Effective Use of Computational Resources in Multicore Distributed Systems

Hidehiro Kanemitsu*, Masaki Hanada**, Takashige Hoshiai***, Hidenori Nakazato****
*Media Network Center, Waseda University, Japan
**Department of Information Systems, Tokyo University of Information Sciences, Japan
***Department of Computer and Information Science, Sojo University, Japan
****Global Information and Telecommunication Institute, Waseda University, Japan.
kanemih@ruri.waseda.jp, mhanada@rsch.tuis.ac.jp, hoshiai@cis.sojo-u.ac.jp, nakazato@waseda.jp

Abstract— **In the last decades, many kinds of task execution models such as grid and cloud computing have been developed. In such distributed systems, each task is processed by respective processor in multicored computers e.g., household PCs which we can easily harness in recent years. If there is one policy to automatically decide the "best" combination and the number of processors (and computers), we effectively utilize those computational resources, thereby large number of jobs can be executed in parallel. In this paper, we propose a method for mapping of execution units for such environments. The method adopts a remapping technology after processor-execution unit mapping[11] is finished. Experimental comparisons by a simulation show the advantages of the proposed method.**

Index Terms— **Task Clustering, Multicore, Distributed Systems, Parallel Computing**

I. Introduction

RECENTLY, task execution models are becoming diverse, e.g., grid computing[1], cloud computing. Such computational models often require parallel execution models such as parametric job execution, background data analysis for data mining, and searching for a target value in a key-value store[2]. Irrespective of dedicated computers or household computers, they should be effectively used when a parallel execution model is applied. Moreover, requirements for job execution tends to become diverse due to multiple demands from the job submitter. To handle with such diversity, multi-objective approaches for task scheduling and resource allocation model have been proposed and being under development[3], [4], [5]. Those approaches assume that requirements (e.g., QoS, deadline for the job execution, total costs, and so on) are decided by a job submitter, i.e., an user. Thus, those multi-objective approaches focus on how to satisfy every objective specified by an user. If multiple objective should be satisfied at the same time in a distributed environment, dynamic natures such as every computer's load and CPU status and so on must be taken into account. To do this, provisioning is needed for preparing calculate every problems provided by those objectives. For example, how to propagate each computer's resource information and how to aggregate "peers" to construct peer groups in the overlay network based on specific criteria in advance are indispensable[3], [6], [7].

Manuscript received June 25, 2013.

Hidehiro Kanemitsu is with Media Network Center, Waseda University, 1-104, Totsuka, Shinjuku, Tokyo, Japan (phone: +81-3-3204-8223; e-mail: kanemih@ruri.waseda.jp).
Masaki Hanada is with Department of Information Systems, Tokyo University of Information Sciences, 4-1, Onaridai, Wakaba, Chiba, Japan (phone:+81-43-236-1101;e-mail: mhanada@rsch.tuis.ac.jp).
Takashige Hoshiai is with Department of Computer and Information Science, Sojo University, 4-22-1, Ikeda, Nishi, Kumamoto, Japan (phone: +81-96-326-3111;e-mail: hoshiai@cis.sojo-u.ac.jp).
Hidenori Nakazato is with Global Information and Telecommunication Institute, Waseda University, Bldg. 29-7, 1-3-10 Nishi-Waseda Shinjuku-ku, Tokyo, Japan (phone: +81-3-3203-4151; e-mail: nakazato@waseda.jp) .

We proposed a method for effective use of computational resources each of which has a single processor in a heterogeneous distributed system [11]. The method automatically derives set of mapping between each processor and each assignment unit (i.e., the set of tasks in a " DAG (Directed Acyclic Graph) " program such as a workflow type job), by which the degree of contribution for each processor toward the response time minimization is maximized. However, the method does not assume more realistic situations, e.g., each computer may have a parallel execution scheme such as multicore and multi OS by virtualization. Thus, our previously proposed method can not be applied to those environments. If a model which accommodates them, the response time can be minimized with a smaller number of computers. In this paper, we propose a theoretical method for deciding the set of mapping between each computer and each assignment unit for effective use of computational resources, where each computer can have parallel execution scheme. The proposed method assumes that a computer with two or more processors as " two or more processors are completely connected over the network with infinite network bandwidth ". Then a lower bound of assignment unit size is mathematically decided for each processor for limiting the number of processors. Actual assignment units are generated by a task clustering algorithm until each size exceeds the lower bound. Among processors which belong to the same computer, every task in assignment units is locally scheduled without degrading task parallelism. Consequently, the required number of computer can be smaller than that of the method proposed in [11]. Experimental comparison is conducted by a simulation program. In the simulation, we evaluate the degree of contribution for each computer and show advantages of the proposed method over related works and the method we proposed in [11].

The remainder of this paper is organized as follows. Sec. II presents our assumed model. Then problem definitions and the objective of the proposal are presented in sec. III. The detailed

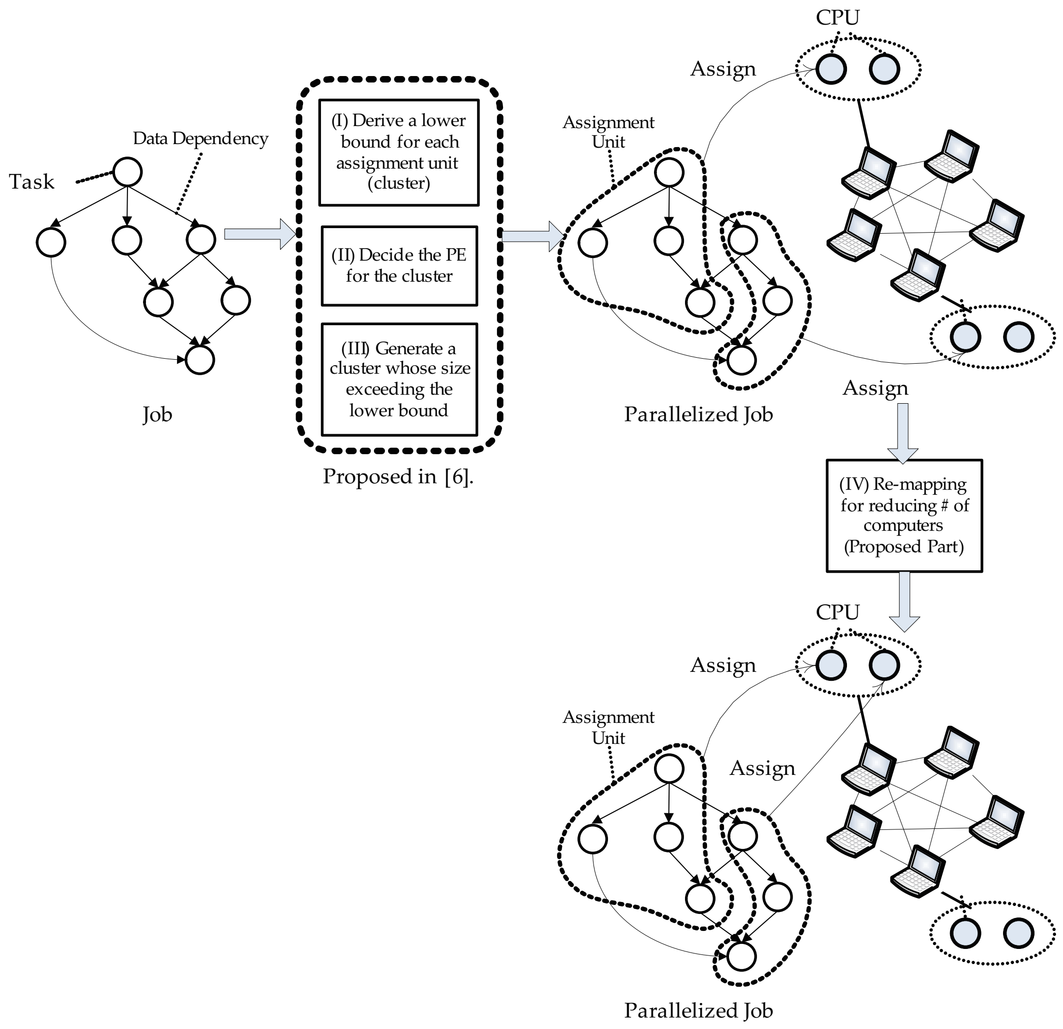

Task
Data Dependency
Job
(I) Derive a lower bound for each assignment unit (cluster)
(II) Decide the PE for the cluster
(III) Generate a cluster whose size exceeding the lower bound
Proposed in [6].
Assignment Unit
Parallelized Job
CPU
Assign
Assign
(IV) Re-mapping for reducing # of computers (Proposed Part)
CPU
Assign
Assignment Unit
Assign
Parallelized Job

Fig. 1. Proposed part

content of the proposal is presented in sec. IV, and theoretical analysis in terms of effect of the proposal is presented in sec. V. Advantages of the proposal is shown by the experimental comparisons in sec. VI. Finally, a conclusion and a future work are presented in sec. VII.

