VOL. 12, NO. 23, DECEMBER 2017 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

A PRO-ACTIVE FAULT TOLERANT DEADLINE HIT COUNT BASED SCHEDULING IN COMPUTATIONAL GRID

S. Gokuldev[1], C. Sowntharya[2] and S. Manishankar[1]
[1]Department of Computer Science, Amrita Vishwa Vidyapeetham, Mysore Campus, Karnataka, India
[2]Department of Computer Science and Engineering, Karpagam College of Engineering, Coimbatore, Tamil Nadu, India
E-Mail: gokuldevs@gmail.com

ABSTRACT

In grid systems, as the number of heterogeneous components increases in the networks, the chance of failure of resources increases. Identifying the various faults that occurs and imparting tolerance to those faults has become the principal area of concern. Many of the existing fault tolerant algorithms concentrate on increasing throughput and reducing the response time but consider less on increasing user satisfaction. The proposed fault-tolerant check-pointing based Deadline Hit Count (DHC) scheduling algorithm concentrates on increasing the efficiency of the resource through identifying the faults and reducing the turn-around time. It also increases the user satisfaction as it combines both the dynamic checkpointing approach and fault tolerant scheduling. In the proposed technique, a DHC scheduling algorithm with check-pointing is implemented to identify and pro-actively tolerate faults to select the appropriate resources. Experiments are performed to assess the performance of the proposed approach using GridSim tool and have shown better performance.

Keywords: computational grid, deadline hit count, dynamic check-pointing, fault identification, fault tolerance, failure counter, success indicator.

1. INTRODUCTION

Grid computing is another form of distributed computing which enables selection and aggregation of a very large number of geographically distributed resources dynamically based on their availability and capability. Considering the general aspects of grid computing functions, though grid faces lot challenges with utilizing the resources to its optimum level, tolerating the faults of resources also plays a major role. In the generalized view of grid system, the overview of the user job submission to the grid system and the specific flow of dealing with the grid job is described. Job scheduling is performed with the system and the system also identifies the resource faults and the fault tolerating mechanism as its objective. The generalized view of grid architecture is depicted in Figure-1. Initially the user submits the job to the scheduler. The scheduler refers to the Grid Information Server (GIS) for collecting information about the resources. The scheduler then schedules the jobs to the appropriate master node in different networks which in-turn schedules to its individual worker nodes.

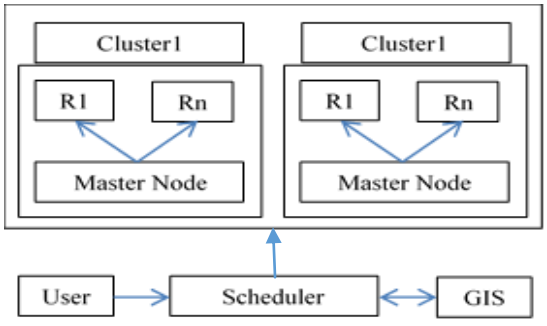

Figure-1. Generalized view of grid network.

Fault tolerance is the ability of the system to execute the tasks efficiently even in the presence of faults. The main scheduling methods used for fault tolerant scheduling are pro-active and post-active scheduling. In the former scheduling for fault tolerance, the fault factors are considered prior to the scheduling decisions so that the probabilities of occurrence of faults are minimized. In post-active scheduling tasks, measures are taken to overcome faults only after they have occurred. The fault tolerance mechanisms used in most of the recent research works are either check pointing or job replication mechanisms. In the check pointing mechanisms, the check points are generated at constant intervals of time and if any fault/failure happens, the system rolls back to the previous checkpoint. This reduces the total time taken to complete the tasks as the tasks need not be executed again from the scratch. With the fault tolerance mechanism based on job-replication, the replica of jobs is created. The main agenda behind this technique is that if a job fails due to some reasons, the scheduler does not have to request the user again for the same job as the replicas of the job exists within the scheduler itself. Different algorithms use different types of replications such as:

- Scheduler can decide the number of replicas depending on the resource to which the jobs are assigned.
- The user can provide the number of replicas depending upon the priority of the tasks.
- The system itself could be trained to consider a fixed number of replicas for each task statically for all incoming jobs.

