JOURNAL OF CRITICAL REVIEWS

ISSN- 2394-5125 VOL 7, ISSUE 8, 2020

EXAMINATION OF FLUCTUATING PARAMETERS FOR TASK SCHEDULING IN HETEROGENEOUS COMPUTING SYSTEMS

Kavuri Roshan[1], Dr. Anil Kumar[2]

[1]Research Scholar, Dept. of Computer Science & Engineering, Sri Satya Sai University of Technology & Medical Sciences, Sehore, Bhopal-Indore Road, Madhya Pradesh, India.
[2]Research Guide, Dept. of Computer Science & Engineering, Sri Satya Sai University of Technology & Medical Sciences, Sehore, Bhopal Indore Road, Madhya Pradesh, India.

Received: 09.03.2020 Revised: 27.04.2020 Accepted: 20.05.2020

ABSTRACT: Generally utilized computing systems are heterogeneous in nature, involving interconnected assets which vary in computational ability of handling hubs and system data transmission. Because of this decent variety, a proficient heuristic is required to accomplish elite in heterogeneous computing framework. In our proposed scheduling calculation, Heterogeneous Edge and Task Scheduling (HETS), we schedule the correspondence between the tasks of utilization diagram onto the system connections of fluctuating data transfer capacity, and schedule these tasks of various calculations on the system processors in the wake of thinking about the computational ability of the accessible processors. In HETS, the prioritization is finished by computing the edge need just as the hub need. HETS calculation chooses the task after the entirety of its approaching edges are scheduled. The proposed calculation limits the correspondence overhead of the application chart edges and gets diminished schedule length as far as the general execution time. Execution of the proposed calculation is examined by fluctuating parameters of the standard task graphs just as on genuine directed acyclic graphs (DAGs) application, for example, Digital Shake, Gaussian End, and Montage. Broad recreation results show the adequacy of HETS calculation as far as decreased make range and improved Schedule Length Ratio (SLR) for the given tasks.

KEYWORDS: Bandwidth-aware scheduling, Heterogeneous computing systems, Task scheduling, Directed acyclic graph.

© 2020 by Advance Scientific Research. This is an open-access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/) DOI: http://dx.doi.org/10.31838/jcr.07.08.358

I. INTRODUCTION

Heterogeneous computing systems are every now and again being utilized for explaining register escalated applications that are sub-separated into various tasks. For the most part these figure concentrated applications are spoken to utilizing directed acyclic chart (DAG). Task scheduling heuristics are significant for heterogeneous computing systems (HCS) as they give elite on these HCS by doling out the tasks to accessible assets with greatest computational speed.

Productive scheduling of tasks is basic to improve the exhibition of a computing framework to such an extent that the general execution time for the whole scheduled task is limited. As of late created PC systems are distinctive both as far as engineering plan and execution when contrasted with the customary systems. Hence the past methods are not completely pertinent to these ongoing PC systems.

In scheduling, a directed chart must be mapped onto the processors. Inside an application task chart not all tasks are free. There is priority imperative among the reliant tasks. Regularly, hubs of the diagram speak to tasks of a given application while the edges show the correspondence or move of information among those hubs, which have between task conditions. The task of these tasks on the given machines or processors is known as task scheduling. Task scheduling is a NP-complete issue.

For task scheduling most regularly utilized algorithms are chart theoretic algorithms. The sources of info required by the chart theoretic algorithms are application diagram clarified as far as task calculation time, correspondence cost, and priority imperative. These algorithms are utilized for conveying the application graphs into tasks and designating every one of these tasks to the appropriate execution hubs. Various task scheduling heuristics were proposed on the suspicions, for example, tasks are independent and that there is no correspondence delay. These suppositions are unreasonable as there is consistently correspondence delay in parallel systems and the tasks of an application diagram are for the most part dependent.

