International Journal of Computer Applications (0975 – 8887)
Volume 107 – No 4, December 2014

Task Scheduling Optimization in Heterogeneous Distributed Systems

Aida A. Nasr
Computer Science & Eng. Dept.,
Faculty of Electronic Eng.
Menouf 32952, Egypt.

Nirmeen A. El-Bahnasawy
Computer Science & Eng. Dept.,
Faculty of Electronic Eng.
Menouf 32952, Egypt.

Ayman El-Sayed
Computer Science & Eng.
Dept.,Faculty of Electronic Eng.
Menouf 32952, Egypt

ABSTRACT

Efficient task scheduling is essential for obtaining high performance in heterogeneous distributed computing systems. Several algorithms are proposed for heterogeneous distributed computing systems. In this paper, a new static scheduling algorithm is proposed called Highest Communicated Path of Task (HCPT) algorithm to efficiently schedule tasks on the heterogeneous distributed computing systems. Our algorithm is based on the list-scheduling technique. The algorithm not only is focused on reducing the *makespan*, but also provides better performance than the other algorithms in terms of speedup and efficiency. It consists of three phases, level sorting phase, task-prioritizing phase and processor selection phase. From the theoretical analysis of the HCPT algorithm with other algorithms for a Directed A-cyclic Graph (DAG), the better performance is observed.

Keywords

Static task scheduling, heterogeneous distributed computing systems, heuristic algorithm.

1. INTRODUCTION

The availability of high-speed networks and diverse sets of resources lead to a new platform, called as heterogeneous platform. Such a platform contains interconnected resources with different computing capabilities and different computing speeds. To run an application in this heterogeneous environment, several issues need to be considered such as partitioning the application, scheduling the tasks, etc. We will refer to such a system as Heterogeneous Distributed Computing System (HDCS) [1].

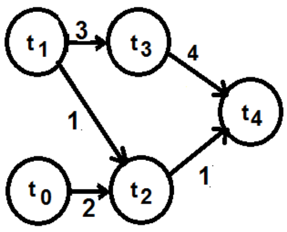
Task scheduling is of vital importance in HDCS since a poor task-scheduling algorithm can undo any potential gains from the parallelism presented in the application. In general, the objective of task scheduling is to minimize the completion time of a parallel application by properly mapping the tasks to the processors [2, 3, 4]. There are typically two categories of scheduling models: static and dynamic scheduling. In the static scheduling case, all the information regarding the application and computing resources such as execution time, communication cost, data dependency, and synchronization requirement is assumed available a priori. Scheduling is performed before the actual execution of the application. On the other hand, in the dynamic mapping a more realistic assumption is used. Very little a priori knowledge is available about the application and computing resources. Scheduling is done at run-time. In this paper, we focus on static scheduling [5, 6]. Static scheduling is classified into list-based, clustering and duplication based. List scheduling consists of two phases: a task prioritization phase, where a certain priority is computed and is assigned to each node of the DAG, and a machine assignment phase, where each task (in order of its priority) is assigned to machine that minimizes a suitable cost function. List scheduling is generally accepted as an attractive approach since it pairs low complexity with good results[4]. Examples of list-based algorithms are Heterogeneous Earliest Finish Time (HEFT) and Critical Path On Processor (CPOP) [7]. Another static scheduling category is task duplication based algorithms [8], in which tasks are duplicated on more than one processor to reduce the waiting time of the dependent tasks. The main idea behind duplication based scheduling is to utilize processor idling time to duplicate predecessor tasks. This may avoid transfer of results from a predecessor, through a communication channel, and may eliminate waiting slots on other processors [9].

In this paper, a new algorithm called Highest Communicated Path of Task (HCPT) is developed for static task scheduling for the HDCS with limited number of processors. The motivation behind this algorithm is to generate the high quality task schedule that is necessary to achieve high performance in HDCS. The developed algorithm is based calculating the average communication parents to give each node a priority, and the maximum child path with highest communication. Finally, our algorithm could decrease time of application.

The remainder of this paper is organized as follows. Section 2 discusses problem definition. Section 3 gives an overview of the related works. Section 4 presents our developed scheduling algorithm namely HCPT with examples. Section 5 discusses the results and finally this paper is concluded in section 6.

2. PROBLEM DEFINITION

A DAG represents a parallel application. A DAG that is defined by the tuple (T, E), where T is a set of n tasks and E is a set of e edges represents a parallel application. Each $t_i \epsilon T$ represents a task in the parallel application, which in turn is a set of instructions that must be executed sequentially in the same processor without interruption. Each edge $(t_i,t_j) \epsilon E$ represents a precedence constraint, such that the execution of $t_j \epsilon T$ starts after $t_i \epsilon T$ finishes its execution. t_i is a parent of t_j and t_j is a child of t_i. A task with no parents (i.e. root) is called an entry task (t_{entry}), and a task with no children (i.e. leaf) is called an exit task (t_{exit}). Each edge $(t_i,t_j) \epsilon E$ has a value that represents the communication cost of that edge. A task can start execution on a processor, if all parents have finished their execution and all data required from its parents become available to that processor. The speed of the inter-processor communication network is negligible. Therefore, when two tasks are scheduled on the same processor the communication cost between them can be ignored. The HDCS is represented by a set P of m processors that have diverse capabilities. The n×m computation cost matrix C stores the execution costs of tasks. Each element $C_{i,j} \epsilon$ C represents the estimated execution time of task t_i on processor p_j. Precise calculation of the running times of the tasks on the processors is unfeasible before running the application [10]. All processors in the HDCS are assumed to be fully connected. Communications between processors occur via independent communication units; this allows for concurrent execution of computation of tasks and communications between processors. After scheduling all the tasks of a parallel application on the processors of a HDCS, the schedule length is defined as the longest finish time of the HDCS processors. Fig.1 presents an example of a parallel application consisting of five tasks and a HDCS with two processors, where the application is represented as a DAG and the execution costs estimated for the five tasks on the HDCS are shown as a computation cost matrix [11].



a. DAG

Task	P_0	P_1
t_0	7	8
t_1	6	9
t_2	5	8
t_3	2	3
t_4	2	4

b. Computation cost matrix

Fig 1: Example of a DAG and Computation Cost Matrix.

