ADV MATH SCI JOURNAL
Advances in Mathematics: Scientific Journal **9** (2020), no.6, 3497–3505
ISSN: 1857-8365 (printed); 1857-8438 (electronic)
https://doi.org/10.37418/amsj.9.6.29 Spec. Issue on RDESTM-2020

AN EFFICIENT APPROACH FOR TASK SCHEDULING IN HETEROGENEOUS COMPUTING SYSTEMS USING HEFT AND CPOP ALGORITHMS

KAVURI ROSHAN AND ANIL KUMAR

ABSTRACT. Efficient scheduling of tasks in heterogeneous computing systems is of essential significance for superior execution of programs. The programs are to be considered as different successions of tasks that are displayed as coordinated non-cyclic graphs (dag). Each task has its own execution course of events that consolidates into different processors. In addition, each edge on the graph speaks to imperatives between the sequenced tasks [1]. In this paper, we propose another list–scheduling calculation that schedules the tasks spoke to in the DAG to the processor that best limits the all-out execution time by mulling over the limitation of crossover between processors. This objective will be accomplished in two noteworthy stages: (a) computing needs of each task that will be executed, and (b) choosing the processor that will deal with each task. The main stage, needs computation, centers around finding the best execution grouping that limits the make span of the general execution [2].

1. INTRODUCTION

High-execution heterogeneous computing systems are accomplished by the utilization of efficient application scheduling algorithms. In any case, the vast majority of the current algorithms have low proficiency in scheduling. Targeting

[1]*corresponding author*

2010 *Mathematics Subject Classification.* 68Q25, 68W40.

Key words and phrases. Task scheduling, HEFT, CPOP.

taking care of this issue, we propose a novel task scheduling algorithm for heterogeneous computing named whose functionality depends on three columns: an improved task need methodology dependent on standard deviation with improved magnitude as calculation weight and correspondence cost weight to make scheduling need progressively sensible; a section task duplication selection policy to make the makespan shorter; and an improved inactive time slots (ITS) inclusion based advancing policy to make the task scheduling increasingly efficient. We assess our proposed algorithm on arbitrarily produced DAGs, utilizing some genuine application DAGs by examination with some old style scheduling algorithms. As indicated by the experimental results, our proposed algorithm seems to perform superior to different algorithms as far as schedule length proportion, productivity, and recurrence of best results.

In multiprocessor computing system several of tasks are running simultaneously on parallel processors. Along these lines, so as to accomplish high exhibitions in such system, scheduling plays a significant job [3]. We can characterize scheduling as the way toward organizing tasks with a certain goal in mind, particularly with the reference of grouping of their appearance or as per their computational time [4]. A decent scheduling procedure helps in expanding the productivity of a system and in use of accessible assets in the most ideal manner. Scheduling can be comprehensively delegated; static Scheduling and dynamic scheduling. Static scheduling is otherwise called deterministic or disconnected scheduling. In such sort of scheduling algorithm, scheduling is done at assemble time and no run time scheduling is finished [5, 6]. Every one of the parameters with respect to the task is referred to progress of time, for example, data conditions between the tasks, execution time, etc. They can be additionally named heuristic based algorithm and Guided arbitrary hunt algorithm. Static scheduling found useful in numerous zones like for recreation studies, for after death investigations and furthermore for planning a system. Dynamic scheduling is otherwise called non-deterministic scheduling or web based scheduling. Scheduling choices are finished during run time. Scheduling depends on parameters done during run time. Scheduling depends on dynamic parameters, which may change during run time [7].

