AENSI Journals

Australian Journal of Basic and Applied Sciences

ISSN:1991-8178

Journal home page: www.ajbasweb.com

Resource Fitness Task Scheduling Algorithm for Scheduling Tasks on Heterogeneous Grid Environment

Dr.G.K. Kamalam,

Department of Information Technology, Kongu Engineering College, Perundurai. Erode, Tamilnadu, India.

ARTICLE INFO

Article history:
Received 10 October 2014
Received in revised form
22 November 2014
Accepted 28 November 2014
Available online 1 December 2014

Keywords:
Grid Computing, Task Scheduling, Resource Allocation, Heuristic Algorithm, Heterogeneous Resources

ABSTRACT

Background: Grid is a heterogeneous system. Grid computing is a service for sharing computing power, heterogeneous resources and data storage facility over the distributed environment. Grid computing is providing big contributions to scientific research problems, helping the scientists located in various geographical locations of the world to analyze and store huge amounts of data distributed over the world. **Objective:** Scheduling independent task is more complicated in grid environment. To adapt to the heterogeneity and dynamism of the grid environment an efficient task scheduling algorithm is required to minimize the makespan and maximize resource utilization. This paper proposes a new heuristic Resource Fitness Task Scheduling Algorithm (RFTSA). Maximizing resource utilization and minimizing makespan are the major goals of this paper. **Results:** In order to utilize the power of grid computing efficiently, the proposed algorithm, Resource Fitness Task Scheduling Algorithm (RFTSA) assigns tasks to the resources by identifying the fitness value of the resource. **Conclusion:** The experimental results reveal that the proposed RFTSA algorithm achieves reduced makespan with higher user satisfaction.

To Cite This Article: Dr.G.K. Kamalam., Resource Fitness Task Scheduling Algorithm for Scheduling Tasks on Heterogeneous Grid Environment. ***Aust. J. Basic & Appl. Sci.,*** *8(18): 128-135, 2014*

INTRODUCTION

The computational capability and network performance have gone to a great extent. There are still problems in the fields of science, engineering, and business, which cannot be effectively dealt by using the current generation of supercomputers. The emergence of the Internet as well as the availability of powerful computers and high-speed network technologies as low-cost commodity components is rapidly changing the computing landscape and society (Lee *et al*., 2011). The emerging technology provides an opportunity that have led to the possibility of using wide-area distributed computers for solving large-scale problems, leading to what is popularly known as Grid computing.

Grid computing is the collection of computer resources from multiple locations to reach a common goal. Grid can achieve the same level of computing power as a supercomputer does, but at a much reduced cost. Grid is like a virtual supercomputer. Distributed computing supports resource sharing. Parallel computing supports computing power. Grid computing aims to harness the power of both distributed computing and parallel computing (Chang *et al*., 2012). The goal of grid computing is to aggregate idle resources on the Internet such as Central Processing Unit (CPU) cycles and storage spaces to facilitate utilization.

Grid computing enables the sharing, selection and aggregation of a wide variety of geographically distributed resources including supercomputers, databases, data sources and specialized devices owned by different organizations. The Grid users need not be aware of the computational resources that are used for executing their applications and storing their data (Foster and Kesselman, 1999).

Grid computing is driven by five major areas: Resource sharing: Global sharing is the notion of Grid computing. Secure access: Trust between Grid users and a resource provider is very important. Security policy for accessing the resources is to be defined, getting grid security right before accessing the resource is essential. Resource use: Efficient utilization of resources is crucial. The death of distance: Distance should never make any difference. Grid users should be able to access the computer resources from wherever they are. Open standards: Interoperability between different grids is successful by the adoption of open standards for the development of grid. Standardization is making successful for industry to benefit in developing commercial grid services.

