Efficient Load Balancing Scheduling for Deadline Constrained Tasks on Grid Computing

Ankita Jindal, RK Bansal, Savina Bansal
Deptt of Electronics and Communication Engineering
Giani Zail Singh - Punjab Tech Univ Campus, Bathinda, Punjab, India
ankita1268799, drrakeshkbansal, savina.bansal, @gmail.com

***Abstract* - Grid Computing provides scalable access to wide-area distributed resources. Since, computational grid selects, shares and aggregates wide variety of geographically distributed computing resources and proposes them as a single resource for solving large scale computing applications but there is a need for a scheduling algorithm which takes into account the several requirements of grid environment. Hence, this research proposes a new scheduling algorithm for computational grids that considers user satisfaction, load balancing; fault tolerance based on the resource availability and job characteristics such as user deadline. This algorithm decreases the make-span of the schedule along with user satisfaction means more tasks are successfully completed within user deadline and balanced load means load on all the resources is balanced. The proposed algorithm improves the system parameter such as maximum variation in load; deadline hit count and make-span as compared to BSA.**

***Keywords*: Scheduling, Grid Computing, Load Balancing, User Deadline, Resource Utilization,**

I. INTRODUCTION

Grid computing is a computer network in which each computer resources are shared with every other computer in a system. It is a form of distributed computing that involves coordinating and sharing computation power, data storage and network resources across dynamic and geographically dispersed organizations. In grid computing, the probability of failure is much greater than in traditional parallel computing because the resources are geographically distributed. The failure of resources affects job execution so it is an important property in order to achieve dependability and QoS.

The main aspect of grid computing that is to be considered is the dynamicity of resources. The resources of grid can be freely withdrawn or added at any time according to the owner's discretion. So, the performance of grid nodes and their load frequently changes with respect to time [10]. Grid Scheduling is a process of splitting a larger problem to a number of sub problems and allocating those tasks to the resources based on resource capability and job requirements. The Scheduler plays an important role in computational grids. The selection of proper scheduling algorithm should be of at most care in order to maximize the throughput.

User satisfaction is the major challenge to every field in today's era. In grid computing also, the concept of user deadline is very important. User deadline is defined as the given time from user side to complete the task successfully within that period. The main objective of this paper is to develop a scheduling algorithm which balances the load of each resource and increases the deadline hit count. It also ensures that the resources are utilized in a better way. The proposed strategy improves system performance and reduces make-span. Rest of the paper is organized as follows: Section II contains the description of the related work. In section III, proposed work is described. Section IV provides the experimental results and finally Section V presents the major conclusion

II. RELATED WORK

This section II discusses some of the recent algorithms designed for load balancing; fault tolerance and user satisfaction based scheduling algorithms. A Grid task scheduling framework must be able to deal with issues like security, QoS and lack of central control within distributed administrative domains. He *et al.* [1] in 2003 introduced a novel QoS guided task scheduling algorithm for Grid computing. This novel algorithm is based on a general adaptive scheduling heuristics that includes QoS guidance. The new QoS guided Min-Min heuristic can lead to significant performance gain for a variety of applications.

Buyya et al. [2] described a cost optimization scheduling algorithm to optimize the cost to execute the jobs. It reduces the execution time of the jobs. In this algorithm, failure rate of the resources and user deadline of the jobs are not considered.

For grid environment, Wrzesinska *et al.* [5] in 2006 proposed that there are several reasons for workflow execution failure. The basis are variation in the execution environment configuration, non availability of required software components or services, system running out of memory, overloaded resource conditions, and faults in computational and network fabric components. Grid workflow management systems must be able to identify and handle failures and support reliable execution in the presence of concurrency and failures.

