A Processor Mapping Strategy for Processor Utilization in a Heterogeneous Distributed System

Hidehiro Kanemitsu, Gilhyon Lee, Hidenori Nakazato, Takashige Hoshiai, and Yoshiyori Urano

Abstract—In a distributed system where processors are connected over the network, how to minimize the schedule length is one of major objectives in the field of task scheduling. Conventional task clustering heuristics and processor assignment methods for heterogeneous distributed systems are used to generate the assignment unit for the given set of processors. Such approaches try to use all processors for minimizing the schedule length. However, there is no way to know automatically how many processors are needed to minimize the schedule length. Thus, there is no criterion for maximizing the degree of contribution toward reduction of the schedule length per a processor. In this paper, we propose a method for processor mapping for processor utilization. Our proposal imposes the lower bound for every assignment unit size for limiting the number of required processors. Under the constraint, the processor assignment is performed to minimize the schedule length. In the mapping, the processor, by which the indicative value for the schedule length can be minimized, is selected for the assignment. Experimental results by simulations show that our proposal has advantages of processor utilization over conventional approaches.

Index Terms—Processor Mapping, Task Scheduling, Processor Utilization, Hegerogeneouse Distributed Systems

1 Introduction

Task execution models harnessing processors connected over the network, e.g., grid and utility computing, have been studied for minimizing the response time (the schedule length). One of applications to be executed is the DAG (Directed Acyclic Graph), in which every task (a chunk of execution instructions such as functions, loops, and so on) is defined with precedence constraints. This is one type of task graphs which abstracts any program codes. As one approach for generating assignment units with minimizing the schedule length, task clustering heuristics [1,2,3] have been known. In a task clustering, several tasks are merged into one assignment unit (in the following, we shall call "cluster"), such that data communications among them are localized. As a result, the schedule length may be reduced by the data localization.

In a heterogeneous distributed system, where each processor has arbitrary processing speed and communication bandwidth, a processor assignment phase and a task clustering phase are needed to decide the schedule length[4,5,6]. Existing approaches try to use all given processors to minimize the schedule length, so that there is no way to decide the number of processors maximizing the degree of contribution for minimizing the schedule length per a processor. If such a method is realized, it can be said that processors in a heterogeneous distributed system can be utilized more effectively.

In this paper, we propose a method for processor assignment, assuming that each cluster size (the total execution time of tasks in the cluster) is over the lower bound [7] to limit the number of processors. The processor assignment proposed uses the indicative value (detailed in the latter section) for the schedule length and then selects the processor by which the indicative value can be minimized. Then experimental results by simulations are presented in terms of the validity of the processor assignment.

This paper is organized as follows. Sec. 2 shows related works involving the field of cluster generation and processor assignment. Sec. 3 shows the assumed model, i.e., the application to be executed, the system environment. Sec. 4 shows definitions and characteristics of the indicative value for the schedule length. Then we briefly state how to decide the lower bound for each cluster size proposed in [7]. Sec. 5 shows the processor assignment as our proposal in this paper. Sec. 6 shows the overall procedure for the processor assignment. Experimental results are shown in Sec. 7, and finally conclusions and future works are presented in Sec. 8.

- *Hidehiro Kanemitsu, Graduate School of Global Information and Telecommunication Studies, Waseda University, Japan.*
- *Gilhyon Lee, Graduate School of Global Information and Telecommunication Studies, Waseda University, Japan.*
- *Hidenori Nakazato, Global Information and Telecommunication Institute, Waseda University, Japan.*
- *Takashige Hoshiai, Global Information and Telecommunication Institute, Waseda University, Japan.*
- *Yoshiyori Urano, Global Information and Telecommunication Institute, Waseda University, Japan.*

2 Related Works

If we try to perform a task clustering in a heterogeneous distributed system, the objective is to find an optimal processor assignment, i.e., which processor should be assigned to the cluster generated by a task clustering. Furthermore, since the processing time and the data communication time depend on each assigned processor's performance, each cluster should be generated with taking that issue into account. As related works for task clustering in heterogeneous distributed systems, CHP[4], Triplet[5], and FCS[6] have been known.

CHP[4] firstly assumes that "virtual identical processors", whose processing speed is the minimum among the given set of processors. Then CHP performs task clustering to generate a set of clusters. In the processor assignment phase, the cluster which can be scheduled in the earliest time is selected, while the processor which has possibility to make the cluster's completion time the earliest among other processors is selected. Then the cluster is assigned to the selected processor. Such a procedure is iterated until every cluster is assigned to a processor. In CHP algorithm, an unassigned processor can be selected as the next assignment target because it has no waiting time. Thus, each cluster is assigned to a different processor, so that many processors are required for execution and therefore CHP can not lead to the processor utilization.

In Triplet algorithm [5], task groups, each of which consists of three tasks, named as "triplet" according to the data size to be transferred among tasks and the out degree of each task. Then a cluster is generated by merging two triplets according to its execution time and data transfer time on the fastest processor and the slowest processor. On the other hand, each processor is grouped as a function of its processing speed and communication bandwidth, so that several processor groups are generated. As a final stage, each cluster is assigned to a processor groups according to the processor group's load. The processor assignment policy in Triplet is that one cluster is assigned to a processor groups composed of two or more processors. Thus, such a policy does not match with the concept of processor utilization.