Corresponding Author: Dr.G.K.Kamalam, Department of Information Technology, Kongu Engineering College, Perundurai, Erode, Tamilnadu. India.
Ph: +91 9578092100, E-mail: kamalamparames@gmail.com

Australian Journal of Basic and Applied Sciences, 8(18) December 2014, Pages: 128-135

Grid scheduling techniques are: Mapping heuristics: Given a set of 'n' tasks, both independent or dependent tasks and a set of 'm' heterogeneous resources distributed over the world. The problem of mapping 'n' tasks into the 'm' resources subject to the minimization of the overall completion time of all the tasks is a well-known NP-Complete problem (Kamalam and Murali Bhaskaran, 2011). Resource reservation: Grid resource can be reserved in advance to execute a designated set of tasks. If the reservation period of the resource is reached, the currently running tasks must be removed or suspended and scheduled to the resources in the grid environment resource reservation is done to meet the user deadlines and to guarantee quality of service. Job monitoring, check pointing and migrating: Job monitoring is responsible for detecting alert situations that could trigger a migration. Check pointing is the capability of capturing periodically a snapshot of the state of a running job, which enables a job to be restarted from that state in a later time in case of migration. Check pointing is beneficial for enabling us to resume a job instead of re-running it after its long execution. The main migration policies considered include performance slowdown, target system failure, job cancellation, detection of a better resource, etc. Reliability: A reliable Grid scheduling system should provide some level of fault tolerance. A Grid is a large collection of loosely coupled resources and therefore it is inevitable that some of the resources may fail due to diverse reasons. The scheduler should handle such frequent resource failures. For example, in case of resource failure, the scheduler should guarantee an application's completion.

Literature Review:

Saha *et al*. (1995) proposed the Fastest Processor to Largest Task First Scheduling Algorithm (FPLTF) is a good representative for Bag-of-Tasks applications. The strategy of the FPLTF scheduling algorithm is to schedule jobs according to the workload of jobs and computing power of resources.

Maheswaran *et al*. (1999) proposed the Min-min scheduling algorithm. In this algorithm, each job will be always assigned to the resource which can complete it earliest in order to spend less time completing all jobs. The Max-min scheduling algorithm is similar to Min-min scheduling algorithm. It gives the highest priority to the job with the maximum earliest completion time.

Maheswaran *et al*. (1999) proposed the On-line mode heuristic scheduling algorithms. Jobs are scheduled as soon as it arrives. Because a grid environment is heterogeneous with different types of resources, on-line mode heuristic scheduling algorithms are more appropriate for grid environment. Dynamic FPLTF Scheduling Algorithm (DFPLTF) is based on FPLTF scheduling algorithm and is modified to make the FPLTF scheduling algorithm more adaptive for grid environment.

Zhang *et al*. (2003) proposed the simple grid simulation architecture and modified the basic ant algorithm for job scheduling in grid. The scheduling algorithm they proposed needs some information such as the number of CPUs, Million Instructions Per Second (MIPS) of every CPU for job scheduling. A resource broker must submit the information mentioned above to the resource monitor.

Wang *et al*. (2005) proposed the Most Fit Task First Scheduling Algorithm (MFTF), mainly attempts to assign the most suitable resource to the task by a value called fitness.

Silberschatz *et al*. (2011) proposed the Batch mode heuristic scheduling algorithms. In that, Jobs are queued and collected into a set when they arrive in the batch mode. They will be scheduled afterwards by the scheduling algorithm. Batch mode heuristic scheduling algorithms are more appropriate for the environment with the same type of resources. First-Come, First- Served Scheduling Algorithm (FCFS) is the simplest algorithm for job scheduling. Jobs are executed according to the sequence of job submitting. The second job will be executed when the first job is done, and therefore FCFS has a serious problem called convoy effect. The convoy effect will happen when there is a job with large workload in the front of the job sequence. All other small workload jobs have to wait until the big one finishes.

