AN IMPROVED RESCHEDULING AND FAULT TOLERANCE WITH TCSA FOR PERFORMANCE IMPROVEMENT IN COMPUTATIONAL GRID

[1]S GOKULDEV, [2]ANAGHA P, [3]SUJANA SAGAR S J, [4]AKSHATHA TANTRY H
[1,2,3,4] Department of Computer Science, Amrita Vishwa Vidyapeetham,
Mysuru, Karnataka, India
E-mail: [1]gokuldevs@gmail.com, [2]anaghapramesh6@gmail.com , [3]sujanasagar.j68@gmail.com, [4]akshathatantry19@gmail.com

ABSTRACT

Fault tolerance is one of the major challenges during fault occurrence in any computational environment. Since, computational grids provide access to huge pool of shared processing power, the chances of fault occurrence are high. The proposed system is focused on performing job scheduling using Two Choices Scheduling Algorithm (TCSA) with tolerating the faults that occur during the process. Job allocation is done with the help of grid system consisting of resource broker and GIS. Once, the faulty resources are identified, scheduler analyses the load of remaining idle resources and reschedules the faulty jobs to the new set of resources which currently has the least load. A comparison is done between Round Robin algorithm and TCSA before and after incorporating fault tolerance. It is observed that the time taken for the completion of allocated tasks has been minimized after the implementation of improvised TCSA incorporating fault tolerance. The work focuses on tolerating the faults that are occurred during task allocation and performs rescheduling by reallocating the tasks to different resources which helped the system to obtain significant results while the system is exposed to high loads.

Keywords: *Average Utilization, Computational grid, Fault Tolerance, Improvised TCSA, Rescheduling*

1. INTRODUCTION

Computational grid is a network of nodes in which a particular task is assigned among individual resources, which runs calculations in parallel and returns the results. The grid system contributes in coordinating, sharing, computing, network resources across organizations that are dynamic in nature and distributed geographically. A grid technology tackles complex computational issues. In grid computing, the job is controlled by a master node which acts as a resource broker. The job is equally divided in to multiple tasks and it will be distributed to all other computing elements for processing. Grid environment is extremely heterogeneous and dynamic because of the reason that its components are joining and leaving the system all the time. So possibility of occurring faults is high. Therefore, it is essential to perform fault identification and tolerance mechanism among the resources in grid. The proposed work mainly focuses on identifying and tolerating the faults that occurs during job scheduling. Fault tolerance can be defined as the property which helps the system to carry out its operation efficiently even if faults occur in its components. Figure 1 depicts the overall working mechanism of the system.

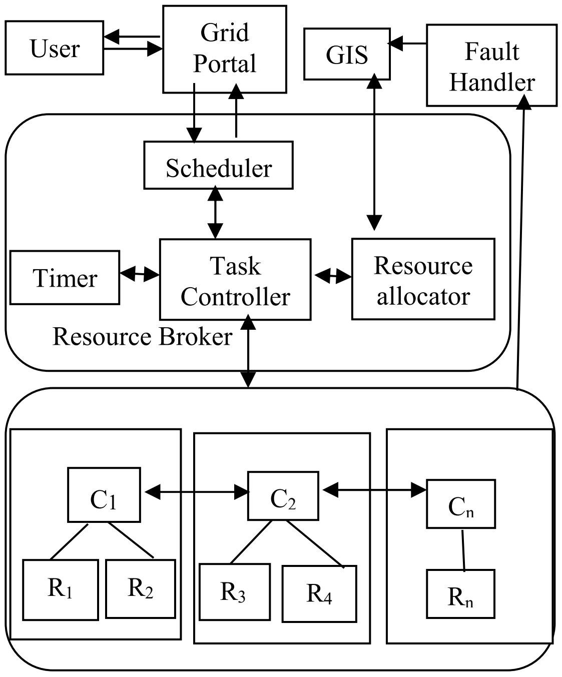


Figure 1: System Model

In Figure 1, the user input the jobs to the scheduler through grid portal. The TCSA algorithm gets invoked within the resource broker at once, when the scheduler receives the input jobs. The coordination between resources and resource broker is established by Grid Information Server (GIS) and fault handler [12]. The fault handler identifies the faulty resources in grid. GIS also enables frequent updation of the status of the resources. The copy of the status will be send to resource allocator.

Once the user submits jobs to the scheduler, it splits the job into small tasks and send to task controller. The task distribution among available idle resources is coordinated by the task controller with the help of resource allocator. Task controller takes the information regarding the idle resources with the help of resource allocator from GIS and allocates these tasks to the resources in the order of their arrival. The TCSA [2] allocate tasks to resources in a dynamic manner for reducing the time of task execution and maximizing the utilization of resources.

