PAPER

Parallel Transferable Uniform Multi-Round Algorithm for Minimizing Makespan

Hiroshi YAMAMOTO[†a)], **Masato TSURU**[††], ***Members***, **Katsuyuki YAMAZAKI**[†], ***and*** **Yuji OIE**[††], ***Fellows***

SUMMARY In parallel computing systems using the master/worker model for distributed grid computing, as the size of handling data grows, the increase in the data transmission time degrades the performance. For divisible workload applications, therefore, multiple-round scheduling algorithms have been being developed to mitigate the adverse effect of longer data transmission time by dividing the data into chunks to be sent out in multiple rounds, thus overlapping the times required for computation and transmission. However, a standard multiple-round scheduling algorithm, Uniform Multi-Round (UMR), adopts a sequential transmission model where the master communicates with one worker at a time, thus the transmission capacity of the link attached to the master cannot be fully utilized due to the limits of worker-side capacity. In the present study, a Parallel Transferable Uniform Multi-Round algorithm (PTUMR) is proposed. It efficiently utilizes the data transmission capacity of network links by allowing chunks to be transmitted in parallel to workers. This algorithm divides workers into groups in a way that fully uses the link bandwidth of the master under some constraints and considers each group of workers as one virtual worker. In particular, introducing a Grouping Threshold effectively deals with very heterogeneous workers in both data transmission and computation capacities. Then, the master schedules sequential data transmissions to the virtual workers in an optimal way like in UMR. The performance evaluations show that the proposed algorithm achieves significantly shorter turnaround times (i.e., makespan) compared with UMR regardless of heterogeneity of workers, which are close to the theoretical lower limits.

key words: *grid computing, Master/Worker Model, divisible workload, Multi-Round scheduling, UMR*

1. Introduction

Grid computing has recently become increasingly common distributed applications [1], [2]. The master/worker model is suited to grid computing environments that includes 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 [3]–[8], where the master divides the application data into an arbitrary number of chunks and dispatches them to multiple workers as shown in Fig. 1. For a given application, computation and transmission times for a chunk are assumed to be roughly proportional to the size of the chunk.

The existing Uniform Multi-Round algorithm (UMR) can handle a large amount of data related to an application in a 'multiple-round' manner, where the data are divided into arbitrarily sized chunks and processed in multiple rounds in order to overlap the time required for communication with that required for computation [6], [9]–[11]. However, it utilizes a sequential transmission model, i.e. the master transmits data to one worker at a time [12], [13]. In actual networks, while the master that plays a central role in computing may have higher data transmission capacity than workers, the actual data transmission capacity between the master and each worker is limited to the worker-side capacity. Therefore, UMR cannot minimize the adverse effects of data transmission on the application turnaround time needed for processing the whole application data.

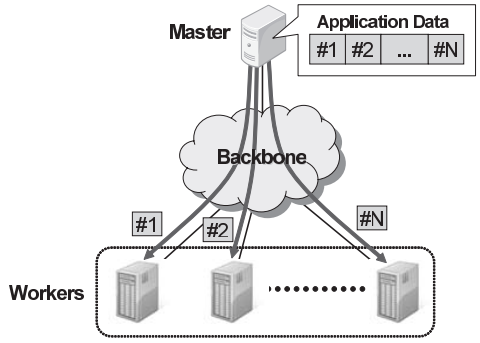


Fig. 1 A simple master-worker distributed computing model.

In this study, by integrating our previous works [14], [15], we propose a new scheduling algorithm, Parallel Transferable Uniform Multi-Round (PTUMR), that efficiently utilizes the data transmission capacity of network links by allowing chunks of application data to be transmitted in parallel to workers through simultaneously established multiple connections of data transmission flows. By effective grouping of workers, the proposed algorithm is adapted to the actual environments with large heterogeneity in data transmission and computation capacities of workers. This work assumes the use of Transmission Control Protocol (TCP) for data transmission flow.

We first introduce how the PTUMR derives parameters that determine how the master divides the application data and when it sends the data to the workers in order to minimize the application turnaround time. The master divides workers into appropriate groups on the basis of both computation and communication capacities of individual workers, and handles the set of workers in each group as a single vir-

Manuscript received November 16, 2011.

[†]The authors are with the Department of Electrical Engineering, Nagaoka University of Technology, Nagaoka-shi, 940-2188 Japan.

[††]The authors are with Graduate School of Computer Science and Systems Engineering, Kyushu Institute of Technology, Iizuka-shi, 820-8502 Japan.

a) E-mail: hiroyama@nagaokaut.ac.jp

DOI: 10.1587/transcom.E95.B.1669

tual worker. The master then optimally transmits chunks to the virtual workers sequentially as in UMR. Secondly, we evaluate the performance of the 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, achieving turnaround times close to the theoretical lower limit.

This paper is organized as follows. Section 2 introduces the conceptual basis for multiple-round scheduling and the conventional UMR algorithm. Section 3 presents the proposed PTUMR algorithm, and Sect. 4 evaluates its performance in a homogeneous environment. Section 5 validates the efficiency of the PTUMR in a realistic environment. Section 6 provides the conclusion.

2. Related Work

As mentioned in Sect. 1, 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 [9]–[13]. Figure 2 shows a timing chart of a simple example of this scenario where the master dispatches a workload of the application to a worker. In this figure, black rectangles represent the fixed-length overhead for one round of computation, and gray squares represent the fixed-length overhead in one round of data transmission. In multiple-round scheduling, the entire application data set is divided into multiple arbitrarily sized chunks and processed in M rounds ($M = 3$ in this example) to reduce the adverse effects of longer data transmission time on the application turnaround time. However, using a large number of rounds increases the total overhead. Thus, optimizing the number of rounds to minimize the application turnaround time is key to multiple-round scheduling.

