View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by IntechOpen

We are IntechOpen,
the world's leading publisher of
Open Access books
Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

154
Countries delivered to

Our authors are among the
TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS
BOOK CITATION INDEX
INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

2

On the Effect of Applying the Task Clustering for Identical Processor Utilization to Heterogeneous Systems

Hidehiro Kanemitsu[1], Gilhyon Lee[1], Hidenori Nakazato[2], Takashige Hoshiai[2] and Yoshiyori Urano[2]

[1]*Graduate School of Global Information and Telecommunication Studies, Waseda University*

[2]*Global Information and Telecommunication Institute, Waseda University*

Japan

1. Introduction

Actual task execution models over the networked processors, e.g., cluster, grid and utility computing have been studied and developed for maximizing the system throughput by utilizing computational resources. One of major trends in task execution types is to divide the required data into several pieces and then distribute them to workers like "master-worker model". In contrast to such a data intensive job, how to divide a computational intensive job into several execution units for parallel execution is under discussion from theoretical points of view. If we take task parallelization into account in a grid environment such as a computational grid environment, an effective task scheduling strategy should be established. In the light of combining task scheduling concepts and grid computing methodologies, heterogeneity with respect to processing power, communication bandwidth and so on should be incorporated into a task scheduling strategy. If we assume the situation where multiple jobs are being submitted in the unknown number of computational resources over the Internet, objective functions can be considered as follows: (i) Minimization of the schedule length (the time duration from per each job, (ii) Minimization of the completion time of the last job, (iii) Maxmization of the degree of contribution to the total speed up ratio for each computational resources. As one solution for those three objective functions, in the literature(Kanemitsu, 2010) we proposed a method for minimizing the schedule length per one job with a small number of computational resources (processors) for a set of identical processors. The objective of the method is “utilization of computational resources”. The method is based on “task clustering” (A. Gerasoulis, 1992), in which tasks are merged into one “cluster” as an execution unit for one processor. As a result, several clusters are generated and then each of which becomes one assignment unit. The method proposes to impose the lower bound for every cluster size to limit the number of processors. Then the literature theoretically showed the near-optimal lower bound to minimize the schedule length.

However, which processor should be assigned to a cluster is not discussed because the proposal assumes identical processors. If we use one of conventional cluster assignment methods such as CHP(C. Boeres, 2004), triplet(B. Cirou, 2001), and FCS(S. Chingchit, 1999), almost all processors may be assigned to clusters because they try to achieve the maximum task parallelism to obtain the minimized schedule length. Thus, the third objective function may not be achieved by those cluster assignment strategies.

In this chapter, we propose a method for deriving the lower bound of the cluster size in heterogeneous distributed systems and a task clustering algorithm. From results of experimental simulations, we discuss the applicability of the proposal to obtain better processor utilization.

The remainder of this chapter is organized as follows. Sec. 2 presents other conventional approaches related to task clustering for heterogeneous distributed systems, and sec. 3 presents our assumed model, then the lower bound of the cluster size is derived in sec. 4. Sec. 5 presents a task clustering algorithm which adopts the lower bound shown in sec. 4. Experimental results are shown in sec. 6, and finally we present conclusion and future works in sec. 7.

2. Related works

In a distributed environment, where each processor is completely connected, task clustering(A. Gerasoulis, 1992; T. Yang, 1994; J. C. Liou, 1996) has been known as one of task scheduling methods. In a task clustering, two or more tasks are merged into one cluster by which communication among them is localized, so that each cluster becomes one assignment unit to a processor. As a result, the number of clusters becomes that of required processors. On the other hand, 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(C. Boeres, 2004), Triplet(B. Cirou, 2001), and FCS(S. Chingchit, 1999) have been known.

CHP(C. Boeres, 2004) 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 earliest time is selected, while the processor which has possibility to make the cluster's completion time 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 a next assignment target because it has no waiting time. Thus, each cluster is assigned to different processor, so that many processors are required for execution and therefore CHP can not lead to the processor utilization.

In Triplet algorithm(B. Cirou, 2001), task groups, each of which consists of three tasks, named as "triplet" according to data size to be transferred among tasks and 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 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(S. Chingchit, 1999), 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 $G^s_{cls} = (V_s, E_s, V^s_{cls})$ be the DAG, where s is the number of task merging steps(described in sec. 3.2), 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 $n^s_i \prec n^s_j$.

3.2 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 is a set of task merging steps, that is finished when certain criteria have been satisfied.

Throughout this chapter, we denote that $cls_s(i)$ is "linear" if and only if $cls_s(i)$ contains no independent task(A. Gerasoulis, 1993). 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.

3.3 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 $P = \{P_1, P_2, \ldots, P_m\}$,

Parameter	Definition
$top_s(i)$	$\{n_k^s \mid \forall n_l^s \in pred(n_k^s) s.t., n_l^s \notin cls_s(i)\} \cup \{START\ Tasks \in cls_s(i)\}.$
$in_s(i)$	$\{n_k^s \mid \exists n_l^s \in pred(n_k^s) s.t., n_l^s \notin cls_s(i)\} \cup \{START\ Tasks \in cls_s(i)\}.$
$out_s(i)$	$\{n_k^s \mid \exists n_l^s \in suc(n_k^s) s.t., n_l^s \notin cls_s(i)\} \cup \{END\ Tasks \in cls_s(i)\}.$
$btm_s(i)$	$\{n_k^s \mid \forall n_l^s \in suc(n_k^s), s.t., n_l^s \notin cls_s(i)\} \cup \{END\ Tasks \in cls_s(i)\}.$
$desc(n_k^s, i)$	$\{n_l^s \mid n_k^s \prec n_l^s, n_l^s \in cls_s(i)\} \cup \{n_k^s\}$
$S(n_k^s, i)$	$\sum_{n_l^s \in cls_s(i)} t_p(n_l^s, \alpha_p) - \sum_{n_l^s \in desc(n_k^s, i)} t_p(n_l^s, \alpha_p)$
$tlevel(n_k^s)$	$\begin{cases} \max\limits_{n_l^s \in pred(n_k^s)} \{tlevel(n_l^s) + t_p(n_l^s, \alpha_p) + t_c(e_{l,k}, \beta_{q,p})\}, \ if\ n_k^s \in top_s(i). \\ TL_s(i) + S(n_k^s, i),\ otherwise. \end{cases}$
$TL_s(i)$	$\max\limits_{n_k^s \in top_s(i)} \{tlevel(n_k^s)\}$
$blevel(n_k^s)$	$\max\limits_{n_l^s \in suc(n_k^s)} \left\{t_p(n_k^s, \alpha_p) + t_c(e_{k,l}^s, \beta_{p,q}) + blevel(n_l^s)\right\}$
$level(n_k^s)$	$tlevel(n_k^s) + blevel(n_k^s)$
$BL_s(i)$	$\max\limits_{n_k^s \in out_s(i)} \{S(n_k^s, i) + blevel(n_k^s)\}$
$LV_s(i)$	$TL_s(i) + BL_s(i) = \max\limits_{n_k^s \in cls_s(i)} \{level(n_k^s)\}$
ϕ_s	$\{\ldots, < cls_s(i), P_p >, \ldots\}$
$sl_w(G_{cls}^s, \phi_s)$	$\max\limits_{cls_s(i) \in V_{cls}^s} \{LV_s(i)\}$

Table 1. Parameter Definition Related to $sl_w(G_{cls}^s)$ (Here $n_k^s \in cls_s(i)$).

and let the set of processing speeds as $alpha$, i.e., $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_m\}$. Let the set of communication bandwidths as 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 communication bandwidth from P_i to P_j. The processing time in the case that n_k^s is processed on P_i is expressed as $t_p(n_k^s, \alpha_i) = w(n_k^s)/\alpha_i$. 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 PE is negligible.

