PAPER

Scheduling Parallel Tasks with Communication Overhead in an Environment with Multiple Machines

Jiann-Fu LIN[†a)], ***Member***

SUMMARY This paper investigates the problem of nonpreemptively scheduling independent parallel tasks in an environment with multiple machines, which is motivated from the recent studies in scheduling tasks in a multi-machine environment. In this scheduling environment, each machine contains a number of identical processors and each parallel task can simultaneously require a number of processors for its processing in any single machine. Whenever tasks are processed in parallel in a parallel machine, message communication among processors is often inevitable. The problem of finding a shortest schedule length on scheduling independent parallel tasks with the consideration of communication overhead in a multi-machine environment is NP-hard. The aim of this paper is to propose a heuristic algorithm for this kind of problem and to analyze the performance bound of this heuristic algorithm.

key words: *parallel task scheduling, multi-machine, communication overhead, performance bound*

1. Introduction

The conventional tasks scheduling approach is that tasks are to be scheduled in a single machine with multiple identical processors, and that each task is processed only on one processor at a time. The conventional tasks scheduling problems have been extensively studied for decades, and these problems have evolved to become parallel tasks scheduling problems [3], [7], [8], [14], [15]. The difference between the parallel task and conventional task is that a parallel task can simultaneously require a number of processors for its processing, under the constraint that the number of processors required cannot be greater than a given maximum degree of parallelism of that task, while the conventional task only requires one processor at a time. The parallel task scheduling problem considers a set of n independent parallel tasks $\boldsymbol{T} = \{T_1, T_2, \ldots, T_n\}$ that are to be scheduled in a scheduling environment. Each task T_i, $i = 1, 2, \ldots, n$, has its computation requirement t_i and associates with a maximum degree of parallelism Δ_i. The maximum degree of parallelism of task T_i means that task T_i may be scheduled to process on up to Δ_i processors, and this degree of parallelism, once determined for T_i, will not be altered during its processing. Suppose that a task T_i is scheduled to be processed on δ_i processors, $1 \le \delta_i \le \Delta_i$, δ_i is called the degree of scheduled parallelism of T_i and the processing time required by T_i, under the linear speedup assumption, will be (t_i/δ_i). One of the goals of the above tasks scheduling problems is to find an optimal schedule, a schedule with minimum length (makespan), which is known as an NP-hard problem [3]. Hence, polynomial time heuristic scheduling algorithms are usually used to get approximate solutions. A heuristic scheduling algorithm H has a performance bound of β, if $(S_H/S_{OPT}) \le \beta$ for all problem instances, where S_H and S_{OPT} denote the length of a heuristic schedule and that of an optimal schedule, respectively.

Under the linear speedup assumption, Wang and Cheng [14] applied the concept of Graham's *List Scheduling* (LS) algorithm [6] to the parallel task scheduling problem in a single machine with P processors, and showed that the performance bound is $\left(\Delta + \frac{P-\Delta}{P}\right)$, where $\Delta = \max\{\Delta_i \mid i = 1, 2, \ldots, n\}$. They also proposed the *Earliest Completion Time* (ECT) algorithm [15] for the same scheduling problem and derived the performance bound as $\left(3 - \frac{2}{P}\right)$. Then, Lin, See and Chen [8] also discussed the problem of scheduling parallel tasks in a single machine, but with communication overhead among processors also taken into consideration. They proposed the *Largest Scheduled Parallelism First* (LSPF) algorithm and derived the performance bounds as $\left(4 + \frac{2}{k} - \frac{2}{P}\right)$ and $\left(2 + \ln 2^d - \frac{1+\ln 2^d}{2^d}\right)$ on a P-processor machine and on a d-dimension hypercube machine, respectively, where k is a given positive constant.

The above studies only discussed the problems of scheduling independent parallel tasks on a single machine with multiple identical processors. With the vast improvements in wide-area network performance and powerful computers, the grid environment has emerged as a promising computing platform that can support the execution of next generation scientific applications, and will open up avenues in many research fields [5], [13]. Grid environment is a large virtual organization that integrates a large amount of distributed resources and high performance computing capabilities into a super service, which can provide huge computing services, storage capability and so on. Grid environment allows the use of geographically distributed computing systems belonging to multiple organizations as a single system. Thus, for simplicity, a grid environment can be seen as a multi-machine environment in which each machine contains multiple processors. In the multi-machine environment, users submit their tasks from any one of machine and a scheduler allocates tasks to machines. In general, on receipt of a task request, the scheduler check whether the task can be processed on the available resources and meet the

Manuscript received March 4, 2008.
Manuscript revised June 12, 2008.

[†]The author is with the Department of Management Information System, Takming University of Science and Technology, No. 56, Sec. 1, HuanShan Rd., NeiHu, Taipei, 11451, Taiwan.
a) E-mail: alfu@takming.edu.tw

DOI: 10.1093/ietisy/e91–d.10.2379

user-specified requirements. Thus, instead of computing locally, users dispatch their tasks to the remote machines. To achieve the potentials of a multi-machine environment, an effective and efficient scheduling framework within a multi-machine environment is fundamentally important.