II. Assumed Model

A. Job Model

An assumed job is expressed as a Directed Acyclic Graph (DAG), which is known as one of task graph representations. Let $G^s_{cls} = (V_s, E_s, V^s_{cls})$ be the DAG, where s is the number of task merging steps (described in the latter half part in this subsection), V_s is the set of tasks after s task merging steps, E_s is the set of edges (data communications among tasks) after s task merging steps, and V^s_{cls} is the set of clusters which consists of one or more tasks after s task merging steps. An i-th task is denoted as n^s_i. Let $w(n^s_i)$ be a size of n^s_i, i.e., $w(n^s_i)$ is the sum of unit times taken for being processed by the reference processor element. We define data dependency and direction of data transfer from n^s_i to n^s_j as $e^s_{i,j}$. And $c(e^s_{i,j})$ is the sum of unit times taken for transferring data from n^s_i to n^s_j over the reference communication link.

One constraint imposed by a DAG is that a task can not be started execution until all data from its predecessor tasks arrive. For instance, $e^s_{i,j}$ means that n^s_j can not be started until data from n^s_i arrives at the processor which will execute n^s_j. And let $pred(n^s_i)$ be the set of immediate predecessors of n^s_i, and $suc(n^s_i)$ be the set of immediate successors of n^s_i. If $pred(n^s_i) = \emptyset$, n^s_i is called START task, and if $suc(n^s_i) = \emptyset$, n^s_i is called END task. If there are one or more paths from n^s_i to n^s_j, we denote such a relation as $n^s_i \prec n^s_j$.

B. Task Clustering

We denote the i-th cluster in V^s_{cls} as $cls_s(i)$. If n^s_k is included in $cls_s(i)$ by "the $s+1$ th task merging", we formulate one task merging as $cls_{s+1}(i) \leftarrow cls_s(i) \cup \{n^s_k\}$. If any two tasks, i.e., n^s_i and n^s_j, are included in the same cluster, they are assigned to the same processor. Then the communication between n^s_i and n^s_j is localized, so that we define $c(e^s_{i,j})$ becomes zero. Task clustering[8], [9], [10] is a set of task merging steps, that is finished when certain criteria have been satisfied.

C. System Model

We assume that each computer is completly connected to others, with not identical processing speeds and communication bandwidths. Each computer has one or more processing elements (PE). The set of computers is expressed as $M = \{M_1, M_2, \ldots, M_m\}$. The set of PEs in M_i is expressed as $P_i = \{P_{i,1}, P_{i,2}, \ldots, P_{i,|P_i|}\}$, and let the set of processing speeds in P_i as α_i, i.e., $\alpha_i = \{\alpha_{i,1}, \alpha_{i,2}, \ldots, \alpha_{i,|P_i|}\}$, where $|P_i| = |\alpha_i|$.

As for data communication among computers, let the set of communication bandwidths be β, i.e.,

$$\beta = \begin{pmatrix} \infty & \beta_{1,2} & \beta_{1,3} & \cdots & \beta_{1,m} \\ \beta_{2,1} & \infty & \beta_{2,3} & \cdots & \beta_{2,m} \\ \beta_{3,1} & \beta_{3,2} & \infty & \cdots & \beta_{3,m} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \beta_{m,1} & \beta_{m,2} & \beta_{m,3} & \cdots & \infty \end{pmatrix} \tag{1}$$

$\beta_{i,j}$ means the bandwidth from M_i to M_j. We assume that a communication bandwidth between two computers, e.g., M_i and M_j is the same between any one of P_i and any one of P_j. The processing time in the case that n_k^s is processed on $P_{i,p}$ is expressed as $t_p(n_k^s, \alpha_{i,p}) = w(n_k^s)/\alpha_{i,p}$. The data transfer time of $e_{k,l}^s$ over $\beta_{i,j}$ is $t_c(e_{i,j}^s, \beta_{k,l}) = c(e_{i,j}^s)/\beta_{k,l}$. This means that both processing time and data transfer time are not changed with time, and suppose that data transfer time within one computer is negligible.

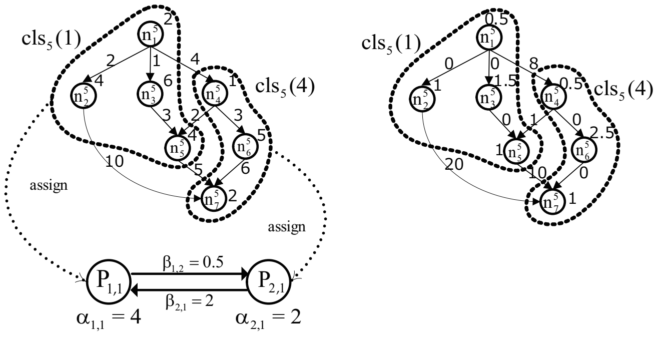
III. Problem Definition

A. Processor Utilization in Singlecore Distributed systems

In this paper, we propose a method for processor utilization in a distributed system, where each computer has one or more PEs. The difference between the method proposed in [11] and a method in this paper is whether such multicore computers are considered or not. According to [11], "processor utilization", in other words, "effective use of computational resources", means to maximize the degree of contribution for reducing the schedule length for each PE. Fig. 1 shows a process flow we are assuming. A job is processed by three steps, i.e., (I) a lower bound for each cluster (set of tasks) execution time is derived, (II) mapping procedure to assign a PE to a cluster, (III) task clustering algorithm to generate actual clusters. Since these procedures were proposed in [11], we describe only the abstract.

1) Lower Bound Derivation for each Cluster Execution Time: According to fig. 1, the method proposed in [11] derives a lower bound for each cluster execution time (total task execution time in a cluster on a PE). To impose a lower bound, the number of required PEs is limited to some extent. At the same time, the response time should be minimized for achieving processor utilization.

Since the schedule length can not be derived until every task is scheduled, each lower bound should be decided with estimating the schedule length. This is because these three procedures in fig. 1 are performed before a task scheduling[14]. Thus, the method in [11] defines an indicative value for the schedule length as WSL (Worst Schedule Length) [11]. Let



(a) After 5 tesk merging steps have been done. (b) After processor assignments have been done.

Fig. 2. Example of WSL

define "WSL after s-th task merging step" as $sl_w(G_{cls}^s, \phi_s)$, which means the largest value that the response time can take when every task is executed as late as possible after s task merging steps and the mapping state ϕ_s has been decided.

Fig. 2 shows one example of how WSL is decided. In this figure, (a) is the state after 4 task merging steps have been finished, and (b) is the state that each generated cluster is assigned to one actual PE. In (b), note that data communication time in the same PE is set to 0. Then we WSL is derived by tracing $n_1^5 \to n_3^5 \to n_5^5 \to n_2^5 \to n_7^5$, i.e., $sl_w(G_{cls}^6, \phi_6) = 25$.

