ATLANTIS PRESS

Advances in Engineering Research (AER), volume 148

3rd Workshop on Advanced Research and Technology in Industry Applications (WARTIA 2017)

A novel HEFT$_{S-L}$ algorithm for scheduling large number of DAG tasks

Xin Liu[1,2,a], Rongbin Xu[1,2,3,b,*], Yongliang Cheng[2,c] and Pengfei Zhang[2,d]

[1] Key Laboratory of Intelligent Computing & Signal Processing, Ministry of Education, Anhui University,

[2] School of Computer Science and Technology, Anhui University, Anhui Province, China

[3] Co-Innovation Center for Information Supply & Assurance Technology, Anhui University, Hefei, China

[a] 1025722404@qq.com, [b] xurb_910@ahu.edu.cn, [c] 1399603339@qq.com, [d] 1982471053@qq.com

Keywords: Scheduling; Business process management; DAG; HEFT algorithm; HEFT$_{S-L}$ algorithm

Abstract: In recent years, along with the development of technologies for distributed computing such as big data and clouds workflow systems, efficiency of workflow scheduling has become very impotent. Hence scheduling of multiple DAGs sharing on heterogeneous distributed resources has attracted intensive attention recently. This paper issues on scheduling of multiple DAGs with Deadline constraints surrounding scheduling of multiple DAGs sharing on heterogeneous distributed resources on traditional DAG-based task. It brings forward two parameter association methods to balance DAG Deadline constraint priority including relative strictness and laxity, which are used to balance priority of multiple DAGs. An improved HEFT algorithm based on priority was put forward to schedule multiple DAGs with deadline constraint. Through experiments, the phenomenon of "overload" can be detected due to the high degree of emergency of DAG scheduling. Once a task was detected, it can be selective discarded with corresponding measures, so as to maximize DAG before deadline.

1. Introduction

Scheduling is a fundamental problem in computer science and has a wide range of applications in many fields [1]. In recent years, with the development of big data and cloud computing, a large number of business processes have been generated, and the reasonable scheduling of business processes has become a focus of scholars at home and abroad [2-3]. Business process management can generally be abstracted as data path, structured and unstructured data storage and access, and there is generally a mutual dependence relationship between each process [4]. In the process, for example, between a data transmission, parallel relationship constraints and timing constraints, etc., these relationships can be through the use of Directed Acyclic Graph modelling, ensure all tasks are completed in a limited amount of time scheduling, improve the efficiency and reliability of the scheduling system. Meanwhile, the DAG model can be used for real-time scheduling system because the business process usually pursues timeliness.

The scheduling of a large number of DAG tasks is a classic NP problem [2, 5], and the tasks have their own constraints. Task scheduling can be divided into uniprocessor scheduling and multiprocessor scheduling. At present, the problem of multiprocessor scheduling for a DAG task is approaching, and the research on different target resources, scheduling objectives and scheduling methods is approaching maturity [6-7]. With the rapid development of grid computing and cloud computing applications, in a heterogeneous distributed computing environment on how to improve the multiple task scheduling performance aspects put forward the new requirements, also caused the wide attention of scholars both at home and abroad [8-9]. For multiprocessor task scheduling, how to improve the throughput of multi-task scheduling system is a research hotspot [10]. At present, the researchers have proposed many about methods of improving the performance of DAG task scheduling, but most of these articles focus on how to improve the efficiency of without time constraints of scheduling, in order to complete all the tasks as fast as possible. And it does not take into account a large number of real-time scheduling parallel tasks, more does not take into account

temporal constraints the veracity and the rationality of the task scheduling and resource utilization, etc. To meet all these constraints for scheduling a lot with the task of sequence limit is a big challenge [11], because not only need to pay attention to the correctness of the scheduling results, also need to consider all the execution of task execution time, deadline and concurrent execution and other factors.

For a large number of DAG task sharing processing resources, this paper is based on the $HEFT_{S-L}$ algorithm based on relative rigor and relative laxity. This algorithm will DAG task completion and the fairness of task scheduling algorithm performance index, according to different user task scheduling in the process of submitting the different deadline, dynamically determine the priority of the task, and greatly improve the throughput of task scheduling. At the same time, the task overload occurred in the scheduling process, and the reasonable task ejection mechanism is used to improve the completion rate of all tasks and the fairness of scheduling. By comparing with EDF algorithm and improving LLF algorithm, the effectiveness of $HEFT_{S-L}$ algorithm is further verified, and new ideas are proposed on how to improve the throughput of large number of business process task scheduling.