In 2001, Braun et al. [2] made a performance comparison among eleven heuristic algorithms for scheduling a set of independent tasks onto heterogeneous distributed computing systems by simulation. In 2004, Martino and Mililotti [9] developed a simulation grid environment to study the usefulness of genetic algorithms for scheduling tasks in a distributed group of parallel machines. They found that the genetic algorithm for scheduling 32 tasks does not converge to the optimal schedule within a limited number of trial performed, only a sub-optimal schedule could be got. In 2005, Weng and Lu [16] proposed a heuristic to schedule independent tasks in the grid environment. According to the experimental results, they showed that their heuristic algorithm could obtain a better performance compared to the other four existing algorithms. In 2007, Pascual, Rzadca and Trystram [10] discussed the problem of scheduling rigid parallel tasks (multiprocessor tasks [1]) in a grid environment, they proposed the Multi-Organization Load Balancing Algorithm (MOLBA) and derived the worst case performance bound as 3 if the last completed task requires at most half of the available processors, and 4 in the general case.

Due to the reasons that tasks scheduling are the important components in a grid environment, a multi-machine environment, and that we found the proof of Lemma 1 in [8] is not correct, we will extend the problem of scheduling independent parallel tasks with the consideration of communication overhead to a multi-machine environment and analyze its performance bound on such an environment. This problem of scheduling parallel tasks with communication overhead in a multi-machine environment, in which each machine contains a number of identical processors, is NP-hard, because scheduling independent parallel tasks without considering communication overhead in a single machine with multiple processors has been known as NP-hard [3]. Therefore, we are interested in developing a polynomial time heuristic algorithm for such a problem, and in deriving its performance bound. The rest of this paper is organized as follows: Under a communication overhead assumption, we illustrate a strategy for deciding a beneficial degree of parallelism of each task in Sect. 2. In Sect. 3, we modify the *Largest Scheduled Parallelism First* algorithm into the *Modified Largest Scheduled Parallelism First* algorithm (MLSPF) for the problem of scheduling parallel tasks in a multi-machine environment, and derive its performance bound. Finally, some concluding remarks are given in Sect. 4.

2. Preliminaries

2.1 Scheduling Environment and Task Model

In this paper, we consider that a set of n independent parallel tasks $\boldsymbol{T} = \{T_1, T_2, \ldots, T_n\}$ are to be scheduled in an M-machine environment. In this tasks scheduling environment, each machine m_a consists of p_a identical processors and each parallel task T_i can only be processed in a single machine with its required processors simultaneously, where $1 \le a \le M$ and $1 \le i \le n$. Each parallel task is assumed to be processable in any machine with its maximum degree of parallelism, that is, $\max\{\Delta_i \mid i = 1, 2, \ldots, n\} \le \min\{p_a \mid a = 1, 2, \ldots, M\}$. Whenever a task is processed in parallel in a machine, communication overhead is an important factor that has great effect on the total processing time of a task. Thus, if communication overhead is taken into consideration, a linear speedup is hard to achieve. Generally, communication overhead depends on the characteristics of a task and a system, the degree of parallelism adopted by a task, and the topology of processors allocated to a task [4], [11], [12]. In most situations, the average communication overhead will be increased with the degree of parallelism adopted by a task [12]. In [4], the total communication overhead is computed as (total number of messages×communication delay per message), in which the total number of messages is defined as the number of processor pairs. Namely, the total number of messages is proportional to the power of the degree of parallelism adopted by a task. As a consequence, a reasonable assumption of the average communication overhead among subtasks of a task is given as $comm(x) = cx^k$ [8], where x is the degree of parallelism adopted by a task, and c and k are positive constants which depend on the characteristics of the interconnection networks among processors.

In this paper, we adopt the communication overhead assumption in [8], and we only consider a specific condition in which the characteristics of each machine are all the same except the number of processors, say, the interconnection network in each machine is fully connected, and the transmission media in each machine is the same. That is, the same constants c and k are used in calculating the communication overhead of each task when it is processed in any machine. Thus, under the communication overhead assumption of [8], the total time required for processing a task T_i with a degree of parallelism x in any machine can be defined as

$$f(x, t_i) = \begin{cases} t_i, \text{ if } x = 1. \\ \dfrac{t_i}{x} + cx^k, \text{ if } x > 1. \end{cases}$$

2.2 The Beneficial Scheduled Parallelism

Since the communication overhead is taken into consideration, a highest degree of parallelism does not guarantee that it always leads to the shortest processing time. Though a higher degree of parallelism will reduce the computation time of a task, a higher degree of parallelism will also incur much more message communicating among processors. In order to get the shortest total processing time of task T_i, we have to find out the minimum value of $f(x, t_i)$. By calculus, the first derivative and the second derivative of $f(x, t_i)$ are

$$\frac{\partial f(x,t_i)}{\partial x} = \begin{cases} 0, \text{ if } x = 1. \\ \dfrac{-t_i}{x^2} + ckx^{k-1}, \text{ if } x > 1. \end{cases}$$

and

$$\frac{\partial^2 f(x,t_i)}{\partial x^2} = \begin{cases} 0, \text{ if } x = 1. \\ \dfrac{2t_i}{x^3} + ck(k-1)x^{k-2}, \text{ if } x > 1. \end{cases}$$

respectively.

