Research Journal of Applied Sciences, Engineering and Technology 10(6): 663-671, 2015
DOI:10.19026/rjaset.10.2475
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: January 8, 2015 Accepted: February 13, 2015 Published: June 20, 2015

Research Article

Effective Scheduling Based on Task Duplication for Scheduling Parallel Applications in Grid Environment

[1]D.I. George Amalarethinam and [2]P. Muthulakshmi
[1]Department of CS, Jamal Mohamed College, Trichy,
[2]Department of Computer Science, SRM University, Chennai, Tamil Nadu, India

Abstract: This study addresses a duplication based scheduling algorithm called Effective Scheduling based on Task Duplication (ESTD) for grid computing environment. Duplications are made based on task dependencies. The algorithm ensures beneficial duplications and avoids unnecessary duplications. Idle time slots between task execution times are effectively used. The algorithm aims to avoid the communication contention, which will happen when there is frequent transportation of large sets of data. The performance of the algorithm is scaled by comparing it with the algorithms of its kind. The results show minimized make span and effective resource utilization with balanced loads across resources in grid.

Keywords: Active scheduling, duplication mechanism, forward directive, include directive, load balancing, passive scheduling, work flow

INTRODUCTION

Grid computing distinguishes itself from other parallel and distributed systems through its unique features which may include heterogeneity of computing resources and their dynamic participation, varied administrative domains of resources and networks, dynamic accessibility of resources by the users and so on. High speed networks and resources, low cost super computing power and high end technologies have taken the grid computing technology to be realized as the next generation computing solution. Generally, grids are formed out of dedicated resources that form a parallel computer; also use the computing power of personally owned computers that are available on the internet. Later category of resources let their computing cycles to be used whenever they are connected over internet and also necessarily their CPU cycles are found idle. These resources are identified as non dedicated resources.

Grid computing emphasizes on proper resource utilization and shortest makespan in executing the tasks. In such environments, the resources cannot be kept idle as it could lead to maximum completion time of tasks. Scheduling is the process of assigning and executing tasks on available resources. An optimized scheduling algorithm can improve the resource utilization through proper selection of resources for tasks to execute. Grid schedulers map the tasks on to the available resources in a competent mode. The dedicated grid computing resources can run for a long time and the grid schedulers efficiently allocate tasks to these resources by using the scheduling algorithms to complete task execution in shorter makespan. Makespan is the difference between the start time and finish time of a sequence of tasks. A proper Resource Management System (RMS) ensures the efficiency of any computing system as it is closely associated with scheduling. RMS sets the permissions for grid users to use the grid resources and tracks the resource utilization, which would be used for billing purpose The Grid Resource Broker System (GRBS) is an interface between grid service providers and grid users; it helps to use the available grid resources. The resources are owned by service providers and are traded for their CPU time. Sometimes the resource itself may be identified as service provider. The grid users use the resources to execute their applications in a parallel fashion; therefore the overall execution time of the application is reduced. An application is a group of tasks which may have interdependencies among them.

A task is the smallest unit of work to be executed on a computing resource. A sequential or random procedure to be executed is known as workflow. The tasks and their dependencies are shown through the workflow (Prodan and Marek, 2010). Workflow includes scheduling, submitting data, transferring contents for execution across resources, fault tolerance management and so on. Among the above said, scheduling is a significant component as it does resource discovery, resource filtering, resource selection, mapping resources and tasks. A best effort work flow scheduling concentrates on minimum execution time.

Corresponding Author: P. Muthulakshmi, Department of Computer Science, SRM University, Chennai, Tamil Nadu, India

The deterministic workflows are parallel applications whose inter dependencies are known in advance. These applications can be represented through task graph or Directed Acyclic Graph (DAG). DAG is a collection of set of nodes and set of edges. A node represents a task, which is a set of codes to be executed and hence it has a need for computational power known as weight. The dependencies are followed through edges that connect a pair of nodes. An edge helps in fulfilling the communication need. A task system contains a set of tasks with a precedence relation determined only by data dependencies (Kai and Faye, 1985). The precedence constraint is said to be obeyed when the dependent task (successor/child) starts its execution only after receiving the necessary information from all its predecessors (parents). The dependency can also be represented as parent-child relationship.

