Static Task Cluster Size Determination in Homogeneous Distributed Systems

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

Graduate School of Global Information and Telecommunication Studies, Waseda University
1-3-10, Nishiwaseda, Shinjyuku, Tokyo, Japan
kanemih@ruri.waseda.jp

Abstract. In a distributed system which consists of unknown number of machines, many task clustering heuristics have been proposed in order to satisfy requirements such as determining the number of machines and minimizing response time for scheduling Directed Acyclic Graph(DAG). However, those heuristics are not aware of the actual number of existing machines, because their objective is only to minimize response time. As a result, the number of machines determined by an existing task clustering may exceed the number of actually existing machines. Therefore, conventional approaches adopt merging each cluster for reducing the number of clusters at the expense of increase of response time.
In this paper, we present a static cluster size determination method and requirements for a task clustering in order to suppress response time while reducing the number of clusters. Our experimental evaluation by simulation shows the effectiveness of a task clustering which adopts the requirements.

Key words: Task Clustering, Task Scheduling, DAG, Cluster Size

1 Introduction

In a distributed system where each machine sends or receives data over the network, scheduling tasks to minimize response time is very important issue[2, 8, 7, 6]. As one of existing task scheduling models to be assumed, scheduling task graph which consists of directed edges and tasks, i.e., Directed Acyclic Graph(DAG), has been considered as an NP-complete problem[6]. Especially, if the number of machines is not given, we must derive not only execution order of each task, but also the number of machines in order to obtain good response time. As one approach which meets those requirements, task clustering[1] has been known. One fundamental feature of task clustering is to merge several tasks into one cluster by localizing communication overhead among them, so that each cluster corresponds to each assignment unit per one machine. However, if the smaller communication overheads among tasks become, the longer the response time becomes due to the fact that increase of task parallelism can

prolong response time. In a distributed environment to which several application can be submitted, e.g., in a grid environment, utilizing computational resources effectively is a key factor for achieving reduced response time for executing all applications. For realizing utilization of computational resources, it is important to derive execution order of every task which can minimize response time, while reducing the number of clusters as much as possible. There are several heuristic approaches whose purposes are to reduce the number of clusters by merging several clusters into a larger one after task clustering. Pyrros compiling infrastructure[4] adopts a criterion for equalizing each cluster size. Liou et al. proposed two task merging approach, i.e., LB(Load Balancing) and CTM(Communication Traffic Minimizing)[5]. Merging criterion of LB is the same as the cluster merging adopted in Pyrros[4] except that LB does not consider data dependencies among tasks. On the other hand, the criterion of cluster merging performed by CTM is that sum of data transfer time among clusters to be merged is minimized as far as possible. According to the results of performance comparison between LB and CTM, both of cluster merging approaches have bad effects on response time when they are performed after task clustering, e.g., CASS-II[5]. According to [5], performing LB after CASS-II, makes response time prolonged up to 19 % compared with response time obtained by only CASS-II. In the case of performing CTM after CASS-II, response time is prolonged up to 55 %. This means that a cluster merging approach which sacrifices task parallelism can prolong response time. Thus, a cluster merging strategy for maintaining task parallelism with the small number of clusters is required.

In this paper, we present a method for cluster size determination in order to obtain a good response time with small number of clusters. As one heuristic for reducing the number of clusters generated by a task clustering, we impose a lower bound for each cluster size as "δ". Since imposing such a lower bound can lead to decrease of task parallelism in each cluster, the fundamental objective is to suppress the increase of response time. Therefore, we derived the cluster size in the case that an upper bound of response time can be minimized as much as possible. Then we present requirements for a task clustering heuristic for minimizing an upper bound of response time.

The remainder of this paper is organized as follows. Section 2 presents assumption and problem definition, and defiinition of an upper bound of response time is presented in section 3. Derivation method of cluster size is presented in section 4. Experimental evaluation by simulation is presented in section 5, and finally, conclusion and our future work are presented in section 6.

2 Problem Definition and Assumptions

2.1 Assumed Model

Let $G_{org} = (V, E)$ be the initial DAG, and V be a set of tasks, E be a set of data communications among tasks. An i-th task is denoted as n_i. Also, here we assume every machine has the same processing speed and data transfer rate. Let $w(n_i)$ be a size of n_i, i.e., $w(n_i)$ is the processing time of n_i. We define data

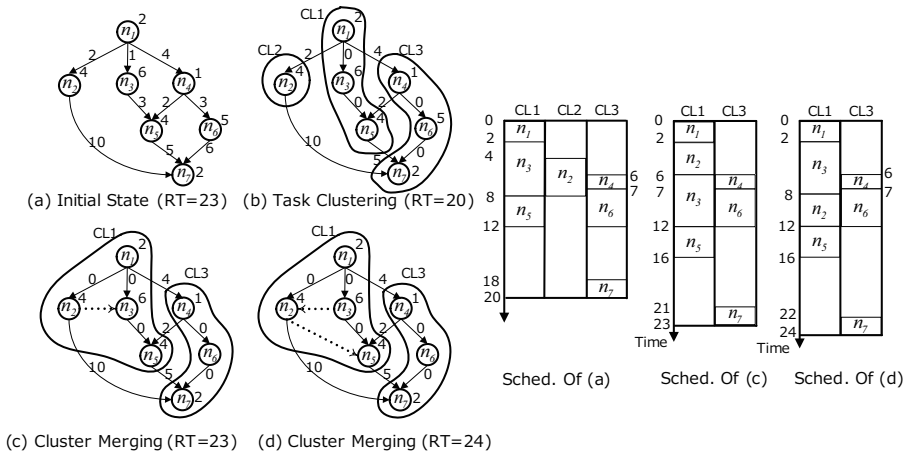

Fig. 1. Derivation of Response Times by Task Clustering and Cluster Merging.