Opportunistic Load Balancing (OLB) algorithm assigns each job in random order to the next available machine without considering the job's expected execution time on the machine. Hence, it produces poor makespan. The main aim is to improve resource utilization (Armstrong *et al*., 1998, Freund and Siegel, 1993). The Minimum Execution Time (MET) algorithm assigns each job to the machine that has the minimum expected execution time. It does not consider the availability of the machine and the current load of the machine. The advantages of OLB and MET combined to design new algorithms called as Minimum Completion Time (MCT). The MCT algorithm calculates the completion time for a job on all machines by adding the machine's availability time and the expected execution time of the job on the machine. The algorithm selects the machine with the minimum completion time for executing the job. The MCT considers only one job at a time. The selected machine may have the expected best execution time for any other job (Braun *et al*., 2001).

Min-min algorithm starts with a set of all unmapped tasks. The algorithm calculates the completion time of each job on the each machine. It selects the machine that has the minimum completion time for each job and then selects the job with the overall minimum completion time and allocates to the corresponding machine. Again, this process repeats with the remaining unmapped tasks. Compared to MCT, Min-min algorithm considers all unmapped tasks at a time. Max-min algorithm begins with a set of all unmapped tasks. The algorithm calculates the completion time of each job on the each machine. It selects the machine that has the

Australian Journal of Basic and Applied Sciences, 8(18) December 2014, Pages: 128-135

minimum completion time for each job. The algorithm assigns the job with the overall maximum completion time within the set of minimum completion time to the machine. Again the above process repeats with the remaining unmapped tasks. Similar to Min-min, Max-min also considers all unmapped tasks at a time (Braun *et al*., 2001). The Duplex heuristic is literally a combination of the Min-min and the Max-min heuristic algorithms (Freund and Gherrity, 1998, Armstrong *et al*., 1998).

The previous work (Kamalam and Murali Bhaskaran, 2010a) Min-mean heuristic scheduling algorithm works in two phases. In the first phase, Min-mean heuristic scheduling algorithm starts with a set of all unmapped tasks. The algorithm calculates the completion time for each task on each resource and finds the minimum completion time for each task. From that group, the algorithm selects the task with the overall minimum completion time and allocates to the appropriate resource. Removes the task from the task set. This process repeats until all the tasks get mapped. The algorithm calculates the total completion time of all the resources and the mean completion time (Kamalam and Murali Bhaskaran, 2012a). In phase 2, the mean of all resources completion time is taken. The resource whose completion time is greater than the mean value is selected. The tasks allocated to the selected resources are reallocated to the resources whose completion time is less than the mean value.

The current research problem in task scheduling is to bring out an efficient scheduling algorithm to improve the resource utilization and reduce makespan. The scope of this work is to propose an efficient task scheduling algorithm on the basis of the fitness value of the resource. The RFTSA algorithm efficiently schedules the tasks based on the best fitness value of the resource.

MATERIALS AND METHODS

Problem Formulation:

An application is consisting of 'n' independent tasks and a set of 'm' heterogeneous resources. The problem of mapping the 'n' tasks to the 'm' resources in a gird environment is an NP-Complete problem. The heterogeneous and the dynamic nature of the grid is making the scheduling a complicated problem. In the heterogeneous and dynamic grid environment, the status information of the available resource in the grid is essential for a Grid Scheduler (GS) to make an efficient task-to-resource mapping. The role of Grid information service (GIS) is to provide the status information of the resources to GS. GIS is responsible for collecting the resource information such as CPU capacity, memory size, bandwidth, type of the resource and current load of the resource (Kamalam and Murali Bhaskaran, 2012b).

GS receives the tasks from grid users, selects the feasible resources for these tasks based on acquired information from the GIS and generates task-to-resource mapping based on the objective function, the scheduling criteria and the resource performance.

Grid scheduling is a software framework with which scheduler collects the resource status information from GIS, selects the appropriate resource and based on the objective function of scheduling it determines the best schedule for the task to be executed on a grid system.