The resource allocator contains all information and updated status regarding grid resources sent by GIS. The function of the timer is to allocate the time to each task assigned to grid resources. When the assigned task is finished, timer will calculate the time taken for the completion of that particular task and pass the results back to the scheduler. Once any fault is identified among any one of the assigned resources, scheduler analyses the load of remaining available resources and rescheduling is performed by reallocating the task of faulty resource to the resource that have the least load. Timer keeps identifying the faulty instance along with the maintainance of execution time.

The organization of the work is as follows: Section 1 deals with the system model along with the problem statement. Existing works about fault tolerance methods in computational grids, job scheduling with resource broker are discussed in section 2. Methodology is described in Section 3 and Section 4, brings out the experimental results. Finally, conclusion of paper is given by section 5.

2. RELATED WORK

Many researches have been performed from the former period of time to study the problems in identification and tolerance of faulty resources in grid.

2.1. Review on Job Scheduling

Aparna et al. (2013) implemented an Adaptive Scoring Job Scheduling (ASJS) [3] algorithm where user has to mention whether the job is concentrated more on computations or data. Because of this issue, the jobs are not been completed on time. To overcome this problem, an Enhanced Adaptive Scoring Job Scheduling algorithm (EASJS) is introduced with a method of duplication strategy. In the above work, cluster score is considered which can be enhanced by using other properties of clusters such as latency and bandwidth.

Maryam et al. (2014) proposed an Improvement Hierarchy Load Balancing (IHLBA) algorithm [5] and it has been compared with other four scheduling algorithm called On-line Mode Heuristic Scheduling Algorithm, Most Fit Job Scheduling Algorithm (MFTF), Dynamic Load Balancing Algorithm (DLBA) and the Hierarchical Load Balancing Algorithm (HLBA) based on correlation of load along with its hierarchical structure were explored. In future, these algorithms can be enhanced by testing them on heterogeneous processors and heterogeneous computing environment.

Ying L et al. (2015) described that the number of tails sampled can be minimized highly with the help of the fact that the tasks come as batches (called jobs). The authors focused more on taking the samples from a subset of queues such that the size of the subset is little larger than the size of the batches. A subset of the tails is sampled at random. The proposed method of load balancing called batch filling [8] tried for equalizing the load between the sampled servers. The algorithm drastically minimizes the sample complexity when compared to other algorithms. In future, mechanisms that deals with the faults during load balancing can be incorporated with the batch filling method.

AlEbrahim et al. (2016) proposed a scheduling algorithm [1] which dispatches the tasks and minimizes the total time of execution of the interaction between the processors. This is attained in two main steps: (a) the computational priority of all tasks will be analyzed individually and (b) the processor which processes each task will be chosen. The future work focuses on considering task duplication in the part of processor selection to further reduce the makespan by making the interaction of processors low.

Attiya I et al. (2016) introduced a fruitful algorithm called Two Choices Scheduling Algorithm (TCSA) [2] to allocate jobs in a dynamic manner to resources for reducing the job execution time and for maximizing the utilization of resources there by performing the user's work successfully and make use of the distributed resources. The data transmission rate and total time of execution during the task scheduling can be explored as future enhancement.

Manishankar S et al. (2016) introduced a security platform [29] along with the security-approaches for obtaining high-performance computing clusters. Scheduling is performed implementing ALPHA scheduler approach. In future, incorporating fault tolerance mechanism along with security features may help in increasing the efficiency.

Bharathi P. D et al. (2017) proposed a scheduling model [4] for the cloud in two-level. The load is equalized for the utilization of resources in an effective manner and efficiency of energy is attained that contributes to the green cloud computing. The Quality of Service (QoS) with a minimized Service Level Agreement (SLA) negligence is also attained. Future work could focus on the possibility of occurring faults during the load balancing and its solutions.

Banerjee S et al. (2017) used a computer approach, which is decentralized, [6] in order to assign and arrange tasks in grid which is dispersed on a vast scale. With the help of multi-agent systems' properties, this approach generates and disassociates the clusters in a dynamic way, in order to meet the global task queue resource demands which are continuously changing. Comparison of algorithm is performed with the standard FIFO. Experiments performed shows that the Distributed Resource Allocation Protocol (DRAP) algorithm overcomes the FIFO in terms of empty queue and CPU utilization. The future work may should be designed to improve version of DRAP for attaining more CPU utilization and minimized execution time.

Suresh S et al. (2017) presented a suitable formulation for a wireless network with effective rate [7] which have the ability of transmitting sensor data responsive to time through the network of transmission line along with obstruction and bandwidth restrictions. In future, steps to overcome the faults that occur during the sensor data transmission can be incorporated with the formulation.

2.2. Review on Load Balancing

Sarmila G P et al. (2015) compared the efficiency of load-balancing algorithms [17] that are fault-tolerant. If any algorithm is weak in any of the parameters such as throughput, response time etc. then it points out that a fault-tolerant load-balancing algorithm which is efficient and autonomic in nature have to be made. The scope is to design an algorithm with efficiency that can provide solutions for the mentioned limitations.