In [11], it is proved that minimizing WSL has good effect on the schedule length, i.e., it is necessary that WSL should be minimized to minimize the schedule length. Since actual WSL after s-th task merging step can not be decided until s task merging steps have been finished, we must estimate the upper bound of WSL after $s-1$ task merging steps have been finished(i.e., before s-th task merging step). If we define an upper bound of the difference between WSL after s-th task merging step and the initial WSL(i.e., 0-th task merging step) as $\Delta sl_w^{s-1}(\delta)$, we have

$$\Delta sl_{w,up}^{s-1}(\delta) = \left(\frac{\max\limits_{n_k^0 \in V_0} \left\{ w(n_k^0) \right\}}{\alpha_p} + \frac{\max\limits_{e_{k,l}^0 \in E_0} \left\{ c(e_{k,l}^0) \right\}}{\beta(p)} \right) \frac{\sum\limits_{n_k^0 \in seq_{s-1}^{\prec}} w(n_k^0)}{\delta \alpha_p} + \delta - \frac{\max\limits_{e_{k,l}^0 \in E_0} \left\{ c(e_{k,l}^0) \right\}}{\max\limits_{\beta_{p,q} \in \beta} \{\beta_{p,q}\}}, \tag{2}$$

where ϕ_0 is the initial mapping state that each task is assigned to a "virtual PE" having the maximum processing speed and the maximum communication bandwidth. At the mapping state ϕ_0, the critical path length equals to the WSL at the initial state ϕ_0 (for more details, see [11]). At eq. (2), δ is the lower bound of each cluster execution time. δ is derived by temporarily assuming identical network, where we derive the lower bound for every cluster execution time on tasks dominating the WSL. $seq_{s-1}^{\prec}$ is a set of tasks on a path dominating $sl_w(G_{cls}^{s-1}, \phi_{s-1})$ (e.g., at (b) in fig. 2, $seq_5^{\prec}$ is $\{n_1^5, n_3^5, n_5^5, n_7^5\}$ or $\{n_1^5, n_2^5, n_7^5\}$).

Fig. 3 shows one example how δ is derived. In this figure, (a) is the state after 4 task merging steps have been finished. In (a) there are unmerged tasks each of which is assigned to the

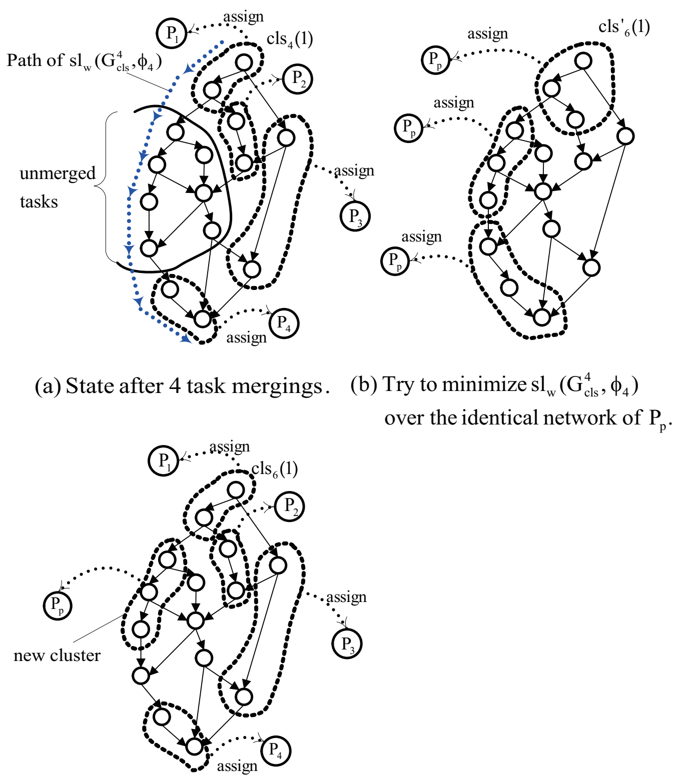

assign
P_1
$cls_4(1)$
Path of $sl_w(G^4_{cls},\phi_4)$
P_2
unmerged tasks
assign
P_3
P_4
assign
(a) State after 4 task mergings.
$cls'_6(1)$
assign
P_p
assign
P_p
assign
P_p
(b) Try to minimize $sl_w(G^4_{cls},\phi_4)$ over the identical network of P_p.
assign
P_1
$cls_6(1)$
P_2
P_p
assign
new cluster
P_3
P_4
assign
(c) State after 6 task mergings.

Fig. 3. Assumed model for WSL derivation

"virtual PE" having the maximum processing speed and communication bandwidth, respectively. On the other hand, other tasks are assigned to an actual PE. In this state, the dashed line corresponds to the path of WSL, i.e., $sl_w(G^4_{cls},\phi_4)$. Then we temporarily assume these tasks will be clustered and each cluster is assigned to an identical PE(in (b), the PE is P_p). We have $\Delta sl^4_{w,up}(\delta)$, which is a function with respect to δ. From (b) we obtain the optimal value of δ (let call "δ_{opt}") by which $\Delta sl^4_{w,up}(\delta)$ is minimized. Once δ_{opt} is obtained, every cluster in (b) is restored to clusters in (a) and then unmerged tasks in (a) are merged into one new cluster until its execution time exceeds δ_{opt} (see (c)). It is assumed that the cluster execution time of the new cluster in (c) exceeds δ_{opt} after 6 task merging steps (it was not enough after 5 task merging steps).

We obtain the minimum value of $sl^{s-1}_{w,up}$ by differentiating eq. (2) with respect to δ. If we define the value of δ when $sl^{s-1}_{w,up}$ takes the minimum as $\delta^s_{opt}(P_p)$, we have

$$\delta^s_{opt}(P_p) = \sqrt{\frac{\sum\limits_{n^0_k\in seq^{\prec}_{s-1}} w(n^0_k)}{\alpha_p}\left(\frac{\max\limits_{n^0_k\in V_0}\{w(n^0_k)\}}{\alpha_p}+\frac{\max\limits_{e^0_{k,l}\in E_0}\left\{c(e^0_{k,l})\right\}}{\beta(p)}\right)},$$

where $\beta(p)$ is the minimum communication bandwidth of P_p. $\delta^s_{opt}(P_p)$ is the lower bound of the cluster execution time to be generated after s-th task merging step. If $\delta^s_{opt}(P_p)$ is applied to $\Delta sl^{s-1}_{w,up}(\delta)$, we have

$$\Delta sl^{s-1}_{w,up}(\delta^s_{opt}(\alpha_p,\beta(p))) = \frac{2}{\alpha_p}\times \sqrt{\sum_{n^0_k\in seq^{\prec}_{s-1}} w(n^0_k)\left(\frac{\alpha_p \max\limits_{e^0_{k,l}\in E_0}\left\{c(e^0_{k,l})\right\}}{\beta(p)} + \max_{n^0_k\in V_0}\{w(n^0_k)\}\right)} - \frac{\max\limits_{e^0_{k,l}\in E_0}\left\{c(e^0_{k,l})\right\}}{\max\limits_{\beta_{p,q}\in\beta}\{\beta_{p,q}\}}. \quad (3)$$

2) Processor Selection: At this phase, a PE to be assigned to a new cluster(e.g., "new cluster" at fig. 3) is decided. If $\Delta sl^{s-1}_{w,up}(\delta^s_{opt}(\alpha_p,\beta(p)))$ is minimized by applying actual values of α_p and $\beta(p)$, the PE having these values should be selected for processor assignment. Then the minimized $\Delta sl^{s-1}_{w,up}(\delta^s_{opt}(\alpha_p,\beta(p)))$ is obtained, by which the upper bound of WSL can be minimized.

3) Task clustering: Since the PE to be assigned has been decided, both values of $\delta^s_{opt}(P_p)$ and $\Delta sl^{s-1}_{w,up}(\delta^s_{opt}(\alpha_p,\beta(p)))$ are decided. The objective of the task clustering is to generate the cluster whose execution time exceeds $\delta^s_{opt}(P_p)$ and to minimize WSL, i.e., $sl_w(G^s_{cls},\phi_s)$. One policy of the task clustering is to select $pivot$ (a cluster in which at least one task belongs to $sl_w(G^{s-1}_{cls},\phi_{s-1})$ and $target$ (a cluster in which at least one task has data dependency with one of task in $pivot$). These two clusters are merged into a new cluster, and then if the cluster execution time of the cluster does not exceeds $\delta^s_{opt}(P_p)$, the cluster becomes the "new $pivot$". Then the new $target$ is selected for more merging steps.

By repeating (1) - (3), every cluster execution time becomes larger than each decided lower bound. In such an obtained output DAG, it was found that the degree of contribution for speed up can be maximized[11].