In FCS algorithm [6], it defines two parameters, i.e., β: total task size to total data size ratio (where task size means that the time unit required to execute one instruction) for each cluster and τ: processing speed to communication bandwidth ratio for each processor. During task merging steps are performed, if β of a cluster exceeds τ of a processor, the cluster is assigned to the processor. As a result, the number of clusters depends on each processor's speed and communication bandwidth. Thus, there is one possibility that "very small cluster" is generated and then FCS can not match with the concept of processor utilization.

3 Assumed Model

3.1 Job Model

We assume a job to be executed among distributed processor elements (PEs) is a Directed Acyclic Graph (DAG), which is one of task graphs. Let be the DAG, $G^s_{cls} = (V_s, E_s, V^s_{cls})$, where s is the number of task merging steps (described in 3.2), V_s is the set of tasks after the s-th task merging step, E_s is the set of edges (data communications among tasks) after the s-th task merging step, and V^s_{cls} is the set of clusters which consists of one or more tasks after the s-th task merging step. 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. 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 $pred(n^s_i) = \varnothing$, n^s_i is called START task, and if $suc(n^s_i) = \varnothing$, 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 $n^s_i \prec n^s_j$.

3.2 System Model

We assume that each PE is completly connected to other PEs, with non-identical processing speeds and communication bandwidths The set of PEs is expressed as $P = \{P_1, P_2, P_3, \cdots, P_m\}$, and let the set of processing speeds as α, i.e., $\alpha = \{\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_m\}$. Let the set of communication bandwidths as β, i.e.,

$$\beta = \begin{pmatrix} \infty & \beta_{1,2} & \cdots & \beta_{1,m} \\ \beta_{2,1} & \infty & \cdots & \beta_{2,m} \\ \vdots & \vdots & \vdots & \vdots \\ \beta_{m,1} & \beta_{m,2} & \cdots & \infty \end{pmatrix}. \quad (1)$$

$\beta_{i,j}$ means the communication bandwidth from P_i to P_j. The processing time in the case that n^s_k is processed on P_i is defined as $t_p(n^s_k, \alpha_i) = w(n^s_k)/\alpha_i$. The data transfer time of $e^s_{k,l}$ over $\beta_{i,j}$ is defined as $t_c(e^s_{i,j}, \beta_{k,l}) = c(e^s_{i,j})/\beta_{k,l}$. This means that both processing time and data transfer time are not changed with the time, and suppose that data transfer time within one PE is negligible.

3.2 Definitions of a Cluster and 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, this means that they are assigned to the same processor. Then the communication between n^s_i and n^s_j is localized, so that $c(e^s_{i,j})$ becomes zero. Task clustering is a set of

task merging steps, that is finished when a certain criteria is satisfied.

Throughout this chapter, we denote that $cls_s(i)$ is "linear" if and only if $cls_s(i)$ contains no independent task [8]. Note that if one cluster is linear, at least one path among any two tasks in the cluster exists and task execution order is unique.

4 Determination of The Lower Bound for each Cluster Size

4.1 Basic Concept

Our basic idea behind our research is to impose a certain constraint to each cluster size in order to limit the number of required processors for processor utilization. Then the lower bound is decided before cluster generation. To do this, an indicative value having effect on the schedule length should be derived and its minimization should lead to the minimization of the schedule length under the constraint imposed to each cluster. The assumed procedures for achieving processor utilization are as follows. (i) define the indicative value for the schedule length, which can be derived before task scheduling, (ii) derive the lower bound for each cluster size as a function of the processing speed and the communication bandwidth, (iii) select the processor which can minimize the indicative value, and finally (iv) generate a cluster for the selected processor by merging several tasks. In this section, we briefly present (i) and (ii), which are detailed in [7].

4.2 The Indicative Value of the Schedule Length

The schedule length depends on many factors, i.e., the execution time for each task, the communication time for each data exchanged among tasks, the execution order after the task scheduling, the processing speed, and the communication bandwidth. Furthermore, whether a data transfer time can be localized or not depends on the cluster structure. The proposed method is that a cluster is generated after the lower bound for each cluster size (the total execution time of every task included in the cluster) has been derived. The lower bound is decided when the indicative value for the schedule length is minimized. In this paper, the indicative value is defined as $sl_w(G^s_{cls}, \phi_s)$, that means the indicative value for the schedule length after the s-th task merging step and ϕ_s is the set of mapping between PEs and clusters after the s-th task merging step. $sl_w(G^s_{cls}, \phi_s)$ is the maximum value of the execution path length which includes both the task execution time and the data transfer time, provided that each task is scheduled as late as possible and every data from its immediate predecessors has been arrived before the scheduled time (its start time). For more details about $sl_w(G^s_{cls}, \phi_s)$, refer to the literatures[7,9].

As for characteristics of $sl_w(G^s_{cls}, \phi_s)$, the following two theorems are proved in [7,9]. As stated in those theorems, ϕ_0 corresponds to the mapping state where every task is assigned to "a virtual processor" with the maximum processing speed in α and the maximum communication bandwidth in β.