UMR is an example of a multiple-round scheduling algorithm on the simple distributed computing model shown in Fig. 1 [9], [10]. The master and N workers are connected to a high-speed network that is free of bottlenecks. The total amount of data to be distributed from the master to workers is denoted by W [units], and the data transmission capacity of the link attached to the master is denoted by b_0 [units/s]. This model is heterogeneous in terms of the computation and communication capacities of workers: each worker i is associated with its computation speed s_i [units/s] and data transmission capacity b_i [units/s] of the link attached to the worker, and overheads δ_i [s] and ϵ_i [s] are added to the computation time and data transmission time, respectively.

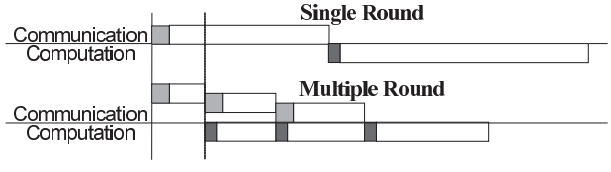


Fig. 2 Timing chart of single-round and multi-round scheduling algorithms.

UMR adopts the sequential transmission model where 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 must be bounded above by $\min\{b_i, b_0\}$. Figure 3 illustrates how the data are transmitted to workers and processed under UMR, where the chunk size 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 subsequent rounds, the size of chunks then grows exponentially.

Many types of extensions of UMR have been proposed already. The algorithms proposed by Jia et al. [16] and Loc and Elnaffar [17] examine and predict the resource capabilities of workers, adapting to the dynamic environments where the availability of both computation and communication resources vary with time. An optimization technique that quickly derives the optimal number of rounds and optimal chuck size to minimize the application turnaround time has been presented by Drozdowski and Lawenda [18]. A new scheduling algorithm described by Lin et al. [19] determines the minimum set of workers that finishes processing the workload by the given deadline.

However, these existing algorithms have utilized the sequential transmission model as well as UMR. In actual networks where the master may have higher communication capacity than workers, these algorithms restrict the actual communication capacity between the master and each worker depending on the worker-side capacity.

In this study, we, therefore, propose a novel scheduling algorithm, Parallel Transferable Uniform Multi-Round (PTUMR), which adopts a new data transmission model in order to minimize the application turnaround time.

The challenge in this study includes how to partition all the workers into subgroups (as a virtual worker), especially when the workers are very heterogeneous in both data transmission and computation capacities. In a general sense, suitably partitioning a set of heterogeneous members into subgroups is often seen as a resource allocation problem to be solved in order to achieve a good performance for the en-

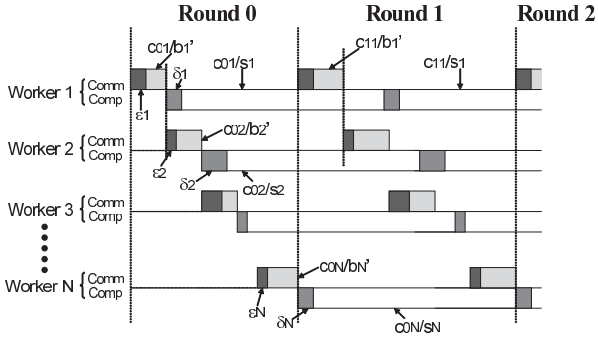


Fig. 3 Timing chart of data transmission and worker processing under UMR.

tire system. For example, problems of how to partition a set of receivers with heterogeneous data transmission capacities into multiple multicast groups have been investigated to maximize receiving throughput averaged over all the receivers (e.g., [20]–[22]), where the sender distributes a single datum using multiple multicast trees, and the receivers in each particular group receive the multicast data at each identical flow rate. On the other hand, in addition to the difference between unicast and multicast, our study investigates a more complex scenario of partitioning the members (the workers) with heterogeneity in both data transmission and computation capacities in cooperation with an optimal scheduling for data transmission and computation.

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 enables the master to transmit data to multiple workers in parallel, assuming that the overhead ϵ_i can be overlapped among concurrent transmissions as shown in Fig. 4. More precisely, the PTUMR divides workers to groups in a way to make full use of link bandwidth of the master, and treats a set of workers in each group as a single virtual worker. Then the master transmits chunks to virtual workers sequentially, as in UMR.

After 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.

Main parameters of PTUMR are summarized in Table 1.

3.1 Computation and Data Transmission Capacities of a Virtual Worker

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

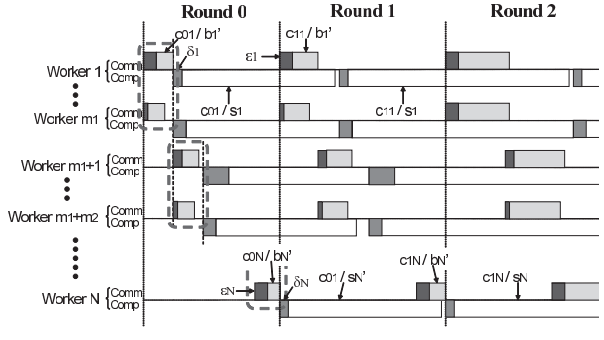


Fig. 4 Timing chart of data transmission and worker processing under PTUMR.