2. Task-Scheduling Algorithm

2.1. Basic Definition

We consider a task set of n real-time Directed Acyclic Graph (DAG) tasks run on a system of q identical processors$(P_1, P_2, P_3, \ldots, P_q)$. The task set is represented by $\{G_1, G_2, ..., G_3, \ldots, G_n\}$. Each DAG task G_i, where $1 \leq i \leq n$, is a deadline graph which consists of a set of subtasks under precedence constraints that determine their execution flow.

Multiprocessor Task Scheduling problem for dependent tasks can be represented by the Directed Acyclic Graph (DAG) G = (V, E), where V is the set of v nodes and E is the set of e edges between the tasks (Task and node terms are interchangeably used in the paper). Each edge $(i, j) \in E$ represents the precedence constraint such that task t_i should complete its execution before task t_j starts. Data is a matrix of communication data, where data i, k is the amount of data required to be transmitted from task t_i to task t_k. The source task is called the parent task while the sink task is called the child task. In a DAG, a task with no any parent is called the start task while the task with no any child is called the end task.

In view of above problems, we assume that the target computing environment compose of a set Q of q heterogeneous processors connected in a fully connected topology in which all inter processor communications are assumed to perform without contention. In our model, we consider that computation can be overlapped with communication. A DAG task t_i is characterized by $(\{t_{i,j} | 1 \leq j \leq n_j\}, W, w_{i,j}, c_{i,k})$,where the first parameter represents the set of subtasks of t_i and n_i is their number, W is characterized by the DAG task execution cost and $w_{i,j}$ is the execution time to complete task n_i on processor P_q. The average execution cost of a task is as formula 1

$$\overline{w_{i,j}} = \sum w_{i,j}/q \tag{1}$$

Where $\overline{B}$ is average transfer rate among the processors in the domain, $\overline{L}$ is the average communication startup time. For the other tasks in the graph, the EFT and EST values are computed recursively, starting from the entry task, as formula 2 and 3.

$$EST(t_i, p_j) = \max\{avail[j], \max_{t_m \in pred(t_i)}(AFT(t_m) + c_{m,i})\} \tag{2}$$

$$EFT(t_i, p_j) = w_{i,j} + EST(t_i, p_j) \tag{3}$$

For the other tasks in the graph, the EFT and EST values are computed recursively, starting from the entry task, as formula 4 and 5.

$$EST(t_i, p_j) = \max\{avail[j], \max_{t_m \in pred(t_i)}(AFT(t_m) + c_{m,i})\} \tag{4}$$

$$EFT(t_i, p_j) = w_{i,j} + EST(t_i, p_j) \tag{5}$$

It is necessary to get the EFT of a task t_i, all immediate predecessor task of t_i must have been scheduled.

Where pred(t_i) is the task t_i parent node and avail[j] is the earliest time at which processor P_j is start for task execution. If t_i is the last assigned task on processor P_j, then avail[j] is the time that processor P_j completed the execution of the task t_i and it is ready to execute another task when we have a non insertion-based scheduling policy. The inner max block in the EST equation returns the ready time, i,e , the time when all data needed by t_i has arrived at processor P_j. After a task t_j is scheduled on a processor P_j, the earliest start time and the earliest finish time of t_i on processor P_j is equal to the actual start time, $AFT(t_m)$ is the actual completion time of t_i parent node t_m. After all tasks in a graph are scheduled, the scheduled, the schedule length (i,e., overall completion time) will be the actual finish time of the end task t_{end}.