4. Processor utilization

4.1 The indicative value for the schedule length

The schedule length depends on many factors, i.e., execution time for each task, communication time for each data exchanged among tasks, execution order after the task scheduling, processing speed, and 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 of the 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 chapter, the indicative value is defined as $sl_w(G^s_{cls}, \phi_s)$, that means the indicative value for the schedule length after s task merging steps and ϕ_s is the set of mapping between PEs and clusters after s task merging steps. $sl_w(G^s_{cls}, \phi_s)$ is the maximum value of the execution path length which includes both task execution time and 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). Table 1 shows notations and definitions for deriving $sl_w(G^s_{cls}, \phi_s)$. In the table, assigned PEs for $cls_s(i)$ and $cls_s(j)$ are P_p and P_q, respectively. And suppose $n^s_k \in cls_s(i), n^s_l \in cls_s(j)$. In table 1, especially $S(n^s_k, i)$ means the degree of increase of execution time by independent tasks for n^s_k. Threrfore, the smaller $S(n^s_k, i)$, the earlier n^s_k can be scheduled. The task n^s_k which dominates $sl_w(G^s_{cls}, \phi_s)$ (In the case of $sl_w(G^s_{cls}, \phi_s) = level(n^s_k)$) means that the schedule length may be maximized if n^s_k is scheduled as late as possible.

Example 1. *Fig. 1 shows one example for deriving $sl_w(G^s_{cls}, \phi_s)(s = 5)$. In the figure, there are two PEs, i.e., P_1 and P_2. The DAG has two clusters, i.e., $cls_5(1)$ and $cls_5(4)$ after 5 task merging steps. In (a), numerical values on tasks and edges mean the time unit to be processed on the reference PE and the time unit to be transferred among reference PEs on the reference communication bandwidth. On the other hand, (b) corresponds to the state that $cls_5(1)$ and $cls_5(4)$ have been assigned to P_1 and P_2, respectively. The bottom are shows the derivation process for $sl_w(G^5_{cls}, \phi_5)$. From the derivation process, it is shown that the schedule length may be maximized if n^5_2 is scheduled as late as possible.*

4.2 Relationship between $sl_w(G^s_{cls}, \phi_s)$ and the schedule length

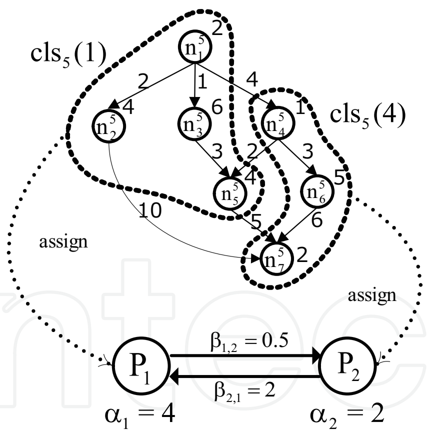
Our objective is to minimize $sl_w(G^s_{cls}, \phi_s)$ with maintaining the certain size of each cluster for processor utilization. The schedule length can not be known before scheduling every task, we must estimate it by using $sl_w(G^s_{cls}, \phi_s)$. Thus, it must be proved that $sl_w(G^s_{cls}, \phi_s)$ can effect on the schedule length. In this section, we show that minimizing $sl_w(G^s_{cls}, \phi_s)$ leads to minimizing the schedule length to some extent. In this section we present that relationship between Table 2 shows notations for showing characteristics of $sl_w(G^s_{cls}, \phi_s)$. In an identical processor system, provided that every processor speed and communication bandwidth are 1, no processor assignment policy is needed. Thus, let $sl_w(G^s_{cls}, \phi_s)$ in an identical processor system as $sl_w(G^s_{cls})$. In the literature (Kanemitsu, 2010), it is proved that minimizing $sl_w(G^s_{cls})$ leads to minimizing the lower bound of the schedule length as follows.

Lemma 1. *In an identical processor system, let $\Delta sl^{s-1}_{w,up}$ which satisfies $sl_w(G^s_{cls}) - cp \leq \Delta sl^{s-1}_{w,up}$ and be derived before s task merging steps. Then we obtain*

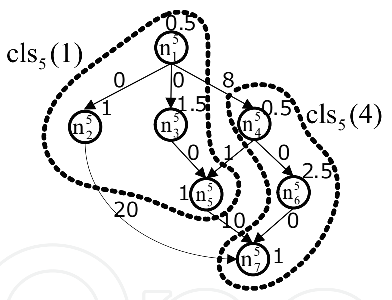
$$sl(G^s_{cls}) \geq \frac{sl_w(G^s_{cls}) - \Delta sl^{s-1}_{w,up}}{1 + \frac{1}{g_{min}}}, \tag{2}$$

where cp and g_{min} are defined in table 2, and $sl(G^s_{cls})$ is the schedule length after s task merging steps. ■

As for $\Delta sl^{s-1}_{w,up}$, it is defined in the literature (Kanemitsu, 2010). Furthermore, it can be proved that the upper bound of the schedule length can be reduced by reducing $sl_w(G^s_{cls})$ by the following lemma.



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



(b) After processor assignments have been done.

$\mathrm{top}_5(1) = \{n_1^5\}, \mathrm{top}_5(4) = \{n_4^5\},$

$\mathrm{in}_5(1) = \{n_1^5, n_5^5\}, \mathrm{in}_5(4) = \{n_4^5, n_7^5\}$

$\mathrm{out}_5(1) = \{n_1^5, n_2^5, n_5^5\}, \mathrm{out}_5(4) = \{n_4^5, n_7^5\}$

$\mathrm{btm}_5(1) = \{n_2^5, n_5^5\}, \mathrm{btm}_5(4) = \{n_7^5\}$

------ $cls_5(1)$ section------

$TL_5(1) = \mathrm{tlevel}(n_1^5) = 0,$

$\mathrm{tlevel}(n_2^5) = TL_5(1) + S(n_2^5, 1) = 0 + t_p(n_1^5, \alpha_1) + t_p(n_3^5, \alpha_1) + t_p(n_5^5, \alpha_1) = 3,$

$\mathrm{blevel}(n_2^5) = t_p(n_2^5) + t_c(e_{2,7}^5, \beta_{1,2}) + \mathrm{blevel}(n_7^5) = 1 + 20 + 1 = 22,$

$\mathrm{level}(n_2^5) = 25,$

$\mathrm{tlevel}(n_3^5) = TL_5(1) + S(n_3^5, 1) = 0 + t_p(n_1^5) + t_p(n_2^5) = 0.5 + 1 = 1.5,$

$\mathrm{blevel}(n_3^5) = t_p(n_3^5) + t_c(e_{3,5}^5, \beta_{1,2}) + \mathrm{blevel}(n_5^5) = 1.5 + 0 + 1 + 10 + 1 = 13.5,$

$\mathrm{level}(n_3^5) = 15,$

$S(n_1^5, 1) = 0,$

$$\begin{aligned}\mathrm{blevel}(n_1^5) &= t_p(n_1^5) + \max\{\mathrm{blevel}(n_2^5), \mathrm{blevel}(n_3^5), t_c(e_{1,4}^5, \beta_{1,2}) + \mathrm{blevel}(n_4^5)\} \\ &= 0.5 + \max\{22, 13.5, 8 + 13.5\} = 22.5 = \mathrm{level}(n_1^5),\end{aligned}$$

$\mathrm{tlevel}(n_5^5) = TL_5(1) + S(n_5^5, 1) = 0 + t_p(n_1^5) + t_p(n_2^5) + t_p(n_3^5) = 3,$

$\mathrm{blevel}(n_5^5) = 4 + 5 + 2 = 12,$

$\mathrm{level}(n_5^5) = 15,$

$$\begin{aligned}BL_5(1) &= \max_{n_k^5 \in \mathrm{out}_5(1)} \{S(n_k^5, 1) + \mathrm{blevel}(n_k^5)\} \\ &= S(n_2^5, 1) + \mathrm{blevel}(n_2^5) = 25,\end{aligned}$$