Scheduling in Grid represents two important things: ordering and mapping. Ordering refers to the order in which the tasks waiting for the execution is to be arranged. Mapping is the process of selecting an appropriate resource and the allocating the task to that resource. For every mapping, the performance potential is measured in order to decide the best task to resource mapping.

Existing Research:

The previous research work Min-mean and Improved Min-mean provides better scheduling results (Kamalam and Murali Bhaskaran, 2010b)

Algorithm Min-Mean:

The scheduling of the tasks to the available resources is done in two major phases.

In phase 1, the task is allocated to the resources based on the Min-Min algorithm.
In phase 2,
Compute $MeanCT = TotalCT / Total\ No.of\ Resources$
Select resources R_k whose $CTk > MeanCT$
Order the resources R_k in the decreasing order of CT_k.
For each tasks scheduled to the selected resources R_k
Reallocate the job to the resource R_j whose $CTj > MeanCT$
Calculate $CTj = ETij + RTj$
Compute $makespan = \max(CTj), 1 \leq j \leq m$

Proposed Work:

The mapping of meta-tasks to the resources is done based on the following assumptions.

• Heuristics derive a mapping statically.

Australian Journal of Basic and Applied Sciences, 8(18) December 2014, Pages: 128-135

• Each resource executes a single independent task at a time.
• The sizes of the tasks and the number of resources are static and known a priori.
• The accurate estimate of the expected execution time of each task on each resource is represented within an ETC matrix of size n*m, where n-represents the number of tasks and m represents the number of resources.
• Task set is represented as T= $\{T_1, T_2, \dots, T_n\}$.
• Subset of tasks is represented as T_α
• Resource set is represented as R= $\{R_1, .. , R_m\}$.
• ET_{ij}-expected execution time of task T_i on resource R_j.
• TCT_{ij}-expected completion time of task T_i on resource R_j.
• RT_j-ready time of resource R_j.
• $TCTij = ETij + RTj.$
• $makespan = \max(TCT(Ti, Rj))$.
• The ETC matrix ETC (Ti, Rj) is computed by the formula

$$ETC(Ti, Rj) = Lengthi/Powerj$$

where Lengthi is the length of the task T_i in MI and Powerj is the processing power of the resource R_j in MIPS.

• The ready time of the resource R_j is the time at which the resource R_j can complete the execution of all the tasks that have been previously assigned to the resource. The ready time of the resource R_j is

$$RT(R_j) = \sum_{i=1}^{n} ETC(T_i, R_j)$$

• The communication time of each task represents the time taken to transfer the input file and the time taken to transfer the output file to the scheduler where the task is submitted. The communication time is calculated by the formula

$$Comm(Ti, Rj) = IFSi/BWj + OFSi/BWj$$

IFS_i = Input file size of task T_i
OFS_i = Output file size of task R_j
BW_j = Bandwidth of the resource R_j.
• The completion time of each task T_i on each resource R_j is calculated by

$$TCT(Ti, Rj) = ETC(Ti, Rj) + RT(Rj) + Comm(Ti, Rj)$$

• The maximum TCT (T_i, R_j) of all the tasks is the overall completion time of all the tasks and is called the makespan.
• Lifetime of the task T_i is calculated by the formula

$$LT(Ti) = Lengthi/Min(Power) + IFSi/Min(BW)$$

• The time consumed ϴj by resource R_j with respect to task lifetime LT(T_i)is calculated by the formula

$$\theta j = (LT(Ti) - TCT(Ti, Rj))/LT(Ti)$$

The value closer to '1' implies that the corresponding task T_i was performed in a faster way.
• The fitness value of the resource R_j is based on its rate of successfully completed tasks ε Ϲj and on its ϴj value and is calculated by

$$Fj = (a * \epsilon j + b * \theta j)/(a + b)$$

where Ϲj – success rate of the resource Rj.

