IJE TRANSACTIONS B: Applications Vol. 27, No. 5, (May 2014) 699-708

International Journal of Engineering

Journal Homepage: www.ije.ir

Stability Assessment Metamorphic Approach (SAMA) for Effective Scheduling based on Fault Tolerance in Computational Grid

A. Shamila Ebenezer * [a], K. Baskaran [b]

[a] *Department of Computer Science and Engineering, Karunya University, Coimbatore, India.*
[b] *Department of Computer Science and Engineering,Government College of Technology, Coimbatore, India.*

PAPER INFO

Paper history:
Received 16 May 2013
Received in revised form 05 September 2013
Accepted 07 November 2013

Keywords:
Checkpointing
Grid Computing
Fault Tolerance
Recovery Rate
Scheduling

ABSTRACT

Grid computing allows coordinated and controlled resource sharing and problem solving in multi-institutional, dynamic virtual organizations. Moreover, fault tolerance and task scheduling is an important issue for large-scale computational grid because of its unreliable nature of grid resources. Commonly exploited techniques to realize fault tolerance is periodic checkpointing that periodically saves the job's state. But an inappropriate checkpointing interval prevails to delay in the job execution, and reduces the throughput. With that concern, this paper endeavors to ensure better performance on computational grid with more effective and reliable fault tolerant system using a novel Stability Assessment Metamorphic Approach (SAMA). Here, the strategy used to attain fault tolerance is by adapting the checkpoints depending on the current status and past failure information about the resources dynamically, which is being maintained in the information server. The effective scheduling process can be achieved by fault tolerance based scheduling that involves the determination of deviation rate of all nodes using some high-stability assessment constraints. This evinces the job to be accomplished within the deadline with improved throughput and paves a way for making the grid environment trustworthy.

doi: *10.5829/idosi.ije.2014.27.05b.04*

1. INTRODUCTION

In general, a computational grid is an assortment of software and hardware resources, which provide seamless and pervasive access to high-end computational potentialities. It has been extensively adopted in science, technology, business processing and engineering computing. The resources on the grid may be geographically deployed in multiple managerial domains and can be installed on any operating system such as Windows, Linux or UNIX. With these applications, users may have variant computing and other resource necessities in the system. Mainly, the role of the grid system can be assorted into three parts. First, a computing broker is used to divide the jobs into smaller tasks. That will make jobs accomplish faster, by processing on multiple available resources. Second, the grid resource information service (GRIS) that constitutes the functions such as resource table maintenance and discovery. Third, the task scheduler is conscientious for dispatching tasks to resources based on the scheduling algorithms [1].

Moreover, grid and cluster architectures have acquired popularity for computationally concentrated parallel applications. Though, the complexity of the infrastructure consists of mass storage, computational nodes and interconnection networks, pretends great challenges with respect to overall system reliability. Compared to other distributed environments, complexity of grid mainly instigates from resource heterogeneity and decentralized management. These characteristics frequently lead to strong discrepancies in availability, which in particular depends on network failure rates and resources, administrative policies, and vacillations in system load. Apparently, runtime modifications in system availability can considerably affect job execution. Since for a large group of time depleting jobs delay and loss are not acceptable, wherein fault tolerance should be taken into the account. There is a great confrontation in providing fault tolerance in a distributed environment, while optimizing

*Corresponding Author Email: shamilaebenezer@gmail.com (A. Shamila Ebenezer)

job execution times and resource utilization. In order to accomplish it, dynamic adaptation of checkpoints technique is used in this proposal, that is based on the failure history of the resource and current status of the job, which conquers the checkpoint overhead that is elicited by unnecessary checkpoints in case of periodic check pointing. And hence, the process achieves fault tolerance with improved throughput. Figure 1 shows the typical gird environment consisting number of grid nodes and a control node.

Fault tolerance makes to reach system dependability. Dependability is associated with some QoS aspects afforded by the system that may include the attributes like reliability and availability [2]. Reliability denotes that a system can run constantly without failure. A highly reliable system is the system that persists to work without any interruption over a comparatively long period of time. Availability terms that a system is instantly ready for use. Fault tolerance techniques are habitually used to increase the availability and reliability rates in computing. Further, reliability is defined as the probability that the system remains continuously operational in the time interval [0, t]. It is also stated that reliability is intimately related to Mean Time to Failure (MTTF) and Mean Time between Failures (MTBF) rates. MTTF is given as the average time the system can operate until a failure occurs, whereas the MTBF is defined as the average time between two successive failures. The difference between the fore mentioned rates is due to the time required to renovate the system following the first failure. Representing the Mean Time to Repair by MTTR, it is obtained that, MTBF=MTTF+MTTR. It is also described that the checkpoint redundancy of the system is based on its MTTR. Figure 2 exemplifies the dependability taxonomy on which the performance of computational grid is based.

Further, grid system provides fault-tolerant scheduling by associating a set of fault tolerance techniques to endure crash faults in components of the network system [3].

