Simulation Modelling Practice and Theory 63 (2016) 83–103

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

A fast predictive algorithm with idle reduction for heterogeneous system scheduling

Anan Niyom[*], Peraphon Sophatsathit, Chidchanok Lursinsap

Advanced Virtual and Intelligent Computing Research Center, Department of Mathematics and Computer Science, Faculty of Science, Chulalongkorn University, Thailand

ARTICLE INFO

Article history:
Received 30 September 2015
Revised 9 February 2016
Accepted 9 February 2016
Available online 9 March 2016

Keywords:
Task graph
Heterogeneous systems
Algorithm running time
Schedule length

ABSTRACT

A heterogeneous task scheduling algorithm called Predict and Arrange Task Scheduling (PATS) algorithm was proposed to achieve a lower bound time complexity with minimum schedule length. Two major steps were introduced, i.e. earliest finish time with level-based task scheduling and idle slot reduction. In the first step, tasks are scheduled according to their predicted earliest finish time from the candidate task list and their dependencies. Scheduling is performed one level at a time starting from top level and transcend downward. In the second step, the idle time slots in each processing unit are minimized. Two sets of experiments were designed to evaluate the merits of proposed algorithm. The first experiment involved the task graphs used by other methods. These graphs are all synthesized. The second experiment concerned the task graphs derived from real world applications such as montage work flow, molecular dynamic code. The experimental results showed that the PATS algorithm yielded better average schedule length ratio, running time, and efficiency than the compared algorithms.

1. Introduction

A heterogeneous system is more practical than a homogeneous system for distributed computing due to the actual availability of various processing units with different computing capabilities. The processing speed of each unit and also the installed software applications may be different. Scheduling and assigning a set of tasks from a host unit to proper heterogeneous units to achieve a minimum response time as well as minimum energy consumption is very challenging. Usually, a set of tasks is represented by a directed acyclic task graph. Recently proposed algorithms for scheduling and assigning tasks in heterogeneous units focused on two optimization issues concerning energy conservation and makespan reduction. The first issue of energy conservation covers the minimization of energy usage of the system and at the same time the process to shorten the scheduled makespan under limited power [1–6]. The second issue of makespan reduction emphasizes the process to reduce the scheduled time span or idle slots. Typically, minimizing energy consumption of a system employs basic methods such as server selection, task ordering, and simulation technologies such as dynamic voltage scaling, dynamic frequency scaling, and dynamic power management [7–12]. But reducing makespan must involve the minimization of schedule length in various situations [13,14]. The underlying algorithms for these two issues can be categorized into four groups which are list-based, clustering-based, duplicate-based, and genetic based algorithms.

[*] Corresponding author. Tel.: +66 877127360.
E-mail addresses: anan.niyom@gmail.com (A. Niyom), speraphon@gmail.com (P. Sophatsathit), lchidcha@gmail.com (C. Lursinsap).

http://dx.doi.org/10.1016/j.simpat.2016.02.006
S1569-190X(15)30094-0/© 2016 Elsevier B.V. All rights reserved.

A list-based scheduling separates task prioritization and server selection [15,16] by traversing throughout a dependent task graph to order the tasks for scheduled assignment on suitable processing units. The list of ordering is an important output which determines the performance of the algorithm. A duplicate-based scheduling is similar to that of the list-based scheduling but redundant tasks are added on different processing units to reduce communication cost and the consequent schedule length [17,18]. A clustering-based scheduling groups tasks suitable for each processing unit and orders the tasks in each cluster [19,20]. It also reduces the communication or computation costs. A genetic algorithm [21,22] is based on natural and genetic selection to find an optimum solution to the problem. Moreover, the first three groups are classified as heuristic whereas the last group is considered an evolutionary algorithm. An evolutionary algorithm usually takes longer time to reach the desired solution although the scheduling length is minimized at the expense of extra time. Its running time could vary greatly for problems with the same number of input tasks and processing units depending on the solution refinement process to assign input tasks on a suitable processing unit. On the other hand, a heuristic algorithm usually focuses on finding the solution using the least extra scheduling overhead. However, the yielded scheduling length so obtained is not always the shortest. Therefore, both evolutionary and heuristic algorithms are appropriated in different situations. For example, in the situation where the same task graph is to be processed many times, the solution obtained from an evolutionary algorithm which yields the shortest scheduling length is preferable since the extra scheduling overhead is only necessary during the initial processing of the task graph. The heuristic approach is, on the other hand, favorable in the situation where varieties of task graphs are to be processed.