In static scheduling, advanced reservation of resources is encouraged and it requires a maximum limit of reserving resources and is (at most) the maximum number of tasks that a level contains in DAG. In DAG, tasks are distributed in levels. In order to show this pictorially, DAG can have the parents in the top level and children are placed in the next successive levels towards the bottom. The child is the dependent of parent(s). There cannot be dependencies between tasks of same level. A child can have parents in any of the levels previous to the level where it is present.

The method of duplication involves the execution of particular tasks in more than one computing resources and thereby making the results locally available for more descendants. Therefore the cost of message passing becomes zero and also the communication between the resources is overlooked. The idle time of a resource can be used to execute a replicated task. Generally, it is observed from researches, that duplication mechanism results in earlier completion time of DAG and avoids communication contention between resources. As the idle slots are used effectively, wait time is avoided and the resource performance is highly appreciated.

Many algorithms have proven themselves for producing quality results in homogeneous computing systems. However, it is a complicated problem to schedule tasks on to resources in heterogeneous environment due to varying nature of resources, speeds and network bandwidths. List scheduling is a proven solution for heterogeneous environments as they result in quality schedules with low complexity. Heterogeneous Earliest Finish Time (HEFT) (Topcuoglu *et al*., 2002) algorithm is found to be most popular DAG scheduling algorithm that schedules tasks on resources with earliest finish time and it uses insertion policy, but task duplication is not supported.

In this study, we propose a novel task duplication algorithm for grid computing environment called Effective Scheduling based on Task Duplication (ESTD). Several duplication algorithms have been proposed and were found to allow unnecessary duplications. The algorithm uses greedy technique to find and reserve the resources. The algorithm does not encourage prolonged idle time of the resources. The support of more resources may be required at the initial stage of the workflow. Resources mapped for a particular application to execute certain initial level tasks need not wait until all the tasks to get over. Hence the resources are encouraged to relinquish themselves after executing the specified tasks. This helps the GRB to make use of the resources effectively for other applications. The algorithm can prove its efficiency in distributed environments by minimizing the makespan. Generally, duplication heuristics are very effective on tasks graphs of high value communication links and such graphs may result in high Communication Computation Ratio (CCR) value. CCR is defined as the ratio between the average communication cost and average computation cost on a computing system.

LITERATURE REVIEW

Task duplication is the heuristic based static task scheduling technique. The problem of minimizing the completion time of task graph has been studied by various researches and is found that most of the algorithms are addressing homogeneous resources. Task duplication is adopted in algorithms proposed in papers (Sandnes and Meson, 2001; Li *et al*., 2003; George Amalarethinam and Malai Selvi, 2012; Ahmad and Yu-Kwong, 1994; Menglan and Bharadwaj, 2012; Oliver *et al*., 2011; Chan-Ik and Tae-Young, 2001; Koichi *et al*., 2006; Amit and Padam, 2010; Savina *et al*., 2005; Ranaweera and Agarwal, 2000; Bozdag *et al*., 2006).

Task duplication is to avoid/reduce the inter process communication cost between the resources (Ahmad and Yu-Kwong, 1994). In study (Menglan and Bharadwaj, 2012), the authors proposed task duplication algorithm called Prudent Algorithm with Replication (PAR), which capture the processing requirements for applications, based on that the applications are executed in very certain applicable resources. Task duplication based scheduling algorithm presented in Oliver *et al*. (2011) uses clusters, uniform communication cost and duplication is applied until the last task in the graph is executed. Another Task Duplication Algorithm referred in Chan-Ik and Tae-Young (2001) uses less restricted optimality condition and is found that many parent tasks of a child task to be extended in the same resources.

Also, there are task duplication based scheduling algorithms that discuss on:

- Avoiding useless duplication (Koichi *et al*., 2006)
- Avoiding the increase in scheduling cost due to the duplicated tasks overhead (Amit and Padam, 2010)
- Limited number of duplication (Savina *et al*., 2005)

Task duplication-based scheduling Algorithm for Network of Heterogeneous system (TANH) decides the schedule based on availability of resources in comparison with resources actually needed (Ranaweera and Agarwal, 2000).