$LV_5(1) = TL_5(1) + BL_5(1) = \mathrm{level}(n_2^5) = 25.$

------ $cls_5(1)$ section END------

------ $cls_5(4)$ section------

$TL_5(4) = \mathrm{tlevel}(n_4) = 0.5 + 8 = 8.5,$

$\mathrm{tlevel}(n_6) = TL_5(4) + t_p(n_4^5) = 8.5 + 0.5 = 9,$

$$\begin{aligned}\mathrm{blevel}(n_4^5) &= t_p(n_4^5) + \max\{1 + \mathrm{blevel}(n_5^5), \mathrm{blevel}(n_6^5)\} \\ &= 0.5 + \max\{1 + 12, 3.5\} = 13.5\end{aligned}$$

$\mathrm{level}(n_4^5) = 22,$

$\mathrm{tlevel}(n_7^5) = TL_5(4) + t_p(n_4^5) + t_p(n_6^5) = 8.5 + 0.5 + 2.5 = 11.5,$

$\mathrm{blevel}(n_7^5) = 1,$

$\mathrm{level}(n_7^5) = 12.5,$

$$\begin{aligned}BL_5(4) &= \max_{n_k^5 \in \mathrm{out}_5(4)} \{S(n_k^5, 4) + \mathrm{blevel}(n_k^5)\} \\ &= S(n_4^5, 4) + \mathrm{blevel}(n_4^5) = 0 + 13.5 = 13.5,\end{aligned}$$

$LV_5(4) = TL_5(4) + BL_5(4) = \mathrm{level}(n_4^5) = 8.5 + 13.5 = 22.$

------ $cls_5(4)$ section END------

$sl_w(G_{cls}^5, \phi) = \max\{LV_5(1), LV_5(4)\} = LV_5(1) = \mathrm{level}(n_2^5) = 25.$

Fig. 1. Example of $sl_w(G_{cls}^5, \phi_5)$ derivation.

Lemma 2. *In an identical processor system, if $sl(G_{cls}^S) \leq cp$, then we obtain*

$$sl(G_{cls}^S) \leq sl_w(G_{cls}^S) + \max_{p \in G_{cls}^0} \left\{ \sum_{\substack{n_k^0, n_l^0 \in p, \\ n_k^0 \in pred(n_l^0)}} c(e_{k,l}^0) \right\}. \blacksquare \tag{3}$$

Proof. In $seq_s^{\prec}$, some edges are localized and others may be not localized. Furthermore, edges in $seq_s^{\prec}$ do not always belong to the critical path. Then we have the following relationship.

$$-\max_{p \in G^0_{cls}} \left\{ \sum_{\substack{n^0_k, n^0_l \in p, \\ n^0_k \in pred(n^0_l)}} c(e^0_{k,l}) \right\} \leq sl_w(G^S_{cls}) - cp. \tag{4}$$

Also, only in the case of $sl(G^S_{cls}) \leq cp$, we have the following rlationship.

$$\begin{aligned} & sl_w(G^S_{cls}) - cp \leq sl_w(G^S_{cls}) - sl(G^S_{cls}) \\ & \Leftrightarrow -\max_{p \in G^0_{cls}} \left\{ \sum_{\substack{n^0_k, n^0_l \in p, \\ n^0_k \in pred(n^0_l)}} c(e^0_{k,l}) \right\} \leq sl_w(G^S_{cls}) - sl(G^S_{cls}) \\ & \Leftrightarrow sl(G^S_{cls}) \leq sl_w(G^S_{cls}) + \max_{p \in G^0_{cls}} \left\{ \sum_{\substack{n^0_k, n^0_l \in p, \\ n^0_k \in pred(n^0_l)}} c(e^0_{k,l}) \right\}. \end{aligned} \tag{5}$$

□

From lemma 1 and 2, it is concluded that in an identical processor system the schedule length can be minimized if $sl_w(G_{cls})$ is minimized.

As a next step, we show the relationship between $sl_w(G^s_{cls}, \phi_s)$ and the schedule length in a heterogeneous distributed system. The following lemma is proved in the literature (Sinnen, 2007).

Lemma 3. In an identical processor system, we have

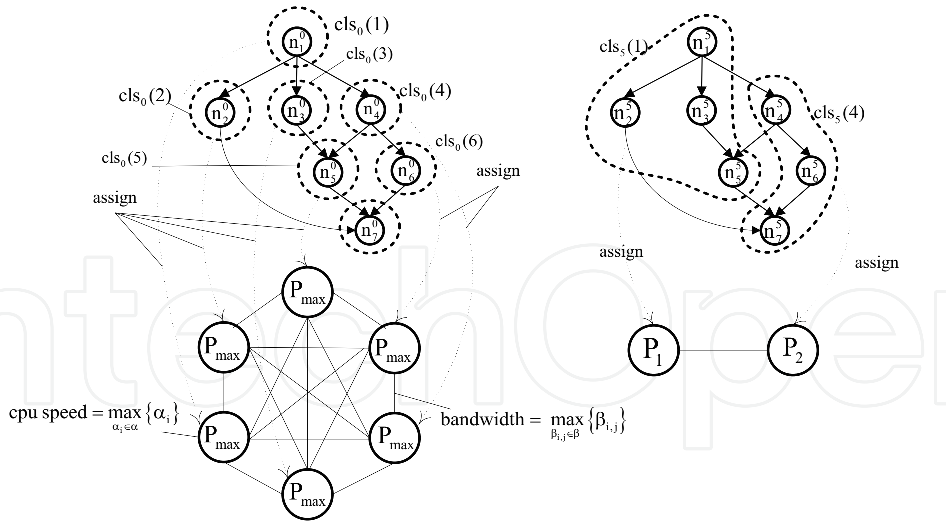
$$cp_w \leq sl(G^s_{cls}).\blacksquare \tag{6}$$

In a heterogeneous distributed system, we assume the state like fig. 2, i.e., at the initial state every task is assigned a processor with the fastest and the widest communication bandwidth (let the processor as P_{max}). In fig. 2 (a), each task belongs to respective processor. Furthermore, we virtually assign P_{max} to each task to decide the processing time for each task and the data transfer time among any two tasks. Let the mapping as ϕ_0. Under the situation, we have the following corollary.

Corollary 1. *In a heterogeneous distributed system, let $cp_w(\phi_0)$ as the one with the mapping ϕ_0 in the table 2. Then we have*

$$cp_w(\phi_0) \leq sl(G^s_{cls}, \phi_s).\blacksquare \tag{7}$$

As for the relationship between cp and cp_w, in the literature (Sinnen, 2007), the following is proved.



(a) Initial state (DAG : G^0_{cls}, Assignment : ϕ_0). (b) The state after task clustering (DAG : G^5_{cls}, Assignment : ϕ_5).

Fig. 2. Assumed condition during cluster generation procedures.