Both the parameters Ϲj and ϴj have associated a weight value denoted respectively as a and b. The weight values are specified by the users based on the requirements of their application. The value closer to '1' implies that the corresponding resource R_j is considered an efficient one.

Resource Fitness Task Scheduling Algorithm (RFTSA):

The proposed algorithm works in two phases.

Phase 1:

Allocates randomly a subset of tasks T_α to the resource R_j based on Min-Mean algorithm.

Phase 2:

The remaining set of tasks T_i in the task set is scheduled by calculating the fitness value of the resource R_j.
For each unprocessed task T_i in the task set 'T' and for each resource R_j where jϵn
Do
Compute $TCT(Ti, Rj)$ of size 1*n
Compute θj of size 1*n
Compute Fj of size 1*n
Select the resource R_k with the fitness value closer to 1.

Australian Journal of Basic and Applied Sciences, 8(18) December 2014, Pages: 128-135

Schedule the task T_i to the resource R_k
Update $RT(Rk)$
If $(LT(Ti) > TCT(Ti, Rk)$
$Count = Count + 1$
Compute $makespan = \max(TCT(Ti, Rj))$
End.

RESULTS AND DISCUSSION

The makespan of the proposed RFTSA algorithm and existing heuristic algorithm Min-Mean were compared using Braun *et al*. (2001) benchmark. Using the ETC matrix, the instance of this benchmark is divided into 12 different types each of them consisting of 100 instances based on the three metrics: task heterogeneity, machine heterogeneity and consistency. Instances are labeled as u-x-yyzz.k where

- u- uniform distribution used to generate ETC matrix.
- x- Type of consistency(c-consistent, i-inconsistent, s-semi-consistent or partially-consistent).
- An ETC matrix is consistent if a Resource R_j executes any task T_i faster than resource R_k, then Resource R_j executes all tasks faster than Resource R_k.
- An ETC matrix is inconsistent if a Resource R_j is faster than resource R_k for some tasks and slower for other tasks.
- An ETC matrix is semi-consistent or partially consistent if it includes a consistent sub-matrix.
- Task heterogeneity: Variation in the execution time of the task for a given resource.
- yy- the heterogeneity of the tasks (hi- represents high, lo-represents low).
- Resource heterogeneity: Variation in the execution time for a particular task among the entire resource.
- zz- the heterogeneity of the resources (hi- represents high, lo-represents low).

Every instance consists of 512 tasks and 16 resources. The task subset size T_α is fixed as 50 tasks.The experimental results are based on the set of 12 instances, which comprises three groups of four instances each. The first group relates to the consistent ETC matrices of various combinations comprising the resource heterogeneity and task heterogeneity. The second and third group relates to the inconsistent and semi-consistent ETC matrices.

Evaluation Parameters:Makespan:

Makespan is the important optimization criteria for grid scheduling. Makespan is calculated as
$makespan = \max(TCT(Ti, Rj))$

Figure 1 shows that the Resource Fitness Task Scheduling algorithm provides better makespan than Min-Mean Heuristic Scheduling Algorithm.

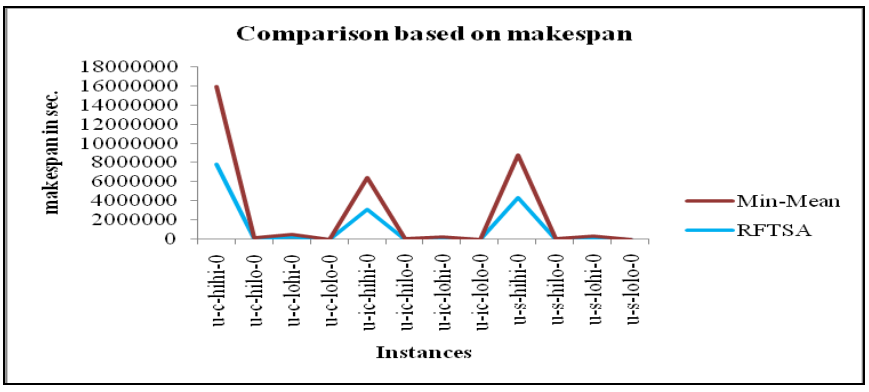

Fig. 1: Comparison based on makespan.