B. Objective of the Proposal

The method for processor utilization described above has good effect on a "single core distributed system", where each PE is completely connected over the network. However, the method does not take into account the locality among PEs. In fig. 1, a multicore distributed system is assumed. If the method proposed in [11] is applied in such an environment, each cluster(assignment unit) is assigned to different computers (see fig. 1). Thus, we propose a processor mapping policy taking into account the locality among PEs. In the top part of fig. 1, two clusters are assigned to different computers. This is because that the method proposed in [11] tries to find the PE to be assigned to a cluster by only applying its performance information to eq. (3). On the other hand, the bottom part of fig. 1 corresponds to the result by applying the proposal. In this case, both clusters are assigned to the same computer. The process of our proposal is performed after three steps ((I)-(III) in fig. 1).

IV. Proposal

As stated in the previous section, the proposed method is performed after three steps proposed in [11] have finished. In

```
INPUT: $< G^R_{cls}, \phi_R >$, where $R$ is ♯ of task merging step required to obtain the output DAG of [11].
OUTPUT: $< G^R_{cls}, \phi_{R'} >$, where $R'$ is $R$ plus "♯ of the remapping required to obtain the output DAG of the proposal".
Define $P(cls_R(k), \phi_R) = P_{i,p}$, where $cls_R(k)$ is assigned to $P_{i,p}$ at $\phi_R$.
Define $M(cls_R(k), \phi_R) = M_i$, where $P(cls_R(k), \phi_R) = P_{i,p}$ is included in $M_i$.
Define $com_R(k)$ as the set of tasks in $cls_R(k)$ s.t., they have outgoing or incoming communication with other clusters.
Define $CHK$ is the set of clusters dominating $sl_w(G^R_{cls}, \phi_R)$ obtained by the method of [11].
1.  WHILE $CHK \neq \emptyset$ DO
2.      Find $cls_R(k)$ s.t.,
          $\varphi(cls_R(k)) = \max_{cls_R(i) \in CHK} \{\varphi(cls_R(i))\}$;
3.      Set an immediate predecessor or successor task of $in_R(k)$
          or $out_R(k)$ as $n^R_u$, s.t., $n^R_u \in cls_R(l)$.;
4.      FOR EACH $cls_R(l)$ DO
5.        $\Delta(k, \alpha_{i,p}, \alpha_{j,q}) \leftarrow \varepsilon(k, \alpha_{i,p}, \alpha_{j,q}) - \omega(k, \alpha_{j,q})$;
6.      END FOR
7.      Find max of $\Delta(k, \alpha_{i,p}, \alpha_{j,q})$ and move $cls_R(k)$ to $P_{j,q}$.;
8.      Remove $cls_R(k)$ from $CHK$.;
9.  END WHILE
```

Fig. 4. Procedures for cluster remapping.

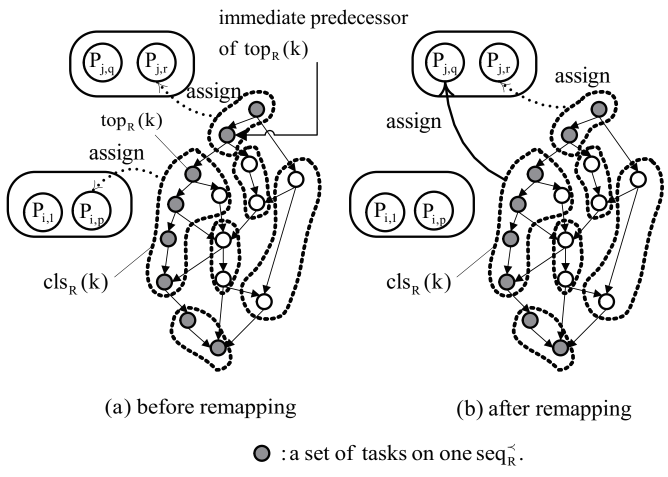


Fig. 5. Example of the cluster remapping algorithm

this section, we present details in the proposal.

A. Algorithm Overview

As stated above, it was proved that minimizing WSL can lead to reduction of the schedule length. Thus, at first clusters dominating WSL must be specified for "remapping". This is because such a remapping can make WSL smaller than that obtained after three steps ((I)-(III) at fig. 1), thereby it is conceivable that the schedule length is also made smaller by the localization of the data communication.

Fig. 4 shows the cluster remapping algorithm, and fig. 5 shows one example of the algorithm with using the same notations as fig. 4. The algorithm firstly tries to find a cluster to be moved by tracing tasks in $seq^{\prec}_R$ (line 1 - 9 at fig. 4). In fig. 5, (a) means the state after procedures proposed in [11] have been finished. In this state, suppose that colored tasks belong to $seq^{\prec}_R$. At first, one cluster in which at least one task belongs to $seq^{\prec}_R$ is selected for the remapping algorithm. The selection criterion is based on calculating the evaluation value(i.e., Δ function at line 7 in fig. 4). This evaluation value means how long the response time can be reduced by moving the cluster to another PE. For each cluster dominating WSL, the algorithm derive Δ function as an evaluation value and then select the cluster as a moving target by which the response time can effectively reduced.

B. Cluster Selection

The proposed cluster remapping is to move each cluster dominating $sl_w(G^R_{cls}, \phi_R)$ to one of unassigned PEs. Thus, as first the algorithm finds one of those clusters for each moving procedure. Intuitively, the cluster to be selected for the remapping is the one which has the greatest effect on the WSL, i.e., that has the most execution time and communication time in $sl_w(G^R_{cls}, \phi_R)$. This is because that WSL can be largely reduced by moving such a cluster to the faster PE. Moreover, localization of large amont of communication can contribute to the reduction of WSL. Thus, the criterion for selecting the cluster to be moved is the one having the maximum sum of communication time and task execution time in WSL.

Next, we describe the details of the criterion for cluster selection. Let us return to the definition of $sl_w(G^R_{cls}, \phi_R)$, whose details of the derivation are defined in table. I(for more details, see [12]). As can be seen in table. I, $sl_w(G^R_{cls}, \phi_R)$ is derived by taking the maximum of $LV_R(i)$, which is derived by applying $TL_R(i)$ and $BL_R(i)$. $TL_R(i)$ means the maximum start time of a task in $cls_R(i)$, while $BL_R(i)$ means the maximum elapsed time from start time of a task to the finish time of the END task including communication time(i.e., the path length from a task to the END task). In the table, $in_R(i)$ and $out_R(i)$ are sets of tasks having incoming communications and outgoing communications, respectively (In fig. 2, $in_5(1) = \{n^5_5\}$, and $out_5(1) = \{n^5_1, n^5_2, n^5_5\}$). Communication time taken by a cluster in WSL is derived by an incoming communications(edge) and an outgoing communications(edge). On the other hand, task execution time taken by a cluster (let define as $cls_R(i)$ in WSL is derived by calculating the maximum of $S(n^R_k, i)$ for each $n^R_k \in cls_R(i)$ defined in table. I. $S(n^R_k, i)$ means the start time of n^R_k in $cls_R(i)$ when n^R_k is scheduled as late as possible. For example, in fig. 2(b), we have

$$\begin{aligned} S(n^5_3) &= \frac{1}{4} \sum_{n^5_k \in cls_5(1)} w(n^5_k) - \frac{1}{4}(w(n^5_3) + w(n^5_5)) \\ &= \frac{1}{4}(w(n^5_1) + w(n^5_2)) = 1.5. \end{aligned} \tag{4}$$

After the time of $S(n^R_k, i)$ has been elapsed, n^R_k can be processed. Then n^R_k sends data to other tasks in other clusters as outgoing communications. Thus, the total time taken by data communication and task execution of one cluster in WSL can be defined as follow.

$$\begin{aligned} \varphi(cls_R(i)) &= com_{in}(cls_R(i)) + S(n^R_k, i) + t_p(n^R_k, \alpha_{p,q}) \\ &\quad + com_{out}(cls_R(i)), \end{aligned} \tag{5}$$

where $com_{in}(cls_R(i))$ is the incoming communication time as a part of $TL_R(i)$, $com_{out}(cls_R(i))$ is the outgoing communication time as a part of $BL_R(i)$. That is, $\varphi(cls_R(i))$ is

TABLE I
PARAMETER DEFINITION RELATED TO $sl_w(G^R_{cls})$ (HERE $n^R_k \in cls_R(i)$).

