Indian Journal of Science and Technology, *Vol 8(15), 71373, July 2015*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Memory Constrained Load Shared Minimum Execution Time Grid Task Scheduling Algorithm in a Heterogeneous Environment

M. Hemamalini[1*] and M. V. Srinath[2]

[1]Department of Computer Science, A. V. C. College (Autonomous), Mannampandal, Mayiladuthurai - 609305, Tamil Nadu, India; maliniavcce@gmail.com

[2]Department of MCA, Sengamala Thayaar Educational Trust Women's College, Mannargudi, Thiruvarur - 61400, Tamil Nadu, India; Sri_induja@rediffmail.com

Abstract

Objective: To increase the resource utilization and balance the load in the grid environment. **Methods:** Memory Constrained Load Shared Minimum Execution Time (MCLSMET) scheduling is proposed to make best use of the resource utilization in a grid environment to reduce makespan. Load balancing is achieved by rescheduling the resources based on memory requirement and execution time of the tasks. This algorithm considers memory as Quality of Service (QoS) factor. **Results:** The proposed algorithm has been implemented in a simulated environment and the results are compared with the Minimum Execution Time (MET) algorithms. In MCLSMET algorithm, the Maximum Completion Time, Resource Utilization is computed to compare with the existing MET scheduling Algorithm. The MET scheduling algorithm produces the makespan 34 ms whereas the proposed method reduces the makespan to 15 ms for a task. In the existing MET scheduling Algorithm produces severe load imbalance problem. In the proposed method load is shared among the available resource and the resource utilization percentage is increased. **Conclusion:** The Memory Constrained Load Shared Minimum Execution Time (MCLSMET) scheduling algorithm is suggested that this algorithm produces higher resource utilization, reduces the makespan and load balancing.

Keywords: Grid Task Scheduling, Heterogeneous Environment and Load Balancing, Quality of Service, Resource Sharing

1. Introduction

Grid Computing[1] is a distributed system that enhances computing facilities. Grid software addressed the problems like fault tolerance, security, heterogeneity and resource allocation[2]. Computational Grids[3] are considered as the next generation of distributed system. Many researchers focus on the challenging issues like scheduling and resource management in the grid computing era. Scheduling[4] is the most emerging area in the Grid Scenario. Effective and efficient task scheduling algorithm is needed to achieve high performance in grid environments. The main aim of grid task scheduling is to increase resource utilization and reduce the makespan. The success of the grid computing relies on how effectively it schedules the tasks with available resources. Grid system allocates the tasks to the available resources based on user's requirement. Heterogeneous computing environment utilizes the different high performance resources to perform massive application that have different computational requirements[1,5-9]. The matching of tasks to resources and scheduling the execution order of these tasks is referred to as mapping. The general problem of

**Author for correspondence*

mapping tasks to resources in a heterogeneous environment has been exposed to be NP-Complete[10,11].

Meta task can be described as a group of independent tasks with no inter task data dependencies. The main objective of this mapping is to reduce the total execution time of the Meta task. It is also assumed that each resource executes a single task at a time based on the order in which the tasks are assigned. The size of the Meta task and the available number of resources are known priori[12]. Many task scheduling algorithms are available to increase the resource utilization and throughput[13-18]. These algorithm schedules the tasks to the resources which will minimize the overall completion time. Simple and known scheduling algorithms are Min-min, Minimum Execution Time, Max-min[14,16,18-21]. These algorithms schedules the tasks based on execution and completion time of each task on each available resource.

Load balancing algorithm distributes the load among all the available resources. The algorithm tries to enhance the utilization of resources with light load and freeing the resources with heavy load. Execution Time and Memory requirement are the two common factor used for load balancing and effective Utilization of resources[22]. These algorithms are mainly used to reduce the makespan and enhance the utilization of resources.