Designing a fault tolerant algorithm to decrease the overall makes pan is a critical issue to be considered. In the existing system using Failure Tendency and Success Indicator (SI) has been used for proposing a fault tolerant scheduling algorithm which reduced the overall turn-around time for the submitted jobs with minimal deadline with the limited number of resources. User satisfaction is also a parameter to measure the success rate of the system, but in the existing system, identifying the occurrences of faults has not been considered. Success indicator is a parameter with a count value that dynamically gets updated irrespective of the completion of jobs. An automatic SI check-pointing mechanism has been introduced that dynamically selects the threshold to update the SI in both the presence and absence of faults. The proposed idea is an efficient fault tolerant system which introduces the idea to combine Deadline Hit Count (DHC) and automatic SI check-pointing approach. DHC gives the number of jobs successfully completed within the user deadline constraint and SI check-pointing introduces the automatic checkpoint depending on which the SI gets updated. To achieve a decreasing turn-around time for jobs with minimum deadline, the system selects the most appropriate resources by considering the parameters such as DHC, SI and FC for the execution of tasks. So there is only less chance of failure of jobs and resources due to appropriate selection of resources and assigning high load to resources.

The above section provides an introduction the problem statement. Section II discusses on the various review of literature on existing scheduling and fault tolerant methodologies, section III illustrates the proposed DHC based scheduling and fault tolerant scheduling in comparison with one of the existing methodologies, section IV shows the experimental results and its significance and section V covers the conclusion and future scope of the proposed approach

2. REVIEW OF THE EXISTING FAULT TOLERANT APPROACHES

Fiaz Gul Khan *et al* (2010) made a comparison among the most commonly used fault tolerant techniques such as check pointing, retrying, alternative task and alternative resource. The comparative study [1] addressed also considered many system centric parameters such as throughput, turn-around time which is a measure of the total time taken from the time user submits the job till the user receives the result, waiting time which is the measure of the time that the user job has to wait in the queue before the job gets scheduled under different workload conditions with the described algorithms. In alternative task selection technique where in case of task failure the system resubmission of the task, it is observed that the algorithm works well under high workload conditions as well as with task failures. Since it causes less network delays, it gives good performance. But, it is found that waiting time is high due to re-submission of failed tasks. Check pointing gives better results for different kinds of faults but only in low workload conditions. In high workload conditions, it is found to cause memory over-head.

Malarvizhi Nandagopal *et al* (2010) in their work, proposed a fault tolerant algorithm to efficiently schedule the tasks. Replication Resource Selection Algorithm (RRSA) [2] is projected that provides Checkpoint eplication Service (CRS). It reduces the Time to Repair (TTR) of the submitted jobs in the grid and the proportion of completed jobs within the given deadline has been increased. But it has been observed that increasing the throughput of the system has not been considered.

Amoon (2011), in the work addressed an approach to deal with task failures even in the presence of resource faults. The work focuses on selection of resources during job replication [4] which depends on the failure tendency of resources calculated using previous history of resources such as response time. It is observed that the system works efficiently by reducing the memory overhead that is usually caused in job replication since the replicas will be located and terminated after the task gets completed. But since failure tendency is used as a parameter for selection of resources, there are cases where it becomes difficult to select the resource for scheduling.

P Keerthika *et al* (2011), a fault tolerant scheduling algorithm [5] has been proposed based on transmission time, fault rate and user deadline and the job is expected to be completed within the user deadline by assigning it to the most appropriate resource. The algorithm has proved efficient with high hit rate (number of jobs successfully completed within the specified deadline) and less miss rate. The addressed algorithm works well for static scheduling but the efficiency of the system in dynamic scheduling has not been considered.

Jasma Balasangameshwara *et al* (2012) in their work, introduced a fault tolerant hybrid load balancing strategy namely AlgHybrid_LB [6]. The strategy takes into account grid architecture, computer heterogeneity, job characteristics, resource availability, communication delay, resource unpredictability and job characteristics. The system has an optimal utilization of computer nodes and minimum response time. To locate effective sites, AlgHybrid_LB integrates static and dynamic load balancing techniques. The proposed approach is of low complexity and reduces the number of additional communication caused due to load balancing.

Altameem (2013) addressed the issue of fault tolerance in grid systems using the method of job replication [7]. Once the client submits the job and the number of replicas of each job, the jobs are assigned to the appropriate resources by considering the fail tendency of resources which is the probability of a resource to fail. The system is found to be working efficiently for job failures but the problem of memory over-head created due to excessive replication of jobs has not been considered.