Definition (1) $EST(t_i, P_j)$ [6]: Denotes the Earliest Start Time of a task t_i on a processor P_j and is defined as shown in Equation (1).

$$EST(t_i, P_j) = max\{ T_{Available}(P_j), max\{AFT(t_k)+c_{k,i}\}\} \ldots\ldots\ldots(1)$$

Where $T_{Available}(P_j)$ is the earliest time at which processor P_j is ready. $AFT(t_k)$ is the Actual Finish Time of a task t_k (where t_k is the parent of task t_i and $k=1, 2, \ldots, n$)on the processor P_j. $c_{k,i}$ is the communication cost from task t_k to task t_i,$c_{k,i}$ equal zero if the predecessor task t_k is assigned to processor P_j. For the entry task, $EST(t_{entry}, P_j)=0$.

Definition (2): Denotes the Earliest Finish Time of a task t_i on a processor P_j ($EFT(t_i, P_j)$) [6] and is defined in Equation (2).

$$EFT(t_i, P_j) = EST(t_i, P_j) + w_{i,j} \ldots\ldots\ldots(2)$$

Which is the Earliest Start Time of a task t_i on a processor P_j plus the computational cost $w_{i,j}$ of t_i on a processor P_j.

3. RELATED WORK

In this section, we give an overview of some algorithms as related work.

3.1 Critical Path on Processor Algorithm

The CPOP algorithm consists of two phases: prioritizing phase and processor selection phase [7]. In task prioritizing phase, the algorithm selects the task with the highest (upward rank + downward rank) value at each step. We can calculate upward rank and downward rank by Equations (3, 4) respectively.

$$Ranku(n_i) = \overline{w_i} + max_{n_j \in succ(n_i)}\left(\overline{c_{i,j}} + rank_u(n_j)\right) \ldots\ldots(3)$$

$$Rankd(n_i) = max_{n_j \in pred(n_i)}(\overline{c_{j,i}} + \overline{w_j} + rank_d(n_j) \ldots\ldots(4)$$

Where pred(n_i) is the set of immediate predecessors of task n_i. The algorithm targets scheduling of all critical tasks (i.e., tasks on the critical path of the DAG) onto a single processor, which minimizes the total execution time of the critical tasks. If the selected task is noncritical, the processor selection phase is based on earliest execution time with insertion-based scheduling. The algorithm has an O(n^2p) time complexity for n nodes and p processors.

3.2 Path-based Heuristic Task Scheduling Algorithm

The *PHTS* algorithm is proposed for a bounded number of heterogeneous processors consisting of three phases namely, a path-prioritizing phase, task selection phase, and processor selection phase [12]. Path prioritizing phase for computing the priorities for all possible paths. Each path is assigned by a value called rank (p_j), is given in Equation 5.

$$Rank(pj) = \sum_{t_i \epsilon p_j} \overline{w_i} + \overline{c_{i,succ(t_i)}} \ldots\ldots\ldots(5)$$

Where $\overline{w_i}$ is the average computation cost of a task t_i. It is computed by $\overline{w_i} = \sum_{j=1}^{m} w_{i,j}/m$, and $\overline{c_{i,succ(t_i)}}$ is the communication cost of edge from task t_i to its successor, if exists.

In task selection phase, the algorithm selects the unscheduled tasks from the paths in the sorted path list. During the task selection, the algorithm applies the following conditions on each task:

- The task should not be scheduled earlier.
- The task has no parents or its parents are scheduled.

Finally, a processor selection phase where the selected task is assigned to a processor in the set of processors that minimizes its finish execution time using the insertion-based scheduling policy [6].The algorithm has an O(np) time complexity for n nodes and p processors.

3.3 Expected Completion Time Based Scheduling Algorithm (ECTS)

ECTS algorithm consists of two phases namely, task prioritization phase and processor selection phase. The task-prioritizing phase consists of two stages such as level wise task priority stage and task selection stage [13]. In the first stage, the algorithm computes the priority for every task at each level by using Expected Completion Time (ECT) value. Average Computation Cost (ACC) and Maximum Data Arrival Cost (MDAC) compute this ECT. Next Equations (6, 7, 8) explains ACC, MDAC and ECT respectively.

$$ACC(t_i) = \sum_{j=1}^{m} {W_{i,j}}/{m} \ldots\ldots\ldots(6)$$

Where $W_{i,j}$ is the estimated execution time to complete task t_i on processor m_j.

$$MDCA(t_i) = max\ t_i \epsilon\ pred(t_j)\ (c_{i,j}) \ldots\ldots\ldots(7)$$

Where t_i is the set of predecessors of task t_j.

$$ECT(t_i) = ACC(t_i) + MDCA(t_i) \ldots\ldots\ldots(8)$$

The second stage related to the task selection in which the tasks are selected from all levels based on their priority. Moreover, in the second phase, the selected tasks are assigned to the best processor, which minimizes its EFT.