It is observed that load balancing among the resources is not considered by the algorithms stated above. In the proposed algorithm, we employed the task duplication mechanism and we concentrated on load balance across resources. We made the comparative study on schedule length and speedup of the proposed algorithm with Triplet Bin Task Grouping and Prioritizing (TBTGP) (George Amalarethinam and Muthulakshmi, 2014), Economical Duplication Scheduling in Grid (EDS-G) (Amit and Padam, 2010), Duplication and Insertion Based Scheduling (DIBS) (Lijun *et al*., 2013).

METHODOLOGY

Research problem and description: In the study it is found that most algorithms use homogeneous computing resources and uniform communication cost and is not the actual scenario of the grid computing environment. It is observed that some algorithms support needless executions, resource requirement is not examined. The superfluous reservation of resources may also result in inefficiency as the resources remain unused for most of the time. To overcome the issues, we proposed an algorithm called ESTD algorithm. ESTD algorithm uses a limited number of resources. It is a static scheduling algorithm follows greedy method. Decision making in mapping a task and resource results in optimal solution. However the algorithms may result in suboptimal solution in very rare cases. It uses the list scheduling and cluster scheduling techniques. The tasks of a particular level is sorted and grouped with respect to their dependent counts. The most prioritized task will be allowed for duplication. The algorithm could finalize the busy time of a resource and it not necessary for resources to wait until the last resource to finish its execution.

Therefore, if a resource finds itself not getting any more tasks for execution after executing some tasks in the initial level then that resource may be relieved from the current application and can be made available for other task graphs. The proposed algorithm allows duplication of tasks up to certain level of the DAG. The algorithm could decide which task to be duplicated from a particular level. Also it fixes the number of duplications to be made by a selected task.

ESTD is a heterogeneous algorithm and it completely avoids needless duplication of tasks. The algorithm is devised to evade unnecessary communication time utilized in passing messages between data interdependent tasks. Therefore overall execution time of the application is considerably reduced. Task duplication is done based on the rank value of each task. The rank value is evaluated based on number of dependents, level in which the dependents are present, communication cost, computation cost. We study the impact of duplications on makespan and the algorithm is optimized to accept only the beneficial duplications. Cost and time are interchangeably used in this study.

Mathematical elements of ESTD algorithm:

Approximate computation cost of a task on each processor: Approximate COMPutation Cost (ACOMP Cost) is computed for each task with respect to speed of each resource is given by:

for i = 1 to last task do
for j = 1 to last processor do

$$ApCOMPCost(t_i, r_j) = COMPCost(t_i) / speed(r_j) \quad (1)$$

Preferred resource of each task: The resource is preferred for its minimum execution time to execute a particular task:

for i = 1 to last task do
for j = 1 to last resource do:

$$PrefRes(t_i) = \min(ApCOMPCost(t_i, r_j)) \quad (2)$$

Average computation cost of a task on available resources: Average COMPutation Cost AvCOMPCost) of each task is determined as follows:

for i = 1 to last task do

$$AvCOMPCost(t_i) = \sum_{j=1}^{\text{last resource}} ApCOMP\ Cost(t_i, p_j) / \text{total resources available} \quad (3)$$

Average computation cost of the DAG: The ratio between the summation of average computation cost of tasks and the number of tasks in the task graph:

$$AvGCOMPCost(G) = \sum_{i=1}^{\text{lasttask}} AvCOMPCost(ti) / numberofTasks(G) \quad (4)$$

Average communication cost of the DAG: The ratio between the summation of communication cost of all edges and the number of edges of the task graph:

$$AvGCOMMCost(G) = \sum_{i=1}^{lastedge} COMPCost(e_i) / /numberofedges(G) \quad (5)$$

Execution time of tasks: Time of Initiation (ToI), Time of Completion (ToC), Finish Time (FT): ToI is given as the time when a resource starts executing a particular task. TOI is zero for all tasks in the first level:

$$ToI(t_i, p_j) = 0;\ 1 \le t_i \le lasttask \text{ and } t_i \in \text{initial level},\ 1 \le p_j \le \text{total resources available} \quad (6)$$

ToC is given as:

$$ToC(ti, pj) = APCompCost(ti, pj) \quad (7)$$

Finish Time is defined as the time that a particular resource completes the execution of a particular task:

$$FT(ti, pj) = ToI(t_i, p_j) + ToC(t_i, p_j) \quad (8)$$

For tasks in other levels, the recursive computation of ToI, ToC is done through the following arithmetic.

