Bulletin of Electrical Engineering and Informatics
ISSN: 2302-9285
Vol. 5, No. 3, September 2016, pp. 373~382, DOI: 10.11591/eei.v5i3.559

A New Duplication Task Scheduling Algorithm in Heterogeneous Distributed Computing Systems

Aida A. Nasr*, Nirmeen A. EL-Bahnasawy, Ayman EL-Sayed
Department of Computer Science and Engineering, Faculty of Electronic Engineering, Menoufia University
Menouf 32952, Egypt
*Corresponding author, e-mail: aida.nasr2009@gmail.com

Abstract

The efficient scheduling algorithm is critical to achieve high performance in parallel and distributed systems. The main objective of task scheduling is to assign the tasks onto the available processors with the aim of producing minimum schedule length and without violating the precedence constraints. So we developed new algorithm called Mean Communication Node with Duplication MCND algorithm to achieve high performance task scheduling. The MCND algorithm has two phases namely, task priority and processor selection. Our algorithm takes into account the average of parents' communication costs for each task to reduce the overhead communication. The algorithm uses new task duplication algorithm. We build a simulation to compare the MCND algorithm with CPOP with duplication algorithm. The algorithms are applied on real application. From results, the MCND algorithm shows the best results.

Keywords*: Static task scheduling, List scheduling algorithm, Heterogeneous distributed systems*

1. Introduction

The availability of high performance networks leads to a new platform, called as heterogeneous distributed platform. Such a platform contains interconnected resources with different computing capabilities and different computing speeds. To run an application in this heterogeneous platform, several issues need to be taken into account such as partitioning the application, scheduling the tasks, etc. The performance of a parallel applications on Heterogeneous Distributed Computing Systems (*HeDCS*) critically depends on the method used to allocate the tasks partitioned from the application onto the appropriate processors in the system [1-4].

Boor task scheduling algorithm can undo any potential gains from the parallelism presented in the application, so selecting task scheduling algorithm is the important step of executing the parallel applications. In general, the objective of task scheduling is to minimize the execution time of a parallel application by properly assigning the tasks to the processors. Static and dynamic scheduling are the categories of scheduling models. In the static model, all information regarding the application and computing resources such as task weight, communication cost and data dependency is available a priori, so tasks scheduling is performed before the execution of the application. On the other hand; in the dynamic scheduling, scheduling is done at run-time. In this paper, we focus on static scheduling [5-7].

Static scheduling is classified into list-based, clustering and duplication based. List-scheduling basically consists of two phases: a task prioritizing phase and processor selection. In task prioritizing, the task priority is computed. In the processor selection phase, each task (in order of its priority) is assigned to processor that minimizes a suitable cost function. List-scheduling is generally accepted as an attractive approach since it characterized low complexity with good results [8-16]. Examples of list-based algorithms are **H**eterogeneous **E**arliest **F**inish **T**ime (*HEFT*) and **C**ritical **P**ath **O**n **P**rocessor (*CPOP*) [17]. Another static scheduling category is task duplication based algorithms, 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 and reduce the communication overheads. An example for duplication algorithms is CPOP with duplication [18].

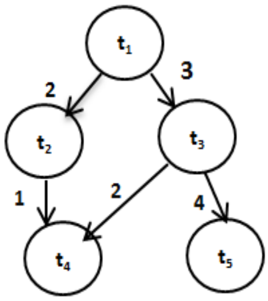
Received April 11, 2016; Revised July 17, 2016; Accepted July 29, 2016

The existing list-scheduling algorithms do not take into account the average communication of parents, data ready time and the maximum path of task. So we proposed new algorithm called Mean Communication Node with Duplication *MCND* algorithm. The *MCND* algorithm is developed for static task scheduling for the *HeDCS* with limited number of processors. It avoids the drawbacks of list scheduling algorithms and duplication algorithms. The objective of new *MCND* algorithm is to generate the quality task scheduling with low complexity. The developed algorithm uses the maximum path of task in calculating priority and uses duplication task to reduce the overhead of communication.

The remainder of this paper is organized as follows. Section 2 discusses task assignment problem. Section 3 gives an overview of CPOP algorithm with duplication. Section 4 presents developed *MCND* algorithm. Section 5 discusses the results and in section 6, conclusions are given.