Braun et al[12] have studied the performance of eleven grid task scheduling algorithms. Their result shows Genetic Algorithm (GA) outperforms the other algorithm. Min-Min algorithm performs next to GA and the rate of improvement is very small. The Task scheduling algorithms proposed by Braun are Min-Min, Max-min, Minimum Execution Time (MET), Opportunistic Load Balancing (OLB) and Minimum Completion Time (MCT). The Algorithm Opportunistic Load Balancing (OLB) assigns the jobs in a random order in the next available resource without considering the execution time of the jobs on those resources. Thus it provides a load balanced schedule but it produces a very poor makespan.

Min-Min grid task scheduling algorithm finds the task which has minimum execution time and schedules the task to the resource that produces minimum completion time. The ready time of resource is updated. This procedure is repeated until all tasks are scheduled.

The Max-min grid task scheduling algorithm is similar to Min-min scheduling algorithm but it schedules the larger task first. The ready time of resource is updated. This process is repeatedly executed until all unmapped tasks are assigned.

Minimum Completion Time (MCT) grid task scheduling algorithm finds the resource which has Minimum Completion Time for the task. It assigns the task to resources based on completion time. Completion time is computed by adding the ready time and the execution time of the resource.

Minimum Execution Time (MET) grid task scheduling algorithm finds the task which has minimum execution time and assigns the task to the resource based on first come first served basis. The main drawback of this algorithm is severe load imbalance. It does not consider the availability of the resource and its load.

Optimal Resource Constraint Scheduling algorithm distributes the task among the available processor based on processor capability. It is an efficient load balanced task scheduling algorithms which reduces the turnaround time and average waiting time. It is suitable for more number of jobs and avoids starvation problem.

T. Kokila vani et al proposed Load Balanced Min-Min scheduling algorithm which produces better results than min-min scheduling algorithm. It reduces the makespan and balance the load. The response time is improved and load balancing is achieved efficiently. This algorithm applies the min-min grid scheduling algorithm in the first phase and rescheduling takes place based on maximum execution time13. Resource utilization for a particular problem is calculated using the formula 1.

$$RU = Ti * 100 / TARU \qquad (1)$$

$$TARU = \sum_{i=1}^{n} CT \qquad (2)$$

/*TARU – Total Amount of Resource Used.*/

T. Kokilavani et al, proposed a grid task scheduling algorithm "An Ant Colony Optimization Based Load Sharing Technique" which distributes the load among available resources based on the behavior of argentine ants. The resources should be chosen and scheduling can be performed based on RAM requirement as Quality of service factor. The ants choose the path based on the probability value and the memory requirement of task. The Probability value Pj can be calculated using the following formula.

$$Pj = (Ri + K)h / \sum_{i=1}^{n} (Ri + K)h$$

Allotment Percentage $Aj = Pj * TRi$

Where Aj is the amount of task allocated in resource Rj and TRi is the memory requirement of task Ti. The probability of choosing the resource will change based on the value of the coefficient K and h. This algorithm shares the load among the resources and reduces the overall response time based on memory as a Quality of service factor[23].

He X., et al[24] have proposed a new grid task algorithm based on the Min-Min algorithm. The QoS guided Min-Min algorithm, schedules tasks which requires high bandwidth. Therefore, if the bandwidth required by different task varies extremely, the QoS guided Min-Min algorithm gives improved results than the Min-Min Meta task scheduling algorithm.

Sameer Singh et al[25] have proposed QoS Guided Weighted Mean Time-Min (QWMTM) Heuristic algorithm and QoS Guided Weighted Mean Time Min-Min Max-Min Selective (QWMTS) scheduling algorithm. In these algorithms network bandwidth is taken as QoS parameter.

2. Problem Statement

Task scheduling is one of the NP-Complete problems. Let T1, T2, T3, T4 and T5 are collection of independent tasks. The tasks that have no dependency among each others are referred as meta task. Each task is assigned to a resource based on the order in which the tasks are arrived and memory requirement of the task.

The input to this algorithm is number of resources, characteristics of resource and size of the meta task. The expected execution time for each task on each resource is known prior to execution. Expected Time to Compute Matrix ETC (Ti, Rj) contains the execution time of each task and memory requirement of each task. Where Ti represents meta-task and Rj represents Resource Set. The Problem can be defined as follows:

Let task set Ti = T1, T2, T3, T4, …, Tn.