Parameter	Definition
$top_R(i)$	$\left\{n^R_k \mid \forall n^R_l \in pred(n^R_k) s.t., n^R_l \notin cls_R(i)\right\} \cup \{START\ Tasks \in cls_R(i)\}.$
$in_R(i)$	$\left\{n^R_k \mid \exists n^R_l \in pred(n^R_k) s.t., n^R_l \notin cls_R(i)\right\} \cup \{START\ Tasks \in cls_R(i)\}.$
$out_R(i)$	$\left\{n^R_k \mid \exists n^R_l \in suc(n^R_k) s.t., n^R_l \notin cls_R(i)\right\} \cup \{END\ Tasks \in cls_R(i)\}.$
$desc(n^R_k, i)$	$\{n^R_l \mid n^R_k \prec n^R_l, n^R_l \in cls_R(i)\} \cup \{n^R_k\}$
$S(n^R_k, i)$	$\sum_{n^R_l \in cls_R(i)} t_p(n^R_l, \alpha_p) - \sum_{n^R_l \in desc(n^R_k, i)} t_p(n^R_l, \alpha_p).$
$tlevel(n^R_k)$	$\begin{cases} \max\limits_{n^R_l \in pred(n^R_k)} \left\{tlevel(n^R_l) + t_p(n^R_l, \alpha_p) + t_c(e_{l,k}, \beta_{q,p})\right\}, & if\ n^R_k \in top_R(i). \\ TL_R(i) + S(n^R_k, i), & otherwise. \end{cases}$
$TL_R(i)$	$\max\limits_{n^R_k \in top_R(i)} \left\{tlevel(n^R_k)\right\}..$
$blevel(n^R_k)$	$\max\limits_{n^R_l \in suc(n^R_k), n^R_l \notin cls_R(i)} \left\{t_p(n^R_k, \alpha_p) + t_c(e^R_{k,l}, \beta_{p,q}) + blevel(n^R_l)\right\}.$
$level(n^R_k)$	$tlevel(n^R_k) + blevel(n^R_k).$
$BL_R(i)$	$\max\limits_{n^R_k \in out_R(i)} \left\{S(n^R_k, i) + blevel(n^R_k)\right\}.$
$LV_R(i)$	$TL_R(i) + BL_R(i) = \max\limits_{n^R_k \in cls_R(i)} \left\{level(n^R_k)\right\}.$
$sl_w(G^R_{cls}, \phi_R)$	$\max\limits_{cls_R(i) \in V^R_{cls}} \{LV_R(i)\}.$

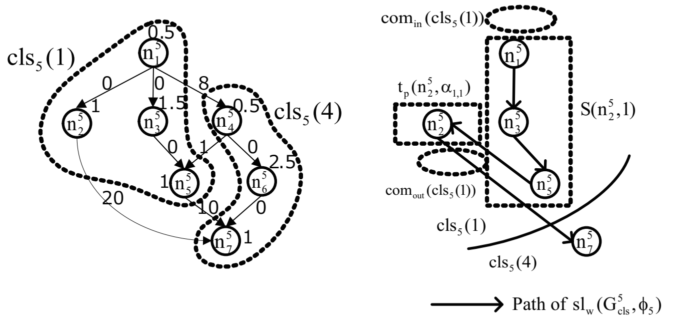


Fig. 6. Example of the cluster selection criteria

total time involved by $cls_R(i)$.

Fig. 6 shows one example of how $\varphi(cls_R(i))$ is decided. In this figure, the path of WSL is the same as fig. 2. At the left side, $cls_5(1)$ and $cls_5(4)$ have been assigned to individual PEs, and the right side shows elements of $\varphi(cls_5(1))$. A dashed circle corresponds to the area for communications, while a dashed rectangle corresponds to the area for task executions. In this example, $com_{in}(cls_5(1))$ is 0 because there is no incoming communication at $cls_5(1)$). Since the path of $sl_5(G^5_{cls})$ is $n^5_1, n^5_3, n^5_5, n^5_2$ as fig. 2, we have

$$\begin{aligned} com_{in}(cls_5(1)) &= 0, \\ com_{out}(cls_5(1)) &= t_c(e^5_{2,7}, \beta_{1,2}) = 10, \\ \varphi(cls_5(1)) &= com_{in}(cls_5(1)) + S(n^5_2, 1) \\ &\quad + t_p(n^5_2, \alpha_{1,1}) + com_{out}(cls_5(1)) \\ &= 0 + 3 + 1 + 20 = 24. \end{aligned} \tag{6}$$

In fig. 4, at line 2 the algorithm finds a cluster $cls_R(k)$ having the maximum φ value in CHK for cluster remapping.

C. Cluster Remapping

Suppose that a cluster which will be checked is $cls_R(k)$, and $cls_R(k)$ was assigned to $P_{i,p} \in M_i$ by the method[11]. Then the algorithm calculates the difference between $\varepsilon(k, \alpha_{i,p}, \alpha_{j,r})$ and $\omega(k, \alpha_{j,q})$. They are defined as follows.

$$\begin{aligned} &\varepsilon(k, \alpha_{i,p}, \alpha_{j,r}) \\ &= \frac{1}{\alpha_{i,p}} \sum_{n^R_u \in cls_R(k)} w(n^R_u) + \frac{1}{\beta_{q,p}} \sum_{\substack{n^R_x \in cls_R(k), \\ n^R_y \in cls_R(l)}} c(e_{y,x}) \\ &+ \frac{1}{\beta_{q,p}} \sum_{\substack{n^R_x \in cls_R(k), \\ n^R_y \in cls_R(l)}} c(e_{x,y}), \end{aligned} \tag{7}$$

$$\omega(k, \alpha_{j,q}) = \frac{1}{\alpha_{j,q}} \sum_{n^R_u \in cls_R(k)} w(n^R_u). \tag{8}$$

$\varepsilon(k, \alpha_{i,p}, \alpha_{j,r})$ is the sum of cluster execution time and data communication time (incoming communication and outgoing communication with a specific cluster (let define as $cls_R(l)$ and $cls_R(l)$ is assigned to $P_{j,r}$)). That is, $\varepsilon(k, \alpha_{i,p}, \alpha_{j,r})$ means the total time taken to process $cls_R(k)$ and to communicate with $cls_R(l)$. On the other hand, $\omega(k, \alpha_{j,q})$ does not require data communication time with $cls_R(l)$ because it is included in the same computer(let assume no cluster is assigned to $P_{j,q}$). Then we define the evaluation value as follow.

$$\Delta(k, \alpha_{i,p}, \alpha_{j,q}) = \varepsilon(k, \alpha_{i,p}, \alpha_{j,q}) - \omega(k, \alpha_{j,q}). \tag{9}$$

For each cluster $cls_R(k)$, the algorithm finds the maximum value of $\Delta(k, \alpha_{i,p}, \alpha_{j,q})$ with varying j and q. This is the same meaning as varying l in $cls_R(l)$(at line 5 in fig. 4). When the cluster $cls_R(l)$ or $P_{j,q}$ is found at line 7 in fig. 4, the cluster is removed from the set of clusters CHK. When CHK becomes empty, the algorithm is finished.

Selection of the target for moving the cluster $cls_R(k)$ by maximizing $\Delta(k, \alpha_{i,p}, \alpha_{j,q})$ means that the algorithm tries to find the PE by which the time taken for processing and data communication is minimized.