In order to fully utilize the computation power of each worker in a 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 [6] (s_i [units/s] denotes the computation speed of worker i). 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

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

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

Next, to make a multi-round scheduling efficient, we adjust the data transmission time of worker i in a round to be identical for all worker in L_k by limiting the data transmission rate b'_i of each worker i. Since the chunk size (i.e., the size of transmitted data) to worker i is set in proportional to s_i, ratio $\rho = \frac{s_i}{b'_i}$ for worker i should be the same among

Table 1 Main parameters of PTUMR.

s_i [units/s]	computation capacity of worker i
b_i [units/s]	communication capacity of link attached to worker i
b_0 [units/s]	communication capacity of link attached to master
b'_i [units/s]	actual communication capacity between master and worker i
δ_i [s]	overhead of computation time of worker i
ϵ_i [s]	overhead of communication time of worker i
S_k [units/s]	computation capacity of virtual worker k
B_k [units/s]	communication capacity between master and virtual worker k
Δ_k [s]	overhead of computation time of virtual worker k
E_k [s]	overhead of communication time of virtual worker k
N	the number of workers
N_v	the number of virtual workers
m_k	the number of workers in virtual worker k
L_k	set of workers in virtual worker k
W [units]	total amount of application data
w_j [units]	size of chunk allocated to workers in Round j
c_{ji} [units]	size of chunk allocated to worker i in Round j
C_{jk} [units]	size of chunk allocated to virtual worker k in Round j
M	the number of rounds

all workers in L_k. Therefore, in order to maximize the actual data transmission rate between the master and virtual worker k, data transmission rate b'_i of worker i is determined by minimizing ratio ρ with the link capacity constraints as follows.

$$\begin{aligned} &\text{minimize } \rho, \\ &\text{subject to } \begin{cases} \forall i \in L_k & \frac{s_i}{\rho} \le b_i, \\ \frac{S_k}{\rho} \le b_0. \end{cases} \end{aligned} \tag{2}$$

Hence $\rho = \max\left\{\frac{S_k}{b_0}, \frac{s_i}{b_i} \middle| i \in L_k\right\}$. We define $B_k = \sum_{i \in L_k} b'_i = \frac{S_k}{\rho}$ as the data transmission rate of virtual worker k. 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

$$T_{comm_{jk}} = \frac{C_{jk}}{\sum_{i \in L_k} b'_i} + \max_{i \in L_k}\{\epsilon_i\} = \frac{C_{jk}}{B_k} + E_k. \tag{3}$$

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 three 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 to determine the number m_k of workers in the virtual worker k.

$$\begin{aligned} &m_k = \max\left\{m \middle| \sum_{l=l_k}^{l_k+m-1} b_l \le b_0\right\} + \lambda, \\ &l_{k+1} = l_k + m_k. \end{aligned} \tag{4}$$

   where l_k indicates the serial number of the first worker composing the virtual worker k and λ indicates the number of additional workers added to the group after the number of workers has been chosen in a way to make full use of the master-network bandwidth. Note that in general it is difficult to derive an optimal value of λ analytically. Only in the homogeneous environment where all workers have the same computation speed and communication capacity, we can find the optimal λ by evaluating the relationship between λ and the turnaround time [14].
2. In highly heterogeneous environments where the workers are very different in their computation power and transmission capacity, it may be beneficial to group workers whose resource capacities are close to each other according to the following equation.

$$\begin{aligned} &m'_k = \max\left\{m \middle| r_{l_k+m-1} \le \frac{\mu}{m-1} \sum_{l=l_k}^{l_k+m-2} r_l\right\}, \\ &m_k = \min\left\{m_k \text{ in (4)}, \ m'_k\right\}. \end{aligned} \tag{5}$$

   If r_i of the next worker is μ times larger than the average r_i of all workers already selected for the group, this next worker will not be included. We adopt μ of 1.5 in this paper.
3. 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.

$$\begin{aligned} &N_v = \max\left\{n \middle| \sum_{k=1}^{n} R_k < 1\right\} \quad if \quad R_1 < 1, \\ &N_v = 1 \quad otherwise. \end{aligned} \tag{6}$$

Step 1 presents a basic grouping method which attempts to preferentially select workers with smaller r_i, as happens in UMR [10] to fully utilize the bandwidth b_0 of the master-network link. In other words, the grouping method gives priority to higher transmission capacity rather than to faster computation speed to reduce the data transmission time for the first worker on the first round and to keep all computation capacities fully active during the data transmission even when a lot of workers are selected. The data transmission time for the first worker on Round 0 should be minimized because it cannot be overlapped with the computation time as shown in Figs. 3 and 4. Furthermore, the additional number λ 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 higher r_i than others, which leads to critical degradation of the data transmission rate B_k which is derived by solving Eq. (2). When heterogeneity is high, the basic grouping in Step 1 does not always result in good performance (shown later in Sect. 5.1). Therefore, Step 2 is introduced for effective grouping where workers in a group are restricted to have similar resource capacities. The version solely with Step 1 is called PTUMR without Grouping Threshold (GT), while that with Step 1 and Step 2 is called PTUMR with GT. Hereafter, PTUMR denotes PTUMR with GT unless otherwise noted. Step 3 then preferentially selects virtual workers with smaller 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

After grouping the workers into the virtual workers as described in Sect. 3.2, the 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). T_{real} under PTUMR can be obtained as presented in Appendix A. 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 $w_j (= \sum_{k=1}^{N_v} C_{jk})$ by the total amount of chunk to be allocated to virtual workers in Round j. From Eq. (1), the relation between w_j and the size C_{jk} of chunk allocated to the virtual worker k is given by

$$C_{jk} = \alpha_k \times w_j + \beta_k, \quad \left(\alpha_k = \frac{S_k}{\sum_{k=1}^{N_v} S_k}, \quad \beta_k = \frac{S_k \times \sum_{k=1}^{N_v} \{S_k \times \Delta_k\}}{\sum_{k=1}^{N_v} S_k} - S_k \times \Delta_k. \right) \tag{7}$$

In addition, from Eqs. (1) and (3), 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.