Task Arrival Time on resources (TAT), Finish Time (FT):

for i = 1 to last task do
for k = 1 to last resource do:

$$PrefRes(task) = Resource(\min(ApCOMPCost(t_i, r_j))) \quad (9)$$

$$If\ (PrefRes(t_i) = PrefRes(t_p(t_i))\ then\ COMMCost(parent, child) = 0 \quad (10)$$

else
for i = 1 to last task do
for j = 1 to parents ($t_c(i)$) do
for k = 1 to lastresource do:

$$TAT(t_p(i), r_k) = FT(t_i, r_k) + COMMCost(t_p(i), t_c(j))\} \quad (11)$$

Task Execution Time (TET), Resource Gear-up Time (RGT): If resource of child is not same as the resource of parent:

$$If\ (PrefRes(task\ is\ free)\ waittime(task) = 0 \quad (12)$$

else:

$$waittime(task) = FT(current\ task, PrefRes(task)) - TAT(task, PrefRes(task)) \quad (13)$$

for i = 1 to last task
for j = 1 to last processor:

$$RGT(p_j) = TAT(task, ores) + wait\ time \quad (14)$$

$$ToI(t_c(i), p_j) = RGT(p_j) \quad (15)$$

$$ToC(t_c(i), p_j) = ApCOMPCost(t_i, p_j) + ToI(t_c(i), p_j) \quad (16)$$

Relation Matrix (RM):
for i = 1 to last task
for j = 1 to last task
if (task (t_j) is the parent (t_i)):

$$rv[i][j] = 1 \quad (17)$$

where,
t_p = Parent task
t_c = Child task
$t_p(t_c)$ = Parent of child task
rv = Relation value

Resource List (RL):

$$for\ (i = 1 to max\ level)\ lmaxtasks = \max(\text{number of tasks of level } (l_i)) \quad (18)$$

$$RL = (lmaxtasks/2) + 1 \quad (19)$$

THE PROPOSED ESTD ALGORITHM

Algorithm 1, Algorithm 2 give the pseudocode for ESTD.

Algorithm 1:
1. Algorithm ESTD (DAG G, ResouceList RL)
2. {
3. //number of resources is expected to be the maximum of tasks contained in levels minus one of the DAG
4. G = (V,E);
5. V = (v_i, COMPCost (v_i)); $1 \le i \le n_v$
6. E = e_i ((v_i, v_j), COMMCost (v_i, v_j)); $1 \le v_i \le n_v$, $1 \le v_j \le n_v$, $1 \le e_j \le n_e$ $i \ne j$
7. RL = (r_i, processing speed (r_i)); $1 \le i \le n_p$
8. if (AvCOMMCost (G) >AvCOMPCost (G))
9. Duplication (G) //Duplication is highly effective
10. }

Algorithm 2:
1. Algorithm Duplication (G)
2. {
3. Find all paths that connect the entry and exit tasks