2. Task Assignment Problem

Task assignment model consists of single application and target computing system. The application is divided into tasks represented by Directed Acyclic Graph *DAG, G=(V, E, P, W),* as shown in Figure 1.Where *V* is the set of v_i tasks, and *E* is the set of *e* edges between the tasks. Each *e(i,j) ϵ E* represents the precedence constraint such that task t_i(i.e. parent) should be executed before task t_j (i.e. child) can be started. The task with no parents is named root and the task with no children is named leaf. *P* is the set of *p* processors available in the heterogeneous 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. When two tasks are scheduled on the same processor the communication cost between these tasks can be negligible, because the speed of the inter-processor communication network is extremely low. All processors in the *HeDCS* are assumed to be fully connected. Communications between processors occur via independent units, so computation of tasks and communications between processors can be executed in parallel. Figure 1 shows an application with fife tasks. The application is represented as a *DAG* and the execution costs estimated for the five tasks on the *HeDCS* are shown as a computation cost matrix [19-21].



a. DAG

Task	P_0	P_1
t_1	7	8
t_2	4	9
t_3	12	8
t_4	2	3
t_5	4	6

b. Computation cost matrix

Figure 1. Example of the DAG and computation matrix

Definition (1) Critical Path (CP): CP of a DAG is the longest path from the entry task to the exit task in the graph.
Definition (2) EST(t_i, P_j) [10]: Denotes the Earliest Start Time of a task $t_i$on a processor $P_j$and is defined as shown in Equation (1).

$$EST(t_i, P_j)=max\{\ TAvailable(P_j)\ ,max\{AFT(t_k)+c_{k,i}\}\ \} \quad (1)$$

Where TAvailable (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 tk is the parent of task ti and k=1, 2, …, n) on the

processor $\mathrm{P_j}$. $\mathrm{c_{k,i}}$ is the communication cost from task $\mathrm{t_k}$ to task$\mathrm{t_i}$,$\mathrm{c_{k,i}}$ equal zero if the predecessor task $\mathrm{t_k}$is assigned to processor $\mathrm{P_j}$. For the entry task, EST($\mathrm{t_{entry}}$,$\mathrm{P_j}$)= 0.
Definition (3) EFT($\mathrm{t_i}$, $\mathrm{P_j}$) [10]: Denotes the Earliest Finish Time of a task $\mathrm{t_i}$on a processor $\mathrm{P_j}$and is defined as shown in the Equation (2).

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

Which is the Earliest Start Time of a task $\mathrm{t_i}$on a processor $\mathrm{P_j}$plus the computational cost $\mathrm{w_{i,j}}$ of $\mathrm{t_i}$on a processor$\mathrm{P_j}$.
Definition (4) Data Ready Time (DRT): is the idle time waited by a ti on processor pj.
Definition (5) Maximum Parent (MP): maximum parent of task ti is a parent task tk such that the value of EFT(tk ,pm) + c(tk,ti) is the largest among all ti's parent tasks.
Definition (6) Very Important Task (VIT): is the task that belongs to the critical path of DAG.

3. Critical Path on Processor with Duplication Algorithm

In this section, we give an overview of Critical Path On Processor (CPOP) algorithm with duplication as arelated work. CPOP with duplication consists of two phases: prioritizing phase and processor selection phase. In task prioritizing phase, the algorithm selects the task with the highest (upward rank + downward rank) value at each step. Upward rank is given in this equation (3).

$$Ranku=\overline{w_\iota} + max_{n_j\in succ(n_i)}(\overline{c_{\iota,J}} + rank_u(n_j)) \quad (3)$$

Where *succ(n_i)* is the set of immediate successors of task *n_i,* $\overline{c_{\iota,J}}$ is the average communication cost of *edge(i,j),* and $\overline{w_\iota}$ is the average computation cost of task *n_i.* The downward rank is computed by using the equation (4).

$$Rankd(ni) =max_{n_j\in pred(n_i)}(\overline{c_{J,\iota}} + \overline{w_J} + rank_d(n_j)) \quad (4)$$

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 algorithm applys task duplication condition to select the processor.

4. The Developed Algorithm

The Mean Communication Node with Duplication MCND algorithm is list-based scheduling algorithm. It uses the main idea of HCPT algorithm with some edits and adding new duplication algorithm. It consists of two phases only, task priority and processor selection. The MCND algorithm removes level sorting phase from HCPT algorithm to reduce executing time of algorithm. The detailed explanation of each phase of the algorithm is described in the following subsections.