$$w_j = \theta^j (w_0 - \gamma) + \gamma, \quad \left(\theta = \frac{1/\sum_{k=1}^{N_v} S_k}{\sum_{k=1}^{N_v} \{\alpha_k / B_k\}}, \quad \gamma = \frac{\frac{\sum_{k=1}^{N_v} \{S_k \times \Delta_k\}}{\sum_{k=1}^{N_v} S_k} + \sum_{k=1}^{N_v} \left\{ E_k - \frac{\beta_k}{B_k} \right\}}{\sum_{k=1}^{N_v} \{\alpha_k / B_k\} - 1/\sum_{k=1}^{N_v} S_k}. \right) \tag{8}$$

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

$$T_{ideal} = \frac{1}{\sum_{k=1}^{N_v} S_k} \left\{ W + M \times \sum_{k=1}^{N_v} (S_k \times \Delta_k) + \sum_{k=1}^{N_v} \left[S_k \times \sum_{t=1}^{k} \left(\frac{\alpha_t \times \left(\frac{1-\theta}{1-\theta^M} \times (W - M\gamma) + \gamma \right) + \beta_t}{B_t} + E_t \right) \right] \right\}. \tag{9}$$

The derivation of the application turnaround time T_{ideal} is presented in Appendix B.

Let M^* denote the real value minimizing T_{ideal} in Eq. (9), 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}. 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 so as to minimize T_{real}.

4. Performance Evaluation in Homogeneous Environment

In this section, we show the performance of our proposed scheduling algorithm, PTUMR, in the homogeneous environment where the set of workers is homogeneous in terms of the computational power ($s_1 = \cdots = s_N = s$), the worker-side transmission capacity ($b_1 = \cdots = b_N = b$) and overheads ($\delta_1 = \cdots = \delta_N = \delta$, $\epsilon_1 = \cdots = \epsilon_N = \epsilon$) added to the computation and data transmission time, respectively.

Note that in the homogeneous environment, PTUMR with GT and that without GT are identical.

Here, every virtual worker consists of m workers, so that, the computation speed ($S_1 = \cdots = S_{N_v} = m \times s$) and the transmission capacity ($B_1 = \cdots = B_{N_v} = \min\{m \times b, b_0\}$) are identical for all virtual workers. However, when the number N of all workers is not divisible by the number m of parallel data transmissions, the last virtual worker includes $N \mod m$ workers, and has the computation speed of ($S_{N_v} = (N \mod m) \times s$) and the transmission speed of ($B_{N_v} = \min\{(N \mod m) \times b, b_0\}$).

T_{bound} denotes 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. (9), T_{bound} is obtained as follows.

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

4.1 Impact of Parallelism on Application Turnaround Time

As mentioned in Sect. 3.2, in the homogeneous environment where the parameters s, b, δ, and ϵ are identical for all workers, we can find the optimal number m^* of workers composing each virtual worker achieving the application turnaround time T_{real} close to the minimum value by evaluating the relationship between m and T_{real}. To find the optimal m^*, the impact of the number of workers composing each virtual worker m on T_{real} is examined in Fig. 5. Here, the parameters $W = 1000$ [units/s], $N = 100$, $b_0 = 120, 600$ [units/s], and $\epsilon = 0.1, 0.01, 0.001$ [s] are used. All other parameters are set according to Table 2.

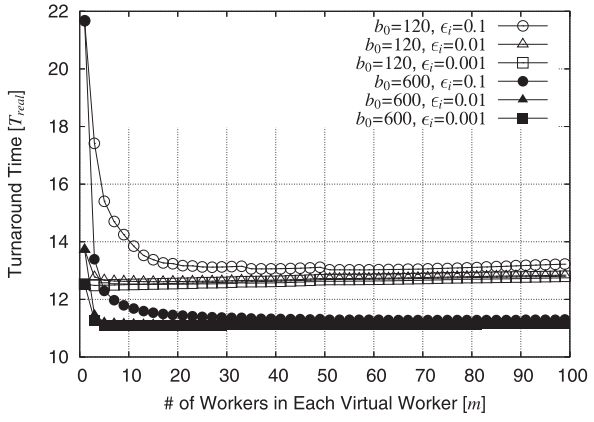


Fig. 5 Impact of m on application turnaround time.

Table 2 Model parameters and their values examined in homogeneous environment.

s	1 [units/s]
b	120 [units/s]
δ	0.5 [s]

The case of $b_0 = 120$ represents a symmetric network, in which the transmission capacity b_0 of the master-side link is equal to that b of the worker-side, which is the case considered in UMR. Figure 5 surprisingly indicates, however, that even in a symmetric network, the transmission to multiple workers in parallel yields better performance than UMR ($m = 1$). For a wide range of ϵ, increasing m can reduce the adverse effects of overhead on the application turnaround time by overlapping ϵ for multiple workers. On the other hand, increasing m also reduces the actual transmission speeds b' for each worker due to sharing of the transmission capacity of the master-side link by multiple data transmissions as shown in Eq. (2), which increases the data transmission time for each worker. Therefore, there exists an optimal number m^* of parallel transmissions that minimizes the application turnaround time T_{real}.

The case of $b_0 = 600$ represents an asymmetric network in which the master-side link has a larger capacity than the worker-side links, as often occurs in real network environments. In such an asymmetric network, as also indicated in Fig. 5, increasing m dramatically reduces the application turnaround time T_{real} regardless of ϵ. This occurs because an increase in m (up to $m = 5$) does not reduce the transmission speed for each worker.

In both cases, symmetric or asymmetric networks, the application turnaround time T_{real} rapidly decreases as m increases above 1, but then becomes constant or increases slightly with increasing m after reaching a nearly-minimum value of T_{real}.

4.2 Impact of the Number of Workers on Application Turnaround Time

The impact of the number N of workers on the application turnaround time T_{real} is examined in Fig. 6 by assuming a total size W of 1000. As N increases, the application turnaround times under both the PTUMR and UMR algorithms remarkably decrease, but particularly PTUMR achieves T_{real} close to the lower bound T_{bound} for any N by increasing the parallelism m to utilize the network resources at the master-side to their maximum, while the difference between T_{real} of UMR and T_{bound} increases with N.