Owing to $\frac{\partial f(x,t_i)}{\partial x} = 0$ and $\frac{\partial^2 f(x,t_i)}{\partial x^2} > 0$ at $x = \left(\frac{t_i}{ck}\right)^{1/(k+1)}$, $f(x,t_i)$ reaches its minimum value at $x = \left(\frac{t_i}{ck}\right)^{1/(k+1)}$. Since the number of processors required by a task must be an integer, the beneficial degree of parallelism ϕ_i of task T_i will be either $\left\lfloor \left(\frac{t_i}{ck}\right)^{1/(k+1)} \right\rfloor$ or $\left\lceil \left(\frac{t_i}{ck}\right)^{1/(k+1)} \right\rceil$, which makes $f(\phi_i, t_i)$ have the smallest value. However, the degree of parallelism really adopted by task T_i cannot be greater than the given maximum degree of parallelism Δ_i. Hence, the degree of scheduled parallelism δ_i of task T_i is given as $\min(\phi_i, \Delta_i)$. The procedure of deciding the degree of scheduled parallelism of each task is described as follows.

Procedure scheduled-parallelism

{ Input the computation requirement t_i and the maximum degree of parallelism Δ_i of task T_i, where $i = 1, 2, \ldots, n$;
For $i = 1$ **to** n do
{ Let ϕ_i be either $\left\lfloor \left(\frac{t_i}{ck}\right)^{1/(k+1)} \right\rfloor$ or $\left\lceil \left(\frac{t_i}{ck}\right)^{1/(k+1)} \right\rceil$, which makes $f(\phi_i, t_i)$ have the smallest value;
Let the degree of scheduled parallelism of task T_i be $\delta_i = \min(\phi_i, \Delta_i)$; }
}

According to the degree of scheduled parallelism δ_i of each task T_i determined by the scheduled-parallelism procedure, the task set $\boldsymbol{T} = \{T_1, T_2, \ldots, T_n\}$ is categorized into δ task subsets: $T^{\delta} = \{T_1^{\delta}, T_2^{\delta}, \ldots, T_{n_\delta}^{\delta}\}$, $T^{\delta-1} = \{T_1^{\delta-1}, T_2^{\delta-1}, \ldots, T_{n_{\delta-1}}^{\delta-1}\}, \ldots$, and $T^1 = \{T_1^1, T_2^1, \ldots, T_{n_1}^1\}$, where $\delta = \max\{\delta_i \mid i = 1, 2, \ldots, n\}$ and $n = \sum_{j=1}^{\delta} n_j$. This means that $\boldsymbol{T} = \bigcup_{i=1}^{n} T_i = \bigcup_{j=1}^{\delta} \bigcup_{r=1}^{n_j} T_r^j$. In other words, there is a bijective function G that maps $\{T_1, T_2, \ldots, T_n\}$ to $\{T_1^{\delta}, T_2^{\delta}, \ldots, T_{n_\delta}^{\delta}, T_1^{\delta-1}, T_2^{\delta-1}, \ldots, T_1^1, \ldots, T_{n_1}^1\}$. Each task T_i is categorized into only one task subset, according to its degree of scheduled parallelism, which turns out to be task T_r^j. Thus, we can assume that the computation requirement $t_i = t_r^j$, the maximum degree of parallelism $\Delta_i = \Delta_r^j$ and the scheduled parallelism $\delta_i = j$, where $1 \le i \le n$, $1 \le j \le \delta$ and $1 \le r \le n_j$. After categorization, each task T_r^j in the task subset T^j means its degree of scheduled parallelism is j and its total processing time is $f(j, t_r^j)$.

3. The Modified Largest Scheduled Parallelism First Scheduling Algorithm

As we have briefly mentioned in Sect. 1, the proof of Lemma 1 in [8] is not correct. The incorrectness originates from the improper depicted of Fig. 1 [8], which is a special case of all possible schedules. Figure 1 [8] illustrated that the maximum number of idle processors is $(j-2)$ between the time at which the first task in T^j is started and the time at which tasks in T^j have all been finished, however, it is possible to be $(j-1)$. Besides, the number of idle regions is not necessarily equal to $\lceil m/j \rceil$, the number of tasks in T^j also has an effect on it, for example there is only one task in T^j.

Due to the reasons that the proof of Lemma 1 in [8] is incorrect and that tasks scheduling are the important component in a multi-machine environment, we modify the LSPF algorithm [8] and propose the *Modified Largest Scheduled Parallelism First* (MLSPF) scheduling algorithm for nonpreemptively scheduling independent parallel tasks in an M-machines environment and will analyze its performance bound on such an environment. The major policy of MLSPF is that, according to the degrees of scheduled parallelism of tasks, tasks are categorized into several task subsets and these task subsets are arranged in nonincreasing order. According to this nonincreasing order, tasks are assigned to any machine that has enough free processors for processing this task. That is, a task has higher priority for assignment if it has a larger degree of scheduled parallelism. The detail of the MLSPF algorithm is described as follows.

Algorithm MLSPF

{ **Call** scheduled-parallelism procedure;
According to the degrees of scheduled parallelism, $T_1, T_2, \ldots$, and T_n are divided into δ task subsets $T^{\delta} = \{T_1^{\delta}, T_2^{\delta}, \ldots, T_{n_\delta}^{\delta}\}$, $T^{\delta-1} = \{T_1^{\delta-1}, T_2^{\delta-1}, \ldots, T_{n_{\delta-1}}^{\delta-1}\}, \ldots$, and $T^1 = \{T_1^1, T_2^1, \ldots, T_{n_1}^1\}$;
For $j = \delta$ **to** 1 do
For $r = 1$ **to** n_j do
{ Wait until there exists a machine m_a that has at least j free processors, where $1 \le a \le M$;
Machine m_a allocates j processors to task T_r^j for execution; }
}

