THINK INDIA JOURNAL

ISSN:0971-1260
Vol-22- Issue-14-December-2019

A Categorical Review of Dynamic Task Scheduling, Load Balancing, and Fault Tolerance Practices for Computational Grid

[1]Sophiya Sheikh, [2]Mohammad Nayeem
[1] Central University of Rajasthan, [2] Osmania University
[1] sophiya.sheikh@gmail.com, [2] nayeemsap@gmail.com

***Abstract*— A computational grid is premeditated to process substantial computational applications. Numerous users residing at diverse geographical locations are requesting services from the grid architecture which possesses heterogeneous resources. This chapter presents a review of dynamic task scheduling (DTS) mechanisms, advance reservation techniques and fault-tolerant approaches those have already been recorded in the research to optimize distinct QoS parameters and grid architecture. Scheduling has already been proved as NP-complete in distributed systems. Therefore, many investigations have been nominated in the literature that can be classified into various categories of heuristics such as nature-inspired, dynamic, heterogeneous, job scheduling and load balancing, etc. Moreover, they are implemented for different kind of tasks for instance batch of tasks (BoT), priority based tasks, independent tasks, and workflow applications.**

Index Terms— Computational Grid, Dynamic Task Scheduling, Fault-Tolerance, Advance Resource Reservation, Grid Systems.

I. INTRODUCTION

The computational grid provides a significant contribution in optimized task scheduling on multiple resources to minimize various QoS parameters. The primary motivation behind proficient task scheduling is to optimize the power of the grid with respect to scheduling parameters. However, if the grid system is dynamic with respect to resources, the chances of error occurrences or faults to appear are anticipated to intensify during task execution [1]. Therefore, it is essential to design an efficient dynamic scheduling algorithm to sustain in such an environment. In a dynamic grid system, job scheduler does not possess any information regarding the job size and processing time required to complete the job. Besides, there is no assertion that once a job has been assigned to a resource then it will be processed by the same resource without any interruption [2]. Similarly, in dynamic task scheduling, a job can start consuming system resources to accomplish the request assigned by the user at any time and can migrate to any other favorable resources within the system. Therefore, faults are more likely to arise [3].

In addition, the grid is a geographically dispersed architecture; there could be numerous possible reasons for the resources to leave a task under processing resulting task execution to be interrupted. Whereas, the primary need for a grid system is to accomplish the task timely. For the purpose, whenever a resource has left any task, some other resource will be assigned to that task so as to make the system adaptive, and fault tolerant. Therefore, fault tolerance is a challenging concern observed in a grid environment [4].

Further, advanced resource reservation is a technique in which tasks are com- mitted to the resources preliminarily for failsafe task execution. The ultimate aspiration of advance reservation (AR) is that the system supplies the required resources on time. Likewise, it also conquers the resource for the expected task up to an estimated duration, computed from the assessment of task completion time. However, for some reasons, if any resource left the task in between execution for a reported interval, the system should provide an alternative resource for that period else the system is considered to be failed [5].

Figure-1 demonstrates various categories of dynamic task scheduling for QoS parameter optimization. Moreover, it shows a number of advance reservation techniques realized in several job scheduling, load balancing approaches and it is designed on the basis of job execution time. In addition, the figure illustrates different categories of fault tolerance in different environments. These techniques are discussed in detail as follows:

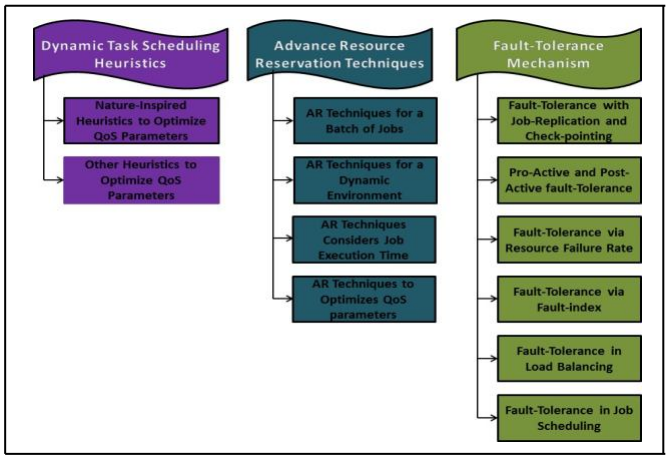


Fig. 1. DTS, AR, and FT Categories

Copyright © 2019Authors

ISSN:0971-1260
Vol-22- Issue-14-December-2019

II. DYNAMIC TASK SCHEDULING TECHNIQUES

Efficient scheduling techniques are necessary to achieve instantaneous, accurate, and reliable consequences by exploiting computational resources in a collective manner. Further, at any instance, the task may enter into the system for execution and after accomplishing task processing, the task may correspondingly leave the system at any point with no restriction from any resource or system. The task can autonomously migrate from a resource to another during its execution. In such an environment, it is extremely challenging to ensure task completion on time and uninterruptedly. Thus, in order to achieve a consistent system in such a dynamic environment, a dynamic task scheduling (DTS) algorithm was an essential requirement, so the DTS process has come into existence. Several QoS parameters can be optimized by variants of DTS in existing literature are as follows:

A. Nature-Inspired Heuristics to optimize QoS Parameters

A number of nature-inspired DTS algorithms are designed to optimize TAT and response time. In [6] agent based job scheduling algorithm is introduced that minimizes TAT, waiting time and response time and also ensures resource scalability. In [7] an enhanced version of ant colony optimization is implemented as a solution to scheduling problems occurred in the computational grid. In [8] a task scheduling scheme is proposed which is based upon the partitioning of memory, resources and both optimize makespan time. In this chain, a gravitational search algorithm [9] has been suggested to optimize makespan time and flowtime. In [10] a dynamic job scheduling algorithm is invented which implies the combination of multiple heuristics such as genetic algorithm, simulation annealing, and tabu search. In order to provide better throughput, an advanced version of an existing ant colony algorithm is developed in [11]. In [12] the author introduced a hyper-heuristic approach for resource appointment using particle swarm optimization to execute the job efficiently and securely. In [13] the author proposed a hierarchical genetic scheduler for a dynamic grid system that works for the independent tasks to minimize makespan and flowtime. Similarly, a genetic algorithm is used in [14] to maximize the best resource available to solve complicated expansion problems.

B. Other Heuristics to optimize QoS Parameters

Further, some other heuristics have also been proclaimed in the literature to minimize the task execution time. In [15] Parallelized Dynamic Task Scheduling approach is constructed for BoT to optimize various QoS parameters and discern with the exploitation of task parallelism. Similarly, a DTS is announced to optimize various grid parameters by promoting the task level parallelization for multiple tasks and users [16]. In [17] the author designed a sort-mid heuristic to minimize the makespan time and maximize the resource utilization for a dynamic system. Some traditional approaches are also suggested for the purpose. Opportunistic Load Balancing (OLB) is an arbitrary task allocation technique that has no concern with task execution time. Therefore, OLB only opportunistic balanced load and failed to minimize TAT [18] [19]. Other strategies are Without Grouping Algorithm (WGA) and Grouping Algorithm (GA). WGA executes each independent task separately, therefore, increasing values of time and cost parameters for job transmission and job processing. Further, WGA is an arbitrary resource allocation and execution technique which increases overheads in time and cost. To take over the limitations of WGA, a dynamic and efficient task grouping algorithm is invented. Grouping algorithm transmitting and executing tasks in order to revamp different time and cost parameters based on resource capacity [20].

Although various approaches have been proclaimed in the literature to solve the purpose, none of them can be advertised with both revamp in grid parameters and resource utilization when the system is dynamic [5] [21]. However, considered static and dynamic methods like OLB, Grouping Algorithm and WGA are not feasible for concerned aspects. The mentioned techniques not only lack in load distribution but also retain giant task waiting time, therefore, failed dreadfully to achieve the proposed objective.

III. FAULT-TOLERANCE TECHNIQUES

Grid is a dynamic system that supports several of dispersed resources that are heterogeneous in nature and performs millions of user tasks. However, resources are not permanently inside the grid periphery, they have the opportunity to join or leave the system at any time. Thus, fault tolerance is considered to be a critical issue in grid systems. In addition, significant benefits can be achieved by implementing fault tolerance in grid computing. Further, it assures task execution even in case of any unexpected scenario such as resource failure or disconnection from the system or task migration or due to any unanticipated user operation.

A. Fault Tolerance using Job Replication and Check-Pointing

The most frequently practiced fault tolerant techniques are job replication and job check-pointing in grid architecture as discussed in [4]. Likewise, the author computed related parameters for the performance evaluation of the grid system. The standard performance matrices such as grid load, throughput, failure tendency and turnaround time were used to estimate the benefits of considered fault tolerance approaches. However, job replication and check-pointing may interrupt job execution if incongruous replica quantity and check pointing figures are picked by the system [22]. Therefore, the author introduced various heuristics that dynamically and efficiently generate job replication and check-pointing statistics depending on the grid prominence. Furthermore, the proposed Dynamic Scheduling in Distributed Environments (DSiDE) is a fault tolerance algorithm designed for dynamic systems to optimize throughput even if faults are reported in the system.

Moreover, in [23] the author introduced a middleware that provides replication services and thus reproducing similar data at distinct grid sites. Besides, the middleware establishes data synchronization by TCP/IP communication protocol

 Copyright © 2019Authors

ISSN:0971-1260
Vol-22- Issue-14-December-2019

among various heterogeneous and dispersed nodes of the grid system. A similar study in the field of data replication has been reported in [34] where the author implements an effective fault-tolerant mechanism related to replication that achieves security demand and maintains trust level of resources. As per the security requirement of grid architecture, the quantity of replicated datasets can be varied adaptively. On the other hand, [25] describes a rescheduling technique that is designed for independent jobs. Further, failed jobs can be rescheduled by using task replication services.