4.1. Task Priority Phase

In this phase, the MCND algorithm assign a priority for each task using rank attribute that is obtained from the equation (5).

$$Rank(ti)= MCP(ti)+max_{t_j\in succ(t_i)}\left(C_{i,j} + Rank(t_j)\right) \quad (5)$$

Where *MCP*(*t_i*) refers to Mean Communication of Parents. It is computed by the equation (6).

$$MCP(ti)=(\textstyle\sum_{k=1}^{n} C_{k,i})/x \quad (6)$$

Where x is the number of Parents, $C_{k,i}$ is the communication between parent t_k and task t_i. For the exit task t_{exit} the Rank value is equal *MCP(t_{exit})*. Tasks List TL is generated by sorting the tasks by decreasing order of *Rank* value. Figure 2 shows MCND algorithm's pseudu code.

```
Set the computation cost of tasks and the communication cost of edges.
Compute Rank for all tasks starting from the exit task by the next equation.
$Rank(t_i)= MCP(t_i)+max_{t_j \in succ(t_i)} \left(C_{i,j} + Rank(t_j)\right)$
Sort the tasks in Tasks List TL in decreasing order of Rank values.
For each task $t_i$ in TL
For each processor $P_j$ in the processor set ($P_j$ є Q) do
      Compute EFT($t_i$, $P_j$) value
   End for
Assign task $t_i$ to the processor $p_j$ that minimizes EFT
  If $t_i$ is VIT
{
  If  DRT ( $t_i$ , $p_j$)>w (MP,$p_j$)
   If EST($t_i$ ,$p_j$)>EFT(MP, $p_j$)
     {
       Duplicate MP  on $P_j$ without violating the dependency constraints
      Update EFT of $t_i$ on $p_j$
     }
}
End for
```

Figure 2. The MCND algorithm steps

4.2. Processor Selection Phase

This phase consists of two stages: processor stage and duplication test stage. In processor stage, *MCND* algorithm selects task t_i from TL. If t_i has no parents or all parents are scheduled, the 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. With high performance algorithms, some processors are idle during the execution of the application because of DRT. If DRT is enough to duplicate MP, the execution time of the parallel application could be reduced. So, the algorithm applies task duplication to reduce the makespan. The algorithm tests, if DRT of task t_i is more than the weight of MP on the same processor p_j, the algorithm duplicates the MP on pj and updates EFT of task t_i. The duplication stage is applied on VIT only. This must be done without violating the precedence constrains among tasks.

5. Results and Discussins

5.1. Comparison Metrics

The comparison metrics are schedule length ratio, the average of speedup and the average running time.

5.1.1. Schedule Length Ratio (SLR)

SLR value is defined by the equation (7)

$$SLR= SL / \sum_{ti \in CP_{min}} min_{pj \in Q} \{wi,j\}. \quad (7)$$

Where SL is the schedule length. The divisor is the summation of the minimum computation costs of tasks on CPmin. (For an unscheduled DAG, if the computation cost of each task ti is set with the minimum value, then the critical path will be based on minimum value, then the critical path will be based on minimum computation cost, which is represented as CPmin) [17]. The SLR can never be less than one, since the divisor is the lower bound. Algorithm that gives smallest SLR of a graph, is the best algorithm with respect to performance.

5.1.2. The Average of 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 parallel application.

$$Speedup = \frac{\underset{pj\epsilon P}{Min}[\sum_{ni\epsilon V} w(i,j)]}{SL} \quad (8)$$

Where $w(i,j)$the weight of task ti on processor pj and SL is is the schedule length.

Speedup is a good measure for the execution of an application on a distributed system. Due to minimize schedule length, all processors have finished tasks execution earlier and speedup of MCP algorithm increases. The results of the comparative study according to the speedup parameter have been presented in Figure 6. According to the results, performance ratio of speedup is calculated as 15%.

5.1.3. Average Running Time (ART)

ART is the average running time of different DAGs. The algorithm with smallest average running time is the best algorithm.

5.2. Simulation Environment

We use tow types of graphs for testing MCND algorithm and the related work: randomly generated graphs and graphs represented the real problems.

5.2.1. Random Graph Generator

For building random DAGs the program requires the following input parameters.

- **N** is the number of DAG tasks, where N {20,40,60,80,100,120}.
- **α (parallelism)**is the shape parameter of DAG. Like [18] we assume that height of a DAG is $\sqrt{N}/\alpha$. And the width of each level is randoml selected from a uniform distribution with mean value to $\sqrt{N}$*α. Where α in {0.5,1,2}.
- **Out_Deg** is the out_Degree. It is the maximum number of task successors.
- **D** is the DAG density. The density of DAG determines the number of dependencies between nodes. D € [0.3,0.8]. There is an edge between ti in level L and tj in level L+Z , if random value ∈ [0.1,1] ≤ D and number of successors ≤ out_Deg of t_i.
- **WDAG** is the average computation cost of given graph. This selected randomly from a predefined set [WDAG/4, 2*WDAG]. WDAG ∈ {50,70,100,150,200}.
- **CCR** is Communication to Computation Ratio. It is the average edge weight divided by the average node weight. CCR ∈ {0.1, 0.5, 1, 5, 10}.
- **Percentage_Ratio** is the heteroginity factor for processors speeds. When this ratio is high, this mean that difference between task's computations on different processors is very high and vice versa when the percentage ratio is low, the difference is low.