Different heuristics are figured for optimizing the answer for task scheduling issue. The tasks are scheduled so as to diminish the inactive time on the machines and correspondence overhead. The scheduling of a DAG on the topological system not just includes the mapping of task hubs on the processor yet in addition remembers the mapping of the edges for the connections of the system. At the point when every one of the sources of info are accessible, i.e., when parent hubs have effectively executed, at exactly that point the task is executed. The tasks are executed without being hindered once they start. Every hub has its own related calculation cost that assigns the execution time of each hub on the processor. On account of homogeneous processor, the expense is the equivalent for each sort of processor.

II. LITERATURE REVIEW

Yuxin Wang (2016), High-execution heterogeneous computing systems are accomplished by the utilization of productive application scheduling algorithms. In any case, the greater part of the present algorithms have low effectiveness in scheduling. Targeting tackling this issue, we propose a novel task scheduling calculation for heterogeneous computing named HSIP (heterogeneous scheduling calculation with improved task need) whose usefulness depends on three columns: an improved task need technique dependent on standard deviation with improved extent as calculation weight and correspondence cost weight to make scheduling need increasingly sensible; a passage task duplication choice strategy to make the make span shorter; and an improved idle time slots (ITS) addition based optimizing arrangement to make the task scheduling progressively effective.

E. Ilavarasan (2007), a heterogeneous computing condition is a suite of heterogeneous processors interconnected by high-speed systems, in this manner promising high speed preparing of computationally escalated applications with different computing needs. Scheduling of an application displayed by Directed Acyclic Graph (DAG) is a key issue while focusing on high performance in this sort of condition. The issue is commonly tended to regarding task scheduling, where tasks are the schedulable units of a program. The task scheduling issues have been demonstrated to be NP-finished all in all just as a few limited cases. In this investigation we present a basic scheduling calculation dependent on list scheduling, in particular, low multifaceted nature Performance Effective Task Scheduling (PETS) calculation for heterogeneous computing systems with unpredictability O (e) (p+ log v), which gives effective outcomes to applications spoke to by DAGs.

Priyanka Mehta (2016), Scheduling computation tasks on processors is the key issue for a propelled computing. Rundown scheduling has continually been a theme of discussion for the specialists because of its inclination of tackling high intricacy issues with least unpredictability and to evaluate the extra scheduling issues for the applied lattice. The task close by is to do diminish the general time uses for task fruition. A ton of prior research works have additionally remembered prioritization for the employments to decrease the computation cost and most punctual completion time of the framework. This paper presents an exchange about as of late considered scheduling strategies to be specific: Heave, CPOP. These two procedures are contrasted and each other in rest of Schedule Length, Speedup and Proficiency on various number of directed acyclic graphs.

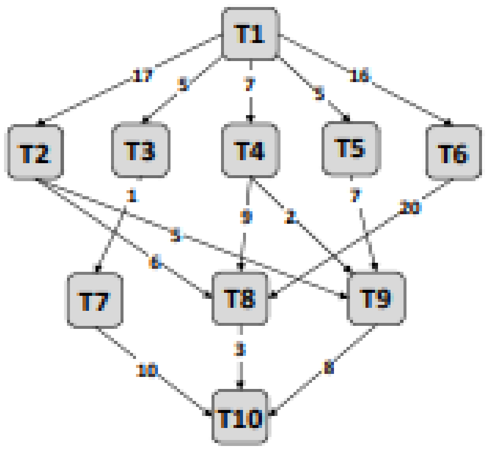
Mohammad I. Daoud (2007), Effective task scheduling is fundamental for acquiring high performance in heterogeneous circulated computing systems (HeDCSs). Be that as it may, finding an effective task schedule in HeDCSs requires the consideration of both the heterogeneity of processors and high bury processor correspondence overhead, which results from non-insignificant information development between tasks scheduled on various processors. In this paper, we present another high-performance scheduling calculation, called the longest dynamic critical path (LDCP) calculation, for HeDCSs with a limited number of processors. The LDCP calculation is a rundown based scheduling calculation that uses another credit to effectively choose tasks for scheduling in HeDCSs. The proficient determination of tasks empowers the LDCP calculation to create high-quality task schedules in a heterogeneous computing condition. The performance of the LDCP calculation is contrasted with two of the best existing scheduling algorithms for HeDCSs: the Weight and DLS algorithms.