Figure 6 shows that the application turnaround time of PTUMR is almost the same as that of UMR which transmits chunks to one worker at a time when the number N of workers is small. However, in order to achieve the optimal performance, we should select the appropriate parallelism as described in Sect. 3.2. Figures 7(a) and 7(b) show the lowest m and the highest m achieving T_{real} within 101% of the minimum one as a function of N, respectively. In these Figs, the lowest m for good performance increases very gradually as N increases, while any parallelism achieves nearly-optimal performance during the small N. In addition, when the master-side transmission capacity b_0 is low, the highest m for good performance decreases with the increase in N. Therefore, when N is large, m should be selected from an appropriate range of parallelism to achieve the good performance. However, in general, the relatively small m can achieves nearly optimal turnaround time for all N.

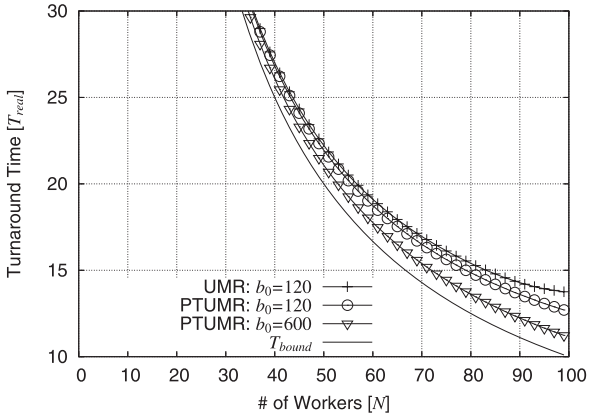
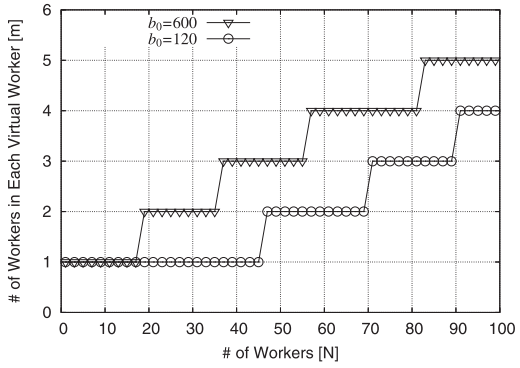
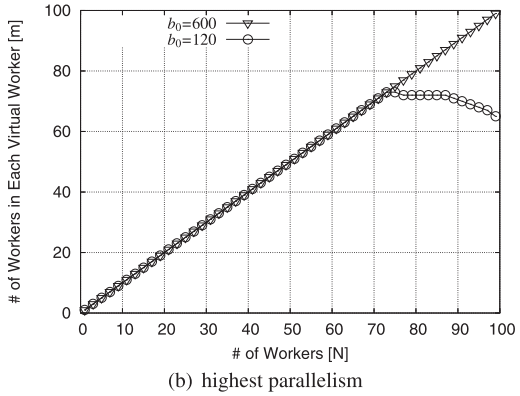


Fig. 6 Impact of number of workers on application turnaround time.



(a) lowest parallelism



(b) highest parallelism

Fig. 7 Impact of number of workers on a range of parallelism which achieves application turnaround time within 101% of minimum one.

Furthermore, this implies that it is possible to determine a nearly-optimal parallelism m^* through the use of binary search-like methods which repeatedly divide the search range of m in half without examining T_{real} for all m, and that PTUMR can utilize a wide range of nearly-optimal parallelism m^* due to the insensitivity of T_{real} to large m, allowing other performance-related factors to be considered. For example, in order to reduce a consumption of worker-side transmission capacity, the master can increase the par-

allelism m.

5. Performance Evaluation in Heterogeneous Environment

In this section, we evaluate the performance of PTUMR in the heterogeneous environment where the workers are different in their computational and communication capacity. In this evaluation, we assume, as was the case in the study proposing UMR [10], 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.

$$\left((1 - \sqrt{3} \times het) \times mean, (1 + \sqrt{3} \times het) \times mean\right). \quad (11)$$

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 Table 3.

The effectiveness of PTUMR is evaluated by comparing the achievable turnaround time T_{real} with the lower bound T_{bound} defined in Eq. (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.

5.1 The Impact of Heterogeneous Resource Capacities

First, we investigated the effect of the heterogeneity *het* on the performance of the 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. (11).

Figure 8 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 λ of additional workers under PTUMR with and without GT was set to 10. As shown in Fig. 8, regardless of *het*, PTUMR without GT is superior to conventional UMR in terms of average normalized turnaround time. However, when the heterogeneity is high, the application turnaround time of PTUMR without GT 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 λ of addition workers of 10. In addition we will investigate the performance of the scheduling algorithms in highly heterogeneous environment, namely $het = \frac{\sqrt{3}}{4}$. Furthermore, the number N of workers is set to 100.

Table 3 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]

5.2 The Impact of Network Resources

The impact of network resources is examined here by assuming a total workload W of 1000. Figure 9 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

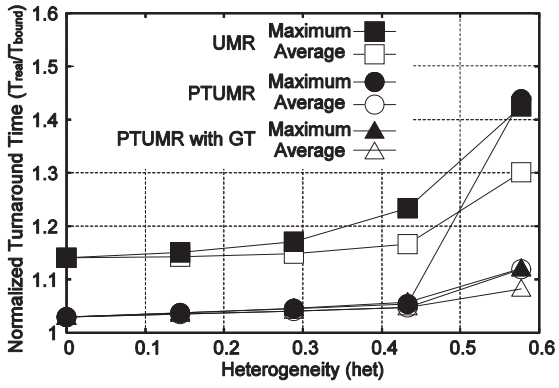


Fig. 8 Impact of heterogeneity on the normalized application turnaround time.