- Fault tolerance in the constructive coordination model can be measured at two levels. Fault tolerance of the tuple space, i.e., the problem of endorsing that the space does not be unsuccessful if there are faults in the tuple space itself; and
- Application-level fault tolerance, which ensures the applications satisfaction and certain dependability properties even if some of the application processes fail.

To tolerate failures in grid systems, parallel applications classically instrument themselves with the capability to checkpoint their computation or execution state to stable storage. When one or more processors fail to accomplish a task, the application can be restarted from the most recent checkpoint; thereyby it reduces the amount of recomputation that must be achieved [4]. With those concerns, the proposed work concentrates on effective scheduling process over the distributed grid nodes that highly engrosses the fault estimations and tolerance. The main motive of the proposal is to provide reliability and efficiency for scheduling in computational grid.

The remainder of this paper is organized as follows. Section 2 gives a deliberation on the related work. Section 3 presents the proposed Stability Assessment Metamorphic Approach (SAMA) pertaining to increase efficiency of scheduling and reliability. Section 4 presents the experimental results and Section 5 concludes the paper with pointers to future work.

2. RELATED WORKS

Since scheduling in computational grid remains an unsettled affair, the research in this field is still an ongoing process. A technique called fault tolerant scheduling strategy has been proposed in some works [5] for computational grid. The process is composed of checkpoint replication based fault tolerance method with MTTR job scheduling mechanism.

Job checkpointing is based on the resource failure state in this fault tolerance algorithm. The main contributions are stated as follows:

- Fault tolerant mechanism for computational grid
- MTTR based scheduling strategy
- Incorporation of checkpointing recovery mechanism
- Implementation has been made with Globus Tool Kit

Following that, a novel system is used for transparent coordinated checkpoint-restart of distributed network applications on computational clusters. They have introduced ZapC checkpoint that effectively restarts the operations in parallel across different computational nodes [6]. Moreover, the function of checkpoint-restart measurement has been accomplished for effective rescheduling. The authors directed the research to provide effective contributions that afford better results.

Another approach presented in the literature [7] explains the different states attained by a task while accomplishing. It is stated that the coupling between the potential resources of faults and their impulsive manifestations in a grid environment in accumulation to the uncertainty of the resource state data and the service operating conditions advocate that deterministic models of fault estimations may be ineffective and inappropriate. However, the different states that were involved in the fault tolerant system are robust state, vulnerable state, failure state and maintenance state. Along with those descriptions, a work [8] demonstrates an approach for performance and reliability to examine various subtask distributions in a grid using tree-structure with the consideration of failure correlation

and data dependence. It has also been given in the paper that the process still needs enhancement and should adopt time-varying failure rate dynamics. A distinctive approach was proposed in the literature [9] for consistent collection of global checkpoints in a distributed system environment. The algorithm made every checkpoint to be under a consistent global checkpoint. This assists a process to take the orderly scheduled basic checkpoint at appropriate scheduled times. As another attempt [10], job scheduling on grid has been consummated, concerning the Service Level Agreement (SLA). Moreover, the process is for effective consumption of shared resources.

Employing the structure of the divide and conquer method admits to create a mechanism that has a lesser overhead and is lesser exploited than the common techniques such as checkpointing, given in the literature [11]. This work minimizes the amount of superfluous work done after a crash of one or more nodes. The work addressed in [12] presents a survey on ensuring grid system reliability. The authors have highlighted the efforts that focus on the distinct functional areas of grid systems.

- Reliability over computational hardware, software and data resources
- Reliability of management services and infrastructures
- Reliability of grid network for data transport and messaging

Moreover, the paper comprises different types of fault, fault detection methods and recovery methods using checkpoints. Checkpoint Processing and Recovery (CPR) is an efficient method that was proposed in the literature [13], trades adept register management is the general case in checkpoint overhead. Some special issues are given in other work [14] focusing on advanced scheduling strategies on grid computing. Further, another work [15] has explained the concept of developing a fault tolerant system for a computational grid environment with the determination of fault occurrence history, the number of checkpoints, etc. As stated in the paper, the major limitation of the work is the implementation of checkpoint replication service by obtaining the replica in a faster manner.

Fault tolerance is determined in terms of resource failure [16]. The strategy used to obtain fault tolerance is by relatively adapting the checkpoints that depends on the history of failure information and the current status of the resources. But still there is a necessity for enhancing the methodology for amending the efficiency and reliability on task scheduling in the cloud. Following that, another survey paper [17] describes the various issues in fault tolerance on the grid. In order to attain a high level of availability and reliability, the grid infrastructure should be an infallible fault tolerant. Since the failure of resources influences job execution mortally, fault tolerant service is vital to satisfy QoS constraints in grid computing. Commonly utilized techniques for bestowing fault tolerance are replication and job checkpointing. Further, dependable and efficient scheduling model was proposed in the literature [18] based on the credibility and availability of grid resources. With improved dependability, there is a great reduction in probability of service requests exploiting failure and deception. Checkpoint mechanism has also been used for fault tolerance. The enhancement would take into account a diversity of the service quality obligation of the user. The rescheduling algorithm was proposed in another work [19] that has a significant feature and termed as generic algorithm, since it can be used with a large array of rescheduling heuristics. It may also use more rescheduling approaches whose classification were elaborated in accordance to the node position in graphic and based on the fault tolerance mechanisms that can be used. Adaptive job scheduling methodologies with fault tolerance has been proposed [20]. Though there are many job scheduling algorithms in the grid, most of them focus on fine-grained jobs scheduling and others concentrate on only light weighted job scheduling methodologies. In the case of larger tasks and larger resources, there are only few scheduling algorithms available. It is also stated that the job scheduling in an unstable and dynamic grid environment is tolerable and also reduces the total execution time taken to accomplish all the jobs allocated in the grid. But there is no incorporation of checkpoints for recovering faults. A demonstration of security checkpoints is given in the literature [21].