We take the average results for each DAG size at p ∈ {2,4,8,16,32,64}. Figures 3, 4, 5 show the results of random DAGs. Figure 3 shows the average SLR with respect to various number of tasks. It is noted that, SLR of MCND algorithm is smaller than CPOP with duplication. The MCND algorithm uses the most important attributes of the task (Mean Communication of Parents to expect DRT(ti) and the maximum path from that task to exit task) to calculate the priority for each task. According this attributes the algorithm sorts the tasks. The algorithm uses also task duplication to reduce DRT of the task successors, so it can reduce the overall time of application. For this reasons, the MCND algorithm is more efficient than CPOP with duplication algorithm. Speedup is a good measure for the execution of an application on a distributed system. Due to minimize schedule length, all processors have finished tasks execution earlier and speedup of MCND algorithm increases. The results of the comparative study according to the average speedup parameter have been presented in Figure 4. We observe that MCND algorithm is faster than the CPOP algorithm, because it applies task duplication on VIT only to reduce the communication overhead. The new algorithm takes small amount of time for execution; this is shown in Figure 5. Because the CPOP with duplication algorithm apply task duplication for each task, it takes more time for execution. We take some of snapshots from our simulation to show the effect of MCND algorithm.

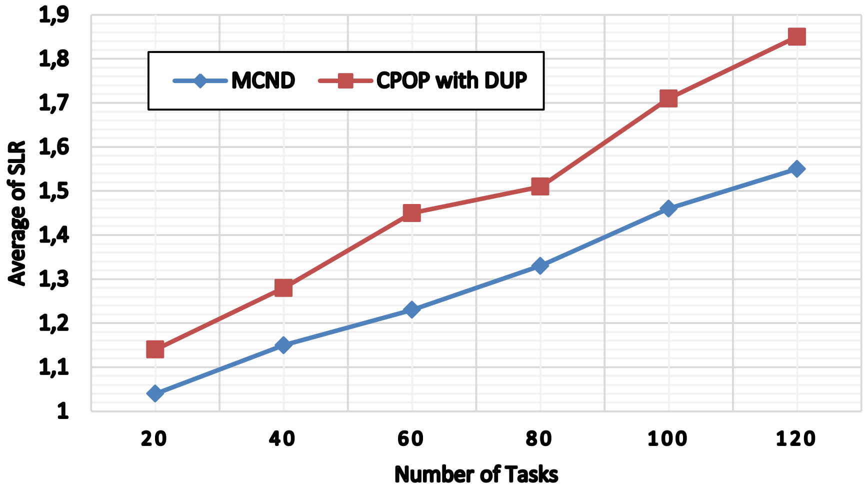


Figure 3. The average SLR with respect of DAG size

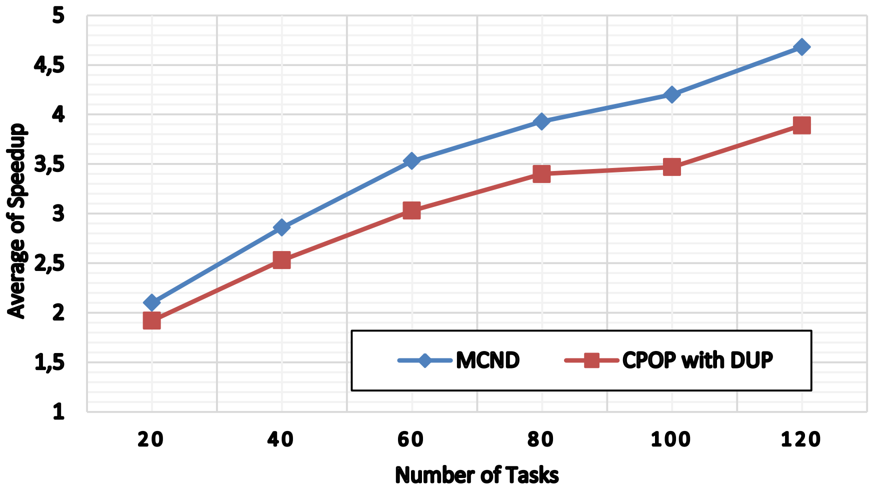


Figure 4. The average speedup with respect of DAG size

Figures 6, 7, and 8 show the Snapshots from the simulation program. We defined the parameters of DAG generation on section 5.2.1. random graph generator. From this snapshots, we observe that the MCND algorithm is more efficient with fine-grain graphs at (CCR≤1). The Percentage Ration parameter also has an effect on the result. With percentage ration less than or 0.75 the performance of MCND algorithm is very high. With Other parameters the MCND algorithm has high performance compared with the CPOP with duplication algorithm.

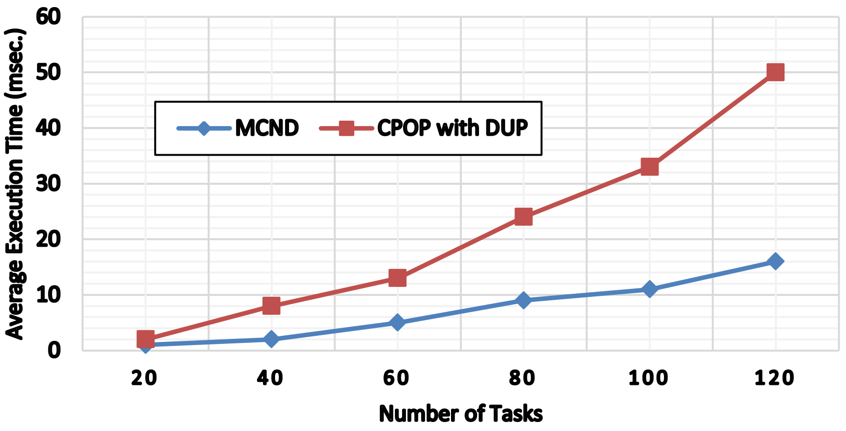
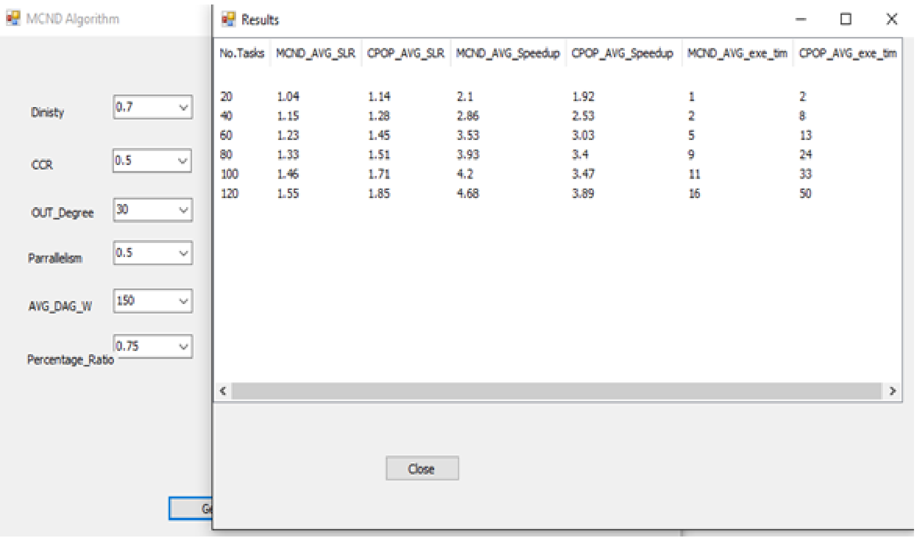


