CHAPTER II

RELATED WORKS

2.1 General distributed task scheduling approaches

Parallel or distributed task scheduling problems have been studied by many researchers [15], [16], [17], [18] and a number of algorithms for solving these problems have been proposed [19], [20], [21], [22], [23]. Each proposed algorithm has different advantages or limitations. Example interesting algorithms for solving distributed task scheduling problems can be categorized in three kinds, namely, task duplication heuristics, clustering heuristics, and list scheduling heuristics. Task duplication heuristics (TDH) approach is employed to reduce computing time. The algorithm reduces the communication time between tasks by pushing redundant tasks to different servers [24] and [25]. This method, however, can be very complex and requires high energy consumption and execution cost as redundant tasks are executed on different servers. As the overhead to find an optimal solution is high, it is only effective in the case that the system has a high ratio of communication and execution cost.

Clustering heuristics (CH) approach focuses on the reduction of transmission cost. In this approach, tasks are grouped into clusters based on their transmission edge before mapping each group to appropriate servers [18], [23], [26], [27]. An example of CH algorithm is introduced by Liou and Palis in 1997 [27]. For this algorithm the scheduling is achieved in four steps; clustering all tasks into groups, merging clusters to a number of servers, mapping individual cluster to a server, and determining the order of task execution. The cluster of tasks which has the highest communication cost will be assigned first. Then the process continues until all clusters are assigned. This way, the computing load on clusters or servers and communication traffic among the clusters can be balanced.

List scheduling heuristics (LSH) is considered the most common scheduling algorithm [28], [29], and [19]. This type of algorithm generally contains two parts; task prioritization and server selection. In the first part, a score is assigned to each task based on its computation cost. The tasks are then prioritized according to the ranking of the score. In the second part, the execution time for each task on every server is evaluated. Tasks are then assigned to the servers which give the least execution time for that task. The Min-Min algorithm proposed by Li et al. in 2011 [19] is a good example of task scheduling algorithm employing LSH approach. This algorithm has been modified from the original Min-Min algorithm in such a way that each task are considered to be dependent. The earliest finishing time for the tasks on every device is calculated. The tasks are then assigned to the devices which give the shortest earliest finishing time.

2.2 Selected prominent distributed task scheduling algorithms

Other related task scheduling algorithms which are referred to in this study are described as follows. The algorithms include the Heterogeneous Earliest Finish Time (HEFT) algorithm, the low complexity Performance Effective Task Scheduling (PETS) algorithm, the Lookahead algorithm, the Constrained Earliest Finish Time (CEFT) algorithm, and the Predict Earliest Finish Time (PEFT) algorithm.

The HEFT algorithm was proposed by Topcuoglu et al. [30]. The algorithm yields a time complexity of $O\left(v^2 \times p\right)$, where v is the number of vertices or tasks and p is the processing units. For this algorithm, two phase process, namely task prioritizing and processor selection, are employed. In the first phase, tasks are prioritized based on their computation and communication costs. In the next phase, earliest execution finish time for all tasks is calculated using insertion-based policy. The tasks are then assigned to an idle time slot that lies between the already scheduled tasks on a processing unit.

The PETS algorithm was proposed by Ilavarasan and Thambidurai [31]. The algorithm yields a time complexity of $O\left(v^2\right)\left(p \times log\ v\right)$, where v is the number of vertices or tasks and p is the processing units. For this algorithm, a priority queue is

constructed by sorting each task in each scheduling level according to their average computation cost, data transfer cost, and rank of predecessor tasks. Earliest finish time of each task is then computed and the tasks are assigned by using insertion-based policy in an idle time slot between the already scheduled tasks on a processing unit.

The Lookahead algorithm was proposed by Bittencourt et al. [32]. The algorithm is based on the HEFT algorithm [30] having the time complexity to be $O\left(v^3 \times p^2 / l\right)$, where v is the number of vertices or tasks and p is the processing units. This is greater than that of the original HEFT by a factor of $\left(p \times c\right)$, where c is the number of the average number of children per task. In the HEFT with Lookahead algorithm. The estimated finish time of all children of the considered task in queue is calculated. Thus, the algorithm looks up the forwarding step to compute the shortest finish time in subsequent tasks. In the case where tasks are spread evenly on every level of scheduling graph, the average number of children must not exceed v / l, where l is the number of level in the graph.

The CEFT algorithm was proposed by Khan [33]. In this algorithm, a constrained critical path is determined from the average execution cost, transmission weight, and weight of predecessor of each task. Calculation begins from the start task of the tasks along constrained critical paths and prunes downward to the exit task of the graph. Nodes are scheduled following a traversal of the critical path in a round-robin scheme by moving them from the adjacent constrained critical path to the longest constrained critical path. The processing unit which gives the minimum finish time for the scheduled node in the constrained critical paths is then assigned to all remaining nodes. The time complexity for this algorithm is $O\left(v^3 \times p\right)$, where v is the number of vertices or tasks and p is the processing units.

The PEFT was proposed by Arabnejad and Barbosa [34]. This algorithm is based on optimistic cost table. The algorithm contains two phases, computing task prioritization and processor selection. Task prioritization is determined from sorted average of optimistic cost table on each processor. Processors selection are taken

from the results of task optimistic earliest finish time (O_{EFT}) which is determined from the minimal summation of optimistic cost tables and earliest finish time. The tasks are then scheduled to each server using insertion-based policy. The time complexity of this algorithm is in the order of $O(v^2 \times p)$ for v vertices or number of tasks and p processing units.

For this work, scheduling algorithms with an emphasis on energy saving aspect were studied. It was found from the study that the overall communication energy could be reduced by grouping tasks to be executed using the same server. As some tasks could not be executed or clustered on the same server due to their inherent restrictions, such a scenario would be further elaborated with details of problem formulation, constraints of task characteristics, and server competency. It was also found from this study that when energy consumption was considered, important aspects which should be addressed include algorithm overhead and wait time of the main processing unit before all tasks were executed. Problem formulation and all pertinent issues of this study are described in the sections that follow.