Reliable Distributed Grid Scheduler (RDGS) has been developed [22] that enhances the successful schedule rate of the mixed tasks. The authors have considered the parameters such as priority and deadline for both the communication and computational intensive mixed tasks. Inclusion of resource discovery algorithms is given as the further migration of task on execution

In another aspect, primary static mapping based task migration has been explained in the literature [23]. Thereby, they have achieved high performance in computational grid. Though the mechanism maximizes the throughput, there is a need of concerning more on delay times and the environmental conditions. Further, on focusing the eminence of checkpoints in fault tolerance, a fine-grained incremental checkpoint has been implemented in another work [24]. This could reduce the checkpoint file size considerably without other overhead. With the checkpoint interval mounting, the differences of checkpoint file mass between two schemes are very petite for most of workloads.

3. PROPOSED MECHANISM

The grid model conceded in this paper consists of geographically deployed computational nodes, resources and a number of services for effectively

executing tasks that are to be assigned from process repositories. As the predominant intention of this paper is to schedule tasks for the grid nodes that can efficiently accomplish the allotted tasks, fault tolerance of the nodes are mainly considered based on its past history. Further, the adaptation of checkpoint technique is for enhancing the reliability over grid environment. The checkpoint data is made persistent throughout the process. The Information Server (IS) collects and maintains the resource and job status information required by the checkpointing procedure and the job scheduler. It prolongs the history of information about every resource. Moreover, by this approach the stability and reliability of each node dispersed under a grid environment is examined with its failure rate and checkpoint assumptions in adept job scheduling process.

3. 1. Failure Rate Estimation According to the past history of the execution capability of grid nodes, the failure rate is estimated here. Let us proceed with the following assumptions and derivations.

f_i= No. of Failures/ (Overall Execution Time), $i = \{1, 2, 3, \ldots, N\}$ where N be the number of nodes in the grid.

t= total time taken for execution at a node

$$\left(1 - f_i\right)^t = e^{-f_i t_i} \tag{1}$$

By taking the log:

$$t \log\log\left(1 - f_i\right) = -f_i * t_i * \log e \tag{2}$$

Differentiating with respect to‘t’:

$$\frac{1}{f_i} = -\frac{(\log e)}{\log\left(1 - f_i\right)} = a\left(Constant\right) \tag{3}$$

With those derived equations reliability rate of each and every node at time t is determined. Thus, the Node Reliability (NR_t) at time is given as:

$$NR_{t_i} = e^{-f_i t_i} \tag{4}$$

where ‘f’ represents the failure rate. Following that, the mean time to get a failure (or) chance of failure (F) of each node is estimated by:

$$F = \int_0^\infty NR_{t_i} = \int_0^\infty e^{-f_i t_i} \tag{5}$$

This case occurs while the machine getting failure unexpectedly. Moreover, it is given that the evaluation based on the average time between two successive failures. In accordance with that, if a failure occurs, the completion time of a particular job will be exceeded that expected rate.

3. 2. Inceptive Checkpointing Rather than logging the occurrences, checkpointing confides on sporadically saving the state of the computation to constant storage. If a failure occurs, the computation is restarted from one of the formerly saved states. Since the computation is distributed, the tradeoff space on local and global checkpointing scenarios and their resulting recovery cost are to be considered. Thus, checkpointing based approaches differ in the way that the processes are coordinated and in the source of a consistent global state, wherein the consistent global state can be attained either at the time of rollback recovery or at the time of checkpointing. The following Figure 3 demonstrates the failure classification with respect to recoverability.

The operation of checkpointing on a resource, executing a single job is diagrammatically represented in Figure 4, which is explained below:

Phase 1. The job is submitted to the resource (r), after an execution interval I, the job executing on an active resource produces a checkpoint request.

Phase 2. For each resource, the process gets the last failure time from the information server, when no failure has occurred, the last failure time is initiated with the system start time.

Phase 3. The checkpoint request produced by the job is estimated by the scheduler. It is assumed that the resource is constant and the checkpoint is omitted to avoid the overhead.

Phase 4. To avert too many checkpoint omissions, a maximum number of omission limits should be decided. Thus this approach reduces the checkpoint overhead by omitting the avoidable checkpoint.