Parameter	Definition
p	One path of G^0_{cls}, i.e., $\{n^0_0, n^0_1, n^0_2, \ldots, n^0_k\} \cup \{e^0_{0,1}, e^0_{1,2}, \ldots e^0_{k-1,k}\}$, by which a sequence $< n^0_0, n^0_1, n^0_2, \ldots, n^0_k >$ is constructed, where $e^0_{l-1,l} \in E_0$, n^0_0 is a START task and n^0_k is an END task.
$seq^{\prec}_s$	One path in which every task belongs to seq_s.
$seq^{\prec}_s(i)$	Set of subpaths in each of which every task in $cls_s(i)$ belongs to $seq^{\prec}_s$.
$proc(n^s_k)$	The processor to which n^s_k has been assigned.
cp	$\max_p \left\{ \sum_{n^0_k \in p} w(n^s_k) + \sum_{e^0_{k,l} \in p} c(e^s_{k,l}) \right\}$.
$cp(\phi_s)$	$\max_p \left\{ \sum_{n^s_k \in p} t_p(n^s_k, \alpha_p) + \sum_{e^s_{k,l} \in p} t_c(c(e^s_{k,l}), \beta_{p,q}) \right\}$, where n^s_k, n^s_l are assigned to P_p, P_q.
cp_w	$\max_{p \in G^0_{cls}} \left\{ \sum_{n^0_k \in p} w(n^0_k) \right\}$.
$cp_w(\phi_s)$	$\max_{p \in G^s_{cls}} \left\{ \sum_{n^s_k \in p} t_p(n^s_k, \alpha_p) \right\}$
$g_{\min}$	$\min_{n^0_k \in V^0_{cls}} \left\{ \frac{\min_{n^0_j \in pred(n^0_k)} \{w(n^0_j)\}}{\max_{n^0_j \in pred(n^0_k)} \{c(e^0_{j,k})\}}, \frac{\min_{n^0_l \in suc(n^0_k)} \{w(n^0_l)\}}{\max_{n^0_l \in suc(n^0_k)} \{c(e^0_{k,l})\}} \right\}$.
$g_{\max}$	$\max_{n^0_k \in V^0_{cls}} \left\{ \frac{\max_{n^0_j \in pred(n^0_k)} \{w(n^0_j)\}}{\min_{n^0_j \in pred(n^0_k)} \{c(e^0_{j,k})\}}, \frac{\max_{n^0_l \in suc(n^0_k)} \{w(n^0_l)\}}{\min_{n^0_l \in suc(n^0_k)} \{c(e^0_{k,l})\}} \right\}$.

Table 2. Parameter Definitions which are used in analysis on $sl_w(G^s_{cls}, \phi_s)$.

Lemma 4. *In an identical processor system, by using g_{min} defined in table 2, we have*

$$cp \leq \left(1+\frac{1}{g_{min}}\right) cp_w. \blacksquare \tag{8}$$

By using lemma 4, in a heterogeneous distributed system, the following is derived.

Corollary 2. *In a heterogeneous distributed system, we have*

$$cp(\phi_s) \leq \left(1+\frac{1}{g_{min}(\phi_s)}\right) cp_w(\phi_s). \blacksquare \tag{9}$$

From corollary 1, the following is derived.

Corollary 3. *In a heterogeneous distributed system, we have*

$$cp_w(\phi_0) \leq cp_w(\phi_s) \leq sl(G^s_{cls}, \phi_s). \blacksquare \tag{10}$$

From corollary 2 and 3, the following theorem is derived.

Thorem 4.1. *In a heterogeneous distributed system, let the DAG after s task merging steps as G^s_{cls}. And assume every cluster in V^s_{cls} is assigned to a processor in P. Let the schedule length as $sl(G^s_{cls}, \phi_s)$. If we define $\Delta sl^{s-1}_{w,up}$ that satisfies $sl_w(G^s_{cls}, \phi_s) - cp(\phi_0) \leq \Delta sl^{s-1}_{w,up}$, the following relationship is derived.*

$$sl(G^s_{cls}, \phi_s) \geq \frac{sl_w(G^s_{cls}, \phi_s) - \Delta sl^{s-1}_{w,up}}{1+\frac{1}{g_{min}(\phi_0)}}. \tag{11}$$

Proof. From the assumption and corollary 2, we have

$$sl_w(G^s_{cls}, \phi_s) - \left(1+\frac{1}{g_{min}(\phi_0)}\right) cp_w(\phi_0) \leq \Delta sl^{s-1}_{w,up}. \tag{12}$$

Also, from corollary 3, we obtain $cp_w(\phi_0) \leq sl(G^s_{cls}, \phi_s)$. Thus if this is applied to (12), we have

$$sl_w(G^s_{cls}, \phi_s) - \left(1+\frac{1}{g_{min}(\phi_0)}\right) sl(G^s_{cls}, \phi_s) \leq \Delta sl^{s-1}_{w,up} \tag{13}$$

$$\Leftrightarrow \tag{14}$$

$$sl(G^s_{cls}, \phi_s) \geq \frac{sl_w(G^s_{cls}, \phi_s) - \Delta sl^{s-1}_{w,up}}{1+\frac{1}{g_{min}(\phi_0)}}. \tag{15}$$

□

Assume that $\Delta sl^{s-1}_{w,up}$ is the value which is decided after $s-1$ task merging steps. Since

$$cp(\phi_0) = sl_w(G^0_{cls}, \phi_0), \tag{16}$$

this value is an upper bound of increase in terms of $sl_w(G^s_{cls}, \phi_s)$ and can be defined in any policy, e.g., the slowest processor is assigned to each cluster and so on. However, at least $\Delta sl^{s-1}_{w,up}$ must be decided before s task merging steps. From the theorem, it can be said that reducing $sl_w(G^s_{cls}, \phi_s)$ leads to reduction of the lower bound of the schedule length in a heterogeneous distributed system.

As for the upper bound of the schedule length, the following theorem is derived.

Thorem 4.2. *In a heterogeneous distributed sytem, if and only if* $sl(G^s_{cls}, \phi_s) \le cp(\phi_0) = sl(G^0_{cls}, \phi_0)$*, we have*

$$sl(G^s_{cls}, \phi_s) \le sl_w(G^s_{cls}, \phi_s) + \zeta - \lambda - \mu, \tag{17}$$

where

$$\zeta = \max_p \left\{ \sum_{\substack{n^0_k, n^0_l \in p, \\ n^0_k \in pred(n^0_l), \\ t_c(e^s_{k,l}, \max_{\beta_{i,j}\in\beta}\{\beta_{i,j}\})=0}} t_c(e^0_{k,l}, \max_{\beta_{i,j}\in\beta}\left\{\beta_{i,j}\right\}) \right\}, \tag{18}$$

$$\lambda = \min_p \left\{ \sum_{\substack{n^0_k \in p, \\ proc(n^s_k)=p_m}} \left(t_p(n^s_k, \alpha_m) - t_p(n^0_k, \max_{\alpha_i \in \alpha}\{\alpha_i\}) \right) \right\}, \tag{19}$$

$$\mu = \min_p \left\{ \sum_{\substack{n^0_k, n^0_l \in p, proc(n^s_k)=P_i, \\ proc(n^s_l)=P_j, \\ n^0_k \in pred(n^0_l)}} \left(t_c(e^s_{k,l}, \beta_{i,j}) - t_c(e^0_{k,l}, \max_{\beta_{i,j}\in\beta}\left\{\beta_{i,j}\right\}) \right) \right\}. \tag{20}$$

p and $proc(n^0_k)$ *are defined in table 2. That is,* ζ, λ, μ *is derived by scanning every path in the DAG.*

Proof. After s task merging steps, there may be both localized edges and not localized edges which compose $sl_w(G^s_{cls}, \phi_s)$. Obviously, we have $sl_w(G^0_{cls}, \phi_0) = cp(\phi_0)$, such edges are not always ones which belongs to $cp(\phi_0)$. Therefore the lower bound of $sl_w(G^s_{cls}, \phi_s) - cp(\phi_0)$ can be derived by three factors, i.e., decrease of the data transfer time by localization in one path, increase of the processing time by task merging steps (from ϕ_0 to ϕ_s), and increase of data transfer time for each unlocalized edges (from ϕ_0 to ϕ_s). The localized data transfer time is derived by taking the sum of localized data transfer time for one path. On the other hand, if increase of the processing time is derived by taking the minimum of the sum of increase of task processing time from ϕ_0 to ϕ_s for each path, this value is λ or more. The unlocalized data transfer time is expressed as μ. Then we have

$$-\zeta + \lambda + \mu \le sl_w(G^s_{cls}, \phi_s) - cp(\phi_0). \tag{21}$$

If $sl(G^s_{cls}, \phi_s) \leq cp(\phi_0) = sl(G^0_{cls}, \phi_0)$, we obtain

$$-\zeta + \lambda + \mu \leq sl_w(G^R_{cls}, \phi_R) - sl(G^R_{cls}, \phi_R) \tag{22}$$

$$\Leftrightarrow sl(G^R_{cls}, \phi_R) \leq sl_w(G^R_{cls}, \phi_R) + \zeta - \lambda - \mu. \tag{23}$$

□

Theorem 4.2 is true if we adopt a clustering policy such that $sl(G^s_{cls}, \phi_s) \leq sl(G^{s-1}_{cls}, \phi_{s-1})$. From theorem 4.1 and 4.2, it can be concluded that reducing the $sl_w(G^s_{cls}, \phi_s)$ leads to the reduction of the schedule length in a heterogeneous distributed system. Thus, the first objective of our proposal is to minimize $sl_w(G^s_{cls}, \phi_s)$.