dependency and direction of data transfer from n_i to n_j as $e_{i,j}$. And we define data transfer time from n_i to n_j as $c(e_{i,j})$.

The assumed network model is that each machine is completely connected, i.e., multiple data from one machine can be transferred in parallel, such as [3, 6, 8, 10, 11]. One constraint imposed by DAG is that a task can not be started execution until all the data from its predecessor tasks arrives. For instance, $e_{i,j}$ means that n_j can not be started until data from n_i arrives at the machine which will execute n_j. And let $pred(n_i)$ be the set of immediate predecessors of n_i, and $suc(n_i)$ be the set of immediate successors of n_i. If $pred(n_i) = \emptyset$, n_i is called START task, and if $suc(n_i) = \emptyset$, n_i is called END task. If there are one or more paths from n_i to n_j, we denote such a relation as $n_i \prec n_j$.

2.2 Task Clustering

In general, a task clustering heuristic performs merging several tasks into one cluster, so that each cluster becomes one assignment unit per one machine. Let us denote the clustered DAG, which are the resultant DAG after task clustering as $G_{cls} = (V', E')$, where V' is the set of clusters. We denote the i-th cluster in V' as $cls(i)$. If n_k is included in $cls(i)$ by task clustering, we formulate such an procedure as $cls(i) \leftarrow cls(i) \cup \{n_k\}$. If any two tasks, i.e., n_i and n_j, are included in the same cluster, they are assigned to the same machine, and the communication between n_i and n_j is localized, so that we define $c(e_{i,j})$ becomes zero. We define that the size of each cluster, i.e., processing time of $cls(i)$ in a machine, is denoted as $w(cls(i))$. Throughout this paper, we denote that $cls(i)$ is "linear" if and only if $cls(i)$ contains no independent task[3]. 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.

2.3 Problems in Conventional Cluster Merging

If the number of clusters is greater than that of machine, some approaches for cluster merging to adjust the number of the former to that of the latter are required. Fig. 1 demonstrates procedures by task clustering(b) and cluster merging((c),(d)). In this figure, (a) is the initial DAG, and (b) is the resultant DAG which consists of 3 clusters, i.e., CL1, CL2, CL3 by performing task clustering. (c) and (d) are the resultant DAGs generated by performing cluster merging after the state of (b). Response time of initial DAG((a) at Fig. 1) corresponds to the length of a critical path, i.e., $n_1 \rightarrow n_3 \rightarrow n_5 \rightarrow n_7$, whose value is 23. From this state, we obtain the state of (b) by performing linear clustering[3]. In a cluster generated as a result of linear clustering, precedence relationship between any two tasks is decided, so that the execution order of all tasks in the cluster is fixed. This means that no scheduling is required for the DAG generated after linear clustering. The response time of DAG (b) is 20. If the number of actual machines is 2, the clusters must be merged as (c) or (d). If cluster 2(CL2) is merged with cluster(CL1), two types of independent relations exist in CL1., i.e., n_2, n_3 and n_2, n_5. As a result, the response time in (c) is 23 by tracing $n_2 \rightarrow n_3 \rightarrow n_5$. On the other hand, the response time in (d) is 24 with considering that arrival time of $c(e_{2,7})$ at n_7 is prolonged, because the execution order in CL1, i.e., $n_3 \rightarrow n_2 \rightarrow n_5$ makes start time of n_2 prolonged than that of (c).

From those results, we can say that start time of each task can be prolonged if task merging is performed after task clustering, depending on dependencies among tasks. Thus, the greater the number of merged clusters is, the longer response time is. Such two step procedures have been adopted in [5, 4]. At the task clustering phase, criteria for minimizing response time with small number of clusters are required for a distributed environment to which multiple applications can be submitted. We impose the lower bound of cluster size in order to limit the number of clusters. Here, we define such a lower bound as "δ".

2.4 Requirements for Reducing both Response Time and the Number of Machines

Conventional task clustering heuristics[2, 6, 8, 10] decide a response time based on a specific scheduling policy, i.e., a scheduling priorities in order to check whether a response time is prolonged or not after each clustering procedure. In such cases, if a response time is prolonged, the task clustering procedure is not accepted and then one cluster generation is completed. On the other hand, if δ is introduced into a task clustering heuristic, each cluster size must exceed the δ, so that the obtained number of cluster can be guaranteed, i.e., the maximum number of cluster is $\lfloor \frac{1}{\delta} \sum_{n_k \in V} w(n_k) \rfloor$.