Hence, inceptive checkpointing provides a capability to save an intermediary job state. It will be useful when long running jobs require storing some intermediate state to guard from node failures. Then, on reviving of a failed node, a job would load the saved checkpoint and persist from where it left off.

3. 3. Node Stability Assessment In this process, the stability of the grid nodes is assessed with some constraints such as the number of failures (n), expected processing time, elapsed processing time and the recovery rate. Moreover, fluctuation rate (FR) stands a significant role in the evaluation of stability and reliability of grid nodes for efficient job scheduling. The proposed models accommodate the failure occurrence of a grid node in Poisson distribution. Let $i = \{1, 2, 3, \ldots, N\}$, where N is the number of nodes, each follows a Poisson distribution, and also are assumed as random variables.

Regarding this case, consider the distribution function $D(x)$, where x is the recovery rate. Here, the definition of recovery rate (x) is given at the time taken by a node to complete a particular job, after failure that concerns the rate of change of fi. For appropriate computations, Erlang distribution is considered. It is a general type of statistical distribution that is related to the beta

distribution and arises consistently in the process for which the nodes having relevant waiting times. Moreover, it generalizes the distribution functions for attaining real number computation results. It is given as:

$$D(x) = P(X_i \le x_i) = 1 - P(X_i \le x_i) \quad (6)$$

Further, the major thing is to estimate the fluctuation rate of expected time completion while the constraint of failures is fixed or allowed with respect to the time of completion. The assumptions to be considered in computing the fluctuation rate are as follows:

Let x_1 and x_2 be the expected recovery rates with respect to the job accomplishment to expect processing time and the elapsed processing time. Following, t_1 and t_2 are the expected probability of completion time, f be the failure rates, and n_1 and n_2 are the number of failures occurred while processing, obtained in the fore mentioned manner. With those assumptions, the following evaluations take place in accordance with Erlang distribution.

$$X_{1i} \sim F(x_i, n_{1i,} f_i) = \int_0^{\infty} \left(\frac{1}{n1i!} \right) * e^{-f_i * x_{1i}} (f_i * x_{1i} * t_{1i}) \quad (7)$$

It is apparent from the conceit that failures may increase while the number of checkpoints increases, (i.e) failures may be beyond the checkpoints. Therefore, the proceeded equation X_2 will be higher than 1, which is subjected to follow Erlang distribution.

$$X_{2i} \sim F(x_i, n_{2i,} f_i) = \int_0^{\infty} \left(\frac{1}{n2i!} \right) * e^{-f_i * x_{2i}} (f_i * x_{2i} * t_{2i}) \quad (8)$$

The fluctuation rate of a node is obtained by solving Equations (7) and (8). It is given as:

$$Fluctuation\ Rate(FR) = e^{-f_{2i}} \left(\left(\frac{\left(e^{-t2i} * x_{2i} (f_{2i} * t_{2i} * x_{2i}) \right)^{n_{2i}}}{(n_{2i}+1) n_{2i}!} \right) + a - \left(\frac{\left(e^{-t1i} * x_{1i} (f_{1i} * t_{1i} * x_{1i}) \right)^{n_{1i}}}{(n_{1i}+1) n_{1i}!} \right) \right) \quad (9)$$

The fluctuation rate of all nodes in a grid environment is attained in this way. Using the results of the computation, the nodes are ranked.

3. 4. Ranking Function After the computation of fluctuation rates of all nodes, ranking would take place with respect to that. The rank generation process involves determining the ranking function R based on the fluctuation rate of a node. The FR is acquired from the processing time of a job with respect to the past history of workloads. However, it incorporates the selection of nodes having low fluctuation rates that enhances the performance of overall grid.

From the set of nodes N, the nodes with high stability and low fluctuations are obtained, ranking functions are employed. This supports effective scheduling. Then, the scheduling is performed by the scheduler for the jobs allotted from the process repository. Figure 5 shows the overall flow of the proposed work

Thus, the model has proven to be a good approximation for grid network behavior, while preserving short simulation times, relatively low computational complexity and attaining adept results. Presently, it is assumed that the proposed model is fully reliable and the fault tolerance can be effectively achieved using checkpointing and estimation of fluctuating rates.

4. EXPERIMENTAL RESULTS

In order to provide evidence for the proposed mechanism, the results are produced using GridSim. Further, to compare the performance of the affirmed checkpointing heuristics, system failure model and realistic workload are utilized. In this simulation strategy, the model parameters are initialized to symbolize a heavily loaded grid system with a job arrival pattern. For simulation, 400 jobs and 200 processor resources are considered. The resource requirements for the jobs are assumed between 1000 and 3000 MIPS (Million Instructions per Second).It represents a job which requires to be executed by a processor resource that is at least as capable as the resource requirement.

Further, the workloads for each job are randomly generated between 300,000 and 400,000 million instructions and the speed of processor resource is arbitrarily generated between 1000 and 3000 MIPS. There are four levels of grid nodes with every 500 MIPS. The comprehensive simulation factors are given in the following table:

Here, the fluctuation rate is defined as the difference between the expected processing time and the exceeded processing time. With those considerations, the evaluation of the proposed work is preceded. The following Figure 6 represents the correlation between the average checkpoint interval and the number of jobs that are successfully executed.