Oueis J et al. (2015) proposed an advanced algorithm [15] developed for formulating the clusters and balancing of load in fog computing. It has two benefits. First, its design is modifiable. Second, it has an optimization method of minimized complexity multi-parameter. Fault tolerant mechanisms can be incorporated with the three variants of this algorithm such as EDF-PC and EDF-LAT which helps for attaining more efficiency as future improvement.

Dave et al. (2016) provided a detailed abstract of the optimization methods of load balancing [11] based on evolution and group algorithms, which permits to overcome the problems of optimization or utilization of resources. The main problems facing the cloud are discovery of resource, load balancing and security. Load balancing is one of the major test to be dealt with along with the key issues and the average role of assigning workloads or tasks on nodes or servers. Further studies can be done on the issues of discovery of resource and fault tolerance in order to examine its suitable solution techniques.

Rahul Singh Chawhan et al. (2016) introduced a mobile agent technology [10] that acts like an alternative for equalizing the load and arranging it in dispersed systems. The protocol of load balancing leverages the Java Complaints Mobile Agent platform named AgletsTM. It allows the development and implementation of mobile agents. In the future enhancements, studies can be performed on the alternative methods for attaining the maximum utilization of existing cluster resources.

Parwekar P et al. (2016) implemented a routing algorithm [13] based on fuzzy texture of network which is reliable for wireless network of sensors. The RSSI, LQI and the number of jumps to destination are used for the estimation of the route from the source to the destination. The simulation results in huge delivery of packets along with the network reliability. In future, an improved version of this algorithm has all the chances to attain more efficiency and reliability.

Chauhan A et al. (2016) presented a novel improvised scheduling algorithm (IDSA) [9] with time limit. Equalizing the full load of the system model, along with making an attempt to reduce the make span of a set of available tasks, was the major result of this work. In future, this algorithm can be implemented for the resources having computational latency. Comparison can also be done with other scheduling algorithms.

Manishankar S et al. (2016) implemented an algorithm [28] to equalize the load that is dynamically efficient. Based on various aspects of cloud environment, the load is distributed and best optimal cloud partition is obtained for the job completion. It also helps in reducing the cost of processing. In future, focus shall be more on performance issues.

Jia Zhao et al. (2016) introduced a task implementation approach, Load Balancing based on Bayes and Clustering (LB-BC), [14] for the effect of load equalization for a long period of time and has used an idea based on Bayes' theorem. The simulation experiments demonstrated that the system is able to implement speedy tasks in a fast manner in data centers of cloud. The data centers attain a load balance for a long period of time. Future work may include the LB-BC in a real cloud computing environment. The performance and efficiency can be evaluated.

Das N. K et al. (2017), incorporated the weighted Round Robin algorithm with Honeybee algorithm [12] in order to attain a reduced cost of processing and response cost. For the prioritized tasks, Honeybee Inspired Algorithm is used by allocating weights to virtual machine individually. The non-prioritized tasks were made to run using the Weighted Round Robin algorithm. In the future, more QoS parameters like waiting time and rate can be taken into consideration.

Khan Z et al. (2017) has proposed a new architecture [16] for load balancing. The results indicate that the combination based job scheduling algorithm, which is dynamic in nature, effectively plan tasks in dispersed systems, equalizes the load and minimizes the execution time, which helps in high performance. The analysis in the performance indicates that this algorithm exceeds FCFS and ACO performance. In future, the mechanism for dealing with the faults that occur during the dynamic job allocation may be introduced.

Manishankar S et al. (2017) introduced several improvements [27] stating that, networks, that are energy-efficient, are always a necessary requirement of communication model in this competitive world along with the technical advancement. Wi-Fi is a simple example which can cover only a certain distance. The only solution for this is the improvement of Wi-Fi with the help of QoS parameters. But, there are some elements that can't be compromised with this. In future, more efficient network technologies can be studied and analyzed along with the associated QoS parameters like loss of packets and jitter.

2.3. Review on Fault Identification and Fault Tolerance

Cirne et al. (2003) conducted a survey on the faults in grid and came to the conclusion that the grid users are not pleased with the situation of occurring faults [26]. There are two basic difficulties in the system that are identified in the management of faults in grid. First is the available solution for the analysis of faults. Second, among the fault tolerance plans that are put into effect, grids only permit crash failures. As grids are susceptible to more composite faults, like those generated by heisenbugs, a person should be able to bear more severe faults. A detailed survey about these failures can bring out more clarifications in future.

Narale et al. (2011) focused on factual time system and the available techniques of fault tolerance [22] used in the domain of cloud computing along with its implementation. The authors also discussed an additional significant issue, fault tolerance model and procedure model. The future scope of this study is the implementation of these fault tolerance techniques to obtain different measurements of robust cloud. Also using these algorithms, how various problems and challenges of the cloud computing can be solved.