V. Theoretical Analysis

Since an objective of the proposed cluster remapping is to reduce WSL in order to minimize the schedule length, with maintaining the number of required PEs. Thus, in this section we analyze the possibility for WSL reduction by each cluster remapping procedure. Assume that an cluster $cls_R(k)$ is assigned to $P_{i,p}$ and $cls_R(k)$ dominates WSL, i.e., $sl_W(G^R_{cls})$. Then suppose that $cls_R(k)$ is moved to $P_{j,q}$ by a cluster remapping procedure. We define $\varphi(cls_R(k))$ after the remapping as $\varphi\prime(cls_R(k))$ and the difference of $\varphi(cls_R(k))$ as $\Delta\varphi(cls_R(k))$. Similarly, let differences of $com_{in}(cls_R(k))$, $S(n^R_m)$, and $com_{out}(cls_R(k))$ be $\Delta com_{in}(cls_R(k))$, $\Delta S(n^R_m)$, and $\Delta com_{out}(cls_R(k))$, respectively. Then we have

$$\begin{aligned}\Delta\varphi(cls_R(k)) &= \varphi(cls_R(k)) - \varphi\prime(cls_R(k))\\ &= \Delta com_{in}(cls_R(k)) + \Delta S(n^R_m)\\ &\quad +t_p(n^R_m, \alpha_{i,p}) - t_p(n^R_m, \alpha_{j,q})\\ &\quad +\Delta com_{out}(cls_R(k))\\ &= \left(\frac{1}{\beta_{u,i}} - \frac{1}{\beta_{u,j}}\right) c(e^R_{r,s})\\ &\quad + \left(\frac{1}{\alpha_{i,p}} - \frac{1}{\alpha_{j,q}}\right) \sum_{n^R_l \in cls_R(k)} w(n^R_l)\\ &\quad + \left(\frac{1}{\beta_{i,v}} - \frac{1}{\beta_{j,v}}\right) c(e^R_{y,z}), \qquad (10)\end{aligned}$$

where $cls_R(k)$ is moved from $P_{i,p}$ to $P_{j,q}$, while $e^R_{r,s}$ is a communication part of $TL_R(k)$ ($n^R_r \notin cls_R(k), n^R_s \in cls_R(k)$), and $e^R_{y,z}$ is a communication part of $BL(k)$ ($n^R_y \in cls_R(k), n^R_z \notin cls_R(k)$). If $\varphi(cls_R(k)) > 0$, WSL can be reduced by this cluster remapping procedure. Otherwise, WSL may not be reduced. In this analysis we present the lower bound of $\varphi(cls_R(k))$ in several cases.

At line 5 in fig. 4, the target PE for cluster remapping is selected by computing every incoming and outgoing communication of the cluster selected at line 2 in fig. 4. Thus, we have 2 cases for analyzing the effect on the WSL reduction by cluster remapping, i.e., (i) $com_{in}(cls_R(k))$ or $com_{out}(cls_R(k))$ is localized and the WSL path is not changed, and (ii) both $com_{in}(cls_R(k))$ and $com_{out}(cls_R(k))$ are not localized (other communications are localized).

(i) $com_{in}(cls_R(k))$ or $com_{out}(cls_R(k))$ is localized and the WSL is not changed:
This case means that $u = j$ or $j = v$ obtained by cluster remapping. Consequently, $\frac{1}{\beta_{u,j}} = 0$ or $\frac{1}{\beta_{j,v}} = 0$. Without loss of generality, suppose that $\frac{1}{\beta_{u,j}} = 0$, while $com_{out}(cls_R(k))$ remains unlocalized, i.e., $\frac{1}{\beta_{j,v}} \neq 0$.

From eq. (10), we have

$$\begin{aligned}\Delta\varphi(cls_R(k)) &= \frac{1}{\beta_{u,i}} c(e^R_{r,s})\\ &\quad + \left(\frac{1}{\alpha_{i,p}} - \frac{1}{\alpha_{j,q}}\right) \sum_{n^R_l \in cls_R(k)} w(n^R_l)\\ &\quad + \left(\frac{1}{\beta_{i,v}} - \frac{1}{\beta_{j,v}}\right) c(e^R_{y,z}). \qquad (11)\end{aligned}$$

From eq. (11), there are 4 cases for satisfying $\varphi(cls_R(cls(k)) \geq 0$ depending on magnitude relationships of $\alpha_{i,p}$ to $\alpha_{j,q}$, and $\beta_{i,v}$ to $\beta_{j,v}$.

(i-I) The case of $\alpha_{i,p} \leq \alpha_{j,q}$, $\beta_{i,v} \leq \beta_{j,v}$:
In this case, the target PE to which $cls_R(k)$ has faster processing speed and wider communication bandwidth. Let the maximum of processing speed, communication bandwidth, and data size be $\alpha_{max}, \beta_{max}, c_{max}$, respectively. Let the minimum of processing speed, communication bandwidth, and data size be $\alpha_{min}, \beta_{min}, c_{min}$, respectively. Then we have

$$\begin{aligned}\Delta\varphi(cls_R(k)) &\geq \frac{1}{\beta_{max}} c_{min}\\ &\quad + \left(\frac{1}{\alpha_{i,p}} - \frac{1}{\alpha_{j,q}}\right) \sum_{n^R_l \in cls_R(k)} w(n^R_l)\\ &\quad + \left(\frac{1}{\beta_{i,v}} - \frac{1}{\beta_{j,v}}\right) c(e^R_{y,z})\\ &\geq \frac{1}{\beta_{max}} c_{min} > 0. \qquad (12)\end{aligned}$$

Thus, in this case, $\varphi(cls_R(k))$ always takes the positive value.

(i-II) The case of $\alpha_{i,p} \leq \alpha_{j,q}$, $\beta_{i,v} \geq \beta_{j,v}$:
In this case, we have

$$\begin{aligned}\Delta\varphi(cls_R(k)) &\geq \frac{1}{\beta_{max}} c_{min}\\ &\quad + \left(\frac{1}{\beta_{max}} - \frac{1}{\beta_{min}}\right) c_{max}\\ &= \frac{1}{\beta_{max}} (c_{min} + c_{max}) - \frac{c_{max}}{\beta_{min}} \qquad (13)\end{aligned}$$

At eq. (13), the condition for satisfying $\varphi(cls_R(k)) \geq 0$ is as follow.

$$\begin{aligned}&\frac{1}{\beta_{max}} (c_{min} + c_{max}) - \frac{c_{max}}{\beta_{min}} \geq 0\\ &\Leftrightarrow \frac{\beta_{max}}{\beta_{min}} \leq \frac{c_{min}}{c_{max}} + 1 \leq 2. \qquad (14)\end{aligned}$$

From eq. (13), the lower bound becomes small if the heterogeneity (i.e., max to min ratio) in terms of communication bandwidth is also small. If eq. (13) is satisfied, $\varphi(cls_R(k))$ takes the positive value irrespective of processing speed.

(i-III) The case of $\alpha_{i,p} \geq \alpha_{j,q}$, $\beta_{i,v} \leq \beta_{j,v}$:
In this case, the target PE has slower processing

speed than $P_{i,p}$. Then we have

$$\begin{aligned}
&\Delta\varphi(cls_R(k)) \\
&\geq \frac{c(e^R_{r,s})}{\beta_{u,i}} + \left(\frac{1}{\alpha_{i,p}} - \frac{1}{\alpha_{j,q}}\right) \sum_{n^R_l \in cls_R(k)} w(n^R_l) \\
&\geq \frac{c_{min}}{\beta_{max}} + \left(\frac{1}{\alpha_{max}} - \frac{1}{\alpha_{min}}\right) \sum_{n^R_l \in cls_R(k)} w(n^R_l).
\end{aligned} \tag{15}$$

The condition for satisfying $\varphi(cls_R(k)) \geq 0$ is as follow.

$$\begin{aligned}
\frac{\alpha_{max}}{\beta_{max}} c_{min} &\geq \left(\frac{\alpha_{max}}{\alpha_{min}} - 1\right) \sum_{n^R_l \in cls_R(k)} w(n^R_l) \\
\Leftrightarrow \frac{\alpha_{max}}{\beta_{max}} &\geq \left(\frac{\alpha_{max}}{\alpha_{min}} - 1\right) \frac{\sum_{n^R_l \in cls_R(k)} w(n^R_l)}{c_{min}}
\end{aligned} \tag{16}$$

In eq. (15), if α_{max} is much larger than β_{max} and heterogeneity in terms of processing speed is small, $\varphi(cls_R(k))$ has possibility of taking the positive value.