It is apparent from the graph that when the number of jobs increases, there is an increase in the checkpoint interval values. The values are adjusted in a manner to adapt with an effective saving of execution states to recover quickly, while occurring fault. Following that, Figure 7 presented below shows the relationship between the jobs and the number of checkpoints. Apparently, the number of jobs is directly proportional to the number of checkpoints fixed. Figure 8 presented above exemplifies the correspondence of nodes and its average fluctuation rate. Here, the fluctuation rate is calculated from Equation (9).

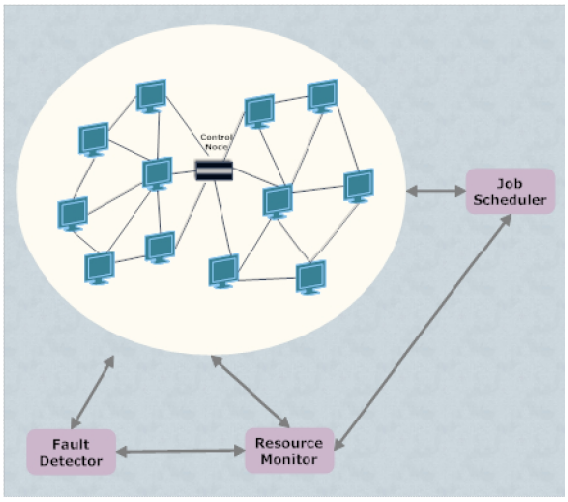

Figure 1. Sample grid environment

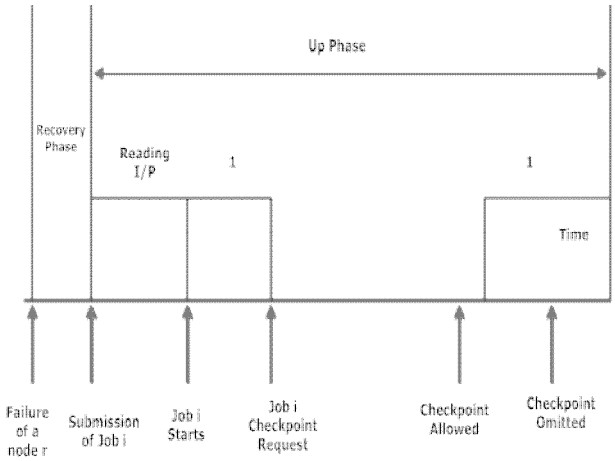

Figure 4. Resource runs a single job with checkpointing

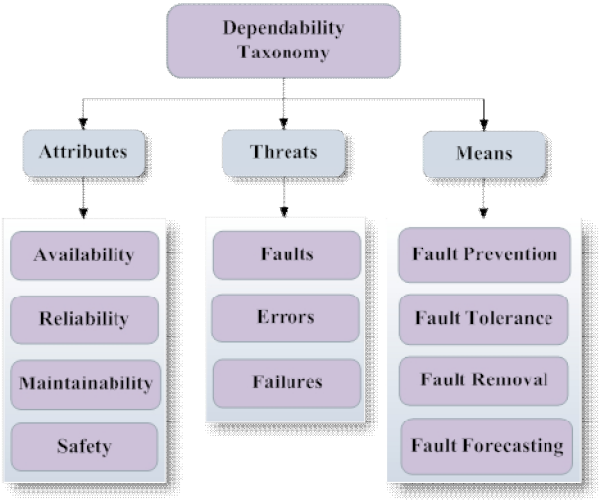

Figure 2. Dependability taxonomy

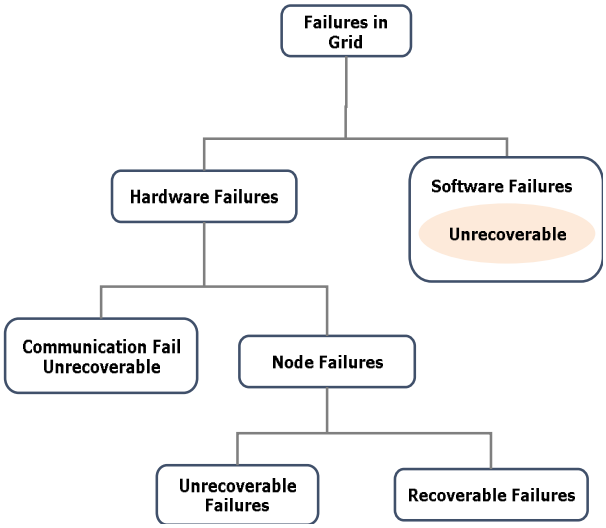

Figure 3. Classification of failures in grid systems with respect to reliability

