採用最長處理時間優先原則於多機器環境下對多處理機式工作排程的最差效能值分析

林建福*
德明財經科技大學 資訊管理系
Email: alfu@takming.edu.tw

廖鴻圖
世新大學 資訊管理系
Email: htliaw@cc.shu.edu.tw

*通訊作者：林建福

摘要

本文主要在探究一個靜態排程問題，文中針對如何將一群獨立的、不可搶用的多處理機式工作分派於具有多部機器的環境中處理。此環境中的每部機器都擁有數量不盡相同的處理機，每個多處理機式工作必需在某一部機器上同時擁有預先設定數量的處理器才能執行。對此問題去找尋一個具有最小長度的排程是個 NP-完全的問題。此類問題一般可採用啟發式排程演算法；並藉由啟發式排程演算法的最差效能值來評估其優劣。基於最長處理時間優先的處理原則，本論文提出"植基於最長處理時間優先" 的排程演算法，經分析推導出其最差效能值為 5/2。

關鍵詞：靜態排程問題、多處理機式工作、多機器環境、最長處理時間優先原則、最差效能值

Worst Performance Analysis of Scheduling Multiprocessor Tasks in a Multi-Machine Environment Using LPT Policy

Jiann-Fu Lin*
Department of Management Information System,
Takming University of Science and Technology,
Email: alfu@takming.edu.tw

Horng-Twu Liaw
Department of Information Management,
Shih Hsin University
Email: htliaw@cc.shu.edu.tw

*Corresponding author: Jiann-Fu Lin

Abstract

The paper investigates a static scheduling problem, in which a set of independent, non-preemptable multiprocessor tasks are to be assigned to an environment with multiple machines. Each machine in the environment contains a number of identical processors and each multiprocessor task requires to be processed on a single machine by a given number of processors. The problem of finding a schedule with minimum scheduling length (makespan) for such a scheduling problem is NP-complete. A heuristic scheduling algorithm is usually used to obtain a feasible schedule and the efficiency of a heuristic scheduling algorithm may resort to evaluating its worst performance. Bases on the largest processing time first (LPT) policy, the paper proposes the LPT-based scheduling algorithm for such a scheduling problem. The worst performance of the LPT-based scheduling algorithm is derived as 5/2.

Keywords: static scheduling problem; multiprocessor task; multi-machine environment; largest processing time first policy; worst performance.

Worst Performance Analysis of Scheduling Multiprocessor Tasks in a Multi-Machine Environment Using LPT Policy

I. INTRODUCTION

Recently, with the continuous improvements in network performance and computers, the construction of a distributed computing over the Internet has become feasible. Along with the rapid development in distributed computing environment, a wide variety of interesting problems are brought on. Task scheduling problem is one of important issues in such an environment. The problem of scheduling independent, non-preemptable multiprocessor tasks in an environment with a single machine for finding a schedule with minimum scheduling length has been known as an NP-complete problem (Blazewicz, Drabowski, and Weglarz 1986), the problem of scheduling independent, non-preemptable multiprocessor tasks in a multi-machine environment is therefore an NP-complete problem. Now that the problem of non-preemptively scheduling non-preemptable multiprocessor tasks in a multi-machine environment is NP-complete, many heuristic scheduling algorithms are proposed for such a problem to obtain near optimal schedules. Performance is a way to evaluate the efficiency of a heuristic scheduling algorithm, which can be evaluated either by experimental results or by mathematical analysis.

A genetic algorithm is a technique that has been widely used in many fields for solving NP-complete problems. Martino and Mililotti (Martino, and Mililotti 2004) developed a simulation grid computing environment to evaluate the usefulness of genetic algorithms for scheduling independent multiprocessor tasks in an environment with several machines. In contrast with their previous work on up to 24 tasks (Martino, and Mililotti 2002), they found that their genetic algorithm for scheduling 32 tasks does not converge to an optimal schedule within a given number of trials performed; only a sub-optimal schedule can be obtained. Pascual, Rzadca and Trystram (Pascual, Rzadca, and Trystram 2007) proposed the Multi-Organization Load Balancing Algorithm (MOLBA) for the problem of scheduling independent, non-preemptable multiprocessor tasks in a grid computing environment, and showed that the worst performance the MOLBA to be 4. Later, Rzadca (Rzadca 2008) took on the same problem and showed the worst performance of the List scheduling algorithm as 3. Almost at the time, Schwiegelshohn, Tchernykh and Yahyapour (Schwiegelshohn, Tchernykh and Yahyapour 2008) also proposed the Grid Concurrent-Submission (GCS) algorithm for the same problem and showed the worst performance of the GCS algorithm as 3. Lin (2010a) discussed the problem of non-preemptively scheduling independent multiprocessor tasks in a distributed computed environment and formulated the performance of the List scheduling algorithm as $(2+\frac{1}{\lambda}-\frac{1}{P})$, where $1 \le \lambda$ and P is the total number of processors in the