4.3 The lower bound of each cluster size

To achieve processor utilization, satisfying only "$sl_w(G^s_{cls}, \phi_s)$ minimization" not enough, because this value does not guarantee each cluster size. Thus, in this section we present how large each cluster size should be. In the literature (Kanemitsu, 2010), the lower bound of each cluster size in an identical processor system is derived as follows.

$$\delta_{opt} = \sqrt{cp_w \max_{n^0_k \in V_0} \left\{ \frac{\displaystyle\max_{n^0_l \in V^0_{cls}} \left\{ w(n^0_l) \right\}}{g_{\max}} \right\}}. \tag{24}$$

(24) is the lower bound of each cluster size when $sl_w(G^R_{cls})$ can be minimized, provided that every cluster size is above a certain threshold, δ. And R corresponds to the number of merging steps when every cluster size is δ_{opt} or more. If taking the initial state of the DAG in a heterogeneous system into account, δ_{opt} is expressed by $\delta_{opt}(\phi_0)$ as follows.

$$\delta_{opt}(\phi_0) = \sqrt{cp_w(\phi_0) \max_{n^0_l \in V_0} \left\{ \frac{\displaystyle\max_{n^0_l \in V_0} \left\{ t_p(n^0_l, \max_{\alpha_i \in \alpha} \{\alpha_i\}) \right\}}{g_{\max}(\phi_0)} \right\}}. \tag{25}$$

By imposing $\delta_{opt}(\phi_0)$, it can be said that at least $sl_w(G^0_{cls}, \phi_0)$ can be minimized. However, for $s \geq 1$ $sl_w(G^s_{cls}, \phi_s)$ can not always be minimized by $\delta_{opt}(\phi_0)$, because the mapping of each cluster and each processor is changed and then $sl_w(G^s_{cls}, \phi_s)$ is not equal to $sl_w(G^0_{cls}, \phi_0)$. In this chapter, one heuristic of our method is to impose the same lower bound ($\delta_{opt}(\phi_0)$) for every cluster which will be generated by the task clustering.

5. Task clustering algorithm

5.1 Overview of the algorithm

In the previous section, we presented how large each cluster size should be set for processor utilization. In this section, we present the task clustering algorithm with incorporating the following two requirements.

1. Every cluster size is $\delta_{opt}(\phi_0)$ or more.

2. Minimize $sl_w(G^R_{cls}, \phi_R)$, where R is the total number of merging steps until the first requirement is satisfied.

Fig. 3 shows the task clustering algorithm. At first, the mapping ϕ_0 is applied to every task. Then $\delta_{opt}(\phi_0)$ is derived. Before the main procedures, two sets are defined, i.e., UEX_s and RDY_s. UEX_s is the set of clusters whose size is smaller than $\delta_{opt}(\phi_0)$, and RDY_s is defined as follow.

$$RDY_s = \{cls_s(r) | cls_s(r) \in UEX_s, pred(n^s_{r'}) = \emptyset, cls_s(r) = \{n^s_{r'}\}\} \\ \cup \left\{ \begin{array}{l} cls_s(r) | cls_s(r) \in UEX_s, cls_s(q) \notin UEX_s, \\ n^s_{q'} \in cls_s(q), n^s_{q'} \in pred(n^s_{r'}) \ for \ \forall n^s_{r'} \in top_s(r) \end{array} \right\}. \tag{26}$$

RDY_s is the set of clusters whose preceding cluster sizes are $\delta_{opt}(\phi_0)$ or more. That is, the algorithm tries to merge each cluster in top-to-bottom manner.

The algorithm is proceeded during $UEX_s \neq \emptyset$, which implies that at least one cluster in UEX_s exists. At line 3, one processor is selected by a processor selection method, e.g., by CHP(C. Boeres, 2004) (In this chapter, we do not present processor selection methods). At line 4, one cluster is selected as $pivot_s$, which corresponds to "the first cluster for merging". Once the $pivot_s$ is selected, "the second cluster for merging", i.e., $target_s$ is needed. Thus, during line 5 to 7, procedures for selecting $target_s$ and merging $pivot_s$ and $target_s$ are performed. After those procedures, at line 7 RDY_s is updated to become RDY_{s+1}, and $pivot_s$ is also updated to become $pivot_{s+1}$. Procedures at line 6 and 7 are repeated until the size of $pivot_s$ is $\delta_{opt}(\phi_0)$ or more. The algorithm in fig. 3 has common parts with that of the literature (Kanemitsu, 2010), i.e., both algorithms use $pivot_s$ and $target_s$ for merging two clusters until the size of $pivot_s$ exceeds a lower bound of the cluster size. However, one difference among them is that the algorithm in fig. 3 keeps the same $pivot_s$ during merging steps until its size exceeds $\delta_{opt}(\phi_0)$, while the algorithm in (Kanemitsu, 2010) selects the new $pivot_s$ in every merging step. The reason of keeping the same $pivot_s$ is to reduce the time complexity in selection for $pivot_s$, which requires scanning every cluster in RDY_s. As a result, the number of scanning RDY_s can be reduced with compared to that of (Kanemitsu, 2010).

5.2 Processor assignment

In the algorithm presented in fig. 3, the processor assignment is performed before selecting $pivot_s$. Suppose that a processor P_p is selected before the $s+1$th merging step. Then we assume that P_p is assigned to every cluster to which P_{max} is assigned, i.e., no actual processor has been assigned. By doing that, we assume that such unassigned clusters are assigned to "an identical processor system by P_p" in order to select $pivot_s$. Fig. 4 shows an example of the algorithm. In the figure, (a) is the state of ϕ_2, in which the size of $cls_2(1)$ is $\delta_{opt}(\phi_0)$ or more. Thus, $RDY_2 = \{cls_2(3) = \{n^2_3\}, cls_2(4) = \{n^2_4\}, cls_2(7) = \{n^2_7\}\}$. The communication bandwidth from P_1 to P_{max} is set as $\underset{1 \leq q \leq \mathrm{m}, 1 \neq q}{min} \{\beta_{1,q}\}$ in order to regard communication bandwidth between an actual processor and P_{max} bottleneck in the schedule length. In (b), it is assumed that every cluster in UEX_2 is assigned to P_p after P_p is selected. Bandwidths among P_p are set as $\underset{1 \leq q \leq \mathrm{m}, p \neq q}{min} \{\beta_{p,q}\}$ to estimate the $sl_w(G^2_{cls}, \phi_2)$ of the worst case. Therefore, $pivot_2$(in this case, $cls_2(3)$) is selected by deriving LV value for each cluster in RDY_2, provided that such a mapping state. After (b), if the size of $cls_3(3)$ is smaller than $\delta_{opt}(\phi_0)$, every cluster in UEX_3 is still assigned to P_p to maintain the mapping state. In (d) if the size of $cls_4(3)$

```
INPUT: $G^0_{cls}$
OUTPUT: $G^R_{cls}$
Set the mapping $\phi_0$ to the input DAG.
Define $UEX_s$ as a set of clusters whose size is under $\delta_{opt}(\phi_0)$;
Define $RDY_s$ as a set of clusters which statisies eq. (26).;
For each $n_k \in V$, let $n^0_k \leftarrow n_k, cls_0(k) = \{n^0_k\}$ and put $cls_0(k)$ into $V^0_{cls}$.
 0.  Derive $\delta_{opt}(\phi_0)$ by eq. (25).
 1.  $E_0 \leftarrow E, UEX_0 \leftarrow V^0_{cls}, RDY_0 \leftarrow \{cls_0(k)|cls_0(k) = \{n^0_k\}, pred(n^0_k) = \emptyset\}$;
 2.  WHILE $UEX_s \neq \emptyset$ DO
 3.    select a processor $P_p$ from $P$;
 4.    $pivot_s \leftarrow getPivot(RDY_s)$;
 5.    WHILE size of $pivot_s < \delta_{opt}(\phi_0)$ DO
 6.      $target_s \leftarrow getTarget(pivot_s)$;
 7.      $RDY_{s+1} \leftarrow merging(pivot_s, target_s)$ and update $pivot_s$;
 8.    ENDWHILE
 9.  ENDWHILE
10.  RETURN $G^R_{cls}$;
```