However, if we impose δ as lower bound of cluster size and then each task is merged into one cluster, several tasks which have no dependencies may be included in the same cluster, so that response time can not be suppressed like (c) and (d) in Fig. 1. We need criteria for suppressing the increase of response time

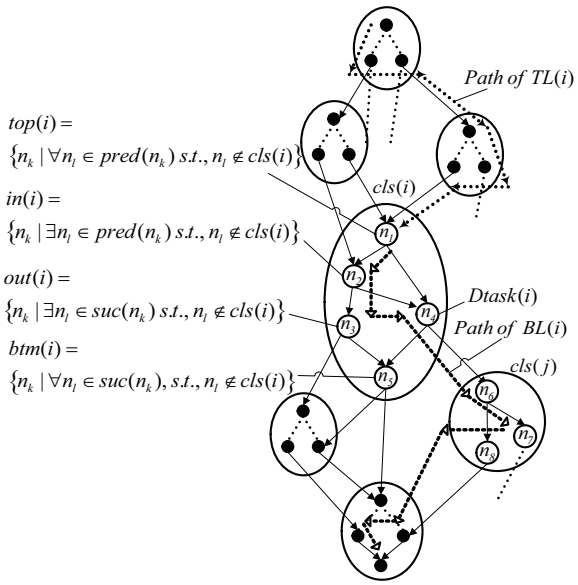

Fig. 2. Notations of a DAG.

if the condition, i.e., δ is imposed to a task clustering heuristic. Our objective in this paper is to derive δ, i.e., δ_{opt} by which an upper bound of the response time is minimized. In this paper, we define the upper bound as "Worst Response Time (WRT)".

3 Definition of WRT

3.1 Abstract of WRT

Fig. 2 illustrates notations required for deriving WRT. In this figure, the DAG consists of several clusters. In $cls(i)$, $top(i)$ is the set of tasks whose all immediate predecessor tasks belong to other clusters. This means that any task in $top(i)$ must be executed before tasks which does not belong to $top(i)$. $in(i)$ is the set of tasks, one or more of whose immediate predecessor tasks belong to other clusters. $out(i)$ is the set of tasks, one or more of whose immediate successor tasks belong to other clusters. $btm(i)$ is the set of tasks whose all immediate successor tasks belong to other clusters.

In reality, for each task we need to derive data arrival time from its immediate predecessor tasks to decide an actual upper bound of response time. However, at each task, waiting time for data arrival depends on the scheduling policy which is applied during each task merging procedure. WRT must be the one which is independent from every scheduling policy. Therefore, we impose a relaxation condition that waiting time (idle time) for each task in a cluster is ignored in order to avoid calculation steps required for scheduling priorities. Here, We define

that WRT is an upper bound of the worst response time with ignoring every idle time of $in(i)$ tasks, for every cluster, $cls(i)$.

3.2 Derivation of WRT

We need to derive start time for each task in $top(i)$ in order to decide WRT. At first, we define $TL(i)$ as the latest time when tasks in $top(i)$ may start.

$$TL(i) = \max_{n_k \in top(i)} \left\{ \max_{n_l \in pred(n_k)} \{tlevel(n_l) + c(e_{l,k})\} \right\}, \tag{1}$$

where $tlevel(n_k)$ is the start time of n_k when it is ready for execution after its all predecessor tasks and independent tasks are executed disregarding data arrival from tasks in other clusters. Note that $n_l \notin cls(i)$. As described in the section 3.1, if n_l does not belong to top tasks of a cluster(here, let the cluster as $cls(h)$), $tlevel(n_l)$ means the start time of n_l with satisfying the following conditions.

- n_m has been executed before n_l, where $\forall n_m \in cls(h), n_m \not\prec n_l, n_l \not\prec n_m, n_m$.
- n_l does not wait for data arrival from all its immediate predecessor tasks.

For each $n_k \in cls(i)$, to derive $tlevel(n_k)$ we need to know the elapsed time from $TL(i)$ to start time of n_k. If the elapsed time in case that n_k is executed as late as possible is defined as $S(n_k, i)$, it is defined as

$$S(n_k, i) = \sum_{n_k \in cls(i)} w(n_k) - \sum_{n_k \in dest(n_k, i)} w(n_k), \tag{2}$$
$$where\ dest(n_k, i) = \{n_m | n_k \prec n_m, n_m \in cls(i)\} \cup \{n_k\}.$$