scheduling environment. Yet, the worst performance of List scheduling algorithm is still 3. In the same year, Lin (2010b) also proposed the Higher Parallelism first scheduling algorithm for the same problem and the performance is still bounded by 3.

Since the problem of finding an optimal schedule for scheduling independent, non-preemptable multiprocessor tasks in a multi-machine environment is an NP-complete problem, a scheduling algorithm based on the Largest Processing Time First (LPT) policy is proposed for such a problem. The rest of this paper is organized as follows. Section 2 describes the multiprocessor task scheduling problem. In Section 3, an LPT-based scheduling algorithm is proposed for such a problem and its worst performance is derived. Conclusions are given in Section 4.

II. DESCRIPTION OF THE SCHEDULING PROBLEM

Considering n independent, non-preemptable multiprocessor tasks, T_1, T_2, …, T_n, are to be scheduled in a c-machine environment. In the environment, each machine m_i consists of p_i identical processors, each multiprocessor task T_k must be processed on a single machine by δ_k processors simultaneously for t_k units of processing time without preemption, where i=1, 2, …, c, $1 \le k \le n$ and δ_k is the parallelism of task T_k. For such a problem, a schedule is feasible if a multiprocessor task T_k can be precisely processed on a single machine by δ_k processors at a time. It is therefore assumed that $\delta \le \rho$, where $\delta = \max\{\delta_k \mid k = 1,2,...,n\}$ and $\rho = \min\{p_i \mid i = 1,2,...,c\}$. The paper would like to consider the worst performance of the LPT-based scheduling algorithm for scheduling multiprocessor tasks under the constraint that $\delta \le \left\lceil \frac{\rho}{2} \right\rceil$ in a multi-machine environment.

III. WORST PERFORMANCE ANALYSIS

Based on the Largest Processing Time First (LPT) policy (Graham 1969), an LPT-based scheduling algorithm is proposed for such a problem. Since the LPT policy is adopted, the LPT-based scheduling algorithm first sorts the multiprocessor tasks in non-increasing order by their processing times. That is, the n multiprocessor task T_1, T_2, …, T_n are organized as a sorted list $\{T_1^S, T_2^S, ..., T_n^S\}$, where $t_k^S \ge t_{k+1}^S$, t_k^S is the processing time of T_k^S and k=1, 2, …, n-1. According to the order of multiprocessor tasks in the sorted list, the LPT-based scheduling algorithm sequentially assigns tasks to available machines. The LPT-based scheduling algorithm is described below.

Algorithm LPT-based {

Input the processing time t_k and parallelism δ_k of multiprocessor task T_k, where k=1, 2, …, n;

Arrange Tasks in non-increasing order by their processing times to form a sorted list

$\{T_1^S, T_2^S, ..., T_n^S\}$, where $t_k^S \geq t_{k+1}^S$, and k=1, 2, …, n-1;

While (the sorted list is not empty) **do {**

Choose task T_k^S at the head of the sorted list;

If (there is a machine m_i with at least δ_k^S free processors) **then {**

Machine m_i allocates δ_k^S processors to T_k^S for execution, where δ_k^S is the parallelism of task T_k^S, k=1, 2, …, n-1 and $1 \leq i \leq c$;

Remove task T_k^S from the sorted list; **}**

}

}

Assuming that S_{LPT} and S_{OPT} are the finish time of an LPT-based schedule and that of an optimal schedule respectively, and tasks are scheduled from time 0. Thus, it is clear that S_{LPT} is the length of the LPT schedule. Let T_x^S be the task with the largest integer x finished at time S_{LPT}. Then, $(S_{LPT} - t_x^S)$ is the starting time of task T_x^S, where t_x^S is the processing time of task T_x^S. Some processors might be idle during the LPT-based schedule. Therefore, the total time that processors are idle in each machine during the LPT-based schedule needs to be calculated to derive the performance of the LPT-based algorithm.