Let Resource Set Ri = R1, R2, R3, R4,…….,Rn.

The main drawback of MET scheduling algorithm is severe load imbalance. The aim of the proposed grid task scheduling algorithm is to effectively utilize the idle time of the resources, minimizes the makespan and balances the load based on memory requirement of the task. This algorithm considers memory as Quality of Service factor. The makespan of the task can be calculated as follows:

Makespan = max (CT (Ti, Rj))

CTij = Rj + ETij

CT = Completion Time.

Rj = Ready Time of Resource j.

ETij = Execution time of Task i on Resource j.

Grid task scheduling is one of the NP-Complete Problem used to find the acceptable solution with fewer cost.

2.1 Memory Constrained LSMET

Our proposed grid task scheduling algorithm, Memory Constrained LSMET, is shown below. The algorithm considers memory as Quality of service factor and starts by executing the steps in Minimum Execution Time scheduling strategy first. It first identifies the tasks having minimum execution time, memory requirement of the task and the resource needed for executing it. Thus the task with minimum execution time is scheduled first in MET with first come first served order.

In this method the resources that are capable of fast execution is overloaded with many tasks and rest of the resources remain idle. It produces severe load imbalance. To avoid this load imbalance problem, proposed scheduling algorithm schedules the task in a better manner and improves the makespan and balances the load. We call this heuristics as Memory constrained Load shared Minimum Execution Time. In the first phase Memory constrained LSMET schedules task based on minimum execution time.

Second Phase is an iterative process. In this phase, it selects the resources which require more memory and reassigns the task to the resource having sufficient memory. This Memory Constrained LSMET algorithm tries to minimize the makespan by swapping tasks between resources. A set of independent task and resource set are the input of this heuristic algorithm. Select the resource Ri requires more memory than the memory capacity of allotted resource. If moving any of the tasks from the resource Ri to some other resource might result in a smaller makespan overall; if such task exists, then it is rescheduled to the resource that minimizes the makespan.

For single iteration three actions need to take place. 1. Select a task which requires more memory than the memory capacity of allotted resource. 2. Find task Ti that has minimum execution time in other resource Rj. 3. If such a resource is found, move the task to the resource for load balancing. Compute the completion time MCT of each resource if task Ti was to be inserted to the resource list Rj. Then the MCT of the task is compared with the makespan produced by MET algorithm. If maximum completion time is less than the makespan then the task is rescheduled in the resource that produces it. Then the ready time of both resources are updated.

This process terminates when none of the resource giving the maximum completion time can be moved to any other resource. Memory constrained LSMET increases load balancing and reduce the overall completion time. Since it compares the maximum completion time with the makespan produced by MET, reduces the overall completion Time and balance the load.

Memory Constrained Load Shared Minimum Execution Time Algorithm (MCLSMET)

For all tasks
For all resource
Find the minimum execution time and the resource producing it
End for
Schedule the task on that resource
End for
/*Rescheduling */
For all resource
Select the task Ti which requires more memory than the memory
Capacity of allotted resource
Select the resource Ri producing the Maximum Completion Time M
End for
For all task in scheduled list
Select the first task Ti
Find the next minimum execution time produced by Resource Rj for task Ti
If M < MCT then
Schedule the task on that resource Rj
End if
Update the Completion Time of all resource.
End for.

Algorithm 1. Algorithm for Memory constrained Load shared Minimum Execution Time Grid task scheduling.

3. An Illustrative Example

Consider a heterogeneous grid environment with two resources R1 and R2 and independent task group M with four tasks T1, T2, T3 and T4. The grid scheduler schedules all the tasks on the available resources R1 and R2.

Since Minimum Execution Time algorithm is simple and produces a better makespan. But load imbalance problem occur in Minimum Execution Time Scheduling Algorithm. To avoid the problem of unbalanced load in MET, the tasks are rescheduled in the second phase based on memory requirement and completion time. The Execution Time of all tasks is known prior. Table 1 represents the resource characteristics and Table 2 represents the Execution Time of the Tasks on each resource.