Figure 5. The average execution time with respect of DAG size



Dinisty 0.7

CCR 0.5

OUT_Degree 30

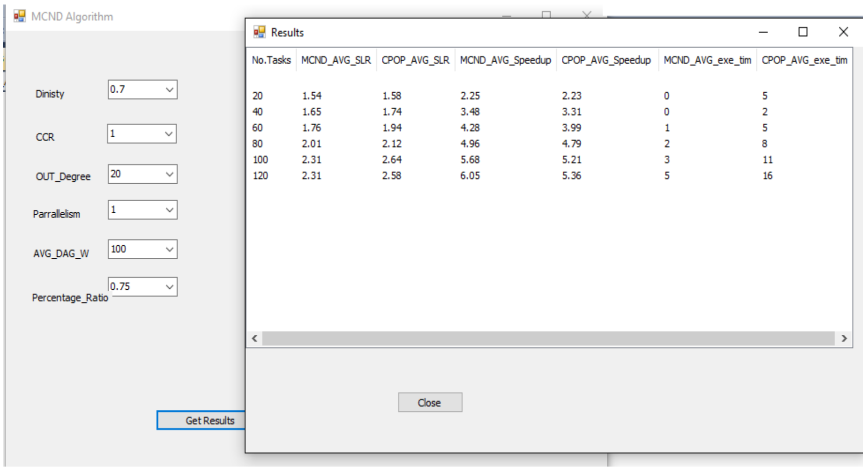
Parrallelism 0.5

AVG_DAG_W 150

Percentage_Ratio 0.75

No.Tasks	MCND_AVG_SLR	CPOP_AVG_SLR	MCND_AVG_Speedup	CPOP_AVG_Speedup	MCND_AVG_exe_tim	CPOP_AVG_exe_tim
20	1.04	1.14	2.1	1.92	1	2
40	1.15	1.28	2.86	2.53	2	8
60	1.23	1.45	3.53	3.03	5	13
80	1.33	1.51	3.93	3.4	9	24
100	1.46	1.71	4.2	3.47	11	33
120	1.55	1.85	4.68	3.89	16	50

Figure 6. Snapshot (1) from the simulation programes



Dinisty 0.7

CCR 1

OUT_Degree 20

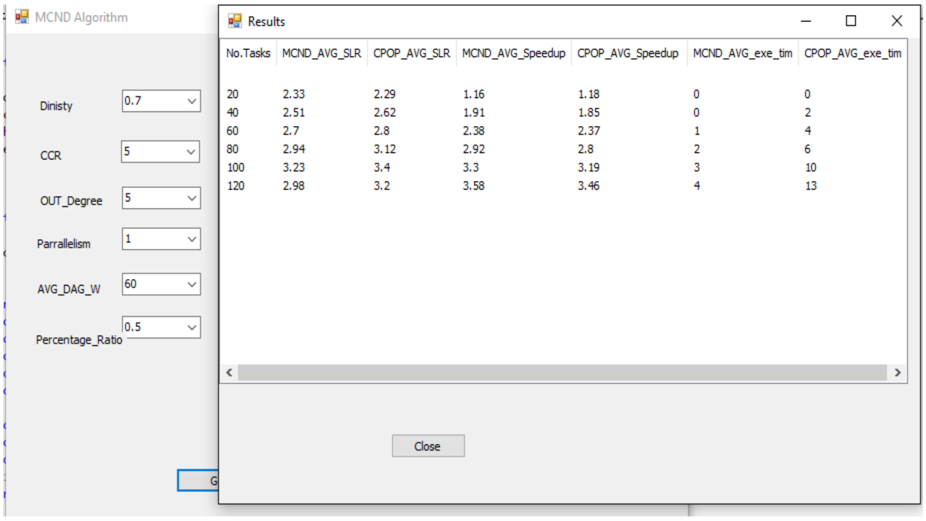
Parrallelism 1

AVG_DAG_W 100

Percentage_Ratio 0.75

No.Tasks	MCND_AVG_SLR	CPOP_AVG_SLR	MCND_AVG_Speedup	CPOP_AVG_Speedup	MCND_AVG_exe_tim	CPOP_AVG_exe_tim
20	1.54	1.58	2.25	2.23	0	5
40	1.65	1.74	3.48	3.31	0	2
60	1.76	1.94	4.28	3.99	1	5
80	2.01	2.12	4.96	4.79	2	8
100	2.31	2.64	5.68	5.21	3	11
120	2.31	2.58	6.05	5.36	5	16

Figure 7. Snapshot (2) from the simulation programes



Dinisty: 0.7
CCR: 5
OUT_Degree: 5
Parrallelism: 1
AVG_DAG_W: 60
Percentage_Ratio: 0.5

No.Tasks	MCND_AVG_SLR	CPOP_AVG_SLR	MCND_AVG_Speedup	CPOP_AVG_Speedup	MCND_AVG_exe_tim	CPOP_AVG_exe_tim
20	2.33	2.29	1.16	1.18	0	0
40	2.51	2.62	1.91	1.85	0	2
60	2.7	2.8	2.38	2.37	1	4
80	2.94	3.12	2.92	2.8	2	6
100	3.23	3.4	3.3	3.19	3	10
120	2.98	3.2	3.58	3.46	4	13

Figure 8. Snapshot (3) from the simulation programes

5.2.2. The Real Applications

The MCND algorithm applies also on application DAGs of real problems like Gaussian elimination and Fast Fourier Transformation. Figures 9, 10 show Gaussian elimination and FFT graphs. The schedule length after applying the MCND algorithm and CPOP with duplication algorithm on Gaussian DAG is 636 and 690 respectively at 4 processors. The two algorithms apply also on FFT DAG and the schedule length of MCND algorithm is 452 and the schedule length after applying CPOP with duplication algorithm is 542 at 4 processors.