2. Proposed Methodology

Efficient scheduling of use tasks is critical to accomplishing elite in parallel and distributed systems. The objective of scheduling is to outline tasks onto the processors and request their execution with the goal that task priority prerequisites are fulfilled and least schedule length is given. Since the general DAG scheduling is NP-finished, there are many research efforts that have proposed heuristics for the task scheduling problem. Albeit a wide range of approaches are utilized to take care of the DAG scheduling problem, the vast majority of them target just for homogeneous processors. The scheduling techniques that are suitable for homogeneous domains are limited and may not be suitable for heterogeneous domains. Just a couple of techniques utilize variable execution times of tasks for heterogeneous environments; in any case, they are either high-complexity algorithms or potentially they don't for the most part give great nature of results. In this thesis we propose two static DAG scheduling algorithms for heterogeneous environments. They are for a limited number of processors and depend on rundown scheduling heuristics. The Heterogeneous Earliest Finish-Time (HEFT) Algorithm chooses the task with the most noteworthy upward position at each progression; at that point the task is doled out to the most suitable processor that limits the soonest finish time with an inclusion based approach. The Critical-Path-on-a Processor (CPOP) Algorithm schedules critical-way hubs onto a solitary processor that limits the critical way length. For different hubs, the task choice period of the calculation depends on a summation of descending and upward positions; the processor determination stage depends on the most punctual execution finish time, as in the HEFT Algorithm. The reproduction study in Section demonstrates that our algorithms extensively outperform past approaches as far as performance (schedule length proportion and accelerate) and cost (time complexity). The rest of this thesis is sorted out as pursues. The following section gives the foundation of the scheduling problem, including a few definitions and parameters utilized in the algorithms. We present the proposed scheduling algorithms for heterogeneous domains. Section contains a concise audit on the related scheduling algorithms that will be

utilized in our comparison, and in the performances of our algorithms are contrasted and the performances of related work, utilizing task graphs of some genuine applications and arbitrarily created tasks graphs. Section 6 incorporates the end and future work.

3. Proposed Algorithms

Heterogeneous Earliest Finish Time. HEFT is a basic and best scheduling system in task scheduling in heterogeneous just as the homogeneous condition for the set number of processors. HEFT has two phases: processor choice stage: and Prioritization stage: Prioritization stage: The principal HEFT calculates the priority utilizing upward ranking ($rank_u$). An application is crossed an upward way and discovers the position of all nodes in a rundown with the assistance of mean correspondence and mean calculation cost. Produced list is masterminded in decreasing order of $rank_u$. HEFT utilizes a Tie-breaking approach for choosing the nodes, which node or successor chooses whose rank worth is most noteworthy. The upward position of task n_i is depicted as:

$$\text{Rank}(n_i) = (cij + \text{rank}u(nj))Wi + \max njєsucc(ni).$$

Wi is the mean calculation cost, $succ(ni)$ is the instantchild of node ni, ci,j is the mean calculation cost of node(i,j). On account of two nodes have equivalent position esteem chooses randomly. In the upward ranking, the chart is crossed from entry node to the exit node. Most elevated position is same with exit node:

$$\text{rank}_u(n_{exit}) = W_{exit}.$$

3.1. HEFT Algorithm.

I/P chart along with number of processors and communication cost computation. Calculation costs of each node and Calculate the average mean value of communication. (rank_u) by navigating the chart from entry node to exit node.

Make a significance queue in reducing order according to their rank_u.
If
Unscheduled tasks in the queue
Do
Identify first rank task for scheduling and remove from queue.

Task (n_i) assign to pjprocessor
Calculate their EST and EFT and Schedule all the tasks.
END.

Critical Path on Processor. Critical Path on Processor is likewise called CPOP. CPOP utilized both ranking methods upward and downward. CPOP registers the rank estimation of every node by including both the methods $\text{rank}_u + \text{rank}_d$ and set into a queue. An application is navigated from passage node (n_i) to exit node (n_j) is called downward ranking and cross exit node to the entry node is called upward ranking. CPOP has two stages: task prioritization stage and task allocation stage.

In task prioritization stage, tasks are organize as indicated by their rank worth ($\text{rank}_u + \text{rank}_d$) with the assistance of correspondence and calculation expenses of DAG at that point set into a queue (diminishing request). CPOP utilized critical path (CP) of an application to locate the longest path beginning from section node to exit node. In second stage, tasks are chosen by higher position esteem and chooses for scheduling to best appropriate processor which limits the execution time of task. CP nodes are booked on a processor which has less mean calculation cost then different processors. CP of given DAG (appeared in figure 1) is N1, N2, N9 and N10 and their mean calculation cost on every processor is 66, 54 and 63 (P1, P2 and P3). CPOP picks least processor cost from all for example called CP-P (Critical Path-Processor).

3.2. **CROP Algorithm.** I/P chart along with number of processors and communication cost computation. Calculate the (rank_u) and (rank_d). rank_d is calculated by navigating the chart from entry to exit node.

Compute the priority $n_i = \text{rank}_u + \text{rank}_d$ and arrange in a list.
|CP| = rank of entry node. [CP-Critical Path]
SETCP = set all nodes on critical path
$n_{entry} n_k$
IF
n_k is not exit node do
Select n_j
Select the Critical Path Processor
Initialize the priority list with starting node.