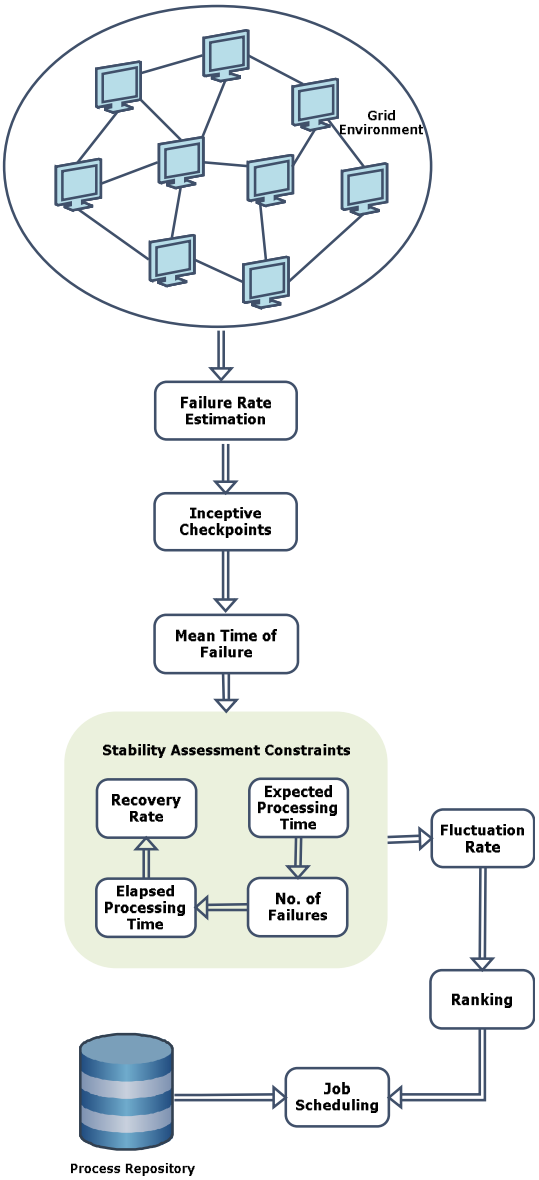

Figure 5. Overall flow of proposed work

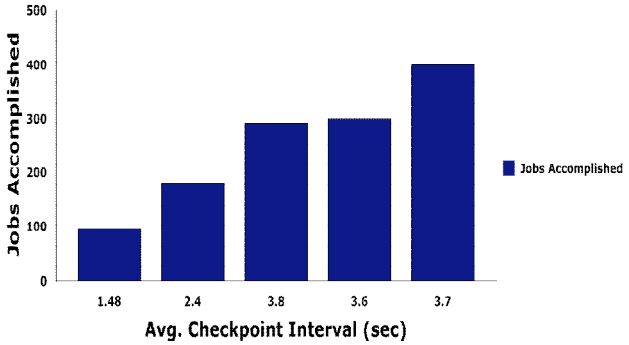

Figure 6. Jobs successfully accomplished

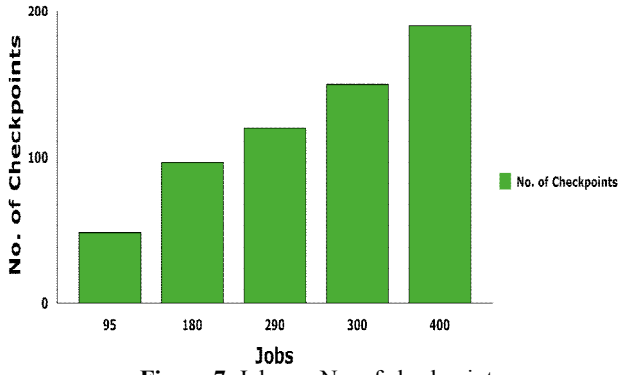

Figure 7. Jobs vs. No. of checkpoints

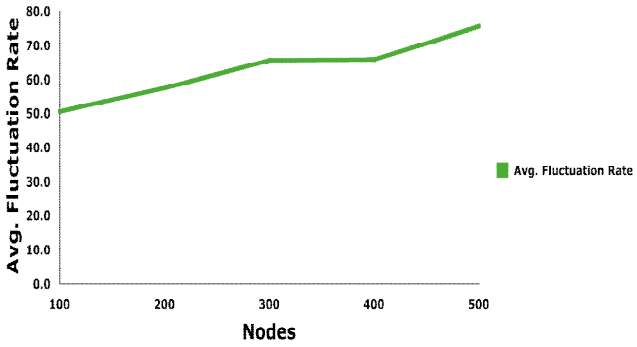

Figure 8. Jobs vs. average fluctuation rate

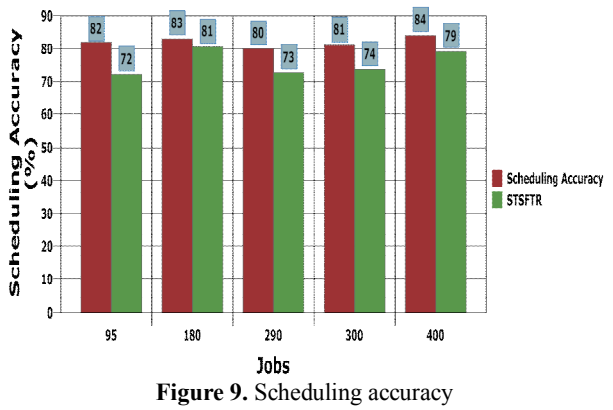

Figure 9. Scheduling accuracy

TABLE 1. Simulation parameters

Parameters	Values
Number of tasks	400
Number of processor resources	200
Resources requirements	1000-3000
Task workload	300000-400000
Processor resources speed (MIPS)	1000-3000

The FR is determined based on the recovery rates of the nodes, expected processing time and elapsed processing time. The graph is pointed upwards when there is an increase in the number of jobs to be processed with a particular node on the grid. Moreover, the stability of the nodes is evaluated with some stability assessment constraints that include the aforementioned parameters that involves in FR determination.