$dest(n_k, i)$ means the set of n_k's successor tasks in $cls(i)$ and n_k. From (1) and (2), we define $tlevel(n_k)$ as

$$tlevel(n_k) = TL(i) + S(n_k, i),\ where\ n_k \notin top(i). \tag{3}$$

Next, we need to derive the elapsed time from the finish time of every task in $top(i)$ to the finish time of END task. We define such an elapsed time which is prolonged as long as possible as $BL(i)$. To decide $BL(i)$, we need to derive the maximum elapsed time from start time for each task to finish time of END task. Let the maximum elapsed time be $blevel(n_k)$ for each $n_k \in V$. Also, let $blevel_{out}(n_k)$ as the maximum elapsed time from start time of $n_k \in out(i)$ to finish time of END task, which does not include execution time of other tasks in $cls(i)$. In Fig. 2, $blevel(n_4)$ and $blevel_{out}(n_4)$ can be defined as

$$blevel(n_4) = \max\{blevel_{out}(n_4), w(n_4) + blevel(n_5)\},$$
$$where\ blevel_{out}(n_4) = w(n_4) + c(e_{4,6}) + blevel(n_6),$$

where $blevel(n_7)$ and $blevel(n_8)$ must be derived to decide $blevel(n_6)$. The possible execution orders in n_6, n_7, n_8 which decide the maximum time duration from

n_6 to END task is respectively, $n_6 \rightarrow n_7 \rightarrow n_8$, $n_6 \rightarrow n_8 \rightarrow n_7$. Thus, $blevel(n_6)$ is defined as

$$blevel(n_6) = \max\{w(n_6) + w(n_7) + blevel(n_8), w(n_6) + w(n_8) + blevel(n_7).\}$$

Here, suppose $n_k, n_l \in cls(i)$ and let $dest'_i(n_k, n_l)$ mean the following relationship.

$$\begin{aligned} dest'_i(n_k, n_l) &= dest(n_k, i) - dest(n_l, i), \\ blevel_{out}(n_k) &= \max_{n_p \in suc(n_k), n_p \notin cls(i)} \{w(n_k) + c(e_{k,p}) + blevel(n_p)\}. \end{aligned}$$

From $dest'_i(n_k, n_l)$, we obtain $blevel(n_k)$ as

$$blevel(n_k) = \begin{cases} \max\left\{ \begin{array}{l} blevel_{out}(n_k), \\ \max\limits_{n_k \prec n_l} \left\{ \sum\limits_{n_m \in dest'_i(n_k, n_l)} w(n_m) + blevel(n_l) \right\} \end{array} \right\}, & \begin{pmatrix} n_k \notin btm(i), \\ n_k \in out(i) \end{pmatrix} \\ blevel_{out}(n_k), & (n_k \in btm(i)) \\ \max\limits_{n_k \prec n_l} \left\{ \sum\limits_{n_m \in dest'_i(n_k, n_l)} w(n_m) + blevel(n_l) \right\}, & (n_k \notin out(i)). \end{cases} \tag{4}$$

$dest'_i(n_k, n_l)$ is the set of tasks which can be executed during from start time of n_k to finish time of n_l. For instance, in Fig. 2, $dest'_j(n_6, n_8)$ and $dest'_j(n_6, n_7)$ are defined as

$$\begin{aligned} dest'_j(n_6, n_8) &= dest(n_6, j) - dest(n_8, j) = \{n_6, n_7, n_8\} - \{n_8\} = \{n_6, n_7\}, \\ dest'_j(n_6, n_7) &= dest(n_6, j) - dest(n_7, j) = \{n_6, n_7, n_8\} - \{n_7\} = \{n_6, n_8\}. \end{aligned}$$

Let $BL(i)$ be the maximum time duration from start time of a task in $top(i)$ to finish time of END task. Then we define $BL(i)$ for each tasks in $out(i)$ as

$$BL(i) = \max_{n_k \in out(i)} \{S(n_k, i) + blevel(n_k)\}. \tag{5}$$

Let $level(n_k)$ be $level(n_k) = tlevel(n_k) + blevel(n_k)$, and then let $LV(i)$ be the maximum time duration from start time of one START task to finish time of END task, which includes execution time of any one task in $cls(i)$ as

$$LV(i) = \max_{n_k \in cls(i)} \{level(n_k)\} = TL(i) + BL(i). \tag{6}$$

Let WRT value be $WRT(G_{cls})$, which is the WRT of the clustered DAG. Then we define $WRT(G_{cls})$ as

$$WRT(G_{cls}) = \max_{cls(i)} \{LV(i)\}. \tag{7}$$

Therefore, the objective described in section 2.4 is expressed as

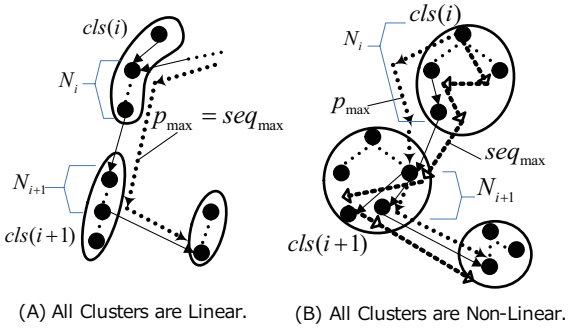

Fig. 3. Concept of Upper Bound of $WRT(G_{cls})$.

Objective 1 *Find* δ_{opt} *such that*

$$\sum_{n_j \in cls(i)} w(n_j) \geq \delta_{opt}, \min_{G_{cls}} \{WRT(G_{cls})\}. \tag{8}$$

4 Derivation of Cluster Size for Minimizing WRT

In this section, we analyze an upper bound of WRT derived under the condition that each cluster size must exceed δ. Then we derive δ_{opt}, by which WRT is minimized as much as possible.