Similarly, the author implemented a fault-tolerant approach for depended and independent tasks to optimize the performance of grid resources while ensuring reliability [26]. To achieve the purpose, this research proposed recovery/rollback through a checkpoint or restart for dependent tasks.

B. Post-Active and Pre-Active Fault Tolerant

The principal requirement of the dynamic grid system is task has to accomplish within a given deadline [27]. However, network or resource failure prevents the system to do so. In [28] author introduced two novel approach one is post-active fault tolerant and another is proactive fault tolerant to handle the faults before or after job scheduling. The post-active technique was easier to implemented and designed by using the job monitoring technique. The resource with minimum response time will be selected for task execution. Therefore, a higher chance of resource selection is established that may fail repeatedly. While proactive strategy takes preventive measures if a fault occurs before or after scheduling, therefore, it holds more information about the grid resources. But, that leads to reallocation of a job so a decrease in throughput. Therefore, in [29][30] the author designed an effective approach which enhances the fault-tolerance rate and throughput.

The literature discussed above divulges that all the resources participate in any computational grid have fair chances of failure due to internal or external factors concerned with their operation of availability. Therefore, a proactive fault tolerant strategy for job scheduling is necessary for uninterrupted performance of the grid systems. The conferred literature [31] [32] [33] [34] which employed proactive scheduling schemes for fault recovery uses a fault index of grid resources. The fault index can be computed by counting the history of fault occurrence in the resources participated in the considered grid system. Whenever an expected task is not executed on time due to a fault in any resource, then fault index will be increased.

C. Fault-Tolerance via Resource Failure Rate

Furthermore, in [35] the author shows his concern towards the publically available raw data on failures by one of the largest high performance computing sites. The author analyzed the statistics of the failure data with the basic reasons for the collapse. It also reports the duration between failures and time taken to repair. The repairing time can vary from one system to another. Intense analysis of failure rate and repair time in the high-performance computing environment is also being discussed. Most of the scheduling strategies which have been designed to achieve fault tolerances are depending on either on fault index or on response time [36]. Further, the author introduced a novel factor i.e. scheduling indicator for resource selection. The scheduling indicator encompasses the resource failure rate and response time. The proposed factor is essentially used by the grid scheduler in order to make the job scheduling decisions. It is designed to opt for the resources for allocation which has the lowest tendency of resources failure. In [37] author presented an idea that short jobs can be executed from the scratch if they failed for any reason, however, fault tolerance can be achieved for the long-running and big jobs containing a million lines of instructions by using check-pointing approach. Though vast research has been made in the field of cost optimization and drops in task execution time, however, the researches may not have considered the resource failure rate and user estimated deadline for job accomplishment [39]. Therefore, in [40] author distinguishes classes of autonomous tasks by fault-tolerant scheduling approach. The proposed strategy encompasses two algorithms such as MCT-LRC and MRC-ETC established to minimize replication-cost and accomplishment-time respectively. However, it miscarried to deal with a fault rate of grid resources [41]. Thus, a resource management service is introduced for fault-tolerance that not only manages the system failure but also optimizes necessary parameters.

D. *Fault-Tolerance via Fault Index*

The fault-tolerance strategy in [42] includes adaptive task check-pointing and has efficiently implemented for the economic grid, it retains the fault-index of grid resources. The scheduling decisions will be generated by considering response time and fault-index of the resources. Another fault-tolerance technique is specifically designed for a dynamic environment that functions by reading the failure information history of grid resources and frequency of check-pointing to improve the system throughput [43]. For the purpose, the author implemented a job rescheduling schemes i.e. fault index-based rescheduling (FIBR) that handles job failure, by re-executing the failed job from last checkpoint information. However, fault-index is considered to be an incompetent factor to estimate resource failure tendency as compared to the failure rate.

Fault tolerance has also been implemented for the workflow applications by the author in [44] to prevent failures. The reasons of failure occurrences are diversions in system resources, unexpected breakage of networks, failure in computational resources operations, inadequate load distributions among resources, insufficient memory for executions, etc. An effective grid workflow management system is designed to detect and correct failures, and assure task execution on time. Further, at the application level, various fault tolerance techniques are discussed for grid and cluster environments Li2006.

In addition, a survey of numerous fault tolerant techniques is discussed in [38] that incorporate a number of probable mechanisms for fault tolerance such as job-replication, job-migration, and check-pointing. Moreover, it provides various optimizations such as time, cost and feasible processing parameters.