4. Find the number of dependents for each task in the paths
5. Find the count of dependents for each task from all paths
6. Group the tasks with respect to the level of their presence
7. priority-list: A non increasing order level-wise//internal sorting is done with respect to the count of dependents of each task in a particular level
8. level = 1
9. while (level< = maxlevel) //while1
10. {
11. priority-list (level) = sorted tasks (level)
    //3≤maxtasksinPriority-list≤6, no of tasks recommended for duplication is one
    //6 (n - 1) +1<maxtasksinPriority-list≤6n, n number of tasks are duplicated
12. Task Duplication Factor (TDF): Duplication is supported for high order tasks of priority-list and is done from level = 1 to level/2
13. while (resource available && tasks remain unassigned in the priority-list) //while2
14. {
15. if (level = 1&& task count of level = 1)
16. Task is scheduled on all available resources
17. else
18. {
19. Task Selection Factor (TSF): Task from priority-list that has minimum execution time on the resource
20. Tasks MFT = find Min Finish Time Tasks (resource-id) //tasks having
    //min APCOMPCost on the available resource
21. If (TasksMFT>1)
22. Task MFT = find Max Comm Cost Task (task-id) //task having maximum
    //communication cost of all Tasks MFT to connect its descendent
23. check for parents in the relation matrix
24. while (parents)
25. {
26. check Finished Task Table for resource-id of parents
27. Resource Priority List = sorted list of count of resource-id of parents
    //Priority is given to high order resource-id
28. if (available resource-id = resource-id of parents)
29. Commute the results from other parents and schedule the task on resource
30. else
31. Task is scheduled on resource-id of next Min Finish Time//if required
    //commute results from parents executed on other resources-id
32. }
33. }//else
34. while (task is executed && resource is free) //while3
35. {
36. update Prioritized Resource Available List (resource-id)
37. update Finished Task Table (task-id, resource-id, Finish Time)
38. if (task-id is not higher order task)
39. Remove task from the priority-list
40. if (task-id is high order task and duplication> (number of dependents/2) +1)
    //Finish Task Table is used to find the number of times a task is executed
41. Remove task from the priority-list
42. }//while3 ends
43. }//while2ends
44. level = level+1
45. }//while1 ends
46. }//Algorithm ends here

During the initial stage itself the algorithm decides the need for duplication. Task Duplication Factor (TDF) is to finalize which task must be duplicated. The process of deciding the task to be duplicated is called include directive. Task Selection Factor (TSF) is responsible for finalizing the task to be executed on available resources while process of execution is called forward directive. The priority-list is loaded with tasks belong to a particular level that are arranged in non-increasing order with respect to the count of descendents of each task from all paths. The position of each task in the priority-list is the static task rank of the particular task. The relation matrix helps to trace the dependencies between tasks. Tasks to be duplicated and the number of duplications to be made are restricted. The levels are also restricted to allow their tasks for duplication.

The Finish Task Table is updated with completed tasks along with the information of resource utilized to execute the task and finish time of task on the resource. As soon as the resource completes its execution, its availability is updated in the Prioritized Resource Availability List. The task having the minimum execution time on the available resource is preferred to execute on the resource. If the task could not get the resource that is preferred, then the resource supporting next minimum execution time is preferred.

Unnecessary duplications may exceed the schedule length and the resources would be kept needlessly busy. If the resource is no longer needed until the completion of DAG, the resource may be relieved for other works after sending the results to other resources that require it. ESTD algorithm ensures that every task is executed on a resource that supports minimum execution time.

Tasks can be easily duplicated for execution only when the number of tasks ready to be executed is lesser than the available resources. Alternatively when the

number of tasks are more than available processors, then based on the dependency in the next successive levels, the task(s) will be duplicated. The number of tasks in the initial (first) level may be one or many. The execution of the first level tasks begins at once since they are not dependents of other tasks. The mapping can be done:

- When it is one task at the first level, then it would be executed in all the available processors.
- When it is multiple tasks at the first level, then based on the dependency population of the task, it would be executed in more processors (i.e., when the dependency population is more; the task is duplicated with respect to availability of resources and the population of dependency).

Experiments: A sample DAG is shown in Fig. 1, marked edge values represent communication costs. The task pointed by the arrow head is the child of the task which is to the other end of the arrow. The computation costs of the tasks on available heterogeneous resources (R1, R2 and R3) are shown in Table 1. The mapping of task and resource is done by the algorithm. Table 1 to 5 is associated with DAG shown in Fig. 1.

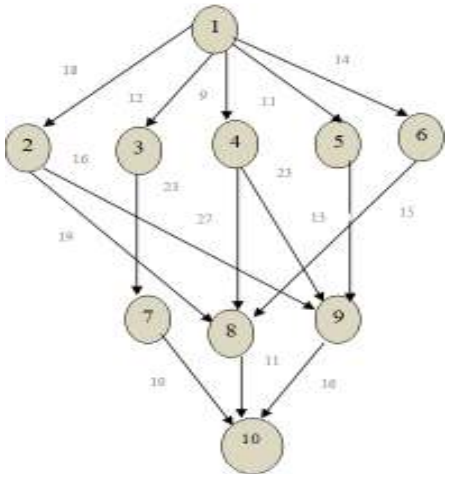

Fig. 1: Sample DAG with communication cost

Table 1: Computation cost of tasks on resources

Computation cost on resources			
Tasks	R1	R2	R3
T1	14	16	9
T2	13	19	18
T3	11	13	19
T4	13	8	17
T5	12	13	10
T6	13	16	9
T7	7	15	11
T8	5	11	14
T9	18	12	20
T10	21	7	16