2.2. *HEFT* Algorithm

According to the above model and definition Topcuoglu [12] proposed this famous HEFT algorithm. With the algorithm, we assume that an example of 4 DAG tasks are scheduled on the heterogeneous processors. We assume that a task set of DAG (T_1, T_2, T_3, T_4) run on a system of 4 identical processors (P_1, P_2, P_3, P_4). Simultaneously, the number on the upper corner of each subtask represents its $\overline{c_{i,k}}$ and the arrows represent their precedence constraints. A directed relation G_i (j, k) between subtasks $t_{i,j}$ and $t_{i,k}$ means that $t_{i,j}$ is a predecessor of $t_{i,k}$, and the latter subtask have to wait for all of its predecessors to complete their execution before it can start its own. Each DAG task $t_{i,j}$ on multiprocessor of task execution cost ($w_{i,1}$, $w_{i,2}$, $w_{i,3}$, $w_{i,4}$) and the upward rank of a task $t_{i,j}$ as shown Figure 1.

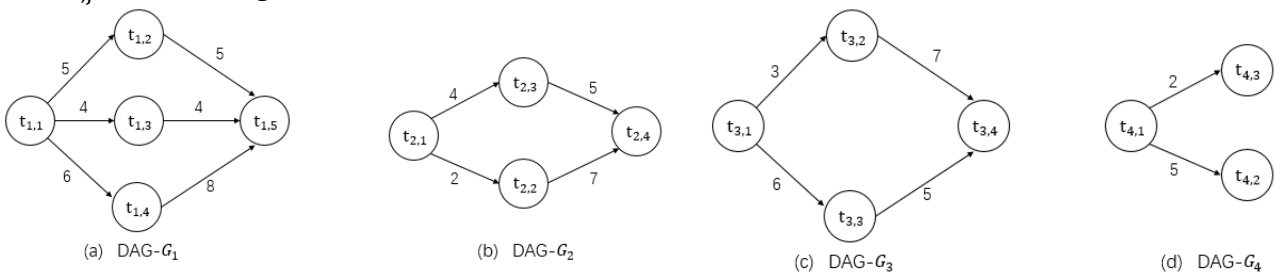

Figure 1 DAG task instances

Table 1 Execution time of 4 DAG tasks and $Rank_u(t_i)$

t_i	$t_{1,1}$	$t_{1,2}$	$t_{1,3}$	$t_{1,4}$	$t_{1,5}$	$t_{2,1}$	$t_{2,2}$	$t_{2,3}$	$t_{2,4}$	$t_{3,1}$	$t_{3,2}$	$t_{3,3}$	$t_{3,4}$	$t_{4,1}$	$t_{4,2}$	$t_{4,3}$
$w_{i,1}$	12	14	23	20	6	10	9	8	10	11	13	9	13	9	13	12
$w_{i,2}$	14	11	22	21	9	13	13	7	12	7	10	7	11	8	11	10
$w_{i,3}$	11	13	19	18	8	15	13	10	11	5	11	5	9	7	9	8
$w_{i,4}$	8	10	27	16	10	9	12	13	15	6	7	6	8	5	8	9
$rank_u(t_i)$	54.5	25.3	35	37	8.3	42.5	30.8	26.5	12	37.3	25.5	24	10.3	22.5	10.3	9.8

The HEFT algorithm is an application scheduling algorithm for a bounded number of heterogeneous processors, which has two phases: a task prioritizing phase for computing the priorities of all tasks and a processor selection phase for selecting the tasks in the order of their priorities.

According to figure 1 and table 1 the data, using the HEFT algorithm, it is readily to get the makespan values of four DAG (G_1,G_2,G_3,G_4) in the four uniprocessor scheduling, and record for $t_{makespan-1}$=41， $t_{makespan-2}$=35， $t_{makespan-3}$=23， $t_{makespan-4}$=15. HEFT algorithm is the scheduling algorithm of single DAG on multiprocessor. The situation of scheduling 4 DAG tasks is shown in fig.3-2. Based on the assumption that deadline for any task is greater than the makespan value, we assume that the deadline for the four DAG tasks is $t_{Deadline-1}$=52， $t_{Deadline-2}$=61， $t_{Deadline-3}$=29 and $t_{Deadline-4}$=27.