P. Keerthika, *et al* (2013), in their work developed a pro-active scheduling algorithm [8] with improved fault tolerance and increased user satisfaction. A new parameter of hit count to represent the number of tasks successfully completed within the user deadline has been introduced. It has been found that it produced increased user satisfaction with increased tolerance to

faults. But the efficiency of the algorithm under high workload conditions has to be verified.

Jairam Naik *et al* (2013) proposed a scheduling algorithm for fault tolerance [9] to increase the efficiency and overall throughput of the system. The work introduced a new parameter called the Scheduling Success Indicator (SSI) which is a grouping of resource experience and success rate of the resources. It has been observed that the system produced better results than the Fault Indicator (FI) based systems where only the failure tendency of the resource has been considered. SSI has been considered but whether the jobs are being completed within the deadline or not has not been considered.

Ch. Ramesh Babu *et al* (2014), presented a strategy for automated checkpointing [10] based on different scheduling algorithms and has been evaluated using OpenMPI and the basic infrastructure delivered by the BLCR 0.8 framework using TORQUE. It was found that overheads due to checkpointing are reduced by automatic checkpointing. It is also found that checkpoint turnaround time is reduced by automated checkpointing which acquired better performance over other techniques.

S.Supriya *et al* (2014) conducted a survey [11] on various fault tolerance techniques, mechanisms and job management. The work addressed the various techniques in fault tolerance which is fault masking (preventing fault in the resource) and reconfiguration (removal of faulty resources). It was observed that check pointing mechanism is an approach to reduce the failure recovery time with reduced makespan.

P. Keerthika *et al* (2015) described a budget constrained scheduling algorithm [12] that concentrated on processing costs. Along with cost factor, it considered the deadline of tasks to satisfy the user. This algorithm has also been designed to take care about fault tolerance with reduced makespan and proper utilization of resources. It has been observed that since reducing cost and user deadline of task has been considered, it has increased the user satisfaction greatly. This algorithm is found to be following a centralized approach which could be changed to hierarchical to improve efficiency.

K Nirmala Devi *et al* (2015) described a scheduling algorithm [13] by altering the execution time dynamically to maximize the throughput. If any failure of node happens, it efficiently reschedules the task to another safe resource based on its previous work history. It has been observed that the recommended algorithm maximizes the throughput and minimizes overhead due to checkpointing. The prediction approach considered history of resources to predict the failure rate of resources.

M. Nakkeeran (2015) addressed fault tolerance in resource failure and made a survey [14] on task checkpointing and replication based fault tolerance. The system achieved fault tolerance by dynamically adjusting the number of checkpoints by Mean Failure Dependent Checkpoint based job execution time and history of failure information, which reduced checkpoint overhead and increased the overall throughput. The system is not considering the system overhead.

Sarpreet singh *et al* (2016), considered the faulty nodes before the scheduling or execution of tasks [15] by considering the resource history and take corrective actions which minimizes execution time, increase execution rate, reduce faults occurrence & execution cost. It has been observed that even in the presence of faults, the time taken to complete the execution of tasks by the system is less as compared with post-active scheduling techniques. Since it also uses check-pointing, it reduces the restart time of the tasks.

It is observed that in the reviewed works, fault indicator represents the total number of task successfully completed versus the total number of tasks executed by the grid resource. For example, if R1 and R2 are two resources, R1 successfully completed 9 tasks out of 10 and R2 has completed 90 tasks out of 100 submitted tasks. But in such a case, fault indicator for both resources will be 0.9. Selection of resources proved to be difficult in such situations. So the proposed work uses fault counter that gives the measure of number of tasks failed to the total number of task that is submitted to the grid. So the fault counter for R1 and R2 will be 5% and 50% if the total tasks submitted to the grid is 5000.

3. METHODOLOGY

3.1 Failure tendency and scheduling success indicator based fault tolerant scheduling

Scheduling Success Indicator (SSI) based fault tolerant scheduling algorithm [9] introduced a new parameter called the SSI that is used to group the resources based on the resource history which indicates the success rate of the resources. A parameter called Failure Indicator (FI) is also used for computing the resource faults. FI gives the percentage of resource failure of individual resource. This helped the system to assign jobs to the resources which has lesser failure rate.