(i-IV) The case of $\alpha_{i,p} \geq \alpha_{j,q}$, $\beta_{i,v} \geq \beta_{j,v}$:
In this case, we have

$$\begin{aligned}
&\Delta\varphi(cls_R(k)) \\
&\geq \frac{c(e^R_{r,s})}{\beta_{u,i}} + \left(\frac{1}{\alpha_{i,p}} - \frac{1}{\alpha_{j,q}}\right) \sum_{n^R_l \in cls_R(k)} w(n^R_l) \\
&\quad + \left(\frac{1}{\beta_{i,v}} - \frac{1}{\beta_{j,v}}\right) c(e^R_{y,z}) \\
&\geq \frac{c_{min}}{\beta_{max}} + \left(\frac{1}{\alpha_{max}} - \frac{1}{\alpha_{min}}\right) \sum_{n^R_l \in cls_R(k)} w(n^R_l) \\
&\quad + \left(\frac{1}{\beta_{max}} - \frac{1}{\beta_{min}}\right) c_{max}.
\end{aligned} \tag{17}$$

The condition for satisfying $\varphi(cls_R(k)) \geq 0$ is as follows.

$$\begin{aligned}
\frac{c_{min} + c_{max}}{\beta_{max}} \geq &\left(\frac{1}{\alpha_{min}} - \frac{1}{\alpha_{max}}\right) \sum_{n^R_l \in cls_R(k)} w(n^R_l) \\
&+ \frac{c_{max}}{\beta_{min}}.
\end{aligned} \tag{18}$$

From this result, at least heterogeneity in terms of processing speed and communication bandwidth should be small in order to satisfy $\varphi(cls_R(k)) \geq 0$.

(ii) Nor $com_{in}(cls_R(k))$ and $com_{out}(cls_R(k))$ are localized: Since the algorithm tries to trace incoming edges and outgoing edges to find the target PE, the edge which corresponding to $com_{in}(cls_R(k))$ or $com_{out}(cls_R(k))$ is not always localized by the proposed cluster remapping. This case assumes that some edges which are not parts of $LV_R(k)$ are localized. In this case, the new WSL is derived by tracing every edges and tasks after cluster remapping. This fact means that the path of WSL can be changed. Thus, the lower bound of $\varphi(cls_R(k))$ can not be estimated.

TABLE II
COMPARISON RESULTS IN TERMS OF THE SCHEDULE LENGTH

No.	α	β	CCR	$\lvert V^R_{cls} \rvert$	Schedule Length		
					A	B	C
1	5	5	0.1	156	1.000	1.002	1.002
2			1.0	58	1.000	1.011	1.008
3			5.0	42	1.000	1.031	1.048
4			10.0	16	1.000	1.109	1.144
5	5	10	0.1	165	1.000	1.001	1.044
6			1.0	55	1.000	1.076	1.071
7			5.0	32	1.000	1.103	1.142
8			10.0	14	1.000	1.107	1.281
9	10	5	0.1	139	1.000	1.001	1.003
10			1.0	48	1.000	1.068	1.042
11			5.0	37	1.000	1.103	1.009
12			10.0	27	1.000	1.198	1.201
13	10	10	0.1	188	1.000	1.004	1.002
14			1.0	67	1.000	1.143	1.107
15			5.0	40	1.000	1.170	1.140
16			10.0	29	1.000	1.179	1.151

From results of the theoretical analysis a cluster to be selected at line 7 in fig. 4 should be the case of (i-I). However, the algorithm can not always select by (i-I) because WSL after each cluster remapping may be changed due to the variation of communication time and processing time for each task. This means that a strategy to use (i-I) in every cluster remapping step has possibility to increase WSL by the change of the path dominating WSL. Thus, at eq. (7) the algorithm tries to find the target PE by summing up all communication times involved by the cluster selected at line 2 in fig. 4 in order to localize communication as many as possible.

VI. EXPERIMENTS

We conducted the experiments by a simulation to confirm advantages of our proposal, i.e., how the cluster remapping can reduce the schedule length. We compared with the method proposed in [11] in terms of the following points.

1) Effect on the cluster remapping with compared to the case of "non remapping".
2) Effect on the proposed remapping policy and algorithm.

Thus, comparison targets are, (A) the proposed cluster remapping, (B) each cluster is randomly remapped. (C) non remapping[11].

A. Experimental Environment

In the simulation, a random DAG is generated. In each DAG, each task size and each data size are decided randomly. Also CCR (Communication to Computation Ratio)[13], [14] is changed from 0.1 to 10. The max to min ratio in terms of data size and task size is set to 100. Also we decided the Parallelism Factor (PF) is defined as ρ, taking values of 0.5, 1.0, and 2.0 [15]. By using PF, the depth of the DAG is defined as $\frac{\sqrt{|V_0|}}{\rho}$. Since we assume a heterogeneous distributed system by multicore computers, each PE's performance and network bandwidth are varied by parameters. The simulation environment was developed by JRE1.6.0_0, the operating system is Windows XP SP3, the CPU architecture is Intel Core 2 Duo 2.66GHz, and the memory size is 2.0GB.

TABLE III
COMPARISON RESULTS IN TERMS OF THE SCHEDULE LENGTH AND SELECTED PE TYPES

No.	α	β	CCR	$\lvert V_{cls}^{R}\rvert$	Schedule Length Raio to (A)			Ratio of of (i-I) (%)			Ratio of (i-IV) (%)		
					A	B	C	A	B	C	A	B	C
1	5	5	0.1	143	1.000	1.003	1.011	24.0	40.0	9.3	6.7	16.0	46.7
2			1.0	67	1.000	1.010	1.013	27.9	44.2	11.6	4.7	14.0	51.2
3			5.0	41	1.000	1.044	1.089	29.2	58.3	8.3	8.3	16.7	45.8
4			10.0	14	1.000	1.072	1.182	33.3	50.0	0.0	0.0	33.3	83.3
5	5	10	0.1	166	1.000	1.002	1.009	27.0	39.2	5.4	4.1	14.9	39.2
6			1.0	73	1.000	1.013	1.023	34.1	53.7	4.9	7.3	17.1	39.0
7			5.0	38	1.000	1.041	1.099	36.4	72.7	4.5	9.1	22.7	45.5
8			10.0	18	1.000	1.102	1.183	57.1	71.4	0.0	11.7	14.3	42.9
9	10	5	0.1	145	1.000	1.000	1.009	22.9	47.1	7.1	8.6	15.2	41.4
10			1.0	55	1.000	1.018	1.026	22.4	53.1	6.1	10.2	16.3	22.4
11			5.0	39	1.000	1.039	1.088	43.5	82.6	8.7	13.0	8.7	21.7
12			10.0	13	1.000	1.121	1.282	33.3	50.0	0.0	0.0	16.7	66.7
13	10	10	0.1	173	1.000	1.006	1.087	28.6	48.6	8.6	7.1	15.7	34.3
14			1.0	66	1.000	1.149	1.181	34.2	50.0	7.9	10.5	15.8	39.5
15			5.0	40	1.000	1.166	1.159	27.3	54.5	4.5	4.5	9.1	45.5
16			10.0	26	1.000	1.133	1.198	37.5	50.0	0.0	0.0	12.5	37.5

B. Effect on Cluster Remapping

One comparison is done by executing 100 random DAGs and averaging the schedule length for each approach (A)-(C). Since (C) does not require cluster remapping, at first the simulation output the schedule length by (C). Then each remapping procedures is performed in (A) and (B), respectively. Then both approaches output each schedule length for the same DAG as (C). Table II shows comparison results by the experiment. In the table, α and β are ratios of maximum to minimum processing speed and maximum to minimum communication bandwidth, respectively. In the 6th column, values mean the schedule length ratio when the schedule length obtained by (A) is set to 1. Thus, a value over 1 is worse than (A). In every cases, it is observed that the proposed method (A) outperforms other approaches. If β is small such as No. 1-5 and No. 9-12, we can not observe noticeable characteristics among three approaches, except that the schedule length obtained by (A) is the smallest. On the other hand, β is large such as No. 5-8, the schedule length obtained by (C) is considerably large with increasing CCR. From this result, it is conceivable that data localization is necessary irrespective of cluster remapping methods if each data size is large. As for No. 13-16, the schedule length obtained by (B) is the largest with increasing CCR. From this result, the cluster remapping method has large effect on the schedule length if the heterogeneity of the network is large.

From these results, it is concluded that the proposed cluster remapping method has good effect on the schedule length by effectively localizing data communications among PEs.