Favarim *et al.* [6] in 2007 introduced a new fault tolerance approach with several masters called brokers. The grid brokers (GB) receive jobs from their users and divide them into tasks. These all subtasks are made available to the resources that compose the grid. Brokers are specific to one class of applications and they only know how to decompose jobs of this class. e.g., if the application deals with processing satellite images, the images are treated as jobs & the broker

decomposes the job into several tasks and analyzes them by various resources. After executing a task, the resources send the result to the broker and the broker assembles all results and returns them to the user.

A minimum time to release job scheduling algorithm is proposed by Malarvizhi *et al.* [8] in 2009 in which time to release (TTR) is evaluated. The tasks are arranged in descending order based on Time to Release (TTR) value. Tasks are scheduled in the sorted order. This algorithm carries out better when compared with First-Come-First-Serve and Min-Min algorithms.

The survey done by Christopher [7] in 2009 shows that both the commercial grid systems and research grid systems that are currently in use behave reliably at present levels of scale using accessible technology. However, efforts to develop reliable and fault tolerant methods for grid environments are in progress with increased scale, dynamism and heterogeneity.

A prioritized user demand algorithm is proposed by Suresh *et al.* [3] in 2011 that considers user deadline for allocating jobs to different heterogeneous resources from different administrative domains. It produces more user satisfaction and better make-span but data requirement is not examined. While scheduling the jobs, failure rate is not examined. So the scheduled jobs may be failed during execution.

Keerthika *et al.* [4] in 2013 proposed an efficient fault tolerant scheduling algorithm (FTMM) which is based on failure rate and data transfer time. System performance is also achieved by reducing the idle time of the resources and distributing the unmapped tasks equally among the available resources.

Keerthika *et al.* [9] in 2013 proposed a new algorithm BSA (Bicriteria Scheduling Algorithm).The main objective of this algorithm is to reduce make-span and idle time of the resources. It ensures that the tasks are completed within the user expected deadline. While scheduling the jobs, (FR) failure rate of the resources is also examined. The proposed algorithm improves system performance and user satisfaction and reduces number of failures of the jobs by considering fault rate of the resources.

The main objective of this paper is to develop a scheduling algorithm which reduces make-span and improves resource utilization. It also ensures that the tasks are completed within the user expected deadline. While scheduling the jobs, the load of each resource is also examined. The proposed algorithm improves system performance, user satisfaction and resource utilization.

III. Proposed Work

Load balancing is one of the primary challenges of existing and future grid based applications and services. A load is the number of tasks in the waiting queue and can be heavy, moderate and light according to their work. Load balancing is a process of improving the performance of computational grid system in such a way that all computing node involve in the grid utilized equally as much as possible. Load balancing is an important function of grid system to distribute the workload among available computing nodes to minimize execution times, maximize node utilization and overall system performance.

For each task submitted with its user deadline
{
Phase 1(Matrix construction):
Construct ETC(Ti, Rj) matrix of size (m×n);
Construct RT(Rj) matrix of size (1×n);
Construct CT(Ti, Rj) matrix of size (m×n);
Construct CMT(Ti, Rj) matrix of size (m×n);
Construct TCT(Ti, Rj) matrix of size (m×n);
Phase 2(Resource selection process):
Calculate optimal value of resource;
Calculate resource load value;
Select resource with min TCT which meets user deadline and min RLV;
Phase 3(Scheduling and Execution):
Schedule the task on selected resource;
Update RLV of resource to which task is scheduled;
If failure occurs during execution, update failure rate of resource Rj;
Compute maximum variation in load, make-span and deadline hit count;
}