Theorem 1. *In a heterogeneous distributed system, let the DAG after the s-th task merging step as G^s_{cls}. And assume every cluster in V^s_{cls} is assigned to a processor in P. Let the schedule length after the s-th task merging step as $sl(G^s_{cls}, \varphi_s)$. If we define $\Delta sl^{s-1}_{w,up}$ that satisfies $sl_w(G^s_{cls}, \varphi_s) - cp(\varphi_0) \le \Delta sl^{s-1}_{w,up}$ and is decided after (s-1)-th task merging step (where $cp(\varphi_0)$ is the critical path length at the mapping state ϕ_0.), the following relationship is hold.*

$$sl(G^s_{cls}, \varphi_s) \ge \frac{sl_w(G^s_{cls}, \varphi_s) - \Delta sl^{s-1}_{w,up}}{1 + \frac{1}{g_{min}(\varphi_0)}}, \quad (2)$$

where $g_{min}(\varphi_0)$ is the minimum task granularity[8] at the mapping state φ_0. □

Theorem 2. *In a heterogeneous distributed system, if and only if $sl(G^s_{cls}, \varphi_s) \le cp(\varphi_0) = sl(G^0_{cls}, \varphi_0)$, it holds that minimizing $sl_w(G^s_{cls}, \varphi_s)$ leads to minimization of the upper bound of $sl(G^s_{cls}, \varphi_s)$.* □

From theorem 1 and 2, it can be said that minimizing $sl_w(G^s_{cls}, \phi_s)$ can leads to reduction of the schedule length. Since $sl_w(G^s_{cls}, \phi_s)$ is derived before a task scheduling, i.e., it can be derived just after the s-th task merging step.

4.3 The Lower Bound for each Cluster Size

Both a task execution time and a data transfer time depend on the assigned processor's processing speed and communication bandwidth. Thus, the lower bound for each cluster size should be decided according to the assigned processor. In the literature[7], the lower bound for P_p i.e., $\delta^s_{opt}(P_p)$, is derived before the s-th task merging step. However, $sl_w(G^s_{cls}, \phi_s)$ is not decided before the s-th task merging step, then $\delta^s_{opt}(P_p)$ is derived by using the upper bound of $sl_w(G^s_{cls}, \phi_s) - cp(\phi_0)$ as $\Delta sl^{s-1}_{w,up}$, which means the upper bound in the increase of the indicative value from the initial state to just after the s-th task merging step (Thus, $\Delta sl^{s-1}_{w,up}$ is decided after (s-1) –th task merging step. $\Delta sl^{s-1}_{w,up}$ is defined as follow [7].

$$\Delta sl^{s-1}_{w,up} = \frac{\max_{n^0_k \in V_0}\{w(n^0_k)\}}{\alpha_p}(N_S - y) + \frac{\max_{e^0_{k,l} \in E_0}\{c(e^s_{k,l})\}}{\beta(p)}(N_S - 1) + y\left(\delta + \frac{\max_{n^0_k \in V_0}\{w(n^0_k)\}}{\alpha_p}\right) - \min_{p \in G^s_{ps}}\{len(p, \phi_0)\}, \quad (3)$$

where N_S is the number of clusters in a path, y is the number of non-linear clusters, δ is the lower bound assumed as a variable, $\min_{p \in G^s_{ps}}\{len(p, \phi_0)\}$ is the minimum path length at the mapping state ϕ_0, and $\beta(p)$ is the minimum communication bandwidth in β. Obviously, $\Delta sl^{s-1}_{w,up}$ becomes larger with the increase of y. Further, y is not decided until the s-th task merging step is finished. Thus, y should be set by assuming that $\Delta sl^{s-1}_{w,up}$ takes the small

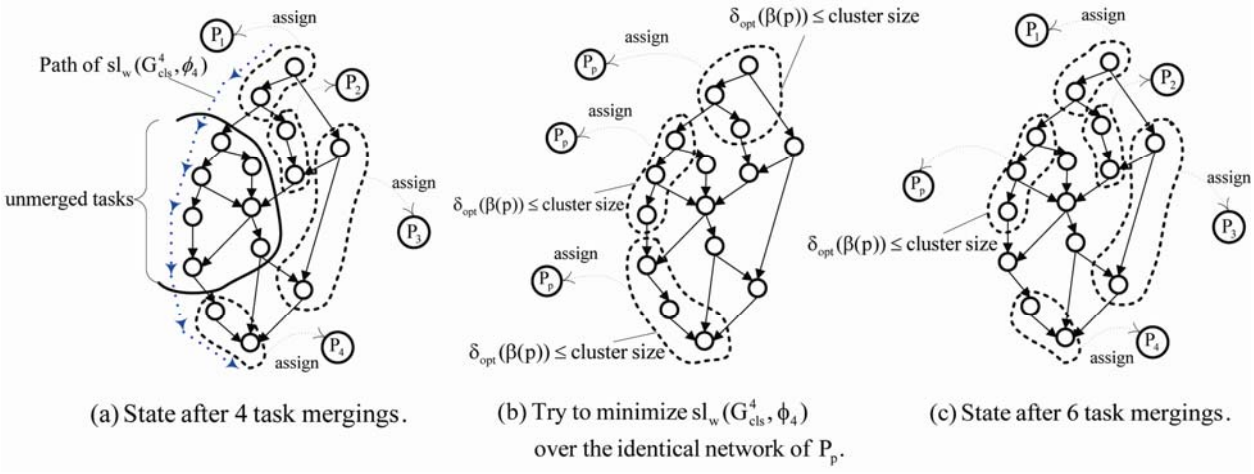

Fig. 1. Example of . $\delta^s_{opt}(P_p)$ derivation (where s=4)

value. At the s-th task merging step, one heuristic for minimizing $sl_w(G^s_{cls}, \phi_s)$ is to merge several tasks dominating $sl_w(G^{s-1}_{cls}, \phi_{s-1})$ over the identical processors.