Copyright © 2019Authors

ISSN:0971-1260
Vol-22- Issue-14-December-2019

E. Fault-Tolerance in Load Balancing

In [46], the author implemented a fault tolerant practice that emphasizes on load balancing. It also deals with the optimization of various other characteristic features such as communication delay, resource availability, job characteristics, grid architecture, network bandwidth, and resource certainty. Hence, the author proposed AlgHybrid LB that implements the specification of cluster-based and neighbor based load distribution algorithms to improve re- source utilization and minimize response time without communication overhead. Sometimes fault recognition and transmission need a wider context in computational grid [47]. Thus, the author designed a fault-tolerant grid structure that delivers the essential to handle the confined faulty performance of hosted services. The proposed research has been taken several grid architecture circumstances into account for example ambiguity issues due to the dispersed fault information and asynchronous nature of communication. This research work actually improvised decision-making strategies for a decentralized fault-tolerant architecture.

F. Fault-Tolerance in Job Scheduling

In [54], the author presented an intense discussion of most frequently used fault tolerance technique (FTT) introduced in a grid architecture. It comprises alternate resource, task, retrying and check-pointing. For the performance evaluation, the author considered a number of matrices such as network delay, turnaround time, throughput and waiting time. Further fault tolerance scheduling technique has also been applied in the schedule generation of direct acyclic graph [49]. The reported heuristic considered communication delay to escape service failure in the occurrence of resource faults. Further, research is implemented with the combination of [49] and a job scheduling algorithm i.e. minimum total time to release (MTTR). In order to make scheduling decisions, scheduler considered resource response time and fault index into account. A distributed fault-tolerant scheduling (DFTS) algorithm [50] designed to work with job replication and job scheduling. The author estimated that grid is a collection of various sites and each site has its local scheduling manager that will perform as an alternative of another scheduling manager.

However, the genetic algorithm is taken into account for effective scheduling judgment to find out the novel computing capacity while faults occurred into the system. The performance evaluation considered system reliability and makespan. Another strategy is introduced for reduced makespan and fault tolerance in the grid system in [51]. For the purpose, the author implemented a novel bicriteria scheduling algorithm (BSA) to deal with the resource disaster and user satisfaction. The parameters are optimized within a given user deadline by pro- posed scheduling technique.

In [52], a new dynamic fault-tolerant technique is introduced which takes run- time decisions and overlooks checkpoint in order to minimize checkpoint over- head. However, it leads to the computational gain i.e. summation of the differences between fault-unaffected and fault-affected job response time [53].

IV. ADVANCE RESOURCE RESERVATION TECHNIQUES

When the system is dynamic, heterogeneous and distributed, we need to deal with some complex mechanism for efficient mapping of tasks to the resources. However, in the advance reservation (AR) technique, resources are reserved for the tasks up to an estimated time period. In literature, it has been implemented for various task categories such as a batch of tasks (BoT), priority based tasks, independent tasks, workflow applications, etc. Furthermore, different environ- mental conditions have been considered such as static, dynamic, centralized and distributed. Additionally, some of the methods in the literature are associated with optimization of various QoS parameters of the grid through the advance allocation of resources while other schemes are related to efficient job scheduling using prior resource reservation. In the reservation based system, the scheduler must be aware of the time interval for which a resource must be committed. Therefore, job start time estimation, job end time must be clearly notable so that they can be used for calculating resource allocation duration.

A. AR Techniques for a Batch of Tasks

In batch scheduling strategy, a review on the advance reservation has been drafted by the GRAAP survey. However, in this research, advance reservation is limited to providing the resources for a specific time period after negotiation in a static environment [54]. In [55] author presents a common model for scheduling a batch of computational tasks that have advance reservation requests for re- sources in order to accomplish the job. Moreover, the author proposed an online scheduling policy that minimizes the drawbacks of advance reservation technique on the schedule's quality and enhances resource utilization, flowtime, and tardiness. Besides, this research considered a static system model for the purpose that could not deal with the dynamic behavior of the resources or a dynamic system.

B. AR for a Dynamic Environment

Though various researches have been introduced to implement resource allocation in a dynamic environment, nevertheless it suffers performance degradation because of the joining and leaving of resources from the system at any given in- stance [56]. Thus, in [57] the author designed a virtual resource container (VRC) that enhances QoS deviation distance (QDD) which is based on logical resource selection algorithm to decrease the rejection ratio of reservation. Additionally, it is a multi-core advance reservation technique adapted in a grid environment. Although this mechanism is meant to work for distributed systems efficiently but failed to optimize various grid scheduling parameters, the primary need of the system. In this chain, Contention-aware multi-resource reservation to optimize QoS is presented in [58]. It optimized QoS parameters so some extent but it cannot be considered as an efficient scheduling scheme of dynamic systems. Furthermore, in [59], the author discusses an advance resource commitment and negotiation strategy that selects a backup resource from a cluster of available resources to ensure competent service to the request. On the other hand, an advance reservation that operates depending upon the task

 Copyright © 2019Authors

ISSN:0971-1260
Vol-22- Issue-14-December-2019

priority is also pro- posed in [60]. Likewise, in [61] the author handles parallel requests utilizing advance reservation mechanism. Resulting, various QoS parameters are optimized by exploiting parallelism in task execution. To achieve this purpose, the authors developed strict FCFS and easy backfill parallel scheduling algorithms.