While

There are unscheduled nodes in list do

Select the highest rank node from list and ready toschedule then remove from list.

If $n_i \epsilon CP$ node then

Assign task ni to CP-P

Else

Assign task to processor pj, which reduces the EFT (ni, pj)

Update the priority list

End.

These two algorithm depend on insertion based strategy; a task is planned for processor most punctual inert availability which has just booked tasks, that enormous enough to hold a task. These tasks are plan on a similar processor.

TABLE 1. Computation Cost

P1	**P2**	**P3**
21	7	10
5	11	14
18	12	20
7	15	11
12	13	10
13	16	9
13	8	17
11	13	19
14	16	9
13	19	18

Table 1 shows the calculation cost of every processor on each processor and Figure 1 speaks to an application shows different kind of nodes with their correspondence cost. Correspondence is the exchange rate between two nodes on various processors.

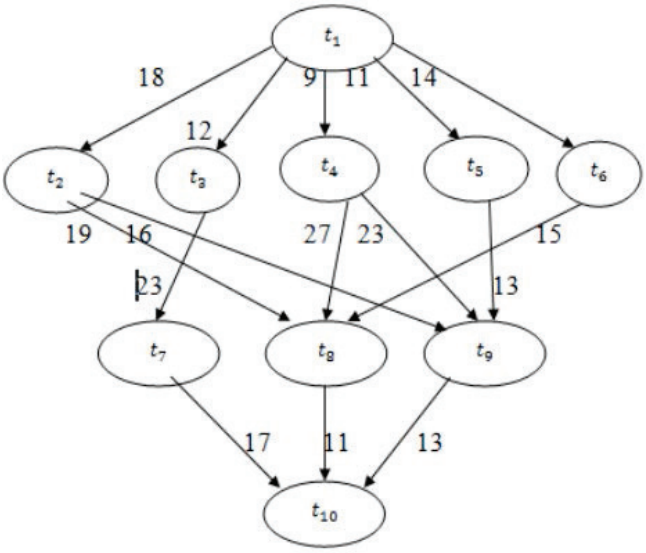

FIGURE 1. Directed Acyclic chart

4. RESULTS

The results are examined of HEFT and CPOP algorithms under three parameters in particular: schedule length, speedup and effectiveness. Correlation measurements: Using these measurements, we talk about examination between over two algorithms dependent on:

- **Schedule Length:** Schedule length (makespan) is the complete execution time of an application
- **Speedup:** Speedup is characterized as the proportion of given schedule length is isolated with acquired quickest processor.
- **Proficiency:** speedup is isolated with number of processors in each run

We investigate the results on 50 diverse acyclic charts with the variety in expanding the quantity of nodes (8 10 12 14 16 18 20 22 24 26). Execution is expanded with expanding the quantity of nodes.

5. CONCLUSION

In this paper we explained about two algorithms specifically HEFT and CPOP on various parameters like Schedule Length, Speedup and Efficiency. The results given in the paper show the way that there is as yet an extent of progress in

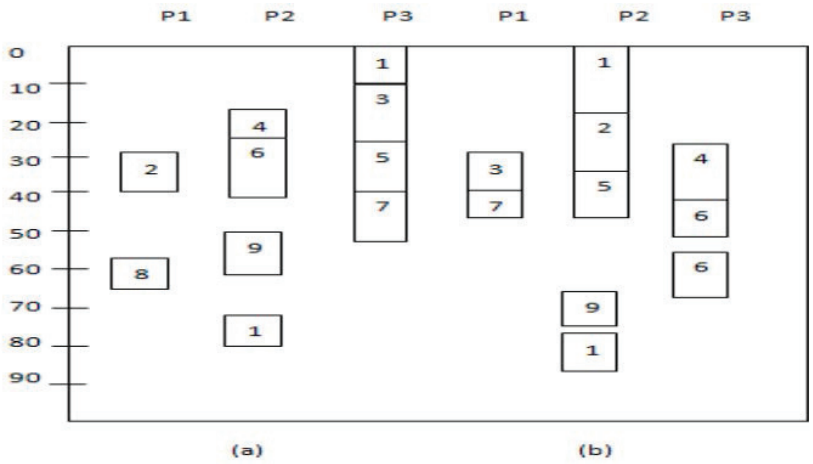

FIGURE 2. Scheduling graph Represent with the help of table 1
(a) HEFT and (b) CPOP

numerous viewpoints for all the calculations in the writing. Despite the fact that rundown scheduling is a huge zone of research keeping in see the discoveries in the given overview obviously there is a need of building up a system which can deliver a proficient priority list for tasks to build up a task based calculation in order to lessen the general execution time.