Figure 1 shows one example for such situation, where let s=5. (a) corresponds to the state after the 4-th (i.e., (s-1)-th) task merging step. Four clusters have been generated with their cluster sizes exceeding each lower bound. Further, those clusters have already been assigned to each processor. In (b), it is assumed that all tasks in the path dominating $sl_w(G^4_{cls}, \phi_4)$ (i.e., $sl_w(G^{s-1}_{cls}, \phi_{s-1})$) are merged into several clusters. At the same time, every cluster size is assumed to be $\delta^5_{opt}(\alpha_p, \beta(p))$ ore more over the identical processors. This means that in the set of tasks denoted as "Path of $sl_w(G^4_{cls}, \phi_4)$ at (a)", we try to find the lower bound for every cluster and the number of those clusters with virtually assuming the identical processors. The number of clusters in such path is defined as follow.

$$\left\lfloor \frac{\sum_{n^{s-1}_k \in seq^{\prec}_{s-1}} w(n^{s-1}_k)}{\alpha_p \delta} \right\rfloor \le \frac{\sum_{n^{s-1}_k \in seq^{\prec}_{s-1}} w(n^{s-1}_k)}{\alpha_p \delta} = N_s, \quad (4)$$

where $seq^{\prec}_{s-1}$ is the path dominating $sl_w(G^{s-1}_{cls}, \phi_{s-1})$. Then by assigning (4) to (3), we obtain

$$\Delta sl^{s-1}_{w,up} = \left(\frac{\max_{n^0_k \in V_0}\{w(n^0_k)\}}{\alpha_p} + \frac{\max_{e^0_{k,l} \in E_0}\{c(e^0_{k,l})\}}{\beta(p)} \right) \frac{\sum_{n^{s-1}_k \in seq^{\prec}_{s-1}} w(n^{s-1}_k)}{\delta \alpha_p} + y\delta - \frac{\max_{e^0_{k,l} \in E_0}\{c(e^S_{k,l})\}}{\max_{\beta_{p,q} \in \beta}\{\beta_{p,q}\}} - \min_{p \in G^0_{cls}}\{len(p, \phi_0)\}. \quad (5)$$

If $\Delta sl^{s-1}_{w,up}$ is differentiated with respect to δ , it can be seen that $\Delta sl^{s-1}_{w,up}$ takes the local minimum value when δ is $\delta^s_{opt}(P_p)$, provided that y = 1 [7] as follows.

$$\delta^s_{opt}(P_p) = \sqrt{\frac{\sum_{n^{s-1}_k \in seq^{\prec}_{s-1}} w(n^{s-1}_k)}{\alpha_p} \left(\frac{\max_{n^0_k \in V_0}\{w(n^0_k)\}}{\alpha_p} + \frac{\max_{e^0_{k,l} \in E_0}\{c(e^0_{k,l})\}}{\beta(p)} \right)}. \quad (6)$$

(6) means the lower bound for the processor P_p. If P_p has been selected before deriving (5), the actual lower bound can be decided. Otherwise, it is necessary to decide both α_p and $\beta(p)$ by which $\Delta sl^{s-1}_{w,up}$ can be minimized.

5 Processor Assignment

5.1 Decision of the next Processor

According to (5), since $\delta^s_{opt}(P_p)$ is varied as a function of α_p and $\beta(p)$, it is necessary to decide the processor by assigning $\delta^s_{opt}(P_p)$ to $\Delta sl^{s-1}_{w,up}$ and to find the best combination of the processing speed and the communication bandwidth. In this case, let the assignment as $\Delta sl^{s-1}_{w,up}(\delta^s_{opt}(\alpha_p, \beta(p)))$ as follow.

$$\Delta sl^{s-1}_{w,up}(\delta^s_{opt}(\alpha_p, \beta(p))) = \frac{2}{\alpha_p} \sqrt{\sum_{n^{s-1}_k \in seq^{\prec}_{s-1}} w(n^{s-1}_k) \left(\frac{\alpha_p \max_{e^0_{k,l} \in E_0}\{c(e^0_{k,l})\}}{\beta(p)} + \max_{n^0_k \in V_0}\{w(n^0_k)\} \right)} - \frac{\max_{e^0_{k,l} \in E_0}\{c(e^0_{k,l})\}}{\max_{\beta_{p,q} \in \beta}\{\beta_{p,q}\}} - \min_p\{len(p, \phi_0)\}. \quad (7)$$

From (7), the larger both α_p and $\beta(p)$ become, the smaller $\Delta sl^{s-1}_{w,up}(\delta^s_{opt}(\alpha_p, \beta(p)))$ becomes. Therefore, the next processor to be selected can be decided by comparing for each $\Delta sl^{s-1}_{w,up}(\delta^s_{opt}(\alpha_p, \beta(p)))$, then the processor which has minimum value of $\Delta sl^{s-1}_{w,up}(\delta^s_{opt}(\alpha_p, \beta(p)))$ should be selected as the next assignment target.