AvGCOMPCost (G) = 13; AvGCOMMCost (G) = 16

Paths and Dependents of Task in Path (DTP) are shown in Table 2. Table 3 lists the count of dependents of each task from all paths, the level-wise sorting of task is shown in Table 4 (Prioritized tasks to be duplicated are shown in bold).

Table 5 presents the execution sequence of the DAG shown in Fig. 1 and the makespan is found to be 71.

RESULTS AND DISCUSSION

Significant performance improvement is observed when assessing the proposed algorithm with TBTGP, EDS-G and DIBS. To assess the performance of algorithms the following factors are measured:

- Makespan
- Schedule length ratio
- Speed-up ratio
- Load balancing

When considering the compared algorithms, the result consistencies on makespan is remarkable for ESTD. The results are assessed using many DAGs of different sizes. Graphs generated for simulations are fully connected. Random Task Graphs are generated using

Table 2: Paths and dependencies

Path	Tasks between entry and exit tasks			
P1	1	2	8	10
DTP1	5	2	1	0
P2	1	2	9	10
DTP2	5	2	1	0
P3	1	3	7	10
DTP3	5	1	1	0
P4	1	4	8	10
DTP4	5	2	1	0
P5	1	4	9	10
DTP5	5	2	1	0
P6	1	5	9	10
DTP6	5	1	1	0
P7	1	6	8	10
DTP7	5	1	1	0

Table 3: Count of dependents of tasks from all paths

Task	Count of dependents
1	35
2	4
3	1
4	4
5	1
6	1
7	1
8	3
9	3
10	0

Table 4: Internal sorting based on count of dependents

Level	Sorted tasks
L1	1
L2	4, 2, 5, 6, 3
L3	9, 8, 7
L4	10

Table 5: Resource consumption of tasks

	Resource utilization by tasks in cost/time						
Time/cost	R1	R2	R3	Time/cost	R1	R2	R3
1	T1	T1	T1	37			
2				38		T7	
3				39			
4				40			
5				41	T8		
6				42			
7				43			
8				44			
9				45			
10			T6	46			
11				47			
12				48			
13				49			
14				50			
15	T2			51			
16				52			
17		T4		53		T9	
18				54			
19			T5	55			
20				56			
21				57			
22				58			
23				59			
24				60			
25		T3		61			
26				62			
27				63			
28	T4			64			
29				65		T10	
30				66			
31				67			
32				68			
33				69			
34				70			
35				71			
36							