Table 1. Resource characteristics

Resource	Available RAM
R1	60
R2	40
R3	70

Table 2. Execution time and memory requirement of the tasks

Task/ Resource	Memory Requirement of Task	R1	R2	R3
T1	30	10	5	9
T2	40	12	8	10
T3	70	13	9	10
T4	55	15	12	16

Scheduling of the tasks to resources based on Minimum Execution Time as given in Algorithm. MET chooses the minimum execution time, so all tasks are assigned to Resource R2 and Resource R1 becomes idle. The makespan produced by MET is 34 sec.

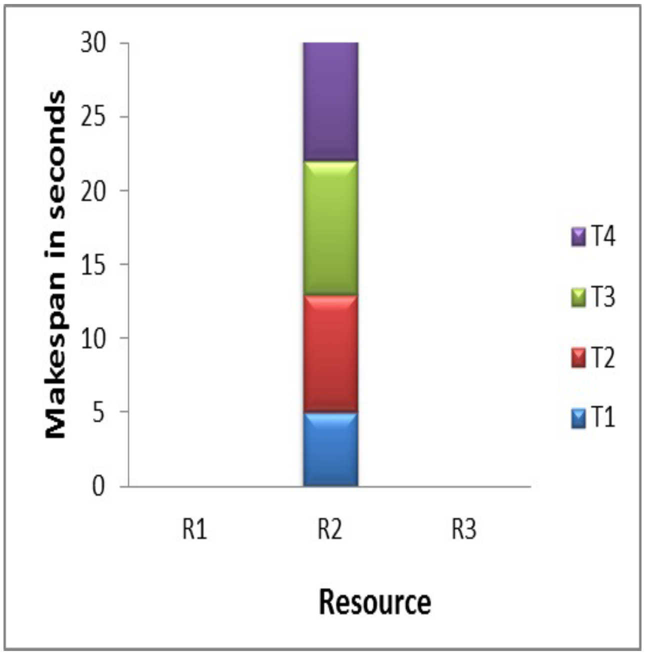

Figure 1. MET.

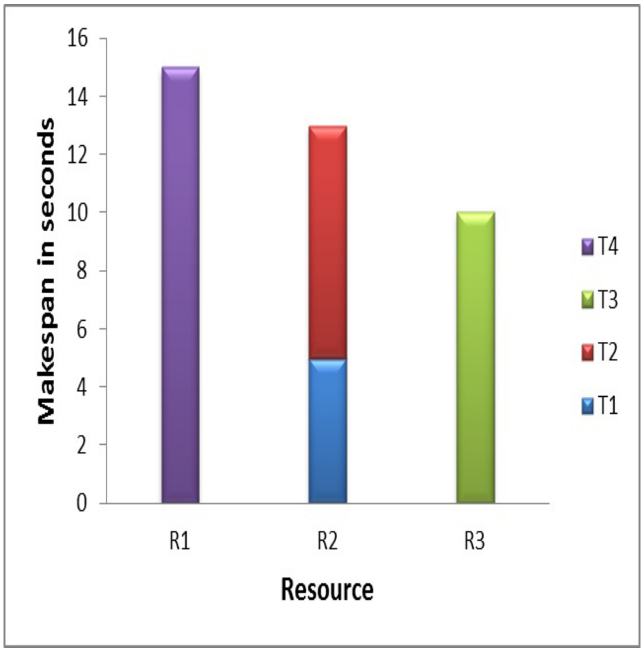

Figure 2. Memory constrained LSMET.

3.1 Memory Constrained LSMET

Proposed algorithm minimizes the makespan by rescheduling the task based on memory requirement of the task. The resource giving the maximal completion Time is R2.The task from R2 is moved on to another resource R1 and R3 based on the memory requirement of the task and memory availability of the resource.