Additionally, in certain scenarios, it is required to work with resource heterogeneity along with job scheduling policy. In this chain, the author investigates the influence of resource heterogeneity on reservation-aware schedulers [62]. The presented algorithm uses computational geometry to manage advance reservations. The simulation results show that resource heterogeneity may have a positive impact on the performance if it is considered in designing of the scheduling algorithm. Further, in [63] the author proposes the what-if algorithm for achieving resource reservations. For each reservation request, the algorithm inserts a place- holder job into the schedule of the LRMS. This action simulates the processing of the generated schedule to determine the job start time.

C. AR Considers Job Execution Time

On the other side, some approaches considered job finish time to reserve resources for the tasks respectively. A job's completion time within grid mainly depends on multiple factors such as transfer time, execution time and queue time, as predicted by multiple kinds of research. QBETS [64] is a batch queuing delay predictor service based on the Network Weather Service. It uses job history data to predict job start times within a user-defined confidence interval. However, QBETS has a major limitation i.e., it requires user input to specify a queue for prediction. Besides, in [65] the author utilizes different machine learning techniques for predicting queue wait time. For this purpose, different algorithms are presented that can be utilized in deriving future wait time with references from historical data. Still, this tends to lead to high processing requirements.

Likewise, based on grid workload traces, [66] presents the results of different data mining methods employed in predicting the job execution time and the queue wait time within grids. The simulation results show that the accuracy of job execution time prediction is below the average and the absolute prediction error is high. Moreover, there exists no prominent prediction method to be considered as most accurate, every prediction method may differ from site to site due to varying user behavior and the nature of the job and diverging system characteristics. Furthermore, in [67] the author demonstrated advance reservation using the earliest start time estimation with lower cost, higher accuracy, and scalability. Further, advance reservation in proposed strategy estimates job execution times by considering current utilization of grid resources. Since advance reservation is not easy to develop in current grid architecture due to its lack of support to heterogeneity and inefficiency to migrate a task among resources in case of resource failure. Hence, it is limited to obtain queue time predictions.

D. AR to Optimize QoS Parameters

In addition, a few works have also been disclosed in the literature using AR. Initially, AR via gridsim simulator has come into existence after [5]. Moreover, in [69] the author presented a similar model for both the scheduling of a batch of computational tasks and tasks having AR requests. In previous research of reservation, some AR negotiation strategy which is based upon selecting a backup resource from the cluster of available resource to ensure the quality of service for the request is introduced. To propose AR in a dynamic environment earlier, we have implemented a Dynamic Load Balancing using Advance Reservation of Resources (DLBAR) on Computational Grid [21]. However, DLBAR was formulated for adequate load distribution only using standard deviation and could not deal with minimization of execution time. Therefore, the above-suggested approaches could neither deal with the dynamism of the system nor with the optimization of QoS parameters in a computational grid.

Nevertheless, numerous researches have been proposed in the literature for advance reservation but none of the approaches is considered to be best. Besides, efficient grid architecture is specified by the author in [68] specifies that grid QoS should have service reliability, accountability of QoS parameters and service security from a performance perspective and service QoS and provisional QoS from the guaranteed perspective. Although, we have seen plentiful limitations in existing research, yet they managed to optimize the system in different scenario up to a certain level.

V. CONCLUSION

In this paper, various dynamic task scheduling procedures have been discussed that commit resources in a lively environment. Further, the time and cost effectiveness of the discussed method has been conferred by performing multiple un- biased comparisons with various state of the art techniques. In addition, various advance resource reservation techniques are discussed, that commit resources to the task before beginning the execution, for the uninterrupted task accomplishment. Moreover, grid scheduler provides an advance reservation of resource as per the demand of a user at any time. If the resources are not reserved in advance, they may quit the system at any point and related task execution will be failed. Therefore, a fault tolerant system is a significant need for the current grid environment. For the purpose, we studied several fault-tolerant practices existing in the literature. With advance resource reservation and fault tolerance approaches, we can acquire the meanest execution time while tasks are executed more efficiently and task completion is guaranteed even in the case of resource failure. It enables the grid to provide better processing and to deliver uninterrupted, ac- curate, reliable and faster results.

VI. REFERENCES

1. Foster, I., Kesselman, C., and Tuecke, S. (2001). The anatomy of the grid: En- abling scalable virtual organizations. The International Journal of High Performance Computing Applications, 15(3), 200-222.

 Copyright © 2019Authors

ISSN:0971-1260
Vol-22- Issue-14-December-2019

2. Krauter, K., Buyya, R., and Maheswaran, M. (2002). A taxonomy and survey of grid resource management systems for distributed computing. Software: Practice and Experience, 32(2), 135-164.