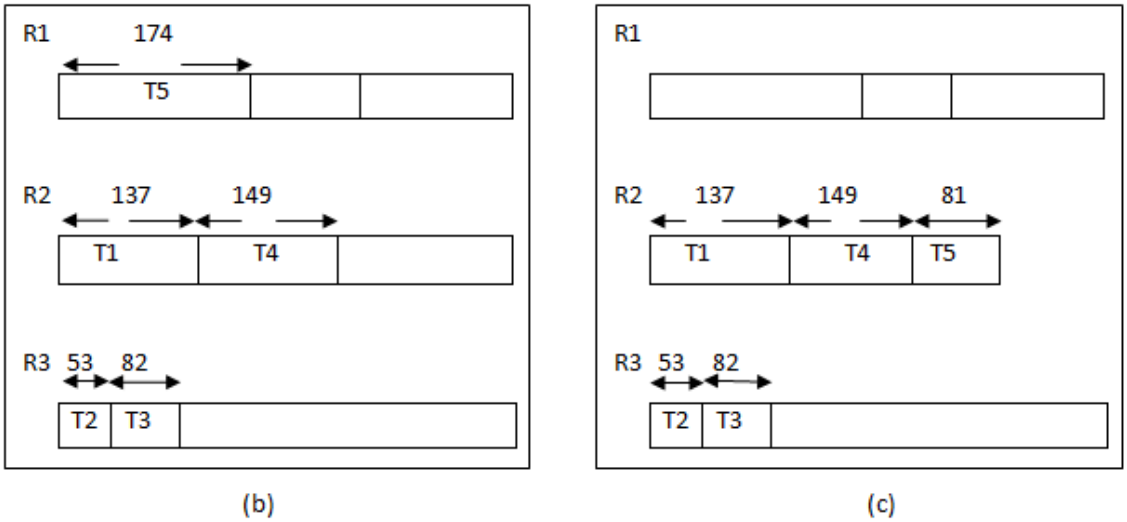
Fig. 1 Proposed Technique

In proposed technique, tasks are submitted by the user with its user deadline. In first phase, matrix construction process by following the procedure as shown in Fig. 1. Total completion time is the basic term which a system needs for scheduling so matrix construction of total completion time (TCT) requires construction of some terms such as ETC(expected time to compute), RT(ready time), CT(completion time) and CMT(communication time). At first ETC matrix of size (m×n) where m represents the total number of tasks submitted by the user with its user deadline and n represents the total number of resources. Expected time to compute is the length of task to the capacity of resource. Now matrix of ready time is constructed of size (1×n) for each resource. The ready time of resource j is the earliest time in which resource Rj can complete the execution of all tasks that have previously been assigned to it. After constructing RT matrix, CT (completion time) and CMT (communication time) matrix is constructed. The completion time for a task on resource is the sum of the machine availability time for Rj and the execution time of task on resource. Communication time is the total time taken for transfer of input file and transfer of output file. Total completion time (TCT) of each task on each resource is addition of the completion time and the communication time.

In second phase, optimum value (OV) and resource load value (RLV) of resources are calculated for selection of resource as shown in Fig. 1. Select resource with minimum TCT and minimum RLV for scheduling of a task.

Resource / Task	R1 TCT (in sec)	R2 TCT (in sec)	R3 TCT (in sec)	User deadline (in sec)
T1	177	137	169	490
T2	151	182	53	139
T3	118	123	82	151
T4	164	149	144	293
T5	174	81	153	206

(a)



(b) (c)

Fig. 2 Comparison of techniques (b) Proposed algorithm (c) BSA with implementation for (a) given example

```
For each task  i submitted by the user
{
  For each resource j in resource list
  {
    Rj→(Rj with mTCT) – Er ;
    If
    {
      RLV <= OV ;
      Schedule (Ti, Rj) ;
      else
      Er→ Er + Rj ;
    }
  }
}
```

Fig. 3 Resource selection process

If resource load value is less than or equal to optimum value then priority is given to resource load value otherwise add resource to excluded resources. The excluded resources are those resources which already has been reached their optimum value. According to the strategy, schedule the task on the suitable resource as described above in resource selection process. After scheduling, update resource load value of resource to which the task is scheduled. Likewise all the tasks are scheduled followed by the above process. During execution, if failure occurs or task is not successfully executed then update the failure rate of resource and get the new task from list. If execution successfully completed the task then increase the hit count. Now we will check that the task is completed within the user deadline or not. If it is completed within the user deadline then increase the deadline hit count otherwise not. This procedure should be repeated until when task list is not empty. At the end, system parameters such as maximum variation in load, makespan and deadline hit count are calculated. With the help of case example, the implementation of allocation of resources according to the total completion time matrix for both algorithms is shown in Fig.2.

