.

The effectiveness of PTUMR is evaluated by comparing the achievable turnaround time $T_{real}(M^+)$ with the lower bound T_{bound}, which corresponds to the best possible turnaround time in an environment where the network resources are sufficient to ensure negligible data transmission times and any latency corresponding to related overheads is ignored. From Eq. (8), T_{bound} is obtained as follows.

$$T_{bound} = \frac{W}{\sum_{i=1}^{N} s_i}. \quad (10)$$

For a given parameter set (heterogeneity *het* and average resource capacities $\bar{s}$, $\bar{b}$, $\bar{\delta}$ and $\bar{\epsilon}$), 100 experiments were conducted. The average and maximum application turnaround time T_{real} in the 100 experiments was then used as a measure of performance of the scheduling algorithms.

Table 1. Model parameters and their values examined in performance evaluation

W	100, 500, 1000, 5000, 10000 [units]
$\bar{s}$	1 [units/s]
b_0	200, 400, $\cdots$, 2000 [units/s]
$\bar{b}$	200 [units/s]
$\bar{\epsilon}$	0.001, 0.01 [s]
$\bar{\delta}$	0.1 [s]

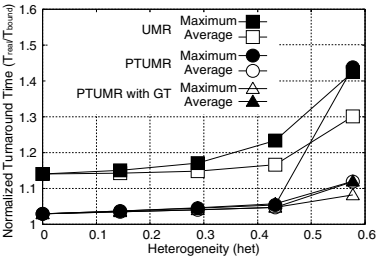

Fig. 5. Impact of heterogeneity

4.1 The Impact of Heterogeneous Resource Capacities

First, we investigated the effect of the heterogeneity *het* on the performance of our scheduling algorithms, UMR and PTUMR. In our evaluation model, we assumed a total amount W of application data of 1000, a master-network transmission capacity b_0 of 1000, an average overhead $\bar{\epsilon}$ at the start of the data transmission of 0.01, and 100 workers (N). When we evaluated the impact of the heterogeneity *het* of each resource capacity, all resource capacities s_i, b_i, δ_i and ϵ_i of each worker were randomly chosen according to Eq. (9).

Figure 5 shows the average and maximum normalized turnaround times T_{real}/T_{bound} for 100 experiments as a function of the heterogeneity *het*, where the number x of additional workers under PTUMR was set to 10. As shown in Fig. 5, regardless of *het*, normal PTUMR is superior to conventional UMR in terms of average normalized turnaround time. However, when the heterogeneity is high, the application turnaround time of normal PTUMR in the worst case becomes larger than that of UMR. In contrast, PTUMR with GT can achieve an application turnaround time close to the lower bound even in the worst case. It is apparent from these results that PTUMR with GT can achieve an excellent turnaround time by grouping workers in an appropriate way.

In the following section, we will consider only PTUMR with GT with the number x of addition workers of 10. In addition we will investigate the performance of the scheduling algorithms in highly heterogeneous environment, namely $het = \frac{\sqrt{3}}{4}$.

4.2 The Impact of Network Resources

The impact of network resources is examined here by assuming a total workload W of 1000 and 100 workers (N). Figure 6 shows the average normalized turnaround time T_{real}/T_{bound}, as a function of the master-network link capacity b_0. Even if b_0 increases, the UMR algorithm cannot effectively utilize the additional network capacity. By contrast, the application turnaround time under PTUMR decreases with increasing b_0 because the algorithm can utilize the full capacity of b_0 by transmitting chunks to multiple workers in parallel. Furthermore, PTUMR achieves T_{real} close to its lower bound T_{bound} across a wide range

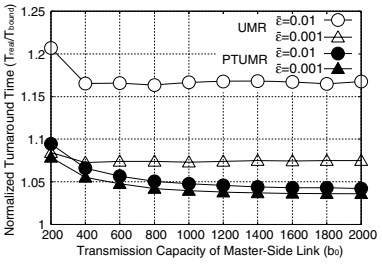

Fig. 6. Impact of network resources

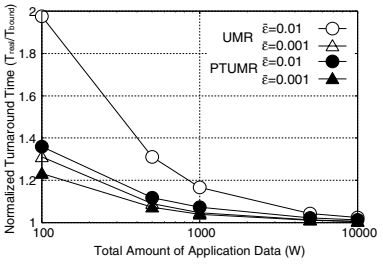

Fig. 7. Impact of total workload

of overhead $\bar{\epsilon}$. This is because PTUMR can reduce the impact of the overhead by aggressively overlapping the overhead ϵ_i for multiple workers.