Mark the entire task in R2 as checked and the remaining task as unchecked. Task T4 requires memory requirement 54MB and is having the minimum completion Time in Resource R1. The available memory for Resource R1 is 60 MB which satisfies memory requirement of the task T4. So Task T4 moved on to Resource R1 and Task T3 is moved on to Resource R3 and the remaining task scheduled in R2. The result of Memory constrained LSMET is shown in Figure 1. Memory constrained LSMET algorithm utilizes the idle resource R1 and R3 and minimizes the makespan to 15 sec.

4. Results and Discussion

Let us take the example problems having both task and resource heterogeneity and executes for both MET and proposed Memory constrained LSMET scheduling algorithm. Software is developed in Eclipse for both algorithms. The Table 3 shows the results (in sec) of both algorithms.

Table 3. Comparison of MET and memory constrained LSMET algorithm

Problem Set	MET	Memory Constrained LSMET
P1	34	15
P2	17	14
P3	33	25
P4	16	14
P5	27	36

The results are plotted in a graph. Memory constrained LSMET produces less makespan than MET scheduling algorithm. The Figure 2 shows Memory constrained LSMET outperforms MET scheduling algorithm.

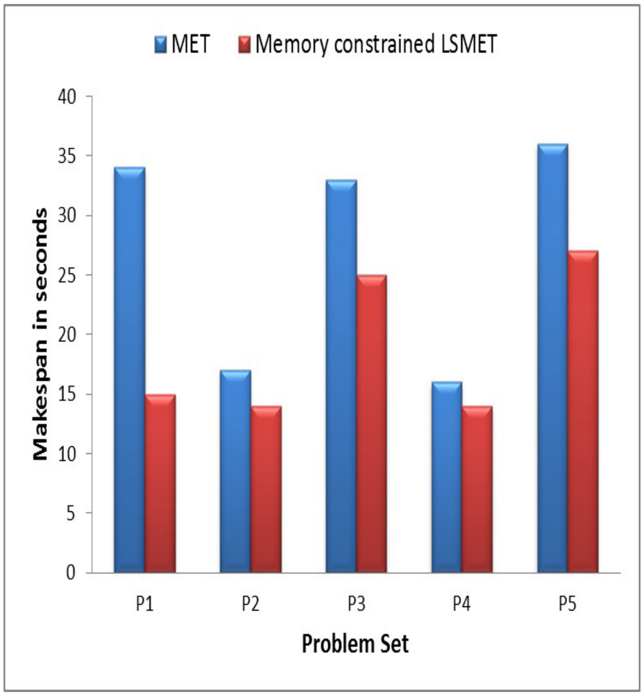

Figure 3. Graphical representation to show the performance of memory constrained LSMET.

Table 4 shows the resource utilization of both MET and Memory constrained LSMET scheduling algorithm. The Memory Constrained LSMET balances the load and reduces the makespan by using unutilized resource in the second phase. Table 4 shows that Memory Constrained LSMET efficiently utilizes all the available resource. Resource Utilization can be calculated using the formula 326.

$$UR = Ti * 100 / TQRU \quad (3)$$

$$TQRU = \sum_{i=1}^{n} CT \quad (4)$$

TQRU = Total Quantity of Resource Used.
Ti = Meta task.
UR = Usage of Resource.
CT = Completion Time of Task.

The resource utilization percentage is shown in Figure 4. From this figure we can observe that Memory Constrained LSMET uses the maximum amount of resources while reducing the makespan obtained from

Table 4. Resource utilization rate

Problem Set	Resource	MET	Memory Constrained LSMET
P1	R1	0	100
	R2	100	86.6
	R3	0	90.9
P2	R1	47.05	71.4
	R2	0	100
	R3	100	85.7
P3	R1	30.3	88
	R2	0	100
	R3	100	60
P4	R1	50	71.4
	R2	0	100
	R3	100	78.5
P5	R1	30.5	85.2
	R2	0	100
	R3	100	59.25