4.1 Upper bound of WRT

If we assume a cluster structure generated by a task clustering, there are 2 possible structures, i.e., the one is a cluster in which execution order for each task is fixed, and the other is not. As described in section 2.2, the former case is said to linear, and let the latter case be named as non-linear, both of which are defined in [3]. Here, we analyze WRT of a clustered DAG with considering those structures.

Here, let ΔL_i be the increment of $LV(i)$ for each $cls(i)$ after a task clustering[1]. Let $\Delta WRT = WRT(G_{cls}) - WRT(G_{org})$, and let T_{max} be the maximum number of tasks in each path of G_{org}. For simplicity, let w_{max} be the maximum task size in G_{org}. Fig. 4 shows definitions used in this section. sec_{max} is the set of tasks and edges, by which $WRT(G_cls)$ is derived. In sec_{max}, p_{max} is the set of tasks and edges in which every task has dependencies each other. Here, we assume 3 types of the clustered DAG in which WRT is determined, i.e., (A)

[1] At the initial DAG, each cluster includes only one task. Thus, $LV(i)$ for each $cls(i)$ in the initial DAG is the maximum path length from START task to END task, including $cls(i) = \{n_i\}$. Therefore, $WRT(G_{org})$ equals to critical path length of the initial DAG.

every cluster which determine WRT is linear, (B) no linear cluster exists, (c) some clusters is linear and other clusters are non-linear.

(A) Every cluster, which includes one or more tasks on p_{max}, is linear ((A) at Fig. 4). For each task in $cls(i)$, let N_i be the set of tasks and edges in which every task has dependencies each other. Then we obtain the following equation.

$$\Delta L_i = -\sum_{e_{k,l} \in N_i} c(e_{k,l}). \tag{9}$$

Here, as DAG granularity is defined in [3], we define g_{max} as

$$\begin{aligned} g_{\max} &= \max\left\{ \max_{n_k \in V} \{g_{pred}(n_k)\}, \max_{n_k \in V} \{g_{suc}(n_k)\} \right\}, \\ g_{pred}(n_k) &= \max_{n_j \in pred(n_k)} \left\{ \frac{w(n_k)}{c(e_{j,k})} \right\},\ g_{suc}(n_k) = \max_{n_l \in suc(n_k)} \left\{ \frac{w(n_k)}{c(e_{k,l})} \right\}. \end{aligned} \tag{10}$$

By using (10), we obtain

$$\begin{aligned} \Delta L_i &\leq -\left(\sum_{n_k, n_l \in N_i} \frac{c(e_{k,l}) w(n_k)}{w(n_k)} - \frac{w_{\max}}{g_{\max}} \right) \\ &\leq -\left(\sum_{n_k \in N_i} \frac{c(e_{k,l}) w(n_k)}{w(n_k)} - \frac{c(e_{m,END_i}) w(n_{END_i})}{w(n_{END_i})} \right) \\ &\leq \frac{1}{g_{\max}} \left(w_{\max} - \sum_{n_k \in N_i} w(n_k) \right), \end{aligned} \tag{11}$$

where $n_m \in pred(n_{END_i})$, and n_{END_i} is $btm(i)$, which includes only one task in (A). Since the lower bound of the number of tasks inclued in each cluster is δ / w_{max}, we define the number of tasks in each cluster as ϕ. Then we obtain

$$\phi \leq \frac{T_{max} w_{max}}{\delta}. \tag{12}$$

By summing ΔL_i for each cluster which is one part of the path of $WRT(G_{cls})$, the following inequality is obtained.

$$\begin{aligned} \Delta WRT &= \sum_{N_i \in p_{max}} \Delta L_i \\ &\leq \frac{1}{g_{\max}} \sum_{N_i \in p_{max}} \left(w_{\max} - \sum_{n_k \in N_i} w(n_k) \right). \end{aligned} \tag{13}$$

Let sum of task size in N_i be $w(N_i)$ and let the minimum summed value of task size in one path be $\min\{w_{path}\}$. By applying (12) to (13), we obtain

$$\Delta WRT \leq \frac{1}{g_{\max}} \left(\frac{w_{\max}^2 T_{\max}}{\min\{w(N_i)\}} - \min\{w_{path}\} \right). \tag{14}$$

As a special case in (14), if every task in each cluster is included in p_{max}, i.e., if $\min\{w(N_i)\} \geq \delta$, we obtain

$$\Delta WRT \leq \frac{1}{g_{\max}} \left(\frac{w_{\max}^2 T_{\max}}{\delta} - \min\{w_{path}\} \right). \tag{15}$$

From (15), if every cluster in p_{max} is linear and every task in such a cluster belongs to p_{max}, it can be said that WRT is suppressed if δ is large value.