5.2 Characteristics of the Next Processor

In this section we analyze characteristics of the processor to be assigned. We present relationship between the processing speed and the communication bandwidth of the processor. Then by expressing $\beta(p)/\alpha_p$ as k, $\Delta sl^{s-1}_{w,up}(\delta^s_{opt}(\alpha_p, \beta(p)))$ is defined as

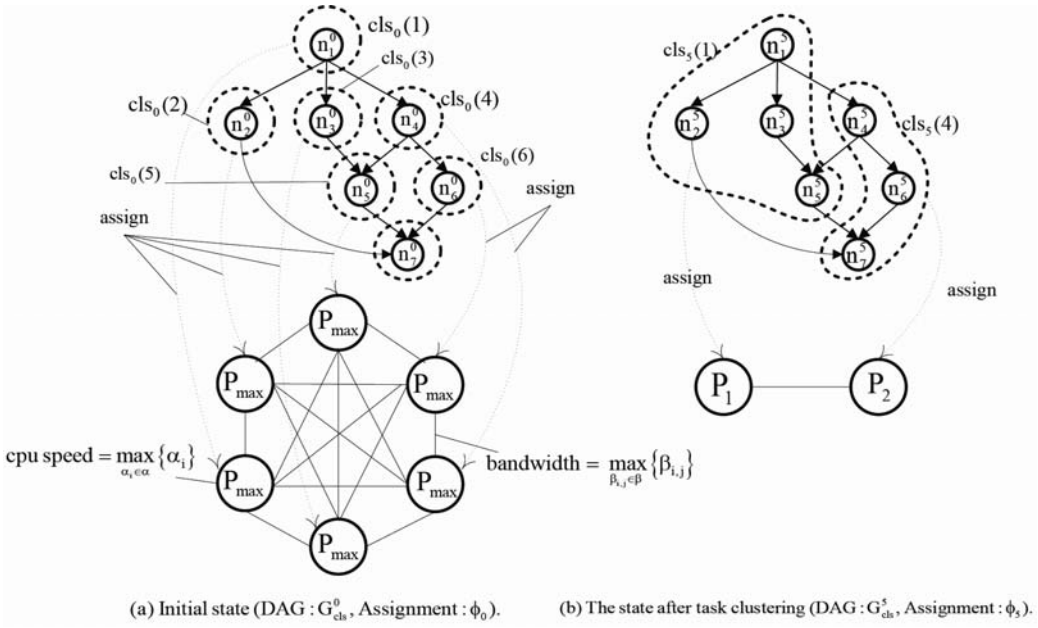

Fig. 2. Example of the Mapping ϕ_0

$$\Delta sl_{w,up}^{s-1}(\delta_{opt}^{s}(\alpha_p, k\alpha_p)) = \frac{2}{\alpha_p}\sqrt{\sum_{n_k^{s-1}\in seq_{s-1}^{<}} w(n_k^{s-1})\left(\frac{\max_{e_{k,l}^0\in E_0}\{c(e_{k,l}^0)\}}{k} + \max_{n_k^0\in V_0}\{w(n_k^0)\}\right)} - \frac{\max_{e_{k,l}^0\in E_0}\{c(e_{k,l}^0)\}}{\max_{\beta_{p,q}\in\beta}\{\beta_{p,q}\}} - \min_p\{len(p,\phi_0)\}. \quad (8)$$

From (8), obviously k should be large in order to minimize $\Delta sl_{w,up}^{s-1}(\delta_{opt}^{s}(\alpha_p, k\alpha_p))$ under the condition that α_p is not varied. Thus, the variation of β(p) has more critical impact on $\Delta sl_{w,up}^{s-1}$ than that of α_p. Therefore, even if a processor's processing speed is not so high, its communication bandwidth can precede for selecting the next processor. This characteristic is consistent with the fact that communications among processors can be a bottleneck in the schedule length in a distributed system. Generally, the data transfer time between processors has larger impact on the schedule length than the processing time, especially for data-intensive DAG.

TABLE 1
OVERALL PROCEDURE FOR PROCESSOR ASSIGNMENT

Procedures for Processor Assignment

INPUT : G_{cls}^0
OUTPUT : G_{cls}^R
Set the mapping state ϕ_0 (each task is "single cluster");
Define UEX_s as the set of unmerged clusters;
Define RDY_s as the set of candidates as $pivot_s$;
$s \leftarrow 0$;
1: **WHILE** $UEX_s \neq \varnothing$ **DO**
2: Derive $\delta_{opt}^{s+1}(P_p)$ by (6);
3: Find P_p s.t., $\Delta sl_{w,up}^{s-1}(\delta_{opt}^{s}(P_p)) = \min_{P_t \in \text{unassigned PEs}}\{\Delta sl_{w,up}^{s-1}(\delta_{opt}^{s}(P_t))\}$;
4: $pivot_s \leftarrow$ getPivot(RDY_s);
5: **WHILE** size of $pivot_s < \delta_{opt}^{s+1}(P_p)$ **DO**
6: $target_s \leftarrow$ getTarget($pivot_s$);
7: $pivot_{s+1} \leftarrow$ merge($pivot_s$, $target_s$); and update RDY_s as RDY_{s+1};
8: $s \leftarrow s+1$;
9: **ENDWHILE**
10: Assign P_p to $pivot_s$;
11: **ENDWHILE**

6 OVERALL PROCEDURES

6.1 Overview

The overall procedure for processor assignment consists of three phases: (i) derive the lower bound for the cluster size as (6), (ii) decide the processor to be assigned, which minimize (7). Then (iii) merge several task into a cluster until its size exceeds the lower bound derived in (i). Table 1 shows the whole procedures including those three phases.