C. Relationship between PE Selection Policies and the Schedule Length

According to the results obtained in sec. V, selecting the target PE having faster processing speed and larger communication bandwidth (type of (i-I)) in sec. V than the original PE can lead to the reduction of WSL after one cluster remapping procedure has finished. Nonetheless, the proposed algorithm tries to find the target PE by summing up all communication times involved by the moving target cluster rather than finding a cluster of (i-I) in sec. V. This is because not always a cluster of type (i-I) can be found and WSL path may be changed by a cluster remapping procedure, thereby

In this experiment, we compared the schedule length and distribution of the selected PE type ((i-I) to (ii) at sec. V) in (A) proposed algorithm, (B) trying to find a cluster of type (i-I), (C) trying to find a cluster of type (i-IV). This comparison is conducted by averaging the schedule length of DAGs having 1000 tasks by 100 tries for all three approaches. In the method (B), if a cluster of type (i-I) is not found, the method selects a PE randomly from the set of PEs by which at least one communication involved by a cluster selected at line 2 in fig. 4 is localized. In the method (C), if a cluster of type (i-IV) is not found, the target PE is selected as the same policy as (B). Table. III shows the comparison results. "Ratio of (i-I)" means the percentile of the number of that a PE of type (i-I) is selected as the target PE to the number of all cluster remapping, while "Ratio of (i-IV)" means the percentile of the number of that a PE of type (i-IV) is selected as the target PE to the number of all cluster remapping. From this table, it is observed that in every cases (No. 1 - 16), the proposed algorithm has the smallest schedule length, while (B) has the largest ratios of (i-I) and (C) has the largest ratios of (i-IV) in every cases. From this result, it can be conceived that selecting a PE of type (i-IV) has bad effects on both WSL and the schedule length. Furthermore, Even if a PE of type (i-I) is selected as the moving target, the schedule length can be made larger by the facts that WSL path may be changed. This is because an communication time dominating $com_{in}(cls_R(i))$ or $com_{out}(cls_R(i))$ is made smaller by a cluster remapping, while other edges may dominates $com_{in}(cls_R(i))$ or $com_{out}(cls_R(i))$.

From this result, it is concluded that the PE selection criteria by the proposed cluster remapping have good effect on the schedule length.

VII. CONCLUSION AND FUTURE WORKS

In this paper, we presented a cluster remapping method in a multicore distributed system for processor utilization. The method tries to move clusters for data localization with taking into account both data dependencies which can have good effect on the schedule length. This procedure is performed by calculate the difference between time taken to execute every task in the cluster in the new PE and the time taken to execution and data communication in the previously decided PE decided by the method of [11]. From preliminary experiments, the proposed method was proved to have good effect on the schedule length.

As a future work, we develop more efficient method which does not use remapping in multicore distributed systems for processor utilization.

References

[1] I. Foster, "The Grid: A new infrastructure for 21st century science," *Physics Today*, Vol.55(2), pp. 42–47, 2002.
[2] G. DeCandia et al, "Dynamo: amazon's highly available key-value store," *Proc. of ACM SIGOPS symposium on Operating systems principles (SOSP'12)*, pp. 205 – 220, 2007.
[3] S. Di and C.L. Wang, "Dynamic Optimization of Multiattribute Resource Allocation in Self-Organizing Clouds," *IEEE Trans. on Parallel and Distributed Systems*, Vol. 24, No. 3, pp. 464-478, 2013.
[4] M. Xu, L. Cui, H. Wang, and Y Bi, "A Multiple QoS Constrained Scheduling Strategy of Multiple Workflows for Cloud Computing," *Proc. of 2009 IEEE International Symposium on Parallel and Distributed Processing with Applications*, pp. 629–634, 2009.
[5] H. Mohammadi Fard, R. Prodan, J. J. D. Barrionuevo, and T. Fahringer, "A Multi-Objective Approach for Workflow Scheduling in Heterogeneous Environments," *Proc. of 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pp. 300–309, 2012.
[6] J.S. Kim et al., " Using Content-Addressable Networks for Load Balancing in Desktop Grids, " *Proc. of 16th ACM International Symposium on High Performance Distributed Computing (HPDC '07)*, pp. 189- 198, 2007.
[7] A. Leite, H. Mendes, L. Weigang, A. de Melo, and A. Boukerche, "An Architecture for P2P Bag-of-Tasks Execution with Multiple Task Allocation Policies in Desktop Grids, " *Proc. of IEEE International Conference on Cluster Computing*, pp. 1-11, 2011.
[8] A. Gerasoulis and T. Yang, "A Comparison of Clustering Heuristics for Scheduling Directed Acyclic Graphs on Multiprocessors," *Journal of Parallel and Distributed Computing*, Vol. 16, pp. 276-291, 1992.
[9] T. Yang and A. Gerasoulis, "DSC: Scheduling Parallel Tasks on an Unbounded Number of Processors," *IEEE Trans. on Parallel and Distributed Systems*," Vol. 5, No. 9 pp. 951-967, 1994.
[10] J. C. Liou, M. A. Palis, "An Efficient Task Clustering Heuristic for Scheduling DAGs on Multiprocessors," *Proc. of the 8th Symposium on Parallel and Distributed Processing*,October, 1996.
[11] Hidehiro Kanemitsu, Gilhyon Lee, Hidenori Nakazato, Takashige Hoshiai, and Yoshiyori Urano, "A Processor Mapping Strategy for Processor Utilization in a Heterogeneous Distributed System," *Journal of Computing*, Vol. 3, Issue 11, pp. 1 – 8, November 2011.
[12] Hidehiro Kanemitsu, Gilhyon Lee, Hidenori Nakazato, Takashige Hoshiai, and Yoshiyori Urano, "On the Effect of Applying the Task Clustering for Identical Processor Utilization to Heterogeneous Systems," *Grid Computing - Technology and Applications, Widespread Coverage and New Horizons*, InTech(ISBN: 978-953-51-0604-3), pp. 29 - 46, March, 2012.
[13] O. Sinnen and L. A. Sousa, "Communication Contention in Task Scheduling," *IEEE Trans. on Parallel and Distributed Systems*, Vol. 16, No. 6., pp. 503-515, 2005.
[14] O. Sinnen, "Task Scheduling for Parallel Systems," *Wiley*, 2007.
[15] H. Topcuoglu, S. Hariri, M. Y. Wu, "Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing," *IEEE Trans. on Parallel and Distributed Systems*, Vol. 13, No. 3 pp. 260-274, 2002.

Hidehiro Kanemitsu received the B.S. degree from Waseda University, Japan in Science, and the M.S. and D.S. degrees from Waseda University, Japan in Global Information and Telecommunication Studies (GITS). His research interests are in areas of parallel and distributed computing, grid, peer-to-peer computing, and web service technology. He is currently an assistant professor at Media Network Center, Waseda University, Japan.

Masaki Hanada received the B.E. degree in Resources Engineering from Waseda University in 1996, and the M.S. and D.S. degrees in Global Information and Telecommunication Studies from Waseda University in 2003 and 2007, respectively. He has been an assistant professor at the Department of Information Systems, Tokyo University of Information Sciences. His research interests include QoS/traffic control and resource management in communication networks.

Takashige Hoshiai has been a Professor at Sojo University since 2012. He was a senior research scientist supervisor at NTT Network Service Systems Laboratories from 1986 to 2012. He received his Ph.D. degree in Communications and Systems from The University of Electro-Communications, Japan. His research areas are distributed object technologies, real-time systems, agent systems and P2P. Since he proposed a new business model called " Brokerless Model " in 1998, especially, he has studied SIONet architecture that is a solution of P2P platforms.

Hidenori Nakazato received his B. Engineering degree in Electronics and Telecommunications from Waseda University in 1982, and his MS and Ph.D. degrees in Computer Science from University of Illinois in 1989 and 1993, respectively. He wase with Oki Electric from 1982 to 2000 where he developed equipment for public telephony switches, distributed environment for telecommunications systems, and communications quality control mechanisms. He has been a Professor at Graduate School of Global Information and Telecommunications Studies, Waseda University since 2000. His research interests include performance issues in distributed systems and networks, cooperation mechanisms of distributed programs, distributed real-time systems, and network QoS control.