(B) Every cluster which includes one ore more tasks in p_{max}, is not linear. Here, we assume that each cluster generation procedure must be completed when a cluster size exceeds B, otherwise we can not estimate an upper bound of each cluster size. Under the constraint, the fact that each cluster size is less or equal to $\delta + w_{max}$, is obtained. Therefore, the upper bound of ΔL_i in (B) is expressed as

$$\begin{aligned} \Delta L_i &\leq (\delta + w_{\max}) - \sum_{n_k \in N_i} w(n_k) - \sum_{e_{p,q} \in N_i} c(e_{p,q}) \\ &\leq (\delta + w_{\max}) - \sum_{w_k \in N_i} w(n_k) - \frac{1}{g_{\max}} \left(\sum_{n_k \in N_i} w(n_k) - w_{\max} \right) \\ &= (\delta + w_{\max}) - \sum_{n_k \in N_i} w(n_k) \left(1 + \frac{1}{g_{\max}} \right) + \frac{w_{\max}}{g_{\max}}. \end{aligned} \tag{16}$$

Here, let the number of clusters each of which includes one or more tasks in p_{max} as ϕ, we obtain

$$\begin{aligned} \Delta WRT &= \sum_{N_i \in p_{max}} \Delta L_i \\ &\leq \sum_{N_i \in p_{max}} (\delta + w_{\max}) - \sum_{N_i \in p_{max}} \sum_{n_k \in N_i} w_k \left(1 + \frac{1}{g_{\max}} \right) + \sum_{N_i \in p_{max}} \frac{w_{\max}}{g_{\max}} \\ &\leq \phi \left\{ \delta + w_{\max} \left(1 + \frac{1}{g_{max}} \right) \right\} - \min\{w_{path}\} \left(1 + \frac{1}{g_{max}} \right). \end{aligned} \tag{17}$$

From (17), in this case, the larger δ becomes, the longer WRT becomes.

(C) In all cluster which includes tasks in p_{max}, some clusters are linear and others are non-linear. Since the number of linear clusters is less than $T_{max} w_{max}/\delta$, here, let the number of them be $(T_{max} w_{max}/\delta) - y$ and let the number of non-linear clusters be y. From (11) and (16), we obtain the upper bound of ΔWRT as

$$\begin{aligned} \Delta WRT \leq \frac{1}{g_{\max}} \left(\frac{w_{\max} T_{\max}}{\delta} - y \right) w_{\max} - \frac{1}{g_{\max}} \sum_{n_k \in N_i} w(n_k) \\ + y \left\{ \delta + w_{\max} \left(1 + \frac{1}{g_{\max}} \right) \right\} - y \sum_{n_k \in N_i} w(n_k) \left(1 + \frac{1}{g_{max}} \right). \end{aligned} \tag{18}$$

If we differentiate (18) with respect to δ, we obtain

$$\delta = w_{\max}\sqrt{\frac{T_{\max}}{g_{\max}y}}. \tag{19}$$

From (19), it can be said that there is a minimal value in ΔWRT.

4.2 Derivation of δ_{opt}

From (15), ΔWRT is monotonically decreasing in (A). On the other hand, from (17), ΔWRT is monotonically increasing in (B). If structure of each cluster changes from linear to non-linear by a task clustering heuristic, we can say that the minimum WRT can be obtained during (C). Here, to achieve static and automatic determination of an appropriate lower bound of cluster size, we derive δ_{opt} by which WRT is minimized as much as possible.

In (19), since $y \geq 1$, the range of δ for minimizing WRT as much as possible is defined as

$$\delta \leq w_{\max}\sqrt{\frac{T_{\max}}{g_{\max}}}. \tag{20}$$

Since it is required that δ must be large in order to suppress the number of cluster, we obtain δ_{opt} as

$$\delta_{opt} = w_{\max}\sqrt{\frac{T_{\max}}{g_{\max}}}. \tag{21}$$

4.3 Requirements for Minimizing WRT in a Task Clustering

From results in section 4.2, requirements for a task clustering heuristic in order to obey the state transition described in 4.2 are defined as follows.

1. Merge tasks in order to generate a cluster until the cluster size exceeds δ_{opt}.
2. A unmerged task, which has dependencies with one of a merged task in a cluster, must be chosen in order to generate a linear cluster.
3. If no task which is required for generating a linear cluster does not exist, a task which is required for suppressing the increase of WRT must be chosen.

In [9], an exemplary clustering heuristic, which obeys requirement 2 and 3, is described. The heuristic in [9] performs merging each task until each cluster size exceeds a manually defined δ. Therefore, by applying (21) into the heuristic, requirement 1 is satisfied.

5 Experimental Evaluation

In this section, we present results of experimental comparison in terms of WRT and response time among several task clustering heuristics, i.e., the one which combines [9] and the requirements presented in section 4.3, and other existing approaches.

5.1 Simulation Environment

The objective of the evaluation is to clarify the relationship between minimizing WRT and response time. In this evaluation, we derive WRT and response time obtained by each approaches while changing both size and structure of initial DAG. The number of tasks, i.e., $|V|$ is respectively set to 500 and 1000, in order to confirm whether scale of the problem has effect on an obtained response time or not. P, which is the number of successor tasks of each task, is varied. We also changed R, the value of the ratio of data transfer time to task execution time. δ_{opt} is derived as (21) as a lower bound of each cluster size.