4. OUR SCHEDULING ALGORITHM

The developed Highest Communicated Path of Task (HCPT) algorithm consists of three phases, level sorting, task prioritization, and processors selection. The detailed explanation of each phase of the algorithm is given below:

Level sorting phase: In this phase, the given DAG is traversed in a top-down fashion to sort tasks at each level in order to group the tasks that are independent of each other.

Task prioritizing phase: In this phase, the HCPT algorithm selects level and gives a priority to its tasks. It computes the priority for each task according to new attribute called *Rank* as shown in Equation (9).

$$Rank(t_i)= MCP(t_i)+\max_{t_j \in succ(t_i)} \left(\overline{c_{i,j}} + Rank(t_j)\right) \ldots\ldots(9)$$

Where MCP(t_i) refers to Mean Communication of Parents. It is computed by Equation (10).

$$MCP(t_i)=(\sum_{j=1}^{n} C_{j,i})/n \ldots\ldots\ldots\ldots(10)$$

Where n is the number of Parents, $C_{j,i}$is the communication between parent t_j and task t_i. The algorithm starts from t_{exit} where Rank (t_{exit})=MCP(t_{exit}). Fig. 2 shows HCPT algorithm steps. After the algorithm assigns a priority for each task in selected level, it creates a new Tasks List (TL), in which the HCPT algorithm sorts all level tasks in decreasing order to execute the next phase.

Processor Selection Phase: the HCPT algorithm calculates EFT of task t_i by Equation (2) for each processor, and selects the processor that has a minimum EFT to assign the task by using the insertion-based scheduling policy [7].

```
Generate the DAG
Sort the DAG levels according to dependency ordering
For each level $L_k$
{
  For each task $t_i$ in $L_k$
      Compute
          $Rank(t_i)= MCP(t_i)+\max_{t_j \in succ(t_i)} \left(\overline{c_{i,j}} + Rank(t_j)\right)$
  End for
  Create new Tasks List TL
  Sort all tasks in decreasing order of Rank value in TL
  For each processor $P_m$ in the processor set ($P_m \in Q$) do
      Compute EFT($t_i$, $P_m$) value
  End for
      Assign task $t_i$ to the processor $p_m$ that minimizes EFT
    using the insertion based scheduling policy
}
End for
```

Fig 2: Highest Communicated Path of Task (HCPT) Algorithm.

The insertion-based scheduling policy considers the possible insertion of a task in an earliest idle time slot between two already-scheduled tasks on a processor. The length of an idle time-slot, i.e., the difference between execution start time and finish time of two tasks that were consecutively scheduled on the same processor, should be at least capable of computation cost of the task to be scheduled. Additionally, scheduling on this idle time slot should preserve precedence constraints. Time complexity is the amount of time taken to assign every task to specific processor according to specific priority. Our algorithm has O(N^2P) time complexity for N number of tasks and P number of processors.

Case Study:

Considering the application DAG shown in Fig.3, Table 1 shows the computation matrix. Initially the HCPT algorithm sorts tasks into levels by applying level sorting phase. DAG in Fig. 3 has four levels. Task t_0 and t_1 belong to L_0, t_2 and t_3 belong to L_1 and so on. In the task-prioritizing phase, the algorithm gives a priority for each task according to equation 9. It starts from the exit tasks (L_3) where Rank(t_{exit})=MCP(t_{exit}). Rank(T_6)=(16+4)/2=10 and Rank(t_7)=(10+2+8)/3=6.667, then the algorithm go to the next level L_2. Rank (t_4)=(17+8)/2 +10+6.667=29.167 and Rank(t_5)=(4+1)/2+2+6.667=11.167 and so on. The algorithm sorts tasks of each level according Rank value. Table 2 shows the stepwise trace of the HCPT algorithm. In processor selection phase, the HCPT algorithm computes EFT for every task at each processor and assigns the task to the processor with minimum EFT. The generated schedule length after applying the HCPT algorithm and other algorithms shown in Fig.4. The schedule length generated by PHTS, CPOP, ECTS and HCPT algorithms respectively are 109, 125, 101 and 99. Therefore, the HCPT algorithm has shorter execution length than the other algorithms. This leads to good utilization of processors in the system.

Table 1.Computation Matrix

t_i	P_0	P_1
t_0	12	7
t_1	63	2
t_2	48	22
t_3	12	36
t_4	59	31
t_5	6	25
t_6	10	49
t_7	42	18