Nasir et al. (2015) performs testing of Partial Key Grouping (PKG) [19] on several large datasets by studying the load balancing problem in the distributed stream processing engine and classifies the key grouping PKG into the distributed flow settings by using two novel techniques. One, classifies the packets into two categories and second divides the process into two options, namely, grouping and assessment of local load. It attains a load balancing which is finer than the key grouping. Future work can be done in finding whether it is considerable or not to think about attaining good load balance without the predefined atomicity of key processing.

Hannache O et al. (2015) implemented a proactive approach of fault tolerance [24] based on the preventive migration of virtual machines. The research had a main objective of modelling an effective cloud environment in which the fault tolerance approach was evaluated. The system becomes more dynamic in nature with the use of simulated Lm sensor. The results clearly indicated that the cloud availability can be supported using this approach. Introduction of fault recovery to this approach may make it more efficient in future.

Lee J S et al. (2015) proposed a Three-Level Switching Oriented (3LSO) tolerance control [25], which is beneficial with respect to the efficiency of Sx2 and Sx3 open breaker failures of type T rectifier. In particular, the intermediate circuit wavelet of voltage and the switching losses of the above mentioned controls are determined through the use of simulation and results. Since the above mentioned controls have advantages that are different from each other, it need to be selected depending upon the specific requirements of applications. In future, a mechanism that reduces the execution time can be introduced to this approach.

Kaur P et al. (2015) proposed a method for enhancing the fault tolerance [21] of the most suitable algorithm of job scheduling, by arranging the work along with work duplication when the reliability of the resource is low. The results show that, execution time of the tasks is reduced comparatively with this algorithm using the real-time approach instead of simulator. The future scope would be to test the algorithm for other faults.

Indhumathi et al. (2016) designed a new architecture to reduce the implementation time in computing grid. The Load Balancing Fault Tolerance (LBFT) architecture along with the use of SOA, [18] focused on a new algorithm which is dynamic and complementary in nature, that helps in attaining load balancing and fault tolerance successfully. In future, the algorithm can be incorporated with other grid simulators like SimGrid, which helps in determining the efficiency of algorithm in obtainable simulators.

J Pinto et al. (2016) described the Hadoop architecture [20], it is needed to predict node failure at a fairly initial phase so that rescheduling work is not expensive when time and efficiency are considered as parameters. The final results indicated that the existence of a fault is predictable with the help of existing acquired knowledge with a minimum delay of time. Future enhancements can be done by replacing neural networks in place of Support Vector Machine (SVM) that have good accuracy in classification, when computational complexity is considered.

Goyal N et al. (2018) proposed a Fault Detection and Recovery Technique (FDRT) [23] for a network, based on clusters. A Backup Cluster Head (BCH) is selected, while selecting Cluster Head (CH), using the fuzzy logic technique based on parameters such as residual energy, load, node density, distance to the sink and link quality. In the future, work could be done to further improve the delivery relationship of FDRT with a greater number of faults.

Most of the existing systems provide vast number of ways for fault identification occurring within the grid environment. Among the available fault tolerance methods, the existing system lacks in a fault tolerance method that can be performed with reduced makespan.

3. METHODOLOGY

The system proposes a new method of fault tolerance with the help of TCSA algorithm. The primary step is resource allocation which helps to reduce the makespan and load balancing among resources effectively. Job allocation is done dynamically along with minimized job execution time and maximized resource utilization time. During the job allocation, if any faulty resources are identified, then the tasks allocated to those faulty resources are reassigned to some other resources with the least load.

3.1. Round Robin Algorithm

Round robin algorithm is a preemptive algorithm designed for time-sharing system. In this algorithm, a predefined quantum is maintained for preemption. This can be defined as the fixed time allocated to all the tasks for execution. If the task is not completed within the given quantum, it will get interrupted. The task is resumed next time a time slot is assigned to that process. If, during the attributed time quantum of process, if its state is changed to waiting, then the scheduler selects the first process in the ready queue to execute. The states of preempted tasks are stored using the mechanism of context switching. In most of the grid scenarios, the round robin algorithm has shown a better makespan and hence this algorithm is compared with that of the proposed system considering the fault tolerance.

3.2. Implementation of TCSA Algorithm with Fault Tolerance.

According to TCSA algorithm which incorporates fault tolerance, there exists an independent user task set, $T = \{T_1, T_2 \ldots T_n\}$ to a set of heterogeneous grid nodes $N = \{N_1, N_2 \ldots N_m\}$. Makespan, load balancing and flow time are the three popular criteria used to evaluate the job scheduling effectiveness and efficiency. Improvised TCSA, using load balancing and makespan, optimizes the process of task scheduling in diversely scattered computing systems.