3. Xhafa, F., and Abraham, A. (2010). Computational models and heuristic methods for Grid scheduling problems. Future generation computer systems, 26(4), 608-621.

4. Altameem, T. (2013). Fault tolerance techniques in grid computing systems. Inter- national Journal of Computer Science and Information Technologies, 4(6), 858-862.

5. Sulistio, A., and Buyya, R. (2004, November). A grid simulation infrastructure supporting advance reservation. In 16th International Conference on Parallel and Distributed Computing and Systems (PDCS 2004) (Vol. 11, pp. 9-11).

6. Shah, S. N. M., Zakaria, M. N. B., Mahmood, A. K. B., Pal, A. J., and Haron, N. (2012). Agent based priority heuristic for job scheduling on computational grids. Procedia Computer Science, 9, 479-488.

7. Kumar, E. S., Sumathi, A., and Zubar, H. A. (2015). A hybrid Ant Colony Opti- mization algorithm for job scheduling in Computational grids.

8. Wu, M., and Sun, X. H. (2004, November). Memory conscious task partition and scheduling in grid environments. In Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing (pp. 138-145). IEEE Computer Society.

9. Zarrabi, A., and Samsudin, K. (2014). Task scheduling on computational grids using gravitational search algorithm. Cluster computing, 17(3), 1001-1011.

10. Abraham, A., Buyya, R., and Nath, B. (2000, December). Nature's heuristics for scheduling jobs on computational grids. In The 8th IEEE international conference on advanced computing and communications (ADCOM 2000) (pp. 45-52).

11. Sathish, K., and Reddy, A. R. M. (2008). Enhanced ant algorithm based load balanced task scheduling in grid computing. IJCSNS, 8(10), 219.

12. Aron, R., Chana, I., and Abraham, A. (2015). A hyper-heuristic approach for resource provisioning-based scheduling in grid environment. The Journal of Super-computing, 71(4), 1427-1450.

13. Kolodziej, J., and Khan, S. U. (2012). Multi-level hierarchic genetic-based scheduling of independent jobs in dynamic heterogeneous grid environment. Information Sciences, 214, 1-19.

14. Prakash, S., and Vidyarthi, D. P. (2015). Maximizing availability for task scheduling in computational grid using genetic algorithm. Concurrency and Computation: Practice and Experience, 27(1), 193-210.

15. Sheikh, S., Nagaraju, A., and Shahid, M. (2019). A Parallelized Dynamic Task Scheduling for Batch of Task in a computational grid. International Journal of Com- puters and Applications, 41(1), 39-53.

16. Sheikh, S., Shahid, M., and Nagaraju, A. (2017, December). A novel dynamic task scheduling strategy for computational grid. In 2017 International Conference on Intelligent Communication and Computational Techniques (ICCT) (pp. 102-107). IEEE.

17. Reda, N. M., Tawfik, A., Marzok, M. A., and Khamis, S. M. (2015). Sort-Mid tasks scheduling algorithm in grid computing. Journal of advanced research, 6(6), 987-993.

18. Braun, T. D., Siegel, H. J., Beck, N., B¨ol¨oni, L. L., Maheswaran, M., Reuther, A. I., ... and Freund, R. F. (2001). A comparison of eleven static heuristics for map- ping a class of independent tasks onto heterogeneous distributed computing systems. Journal of Parallel and Distributed computing, 61(6), 810-837.

19. Sheikh, S., and Nagaraju, A. (2017). A comparative study of task scheduling and load balancing techniques with mct using etc on computational grids. Indian Journal of Science and Technology, 10(32)

20. Muthuvelu, N., Liu, J., Soe, N. L., Venugopal, S., Sulistio, A., and Buyya, R. (2005, January). A dynamic job grouping-based scheduling for deploying applications with fine-grained tasks on global grids. In Proceedings of the 2005 Australasian workshop on Grid computing and e-research-Volume 44 (pp. 41-48). Australian Computer So-ciety, Inc..

21. Sheikh, S., Nagaraju, A., and Shahid, M. (2018). Dynamic Load Balancing with Advanced Reservation of Resources for Computational Grid. In Progress in Computing, Analytics and Networking (pp. 501-510). Springer, Singapore.

22. Chtepen, M., Claeys, F. H., Dhoedt, B., De Turck, F., Demeester, P., and Vanrolleghem, P. A. (2009). Adaptive task checkpointing and replication: Toward efficient fault-tolerant grids. IEEE Transactions on Parallel and Distributed Systems, 20(2), 180-190.

23. Srinivasa K, Siddesh G, Cherian S. Fault-tolerant middleware for grid computing. In: Proceedings of 12th IEEE international conference on high performance computing and communications, Melbourne, Australia, September 1–3; 2010. p. 635–40.