This evaluation demonstrates that the PTUMR algorithm can achieve application turnaround time quite close to the lower bound through effective utilization of the transmission capacity of the master-network link and the overlapping of overheads for multiple workers.

4.3 The Impact of Total Workload

Finally, the effect of the total amount W of application data is evaluated assuming a master-network transmission capacity b_0 of 1000 and 100 workers (N). Figure 7 shows the average normalized turnaround time T_{real}/T_{bound}, as a function of the application data size W. PTUMR provides excellent performance quite close to the lower bound for any W and any $\bar{\epsilon}$, that is, the PTUMR algorithm effectively eliminates the performance degradation associated with these factors. Under UMR, the normalized turnaround time becomes quite poor as the total data size W decreases, although good performance is achieved for large W. The degradation of performance for low values of W under UMR can be attributed to the increase of the overhead ratio which comes about as a result of decreasing the chunk size. This increase in the overhead ratio can be neutralized by PTUMR. These results therefore show that the PTUMR algorithm can effectively schedule applications of any size by minimizing the adverse effect of overheads on the application turnaround time.

5 Conclusion

We have proposed a novel scheduling algorithm called PTUMR for divisible workload-type applications based on a master-worker model in grid computing environments. The PTUMR allows the master to optimally transmit data to workers in parallel in a multi-round manner, which can considerably reduce application turnaround time compared with conventional multi-round scheduling algorithms such as UMR. The PTUMR presented in this paper is greatly extended from that in our previous study [11] in terms of adaptability to heterogeneous environments with respect to computation and communication resources.

This extension can be done by grouping workers appropriately for parallel data transmission, while taking heterogeneity in resources into account. Extensive performance evaluations show that the (extended) PTUMR can achieve an application turnaround time close to the theoretical lower limit under a variety of resource heterogeneity conditions.

Acknowledgments

This work was supported in part by the Ministry of Education, Culture, Sports, Science and Technology, Japan, under the Project National Research GRID Initiative (NAREGI) and in part by the Ministry of Internal Affairs and Communications, Japan.

References

1. I. Foster and C. Kesselman, *The GRID Blueprint for a New Computing Infrastructure,* Morgan Kaufmann Publishers, 1998.
2. I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid," *International Journal of Supercomputer Applications*, Vol. 15, No. 3, pp. 200–222, 2001.
3. V. Bharadwaj, D. Ghose, V. Mani, and T. G. Robertazzi, "Scheduling Divisible Loads in Parallel and Distributed Systems," *IEEE Computer Society Press,* 1996.
4. T. G. Robertazzi, "Ten Reasons to Use Divisible Load Theory," *Jounal of IEEE Computer,* Vol. 36, No. 5, pp. 63–68, May 2003.
5. D. Gerogiannis and S. C. Orphanoudakis, "Load Balancing Requirements in Parallel Implementations of Image Feature Extraction Tasks," *IEEE Trans. Parallel and Distributed Systems,* Vol. 4, No. 9, pp. 994–1013, 1993.
6. Y. Yang and H. Casanova, "UMR: A Multi-Round Algorithm for Scheduling Divisible Workloads," *Proc. of International Parallel and Distributed Processing Symposium (IPDPS'03),* Nice, France, April 2003.
7. Y. Yang and H. Casanova, "A Multi-Round Algorithm for Scheduling Divisible Workload Applications: Analysis and Experimental Evaluation," *Technical Report of Dept. of Computer Science and Engineering, University of California CS20020721,* 2002.
8. O. Beaumont, A. Legrand, and Y. Robert, "Optimal Algorithms for Scheduling Divisible Workloads on Heterogeneous Systems," *Proc. of International Parallel and Distributed Processing Symposium (IPDPS'03),* Nice, France, April 2003.
9. C. Cyril, O. Beaumont, A. Legrand, and Y. Robert, "Scheduling Strategies for Master-Slave Tasking on Heterogeneous Processor Grids," *Technical Report 2002-12,* LIP, March 2002.
10. A. L. Rosenberg, "Sharing Partitionable Workloads in Heterogeneous NOWs: Greedier Is Not Better," *Proc. of the 3rd IEEE International Conference on Cluster Computing (Cluster 2001),* pp. 124–131, California, USA, October 2001.
11. H. Yamamoto, M. Tsuru, and Y. Oie, "Parallel Transferable Uniform Multi-Round Algorithm for Achieving Minimum Application Turnaround Times for Divisible Workload," *Proc. of the 2005 International Conference on High Performance Computing and Communication (HPCC-05), LNCS 3726,* pp. 817–828, Capri-Sorrento Penisular, Italy, September 2005.