In other words, the system chooses a best resource for job considering lesser FI and higher SSI. From this scheduling approach, it has been observed that the system produced better results when compared with many of the latest existing techniques and it showed improved throughput. Since the work focused only on success rate of the resource and failure tendency and not focused much on user satisfaction. This paved the way to propose a new scheduling and fault tolerant technique to focus on user satisfaction by considering deadline constraint through assigning optimal resource by considering success rate and failure rate of the resources.

3.2 Proposed pro-active fault tolerant deadline hit count based scheduling

The proposed system follows a pro-active fault tolerant scheduling approach which takes actions prior to scheduling of tasks to minimize the occurrence of faults and the overall turn-around time. Along with SI and FC, an additional parameter called DHC is used which computes the measure of the number of jobs successfully completed within the user deadline. The user submits the job with the QoS requirements to the grid system i.e., the

user deadline. Before scheduling the jobs to the resources, the scheduler, once after receiving the job from the user, consults the Resource Information Server (RIS) for FC, SI and DHC of resources which are computed based on the history of resources which is updated in RIS by the fault handler. The jobs are scheduled to the appropriate resources based on the corresponding computed counter values so that the occurrences of resource faults are minimized. The illustration of the proposed system model is shown in Figure-2.

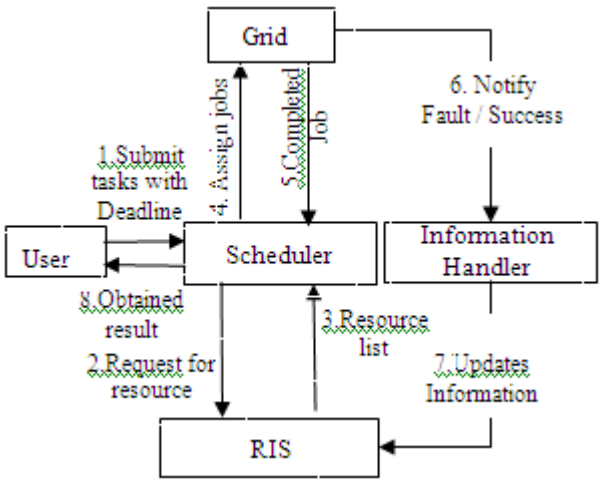

Figure-2. System model.

The proposed system minimizes the resource faults and task faults by selecting appropriate resources to the incoming tasks by considering parameters such as FI, SI and DHC. Since the optimum resources for the jobs are selected, the system decreases the overall time taken to complete the set of user submitted tasks.

a) User submits the jobs with the task deadline as QOS requirement to the grid scheduler.

b) Once when the scheduler receives the job, this sends a request to the RIS informing about the task to be done.

c) RIS consists of information about the computing resources. It maintains a table which consists of the network_id, resource_id, deadline hit-count, success rate of the resources and deadline. When a request for a resource arrives, the RIS, based on the computing information, finds the optimum resource and informs the scheduler with respective resource_id and network_id.

d) According to the size and deadline of different jobs, the system decides the number of resources and the number of searches to be performed to find optimum resources among the idle ones.

e) The scheduler decides the allocation of resources based on the received information.

f) At constant intervals of time, the resources in the network send live messages to find whether they are alive.

g) **Case 1:** If a live message has not been received from a resource which has been executing the tasks, then the master reports to the fault handler about the failure. Fault handler does the appropriate computation, updates the information in the RIS.

h) **Case 2:** If the job is executed successfully, the computed job is send to the scheduler and a notification is sent to the information handler and simultaneously RIS is updated.

i) In both the cases, the SI acts as an automatic checkpoint count value that gets updated at dynamically selected intervals of time depending on the presence and absence of faults.

j) **Case A:** In the presence of resource fault, the static value which is fixed as threshold for checkpointing interval changes. The new value for checkpoint will be initiated for the threshold, from the time of fault occurred.

k) **Case B:** In the absence of fault, ie. if job is completed successfully, the threshold time interval remains constant which provides a mechanism for fault identification.

The scheduling decisions are taken by the scheduler depending on different parameters. The computations regarding the values of the different counters are performed in the information handler. The mathematical formulations for these counters are as follows:

Initially,

$$FC = Ci \quad (1)$$

$$SI = Ci \quad (2)$$

$$DHC = Ci \quad (3)$$

where 'Ci' is a constant and it is initially assigned to all the counters.