The term makespan can be used as a metrics in order to measure the excellence of a scheduling process in grid computing environment. Makespan is the finishing time of the last task assigned. The finishing time of the schedule is analyzed with the help of grid node which have the maximum processing cost for all tasks allocated to it. So, the formulation of makespan is as follows:

$$\mathbf{Time_i = \sum_{(j|A_j \in i)} E_{ij} + W_i} \qquad (1)$$

$$\mathbf{makespan=max\{Time_i\}} \qquad (2)$$

where 'Time$_i$' is the duration of time needed for each node 'i' to finish all the tasks allocated to it; '{j|A$_j$∈i}' indicates the tasks allocated to the node 'i'; 'E$_{ij}$' is the time duration taken by 'N$_i$' to finish the task 'T$_j$'; i ∈ 1,2 … , m; j ∈ 1,2,… , n, and 'W$_i$' is the waiting time of task 'j' till the resource 'i' gets ready.

The individual utilization of resources and average resource utilization of the improvised TCSA is provided as follows:

$$\mathbf{U_i = Time_i/makespan} \qquad (3)$$

$$\mathbf{Avg\text{-}utilization = (\sum_{i=1}^{m} U_i)/m} \qquad (4)$$

where 'U$_i$' indicates the utilization of resource 'i'; 'Avg-utilization' indicates the average utilization of resources. 'm' represents the total number of existing resources.

Algorithm: TCSA with fault tolerance.
Input: Tasks and resources.
Output: Makespan and resource utilization.

```
Initialize the resource-list [No. of resources]
Initialize the task-list [No. of tasks]
while task-list is not empty do
task 'T' from front of the task-list is taken
for each task 'T' do
d random nodes are selected uniformly.
  $N_1(T), N_2(T)… N_d(T)$
Query nodes $N_1(T), N_2(T)…N_d(T)$ for current load
Allocate task T to the node with the less load
if there is a tie for the least loaded node
then
Allocate task T to randomly chosen among them
end if
Fault-list is initialized [No. of faulty resources]
Allocated task 'T' is taken from faulty resource.
if idle resources available
Query nodes for current load
Reallocate task T to the node with less load
if there is a tie for the least loaded node
then
Reallocate task T to randomly chosen among them
end if
else
Wait until idle resources with least load are available.
end if
end for
end while
```

When the user submits the job, the primary step done by the scheduler is splitting the job in to small tasks. Task controller will be having a task-list from where, each task, which is ready for execution, is taken from front of the list. The resource allocator will be maintaining a resource-list from where 'd' grid nodes are randomly selected from available nodes. Scheduler sends a query message to each of the 'd' grid nodes. As a result, scheduler gets the current load information of each of the d nodes. Finally, the scheduler allocates the task to the least loaded of d randomly selected nodes.

During the process of task allocation, there will be situations where the resources will not be able to execute the tasks that are assigned to it. This can be because of the reason that the assigned resources are already loaded with tasks. In such a case, the occurred fault prevents the scheduler from proper task scheduling. So, task allocator identifies the faulty resources and a list of faulty resources is made with the help of fault handler. Scheduler checks for the idle resource in resource list. Once the resource is available, the task is taken from the faulty resource of the fault-list and fault tolerance is performed by reallocating that particular task to the available resource with the least load. This helps in proper scheduling and all the tasks are executed successfully with minimum execution time.

4. RESULTS AND ANALYSIS

In order to compute the execution time, a set of observations are performed using number of resources and number of tasks. From the obtained output, it is observed that the time taken for the completion of tasks has been minimized after implementing TCSA algorithm. Fault tolerance has been done after implementing TCSA algorithm and it is observed that the task execution is getting done successfully by tolerating the occurred faults along with keeping the execution time minimized.

A comparison in the execution time is done between TCSA algorithm and Round Robin algorithm (i) with fault tolerance and (ii) without incorporating fault tolerance.

The execution time after implementing Round Robin Algorithm and TCSA without fault tolerance is depicted in Table 1 keeping the number of resources as constant to thousand.

Table 1: Execution Time After Implementing Round Robin Algorithm and TCSA without fault tolerance.

Number of Tasks	Round Robin (milliseconds)	TCSA (milliseconds)
100000	10531	6548
200000	11632	7240
300000	13012	7984
400000	13998	8154
500000	15765	9246
600000	17398	10230
700000	18837	11090
800000	20326	14304
900000	21485	17305
1000000	23567	18980

The graphical representation of Table 1 is depicted in Figure 2. In this graph, X-axis represents number of tasks along with keeping the number of resources constant to thousand and Y-axis represents the execution time in milliseconds. It can be observed that the execution time of TCSA is less when compared to the execution time of RR without fault tolerance.