While the MLSPF algorithm schedules tasks to machines, it is possible that some processors in a machine are not allocated to any task because the number of free processors for allocation at that moment is smaller than the degree of the scheduled parallelism of a task which is next to be assigned. Hence, there are some processors idle during the period of the MLSPF schedule. In order to calculate the total idle time of processors in each machine, ST_a^u and FT_a^u are defined as the time at which a task in T^u is first started in the machine m_a, and the time at which those tasks in T^u assigned in machine m_a have all been finished, respectively, where $1 \le a \le M$ and $1 \le u \le \delta$. We cannot make sure whether the next task set to be scheduled after scheduling T^u is T^{u-1}, therefore, T^v is assumed to be the task set which is next to be scheduled after T^u, where $1 \le v < u \le \delta$. In addition, let T_z^w be the task finished at time S_{MLSPF} and A_a^u denote the sum of idle time of each processor between ST_a^u and ST_a^v in the machine m_a, where S_{MLSPF} represents

the finish time of an MLSPF schedule, and $1 \le w \le \delta$. A reasonable assumption is that tasks are to be scheduled from time 0. Thus, the finish time of the MLSPF schedule is also the length of the MLSPF schedule.

Lemma 1: $ST_a^u \le ST_a^v \le FT_a^u$, where $1 \le a \le M$ and $1 \le v < u \le \delta$.

Proof: (1) According to the MLSPF scheduling rule, the first task in T^v can be started in the machine m_a only when all tasks in T^u have been assigned. Then, $ST_a^u \le ST_a^v$.
(2) Since $v < u$ and, according to the definition of FT_a^u, there must be at least u free processors in the machine m_a at time FT_a^u, then any task in T^v can be started at that time. Therefore, it is impossible that ST_a^v is greater than FT_a^u. □

Lemma 2: The number of idle processors in the machine m_a between ST_a^u and ST_a^v is smaller than u, where $1 \le a \le M$.

Proof: The number of idle processors in the machine m_a at the time τ, $ST_a^u \le \tau < ST_a^v$, must be smaller than u; otherwise, a task in T^u could be started at time τ. □

Lemma 3: $\sum_{a=1}^{M} A_a^u \le (u-1) \sum_{a=1}^{M} (FT_a^u - ST_a^u)$, where $1 \le u \le \delta$.

Proof: A_a^u is defined as the sum of idle time of each processor between ST_a^u and ST_a^v in the machine m_a. Then, according to Lemma 2 and the definition of A_a^u, $A_a^u \le (u-1)(ST_a^v - ST_a^u)$. By Lemma 1, $A_a^u \le (u-1)(FT_a^u - ST_a^u)$. Thus, $\sum_{a=1}^{M} A_a^u \le (u-1) \sum_{a=1}^{M} (FT_a^u - ST_a^u)$. □

Lemma 4: $\sum_{a=1}^{M} (FT_a^u - ST_a^u) \le \sum_{r=1}^{n_u} f(u, t_r^u)$, where $1 \le u \le \delta$, $f(u, t_r^u)$ is the total processing time of task T_r^u and n_u is the number of tasks in T^u.

Proof: $(FT_a^u - ST_a^u)$ can be seen as a schedule length of assigning partial tasks of T^u in the machine m_a. Then, it is obvious that the sum of these schedule lengths of scheduling partial tasks of T^u in each machine is not greater than the total processing time of all tasks in T^u, that is, $\sum_{a=1}^{M} (FT_a^u - ST_a^u) \le \sum_{r=1}^{n_u} f(u, t_r^u)$. □

Lemma 5: $\sum_{a=1}^{M} A_a^u \le (u-1) \sum_{r=1}^{n_u} f(u, t_r^u)$, where $1 \le u \le \delta$.

Proof: By Lemma 3 and Lemma 4. □

Since task T_z^w is assumed to be the task finished at time S_{MLSPF}, the total processing time of task T_z^w is $f(w, t_z^w)$. Thus, $(S_{MLSPF} - f(w, t_z^w))$ is the starting time of T_z^w. For simplification, we denote $(S_{MLSPF} - f(w, t_z^w))$ as ψ.

Lemma 6: The number of idle processors in any machine m_a between the time $\min\{ST_a^w \mid a = 1, 2, \ldots, M\}$ and ψ is smaller than w, where $1 \le w \le \delta$.

Proof: According to the definition of ST_a^w, ψ is not smaller than $\min\{ST_a^w \mid a = 1, 2, \ldots, M\}$. Hence, there must be at most (w-1) free processors at that period; otherwise, T_z^w could be assigned earlier than ψ. □

Lemma 7: $(j-1) \times f(j, t_r^j) \le \left(1 + \frac{1}{k}\right) t_r^j$, where $1 \le j \le \delta$ and $1 \le r \le n_j$.

Proof: According to the scheduled-parallelism procedure, the degree of scheduled parallelism j, $j \le \Delta_r^j$, makes $f(j, t_r^j)$ have the shortest total processing time among all possible degrees of parallelism. However, the degree of scheduled parallelism j may be equal to $\left\lfloor \left(\frac{t_r^j}{ck}\right)^{1/(k+1)} \right\rfloor$, $\left\lceil \left(\frac{t_r^j}{ck}\right)^{1/(k+1)} \right\rceil$ or Δ_r^j.