A constant unique value is assigned initially to all these three parameters so that it provides an easy mechanism to calculate and compare all the counters for the GIS and to the scheduler to make the scheduling decisions for all appropriate resources.

The FC, SI and DHC of resources 'i' is defined as follows:

FC = Ci+Ei (4)

where,

$$Ei = \text{round} \left(\frac{\text{-off Number of resources failed}}{\text{Total number of resources allocated}} \right) \quad (5)$$

Ei gives the fraction of jobs failed against the total number of jobs allocated. FC is a counter that gets incremented when a resource fails in execution and even when a resource fails sending live message to the master node.

SI = Ci+Fi (6)

where,

$$Fi = \text{round-off} \left(\frac{\text{Number of jobs completed}}{\text{Total number of jobs submitted}} \right) \quad (7)$$

Fi is the fraction of jobs completed successfully against the total number of jobs submitted. SI is a counter value that gets incremented whenever a job gets completed successfully irrespective of user deadline.

DHC = Ci + Gi (8)

where,

$$Gi = \text{round-off} \left(\frac{\text{Total time taken by resources to complete the tasks}}{\text{User provided deadline}} \right) \quad (9)$$

Gi provides the fraction of total time taken by the resource to complete the task against the user provided deadline.

DHC contributes more in the proposed DHC based fault-tolerant scheduling algorithm. It is a measure of number of jobs successfully completed within the user given deadline. DHC counter gets incremented only when the above mentioned condition is met i.e., only when the job is completed within the user given deadline. Thus, the proposed DHC based system incorporates pro-active fault tolerant scheduling with check pointing approach for fault tolerance and fault identification respectively. Once when a fault is identified using the check pointing technique, the FC rate of the respective faulty resource dynamically increases and contributes a negative impact on the resource while scheduling decisions are taken. This leads to the significant improvements in the system and also improves the overall performance of the grid system.

4. EXPERIMENTAL RESULTS

The experiments are conducted with the simulation set up in the simulation environment, considering two clusters with 50 computing resources each. The experimental results for 9000 jobs with minimal deadline of 500 milliseconds as QoS constraint is depicted in Figure-3. As the system allocates minimum number of resources to get the job done considering the QoS requirements, the system finds the exact matching resources for the incoming jobs and the execution time is observed.

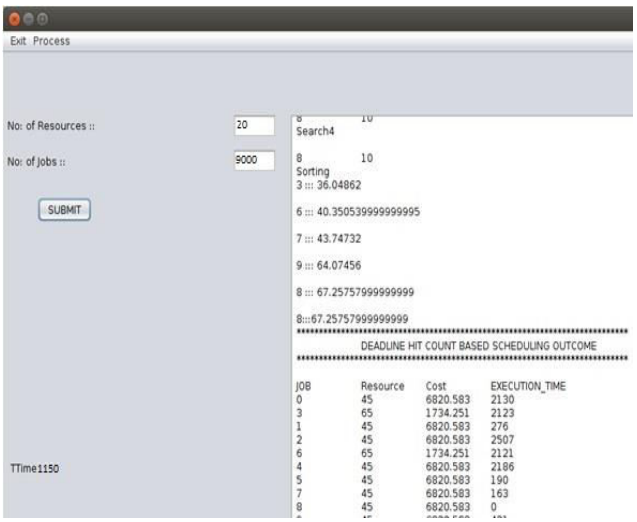

Figure-3. Execution time of DHC scheduling.

From the observations made, it is found that the for 9000 jobs with minimal deadline, the turn-around time obtained is minimal as compared with the existing fault tolerant based scheduling approach. The results provides an evident comparison between the existing Success Indicator fault tolerant approach and the proposed deadline hit count based approach and proves that the proposed approach produced better results. The simulation is also experimented for varying workloads with varying number of resources and varying number of searches.

The experimental results show that with limited number of resources and for the jobs with minimal deadline, the proposed fault tolerant based scheduling algorithm works efficiently. The algorithm produced less turn-around time for jobs with minimal deadline and under limited resources. The algorithm shows improved results and better user-satisfaction as it works well for jobs with minimal deadline by producing less turn-around time.

Simulations had been conducted for comparing the existing SSI based system with the proposed DHC based system.