Fig. 3. Procedures for the Task Clustering.

exceeds $\delta_{opt}(\phi_0)$, the mapping is changed i.e., clusters in UEX_4 are assigned to P_{max} to select the new $pivot_4$ for generating the new cluster.

5.3 Selection for $pivot_s$ and $target_s$

As mentioned in 5.1, one objective of the algorithm is to minimize $sl_w(G^R_{cls}, \phi_R)$. Therefore, in RDY_s, $pivot_s$ should have maximum LV value (defined in table 1), because such a cluster may dominate $sl_w(G^s_{cls}, \phi_s)$ and then $sl_w(G^{s+1}_{cls}, \phi_{s+1})$ after $s+1$ th merging may became lower than $sl_w(G^s_{cls}, \phi_s)$. Our heuristic behined the algorithm is that this policy for selecting $pivot_s$ can contribute to minimize $sl_w(G^R_{cls}, \phi_R)$. The same requirement holds to the selection of $target_s$, i.e., $target_s$ should be the cluster which dominates LV value of $pivot_s$. In fig. 4 (b), $cls_2(3)$ having the maximum LV value in RDY_2 is selected. Then n^2_6, i.e., $cls_2(6)$ dominating $LV_2(3)$ is selected as $target_2$. similarly in (c) n^3_5, i.e., $cls_3(5)$ dominating $LV_3(3)$ is selected as $target_3$.

5.4 Merging $pivot_s$ and $target_s$

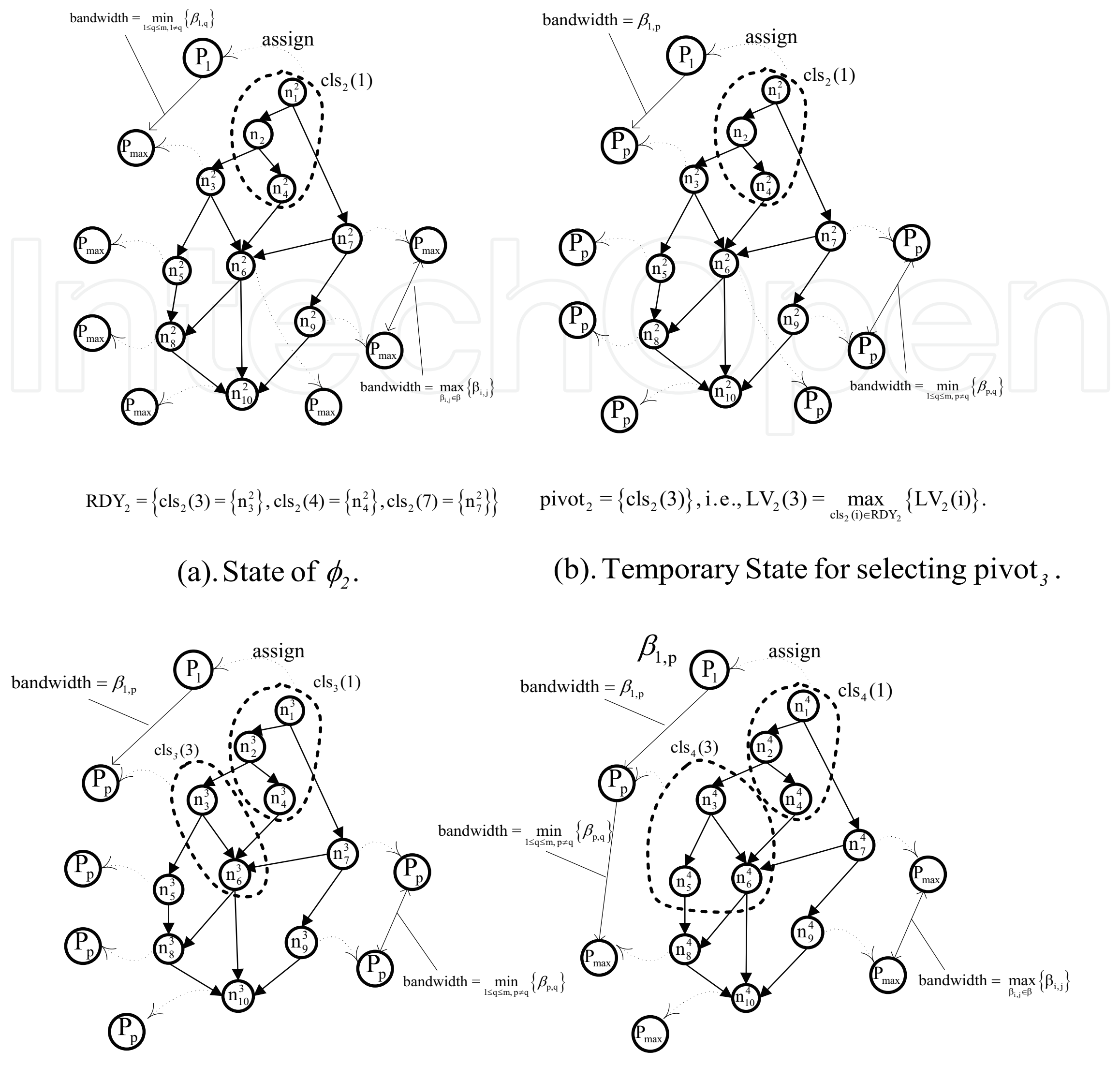
After $pivot_s$ and $target_s$ have been selected, the merging procedure, i.e.,

$$pivot_{s+1} \leftarrow pivot_s \cup target_s \tag{27}$$

is performed. This procedure means that every cluster in $target_s$ is included in $pivot_{s+1}$. Then $pivot_s$ and $target_s$ are removed from UEX_s and RDY_s. After this merging step has been performed, clusters satisfying requirements for RDY_{s+1}(in eq. (26)) are included in RDY_{s+1}. Furthermore, every cluster's LV value is updated for selecting $pivot_{s+1}$ and $target_{s+1}$ before the next merging step.

6. 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.


bandwidth = $\min_{1\le q\le m, 1\ne q}\{\beta_{1,q}\}$
assign
P_1
$cls_2(1)$
P_{max}
bandwidth = $\max_{\beta_{i,j}\in\beta}\{\beta_{i,j}\}$
$RDY_2 = \{cls_2(3) = \{n_3^2\}, cls_2(4) = \{n_4^2\}, cls_2(7) = \{n_7^2\}\}$
bandwidth = $\beta_{1,p}$
P_p
bandwidth = $\min_{1\le q\le m, p\ne q}\{\beta_{p,q}\}$
$pivot_2 = \{cls_2(3)\}$, i.e., $LV_2(3) = \max_{cls_2(i)\in RDY_2}\{LV_2(i)\}$.
$cls_3(1)$
$cls_3(3)$
$\beta_{1,p}$
$cls_4(1)$
$cls_4(3)$

(a). State of ϕ_2.