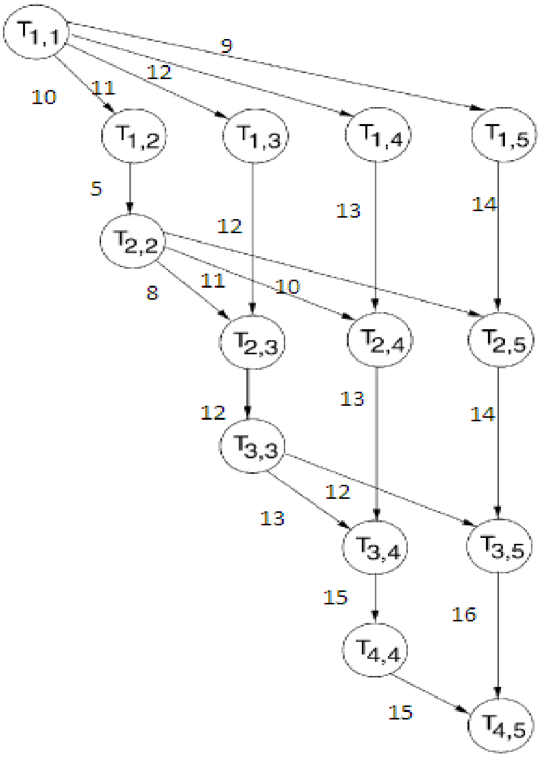


Figure 9. Gaussian elimination graph

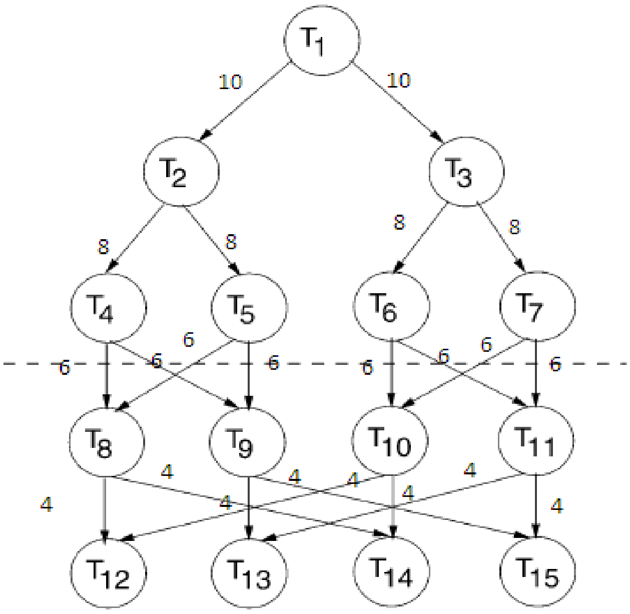


Figure 10. FFT graph

6. Conclusion

In this paper, a new task duplication scheduling algorithm has been presented for heterogeneous distributed computing systems (HDCS) to enhancement scheduling performance. This algorithm uses new attribute called Rank to assign a priority for each task. It also uses task duplication technique to decrease the communication overhead. The performance analysis showed that the proposed MCND algorithm has better performance than CPOP with duplication algorithm. According to the simulation results, it is found that the MCND algorithm is better than the other algorithm in terms of SLR, speedup and execution time. The new algorithm applies new task duplication algorithm to reduce the schedule length of DAG. It applies the task duplication on VIT only not on every task in DAG. So, it takes low execution time to schedule the tasks.

References

[1] R Prodan, M Wieczorek. “Bi-criteria Scheduling of Scientific Grid Workflows”. *IEEE Trans. on Automation Sci. and Eng*. 2010; 7(2): 364 –376.

[2] H Izakian, A Abraham, V Snasel. “Comparison of Heuristics for Scheduling Independent Tasks on Heterogeneous Distributed Environments”. in Proc. of the *International Joint Conference on Computational Sciences and Optimization*, IEEE. 2009; 1: 8-12.

[3] S Hong, C Shi-ping, J Chen, G Kai. “Research and Simulation of Task Scheduling Algorithm in Cloud Computing”. *TELKOMNIKA*., e-ISSN: 2087-278X. 2013; 11(11): 6664~6672.