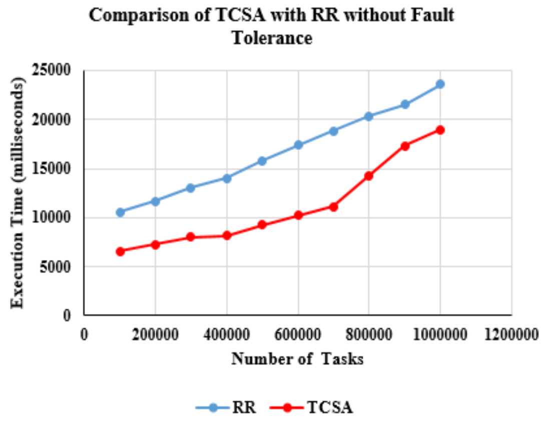
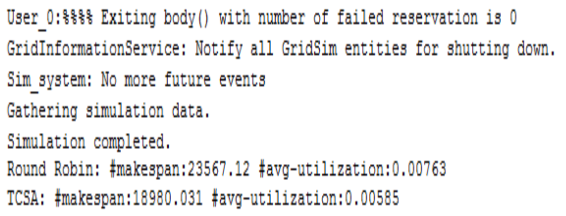


Figure 2: Comparison Between RR and TCSA Algorithm Without Fault Tolerance.

Figure 3 displays the result of makespan and average utilization of RR algorithm and TCSA without fault tolerance. It is observed that TCSA have the minimum makespan of 18980.031 milliseconds and average-utilization of 0.00585 milliseconds when compared to that of Round Robin algorithm which have a makespan of 23567.12 milliseconds and average-utilization of 0.00763 milliseconds.



```
User_0:%%%% Exiting body() with number of failed reservation is 0
GridInformationService: Notify all GridSim entities for shutting down.
Sim_system: No more future events
Gathering simulation data.
Simulation completed.
Round Robin: #makespan:23567.12 #avg-utilization:0.00763
TCSA: #makespan:18980.031 #avg-utilization:0.00585
```

Figure 3: Screenshot that displays the makespan and average-utilization without fault tolerance.

The execution time taken after implementing Round Robin Algorithm and TCSA with fault tolerance is depicted in Table 2 keeping the number of resources as constant to thousand.

Table 2: Execution Time After Implementing Round Robin Algorithm with fault tolerance.

Number of Tasks	Round Robin (milliseconds)	TCSA (milliseconds)
100000	9630	5934
200000	10120	6765
300000	12306	7146
400000	12740	7945
500000	14562	8790
600000	16224	9357
700000	17433	10703
800000	19320	13568
900000	20143	16850
1000000	21989	18362

The graphical representation of Table 2 is depicted in Figure 4. X-axis represents number of tasks along with keeping the number of resources constant to thousand and Y-axis represents the execution time in milliseconds. It is observed that the execution time of TCSA with fault tolerance is less when compared to the execution time of RR with fault tolerance.

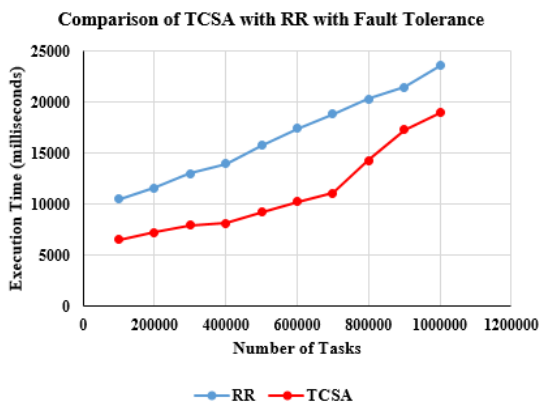


Figure 4: Comparison Between RR and TCSA Algorithm with Fault Tolerance.

The graphical representation of table 4 is shown in Figure 5 using the graph keeping number of resources constant as thousand and a decrease in execution time is observed. The process of fault tolerance helps in making the task allocation process fast and efficient. In this graph, X-axis represents number of tasks along with the number of resources as thousand and Y-axis represents the execution time in milliseconds.

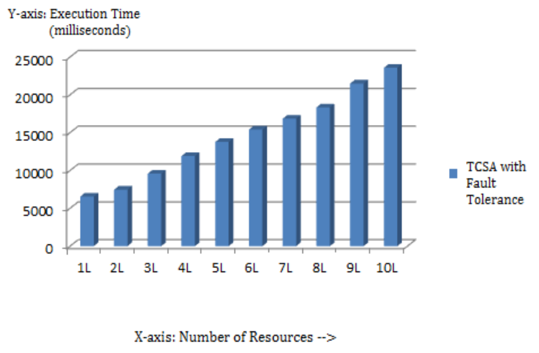


Figure 5: Representation of Minimized Execution Time.

Figure 6 displays the result of makespan and average utilization of RR algorithm and TCSA with fault tolerance. It is observed that TCSA have the minimum makespan of 18362.21 milliseconds and average-utilization of 0.00490 milliseconds when compared to that of Round Robin algorithm which have a makespan of 21898.03 milliseconds and average-utilization of 0.00683 milliseconds.