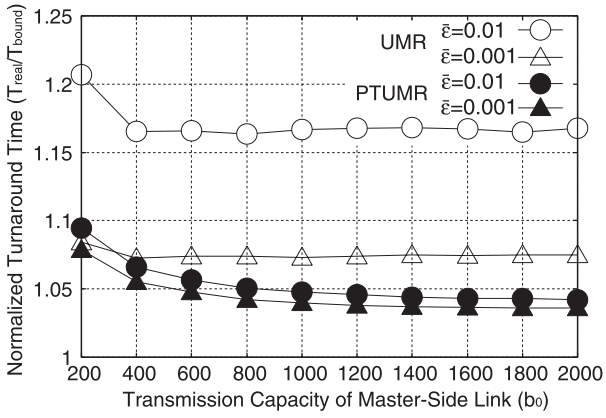


Fig. 9 Impact of network resources on the normalized application turnaround time with different communication setup overhead.

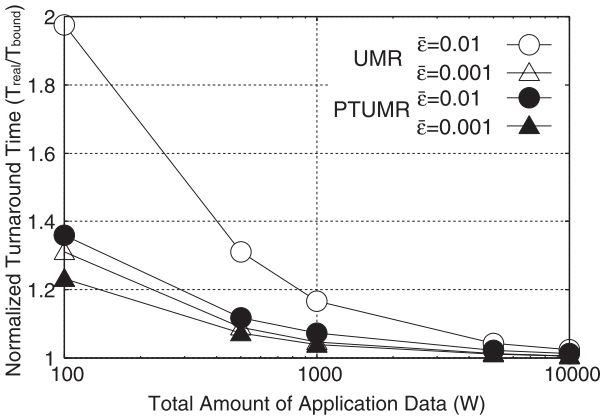


Fig. 10 Impact of total workload on the normalized application turnaround time with different communication setup overhead.

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 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.

5.3 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. Figure 10 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.

6. Conclusion Remarks

As the size of handling data grows, the increase in the data transmission time degrades the performance. The adverse effects of data transmission time on the application turnaround time can be mitigated to a certain extent by using a multiple-round scheduling algorithm such as UMR to overlap the data transmission time with the computation time. However, as UMR adopts the sequential transmission model, it cannot minimize the application turnaround time effectively, especially in asymmetric networks where the master-side link capacity is greater than that of the worker-side.

This study introduces, PTUMR, a new multiple-round scheduling algorithm that adopts a multiple transmission model. In contrast to UMR, PTUMR enables application data to be transmitted to multiple workers in parallel. The proposed algorithm divides workers into groups to fully utilize communication capacity of master-side and computation capacities of individual workers by considering heterogeneity in both computation and communication capacities of individual workers. After that, the algorithm determines the appropriate number of rounds so that the application turnaround time is minimized. The performance of PTUMR has been evaluated in various environments, and the PTUMR algorithm has been found to mitigate the adverse effects of data transmission time significantly, achieving turnaround times close to the lower bound over a wide range of application data sizes and network conditions.

Note that PTUMR has two tuning parameters λ and μ. For λ, in our experimental evaluation, numerically derived semi-optimal values as λ were used in the homogeneous cases (Sect. 4), while a heuristically selected constant value 10 as λ was used throughout the heterogeneous case (Sect. 5). For μ, it is needed only in heterogeneous cases and decided in a heuristic manner as well. A constant value of 1.5 as μ was used throughout the heterogeneous cases. Although our method with heuristic constant (fixed) parameters exhibited considerably good performance in a variety of conditions of the experiments, we will investigate more on selecting those parameters in future work.

Acknowledgments

This study was supported in part by the Japan Society for the Promotion of Science, Grant-in-Aid for Scientific Research (B) (21300024).

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," Int. J. Supercomputer Applications, vol.15, no.3, pp.200–222, 2001.
[3] D.A.L. Piriyakumar and C.S.R. Murthy, "Distributed computation for a hypercube network of sensor-driven processors with communication delays including setup time," IEEE Trans. Syst. Man. Cybern. A, Syst. Humans, vol.28, no.2, pp.245–251, March 1998.
[4] J.T. Hung, H.J. Kim, and T.G. Robertazzi, "Scalable scheduling in parallel processors," Proc. 2002 Conference on Information Sciences and Systems, Princeton University, March 2002.
[5] J.T. Hung and T.G. Robertazzi, "Scalable scheduling for clusters and grids using cut through switching," Int. J. Comput. Appl., vol.26, no.3, pp.147–156, 2004.
[6] V. Bharadwaj, D. Ghose, V. Mani, and T.G. Robertazzi, Scheduling Divisible Loads in Parallel and Distributed Systems, IEEE Computer Society Press, 1996.