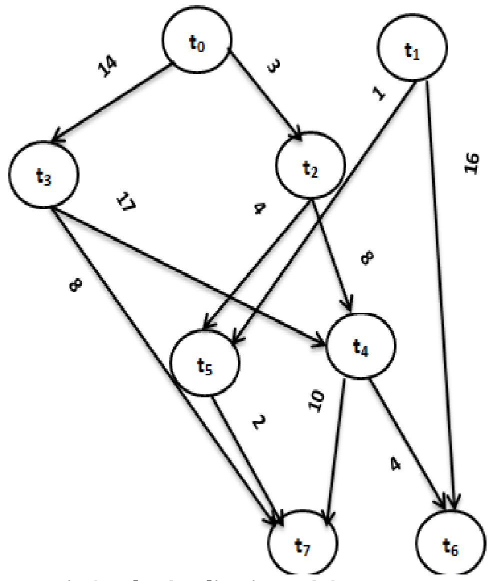
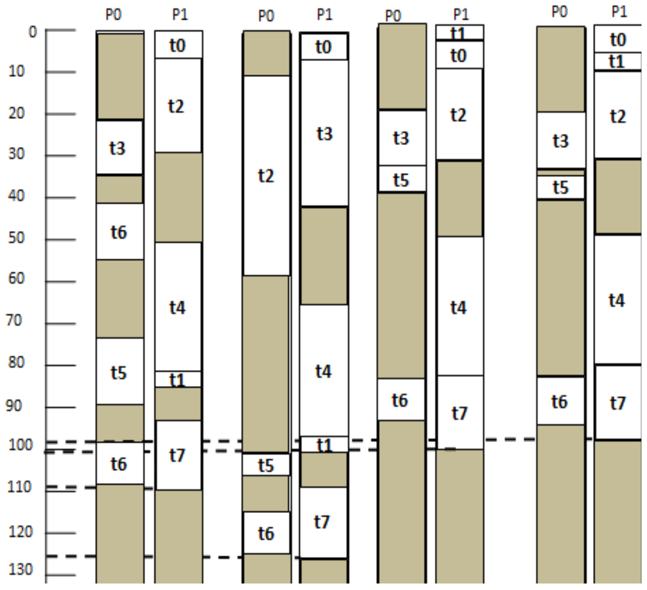


Fig 3: The Application DAG

Table 2. Stepwise Trace of HCPT algorithm

L_k	t_i	$Rank(t_i)=MCP(t_i) + \max_{t_j \in succ(t_i)}\left(\overline{c_{i,j}} + Rank(t_j)\right)$	Priority
1	t_0	0+14+60.167=74.167	1
	t_1	0+16+10=26	2
2	t_2	3+8+29.167=40.167	2
	t_3	14/1+17+29.167=60.167	1
3	t_4	(17+8)/2+10+6.667=29.167	1
	t_5	(1+4)/2+(2+6.667)=11.167	2
4	t_6	(4+16)/2+0=10	1
	t_7	(10+2+8)/3+0=6.667	2



(a) PHTS) (b) CPOP (c) ECTS (d) HCPT

Fig 4: The schedules generated by Algorithms.

5. RESULTS AND DISCUSSIONS

5.1 Simulation Environment

A simulator had been built using visual *C# .NET 4.0* on machine with configuration: Intel(*R*) Core(*TM*) *i3 CPU M 350 @2.27GHz*, *RAM* of *4.00 GB*, and the operating system is window 7, 64-bit.

To test the performance of *HCPT* algorithm with the other algorithms a set of randomly generated graphs created by varying a set of parameters that determines the characteristics of the generated *DAGs*. These parameters described as follows:

- DAG size: n (i.e. the number of tasks in the DAG).
- Density:

We use "*sameprob*" and "*layrprob*" methods to generate the *DAG* [14, 15]. Let *A* denote a task connection matrix with elements *a(i,j)*, where $0 \le i \le n$, and $0 \le j \le n$, represent the task number (t_0 is the entry dummy node and t_n is the exit dummy node). When *a(i,j)=1*, t_i precedes task t_j, when *a(i,j)=0*, t_i and t_j are independent of each other. In the "*sameprob*" edge connection method, *a(i,j)* is determined by independent random values defined as follows:

$P[a(i,j)=1]=p$ for $1 \le i < j \le n$ and $P[a(i,j)=0]=1-p$ for $1 \le i < j \le n$, $P[a(i,j)=0]=1$ if $i \geqq j$, where p indicates the probability that there exists an edge (precedence constraint) between t_i and t_j . In another method, *"layrprob"*, firstly the number of levels *L* in the task graph is generated. Next, the number of independent tasks in each level is randomly decided. Finally, edges between levels are connected with the same probability *p* , as is *"sameprob"*.

- With six different numbers of processors varying from 2, 4, 8, 16, 32 and 64 processors. For each number of processors, six different DAG sizes have been generated varying from 10, 20, 40, 60, 80 and 100 tasks. In each experiment, the probability p and number of levels are assigned from the corresponding sets given below:
  - $SET_p=\{0.3, 0.5, 0.6, 0.7, 0.8, 0.9\}$
  - *L={No. Tasks/3, No. Tasks/4, No. Tasks/5, No. Tasks/6, No. Tasks/8}* according to number of tasks.

Performance improvement ratio has been calculated for each parameter; *schedule length*, *speedup* and *efficiency*.

5.2 Results

5.2.1 Schedule length

Schedule length is the maximum finish time of the exit task in the scheduled *DAG*. From Fig.5, 6, 7, 8, 9, it is noted that the schedule length decreases after applying *HCPT* algorithm, because the *HCPT* algorithm uses the average communication parents and the maximum child path with highest communication to compute priority of each task, which are the most important values for each task. The improvement ratio in schedule length is 16.5%. Table 3 shows, the schedule length of *CPOP, PHTS, ECTS,* and our algorithm *HCPT* of 20, 60,100 tasks at 8 and 32 processors, sample of experimental results.