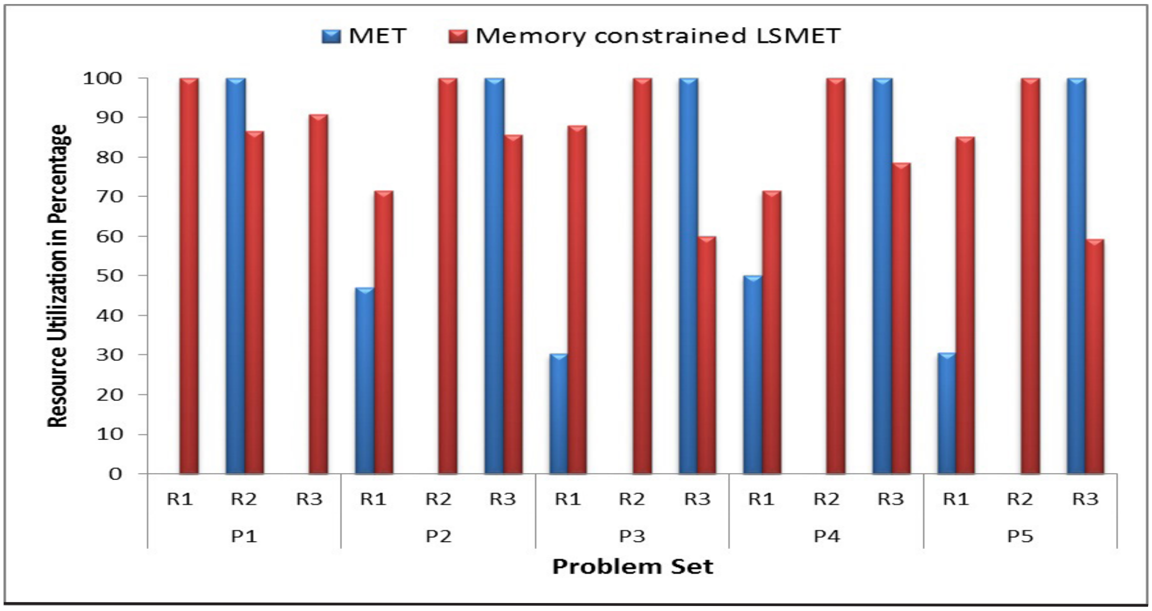

Figure 4. Resource utilization in percentage.

MET algorithm. Thus Memory Constrained LSMET uses the idle resources for small tasks to reduce the makespan.

5. Conclusion

The main aim of Task scheduling algorithm is to utilize the resource effectively and increase the throughput and reduces the makespan. The MCLSMET grid task scheduling algorithms is compared with MET. The experimental results achieved by comparing these scheduling algorithms for different problem set. It shows that Memory constrained LSMET scheduling algorithm minimizes the makespan than MET scheduling algorithms. Memory Constrained LSMET grid task scheduling algorithm utilizes the resource effectively and produces better result. This paper identifies that Memory constrained LSMET grid task scheduling algorithm outperforms the MET algorithm in a heterogeneous distributed environment. This study can be further extended by proposing a new hybrid algorithm which combines the advantages of all the algorithms.

6. References

1. Foster I, Kesselman C, Tuecke C. The anatomy of the grid enabled scalable virtual organizations. International Journal of Supercomputer Applications. 2001.
2. Nirmala Devi K, Tamilarasi A. Dynamic scheduling in grid environment with the improvement of fault tolerant level. Indian Journal of Science and Technology. 2015 Apr; 8(S8): 507-25.
3. Chapman C, Musolesi M, Emmerich W, Mascolo C. Predictive resource scheduling in computational grids in parallel and distributed processing symposium. IEEE International. 2007; 266(6): 1-10.
4. Lorpunmanee S, Abdullah AH, Sap MN, Chompoo-Inwai C. An ant colony optimization for dynamic job scheduling algorithm in grid environment. World Academy of Science, Engineering and Technology. 2007; 29(5):314-21.
5. Eshaghian MM. Heterogeneous Computing. Norwood, MA: Artech House; 1996.
6. Foster I, Kesselman C. The grid: blueprint for new computing infrastructure. New York: Morgan Kaufman; 1998.
7. Freund RF, Siegel HJ. Heterogeneous processing. IEEE Computing. 1993 Jun; 26(6):13-7.
8. Maheswaran M, Braun TD, Siegel HJ. Heterogeneous distributed computing. Encyclopedia of Electrical and Electronics Engineering. In: Webster JG, editor. New York: Wiley; 1999; 8(1): 679-90.
9. Siegel HJ, Dietz HG, Antonio JK. Software support for heterogeneous computing. The Computer Science and Engineering Handbook. In: Tucker AB, Jr., editor. Boca Raton, FL: CRC Press; 1997. p. 1886-1909.
10. Fernandez-Baca D. Allocating modules to processors in a distributed environment. IEEE Trans Software Engg. 1989 Nov; 15(11):1427-36.