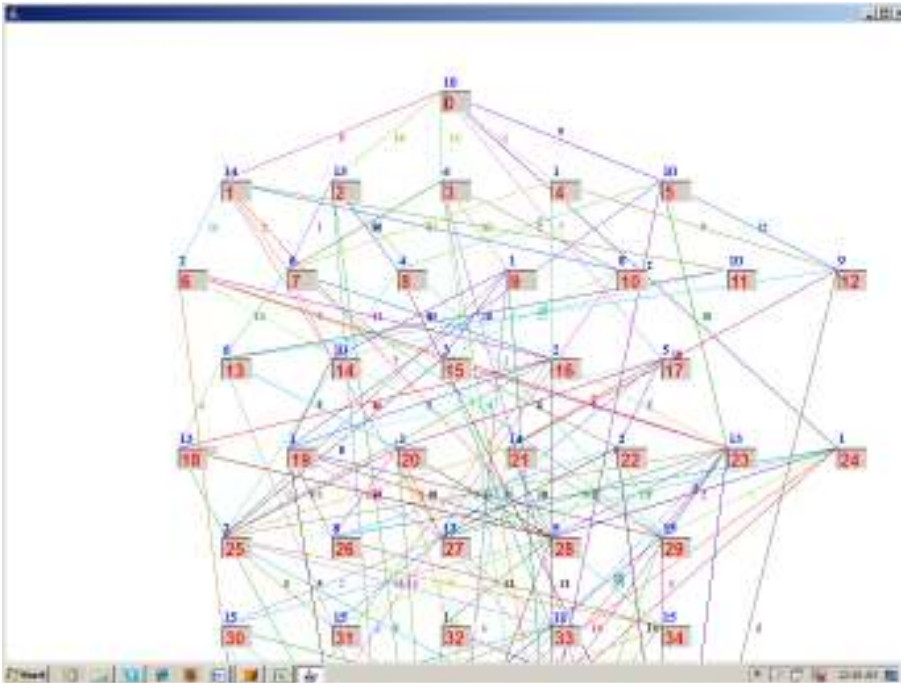

Fig. 2: DAG generated using random DAG generator

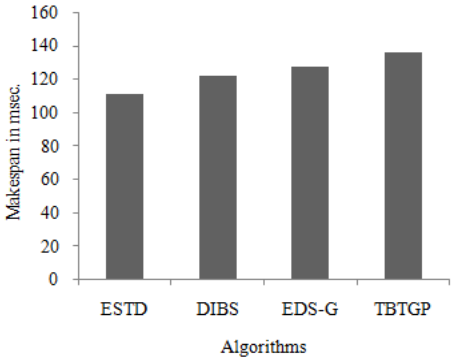

Fig. 3: Comparison of make span of algorithms

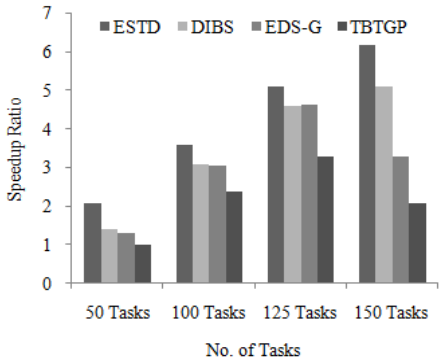

Fig. 4: Speedup ratio of algorithms with different DAG sizes

Table 6: Schedule length of algorithms

Algorithms	Schedule length
ESTD	71
DIBS	74
EDS-G	76
TBTGP	78

Table 7: Algorithms and make span

Tasks in DAG	Algorithms and schedule length (6 resources used)			
	ESTD	DIBS	EDS-G	TBTGP
50	111	122	127	136
75	127	136	139	142
100	202	217	221	236
125	236	243	247	251

a task graph generator, which was developed as part of our work (George Amalarethinam and Muthulakshmi, 2012). Figure 2 shows the DAG generated by task graph generator. It could generate DAGs of various sizes and wide range of attribute values. Consistent load balance is observed in most of the executions. The performance is scaled for a large range of CCR values. The results of experiments show that ESTD surpasses the other compared algorithms.

Schedule length ratio of DAG is defined as the ratio between the schedule length and the summation of minimum computation cost of tasks. Low value is achieved on schedule length ratio in almost all executions that uses randomly generated graphs.

Speedup ratio of DAG is the ratio between minimum computation cost obtained by executing all the tasks of DAG in one resource and the schedule length. Most of the speedup ratio values are found to be more than one.

Table 6 presents the schedule length obtained after executing the sample DAG shown in Fig. 1. Table 7 gives the comparison of schedule length of algorithms for various sizes of DAG. Executions of needlessly duplicated tasks are observed in the other two duplication algorithms. The usage of communication links to commute results from parents to child is more in DIBS and EDS-G.

Figure 3 expresses the performance of algorithms in terms of makespan, Fig. 4 shows the speedup ratio of algorithms.

Time complexity: The time complexity of ESTD algorithm is given as O (2 np), where 'n' is number of tasks in DAG and 'p' represents the resources reserved for executing the DAG.