At first, the mapping state ϕ_0 is applied to the input DAG, G_{cls}^0, where every task is virtually assigned to a processor P_{max} having the maximum processing speed and the maximum communication bandwidth. Figure 2 shows one example of ϕ_0. In this figure, (a) corresponds to the initial state, i.e., ϕ_0. At (a), every task belongs to a single cluster which contains only one task. It is assumed that each cluster is assigned to P_{max} in order to derive the schedule length at the initial state. (b) means the state after 5 task merging steps have been finished, where two actual clusters have been generated and they are assigned to actual processors, respectively.

Objectives of the procedures presented in table 1 are as

follows.

A. make every cluster size exceeds the lower bound
B. Minimize $sl_w(G^R_{cls})$, where R is the number of task mergings taken to satisfy the objective A.

Our approach to satisfy the objective A is to check whether every cluster size exceeds the lower bound or not. Thus, UEX_s, which is the set of clusters whose size is under the lower bound, is defined. If every cluster size exceeds the lower bound, the procedure is finished.

As for the objective B, it complies with theorem 1 and 2, i.e., minimizing $sl_w(G^R_{cls})$ can leads to minimizing the schedule length after R task merging steps. To achieve this objective, our approach is to select two clusters for each task merging by which $sl_w(G^R_{cls})$ is minimized. Thus, At a task merging step (let the step as s-th), $sl_w(G^s_{cls})$ should be minimized. In the procedure in table 1, RDY_s, which contains the set of clusters which may be selected for the (s+1)-th task merging step.

6.2 Processor Selection Phase

Before the cluter generation, the processor to be assignment should be selected. Thus, at line 2 $\delta^{s+1}_{opt}(P_p)$ is defined before the (s+1)-th task merging step. Since $\delta^{s+1}_{opt}(P_p)$ is a function of the processing speed and the communication bandwidth, for each unassigned processor its α_p and $\beta(p)$ are assigned to $\Delta sl^{s-1}_{w,up}(\delta^s_{opt}(\alpha_p,\beta(p)))$ in order to decide the next processor to be assigned. If the processor which minimizes $\Delta sl^{s-1}_{w,up}(\delta^s_{opt}(\alpha_p,\beta(p)))$ is found, it is selected as P_p at line 3.

6.3 Cluster Selection Phase

Since the procedure in this phase complies with the algorithm presented in [9,10], in this paper we present only abstract. At line 4 in table 1, before (s+1)-th task merging step, the cluster dominating $sl_w(G^s_{cls})$ is selected as $pivot_s$ from RDY_s. This is based on the heuristic that reducing $sl_w(G^s_{cls})$ can leads to the reduction of $sl_w(G^{s+1}_{cls})$. Therefore, once $pivot_s$ is selected, the other cluster merged with $pivot_s$ should dominate $sl_w(G^{s+1}_{cls})$. Such a cluster is named as $target_s$. At line 5-9, this loop is proceeded until the size of $pivot_s < \delta^s_{opt}(P_p)$. During the loop, at first $target_s$ is selected and then $pivot_s$ and $target_s$ are merged into the new cluster, i.e., $pivot_{s+1}$ (line 7). In this merging, every task in $target_s$ comes to belong to $pivot_{s+1}$. Then RDY_s is updated to become RDY_{s+1}, which means that the new clusters for the next task merging belongs to RDY_{s+1}, and both $pivot_{s+1}$ and $target_s$ are removed from RDY_{s+1} if they belong to RDY_{s+1}. If the size of $pivot_s$ exceeds $\delta^s_{opt}(P_p)$, the procedure returns to the line 1.

6.4 Processor Assignment Phase

After the next processor, P_p has been decided and the size of $pivot_s$ exceeds $\delta^{s+1}_{opt}(P_p)$, P_p is assigned to $pivot_s$. Then $pivot_s$ is removed from RDY_s. If no cluster exists in UEX_s, the procedure ends, and every task in every processor (cluster) is scheduled by a task scheduling policy to derive the schedule length. Since the proposal in this paper focuses on only the processor assignment, we do not mention about task scheduling policy. After the procedure at table 1 finished, any task scheduling policy can be applied, because every task has been assigned to a processor at a task scheduling such as list scheduling heuristics [11].

7 Experiments

7.1 Objective of the Experiments

We conducted the experimental simulation to confirm advantages of our proposal. Thus, we compared with other conventional methods in terms of the following points of view.

- Whether minimizing $sl_w(G^R_{cls})$ leads to minimsing the schedule length or not.

This point was confirmed with comparing the processor assignment policy, i.e., our proposal and conventional processor assignment [4].

7.2 Experimental Environment

In the simulation, a random DAG is generated. In the DAG, each task size and each data size are decided randomly. Also CCR (Communication to Computation Ratio) [12,13] 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 [14]. By using PF, the depth of the DAG is defined as $\sqrt{V_0}/\rho$.

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.

7.3 Comparison about $sl_w(G^R_{cls})$ and the Schedule Length

In this experiment, we generated 100 random DAGs and averaged each $sl_w(G^R_{cls})$ and the schedule length. Then we compared both averaged values among three approaches.

A. Our proposal in this paper, i.e., each processor is assigned according to the policy of table 1. Then the DAG is scheduled by a list scheduling [11].
B. Conventional processor assignment and changing the lower bound for each cluster size according to the assigned processor. In this case, processor, P_p, is assigned according to CHP[4], Then $\delta^s_{opt}(P_p)$ is derived. Finally, task clustering is performed until every cluster size exceeds $\delta^s_{opt}(P_p)$.
C. Conventional processor assignment by CHP and not changing the lower bound for each cluster size, i.e., every cluster has the same lower bound of its size.