(1) If $j = \left\lceil \left(\frac{t_r^j}{ck}\right)^{1/(k+1)} \right\rceil$, then it implies that $f(j, t_r^j) \le f(j-1, t_r^j)$.

$$\Rightarrow (j-1) \times f(j, t_r^j) \le (j-1) \times f(j-1, t_r^j)$$
$$\le (j-1) \times \left(\frac{t_r^j}{j-1} + c(j-1)^k \right)$$
$$\le t_r^j + c(j-1)^{k+1}$$
$$\le t_r^j + c \left(\left(\frac{t_r^j}{ck} \right)^{1/(k+1)} \right)^{k+1},$$

because $(j-1) \le \left(\frac{t_r^j}{ck}\right)^{1/(k+1)} \le j$.

$$\le \left(1 + \frac{1}{k}\right) t_r^j$$

(2) If $j = \left\lfloor \left(\frac{t_r^j}{ck}\right)^{1/(k+1)} \right\rfloor$ or $j = \Delta_r^j$, it implies that $j \le \left(\frac{t_r^j}{ck}\right)^{1/(k+1)}$.

Thus,

$$(j-1) \times f(j, t_r^j) \le j \times f(j, t_r^j)$$
$$\le (t_r^j + cj^{k+1})$$
$$\le (t_r^j + c \left(\left(\frac{t_r^j}{ck} \right)^{1/(k+1)} \right)^{k+1}$$
$$\le \left(1 + \frac{1}{k}\right) t_r^j$$ □

Lemma 8: $j \times f(j, t_r^j) \le \left(2 + \frac{1}{k}\right) t_r^j$.

Proof: $j \times f(j, t_r^j) = (j-1) \times f(j, t_r^j) + f(j, t_r^j)$.
By Lemma 7, $j \times f(j, t_r^j) \le \left(1 + \frac{1}{k}\right) t_r^j + f(j, t_r^j)$.
Since $f(j, t_r^j) \le t_r^j$, we have that $j \times f(j, t_r^j) \le \left(2 + \frac{1}{k}\right) t_r^j$. □

Lemma 9: For any task T_r^j, $f(j, t_r^j) \le S_{OPT}$, where $1 \le j \le \delta$ and $1 \le r \le n_j$.

Proof: According to the scheduled-parallelism procedure, j is the degree of parallelism that makes task T_r^j have the shortest total processing time $f(j, t_r^j)$. Thus, $f(j, t_r^j) \le S_{OPT}$. □

Theorem 1: The performance bound of MLSPF is $\left(4 + \frac{2}{k} - \frac{1}{P}\right)$, where $P = \sum_{a=1}^{M} p_a$.

Proof: Since T_z^w is the task finished at time S_{MLSPF}, then

$$S_{MLPSF} \le \left[\sum_{j=1}^{\delta} \sum_{r=1}^{n_j} j \times f(j, t_r^j) + \sum_{j=w+1}^{\delta} \sum_{a=1}^{M} A_a^j + \sum_{a=1}^{M} (w-1) \times (\psi - \min(\psi, ST_a^w) \right.$$

$$+ (P-w) \times f(w, t_z^w) \Bigg] \Bigg/ P.$$

$$S_{MLPSF} \le \Bigg[\sum_{j=1}^{\delta} \sum_{r=1}^{n_j} j \times f(j, t_r^j) + \sum_{j=w+1}^{\delta} \sum_{a=1}^{M} A_a^j + \sum_{j=1}^{w} \sum_{a=1}^{M} (j-1) \times (FT_a^j - ST_a^j) + (P-1) \times f(w, t_z^w) \Bigg] \Bigg/ P.$$

By Lemma 4 and Lemma 5,

$$S_{MLSPF} \le \Bigg[\sum_{j=1}^{\delta} \sum_{r=1}^{n_j} j \times f(j, t_r^j) + \sum_{j=1}^{\delta} \sum_{r=1}^{n_j} (j-1) \times f(j, t_r^j) + (P-1) \times f(w, t_z^w) \Bigg] \Bigg/ P.$$

By Lemma 7 and Lemma 8,

$$S_{MLSPF} \le \Bigg[\sum_{j=1}^{\delta} \sum_{r=1}^{n_j} \left(2 + \frac{1}{k}\right) t_r^j + \sum_{j=1}^{\delta} \sum_{r=1}^{n_j} \left(1 + \frac{1}{k}\right) t_r^j + (P-1) \times f(w, t_z^w) \Bigg] \Bigg/ P$$

Because $\sum_{j=1}^{\delta} \sum_{r=1}^{n_j} t_r^j / P \le S_{OPT}$ and according to Lemma 9, $S_{MLSPF} \le \left(4 + \frac{2}{k} - \frac{1}{P}\right) S_{OPT}$. □

In the following, we will discuss the performance bound of MLSPF if $\left\lfloor \frac{\theta}{\delta} \right\rfloor = \gamma$, where $\theta = \min\{p_a \mid a = 1, 2, \ldots, M\}$, and we denote the number of occupied processors and the number of idle processors in the machine m_a at time τ by $O_{\tau|a}$ and $I_{\tau|a}$, respectively.

Lemma 10: If $ST_a^u \le \tau \le ST_a^v$ and $ST_a^v \le \psi$, then $\gamma \cdot I_{\tau|a} \le O_{\tau|a}$, where $\left\lfloor \frac{\theta}{\delta} \right\rfloor = \gamma$, $\theta = \min\{p_a \mid a = 1, 2, \ldots, M\}$ and $1 < u \le \delta$.