As the intention of this work, fault tolerance is mainly concerned for scheduling the jobs. After the computation of fluctuation rate, ranking of nodes is performed for effective scheduling. The nodes with low fluctuation rates are ranked first, which is followed by the nodes having a larger fluctuation rate. Based on the results of ranking, the jobs in the process repository are scheduled. Figure 9 reveals the comparison between the scheduling accuracy of the existing STSFTR [1] (Scheduling Tasks on most Suitable Fault Tolerant Resource) and the proposed model. It is apparent from the figure that the proposed model is more efficient in scheduling the tasks. This evidences the proficiency of the adduced SAMA.

5. CONCLUSIONS AND FUTURE WORK

Fault tolerance composes an imperative problem in all distributed environments. Here, the problem of fault tolerance has been addressed in terms of failure in job accomplishments. The SAMA dynamically involves failure rate estimation, inceptive checkpointing and stability assessment using constraints such as the number of failures, elapsed time, expected time and recovery rate. Following, fluctuation rate is determined and based on that ranking of nodes is performed for effective scheduling. Thus, the proposed work attains fault tolerance by strongly adapting the checkpoint frequency in accordance with the past history of job execution time and failure information, which tremendously reduces checkpoint overhead and enhances the throughput. However, the process reduces the migration cost, amends the reliability of the grid node and directs the job scheduler to allocate jobs from process repository to nodes in an adept manner.

With respect to future enhancement, we intend to work on considering various failures such as hardware failure, link failures, and developing an optimized scheduling framework.

6. REFERENCES

1. Naik, K. J., Kumar, K. V. and Satyanarayana, N., "Scheduling Tasks on Most Suitable Fault tolerant Resource for Execution in Computational Grid", ***International Journal of Grid & Distributed Computing***, Vol. 5, No. 3, (2012).
2. Latchoumy, P. and Khader, P. S. A., "Survey on fault tolerance in grid computing", ***International Journal of Computer Science & Engineering Survey (IJCSES) Vol***, Vol. 2, No., (2011).
3. Favarim, F., Fraga, J. S., Lung, L. C., Correia, M. and Santos, J. F., "Exploiting tuple spaces to provide fault-tolerant scheduling on computational grids", in Object and Component-Oriented Real-Time Distributed Computing, ISORC'07. 10th IEEE International Symposium on, IEEE. (2007), 403-411.
4. Plank, J. S. and Thomason, M. G., "Processor allocation and checkpoint interval selection in cluster computing systems", ***Journal of Parallel and distributed Computing***, Vol. 61, No. 11, (2001), 1570-1590.
5. Nandagopal, M. and Uthariaraj, V. R., "Fault tolerant scheduling strategy for computational grid environment", ***International Journal of Engineering Science and Technology***, Vol. 2, No. 9, (2010), 4361-4372.
6. Laadan, O., Phung, D. and Nieh, J., "Transparent checkpoint-restart of distributed applications on commodity clusters", in Cluster Computing, IEEE International, IEEE. (2005), 1-13.
7. Derbal, Y., "A new fault-tolerance framework for grid computing", ***Multiagent and Grid Systems***, Vol. 2, No. 2, (2006), 115-133.
8. Dai, Y.-S., Levitin, G. and Trivedi, K. S., "Performance and reliability of tree-structured grid services considering data dependence and failure correlation", ***Computers, IEEE Transactions on***, Vol. 56, No. 7, (2007), 925-936.
9. Jiang, Q. and Manivannan, D., "An optimistic checkpointing and selective message logging approach for consistent global checkpoint collection in distributed systems", in IPDPS. (2007), 1-10.
10. Sakellariou, R. and Yarmolenko, V., "Job Scheduling on the Grid: Towards SLA-Based Scheduling", in High Performance Computing Workshop. (2006), 207-222.
11. Wrzesińska, G., Van Nieuwpoort, R. V., Maassen, J., Kielmann, T. and Bal, H. E., "Fault-tolerant scheduling of fine-grained tasks in grid environments", ***International Journal of High Performance Computing Applications***, Vol. 20, No. 1, (2006), 103-114.
12. Dabrowski, C., "Reliability in grid computing systems", ***Concurrency and Computation: Practice and Experience***, Vol. 21, No. 8, (2009), 927-959.
13. Hilton, A., Eswaran, N. and Roth, A., "CPROB: Checkpoint Processing with Opportunistic Minimal Recovery", in Parallel Architectures and Compilation Techniques, 2009. PACT'09. 18th International Conference on, IEEE. (2009), 159-168.
14. Schulze, B. and Fox, G. C., "Advanced Scheduling Strategies and Grid Programming Environments, Wiley, (2010).
15. Das, A. and De Sarkar, A., "On fault tolerance of resources in computational grids", ***International Journal of Grid Computing & Applications***, Vol. 3, No. 3, (2012).
16. Therasa, S. and Lidya, A., "Dynamic adaptation of checkpoints and rescheduling in grid computing", ***International Journal of Computer Applications***, Vol. 2, No. 3, (2010).
17. Garg, R. and Singh, A. K., "Fault tolerance in grid computing: state of the art and open issues", ***International Journal of Computer Science & Engineering Survey (IJCSES)***, Vol. 2, No. 1, (2011), 88-97.
18. Suhasini, B., Reddy, D., Sathyalakshmi and Masillamani, R., "Dependable and Efficient Scheduling Model with Fault Tolerance Service for Grid Applications", ***International Conference on Computer Science and Information Technology***, (2011), 9-14.
19. OLTEANU, A., POP, F., DOBRE, C. and CRISTEA, V., Re-scheduling and error recovering algorithm for distributed environments. UPB Scientific Bulletin, Series C. (2011)
20. Dev, S. G., Sujith, R., V.S.Amirutha and S. Divyalakshmi, "An Adaptive Job scheduling methodologies with Fault Tolerance strategy for Computational Grid Environment", ***International Conference on Computing and Control Engineering***, Vol. 12, No. 13, (2012).
21. Nam, H., Kim, J., Hong, S. J. and Lee, S., "Secure checkpointing", ***Journal of Systems Architecture***, Vol. 48, No. 8, (2003), 237-254.
22. Kovvur, R. M. R., Ramachandram, S., Kadappa, V. and Govardhan, A., "Reputation Aware Reliable Distributed Grid Scheduler for Mixed Tasks", ***International Journal of Computer Applications***, Vol. 43, No., (2012).
23. Delavar, A. G., Boroujeni, A. R. K. and Bayrampoor, J., "A Balanced Scheduling Algorithm with Fault Tolerance and Task Migration based on Primary Static Mapping (PSM) in Grid", ***International Journal of Computer Applications***, Vol. 52, No., (2012).
24. Li, X. and Lu, K., "FCKPT: A fine-grained incremental checkpoint for PCM", in Computer Science and Network Technology (ICCSNT), International Conference on, IEEE. Vol. 3, , (2011), 2019-2023.