Fig. 2(a) shows the TCT matrix and user deadline for different tasks. Both techniques can be understood with the help of implementation and the performance of proposed technique is improved than BSA.

IV. Results

A simulation is conducted in heterogeneous environment using custom java script simulator where each resource has machines with different characteristics such as processing speed, bandwidth and memory size. The parameters such as number of tasks successfully completed within user deadline, resource load value and number of task executed are taken into account to calculate the deadline hit count, make-span and maximum variation in load. The simulation is done to improve the performance of BSA.

TABLE 1
SIMULATION PARAMETERS

Parameters	Values
Number of tasks	100-500
Number of resources	5-15
Resource capacity	5-50 MIPS
Task length	50,000-1,00,000 MI
I/P file size	50-500 MB
O/P file size	100-700 MB

TABLE 2
SIMULATION CASES

Cases	(Tasks, Resources)
Case 1	(100, 5)
Case 2	(200, 8)
Case 3	(300, 10)
Case 4	(400, 12)
Case 5	(500, 15)

A. *Maximum variation in load*

Maximum variation in load is a new metric introduced in this work. It is defined as the difference between the maximum values to the minimum value of load of the resources.

1) *Formula to calculate Optimum Load Value*

Optimum Value (OV) = (1/number of resources)*100

2) *Formula to calculate Resource Load Value*

RLV = (number of task executed/total number of task)*100

Fig.4 shows the maximum variation in load comparison of the proposed system and the existing algorithm. In proposed algorithm, load is balanced so that maximum variation in load is less as compared to BSA. Resources are better utilized in proposed algorithm due to less variation in load.

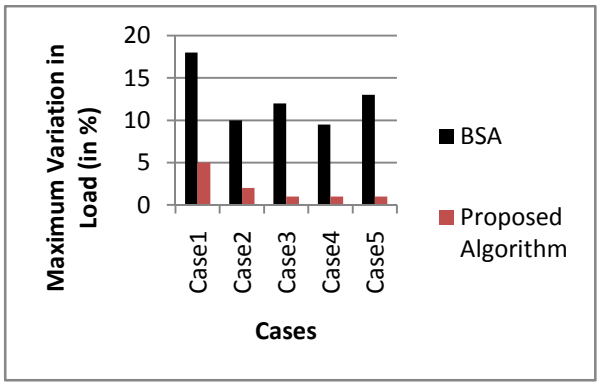


Fig. 4 BSA Vs Proposed algorithm

B. *Deadline hit count*

It is defined as the number of tasks successfully completed within user deadline. It is a measure of user satisfaction which is the basic requirement in grid systems. Fig5 shows the deadline hit count for proposed algorithm and existing algorithm. In proposed algorithm deadline hit count means tasks completed within user deadline is more as compared to BSA because of load balancing.

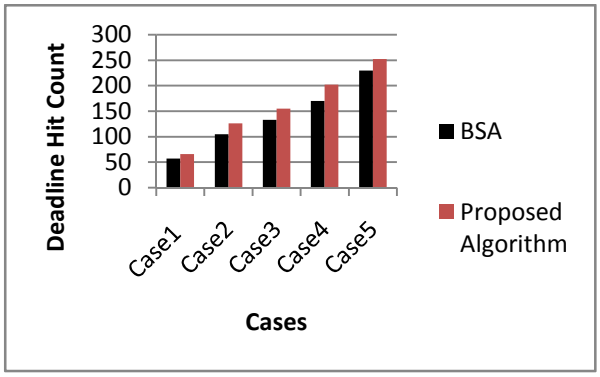


Fig. 5 BSA Vs Proposed algorithm

C. *Make-span*

The make-span is the total length of the schedule i.e. when all the jobs have finished processing. Make-span is one of the most important standard metric of grid scheduling to measure its performance. In proposed algorithm, make-spam is reduced as compared to BSA because waiting time for tasks is reduced due to balanced load on resources. Fig.6 shows the make-span of proposed algorithm and BSA.