Proof: $\left\lfloor \frac{\theta}{\delta} \right\rfloor = \gamma$ implies that the maximum number of tasks all with the degree of scheduled parallelism δ can be processed in a machine with θ processors. If T^v is the task set next to be scheduled after T^u, according to Lemma 1, $ST_a^u \le ST_a^v$.

(1) If $ST_a^u < ST_a^v$, according to the definition of ST_a^v, none of the tasks with the degree of scheduled parallelism v can be started before the time ST_a^v in the machine m_a. This implies that the degrees of scheduled parallelism of tasks processed in the machine m_a between ST_a^u and ST_a^v are all at least u. Since $\theta \le p_a$ and $1 \le u \le \delta$, $\gamma \le \frac{p_a}{u}$. Then, there must be at least γ tasks with the degree of scheduled parallelism u assigned in the machine m_a with p_a processors at time τ, that is, $\gamma \cdot u \le O_{\tau|a}$, where $1 \le a \le M$ and $ST_a^u \le \tau < ST_a^v$. By Lemma 2, $I_{\tau|a} \le (u-1)$. Thus, $\gamma \cdot I_{\tau|a} \le O_{\tau|a}$.

(2) If $ST_a^u = ST_a^v$, then the same discussion with the task sets T^v and $T^{v'}$, where $T^{v'}$ is the task next to be scheduled after T^v. □

Lemma 11: $\gamma \cdot I_{\tau|a} \le O_{\tau|a}$, where $1 \le a \le M$ and $\min\{ST_a^w, \psi\} \le \tau < \psi$.

Proof: By Lemma 6 and Lemma 10. □

Lemma 12: $\gamma \cdot I_{\tau|a} \le O_{\tau|a}$, where $0 \le \tau < \psi$.

Proof: According to Lemma 10 and Lemma 11. □

Lemma 13: $\sum_{\tau=0}^{\psi} \sum_{a=1}^{M} I_{\tau|a} \le \frac{1}{\gamma} \sum_{j=1}^{\delta} \sum_{r=1}^{n_j} j \times f(j, t_r^j)$.

Proof: By Lemma 12, $\sum_{\tau=0}^{\psi} \sum_{a=1}^{M} I_{\tau|a} \le \frac{1}{\gamma} \sum_{\tau=0}^{\psi} \sum_{a=1}^{M} O_{\tau|a}$. Since occupied processors means that those processors are allocated to tasks, $\sum_{\tau=0}^{\psi} \sum_{a=1}^{M} O_{\tau|a} \le \sum_{j=1}^{\delta} \sum_{r=1}^{n_j} j \times f(j, t_r^j)$. Hence, $\sum_{\tau=0}^{\psi} \sum_{a=1}^{M} I_{\tau|a} \le \frac{1}{\gamma} \sum_{j=1}^{\delta} \sum_{r=1}^{n_j} j \times f(j, t_r^j)$. □

Corollary 1: The performance bound of MLSPF is $\left(3 + \frac{1}{k} + \frac{(2+\frac{1}{k})}{\gamma} - \frac{1}{P}\right)$, where $\left\lfloor \frac{\theta}{\delta} \right\rfloor = \gamma$ and $\theta = \min\{p_a \mid a = 1, 2, \ldots, M\}$.

Proof: $S_{MLPSF} \le \Bigg[\sum_{j=1}^{\delta} \sum_{r=1}^{n_j} j \times f(j, t_r^j) + \sum_{\tau=0}^{\psi} \sum_{a=1}^{M} I_{\tau|a} + (P-w) \times f(w, t_z^w) \Bigg] \Bigg/ P.$

$$S_{MLSPF} \le \Bigg[\sum_{j=1}^{\delta} \sum_{r=1}^{n_j} j \times f(j, t_r^j) + \frac{1}{\gamma} \sum_{j=1}^{\delta} \sum_{r=1}^{n_j} j \times f(j, t_r^j) + (P-1) \times f(w, t_z^w) \Bigg] \Bigg/ P.$$

$$S_{MLSPF} \le \left(3 + \frac{1}{k} + \frac{\left(2 + \frac{1}{k}\right)}{\gamma} - \frac{1}{P}\right) S_{OPT}.$$ □

Corollary 2: The performance bound of MLSPF approaches $\left(3 + \frac{1}{k}\right)$ if γ is sufficiently large.

Proof: The term $\frac{(2+\frac{1}{k})}{\gamma}$ approaches 0 if γ is sufficiently large. Since $\gamma = \left\lfloor \frac{\theta}{\delta} \right\rfloor$, it also means that $\frac{1}{P}$ approaches 0. Thus, it is clear that $S_{MLSPF} \le \left(3 + \frac{1}{k}\right) S_{OPT}$. □

Example 1: Assume there are $(2p-1)$ tasks to be scheduled in a 2-machine environment, in which each task T_i, $i = 1, 2, \ldots, (2p-1)$, is associated with a computation requirement t_i and a maximum degree of parallelism Δ_i, and each machine m_a, $a = 1, 2$, contains p processors. Let $t_i = t$ and $\Delta_i = \frac{p}{2} + 1$ for $i = 1, 2, \ldots, 2p-6$; $t_j = t$ and $\Delta_j = p/2$ for $j = 2p-5, 2p-4, \ldots, 2p-2$; $t_{2p-1} = t$ and $\Delta_{2p-1} = 1$; and $c = \frac{t}{2k}\left(\frac{p}{2} + 1\right)^{-(k+1)}$. Then, $f\left(\frac{p}{2} + 1, t_i\right) = \frac{1}{p+2}\left(2 + \frac{1}{k}\right)t$, $f\left(\frac{p}{2}, t_j\right) \approx \frac{2t}{p}$ and $f(1, t_{2p-1}) = t$, where $i = 1, 2, \ldots, 2p-6$ and $j = 2p-5, 2p-4, \ldots, 2p-2$. Figures 1 (a) and (b) show the schedule lengths of an MLSPF schedule and that of an optimal schedule are $\left(3 + \frac{1}{k} + \frac{2}{p} - \frac{5}{p+2}\left(2 + \frac{1}{k}\right)\right)t$, approximately, and t, respectively. If p is an extremely large integer, the performance bound of this example approaches $\left(3 + \frac{1}{k}\right)$.