[7] T.G. Robertazzi, "Ten reasons to use divisible load theory," J. Computer, vol.36, no.5, pp.63–68, May 2003.
[8] D. Gerogiannis and S.C. Orphanoudakis, "Load balancing requirements in parallel implementations of image feature extraction tasks," IEEE Trans. Parallel Distrib. Syst., vol.4, no.9, pp.994–1013, 1993.
[9] Y. Yang and H. Casanova, "UMR: A multi-round algorithm for scheduling divisible workloads," Proc. International Parallel and Distributed Processing Symposium (IPDPS'03), Nice, France, April 2003.
[10] 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.
[11] O. Beaumont, A. Legrand, and Y. Robert, "Optimal algorithms for scheduling divisible workloads on heterogeneous systems," Proc. International Parallel and Distributed Processing Symposium (IPDPS'03), Nice, France, April 2003.
[12] 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.
[13] A.L. Rosenberg, "Sharing partitionable workloads in heterogeneous NOWs: Greedier is not better," Proc. 3rd IEEE International Conference on Cluster Computing (Cluster 2001), pp.124–131, California, USA, Oct. 2001.
[14] H. Yamamoto, M. Tsuru, and Y. Oie, "Parallel transferable uniform multi-round algorithm for achieving minimum application turnaround times for divisible workload," Proc. 2005 International Conference on High Performance Computing and Communications (HPCC-05), LNCS 3726, pp.817–828, Capri-Sorrento Penisular, Italy, Sept. 2005.
[15] H. Yamamoto, M. Tsuru, and Y. Oie, "A parallel transferable uniform multi-round algorithm in heterogeneous distributed computing environment," Proc. 2006 International Conference on High Performance Computing and Communications (HPCC-06), LNCS 4208, pp.51–60, Munich, Germany, Sept. 2006.
[16] J. Jia, B. Veeravalli, and D. Ghose, "Adaptive load distribution strategies for divisible load processing on resource unaware multilevel tree networks," IEEE Trans. Comput., vol.56, no.7, pp.999–1005, 2007.
[17] N.T. Loc and S. Elnaffar, "A dynamic scheduling algorithm for divisible loads in grid environments," J. Commun., vol.2, no.4, pp.57–64, 2007.
[18] M. Drozdowski and M. Lawenda, "Multi-installment divisible load processing in heterogeneous distributed systems," Concurrency and Computation: Practice & Experience, vol.19, no.17, pp.2237–2253, 2007.
[19] X. Lin, Y. Lu, J. Deogun, and S. Goddard, "Multi-round real-time divisible load scheduling for clusters," Proc. 15th International Conference on High Performance Computing, LNCS 5374 Bangalore, India, Dec. 2008.
[20] S. Bhattacharyya, J.F. Kurose, D. Towsley, and R. Nagarajan, "Efficient rate-controlled bulk data transfer using multiple multicast groups," IEEE/ACM Trans. Netw., vol.11, no.6, pp.895–907, 2003.
[21] R.-H. Gau, Z.J. Haas, and B. Krishnamachari, "On multicast flow control for heterogeneous receivers," IEEE/ACM Trans. Netw., vol.10, no.1, pp.86–101, 2002.
[22] Z. Fei, M. Yang, M.H. Ammar, and E.W. Zegura, "A framework for allocating clients to rate-constrained multicast servers," J. Comput. Commun., vol.26, no.12, pp.1255–1262, 2003.

Appendix A: Derivation of Application Turnaround Time T_{real}

Derivation process of the application turnaround time T_{real} under PTUMR is illustrated in Fig. A·1. In this figure,

1. $comm_{-1,N_v} = 0$;
2. **FOR** $(k = 1;\ k \le N_v;\ k++)$
3. $\quad comp_{-1,k} = 0$;
4. }
5.
6. **FOR** $(j = 0;\ j < M-1;\ j++)$ {
7. $\quad comm_{j,1} = comm_{j-1,N_v} + \left(\frac{C_{j,1}}{B_1} + E_1\right)$;
8. $\quad$ **FOR** $(k = 2; k \le N_v; k++)$ {
9. $\quad\quad comm_{j,k} = comm_{j,k-1} + \left(\frac{C_{j,k}}{B_k} + E_k\right)$;
10. $\quad$ }
11. $\quad$ **FOR** $(k = 1;\ k \le N_v;\ k++)$ {
12. $\quad\quad comp_{j,k} = \max\{comp_{j-1,k}, comm_{j,k}\} + \left(\frac{C_{j,k}}{S_k} + \Delta_k\right)$;
13. $\quad$ }
14. }
15.
16. $comm_{M-1,1} = comm_{M-2,N_v} + \left(\frac{C_{last,1}}{B_1} + E_1\right)$;
17. **FOR** $(k = 2;\ k \le N_v;\ k++)$ {
18. $\quad comm_{M-1,k} = comm_{M-1,k-1} + \left(\frac{C_{last,k}}{B_k} + E_k\right)$;
19. }
20. **FOR** $(k = 1;\ k \le N_v;\ k++)$ {
21. $\quad comp_{M-1,k} = \max\{comp_{M-2,k}, comm_{M-1,k}\} + \left(\frac{C_{last,k}}{S_k} + \Delta_k\right)$;
22. }
23.
24. $T_{real} = \max_k\{comp_{M-1,k}\}$;

Fig. A·1 Derivation process of application turnaround time T_{real}.

$comp_{j,k}$ and $comm_{j,k}$ are defined as the time when the virtual worker k completes processing chunks received in Round j, and when the master finishes transmitting the chunk to the virtual worker k in Round j, respectively.

The derivation process of T_{real} consists of three phases. In the first phase, we recursively derive the time at which each virtual worker completes processing chunks received in each round. This is shown in Fig. A·1, line 6–14. Except for the last round (Round M-1), the size $C_{j,k}$ of the chunk transmitted to the virtual worker k in Round j is derived according to Eqs. (7) and (8). In the second phase, the time when each virtual worker completes processing the last chunks is given on line 16–22. Note that the size of the last chunk, which is denoted by $C_{last,k}$, for the virtual worker k, should be chosen in a way that every virtual worker completes the processing of its last chunk with one accord, thereby preventing any virtual worker from entering the idle state before the entire workload has been finally processed. We call it the last-chunk alignment. The need for the last-chunk alignment arises from the difference in time at which each virtual worker began to process its initial chunk. Finally, T_{real} can be obtained on line 24.