24. Jiang C, Xu X, Wan J. Replication based job scheduling in grids with security assurance. In: Proceedings of 3rd international symposium on electronic commerce and security workshops, Guangzhou, China, July 29–31; 2010.

25. Chtepen M, Dhoedt B, Cleays F, Vanrolleghem P. Evaluation of replication and rescheduling heuristics for gird systems with varying resource availability. In: Proceedings of 18th international conference on parallel and distributed computing systems, Anaheim, CA, USA, November 13–1,5; 2006. p. 622–27.

26. Garg, R., and Singh, A. K. (2014). Fault tolerant task scheduling on computational grid using checkpointing under transient faults. Arabian Journal for Science and Engineering, 39(12), 8775-8791.

27. Nandagopal M, Uthariaraj VR. Fault tolerant scheduling strategy for computational grid environment. Int J Eng SciTechnol 2010;2(9):4361–72.

28. Nazir B, Qureshi K, Khan FG. Adaptive checkpointing strategy to tolerate faults in economy based grid. J Supercomput 2009;50:1–18.

29. Jiang C, Xu X, Wan J. Replication based job scheduling in grids with security assurance. In: Proceedings of 3rd international symposium on electronic commerce and security workshops, Guangzhou, China, July 29–31; 2010.

30. Huda M, Schmidt H, Peake I. An agent oriented proactive fault-tolerant framework for grid computing. In: Proceedings of international conference on e-science and grid computing, Melbourne, Australia, December 5–8; 2005. p. 304–11.

 Copyright © 2019Authors

ISSN:0971-1260
Vol-22- Issue-14-December-2019

31. Chang RS, Chang JS, Lin PS. An ant algorithm for balanced job scheduling in grids. J Future Gen ComputSyst 2009;25:20–7.

32. Chunlin L, Xiu ZJ, Layuan L. Resource scheduling with conflicting objectives in grid environments: model and evaluation. J NetwComputAppl2009;32(3):760–9.

33. Huang P, Peng H, Lin P, Li X. Static strategy and dynamic adjustment: an effective method for grid task scheduling. J Future Gen ComputSyst, 2009;25(8):884–92.

34. Lu L, Yang S. DIRSS-G: an intelligent resource scheduling system for grid envi- ronment based on dynamic pricing. Int J Inform Technol 2006;12(4):120–7.

35. Schroeder, B., and Gibson, G. (2010). A large-scale study of failures in high- performance computing systems. IEEE transactions on Dependable and Secure Computing, 7(4), 337-350.

36. Amoon, M. (2012). A fault-tolerant scheduling system for computational grids. Computers and Electrical Engineering, 38(2), 399-412.

37. Mehta, J., and Chaudhary, S. (2008, December). Checkpointing and recovery mechanism in grid. In 2008 16th International Conference on Advanced Computing and Communications (pp. 131-140). IEEE.

38. Garg, R., and Singh, A. K. (2011). Fault tolerance in grid computing: state of the art and open issues. International Journal of Computer Science and Engineering Survey (IJCSES), 2(1), 88-97.

39. Buyya, R., and Murshed, M. (2002). A deadline and budget constrained cost-time optimisation algorithm for scheduling task farming applications on global grids. arXiv preprint cs/0203020.

40. Zheng, Q., Veeravalli, B., and Tham, C. K. (2007, June). Fault-tolerant scheduling for differentiated classes of tasks with low replication cost in computational grids. In Proceedings of the 16th international symposium on High performance distributed computing (pp. 239-240). ACM.

41. Lee, H., Park, D., Hong, M., Yeo, S. S., Kim, S., and Kim, S. (2009, August). A resource management system for fault tolerance in grid computing. In 2009 International Conference on Computational Science and Engineering (Vol. 2, pp. 609-614). IEEE.

42. Nazir, B., Qureshi, K., and Manuel, P. (2009). Adaptive checkpointing strategy to tolerate faults in economy based grid. The Journal of Supercomputing, 50(1), 1-18.

43. Therasa, S., and Lidya, A. (2010). Dynamic adaptation of checkpoints and rescheduling in grid computing. International Journal of Computer Applications, 2(3).

44. Wrzesinska, G., Van Nieuwpoort, R. V., Maassen, J., Kielmann, T., and Bal, H.E. (2006). Fault-tolerant scheduling of fine-grained tasks in grid environments. The International Journal of High Performance Computing Applications, 20(1), 103-114.

45. Li, H., Muskulus, M., and Wolters, L. (2006, June). Modeling job arrivals in a data- intensive grid. In Workshop on Job Scheduling Strategies for Parallel Processing (pp. 210-231). Springer, Berlin, Heidelberg.

46. Balasangameshwara, J., and Raju, N. (2012). A hybrid policy for fault tolerant load balancing in grid computing environments. Journal of Network and computer Applications, 35(1), 412-422.