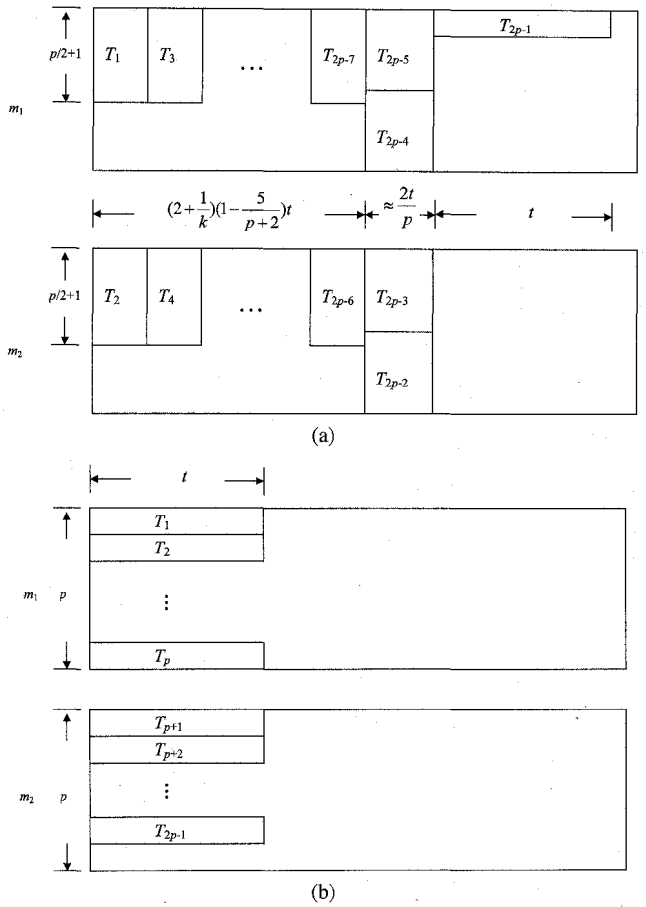

Fig. 1 (a) An MLSPF schedule and (b) an optimal schedule of example 1.

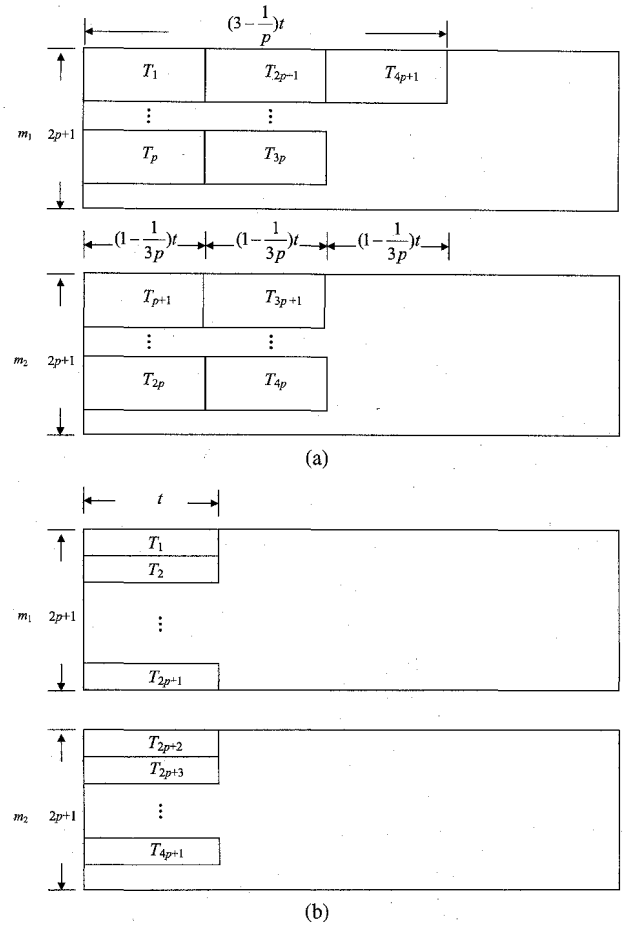

Fig. 2 (a) An MLSPF schedule and (b) an optimal schedule of example 2.

This example looks like that the derived performance bound of Corollary 2 is tight. However, $\gamma = \left\lfloor \frac{2p-1}{p+1} \right\rfloor = 1$ that does not satisfy the condition that γ is an extremely large integer. □

In the following, we will give another example that satisfies the condition that γ is an extremely large integer.