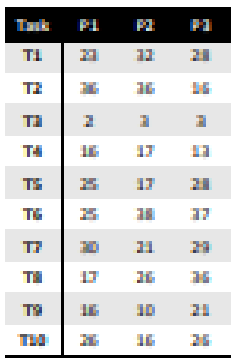
Muhammad Fahad Khan (2018), Effective task scheduling in heterogeneous computing systems is a difficult and pivotal task. Bury process correspondence and the heterogeneity of assets assumes a significant job in task scheduling. To accomplish the proficiency tasks are allocated to most appropriate processor while limiting correspondence cost. This legitimately builds the performance and is eluded as consummation time. Such issues in an appropriated framework are considered as NP difficult issues. Numerous arrangements are proposed in writing for illuminating this issue. Directed Acyclic Graph (DAG) is likewise used to comprehend the issue of performance in circulated systems. Another heuristic is in this paper dependent on DAG is proposed for task scheduling on tasks request, normal correspondence cost and best accessible processor/asset. The trial study shows the proposed approach give promising outcomes.

III. RESEARCH METHODOLOGY

DAG Scheduling

The problem addressed in this paper is the static scheduling of a single application on a heterogeneous system. An application can be represented by a Directed Acyclic Graph (DAG), G = (V, E, P, W), as shown in Figure 1.





Task	P1	P2	P3
T1	22	32	28
T2	26	16	16
T3	2	3	3
T4	16	17	13
T5	25	17	28
T6	25	18	17
T7	30	21	29
T8	17	26	26
T9	16	10	21
T10	26	16	26

Fig. 1: Application Model and Computation Time Matrix of the Tasks in each Processor

Where V is the set of v nodes, and each node $v_i \in V$ represents an application task, which includes its instruction that must be executed on the same machine. E is the set of e communication edges between tasks, each $e(i, j) \in E$ represents the task-dependency constraint such that task n_i should complete its execution before task n_j can be started. P is the set of p heterogeneous processors available in the system. W is a v × p computation cost matrix, where v is the number of tasks and p is the number of processors in the system. $w_{i,j}$ gives the estimate time to execute task v_i on machine p_j . The mean execution time of task ni can be calculated by $w_i = (P\ _{j \in P}{}^{w}{}_{i,j}\ )/p$. Each edge e (i, j) ∈ E is associated with a non-negative weight $c_{i,j}$ representing the communication cost between the tasks n_i and n_j. Note that, when task i and j are assigned to the same processor, the real communication cost is considered to be zero because it is negligible compared to inter processor communication costs. Additionally, in our model, I consider that processors are connected in a fully connected topology. The execution of tasks and communications with other processors can be done for each processor simultaneously and without contention. Also, the execution of any task is considered non preemptive. These model simplifications are common in list scheduling problem, and I consider them in order to have a fair comparison to the state of the art algorithms.

Heterogeneous earliest finish time (HEFT) algorithm: It is likewise two-stage task scheduling calculation for a limited number of heterogeneous processors. The primary stage in particular, task-organizing stage is to allot the need to all tasks. To allot need, the upward position of each task is figured. The upward position of a task is the critical path of that task, which is the highest entirety of correspondence time and normal execution time beginning from that task to leave task. In view of upward position need will be allocated to each task. The subsequent stage (processor determination stage) is to schedule the tasks onto the processors that give the most punctual completion time for the task. It utilizes an inclusion based arrangement which considers the conceivable addition of a task in a soonest idle time space between two previously scheduled tasks on a processor, ought to be at any rate equipped for computation cost of the task to be scheduled and furthermore scheduling on this idle time opening should safeguard priority imperatives. The time multifaceted nature of Heave calculation is equivalent to O (v2 x p) where v is the quantity of tasks in a thick graph and p is the quantity of processors.