Appendix B: Derivation of Application Turnaround Time T_{ideal} under Ideal Assumption

First we derive T'_{ideal}, which is the ideal application turnaround time in case without the last-chunk alignment, as follows.

$$T'_{ideal} = \sum_{k=1}^{N_v}\left(\frac{C_{0,k}}{B_k} + E_k\right) + \sum_{j=0}^{M-1}\left(\frac{C_{j,N_v}}{S_{N_v}} + \Delta_{N_v}\right). \quad \text{(A·1)}$$

which indicates the time at which the virtual worker N_v completes processing the chunk received in the last round.

Next we consider the last-chunk alignment to reduce the application turnaround time. In order to have all virtual workers to complete processing the last round chunk with one accord, the master modifies the size of chunks allocated to virtual workers in the last round. Here, the difference D_k in the time required for the virtual worker k to perform computation of the last chunk and that for the virtual worker N_v is expressed as follows.

$$D_k = \sum_{K=k+1}^{N_v} d_k = \left(\frac{C_{last,k}}{S_k} + \Delta_k\right) - \left(\frac{C_{last,N_v}}{S_{N_v}} + \Delta_{N_v}\right). \quad (\text{A}\cdot 2)$$

where d_k indicates the difference in time at which the virtual worker $k-1$ starts computing the last chunk and the time at which the virtual worker k starts computing the last chunk. In our method, it is assumed that the amount of all chunks received by all virtual workers in the last round in case with the last-chunk alignment is equal to that in case without the last-chunk alignment. For this assumption, from Eq. (A· 2), the relation between the total size w_{M-1} of the last chunks and the size C_{last,N_v} of the last chunk allocated to the virtual worker k can be formulated as follows.

$$C_{last,N_v} = \frac{S_{N_v}}{\sum_{k=1}^{N_v} S_k}\left\{w_{M-1} + \sum_{k=1}^{N_v} S_k\left(\Delta_k - \sum_{K=k+1}^{N_v} d_K\right)\right\} - S_{N_v}\times\Delta_{N_v}. \quad (\text{A}\cdot 3)$$

In PTUMR, the difference in the time at which the master starts allocating the initial chunk to the virtual worker $k-1$ and that to the virtual worker k becomes d_k, because the computation time in each round is assumed to be identical for all virtual workers. Therefore, d_k is given by

$$d_k = \frac{C_{0,k}}{B_k} + E_k. \quad (\text{A}\cdot 4)$$

Under the last-chunk alignment, the application turnaround time T_{ideal} is smaller than that T'_{ideal} in case without the alignment by the difference in the computation time of the chunk of C_{M-1,N_v} and that of C_{last,N_v}. Therefore, from Eqs. (A· 2), (A· 3) and (A· 4), T_{ideal} is formulated as follows.

$$\begin{aligned} T_{ideal} &= T'_{ideal} - \left\{\left(\frac{C_{M-1,N_v}}{S_{N_v}} + \Delta_{N_v}\right) - \left(\frac{C_{last,N_v}}{S_{N_v}} + \Delta_{N_v}\right)\right\}, \\ &= \frac{1}{\sum_{k=1}^{N_v} S_k}\left\{W + M\times\sum_{k=1}^{N_v}(S_k\times\Delta_k) \right. \\ &\left. + \sum_{k=1}^{N_v}\left[S_k\times\sum_{t=1}^{k}\left(\frac{\alpha_t\times\left(\frac{1-\theta}{1-\theta^M}\times(W-M\gamma)+\gamma\right)+\beta_t}{B_t} + E_t\right)\right]\right\}. \end{aligned} \quad (\text{A}\cdot 5)$$

Hiroshi Yamamoto received M.E. and D.E. degrees from Kyushu Institute of Technology, Iizuka, Japan in 2003 and 2006, respectively. From April 2006 to March 2010, he worked at FUJITSU LABORATORIES LTD., Kawasaki, Japan. Since April 2010, he has been an Assistant Professor in the Department of Electrical Engineering, Nagaoka University of Technology. His research interests include computer networks, distributed applications, and networked services. He is a member of the IEEE.

Masato Tsuru received B.E. and M.E. degrees from Kyoto University, Japan in 1983 and 1985, respectively, and then received his D.E. degree from Kyushu Institute of Technology, Japan in 2002. He worked at Oki Electric Industry Co., Ltd., Information Science Center, Nagasaki University, and Japan Telecom Information Service Co., Ltd. In 2003, he moved to the Department of Computer Science and Electronics, Kyushu Institute of Technology as an Associate Professor, and then has been a Professor in the same department since April 2006. His research interests include performance measurement, modeling, and management of computer communication networks. He is a member of the ACM, IPSJ, and JSSST.

Katsuyuki Yamazaki received B.E. and D.E. degrees from the University of Electro-communications and Kyushu Institute of Technology in '80 and '01, respectively. At KDD Co. Ltd., he had been engaged in R&D and international standardization of ISDN, S.S. No.7, ATM networks, L2 networks, IP networks, mobile and ubiquitous networks, etc., and was responsible for R&D strategy of KDDI R&D Labs. He is currently a Professor of Nagaoka University of Technology.

Yuji Oie received B.E., M.E. and D.E. degrees from Kyoto University, Kyoto, Japan in 1978, 1980 and 1987, respectively. From 1980 to 1983, he worked at Nippon Denso Company Ltd., Kariya. From 1983 to 1990, he was with the Department of Electrical Engineering, Sasebo College of Technology, Sasebo. From 1990 to 1995, he was an Associate Professor in the Department of Computer Science and Electronics, Faculty of Computer Science and Systems Engineering, Kyushu Institute of Technology, Iizuka. From 1995 to 1997, he was a Professor in the Information Technology Center, Nara Institute of Science and Technology. Since April 1997, he has been a Professor in the Department of Computer Science and Electronics, Faculty of Computer Science and Systems Engineering, Kyushu Institute of Technology. His research interests include performance evaluation of computer communication networks, high speed networks, and queueing systems. He is a fellow of the IPSJ and a member of the IEEE.