[4] L Hai-Xiang. "Communication Scheduling in Parallel Task Executions on Large Parallel Systems". *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2012; 10(5): 1151~1156.

[5] C Hui. "A High Efficient Task Scheduling Algorithm Based on Heterogeneous Multi-Core Processor". *IEEE, Database Technology and Application (DBTA)*. 2010: 1-4.

[6] R Eswari, S Nickolas. “Expected completion time based scheduling algorithm for heterogeneous processors”. in Proc. *International Conf. Information Communication and Management, IPCSIT*. 2011; 16: 72-77.

[7] Sang Cheol Kim, Sunggu Lee, Jaegyoon Hahm. “Push-Pull: Deterministic Search-Based DAG Scheduling for Heterogeneous Cluster Systems”. *IEEE Transactions on Parallel and Distributed Systems*. 2007; 18: 1489-1502.

[8] Yang H, Lee P, Chung C. “Improving Static Task Scheduling in Heterogeneous and Homogeneous Computing Systems”. *IEEE International Conference on Parallel Processing, ICPP*. 2007: 45-45.

[9] Thambidurai P, Mahilmannan R. “Performance Effective Task Scheduling Algorithm for Heterogeneous Computing System”. IEEE Proceedings of the *4th International Symposium on Parallel and Distributed Computing*. 2005: 28-39.

[10] R Bajaj, DP Agrawal. “Improving Scheduling of Tasks in a Heterogeneous Environment”. *IEEE Transactions Parallel Distributed System*. 2004; 15: 107–118.

[11] M Ehsan, M Sajjad, H Altaf, N Muhammad, A Shoukat. "SDBATS: A Novel Algorithm for Task Scheduling in Heterogeneous Computing Systems". *IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum*. 2013: 43-53.

[12] H Arabnejad, J Barbosa. “List Scheduling Algorithm for Heterogeneous Systems by an Optimistic Cost Table”. *IEEE Transactions on Parallel and Distributed Systems*. 2013; 99: 1–1.

[13] A Saima, M Ehsan, N Wasif. “A Segmented Approach for DAG Scheduling in Heterogeneous Environment”. in 12th *International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT). IEEE*. 2011: 362–367.

[14] N Wahid, N Wafa. “A New DAG Scheduling Algorithm for Heterogeneous Platforms”. in 2nd *IEEE International Conference on Parallel Distributed and Grid Computing (PDGC). IEEE*. 2012: 114–119.

[15] B Luiz, S Rizos, M Edmundo. “DAG Scheduling Using a Lookahead Variant of the Heterogeneous Earliest Finish Time Algorithm”. in 18th *International Conference on Parallel, Distributed and Network-Based Processing (PDP). IEEE.* 2010: 27–34.

[16] Bajaj R, Agrawal DP. "Improving Scheduling in a Heterogeneous Environment". *IEEE Trans. on Parallel and Distributed systems*. 2004; 15(2): 107-118.

[17] H Topcuoglu, S Hariri, MY Wu. “Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing”. *IEEE Trans. Parallel and Distributed Systems*. 2002; 13(3): 260-274.

[18] RS Singh, AK Tripathi, S Saurabh, V Singh. Duplication based List Scheduling in Heterogeneous Distributed Computing, National Conference on Advancement of Technologies – Information Systems & Computer Networks (ISCON – 2012). Proceedings published in *International Journal of Computer Applications® (IJCA)*.

[19] R Bajaj, DP Agrawal. "Improving Schedule of Tasks in a Heterogeneous Environment". *IEEE Transactions on Parallel and Distributed Systems.* 2004; 15(2): 107-118.

[20] Illavarasan E, Thambidurai P. “Performance Effective Task Scheduling Algorithm for Heterogeneous Computing System". in Proc. of the *4th International Symposium on Parallel and Distributed Computing, ISPDC’05, France, IEEE*. 2005.

[21] Donfack S, Grigori L, Gropp WD, Kale V. "Hybrid Static/dynamic Scheduling for Already Optimization Dense Matrix Factorization". Published in: *Parallel & Distributed Processing Symposium (IPDPS), IEEE 26th International*. 2012: 496 – 507.

[22] E Illavarasan, P Thambidurai. "Low Complexity Performance Effective Task Scheduling Algorithm for Heterogeneous Computing Environments". *J. of Computer Sci.[Online]*. 2007; 3(2): 94-103. Available: http://thescipub.com/issue-jcs/3/2.