Lemma 3.1. *If* T_x^S *is task* T_1^S*, then* $S_{LPT} = S_{OPT}$.

Proof: Since $T_1^S = T_x^S$ and T_1^S starts at time 0, the length of the LPT schedule is t_1^S. That is $S_{LPT} = t_1^S$. Now that the tasks are non-preemptable, $S_{OPT} = t_1^S$, $S_{LPT} = S_{OPT}$. ■

Lemma 3.2. *If* $S_{OPT} > t_1^S$*, then an optimal schedule must include at least two tasks.*

Proof: Due to the processing time of any task is not greater than t_1^S, it is impossible that the optimal schedule length containing only one task is greater than t_1^S. ■

Lemma 3.3. *If* $S_{OPT} > t_1^S$*, then* $t_x^S \leq \frac{S_{OPT}}{2}$.

Proof: Since tasks in sorted list $\{T_1^S, T_2^S, ..., T_n^S\}$ are arranged in non-increasing order of

their processing times, $t_1^S \geq t_2^S \geq ... \geq t_x^S \geq ... \geq t_n^S$. Based on Lemma 3.2 and that T_x^S is the xth task in the sorted list finished at time S_{LPT}, there exists an integer $y \leq x$ that the y tasks can obtain an optimal schedule according to the LPT-based scheduling algorithm. Due to $S_{OPT} \geq t_1^S$, and tasks T_{y-1}^S and T_y^S are the two tasks with smallest processing times of the y tasks, it follows that

$$S_{OPT} \geq t_{y-1}^S + t_y^S . \tag{1}$$

For the reason that tasks are sorted in non-increasing order and $y \leq x$, $t_{y-1}^S \geq t_y^S \geq t_x^S$. According to Inequality (1), $S_{OPT} \geq t_x^S + t_x^S$. Hence, $\frac{S_{OPT}}{2} \geq t_x^S$. ■

Lemma 3.4. *The number of idle processors in machine m_i at any time between 0 and $(S_{LPT} - t_x^S)$ is at most $(\delta - 1)$, where $\delta = \max\{\delta_k \mid k = 1,2,...,n\}$.*

Proof: If the number of idle processors in machine m_i at time τ, $0 \leq \tau \leq (S_{LPT} - t_x^S)$, were greater than $(\delta - 1)$, a task T_k would be assigned at that time. That leads to a contradiction. ■

Lemma 3.5. *The total idle time of all machine* between the times 0 and $(S_{LPT} - t_x^S)$ *is at most* $\frac{(S_{LPT} - t_x^S)}{2} \sum_{i=1}^{c} p_i$.

Proof: According to Lemma 3.4, at any moment between the times 0 and $(S_{LPT} - t_x^S)$ the maximum number of idle processors in each machine is $(\delta - 1)$. Now that $\delta = \max\{\delta_k \mid k = 1,2,...,n\}$, $\rho = \min\{p_i \mid i = 1,2,...,c\}$ and $\delta \leq \left\lceil \frac{\rho}{2} \right\rceil$,

$$\sum_{i=1}^{c} \sum_{\tau=0}^{(S_{LPT} - t_x^S)} (\delta - 1) \leq \sum_{\tau=0}^{(S_{LPT} - t_x^S)} \sum_{i=1}^{c} (\left\lceil \frac{\rho}{2} \right\rceil - 1) . \tag{2}$$

According to Inequality (2) and the result that $\left\lceil \frac{\rho}{2} \right\rceil \leq \frac{\rho + 1}{2}$, it follows that $\sum_{i=1}^{c} \sum_{\tau=0}^{(S_{LPT} - t_x^S)} (\delta - 1) \leq \sum_{\tau=0}^{(S_{LPT} - t_x^S)} \sum_{i=1}^{c} \frac{p_i}{2}$. Then,
$\sum_{i=1}^{c} \sum_{\tau=0}^{(S_{LPT} - t_x^S)} (\delta - 1) \leq \frac{(S_{LPT} - t_x^S)}{2} \sum_{i=1}^{c} p_i$. ■

Theorem 3.1. *The performance of the LPT-based algorithm for scheduling multiprocessor*

tasks with parallelisms not greater than δ *in a multi-machine environment is* $\frac{5}{2}$.