Table-1. Comparative study of turn-around time of SSI versus DHC based systems.

No. of tasks	SSI based fault tolerant scheduling	DHC based fault tolerant scheduling
500	673	592
1000	892	746
1500	947	869
2000	1050	900
4000	2099	1110
9000	4978	1150

The Table-1 shows the comparison of turn-around time of the two approaches, SSI and DHC based fault tolerant scheduling algorithms considering 500, 1000, 1500 and 2000 tasks.

Observations from the above table provides a clarity on the fact that the DHC based system produces better turn-around time as compared with SSI based system. As it produces less turn-around time for higher number of tasks with minimal deadline, it is assured that the proposed approach contributes in increasing user satisfaction as jobs gets completed within the limited available time. Since most suitable resources are selected for execution by considering various parameters, probability of occurrence of faults are found to be minimized.

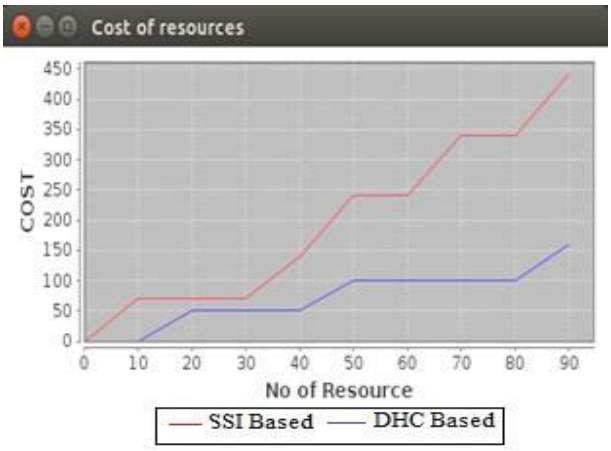

Figure-4. Comparison of the existing approach with DHC scheduling based on resource cost.

In Figure-4, the cost of resources is plotted against the number of resources. It is clearly observed that the proposed pro-active fault tolerant scheduling algorithm produces less cost for resources as compared with the other existing SSI based fault tolerant algorithm.

In Figure-5, the turn-around time is plotted against the number of resources and it is observed.

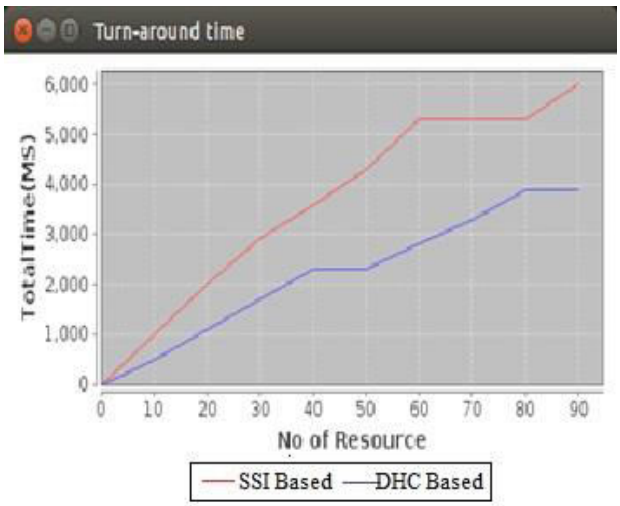

Figure-5. Comparison of the existing approach with DHC scheduling based on execution time.

There is a drastic decrease in the total time taken by the proposed DHC algorithm when compared with the existing SSI based fault tolerant approach.

With certain number of resources from the simulation results, it has been observed that the turn-around time has been stabilized and it is parallel to x-axis. Hence, the conclusion is that, as the number of jobs and resources increase, after a certain threshold value, the turn-around time becomes static for some particular number of resource and again the time increases with respect to the resource which conveys that the turn-around time which is static increases the user satisfaction as far as any higher number of jobs are concerned. Hence the system produces minimized turn-around time.

The proposed approach contributes on reducing the total turn-around time and simultaneously helps in identification of faults using check-pointing technique. It has been observed that the implemented check-pointing technique helps in identification of faults which is inferred from the variation in threshold time that updates the SI. If a time variation in observed in the time SI is updated, it is concluded that a fault has occurred which has caused the variation.