Existing approaches to be compared are categorized as two types. The first type is "two step merging"(task clustering and cluster merging), e.g., CASS-II+LB[8][5], DSC+CM(Cluster Merging)[2][4]. Both LB and CM perform merging tasks in order to equalize each cluster size. One difference between them is whether every task in each cluster has dependencies to its immediate predecessor tasks which belong to the same cluster or not. LB adopts such a policy and CM does not, so that every task in a cluster in case of CM may have no dependency to every task in a cluster. The other type is "one step merging", e.g., LB[5] and a task clustering heuristic presented in [9] with satisfying the constraint that each cluster size is above δ_{opt}(hereinafter, we denote it as "[9] with δ_{opt}"). We derived both WRT after each approach finishes. Then we derived response time for each approach by applying Ready Critical Path(RCP) scheduling[7]. RCP is one of list scheduling heuristic, whose characteristic is to impose priority as a path length from START task to each ready task. Then the task which has the minimum priority is scheduled. Therefore, one characteristic of RCP is to reduce a possible start time for each task.

Both of WRT and response time are evaluated with equalizing the number of clusters in those approaches. The number of clusters (here, we denote N) is firstly derived by applying "[9] with δ_{opt}", and then other approaches perform merging steps until the number of clusters reaches N. And both WRT and response time to be compared among those approaches are mean values in 100 tries. In each try, the simulation generates a random DAG after P value and R value have been defined.

5.2 Comparison of WRT

The comparison in terms of WRT is presented in table 1. In the table, variables are $|V|$, P, and R. The fourth column means the number of clusters obtained by "[9] with δ_{opt}". The remained columns are the ratio of WRT to that of "[9] with δ_{opt}". Lower value means that WRT is efficiently suppressed. From this table, it can be seen that "[9] with δ_{opt}" efficiently suppress WRT with compared to other approaches in every cases (From No. 1-12). The difference of WRT between "[9] with δ_{opt}" and CASS-II+LB becomes small as size of data among tasks is large and degree of task parallelism is low (No. 1-3, 7-9). The same behaviors are applied to the case of LB. On the contrary, The difference of WRT between "[9] with δ_{opt}" and DSC+CM becomes small as size of data among tasks in every case (No. 1-12), regardless of degree of task parallelism. However, both LB and

CM do not have any policy for suppressing WRT. Therefore, it is concluded that an approach which adopts LB or CM does not suppress than an approach which adopts the three requirements described in section 4.

No.	♯ of Tasks	Range of R	Range of P	♯ of Clusters	[9] with δ_{opt}	CASS-II+LB	DSC+CM	LB
1	500	0.5≤R≤ 1.0	1≤P≤2	73	1.000	1.637	1.868	1.602
2	500	4.0≤R≤8.0	1≤P≤2	28	1.000	1.426	1.547	1.530
3	500	8.0≤R≤16.0	1≤P≤2	21	1.000	1.304	1.510	1.553
4	500	0.5≤R≤ 1.0	1≤P≤8	58	1.000	1.256	1.592	1.447
5	500	4.0≤R≤8.0	1≤P≤8	20	1.000	1.657	1.492	1.690
6	500	8.0≤R≤16.0	1≤P≤8	15	1.000	1.732	1.377	1.768
7	1000	0.5≤R≤ 1.0	1≤P≤2	133	1.000	1.908	1.971	1.812
8	1000	4.0≤R≤8.0	1≤P≤2	52	1.000	1.403	1.556	1.573
9	1000	8.0≤R≤16.0	1≤P≤2	38	1.000	1.347	1.470	1.528
10	1000	0.5≤R≤ 1.0	1≤P≤8	103	1.000	1.210	1.642	1.393
11	1000	4.0≤R≤8.0	1≤P≤8	38	1.000	1.684	1.497	1.654
12	1000	8.0≤R≤16.0	1≤P≤8	27	1.000	1.724	1.317	1.718

Table 1. Comparison of WRT

5.3 Comparison of Response Time

The comparison in terms of response time is presented in table 2. Values in 4 approaches are the ratio between response time obtained by the approach to that of "[9] with δ_{opt}". In every cases (No. 1-12), "[9] with δ_{opt}" is smaller than other approaches. If size of data among tasks is small (e.g., No. 1, 4, 7 and 10), difference of response time between "[9] with δ_{opt}", CASS-II+LB and DSC+CM is small. On the other hand, if data size among tasks is large and degree of task parallelism is low (e.g., No. 3, 9), "[9] with δ_{opt}" efficiently suppresses response time. In other approaches, i.e., CASS-II+LB, DSC+CM and LB, difference of response time to "[9] with δ_{opt}" becomes larger as data size among tasks becomes larger, except the case of CASS-II+LB in No. 12. From these results, it is concluded that an approach which adopts the three requirements described in section 4.3 has good effect on response time when the input DAG has large data size among tasks.