2.3. $HEFT_{S-L}$ Algorithm

According to HEFT algorithm, every DAG task G_i has a makespan value, and the value is easy to calculate. We propose two concepts of relative stringency and laxity about multi-resource scheduling. We use $t_{makespan-G1}$ as the Makespan value for a DAG task, and the initial relative rigor is defined as formula 6.

$$r_{rio} = t_{makespan-G}/t_{Deadline-G} \tag{6}$$

In the resource scheduling mapping process, the value of $t_{makespan-G1}$ is continuously executed with the subtasks of the DAG task, and be changed dynamically. At some point, the makespan value of the remaining unperformed tasks is represented as $t_{remain-G1}$, the start time of the remaining unexecuted tasks is ts_G and the earliest end time is tf, so the Makespan value is defined as formula 7.

$$t_{remain-G} = tf_G - ts_G \tag{7}$$

According to LLF algorithm, the laxity of each DAG task can be calculated as follows.

$$L_{Gi} = t_{Deadline-G} - t_{makespan-G} \tag{8}$$

Similarly, at some point, the laxity of the remaining unexecuted tasks can be calculated, as formula 9.

$$L_{re(LG)} = t_{Deadline-G} - tf_G \tag{9}$$

So initial relative laxity is defined as formula 10.

$$L_{LG}=L_{Gi}/(L_{Gi}+ t_{makespan-G}) \tag{10}$$

The relative laxity of the remaining unexecuted task is defined as formula 11.

$$L_{re(LG)}=L_{re(LG)}/(L_{re(LG)}+ t_{remain-G}) \tag{11}$$

In the same way, we can define the initial relative rigor as formula 12.

$$r_{rio} = t_{makespan-G}/(L_{Gi} + t_{Deadline-G}) \tag{12}$$

The relative rigor of the remaining unexecuted task is defined as formula 13.

$$r_{re(rio)} = t_{makespan-G}/(L_{re(LG)} + t_{Deadline-G}) \tag{13}$$

Therefore, according to the above parameters, we describe the degree of urgency between multiple DAG tasks, and the degree of urgency is defined as formula 14.

$$P_{priority}=L_{LG}/r_{\mathrm{rio}}=(L_{Gi}/L_{Gi}+ t_{makespan-G})/(t_{makespan-G}/L_{Gi}+ t_{makespan-G}) \tag{14}$$

The relative degree of urgency is defined as formula 15.

$$P_{re(priority)} = L_{re(LG)}/t_{remain-G} \tag{15}$$

Do not add any text to the headers (do not set running heads) and footers, not even page numbers, because text will be added electronically.

For a best viewing experience the used font must be Times New Roman, on a Macintosh use the font named times, except on special occasions, such as program code.

2.4. Example of $HEFT_{S-L}$ Algorithm

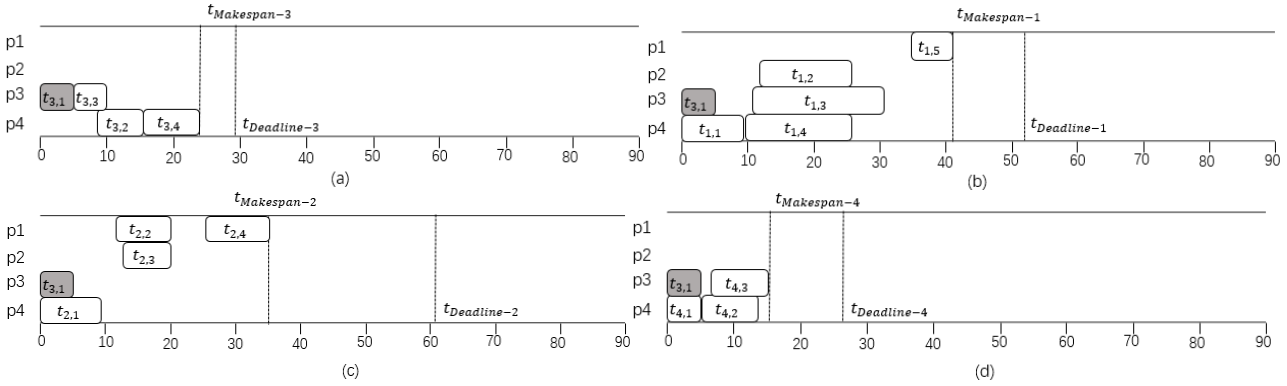

Figure 2 the urgency changes of other tasks after algorithm selection of $t_{3,1}$ task.