11. Ibarra OH, Kim CE. Heuristic algorithms for scheduling Meta tasks on non identical processors. J Assoc Comput Mach. 1977 Apr; 24(2):280-89.
12. Braun TD, Siegel HJ, Beck N, Boloni LL, Maheswaran M, Reuther AI, Yao B, Theys MD, Robertson JP. A comparison of eleven static heuristics algorithm for mapping a class of Meta tasks onto heterogeneous distributed computing Environment. Journal of Parallel and Distributed Computing. 2001; 61:810-37. DOI: 10.1006_jpdc.2000.1714
13. Kokilavani T, Amalarethinam GDI. Load balanced min-min static meta-task scheduling algorithm in a grid computing environment. International Journal of Computer Applications. 2011; 20(2):43-9.
14. He X, Laszewski GV, Sun X-H. QoS guided min-min heuristic for grid task scheduling. Journal of Computer Science and Technology. 2003; 18:442-51.
15. Chauhan RSS, Joshi C. QoS guided heuristic for grid task scheduling algorithm.International Journal of Computer Applications. 2010 June; 2(9):24-31.
16. Etminani K, Naghibzadeh M. A min-min max-min selective grid task scheduling algorithm. The 3rd IEEE International Conference on Internet; 2007; Uzbekistan. p.1-7.
17. Ranganathan K, Foster I. Decoupling computation and data scheduling in distributed data intensive applications. HPDC-11. Proceedings of the 11th IEEE Symposium on High Performance Distributed Computing Environment; Edinburgh, Scotland. 2002 Jul. p. 1–7.
18. Munir UE , Li J, Shi S. QoS sufferage heuristic for independent task scheduling in grid. Information Technology Journal. 2007; 6(8):1166-70.
19. Ali S, Braun TD, Siegel HJ, Maciejewski AA. Heterogeneous Computing Encyclopedia of Distributed Computing. In: Urbana J, Dasgupta P, editors. Norwell, MA: Kluwer Academic; 2001.
20. Dong F, Ge L, GaoL, Luo J. A grid task scheduling algorithm based on QoS priority grouping. IEEE Proceedings of the 5th International Conference on Grid and Cooperative Computing (GCC'06); 2006. p. 58–61.
21. Maheswaran M, Ali S, Siegel HJ, Hensgen D, Freund RF. Dynamic mapping of a class of independent tasks onto heterogeneous computing systems. Journal of Parallel and Distributed Computing. 1999; 59:107-31.
22. Hemamlini M, Srinath MV. State of the art: task scheduling algorithm in a heterogeneous grid computing environment. Elysium Journal of Engineering Research and Management. 2014; 1(1):1-7.
23. Kokilavani T, Amalarethinam DIG. An ant colony optimization based load sharing technique for meta task scheduling algorithm in grid computing. Advances in Intelligent Systems and Computing Springer. 2013; 177(2):395-404.
24. He X, Sun XH, Laszewski GV. QoS guided min-min heuristic for grid task scheduling. Journal of Computer Science and Technology. 2003; 18:442-51.
25. Chauhan RSS, Joshi C. QoS guided heuristic algorithms for grid task scheduling. International Journal of Computer Applications (0975-8887). 2010 Jun; 2(9):24-31.
26. Hemamlini M. Review on grid task scheduling algorithm in a distributed heterogeneous environment. International Journal of Computer Applications. 2012 Feb; 40(2):24-30.