5.2.2 Speedup:

Speedup of a schedule is defined as the ratio of the schedule length obtained by assigning all tasks to the fastest processor, to the schedule length of an application [4]. Equation (11) describes the speedup ratio.

$$\textbf{Speedup}=\frac{\underset{p_j \in P}{Min}\left[\sum_{n_i \in V} w(i,j)\right]}{SL} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \textbf{(11)}$$

Where $w(i,j)$ means the weight of task t_i on processor p_j and *SL* means the schedule length. Speedup is a good measure for the execution of an application program on a parallel system. The results of the comparative study according to the speedup parameter have been presented in Fig. 10, 11, 12, 13, 14, and 15. According to the results, it is clear that speedup of *HCPT* algorithm is better than speedup of the other algorithms, because all processors have finished tasks execution earlier than other algorithm, so our proposed algorithm outperforms the other algorithms in speedup parameter. The improvement ratio in speedup is 15.85%. Table 4 shows, the speedup of *CPOP, PHTS, ECTS,* and our algorithm *HCPT* of 20, 60,100 tasks at 8 and 32 processors, sample of experimental results.

5.2.3 Efficiency:

Efficiency as the speedup divided by the number of processors used [4] that is described in Equation (12).

$$\textbf{Efficiency}=\frac{\textbf{Speedup}}{\textbf{number of processors used}} \ldots\ldots\ldots\ldots\ldots \textbf{(12)}$$

The efficiency of the parallel computers is an indication to what percentage of a processors time is being spent in useful computation. From Fig. 16, 17, 18, 19, 20, 21. It is noted that, *HCPT* algorithm has better performance than the other algorithms. The improvement ratio in efficiency which has been achieved by *HCPT* algorithm is 16.4%. Table 5 shows, efficiency of *CPOP*, *PHTS*, *ECTS*, and our algorithm *HCPT* of 20, 60,100 tasks at 8 and 32 processors, sample of experimental results.

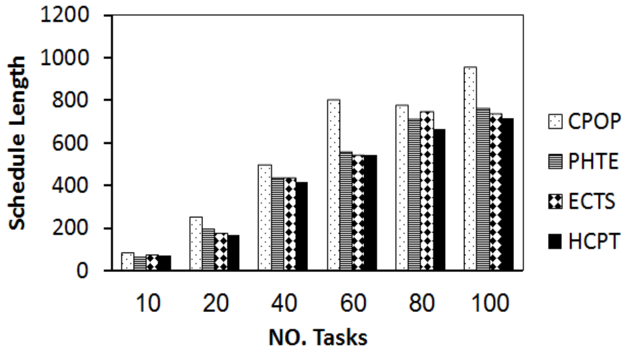


Fig 5: Schedule Length with 4 Processors.

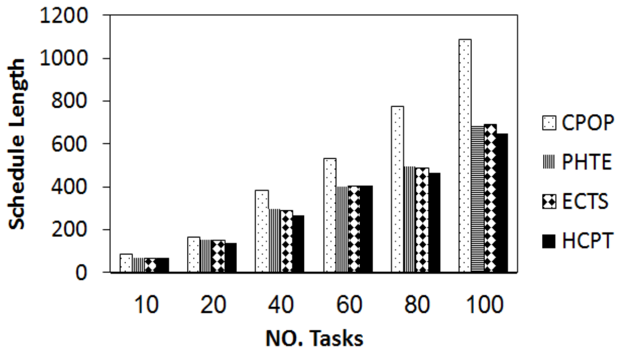


Fig. 6: Schedule Length with 8 Processors.

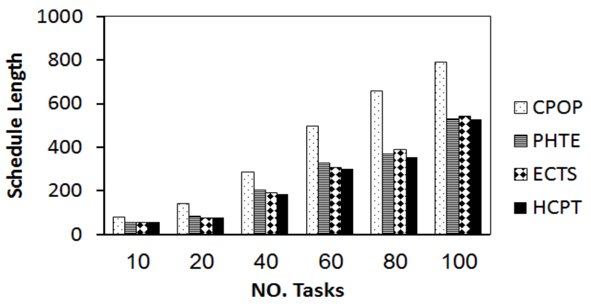


Fig. 7: Schedule Length with 16 Processors.

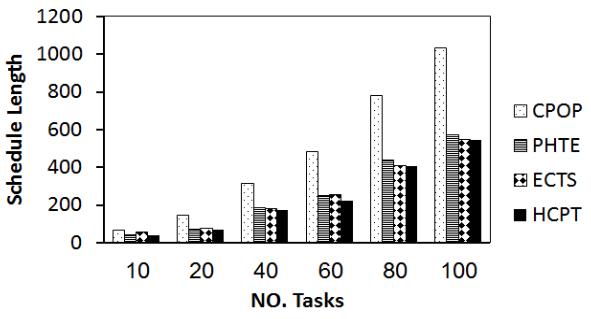


Fig. 8: Schedule Length with 32Processors.

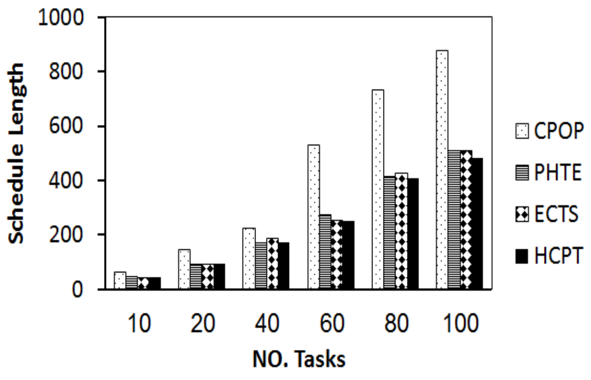


Fig. 9: Schedule Length with 64 Processors.