Stability Assessment Metamorphic Approach (SAMA) for Effective Scheduling based on Fault Tolerance in Computational Grid

A. Shamila Ebenezer [a] , K. Baskaran [b]

[a] *Department of Computer Science and Engineering, Karunya University, Coimbatore, India.*
[b] *Department of Computer Science and Engineering,Government College of Technology, Coimbatore, India*

P A P E R I N F O

Paper history:
Received 16 May 2013
Received in revised form 05 September 2013
Accepted 07 November 2013

Keywords:
Checkpointing
Grid Computing
Fault Tolerance
Recovery Rate
Scheduling

چکیده

محاسبات شبکه ای اجازه به اشتراک گذاری هماهنگ و کنترل شده منابع و حل مساله در چند موسسه و سازمان های مجازی پویا را می دهد. علاوه بر این، به دلیل ماهیت غیر قابل اعتماد منابع شبکه، تحمل خطا و وظیفه برنامه ریزی، یک مسئله مهم برای شبکه محاسباتی در مقیاس بزرگ است. روش معمول بهره برداری برای تحقق بخشیدن به تحمل خطا، محرکه تناوبی است که به صورت دوره ای موجب صرفه جویی در کار می شود. اما فاصله زمانی نقطه بررسی نامناسب موجب تاخیر و کاهش توان عملیاتی در این کار می گردد. با توجه به این مسأله، این مقاله برای اطمینان از عملکرد بهتر در شبکه های محاسباتی با سیستم قابل تحمل خطای موثر و قابل اعتماد، با استفاده از روش جدید ثبات ارزیابی دگرگونی (*SAMA*) تلاش می کند. در اینجا، از استراتژی که با تغییراتی اندک در نقاط بررسی و با توجه به وضعیت فعلی و اطلاعات مربوط به خرابی گذشته در مورد منابع به صورت پویا، که در سرور اطلاعات وجود داشت، برای رسیدن به تحمل خطا استفاده شده است. فرایند برنامه ریزی موثر را می توان با اغماض بر اساس برنامه ریزی خطا که شامل تعیین میزان انحراف از تمام گره ها با استفاده از برخی محدودیت های ارزیابی با ثبات بالا به دست آورد. این روش به روشنی نشان داد که کار در طول زمان مقرر و با توان بهبود یافته انجام می شود و راهی قابل اعتماد برای ساخت محیط شبکه است.

doi: *10.5829/idosi.ije.2014.27.05b.04*