The proposed RFTSA algorithm is tested for different cases. In each case, the task size is fixed at 250 tasks while in every case the size of the random subset of tasks T_α varies form 5 to 25 tasks is shown in Table 1. Figure 2 shows the graphical representation of the makespan values obtained for the five different cases and it is evident from the figure that the proposed algorithm RFTSA provides better makespan than the Min-Mean algorithm.

Australian Journal of Basic and Applied Sciences, 8(18) December 2014, Pages: 128-135

Table 1: Configuration Characteristics.

Cases	T_α	Task Size
Case 1	5	250
Case 2	10	250
Case 3	15	250
Case 4	20	250
Case 5	25	250

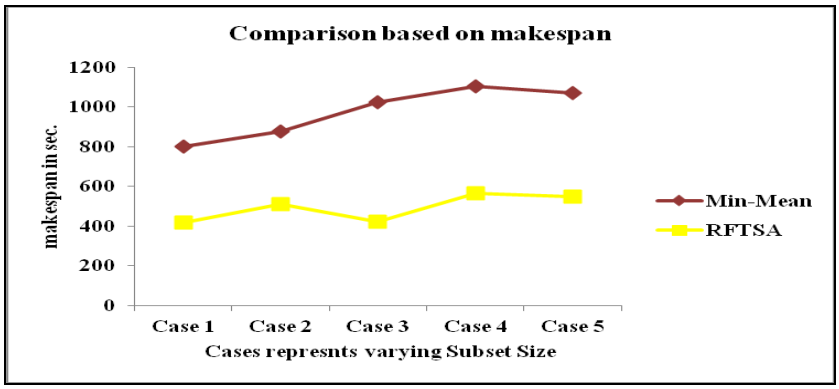

Fig. 2: Comparison based on makespan for different Task Subset Size.

Hit Count:

New QoS parameter is introduced in this work. It represents the number of tasks completed within the lifetime of the task. Figure 3, 4, 5, 6 show the graphical representation of the Hitcount values obtained by Min-mean and RFTSA in all the four instances which comprises High Task High Machine, High Task Low Machine, Low Task High Machine, Low Task Low Machine.

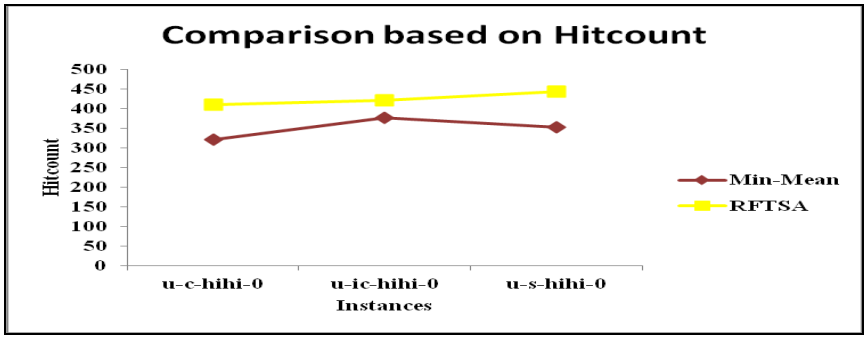

Fig. 3: Comparison based on Hitcount for High Task High Machine Heterogeneity.

The four instances are represented for consistent, inconsistent, semi-consistent or partially consistent heterogeneous computing systems. The simulation result shows that the proposed RFTSA Algorithm shows a high Hitcount than the Min-Mean Heuristic Scheduling Algorithm.