In the first stage of Prioritization: HEFT estimates the importance of the tasks by using the upward ranking ($rank_u$). An application operation is moving over in upward path and achieves entire list nodes with its rank with the service provided by the mean communication and performance cost. An obtained list is organized in systematized descending order of the $rank_u$. Every task with upward rank is defined as:

$$\text{Rank}(n_i)=W_i + \max_{nj\epsilon succ(ni)}(C_{ij} + rank_{u(nj)})$$

Wi describes the mean computation cost, succ(ni) is described as the immediate child of node ni, ci,j is described as the mean communication cost of node (i,j). If any two nodes have when two nodes obtain the similar rank value, it will prefer indiscriminately. The attained graph is moved from the end node to start node. The maximum rank value is similar to the end node

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

These two algorithms are based on insertion based policy; a task is scheduled in processor earliest idle time slot which has already scheduled tasks, that large enough to hold a task. These tasks are schedule on the same processor.

Table 1: Computation Cost

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

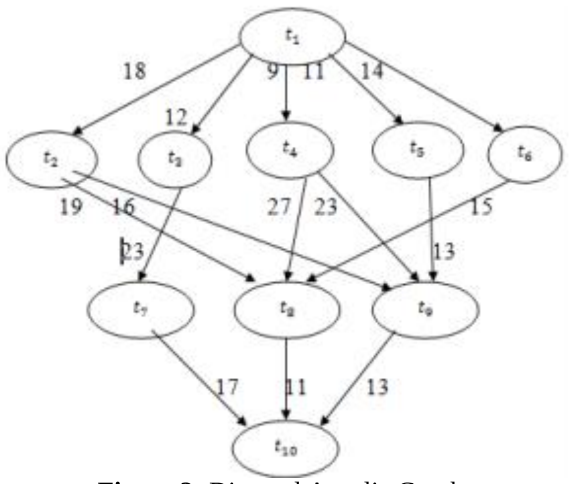


Figure 2: Directed Acyclic Graph

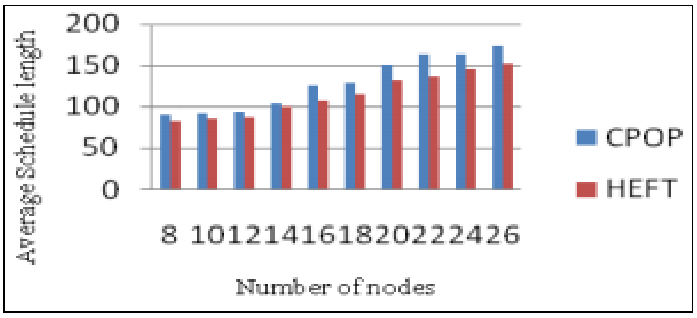
Table 1 represents the computation cost of each processor on every processor and Figure 2 represents an application shows various type of nodes with their communication cost. Communication is the transfer rate between two nodes on different processors. We discuss this application or DAG on HEFT and CPOP, in heterogeneous environment.

Critical path on a processors (CPOP) algorithm: This is like Heave calculation, yet it utilizes various systems in each stage. The principal stage in particular, task organizing stage is to allot the need to each task. In this stage, upward position (given in Heave calculation) and descending position esteems for all tasks are figured. The descending position is processed by including normal execution time and correspondence time beginning from section task to the task barring execution time of the task for which descending position is figured. The total of descending and upward position is utilized to dole out the need to each task. At first, the passage task is the chosen task and set apart as a critical path task. A prompt successor (of the chose task) that has the highest need esteem is chosen and it is set apart as a critical path task. This procedure is rehashed until the leave hub is come to. In the subsequent stage, task with highest need is chosen for execution. On the off chance that the chose task in on the critical path, at that point it is scheduled on the critical path processor. The critical processor is the one that limits the aggregate computation expenses of the tasks on the critical path; else, it is appointed to a processor, which limits the most punctual execution finish time of the task. The time unpredictability of CPOP calculation is equivalent to O (v2 x p) where v is the quantity of tasks in a thick graph and p is the quantity of processors.