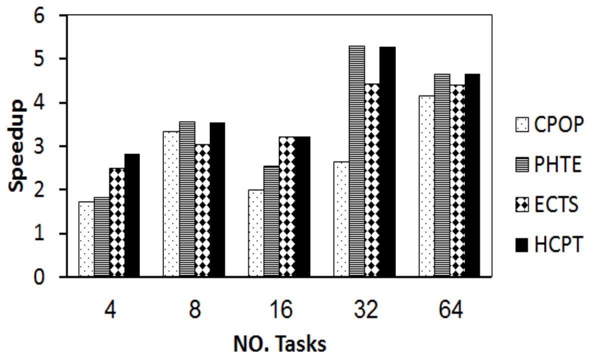


Fig. 10: Speedup with 10 Tasks.

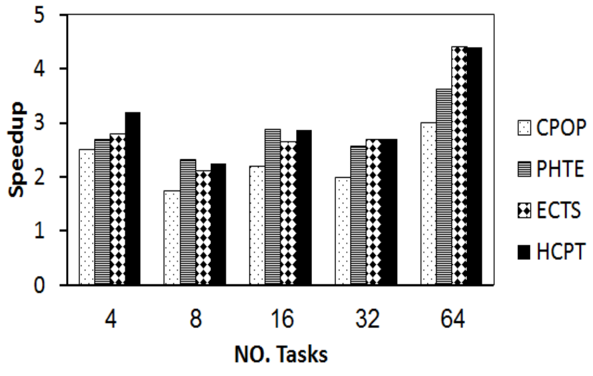


Fig. 11: Speedup with 20 Tasks.

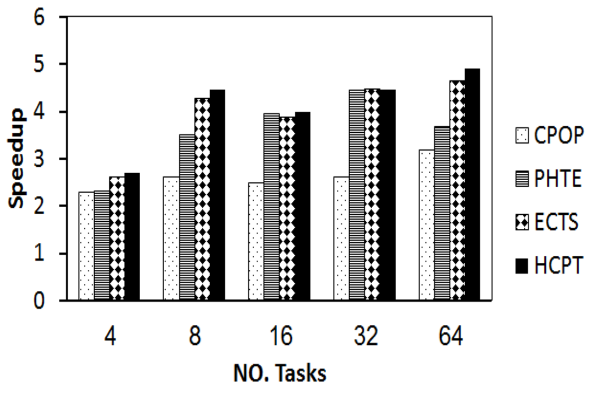


Fig. 12: Speedup with 40 Tasks.

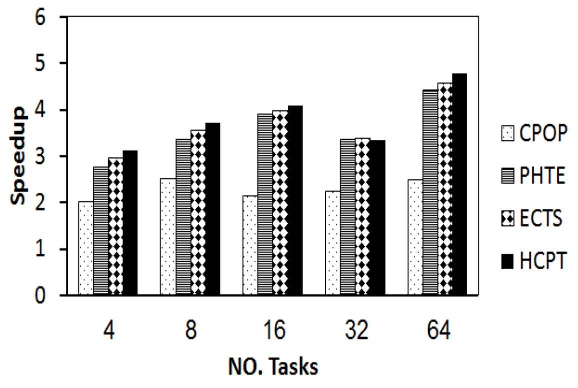


Fig. 13: Speedup with 60 Tasks.

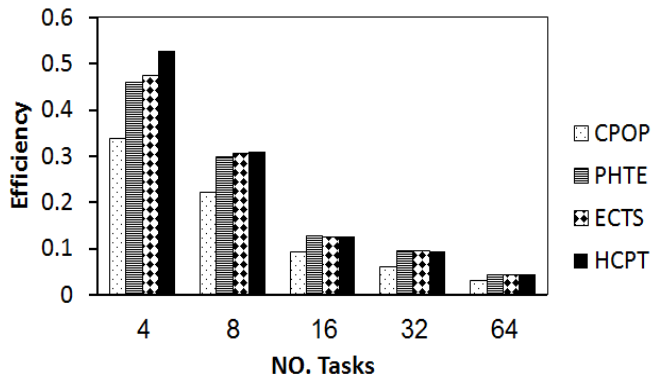


Fig. 17: Efficiency with 20 Tasks.

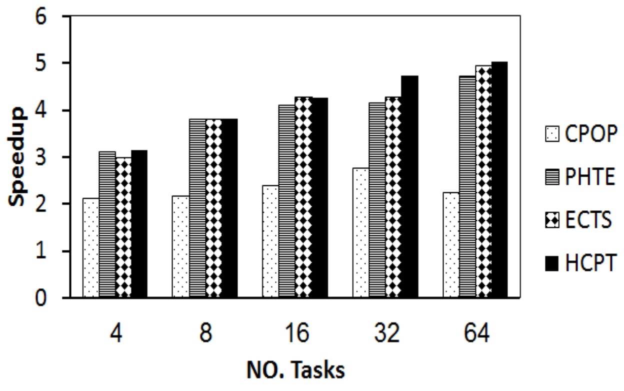


Fig. 14: Speedup with 80 Tasks.

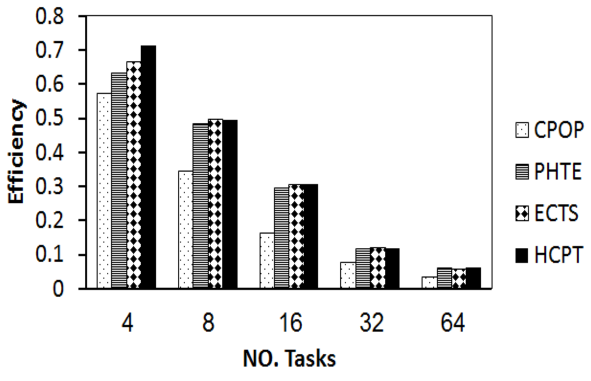


Fig. 18: Efficiency with 40 Tasks.