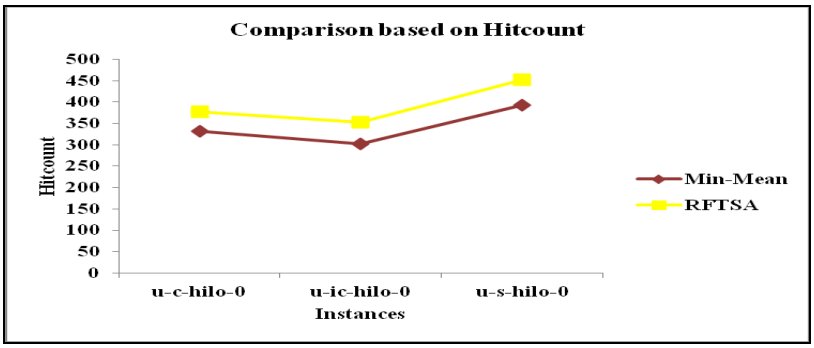

Fig. 4: Comparison based on Hitcount for High Task Low Machine Heterogeneity.

Australian Journal of Basic and Applied Sciences, 8(18) December 2014, Pages: 128-135

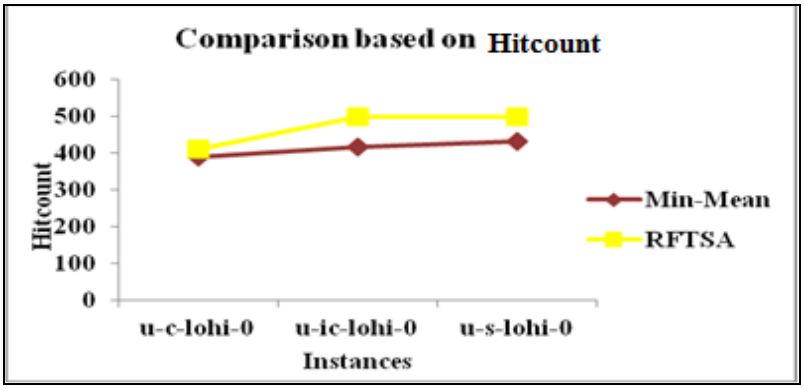

Fig. 5: Comparison based on Hitcount for Low Task High Machine Heterogeneity.

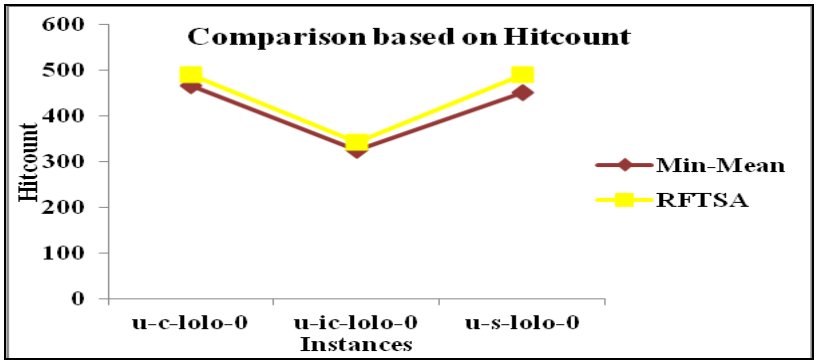

Fig. 6: Comparison based on Hitcount for Low Task Low Machine Heterogeneity.

Conclusion and Future Work:

In this paper, an efficient task scheduling algorithm is proposed. The proposed algorithm RFTSA and the Min-mean heuristic scheduling algorithm are tested using the benchmark simulation model for distributed heterogeneous systems by Braun *et al*. (2001). The proposed algorithm RFTSA considers the dynamic nature of the heterogeneous resources, resource heterogeneity, and task heterogeneity and life time of the task while scheduling the tasks. By considering the life time of the task, the proposed scheduling algorithm effectively schedules the tasks to the best resources. The number of tasks to be completed within the lifetime of the task is increased and the user satisfaction is achieved. From section Results and Discussion it is observed that the proposed algorithm RFTSA achieve a better makespan than the existing Min-Mean heuristic scheduling algorithm. The results show that the proposed RFTSA algorithm reduces the makespan, improves the resource utilization and balances the load across the grid environment. To achieve higher efficiency in scheduling, the proposed algorithm can be improved along with the other scheduling features such as availability of the resources, fault tolerant techniques. The proposed algorithm can be extended to handle data-intensive tasks and to schedule dependent tasks in the future.