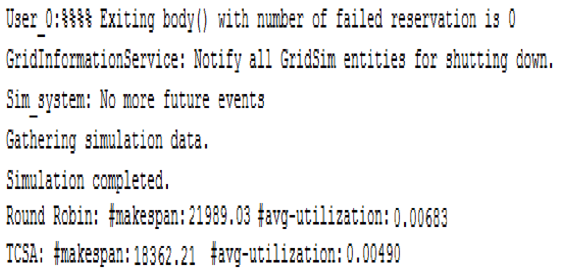

```
User_0:%%%% Exiting body() with number of failed reservation is 0
GridInformationService: Notify all GridSim entities for shutting down.
Sim_system: No more future events
Gathering simulation data.
Simulation completed.
Round Robin: #makespan:21989.03 #avg-utilization:0.00683
TCSA: #makespan:18362.21 #avg-utilization:0.00490
```

Figure 6: Screenshot that displays the makespan and average-utilization without fault tolerance.

It is very clearly evident that, TCSA with fault tolerance shows better performance with more number of resources and also the varying workload assigned. The efficiency is more when the incoming jobs are high. This helped the system to improve its overall performance by incorporating rescheduling with TCSA and in turn achieving high fault tolerance.

5. CONCLUSION

In any global computing environment, as the number of jobs to be executed increases, the failure rate of jobs also increases due to various faults with the systems. In order to efficiently deal with this issue, a suitable method for tolerating the faults has been proposed by incorporating fault tolerance with TCSA algorithm. The system had overcome the problems such as rescheduling of faulty jobs during task scheduling. Identification of faulty jobs and rescheduling of those jobs are internally done with the scheduler with the help of fault handler by the resource broker. The system focused on the comparison of the algorithms such Round Robin with fault tolerance with that of TCSA algorithm incorporating rescheduling with fault tolerance. It is observed that TCSA algorithm with fault tolerance is able to minimize the execution time by tolerating faults that occurs during scheduling than RR algorithm. Since the rescheduling of tasks is done internally with the scheduling mechanisms in TCSA, a significant improvement is observed as the system showed a better execution time than RR and with a better fault tolerance rate and hence the overall system performance are found to be enhanced .

Further studies in future can explore different ways of efficient fault tolerance mechanisms for other faults that occurs during job scheduling.

REFERENCES:

[1] AlEbrahim. S, and Ahmad. I, "Task scheduling for heterogeneous computing systems", *The Journal of Supercomputing*, Vol. 73, No. 6, 2017, pp. 2313-2338.

[2] Attiya. I, Zhang. X and Yang. X, "TCSA: A dynamic job scheduling algorithm for computational grids.", *In Computer Communication and the Internet (ICCCI), IEEE International Conference*, October, 2016, pp. 408-412.

[3] Aparna. S.K, K. Kousalya, "An Enhanced Adaptive Scoring Job Scheduling Algorithm with replication strategy in Grid Environment", *International Journal of Research in Engineering and Technology (IJRET)*, eISSN: 2319-1163, pISSN: 2321-7308, vol.3, No. 4, 2013, pp. 680-684.

[4] Bharathi. P.D, Prakash. P and Kiran. M.V. K, "Energy efficient strategy for task allocation and VM placement in cloud environment", *In Power and Advanced Computing Technologies (i-PACT)*, Innovations in IEEE, pp. 1-6, April, 2017.

[5] Maryam Masouding Khorsand, Shahram Jamali and Morteza Analoui, "A Survey of Job Scheduling Algorithms Whit Hierarchical Structure to Load Balancing in Grid Computing Environments", *International Journal of Computer Application Technology and Research (IJCATR)*, ISSN: 2319-8656, vol.3, No. 1, 2014, pp. 68-72.

[6] Banerjee. S, Hecker. J. P, "A Multi-Agent System Approach to Load-Balancing and Resource Allocation for Distributed Computing", *First Complex Systems Digital Campus World E-Conference 2015, Springer, Cham*, 2017, pp. 41-54.

[7] Suresh. S, Nagarajan. R, Sakthive. L, Logesh. V, Mohandass. C and Tamilselvan. G, "Transmission Line Fault Monitoring and Identification System by Using Internet of Things", *International Journal of Advanced Engineering Research and Science (IJAERS)*, April, 2017, Vol. 4, pp. 9-14.

[8] Ying. L, Srikant. R and Kang. X, "The power of slightly more than one sample in randomized load balancing", *In Computer Communications (INFOCOM), 2015 IEEE Conference,* April, 2015, pp. 1131-1139.

[9] Chauhan. A, Singh. S, Negi. S and Verma. S.K," Algorithm for deadline based task scheduling in heterogeneous grid environment", *In Next Generation Computing Technologies (NGCT), 2nd International Conference,* October, 2016, pp. 219-222.

[10] Chowhan. R.S, Mishra. A. and Mathur. A, "Aglet and kerrighed as a tool for load balancing and scheduling in distributed environment", *In Recent Advances and Innovations in Engineering (ICRAIE), 2016 International Conference*, December, 2016, pp. 1-6.