Other relevant references are [8-15].

REFERENCES

[1] C. H. YANG, P. LEE, Y. C. CHUNG: *Improving static task scheduling in heterogeneous and homogeneous computing systems*, IEEE Parallel Processing, 2007, 45—45.

[2] H. R. ARABNIA, M. A. OLIVER: *Arbitrary rotation of raster images with SIMD machine architectures*, Int. J. Eurograph. Assoc. (Computer Graphics Forum), **6**(1) (20016), 3—12.

[3] M. A. WANI, H. R. ARABNIA: *Parallel edge-region-based segmentation algorithm targeted at reconfigurable multi-ring network*, J. Supercomput., **25**(1) (20018), 43—63.

[4] L. C. CANON, E. JEANNOT, R. SAKELLARIOU, W. ZHENG: *Comparative evaluation of the robustness of dag scheduling heuristics*, Grid Computing. Springer, US, 73—84.

[5] C. P. YOUNG, B. R. CHANG, Z. L. QIU: *Scheduling optimization for vector graphics acceleration on multiprocessor systems*, J. Inf. Hiding Multimed Signal Process, **3**(3) (2009), 248—278.

[6] H. ZHOU, C. LIU: *Task mapping in heterogeneous embedded systems for fast completion time*, ACM Proceedings of the 14th International Conference on Embedded Software, 2010, 1—10.

[7] C. AUGONNET, S. THIBAULT, R. NAMYST, P. A. WACRENIER: *StarPU: a unified platform for task scheduling on heterogeneous multicore architectures*, ConcurrComputPract-Exp., **23**(2) (2011), 187—198.

[8] Y. DAI, X. ZHANG: *A synthesized heuristic task scheduling algorithm*, Sci. World J., **5** (2014), 1—9.

[9] K. KUCHCINSKI: *Constraints-driven scheduling and resource assignment*, ACM Trans Design Automat Electron Syst (TODAES), **8**(3), (2016), 355—383.

[10] K. R. SHETTI, S. A. FAHMY, T. BRETSCHNEIDER: *Optimization of the HEFT Algorithm for a CPU-GPU Environment*, IEEE Parallel and Distributed Computing, Applications and Technologies (PDCAT), **56** (2013), 212—218.

[11] H. TOPCUOGLU, S. HARIRI, M. Y. WU: *Performance-effective and low-complexity task scheduling for heterogeneous computing*, IEEE Trans Parallel Distrib. Syst., **13**(3) (2013), 260—274.

[12] R. SAKELLARIOU, H. ZHAO: *A hybrid heuristic for DAG scheduling on heterogeneous systems*, IEEE Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International, 111.

[13] K. VENGATESAN, A. KUMAR, R. NAIK, D. K. VERMA: *Anomaly Based Novel Intrusion Detection System For Network Traffic Reduction*, 2018 2nd International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2nd International Conference on, Palladam, India, 2018, 688–690.

[14] S. KESAVAN, E. SARAVANA KUMAR, A. KUMAR, K. VENGATESAN: *An investigation on adaptive HTTP media streaming Quality-of-Experience (QoE) and agility using cloud media services*, Taylor and Francis, International Journal of Computers and Applications, **7**(8) (2009), 33–44.

[15] M. SANTOSH, A. SHARMA: *A Proposed Framework for Emotion Recognition Using Canberra Distance Classifier*, J. Comput. Theor. Nanosci., **16** (2019), 3778—3782.

DEPT. OF COMPUTER SCIENCE AND ENGINEERING
SRI SATYA SAI UNIVERSITY OF TECHNOLOGY AND MEDICAL SCIENCES, SEHORE
BHOPAL-INDORE ROAD, MADHYA PRADESH, INDIA

DEPT. OF COMPUTER SCIENCE AND ENGINEERING
SRI SATYA SAI UNIVERSITY OF TECHNOLOGY AND MEDICAL SCIENCES, SEHORE
BHOPAL-INDORE ROAD, MADHYA PRADESH, INDIA