REFERENCES

Armstrong, R., D. Hensgen and T. Kidd, 1998. The Relative Performance of Various Mapping Algorithms is Independent of Sizable Variances in Run-time Predictions. In 7th IEEE Heterogeneous Computing Workshop(HCW'98), 79-87.

Braun, T.D., H.J. Siegel and N. Beck, 2001. A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems. Journal of Parallel and Distributed Computing, 61: 810-837.

Chang, R.S., C.Y. Lin and C.F. Lin, 2012. An Adaptive Scoring Job Scheduling algorithm for grid computing. Journal of Information Sciences, 79-89.

Foster, I. and C. Kesselman, 1999. The GRID: Blueprint for a New Computing Infrastructure. Morgan, Kaufmann.

Freund, R.F. and H.J. Siegel, 1993. Heterogeneous Processing. IEEE Computer, 26(6): 13-17.

Freund, R.F. and M. Gherrity, 1998. Scheduling Resources in Multi-user Heterogeneous Computing Environment with Smart Net. In Proceedings of the 7th IEEE HCW.

Kamalam, G.K. and Dr.V. Murali Bhaskaran, 2010a. A New Heuristic Approach:Min-Mean Algorithm For Scheduling Meta-Tasks On Heterogeneous Computing Systems. IJCSNS International Journal of Computer Science and Network Security, 10(1): 24-31.

Kamalam, G.K. and Dr.V. Murali Bhaskaran, 2010 b. An Improved Min-Mean Heuristic Scheduling Algorithm for Mapping Independent Tasks on Heterogeneous Computing Environment. International Journal of Computational Cognition, 8(4): 85-91.

Kamalam, G.K. and Dr.V. Murali Bhaskaran, 2011. An Efficient Hybrid Job Scheduling for Computational Grids. International Journal of Computer Applications.

Kamalam, G.K. and Dr.V. Murali Bhaskaran, 2012a. New Enhanced Heuristic Min-Mean Scheduling Algorithm for Scheduling Meta-Tasks on Heterogeneous Grid Environment. European Journal of Scientific Research, 70(3): 423-430.

Kamalam, G.K. and Dr.V. Murali Bhaskaran, 2012b. Novel Adaptive Job Scheduling algorithm on Heterogeneous Grid Resources. American Journal of applied Sciences, 1294-1299.

Lee, Y.H., S. Leu and R.S. Chang, 2011. Improving job scheduling algorithms in a grid environment. Journal of Future Generation Computer Systems, 991-998.

Maheswaran, M., S. Ali, H.J. Siegel, D. Hensgen and R. Freund, 1999. Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing system. Journal of Parallel and Distributed Computing, pp: 107-131.

Silberschatz, A., P.B. Galvin and G. Gagne, 2011. Operating System Concepts, eighhth ed., John Wiley & Sons.

Saha, D., D. Menasce and S. Porto, 1995. Static and dynamic processor scheduling disciplines in heterogeneous parallel architectures. Journal of Parallel and Distributed Computing, 1-18.

Wang, S.D., I.T. Hsu and Z.Y. Huang, 2005. Dynamic scheduling methods for computational grid environment. International Conference on Parallel and Distributed Systems, 22-28.

Zhang, X. and L. Txang, 2005. CT-ACO – hybridizing ant colony optimization with cycle transfer search for the vehicle routing problem. Congress on Computational Intelligence Methods and Applications, 6.