Table 2 and 3 show comparison results in random DAG and FFT DAG among three approaches. In both tables, α and β mean max to min ration of the processing speed and communication bandwidth, respectively. The larger value means high degree of heterogeneity. $|V^R_{cls}|$ is the number of clusters in the case of A. To equalize the number in every approach, in B and C cluster merging steps are performed by "load balancing [15]". "$sl_w(G^R_{cls})$ ratio" corresponds to the $sl_w(G^R_{cls})$ ratio to A. Similary, "$sl(G^R_{cls})$ ratio" corresponds to the schedule length ration

TABLE 2
COMPARISON IN RANDOM DAGS (# OF TASKS:1000)

No.	α	β	CCR	$\lvert V_{cls}^R \rvert$	$sl_w(G_{cls}^R)$ ratio			$sl(G_{cls}^R)$ ratio		
					A	B	C	A	B	C
1	5	5	0.1	161	1.000	1.005	1.052	1.000	1.024	1.051
2			1.0	62	1.000	1.012	1.099	1.000	1.063	1.077
3			5.0	47	1.000	1.008	1.143	1.000	1.080	1.138
4			10.0	17	1.000	1.104	1.132	1.000	1.132	1.171
5	5	10	0.1	159	1.000	1.002	1.035	1.000	1.012	1.069
6			1.0	51	1.000	1.051	1.181	1.000	1.077	1.082
7			5.0	28	1.000	1.042	1.124	1.000	1.063	1.124
8			10.0	17	1.000	1.132	1.215	1.000	1.092	1.167
9	10	5	0.1	146	1.000	1.009	1.034	1.000	1.015	1.072
10			1.0	49	1.000	1.018	1.067	1.000	1.056	1.061
11			5.0	38	1.000	1.113	1.133	1.000	1.121	1.132
12			10.0	26	1.000	1.119	1.171	1.000	1.126	1.177
13	10	10	0.1	183	1.000	1.002	1.081	1.000	1.002	1.041
14			1.0	65	1.000	1.108	1.133	1.000	1.042	1.089
15			5.0	43	1.000	1.112	1.177	1.000	1.098	1.075
16			10.0	28	1.000	1.162	1.248	1.000	1.108	1.137

TABLE 3
COMPARISON IN FFT DAGS (# OF TASKS:2048)

No.	α	β	CCR	$\lvert V_{cls}^R \rvert$	$sl_w(G_{cls}^R)$ ratio			$sl(G_{cls}^R)$ ratio		
					A	B	C	A	B	C
1	5	5	0.1	277	1.000	1.009	1.077	1.000	1.003	1.023
2			1.0	131	1.000	1.007	1.156	1.000	1.032	1.041
3			5.0	61	1.000	1.018	1.241	1.000	1.044	1.077
4			10.0	36	1.000	1.054	1.351	1.000	1.071	1.109
5	5	10	0.1	273	1.000	1.008	1.062	1.000	1.005	1.021
6			1.0	129	1.000	1.132	1.142	1.000	1.021	1.058
7			5.0	51	1.000	1.073	1.122	1.000	1.077	1.104
8			10.0	29	1.000	1.149	1.281	1.000	1.074	1.122
9	10	5	0.1	252	1.000	1.003	1.221	1.000	1.002	1.049
10			1.0	118	1.000	1.098	1.091	1.000	1.039	1.081
11			5.0	45	1.000	1.117	1.155	1.000	1.091	1.133
12			10.0	30	1.000	1.083	1.182	1.000	1.104	1.176
13	10	10	0.1	290	1.000	1.006	1.032	1.000	1.008	1.003
14			1.0	135	1.000	1.031	1.139	1.000	1.092	1.051
15			5.0	60	1.000	1.136	1.228	1.000	1.092	1.098
16			10.0	35	1.000	1.122	1.243	1.000	1.109	1.127

to A. Thus, a value over 1 means worse than A.

In table 2, in every try, our proposal (A) has smaller value in $sl_w(G_{cls}^R)$ and $sl(G_{cls}^R)$ than other approaches. Although it is not found that the correlation in heterogeneity and the schedule length, the correlation in CCR is found. The larger CCR becomes, the worse both $sl_w(G_{cls}^R)$ and $sl(G_{cls}^R)$ of C become. This is because a DAG with high CCR requires huge data communication among processors. Thus there may be some small clusters despite they are assigned to processors having wide communication bandwidth. Therefore, at least the lower bound for each cluster size should be adjusted according to the assigned processor's capability. As for comparison between A and B, it can be said that proposed he processor assignment policy has better impact on the schedule length in a DAG with high CCR. In table 3, similar results as table 2 are obtained.

From comparison results in table 2 and 3, it is concluded that the proposed processor assignment is superior to a conventional processor assignment method.

7.3 Discussion

From theorem 1 and 2, it can theoretically be seen that an approach for minimizing $sl(G_{cls}^R)$ has good impact on the schedule length. However, since $\delta_{opt}^s(P_p)$ is the lower bound when the upper bound of $sl_w(G_{cls}^R)$ can be minimized, the lower bound for each cluster size is "near-optimal" value. Thus, it is necessary to confirm by experiments adventages of the combination of "$sl(G_{cls}^R)$ minimization" and "adjusting the lower bound according to the assigned processor's capability". From comparison results presented in table 1 and 2, superiority of our proposed processor assignment is found. The number of processors is limited by imposing the lower bound for each processor (cluster). Further, with taking heterogeneity of each processor into account, it is found that the lower bound for each processor should be adjusted to minimize the schedule length. This result was obtained in the literature [7]. The approach A is the combination of those two concepts. Thus, it is concluded that those two concepts

are necessary for achieving processor utilization.