(b). Temporary State for selecting $pivot_3$.

(c). The size of $cls_3(3) < \delta_{opt}(\phi_0)$.

(d). State of ϕ_4, and the size of $cls_4(3) \ge \delta_{opt}(\phi_0)$.

Fig. 4. Example of the Task Clustering Algorithm.

1. Whether minimizing $sl_w(G^R_{cls}, \phi_R)$ leads to minimzing the schedule length or not.

2. The range of applicability by imposing $\delta_{opt}(\phi_0)$ as the lower bound of every cluster size.

We showed by theorem 4.1 and 4.2 that both the lower bound and the upper bound of the schedule length can be expressed by $sl_w(G^s_{cls}, \phi_s)$. Thus, in this experiment we confirm that the actual relationship between the schedule length and $sl_w(G^s_{cls}, \phi_s)$.

6.1 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)(Sinnen, 2005; 2007) 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 (H. Topcuoglu, 2002). By using PF, the depth of the DAG is defined as $\frac{\sqrt{|V_0|}}{\rho}$.

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.

6.2 Comparison about $sl_w(G^R_{cls}, \phi_R)$ and the schedule length

In this experiment, we compared $sl_w(G^R_{cls}, \phi_R)$ and the schedule length to confirm the validity of theorem 4.1 and 4.2. Comparison targets are as follows.

IntechOpen

No.	α	β	CCR	$\lvert V^R_{cls}\rvert$	$sl_w(G^R_{cls}, \phi_R)$ Ratio			$sl(G^R_{cls}, \phi_R)$ Ratio		
					A	B	C	A	B	C
1	5	5	0.1	168	1.000	1.054	1.097	1.000	1.018	1.123
2			1.0	56	1.000	1.241	1.391	1.000	1.131	1.209
3			5.0	34	1.000	1.320	1.514	1.000	1.140	1.288
4			10.0	19	1.000	1.378	1.611	1.000	1.219	1.341
5	5	10	0.1	160	1.000	1.072	1.105	1.000	1.012	1.073
6			1.0	49	1.000	1.203	1.419	1.000	1.144	1.248
7			5.0	30	1.000	1.355	1.428	1.000	1.172	1.301
8			10.0	16	1.000	1.329	1.503	1.000	1.237	1.344
9	10	5	0.1	150	1.000	1.032	1.066	1.000	1.016	1.071
10			1.0	47	1.000	1.177	1.284	1.000	1.097	1.198
11			5.0	41	1.000	1.209	1.615	1.000	1.173	1.291
12			10.0	26	1.000	1.482	1.598	1.000	1.227	1.302
13	10	10	0.1	187	1.000	1.069	1.044	1.000	1.044	1.050
14			1.0	67	1.000	1.292	1.157	1.000	1.179	1.132
15			5.0	44	1.000	1.344	1.419	1.000	1.203	1.297
16			10.0	28	1.000	1.461	1.433	1.000	1.272	1.301

Table 3. Comparison of $sl_w(G^R_{cls}, \phi_R)$ and the Schedule Length in Random DAGs($|V_0| = 1000$).

IntechOpen

No.	α	β	CCR	$\lvert V^R_{cls}\rvert$	$sl_w(G^R_{cls}, \phi_R)$ Ratio			$sl(G^R_{cls}, \phi_R)$ Ratio		
					A	B	C	A	B	C
1	5	5	0.1	351	1.000	1.032	1.041	1.000	1.021	1.049
2			1.0	144	1.000	1.193	1.241	1.000	1.098	1.142
3			5.0	78	1.000	1.216	1.266	1.000	1.126	1.172
4			10.0	44	1.000	1.272	1.371	1.000	1.190	1.193
5	5	10	0.1	346	1.000	1.048	1.044	1.000	1.013	1.013
6			1.0	136	1.000	1.177	1.233	1.000	1.133	1.152
7			5.0	75	1.000	1.242	1.385	1.000	1.152	1.189
8			10.0	41	1.000	1.238	1.411	1.000	1.273	1.206
9	10	5	0.1	344	1.000	1.022	1.037	1.000	1.044	1.031
10			1.0	135	1.000	1.093	1.133	1.000	1.086	1.099
11			5.0	72	1.000	1.203	1.192	1.000	1.173	1.162
12			10.0	39	1.000	1.288	1.370	1.000	1.234	1.221
13	10	10	0.1	367	1.000	1.017	1.041	1.000	1.013	1.011
14			1.0	149	1.000	1.188	1.241	1.000	1.076	1.081
15			5.0	80	1.000	1.279	1.339	1.000	1.147	1.175
16			10.0	46	1.000	1.341	1.367	1.000	1.198	1.201

Table 4. Comparison of $sl_w(G^R_{cls}, \phi_R)$ and the Schedule Length in FFT DAGs($|V_0| = 2048$).

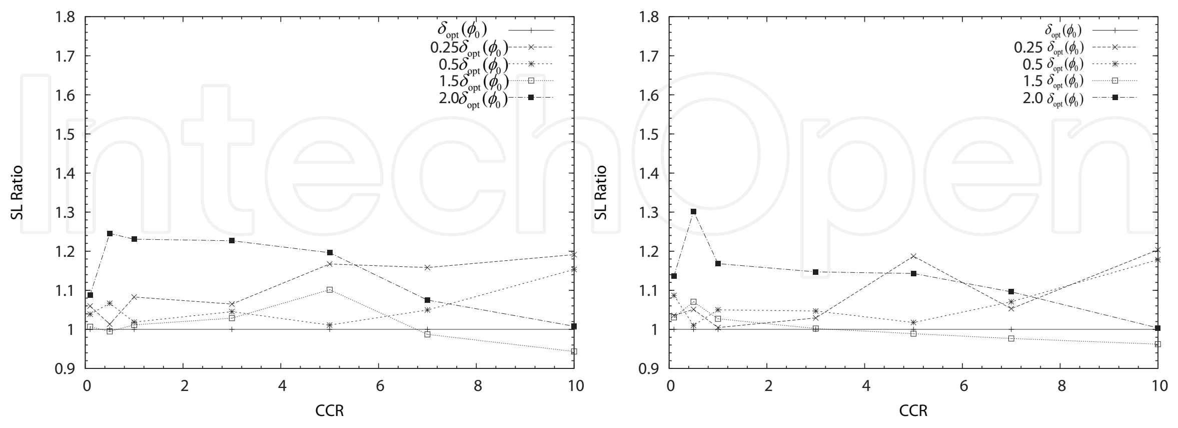
A. At first, the lower bound of the cluster size is derived as $\delta_{opt}(\phi_0)$. Then the task clustering algorithm in fig. 4 is performed, while processor assignment policy is based on CHP(C. Boeres, 2004).

B. The lower bound of the cluster size is derived as $\delta_{opt}(\phi_0)$. Then the task clustering policy is based on "load balancing" (J. C. Liou, 1997), while processor assignment policy is based on CHP(C. Boeres, 2004), in which merging step for generating one cluster is proceeded until the cluster size exceeds $\delta_{opt}(\phi_0)$.

C. The lower bound of the cluster size is derived as $\delta_{opt}(\phi_0)$. Then the task clustering policy is random-basis, i.e., two clusters smaller than $\delta_{opt}(\phi_0)$ are selected randomly to merge into one larger cluster, while processor assignment policy is based on CHP(C. Boeres, 2004), in which merging step for generating one cluster is proceeded until the cluster size exceeds $\delta_{opt}(\phi_0)$.

The difference between A, B and C is how to merge clusters, while they have the common lower bound for the cluster size and the common processor assignment policy. We compared $sl_w(G^R_{cls}, \phi_R)$ and the schedule length by averaging them in 100 random DAGs.