CONCLUSION

In this study, a duplication based DAG scheduling algorithm called ESTD algorithm is proposed. Many algorithms of its kind are implemented using Java and it encourages using Java for the implementation of algorithms for comparison and analysis. It adopts list and cluster scheduling for task prioritization. Greedy method is employed to find the resource that minimizes task execution time. Tasks are grouped based on level of their presence and are ranked based on their dependents. The prioritized tasks in the group are chosen for duplication. The algorithm limits the number of duplications. The number of resources to be employed is fixed and is held back only when they are expected to be busy; otherwise the resources would be relinquished to be used by other applications. As it is a static algorithm the plan of execution is already known and it is obvious that the busy time of the employed resources must be known. Hence the resources are released as soon as the mapped tasks are executed and found no longer needed. The message passing across resources is considerably reduced and therefore contention free execution is supported. It is found that most of the tasks are executed in resources that support minimum execution time. The results demonstrate the performance of ESTD algorithm in comparison with algorithms of its kind.

REFERENCES

Ahmad, I. and K. Yu-Kwong, 1994. A new approach to scheduling parallel programs using task duplication. Proceedings of the International Conference on Parallel Processing, pp: 47-51.

Amit, A. and K. Padam, 2010. Economical task scheduling algorithm for grid computing systems. Global J. Comput. Sci. Technol., 10: 48-53.

Bozdag, D., U. Catalyurek and F. Ozguner, 2006. A task duplication based bottom-up scheduling algorithm for heterogeneous environments. Proceeding of the 20th International Parallel and Distributed Processing Symposium.

Chan-Ik, P. and C. Tae-Young, 2001. An optimal scheduling algorithm based on task duplication. Proceeding of the 8thInternational Conference on Parallel and Distributed Systems, pp: 9-14.

George Amalarethinam, D.I. and F.K. Malai Selvi, 2012. A task duplication based efficient multi-objective grid workflow scheduling algorithm. Int. J. Adv. Res. Comput. Sci., 3: 87-92.

George Amalarethinam, D.I. and P. Muthulakshmi, 2012. DAGitizer-a tool to generate directed acyclic graph through randomizer to model scheduling in grid computing. In: Wyld, D.C. *et al*. (Eds.), Advances in Computer Science, Engineering and Application. AISC 167, Springer-Verlag, Berlin, Heidelberg, pp: 969-978.

George Amalarethinam, D.I. and P. Muthulakshmi, 2014. A proficient low complexity algorithm for preeminent task scheduling intended for heterogeneous environment. J. Theor. Appl. Inform. Technol., 67: 1-11.

Kai, H. and A.B. Faye, 1985. Parallel Architecture and Parallel Processing. McGraw Hill International Editions, NY.

Koichi, A., S. Bing and W. Toyohide, 2006. A task duplication based scheduling algorithm for avoiding useless duplication. Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications and Conference on Real-time Computing Systems and Applications.

Li, G.D., D.X. Chen, D.M. Wang and D.F. Zhan, 2003. Task clustering and scheduling to multiprocessors with duplication. Proceedings of the International Parallel and Distributed Processing Symposium.

Lijun, C., L. Xiyin, H.G. Torkel and Z. Zhongping, 2013. Task scheduling algorithm in grid environment based on duplication and insertion. J. Softw., 10: 2447-2454.

Menglan, H. and V. Bharadwaj, 2012. Requirement-aware scheduling of bag-of-tasks applications on grids with dynamic resilience. IEEE T. Comput., 62: 2108-2114.

Oliver, S., T. Andrea and K. Manpreet, 2011. Contention-aware scheduling with task duplication. J. Parallel Distr. Com., 71: 77-86.

Prodan, R. and W. Marek, 2010. Bi-criteria scheduling of scientific workflows. IEEE T. Autom. Sci. Eng., 7: 364-376.

Ranaweera, S. and D.P. Agarwal, 2000. A task duplication based scheduling algorithm for heterogeneous system. Proceeding of the 14th International Parallel and Distributed Processing Symposium.

Sandnes, F.E. and G.M. Meson, 2001. An evolutionary approach to static task graph scheduling with task duplication for minimized interprocess or traffic. Proceeding of the International Conference on Parallel and Distributed Computing, Applications and Technologies.

Savina, B., K. Padam and S. Kuldip, 2005. Dealing with heterogeneity through limited duplication for scheduling. J. Parallel Distr. Comp., 65: 479-491.

Topcuoglu, H., S. Hariri and W. Min-You, 2002. Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE T. Parall. Distr., 13: 260-274.