8 Conclusions and Future Works

In this paper, we presented a processor assignment strategy for processor utilization. The number of processors is limited by imposing the lower bound for each cluster size. Under the constraint, we theoretically showed that which processor should be assigned to the cluster (assignment unit). The processor to be assigned is the one which has good impact on minimizing indicative value for the schedule length. From comparison results by experimental simulations, the proposed processor assignment was confirmed to be superior to other method.

As a future work, we will study on more realistic situation, e.g., how to achieve processor utilization in heterogeneous distributed systems with limited communication ports, various network topologies.

References

[1] 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.

[2] T. Yang and A. Gerasoulis., DSC: Scheduling Parallel Tasks on an Unbounded Number of Processors, *IEEE Trans. on Parallel and Distributed Ssytems*, Vol. 5, No. 9 pp. 951-967, 1994.

[3] J. C. Liou, M. A. Palis., An Efficient Task Clustering Heuristic for Scheduling DAGs on Multiprocessors, *Procs. of the 8th Symposium on Parallel and Dis-tributed Processing*, October, 1996.

[4] C. Boeres, J. V. Filho and V. E. F. Rebello, ACluster-based Strategy for Scheduling Task on Heterogeneous Processors, *Procs. of the 16th Symposium on Computer Architecture and High Performance Computing (SBAC-PAD'04)*, pp.214 – 221,2004.

[5] B. Cirou, E. Jeannot, Triplet : a Clustering SchedulingAlgorithm for Heterogeneous Systems, *Procs. of 2001 International Conference on Parallel Processing Workshops (ICPPW'01)*, pp. 231 – 236, 2001.

[6] S. Chingchit, M. Kumar and L.N. Bhuyan, A FlexibleClustering and Scheduling Scheme for Efficient ParallelComputation, *Procs. of the 13th Internationaland 10th Symposium on Parallel and DistributedProcessing*, pp. 500 – 505, 1999.

[7] Hidehiro Kanemitsu, Gilhyon Lee, Hidenori Nakazato, Takashige Hoshiai, and Yoshiyori Urano, Static Assignment Unit Size Derivation for Utilization of Computational Resources in Heterogeneous Distributed Environments, *IEICE Trans. on Information and Systems* (Unplublished manuscript).

[8] A. Gerasoulis and T. Yang., On the Granularity and Clustering of Directed Acyclic Task Graphs, *IEEE Trans. on Parallel And Distributed Systems*, Vol. 4,No. 6, June, 1993.

[9] 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*, InTech, ISBN:979-953-307-540-1 (Unplublished manuscript).

[10] Hidehiro Kanemitsu, Gilhyon Lee, Hidenori Nakazato, Takashige Hoshiai, and Yoshiyori Urano, A Task Clustering for Utilizing Computational Resources, *IPSJ Trans. on Advanced Computing Systems*, Vol. 4, No. 1, pp. 111-146, 2011.

[11] T. Yang and A. Gerasoulis., List scheduling with andwithout communication delays, *Parallel Computing*, pp. 1321 – 1344, 1993.

[12] O. Sinnen., *Task Scheduling for Parallel Systems*, Wiley, 2007.

[13] O. Sinnen and L. A. Sousa., Communication Contentionin Task Scheduling, *IEEE Trans. on Parallel and Distributed Systems*, Vol. 16, No. 6., pp.503-515, 2005.

[14] H. Topcuoglu, S. Hariri, M. Y. Wu., Performance-Effective and Low-Complexity TaskScheduling for Heterogeneous Computing, *IEEE Trans. on Parallel and DistributedSystems*, Vol. 13, No. 3 pp. 260-274, 2002.

[15] J. C. Liou and M. A. Palis., A Comparison of General Approaches to Multiprocessor Scheduling, *Procs. of the 11th International Symposium on Parallel Processing*, pp. 152 – 156, 1997.

Hidehiro Kanemitsu received the B.S. from Waseda University, Japan in Science, and the M.S. degree 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 a Ph.D candidate at Graduate School of Global Information and Telecommunication Studies, Waseda University, Japan.

Gilhyon Lee received the B.S. from Chosun University, Computer Science and Statistics, and the M.S. degree from Hanyang University, Korea in Department of Electronics and Computer Engineering. His research interests are in areas of parallel and distributed computing, grid, peer-to-peer computing, and network coding. He is currently a Ph.D candidate at Graduate School of Global Information and Telecommunication Studies, Waseda University, Japan.

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.

Takashige Hoshiai is a senior research scientist supervisor at NTT Network Service Systems Laboratories, in Japan. He holds a 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.

Yoshiyori Urano received B.E., M.E. and Doctor of Engineering from Waseda University, in 1960, 1962 and 1965 respectively. Dr. Urano joined KDD (now KDDI) in 1965, served as Director of KDD Research and Development Laboratories in 1993-1996, and moved to Waseda University in 1996. He is now a Professor, Graduate School of Global Information and Telecommunication Studies, Waseda University. His current interests include Next-Generation Internet, Networked Applications (e-Learning/Ubiquitous Learning/Distance Education, e-Healthcare, Disaster Prevention Information System, etc.).