[11] Dave. A, Patel. B, and Bhatt. G, "Load balancing in cloud computing using optimization techniques: A study", *In Communication and Electronics Systems (ICCES), International Conference,* October 2016, pp. 1-6.

[12] Das. N.K, George. M.S and Jaya. P, "Incorporating weighted round robin in honeybee algorithm for enhanced load balancing in cloud environment", *In Communication and Signal Processing (ICCSP), 2017 International Conference,* April, 2017, pp. 0384-0389.

[13] Parwekar. P, and Rodda. S, "Fault Tolerance in Wireless Sensor Networks: Finding Primary Path", *In Proceedings of the Second International Conference on Computer and Communication Technologies,* Springer(New Delhi), 2016, pp. 593-604.

[14] Zhao. J, Yang. K, Wei. X, Ding. Y, Hu. L and Xu. G, "A heuristic clustering-based task deployment approach for load balancing using bayes theorem in cloud environment", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 27, No. 2, pp.305-316.

[15] Oueis. J, Strinati. E.C and Barbarossa. S, "The fog balancing: Load distribution for small cell cloud computing", *In Vehicular Technology Conference (VTC Spring), 2015 IEEE 81st*, May, 2015, pp. 1-6.

[16] Khan. Z.F, "Novel architecture for effective load balancing and dynamic group scheduling in grid computing topology", *In Circuit, Power and Computing Technologies (ICCPCT), 2017 International Conference,* April, 2017, pp. 1-7.

[17] Sarmila. G.P, Gnanambigai. N and Dinadayalan. P, "Survey on fault tolerant—Load balancing algorithmsin cloud computing", *In Electronics and Communication Systems (ICECS), 2015*

2nd International Conference, February, 2015, pp. 1715-1720.

[18] Indhumathi. V and Nasira. G.M, "Service oriented architecture for load balancing with fault tolerant in grid computing", *In Advances in Computer Applications (ICACA), IEEE International Conference,* October, 2016, pp. 313-317.

[19] Nasir. M.A.U, Morales. G.D, García-Soriano. D, Kourtellis. N and Serafini. M, "The power of both choices: Practical load balancing for distributed stream processing engines", *In Data Engineering (ICDE), 2015 IEEE 31st International Conference,* 2015, April, 2015, pp. 137-148.

[20] Pinto. J, Jain. P and Kumar. T, "Hadoop distributed computing clusters for fault prediction", *Computer Science and Engineering Conference (ICSEC), 2016 International*, December, 2016, pp. 1-6.

[21] Kaur. P and Aggarwal. D, "Analysis of Fault Tolerance on Grid Computing in Real Time Approach", *World Academy of Science, Engineering and Technology, International Journal of Computer and Information Engineering*, 2015, Vol. *2, No.* 11.

[22] Narale. S.A and Butey. P.K, "Fault-Tolerance Techniques and its Accomplishment in Cloud Computing Environment: A Study", *International Journal of Computer Applications (IJCA)*, 2011, pp. 0975-8887.

[23] Goyal. N, Dave. M and Verma. A.K, "A novel fault detection and recovery technique for cluster-based underwater wireless sensor networks", *International Journal of Communication Systems*, 2018, Vol. *3, No.* 4.

[24] Hannache. O, Batouche. M, "Probabilistic model for evaluating a proactive fault tolerance approach in the cloud", *In Service Operations and Logistics, And Informatics (SOLI), 2015 IEEE International Conference,* November, 2015, pp. 94-99.

[25] Lee. J.S, Choi. U.M and Lee. K.B, "Comparison of tolerance controls for open-switch fault in a grid-connected T-type rectifier", *IEEE Transactions on Power Electronics*, 2015, Vol. *30, No.* 10, 5810-5820.

[26] Medeiros. R, Cirne. W, Brasileiro. F and Sauvé. J, "Faults in Grids: Why are they so bad and What can be done about it?", *In Grid Computing, 2003. Proceedings. Fourth International Workshop,* November, 2003, pp. 18-24.

[27] Manishankar. S, Srinithi. C.R and Joseph. D, "Comprehensive study of wireless networks qos parameters and comparing their performance based on real time scenario", In *Innovations in Information, Embedded and Communication Systems (ICIIECS), International Conference,* March, 2017, pp. 1-6.

[28] Manishankar. S, Sandhya. R. and Bhagyashree. S, "Dynamic load balancing for cloud partition in public cloud model using VISTA scheduler algorithm", *Journal of Theoretical and Applied Information Technology*, May, 2016, Vol.*87, No.* 2, pp. 285-290.

[29] Manishankar. S, Indrajith. A.N and Monika B.R, "Alpha scheduler approach to enhance security and high performance in cluster environment", *Journal of Theoretical and Applied Information Technology*, 2016, Vol. *87, No.* 1, pp. 138-145.