In this study, we developed the heuristic scheduling algorithm with emphasis on the problem of scheduling a task graph, whose tasks have different computational requirements, in a heterogeneous computing system. All processing units in the system have different computing speeds and different available software applications. The objectives are to minimize the length of makespan of the given task graph and time complexity of scheduling process. Since the previously imposed constraints of the list-based scheduling approach is similar to ours, therefore the concept of list-based scheduling was adopted and modified to solve our studied problem.

The rest of this paper is organized as follows. Section 2 summarizes the concept of each compared algorithm. Section 3 formulates the studied problem and constraints. Section 4 explains the proposed algorithm. Sections 5 provides the experimental procedures and results. Section 6 discusses the experimental results. Section 7 concludes the paper.

2. Related works

Many algorithms for task assignment in heterogeneous systems have been proposed. We will briefly summarize those algorithms that are pertinent to our study, such as heterogeneous earliest finish time, low complexity performance effective task scheduling, look-ahead, constrained earliest finish time, and predict earliest finish time algorithm.

Heterogeneous earliest finish time or HEFT algorithm was proposed by Topcuoglu et al. [23]. This algorithm consists of two phases: task prioritizing and processor selection. The task prioritizing phase calculates the priority queue in an upward ranking called $rank_u$ based on computation and communication costs for each task. The processor selection phase calculates the earliest execution finish time using insertion-based policy to assign any pending tasks in an idle time slot lying between already scheduled tasks on a processing unit. This algorithm has $O(v^2 \times p)$ time complexity for v vertices or number of tasks and p processing units.

Low complexity performance effective task scheduling (PETS) algorithm was proposed by Ilavarasan and Thambidurai [24]. This algorithm constructs a priority queue by sorting each task on each scheduling level according to average computation cost, data transfer cost, and rank of predecessor tasks. The algorithm computes the earliest finish time of each task and assigned a task by insertion-based policy in an idle time slot between already scheduled tasks on a processing unit. The time complexity of this algorithm is equal to $O(v^2)(p \times log\, v)$ for v vertices or number of tasks and p processing units.

Lookahead algorithm was proposed by Bittencourt et al. [25]. This algorithm, based on HEFT algorithm [23], calculates the estimated finish time of all children of the considered task in queue of $rank_u$ to find the shortest finish time of subsequent tasks. Therefore, it increases the time complexity of the original HEFT by a factor of $(p \times c)$ for p processing units and average c children per task. In case of tasks spreading evenly on each level of scheduling graph, the average number of children must not exceed v/l, where l is the number of levels in the graph. The time complexity of lookahead HEFT algorithm is equal to $O(v^3 \times p^2/l)$.

Constrained earliest finish time (CEFT) algorithm was proposed by Khan [26]. This algorithm find a set of constrained critical paths by traveling and pruning downward from the start task to the exit task of the graph. A constrained critical path is defined in terms of average execution cost, transmission weight, and weight of predecessor of each task. A schedule queue is established by selecting a task or more than one task ready to be scheduled from the longest constrained critical path. The process is repeated for the second longest constrained critical path, and so on, until all tasks are added to the schedule queue in a round-robin manner. The obtained schedule queue contains several constrained critical sub-paths, each of which may have more than one task. These constrained critical sub-paths are then assigned to the processing unit to yield the minimum finish time. The time complexity of this algorithm is equal to $O(n \times p \times (n + m + deg_{in}))$ for v vertices or number of tasks, p number of processing units, m edges, and deg_{in} in-degrees of task graph.

Predict earliest finish time (PEFT) was proposed by Arabnejad and Barbosa [27]. This algorithm is based on optimistic cost table. There are two phases which are computing task prioritization and processor selection. Task prioritization is determined from sorted average of optimistic cost tables on each processor. A processor is selected from the results of task