First of all, based on the above four DAG tasks scheduling of HEFT algorithm, we reschedule

the tasks with $HEFT_{S-L}$ algorithm. According to the four DAG scheduling instances in figure 1, the emergency degree of G_1, G_2, G_3 and G_4 respectively is $P_{riority-G1}=L_{G1}/\ t_{makespan-G1}=11/41=0.2683$ and $P_{riority-G2}=L_{G2}/\ t_{makespan-G2}=26/35=0.7429$ and $P_{riority-G3}=L_{G3}/\ t_{makespan-G3}=6/23=0.2609$ and $P_{riority-G4} = L_{G4} / \ t_{makespan-G4} =12/15=0.8$. Therefore, you can get the highest level of emergency of G_3, so you should select subtask scheduling from G_3 first. According to the HEFT algorithm and table 1, we select the maximum task $t_{3,1}$ of $rank_u$ value in G_3 to the processor queue, and start to execute. When selecting $t_{3,1}$ with HEFT algorithm, the $t_{3,1}$ task already occupies the processor resources. Therefore, the remaining tasks and the rest of the DAG task in subsequent G_3 changed the relative degree of emergency, repeated this process, will be finished all the task scheduling.

3. Results

In order to verify the effectiveness of $HEFT_{S-L}$ algorithm, this paper analyzes the efficiency of the algorithm from the completion rate of DAG task.

From the perspective of resource management, the throughput of tasks is an important index for evaluating the effectiveness of an algorithm based on the scheduling of deadlines. For users, the DAG task submitted for upload should be completed before the deadline. Apparently, scheduling process adopts the measure of DAG term emergency degree of the more accurate parameters (based on the multiple emergency degree parameters), so the more number of DAG tasks within deadlines, the higher the completion rate. After the scheduling of this algorithm, the number of DAG tasks that is completed in the term constraint is n_f, the total DAG quantity is n, and the completion rate of the scheduling is $R_f = n_f/n * 100\%$. This paper uses this parameter to measure the scheduling performance of the algorithm.

According to the four DAG instances in the figure 1 and table 1, the $HEFT_{S-L}$ algorithm is scheduled to be compared with the EDF algorithm and LLF improved algorithm, which compares the performance of different algorithm and the completion rate of DAG tasks.

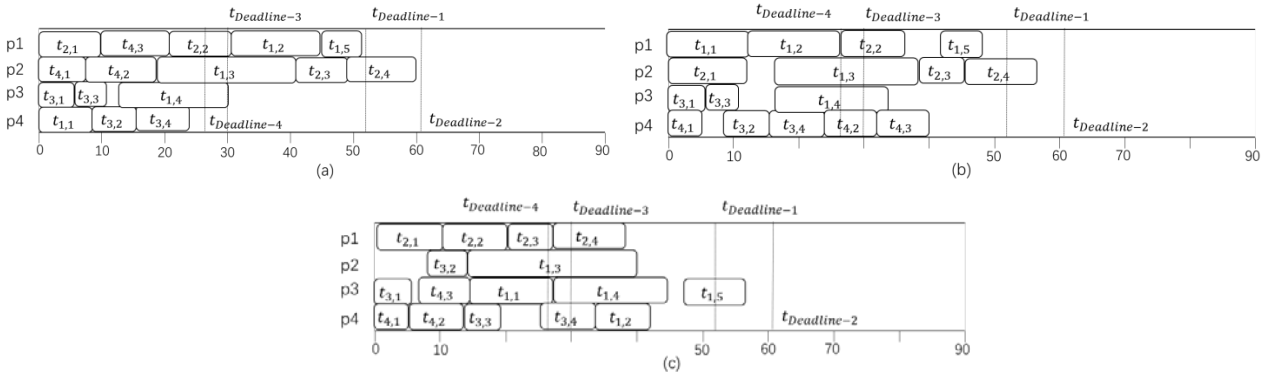

Figure 3 Scheduling 4 DAG tasks using the (a) $HEFT_{S-L}$ algorithm (b) EDF algorithm (c) LLF algorithm