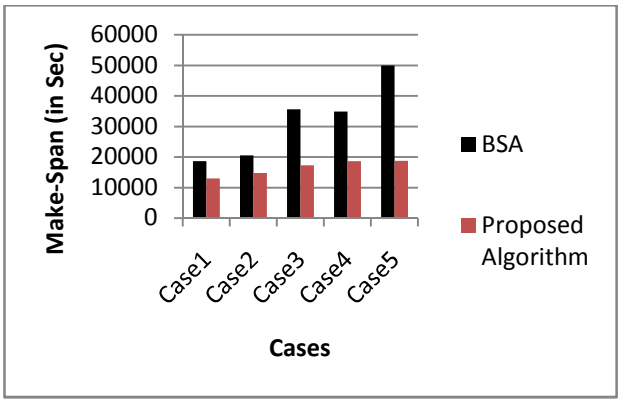


Fig. 6 BSA Vs Proposed algorithm

V. CONCLUSION

The Proposed Algorithm considers both Load Balancing and User deadline while scheduling the jobs. Experiments have been done for make span that serves as a parameter for calculating the efficiency of the algorithm and deadline hit count which is the major parameter for user satisfaction. It is observed that the adoption of load balancing achieves the improved performance than the existing algorithm (BSA). The proposed algorithm considers user deadline of each task and the load of each resource at the time of scheduling which is important in grid environment.

REFERENCES

[1] X. S. He, X. H. Sun, and G. von Laszewski, "QoS guided Min- Min heuristic for grid task scheduling," Journal of Computer Science and Technology, vol. 18, no. 4, pp. 442–451, July 2003

[2] R.Buyya, M.Murshed, and D.Abramson, "A dead line and budget constrained cost-time optimization algorithm for scheduling task farming applications on global grids," in Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, pp.24-27, 2002

[3] P.Suresh, P.Balasubramanie, and P.Keerthika, "Prioritized user demand approach for scheduling meta tasks on heterogeneous grid environment," International Journal of Computer Applications, vol. 23, no. 1, 2011

[4] P. Keerthika and N. Kasthuri, "An efficient fault tolerant scheduling approach for computational grid," American Journal of Applied Sciences, vol. 9, no. 12, pp. 2046–2051, April 2013

[5] G. Wrzesi´nska, R. V. van Nieuwpoort, J. Maassen, T. Kielmann, and H.E.Bal, "Fault-tolerant scheduling of fine-grained tasks in grid environments," International Journal of High Performance Computing Applications, vol. 20, no. 1, pp. 103–114, 2006

6] F. Favarim, J. da Silva Fraga, L. C. Lung, and M. Correia, "GRIDTS: A new approach for fault-tolerant scheduling in grid computing," Proceedings of the 6th IEEE International Symposium on Network Computing and Applications, pp. 187–194, July 2007

[7] D. Christopher, "Reliability in grid computing systems," Concurrency and Computation: Practice and Experience, vol. 21, no. 8, pp. 927–959, 2009

[8] V. Modiri, M. Analoui, and S. Jabbehdari, "Fault tolerance in grid using ant colony optimization and directed acyclic graph," International Journal of Grid Computing & Applications, vol. 2, no. 1, 2011

[9] P. Keerthika and N. Kasthuri, "An efficient Grid Scheduling with Fault Tolerance and User Satisfaction," Mathematical problems in engineering, Article ID 340294, April 2013

10] A. Torkestani, "A new approach to the job scheduling problem in computational grids," Cluster Computing, DOI 10.1007/s10586-011-0192-5, vol. 15, no. 3, pp. 201-210, 2012