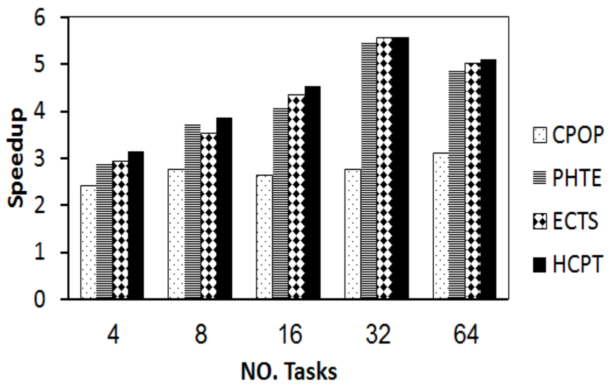


Fig. 15: Speedup with 100 Tasks.

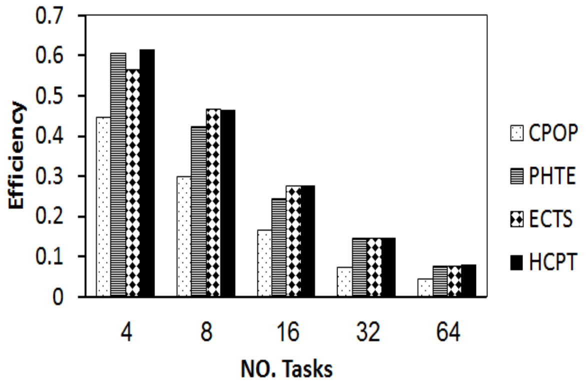


Fig. 19: Efficiency with 60 Tasks.

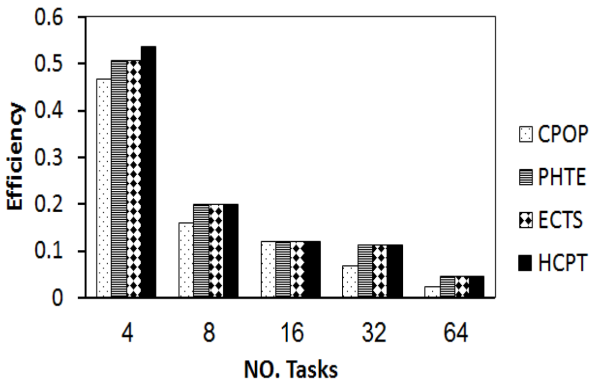


Fig. 16: Efficiency with 10 Tasks.

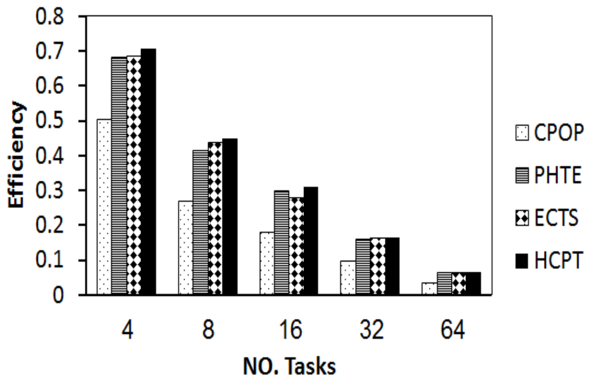


Fig. 20: Efficiency with 80 Tasks.

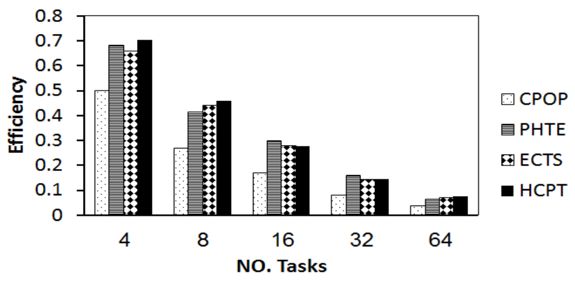


Fig. 21: Efficiency with 100 Tasks.

Table 3. Schedule Length of Algorithms (Result Samples)

NO. Processors	NO. Tasks	HCPT	ECTS	PHTS	CPOP
8	20	139	149	155	166
	60	404	405	402	533
	100	647	690	684	1087
NO. Processors	**NO. Tasks**	**HCPT**	**ECTS**	**PHTS**	**CPOP**
32	20	70	75	71	144
	60	224	256	251	484
	100	543	550	572	1036

Table 4. Speedup of Algorithms (Result Samples)

NO. Processors	NO. Tasks	CPOP	PHTS	ECTS	HCPT
8	20	2.120	1.739	2.323	2.352
	60	2.505	3.35	3.551	3.716
	100	2.762	3.729	3.535	3.860
NO. Processors	**NO. Tasks**	**HCPT**	**ECTS**	**PHTS**	**CPOP**
32	20	2	2.573	2.699	2.699
	60	2.233	3.355	3.333	3.347
	100	2.770	5.466	5.575	5.575

Table 5. Efficiency of Algorithms (Result Samples)