In this paper, the scheduling problem of multi-dag task Shared resources is proposed for a large number of business process models with deadline constraints, and the DAG task is allocated to multi-processor execution. In this paper, an $HEFT_{S-L}$ algorithm based on relative rigor and relative relaxation is proposed to study the scheduling throughput of DAG tasks and the concept of task overload is proposed according to the algorithm. In addition, in order to measure the performance of algorithm, this paper will DAG task completion (within the prescribed deadline for completion and the number of total ratio) and task scheduling fairness as the performance index of the algorithm. In addition, in the experiment, we used a large number of simulation data to perform scheduling and comparison performance of $HEFT_{S-L}$ algorithm and EDF algorithm and improved LLF algorithm respectively. In terms of the final experimental results, the $HEFT_{S-L}$ algorithm can determine the priority of tasks according to the task scheduling process. At the same time, according to different user submission deadline for different throughput, $HEFT_{S-L}$ algorithm can greatly optimize the task

scheduling, and when the overload phenomenon occurred in the process of scheduling, can improve the completion of tasks and fairness by reasonable pop-up mission. The performance aspect is more efficient than the EDF algorithm and the improved LLF algorithm.

About this article, there are several aspects of future research. (1) For the description of task priority, we can consider more factors such as the cost of task scheduling; (2) considering the $HEFT_{S-L}$ algorithm for scheduling DAG tasks that are not submitted simultaneously by users.

Acknowledgements

The research is partly supported by the National Natural Science Foundation of China (Grant No. 61602005), MOE Youth Project of Humanities and Social Sciences (No.14YJCZH169), Natural Science Foundation of Anhui Province (1608085MF130), Humanities and Social Sciences in Universities of Anhui Province (SK2016A007), Anhui University Doctor Startup Fund.

References

[1] Wang W, Zhu K, Ying L, et al. Maptask scheduling in mapreduce with data locality: Throughput and heavy-traffic optimality[J]. IEEE/ACM Transactions on Networking, 2016, 24(1): 190-203.

[2] Xu R, Wang Y, Huang W, et al. Near-optimal dynamic priority scheduling strategy for instance‐intensive business workflows in cloud computing[J]. Concurrency and Computation: Practice and Experience, 2017.

[3] Zhang H, Cheng P, Shi L, et al. Optimal DoS attack scheduling in wireless networked control system[J]. IEEE Transactions on Control Systems Technology, 2016, 24(3): 843-852.

[4] Xu J, Liu C, Zhao X, et al. Resource management for business process scheduling in the presence of availability constraints[J]. ACM Transactions on Management Information Systems (TMIS), 2016, 7(3): 9.

[5] Saikrishna P S, Pasumarthy R, Bhatt N P. Identification and Multivariable Gain-Scheduling Control for Cloud Computing Systems[J]. IEEE Transactions on Control Systems Technology, 2017, 25(3): 792-807.

[6] Convolbo M W, Chou J. Cost-aware DAG scheduling algorithms for minimizing execution cost on cloud resources[J]. The Journal of Supercomputing, 2016, 72(3): 985-1012.

[7] Yue S, Ma Y, Chen L, et al. Dynamic DAG scheduling for many-task computing of distributed eco-hydrological model[J]. The Journal of Supercomputing, 2017: 1-23.

[8] Kaleem R, Barik R, Shpeisman T, et al. Adaptive heterogeneous scheduling for integrated GPUs[C]//Proceedings of the 23rd international conference on Parallel architectures and compilation. ACM, 2014: 151-162.

[9] Xu Y, Li K, Hu J, et al. A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues[J]. Information Sciences, 2014, 270: 255-287.

[10] Kang S, Kang D, Yang H. Real-time co-scheduling of multiple dataflow graphs on multi-processor systems[C]//Proceedings of the 53rd Annual Design Automation Conference. ACM, 2016: 159.

[11] Rongbin Xu, Xin Liu, Zhuangzhuang Yang, Xing Guo, Ying Xie, Jianguo Wu. Real-Time DAG Scheduling Method based on the Deadline of Tasks. Computer Integrated Manufacturing Systems, 2016, 22(2): 455-464.

[12] H.Topcuoglu, S.Hariri, W.Min-You. Performance-effective and Low-complexity Task Scheduling for Heterogeneous Computing[J]. IEEE Transactions,2002,13:260-274.