Example 2: Assume that p is an extremely large integer and that there are $(4p+1)$ tasks to be scheduled in a 2-machine environment, in which each task T_i, $i = 1, 2, \ldots, (4p+1)$, is associated with a computation requirement t_i and a maximum degree of parallelism Δ_i, and each machine m_a, $a = 1, 2$, contains $(2p+1)$ processors. Let $t_i = t$ and $\Delta_i = 2$, $i = 1, 2, \ldots, 4p+1$, and $c = \left(\frac{1}{2} - \frac{1}{3p}\right) t \cdot 2^{-k}$. Then, we calculate that the degree of scheduled parallelism of task T_i is 2 and $f(2, t_i) = \left(1 - \frac{1}{3p}\right) t$, where $i = 1, 2, \ldots, 4p+1$. Figures 2 (a) and (b) show the schedule lengths of an MLSPF schedule and that of an optimal schedule are $\left(3 - \frac{1}{p}\right) t$ and t, respectively. Since p is extremely large, the performance bound of this example is approximately 3. This example has satisfied the condition that $\gamma = \left\lfloor \frac{4p+2}{2} \right\rfloor$ is extremely large, but it shows that the derived performance bound of Corollary 2 is not tight. □

4. Conclusion

In this paper, the problem of nonpreemptively scheduling independent parallel tasks with the consideration of communication overhead in a multi-machine environment is discussed. For such a problem, we proposed the MLSPF algorithm and derived its performance bound as $\left(4 + \frac{2}{k} - \frac{1}{P}\right)$. The performance bound of the MLSPF can also be presented as $\left(3 + \frac{1}{k} + \frac{(2+\frac{1}{k})}{\gamma} - \frac{1}{P}\right)$, where $\theta = \min\{p_a \mid a = 1, 2, \ldots, M\}$ and $\left\lfloor \frac{\theta}{\delta} \right\rfloor = \gamma$. The performance bound of the MLSPF approaches $\left(3 + \frac{1}{k}\right)$ if γ is sufficiently large, but the derived performance bound of the MLSPF in Corollary 2 is still not tight.

Acknowledgements

The author would like to thank the anonymous referees for their helpful comments and their carefully correcting the mistakes in the paper.

References

[1] J. Blazewicz, M. Drabowski, and J. Weglarz, "Scheduling multiprocessor tasks to minimize schedule length," IEEE Trans. Comput., vol.35, no.5, pp.389–393, 1986.

[2] T.D. Braun, H.J. Siegel, N. Beck, L.L. Boloni, M. Maheswaran, A.I. Reuther, J.P. Robertso, M.D. Theys, B. Yao, D. Hensgen, and R.F. Freund, "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," J. Parallel Distrib. Comput., vol.61, pp.810–837, 2001.

[3] J. Du and J.Y. Leung, "Complexity of scheduling parallel task system," SIAM J. Discrete Math., vol.2, pp.473–487, 1989.

[4] H. El-Rewini and M. Abd-El-Barr, "Scheduling and task allocation," in Advanced Computer Architecture and Parallel Processing, pp.235–265, John Wiley & Sons, 2005.

[5] I. Foster and C. Kesselman, The Grid 2: Blueprint for a New Computing Infrastructure, Elsevier, Amsterdam, 2004.

[6] R.L. Graham, "Bounds on multiprocessing timing anomalies," SIAM J. Appl. Math., vol.17, no.2, pp.416–429, 1969.

[7] J.F. Lin and S.J. Chen, "Performance bounds on scheduling parallel tasks with setup time on hypercube systems," Informatica, vol.19, pp.313–318, 1995.

[8] J.F. Lin, W.B. See, and S.J. Chen, "Performance bounds on scheduling parallel tasks with communication cost," IEICE Trans. Inf. & Syst., vol.E78-D, no.3, pp.263–268, March 1995.

[9] V. Di Martino and M. Mililotti, "Sub optimal scheduling in a grid using genetic algorithms," Parallel Comput., vol.30, pp.553–565, 2004.

[10] F. Pascual, K. Rzadca, and D. Trystram, "Cooperation in multi-organization scheduling," Euro-Par 2007, pp.224–233, Rennes, France, Aug. 2007.

[11] G.S. Sajjan, "Array processors," in Advanced Computer Architectures, pp.167–220, Taylor & Franics Group, 2006.

[12] J.P. Singh, J.L. Hennessy, and A. Gupta, "Scaling parallel programs for multiprocessors: Methodology and examples, computer," Computer, pp.42–50, July 1993.

[13] B. Tierney, W. Johnston, J. Lee, and M. Thompson, "A data intensive distributed computing architecture for 'Grid' applications," Future Gener. Comput. Syst., vol.16, pp.473–481, 2000.

[14] Q. Wang and K.H. Cheng, "List scheduling of parallel tasks," Inf. Process. Lett., vol.37, no.5, pp.291–297, 1991.

[15] Q. Wang and K.H. Cheng, "A heuristic of scheduling parallel tasks and its analysis," SIAM J. Comput., vol.21, no.2, pp.281–294, 1992.

[16] C. Weng and X. Lu, "Heuristic scheduling for bag-of-tasks applications in combination with QoS in the computational grid," Future Gener. Comput. Syst., vol.21, pp.271–280, 2005.

Jiann-Fu Lin received the B.S. and M.S. degrees in Applied Mathematics from National Chung-Hsing University, Taipei, Taiwan in 1988 and 1990, respectively, and the Ph.D. degree in Electrical Engineering from National Taiwan University, Taipei, Taiwan in 1994. He is currently an associate professor in the department of Management Information System, Takming University of Science and Technology, Taipei, Taiwan. Dr. Lin's current research interests include tasks scheduling problems and data security.