No.	♯ of Tasks	Range of R	Range of P	♯ of Clusters	[9] with δ_{opt}	CASS-II+LB	DSC+CM	LB
1	500	0.5≤R≤ 1.0	1≤P≤2	73	1.000	1.025	1.050	1.093
2	500	4.0≤R≤8.0	1≤P≤2	28	1.000	1.249	1.292	1.707
3	500	8.0≤R≤16.0	1≤P≤2	21	1.000	1.388	1.576	2.133
4	500	0.5≤R≤ 1.0	1≤P≤8	58	1.000	1.133	1.034	1.300
5	500	4.0≤R≤8.0	1≤P≤8	20	1.000	1.173	1.167	1.768
6	500	8.0≤R≤16.0	1≤P≤8	15	1.000	1.192	1.177	1.878
7	1000	0.5≤R≤ 1.0	1≤P≤2	133	1.000	1.039	1.034	1.113
8	1000	4.0≤R≤8.0	1≤P≤2	52	1.000	1.267	1.388	1.780
9	1000	8.0≤R≤16.0	1≤P≤2	38	1.000	1.389	1.555	2.155
10	1000	0.5≤R≤ 1.0	1≤P≤8	103	1.000	1.135	1.091	1.280
11	1000	4.0≤R≤8.0	1≤P≤8	38	1.000	1.169	1.137	1.720
12	1000	8.0≤R≤16.0	1≤P≤8	27	1.000	1.030	1.158	1.881

Table 2. Comparison of Response Time

5.4 Discussion

From results obtained by table 1 and 2, it is concluded that a task clustering heuristic which adopts the three requirements described in section 4.3 has a good effect on both WRT and response time, if lower bound of cluster size has appropriately been determined. On the other hand, when data size is small with compared to task size, response time is closed to other existing approaches. In such a case, localizing each data transfer among tasks has smaller effect on reducing response time, because decrease of task parallelism which may be occurred by each merging step may lead to increase of response time. At other approaches, e.g., CASS-II+LB and DSC+CM neglect lower bound of cluster size, so that each cluster size may vary according to their merging policies and structure of the input DAG. As a result, such a variation in each cluster size may not have bad effect on communication overhead, while maintaining degree of task parallelism due to a small cluster size. We conclude that three requirements described in section 4.3 is an advantageous policy, if the required number of machines is unknown despite machines must effectively be utilized.

Next we explain relationship between minimizing WRT and response time obtained after a task scheduling. In a derivation of *tlevel* value in (3) for each *in* task in a cluster, waiting time (idle time) for data arrival from its immediate predecessor tasks in different clusters is neglected. Under the constraint, $tlevel(n_k)$ is the latest start time of n_k when every task which has no dependencies to n_k has been executed. Therefore, if data transfer time is small with compared to task execution time in a DAG, there is possibility that a data has already been arrived before $tlevel(n_k)$. In this case, $tlevel(n_k)$ equal to the actual latest start time of n_k, so that minimizing WRT may become synonymous with an upper bound of actual response time. Since RCP[7] which applied in this simulation tries to minimize start time for each task, trying to minimize waiting time for data arrival is well matched to RCP's scheduling strategy. As a result, we conclude that a task scheduling, which adopts the strategy to minimize start time for each task, is required for minimizing response time.

6 Conclusion and Future Work

In this paper, we presented one method for adjusting each assignment unit size, i.e., cluster size in order to minimize response time with the small number of machines in homogeneous distributed systems. To make the number of machines small, we derived a lower bound of cluster size, i.e., δ_{opt} by which an upper bound of response time is minimized as much as possible. Then we showed requirements for a task clustering heuristic in order to suppress increase of an upper bound of response time. One contribution in this paper is to present an approach for achieving efficient utilization of limited computational resources for parallel application. Our future work is to derive a lower bound of each cluster size in heterogeneous systems.

References

1. A. Gerasoulis, 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, 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.
3. 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.
4. T. Yang, A. Gerasoulis., PYRROS: Static scheduling and code generation for message passing multiprocessors, *Procs. of the 6th ACM International Conference on Supercomputing*, pp. 428 – 437, 1992.
5. J. C. Liou, M. A. Palis., A Comparison of General Approaches to Multiprocessor Scheduling, *Procs. of the 11th International Symposium on Parallel Processing*, pp. 152 – 156, 1997.
6. V. Sarkar, Partitioning and Scheduling Parallel Programs for Execution on Multiprocessors, Cambridge, *MA: MIT Press*, 1989.
7. T. Yang, A. Gerasoulis, List scheduling with and without communication delays, *Parallel Computing*, pp. 1321 – 1344, 1993.
8. 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.
9. Hidehiro Kanemitsu, Yi Lu, Yoshihiro Otani, Gilhyo Lee, Hidenori Nakazato, Takashige Hoshiai, and Yoshiyori Urano, A Task Clustering Minimizing Worst Response Time in Distributed Processing Environment, *IEICE Technical Reports*, vol. 108, no. 361, pp. 13-18, December 2008.
10. M. Y. Wu and D. D. Gajski, Hypertool: A programming aid for message-passing systems, *IEEE Trans. on Parallel and Distributed Systems*, vol. 1, No. 3, pp. 330–343, 1990.
11. Albert Y. Zomaya and Gerard Chan, Efficient clustering for parallel tasks execution in distributed systems, *Procs. of the 18th International Parallel and Distributed Processing Symposium (IPDPS'04)*, pp. 167-175, Santa Fe, NM, USA, April 2004.