Proof: Since S_{LPT} is the finish time of an LPT schedule, and task T_x^S finishes at time S_{LPT}, there follows

$$S_{LPT} \le \{\sum_{\tau=0}^{x-1}(\delta_k^S \times t_k^S) + (\sum_{i=1}^{c} p_i) \times t_x^s + \sum_{i=1}^{c} \sum_{\tau=0}^{(S_{LPT}-t_x^S)} (\delta - 1)\} / \sum_{i=1}^{c} p_i .$$

According to Lemmas 3.5,

$$S_{LPT} \le \{\sum_{k=1}^{x-1}(\delta_k^S \times t_k^S) + (\sum_{i=1}^{c} p_i) \times t_x^s + \frac{(S_{LPT} - t_x^S)}{2} \sum_{i=1}^{c} p_i\} / \sum_{i=1}^{c} p_i$$

$$\Rightarrow S_{LPT} \le \{\sum_{k=1}^{n}(\delta_k^S \times t_k^S) + (\sum_{i=1}^{c} p_i) \times t_x^s + \frac{(S_{LPT} - t_x^S)}{2} \sum_{i=1}^{c} p_i\} / \sum_{i=1}^{c} p_i$$

Since $t_x^S \le \frac{S_{OPT}}{2}$, according to Lemma 3.3, and $\{\sum_{k=1}^{n}(\delta_k^S \times t_k^S) / \sum_{i=1}^{c} p_i\} \le S_{OPT}$,

$S_{LPT} \le S_{OPT} + \frac{1}{2} S_{OPT} + \frac{S_{LPT} - \frac{1}{2} S_{OPT}}{2}$. Hence,

$S_{LPT} \le \frac{5}{2} S_{OPT}$ ■

IV. CONCLUSIONS

The problem of non-preemptively scheduling independent multiprocessor tasks with a parallelism constraint in a multi-machine environment is an NP-complete problem. Based on the LPT policy, the paper proposed the LPT-based scheduling algorithm for such a problem and derived its worst performance as $\frac{5}{2}$. The derived worst performance of the LPT-based scheduling algorithm is much better than those of the others scheduling algorithm for the problem.

ACKNOWLEDGMENT

The work is partially supported by the National Science Council under project number NSC 100-2410-H-147-004. The authors would like to express the appreciation for the support.

REFERENCES

1. Blazewicz, J., Drabowski, M. and Weglarz, J. "Scheduling multiprocessor tasks to minimize schedule length," *IEEE Trans. Comput.* (35:5) 1986, pp:389-393.
2. Graham, R.L. "Bounds on multiprocessing timing anomalies," *SIAM J. Appl. Math.* (17:2) 1969, pp:416-429.

3. Lin, J. F. “List scheduling multiprocessor tasks in grid computing environments,” *ICIC Express Letters* (4:1) 2010a, pp:245-248.
4. Lin, J. F. “Performance analysis and discussion on a heuristic for scheduling multiprocessor tasks in a grid computing environment,” *International Journal of Innovative Computing, Information and Control* (6:12) 2010b, pp:5451-5462.
5. Martino, V. D. and Mililotti, M. “Scheduling in a grid computing environment using genetic algorithms.” *Proceedings of the International Parallel and Distributed Processing Symposium*: IPDPS 2002 Workshops, 2002, pp:235-239,.
6. Martino, V. D. and Mililotti, M. “Sub optimal scheduling in a grid using genetic algorithms.” *Parallel Computing* (30) 2004, pp:553-565.
7. Pascual, F., Rzadca, K. and Trystram, D. “Cooperation in multi-organization scheduling,” Euro-Par 2007, pp:224-233, 2007.
8. Rzadca, K., “Scheduling in multi-organization grids: measuring the inefficiency of decentralization,” *LNCS* 4967, Springer, 2008, pp: 1048–1058.
9. Schwiegelshohn, U., Tchernykh, A. and Yahyapour, R. “Online scheduling in grids,” *Proceedings of the International Parallel and Distributed Processing Symposium*: IPDPS 2008 Workshops, 2008, pp:1-10.