ISSN- 2394-5125 VOL 7, ISSUE 8, 2020

Compute the upward rank ($rank_u$) and downward rank ($rank_d$). Downward rank is computed by traversing the graph from entry node to exit node.

```
Compute the priority ni = ranku+ rankd and arrange in a list.
|CP|= rank of entry node. [CP-Critical Path]
SETCP = set all nodes on critical path
$n_k \leftarrow n_{entry}$
While
$n_k$ is not exit node do
Select $n_j$ where ($n_j$€ succ ($n_k$)) and (priority ($n_j$== |CP|))
Select the Critical Path Processor (CP-P) ∑ni∈CP node
$w_{ij} \leftarrow$ computation cost.
Initialize the priority list with starting node.
While
 There are unscheduled nodes in list do
Select the highest rank node from list and ready to
 Schedule then remove from list.
If $n_i$€ CP node then
Assign task $n_i$ to CP-P
Else
Assign task to processor pj, which reduces the EFT ($n_i$, $p_j$)
Update the priority list
End.
```



Graph 1: Comparison of HEFT and CPOP w.r.t to Average Schedule Length

Graph1 shows HEFT gives better results than CPOP for average schedule length. HEFT is 12% more efficient than CPOP. If number of nodes is increased then schedule length is also increased.

IV. CONCLUSION

In this paper we examined two algorithms in particular HEFT and CPOP on various parameters like Schedule Length, Speedup and Effectiveness. These algorithms are scheduled on various numbers of DAGs in static task scheduling algorithms in heterogeneous condition. Results give us HEFT is superior to CPOP for all talked about parameters. As think about expanding the quantity of hub. It gives better outcomes to every one of the parameters. It gives better outcomes to every one of the parameters.. The outcomes given in the paper show the way that there is as yet an extent of progress in numerous perspectives for every one of the algorithms in the writing.

V. REFERENCES

[1] Baldeep Singh (2016), 'A Survey of Scheduling Algorithms for Heterogeneous Systems and Comparative Study of HEFT and CPOP Algorithms' *Published* by: http://www.ijert.org ISSN: 2278-0181, Vol. 5 Issue 05, May-2016

[2] P. Thambidurai (2007), 'Low Complexity Performance Effective Task Scheduling Algorithm for Heterogeneous Computing Environments' © 2007 *Science Publications*, 94-103, 2007

[3] Guan Wang (2016), 'HSIP: A Novel Task Scheduling Algorithm for Heterogeneous Computing' *Hindawi Publishing Corporation*, Volume 2016, Article ID 3676149, 11 pages

[4] E.U. Munir, S. Mohsin, A. Hussain, M.W. Nisar, and S. Ali, "SDBATS: a novel algorithm for task scheduling in heterogeneous computing systems," in *Proceedings of the IEEE 27th International Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW '13)*, pp. 43–53, IEEE, Cambridge, Mass, USA, May 2013.

[5] Mahmood Fazlali, Mojtaba Sabeghi, Ali Zakerolhosseini And Koen Bertels, Efficient Task Scheduling For Runtime Reconfigurable Systems, Journal Of Systems Architecture, Volume 56, Issue 11, Pages 623-632, November 2010.

[6] K. Vijaya Kumar(2019), 'Extended Optimization Procedures for Static List based Task Scheduling Algorithms for HeDCS' International Journal of Recent Technology and Engineering (IJRTE), ISSN: 2277-3878, Volume-8, Issue-2S11, September 2019

[7] Ayman El-Sayed(2015). 'Performance Enhancement of Scheduling Algorithm in Heterogeneous Distributed Computing Systems' *(IJACSA) International Journal of Advanced Computer Science and Applications,* Vol. 6, No. 5, 2015

[8] H. Arabnejad and J.G. Barbosa, "List scheduling algorithm for heterogeneous systems by an optimistic cost table," *IEEE Transactions on Parallel and Distributed Systems,* vol. 25, no. 3, pp. 682–694, 2014.

[9] J.K. Kim, S. Shivle, H.J. Siegel et al., "Dynamically mapping tasks with priorities and multiple deadlines in a heterogeneous environment," *Journal of Parallel and Distributed Computing,* vol. 67, no. 2, pp. 154–169, 2007.

[10] M I. Daoud and N. Kharma, "A high performance algorithm for static task scheduling in heterogeneous distributed computing systems," *Journal of Parallel and Distributed Computing,* vol. 68, no. 4, pp. 399–409, 2008.