47. Derbal, Y. (2006). A new fault-tolerance framework for grid computing. Multiagent and Grid Systems, 2(2), 115-133.

48. Khan FG, Qureshi K, Nazir B. Performance evolution of fault tolerance techniques in grid computing system. J ComputElectrEng 2010;36:1110–22.

49. Zheng Q, Veeravalli B. On the design of communication-aware fault-tolerant scheduling algorithms for precedence constrained tasks in grid computing sys- tems with dedicated communication devices. J Parallel Distributed Comput 2009;69:282–94.

50. Abawajy J. Fault-tolerant scheduling policy for grid computing systems. In: Pro- ceedings of 18th IEEE international parallel and distributed processing symposium, April 26–30; 2004.

51. Keerthika, P., and Kasthuri, N. (2013). An efficient grid scheduling algorithm with fault tolerance and user satisfaction. Mathematical problems in Engineering, 2013.

52. Oliner, A. J., Rudolph, L., and Sahoo, R. K. (2006, June). Cooperative checkpoint- ing: A robust approach to large-scale systems reliability. In Proceedings of the 20th annual international conference on Supercomputing (pp. 14-23). ACM.

53. Katsaros, P., Angelis, L., and Lazos, C. (2007). Performance and effectiveness trade-off for checkpointing in fault-tolerant distributed systems. Concurrency and Computation: Practice and Experience, 19(1), 37-63.

54. MacLaren, J. (2003). Advance reservations: state of the art, GGF GRAAP-WG. See Web Site at: http://www. fz-juelich. de/zam/RD/coop/ggf/graap/graap-wg. html, Aug.

55. Park, J. (2004). A deadlock and livelock free protocol for decentralized internet resource coallocation. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 34(1), 123-131.

56. Liu, Y., Liu, X., Xiao, L., Ni, L. M., and Zhang, X. (2004, March). Location-aware topology matching in P2P systems. In IEEE INFOCOM 2004 (Vol. 4, pp. 2220-2230). IEEE.

57. Luo, J., Wu, Z., Cao, J., and Tian, T. (2012). Dynamic multi-resource advance reservation in grid environment. The Journal of Supercomputing, 60(3), 420-436.

58. Xu, D., Nahrstedt, K., and Wichadakul, D. (2001). QoS and contention-aware multi-resource reservation. Cluster Computing, 4(2), 95-107.

59. Li, C., Xiao, N., and Yang, X. (2003, December). Selection and advanced reservation of backup resources for high availability service in computational grid. In International Conference on Grid and Cooperative Computing (pp. 26-33). Springer, Berlin, Heidelberg.

60. Min, R., and Maheswaran, M. (2002, May). Scheduling co-reservations with priorities in grid computing systems. In 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'02) (pp. 266-266). IEEE.

61. Chen, J., Li, B., and Wang, E. F. (2014). Parallel scheduling algorithms investigation of support strict resource reservation from grid. In Applied Mechanics and Materials (Vol. 519, pp. 108-113). Trans Tech Publications.

62. Castillo, C., Rouskas, G. N., AND Harfoush, K. (2008, April). Efficient resource management using advance reservations for heterogeneous grids. In 2008 IEEE International Symposium on Parallel and Distributed Processing (pp. 1-12). IEEE.

63. ROblitz, T., and Rzadca, K. (2006, August). On the placement of reservations into job schedules. In European

 Copyright © 2019Authors

ISSN:0971-1260
Vol-22- Issue-14-December-2019

Conference on Parallel Processing (pp. 198-210). Springer, Berlin, Heidelberg.

64. Nurmi, D., Brevik, J., and Wolski, R. (2007, June). QBETS: queue bounds estimation from time series. In Workshop on Job Scheduling Strategies for Parallel Processing (pp. 76-101). Springer, Berlin, Heidelberg.
65. Li, H., Chen, J., Tao, Y., Gro, D., and Wolters, L. (2006, May). Improving a local learning technique for queuewait time predictions. In Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID'06) (Vol. 1, pp. 335-342). IEEE.
66. Sonmez, O., Yigitbasi, N., Iosup, A., and Epema, D. (2009, June). Trace-based evaluation of job runtime and queue wait time predictions in grids. In Proceedings of the 18th ACM international symposium on High performance distributed computing (pp. 111-120). ACM.
67. Liang, F., Ma, S., Luckow, A., and Schnor, B. (2012). Advance Reservation-based Computational Resource Brokering using Earliest Start Time Estimation. JCP, 7(6), 1329-1336.
68. Al-Ali R, ShaikhAli A, Rana O,Walker D (2003) Supporting QoS-based discovery in service-oriented grids. In: International parallel and distributed processing symposium (IPDPS'03), Nice, France, 2003
69. Depoorter, W., Vanmechelen, K., and Broeckhove, J. (2014). Advance reservation, co-allocation and pricing of network and computational resources in grids. Future Generation Computer Systems, 41, 1-15.

Copyright © 2019Authors