Table 3 and 4 show comparison results in terms of $sl_w(G^R_{cls}, \phi_R)$ and the schedule length. The former is the result in the case of random DAGs. On the other hand, the latter is the result in the case of FFT DAGs. In both tables, α corresponds to max-min ratio for processing speed in P, and β corresponds to max-min ratio for communication bandwidth in P. "$sl_w(G^R_{cls}, \phi_R)$ Ratio" and "$sl(G^R_{cls}, \phi_R)$ Ratio" correspond to ratios to "A", i.e., a value larger than 1 means that $sl_w(G^R_{cls}, \phi_R)$ or $sl(G^R_{cls}, \phi_R)$ is larger than that of "A". In table 3, it can be seen that both $sl_w(G^R_{cls}, \phi_R)$ and $sl(G^R_{cls}, \phi_R)$ in "A" are better than "B" and "C" as a whole. Especially, the larger CCR becomes, the better both $sl_w(G^R_{cls}, \phi_R)$ and $sl(G^R_{cls}, \phi_R)$ in "A" become. It can not be seen that noteworthy characteristics related to $sl_w(G^R_{cls}, \phi_R)$ and $sl(G^R_{cls}, \phi_R)$ with varying the degree of heterogeneity (i.e., α and β). The same results hold to table 4. From those results, it can be concluded that minimizing $sl_w(G^R_{cls}, \phi_R)$ leads to minimizing the schedule length as theoretically proved by theorem 4.1 and 4.2.



(a) $(\alpha, \beta) = (5, 5)$ (b) $(\alpha, \beta) = (10, 10)$

Fig. 5. Optimality for the Lower Bound of the Cluster Size.

6.3 Applicabiligy of $\delta_{opt}(\phi_0)$

In this experiment, we confirmed that how optimal the lower bound of the cluster size, $\delta_{opt}(\phi_0)$ derived by eq. (25). Comparison targets in this experiment are based on "A" at sec. 6.2, but only the lower bound of the cluster size is changed, i.e., $\delta_{opt}(\phi_0)$, $0.2\delta_{opt}(\phi_0)$, $0.5\delta_{opt}(\phi_0)$, $1.5\delta_{opt}(\phi_0)$, and $2.0\delta_{opt}(\phi_0)$. The objective of this experiment is to confirm the range of applicability of $\delta_{opt}(\phi_0)$, due to the fact that $\delta_{opt}(\phi_0)$ is not a value when $sl_w(G^s_{cls}, \phi_s)$ can be minimized for $1 \leq s$. Fig. 5 shows comparison results in terms of the optimality of $\delta_{opt}(\phi_0)$. (a) corresponds to the case of the degree of heterogeneity $(\alpha, \beta) = (5, 5)$, and (b) corresponds to $(10, 10)$. From (a), it can be seen that $\delta_{opt}(\phi_0)$ takes the best schedule length than other cases during CCR takea from 0.1 to 5.0. However, when CCR is 7 or more, $1.5\delta_{opt}(\phi_0)$ takes the best schedule length. This is because $\delta_{opt}(\phi_0)$ may be too small for a data intensive DAG. Thus, it can be said that $1.5\delta_{opt}(\phi_0)$ is more appropriate size than $\delta_{opt}(\phi_0)$ when CCR exceeds a certain value. On the other hand, in (b), the larger CCR becomes, the better the schedule length by case of $1.5\delta_{opt}(\phi_0)$ becomes. However, during CCR is less than 3.0, $\delta_{opt}(\phi_0)$ can be the best lower bound of the cluster size. As for other lower bounds, 2.0 $\delta_{opt}(\phi_0)$ has the local maximum value of the schedule length ratio when CCR takes from 0.1 to 2.0 in both figures. Then in larger CCR, the schedule length ratio decreases because such size becomes more appropriate for a data intensive DAG. On the other hand, in the case of $0.25\delta_{opt}(\phi_0)$, the schedule length ratio increases with CCR. This means that $0.25\delta_{opt}(\phi_0)$ becomes smaller for a data intensive DAG with CCR increases.

From those results, it can be said that the lower bound for the cluster size should be derived according to the mapping state. For example, if the lower bound can be adjusted as a function of each assigned processor's ability (e.g., the processing speed and the communication bandwidth), the better schedule length may be obtained. For example in this chapter the lower bound is derived by using the mapping state of ϕ_0. Thowever, by using the other mapping state, we may be obtain the better schedule length. To do this, it must be considered that which mapping state has good effect on the schedule length. This point of view is an issue in the future works.

7. Conclusion and future works

In this chapter, we presented a policy for deciding the assignment unit size to a processor and a task clustering for processor utilization in heterogeneous distributed systems. We defined the indicative value for the schedule length for heterogeneous distributed systems. Then we theoretically proved that minimizing the indicative value leads to minimization of the schedule length. Furthermore, we defined the lower bound of the cluster size by assuming the initial mapping state. From the experimental results, it is concluded that minimizing the indicative value has good effect on the schedule length. However, we found that the lower bound of the cluster size should be adjusted with taking an assigned processor's ability into account.

As a future work, we will study on how to adjust the lower bound of the cluster size for obtaining the better schedule length and more effective processor utilization.

8. References

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.

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.

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

Hidehiro Kanemitsu, Gilhyon Lee, Hidenori Nakazato, Takashige Hoshiai, and Yoshiyori Urano, Static Task Cluster Size Determination in Homogeneous Distributed Systems, *Software Automatic Tuning: from concepts to state-of-the-art results*, Springer-Verlag (ISBN: 1441969349), pp. 229 – 252, 2010.

C. Boeres, J. V. Filho and V. E. F. Rebello, A Cluster-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.

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

S. Chingchit, M. Kumar and L.N. Bhuyan, A Flexible Clustering and Scheduling Scheme for Efficient Parallel Computation, *Procs. of the 13th International and 10th Symposium on Parallel and Distributed Processing*, pp. 500 – 505, 1999.

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.

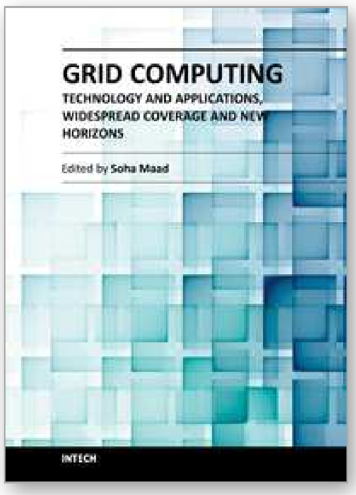
O. Sinnen., Task Scheduling for Parallel Systems, *Wiley*, 2007.

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.

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.

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.

IntechOpen



Grid Computing - Technology and Applications, Widespread Coverage and New Horizons
Edited by Dr. Soha Maad

ISBN 978-953-51-0604-3
Hard cover, 354 pages
Publisher InTech
Published online 16, May, 2012
Published in print edition May, 2012

Grid research, rooted in distributed and high performance computing, started in mid-to-late 1990s. Soon afterwards, national and international research and development authorities realized the importance of the Grid and gave it a primary position on their research and development agenda. The Grid evolved from tackling data and compute-intensive problems, to addressing global-scale scientific projects, connecting businesses across the supply chain, and becoming a World Wide Grid integrated in our daily routine activities. This book tells the story of great potential, continued strength, and widespread international penetration of Grid computing. It overviews latest advances in the field and traces the evolution of selected Grid applications. The book highlights the international widespread coverage and unveils the future potential of the Grid.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Hidehiro Kanemitsu, Gilhyon Lee, Hidenori Nakazato, Takashige Hoshiai and Yoshiyori Urano (2012). 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, Dr. Soha Maad (Ed.), ISBN: 978-953-51-0604-3, InTech, Available from: http://www.intechopen.com/books/grid-computing-technology-and-applications-widespread-coverage-and-new-horizons/on-the-effect-of-applying-the-task-clustering-for-identical-processor-utilization-to-heterogeneous-s

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

© 2012 The Author(s). Licensee IntechOpen. This is an open access article distributed under the terms of the Creative Commons Attribution 3.0 License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.