NO. Processors	NO. Tasks	CPOP	PHTS	ECTS	HCPT
8	20	0.223	0.3	0.306	0.309
	60	0.3	0.424	0.467	0.468
	100	0.268	0.414	0.443	0.461
NO. Processors	**NO. Tasks**	**HCPT**	**ECTS**	**PHTS**	**CPOP**
32	20	0.061	0.095	0.095	0.096
	60	0.075	0.146	0.146	0.147
	100	0.08	0.146	0.144	0.146

6. CONCLUSION

In this paper, a new **H**ighest **C**ommunicated **P**ath of **T**ask (*HCPT)* algorithm is presented for heterogeneous distributed computing systems (*HDCS*). This algorithm based on *Rank* value to give a priority to each task. According to the simulation results, it is found that the *HCPT* algorithm is better than *ECTS, PHTS* and*CPOP* algorithms in terms of schedule length, speedup and efficiency. Performance improvement ratio in schedule length, speedup and efficiency respectively are 16.5%, 15.85% and 16.4%. The *HCPT* algorithm can be tested on real applications and the development can be made on efficiency. Task duplication can be added also as a future work to increase the efficiency of the algorithm. The *HCPT* can apply on directed cyclic graph as a future work.

7. REFERENCES

[1] E. Hesham and A. Mostafa, "Advanced Computer Architecture and Parallel Processing", (Wiley Series on Parallel and Distributed Computing), Wiley-Inter-science ©2005, ISBN 0-471-46740-5.

[2] R. Prodan and M. Wieczorek, "Bi-criteriaScheduling of Scientific Grid Workflows," IEEE Trans. on Automation Science and Engineering, vol. 7, no. 2, pp. 364 –376.April 2010

[3] HuiCheng, "A High Efficient Task Scheduling Algorithm Based on Heterogeneous Multi-Core Processor", IEEE,Second International Workshop on Database Technology and Application (DBTA), Pages 1-4, 27-28 November 2010,doi: 10.1109/DBTA.2010.5659041.

[4] T. Hagras and J. Janecek, "A Near Lower-Bound Complexity Algorithm for Compile-Time Task-Scheduling in Heterogeneous Computing Systems", Third International Symposium on/Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks, and Third International Workshop onParallel and Distributed Computing, pp.106-113, 5-7 July 2004 doi: 10.1109/ISPDC.2004.3

[5] Junghwan Kim, Jungkyu Rho, Jeong-Ook Lee, and Myeong-Cheol Ko, "CPOC: Effective Static Task Scheduling for Grid Computing", in: Proceedings of the International Conference on High Performance Computing and Communications, Italy, 2005, pp. 477–486. DOI:10.1007/11557654_56

[6] E. Ilavarasan and P. Thambidurai, "Low Complexity Performance Effective Task Scheduling Algorithm for Heterogeneous Computing Environments," Journal of Computer Sciences, Vol. 3, No. 2, PP.94-103, 2007.

[7] H. Topcuoglu, S. Hariri, and M. Y. Wu, "Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing," IEEE Trans. Parallel and Distributed Systems (TPDS), Vol. 13, No.3, pp. 260-274, March 2002.

[8] I. Ahmad, and Y. Kwok, "A New Approach to Scheduling Parallel Programs Using Task Duplication", Proc. International Conference of Parallel Processing",1994, Vol.2, pp. 47-51, 15-19 august 1994, doi: 10.1109/ICPP.1994.37

[9] Tarek Hagras and Jan Janeˇcek, "A High Performance, Low Complexity Algorithm for Compile-Time Task Scheduling in Heterogeneous Systems",IEEE, 18th Internationalconference of Parallel and Distributed

Processing Symposium, 2004,pp.107-115, 26-30 April 2004, doi: 10.1109/IPDPS.2004.1303056

[10] Nirmeen A. Bahnasawy, Fatma Omara, and Magdy Qotb, "A New Algorithm for Static Task Scheduling for Heterogeneous Distributed Computing Systems", African Journal of Mathematics and Computer Science Research Vol. 4(6),pp. 221-234,June 2011.

[11] Mohammad I. Daoud, Nawwaf Kharma, "A High Performance Algorithm For Static Task Scheduling in Heterogeneous Distributed Computing Systems," Journal of Parallel and Distributed Computing, Volume 68, Issue 4, PP. 399-409, April 2008, doi:10.1016/j.jpdc.2007.05.015.

[12] R. Eswari and S. Nickolas, "Path-based Heuristic Task Scheduling Algorithm for Heterogeneous Distributed Computing Systems", International Conference on Advances in Recent Technologies in Communication and Computing, PP. 30-34, 16-17 October 2010, DOI:10.1109/ARTCom.2010.19

[13] R. Eswari and S. Nickolas, "Expected Completion Time Based Scheduling Algorithm for Heterogeneous Processors", in Proc. International Conf. Information Communication and Management(IPCSIT), vol.16, pp.72-77, January 2012,.

[14] V. A. F. Almeida, I. M. M Vasconcelos, J. N. C. Árabe and D. A. Menascé. "Using Random Task Graphs to Investigate the Potential Benefits of Heterogeneity in Parallel Systems", Proc. Supercomputing '92, pp. 683-691, 16-20 Nov. 1992. DOI:10.1109/SUPERC.1992.236634

[15] T. Yang and A. Gerasoulis, "DSC: Scheduling Parallel Tasks on an Unbounded Number of Processors", IEEE Transactionon Parallel and Distributed Systems, Vol.5, No.9, pp. 951-967, September 1994.