5. CONCLUSIONS AND FUTURE SCOPE

An efficient fault tolerant scheduling algorithm with fault identification is developed and implemented. The system pro-actively considers fault, schedules the tasks but post-actively collects the data and identifies whether the fault has occurred or not. FC, SI, DHC has been used in the proposed work which improved the overall performance of computational grid system by reducing the response time of jobs and also increases the overall user satisfaction. This is achieved by selecting most appropriate resources in the computational environment, as well as helping in identifying whether the fault has occurred or not. The proposed system resulted with improved performance showing significant results compared with fault tolerant based existing systems.

The scope is to include fault tolerance to increased number of faults and to improve other system centric parameters such as throughput, utilization of resources which could further help to enhance the grid system performance. Also the work could be extended to support excessive number of jobs with minimal deadline and limited number of resources.

REFERENCES

[1] J. H. Abawajy. 2004. Fault-Tolerant Scheduling Policy for Grid Computing Systems. Proc. of the 18th International Parallel and Distributed Processing Symposium (IPDPS'04) IEEE. Vol. 14, no., pp. 238b, IPDPS.2004.1303290.

[2] Fiaz Gul, Kalim Qureshi and Babar Nazi R. 2010. Performance evaluation of fault tolerance techniques in grid computing system. Computers & Electrical Engineering. 36.6, pp. 1110-1122.

[3] Nandagopal Malarvizhi and V. Rhymend Uthariaraj. 2010. Fault tolerant scheduling strategy for computational grid environment. International Journal of Engineering Science and Technology. 2.9: 4361-4372.

[4] Amoon M. 2011. A development of fault-tolerant and scheduling system for grid computing. GESJ: Computer Science & Telecommunications. 32.3: 44-52.

[5] Keerthika, P., and N. Kasthuri. 2011. A New Proactive Fault Tolerant Approach for Scheduling in Computational Grid. In: Proc. of International Conf. On Web Services Computing (ICWSC) published by International Journal of Computer Applications. 201(1): 55-59.

[6] Balasangameshwara Jasma, and Nedunchezhian Raju. 2012. A hybrid policy for fault tolerant load balancing in grid computing environments. Journal of Network and computer Applications. 35.1, pp. 412-422.

[7] Altameem T. 2013. Fault tolerance techniques in grid computing systems. International Journal of Computer Science and Information Technologies. 4.6, pp. 858-862.

[8] Keerthika P. and N. Kasthuri. 2013. An efficient grid scheduling algorithm with fault tolerance and user satisfaction. Mathematical Problems in Engineering. pp. 1-9.

[9] Naik, K. Jairam, and N. Satyanarayana. 2013. A novel fault-tolerant task scheduling algorithm for computational grids. In: Proc. of International Conf. On Advanced Computing Technologies (ICACT), 15th International Conference on. IEEE.

[10] Babu Ch Ramesh and Ch DV Subba Rao. 2014. Automatic checkpointing based fault tolerance in computational grid. In: Proc. of International Conf. Computing, Management and Telecommunications (ComManTel), International Conference on. IEEE. pp. 41-45.

[11] S. Supriya, S.Dinesh Babu. 2014. A Survey on Fault Toerance Mechanisms for job scheduling in Grid computing. IOSR Journal of Computer Engineering (IOSR-JCE). 16(2): 120-122.

[12] Keerthika P. and P. Suresh. 2015. A Multiconstrained Grid Scheduling Algorithm with Load Balancing and Fault Tolerance. The Scientific World Journal. 2015: 1-10.

[13] Tamilarasi A. 2015. Dynamic Scheduling in Grid Envronent with the Improvement of Fault Tolerant Level. Indian Journal of Science and Technology. 8: 507-515.

[14] Nakkeeran M. 2015. A Survey on Task Checkponting and Replication based Fault Tolerance in Grid Computing. International Research Journal on Engineering and Technology. 02(09): 135-144.

[15] S. Manishankar, Sandhya R. and Bhagyashree S. 2016. Dynamic load balancing for cloud partition in public cloud model using VISTA scheduler algorithm. Journal of Theoretical and Applied Information Technology. 87: 285-290.

[16] Ra Pillay, Chandran, S. K, and Punnekkat. 2010. Optimizing resources in real-time scheduling for fault tolerant processors. In: Proc. of International Conf. on 1st International Conference on Parallel, Distributed and Grid Computing. pp. 101-106.