International Journal of Grid and Distributed Computing
Vol. 10, No. 12 (2017), pp.117-124
http://dx.doi.org/10.14257/ijgdc.2017.10.12.11

Dynamic ACO-based Fault Tolerance in Grid Computing

Saufi Bukhari[1], Ku Ruhana Ku-Mahamud[2] and Hiroaki Morino[3]

[1,2]School of Computing, College of Arts and Sciences, Universiti Utara Malaysia, 06010, Sintok, Kedah, Malaysia
[3]Graduate School of Engineering and Science, Shibaura Institute of Technology, 3-7-5 Toyosu, Koto, Tokyo 135-8548, Japan
saufi@ahsgs.uum.edu.my, ruhana@uum.edu.my and morino@shibaura-it.ac.jp

Abstract

Scheduling jobs in distributed conditions of grid computing is nearly impossible to have a completely fault-free system. It is important to integrate fault tolerance capability in the system so that the system can continue to run even in the presence of failure in addition to improving the scheduling process as well as reducing the possibility of faults. Typically, load balancing is not considered in the presence of failure and this may lead to an inefficient scheduling process despite having a good fault tolerance strategy. This paper presents an ant-based fault tolerance algorithm that used checkpoint and resubmission techniques with consideration of execution history in the pheromone updating process to enhance fault tolerance capability. Experimental results showed that the proposed algorithm has better performance as compared to other relevant algorithms in terms of execution time, success rate, and average turnaround time per job.

Keywords: *Grid Computing, Job Scheduling, Fault Tolerance, Ant Colony Optimization, Ant Colony System*

1. Introduction

Grid computing is the collection of computer resources located in different locations that work together to complete assigned tasks. It has been widely used in solving challenging problems in the real world, such as protein folding [1, 2], hydrology modelling [3], and natural disasters simulation [4]. The main reason for deploying grid computing is to introduce a system that is scalable, simple to use, autonomic, and able to deal with faults [5]. Grid computing emerged from meta-computing back in 1990s to support diverse online processing and data intensive application [6, 7, 8].

In typical distributed computing systems that involve parallel computation such as grid, cluster, and cloud computing, there are many shared resources to process submitted jobs, and it is common for failure to occur during job processing. There are many types of failure that can possibly occur such as network failure, packet loss and corruption, physical failure to the CPU, hard drive and storage drive in the processing machine, user termination, service and protocol failure, software failure, and processing failure [9]. Out of the most common types of failure, network and processing failures are possible to be resolved in real time through proper fault tolerance strategies. If failure happens in the processing machine, users will experience a delay in response time [10]. This is because submitted jobs cannot be processed effectively and the resource will not be released to process subsequent jobs in the queue. Thus, stagnation will occur where the throughput will be greatly decreased or totally stalled due to the limited resources available to process the jobs in queue.

Received (July 16, 2017), Review Result (October 21, 2017), Accepted (November 26, 2017)

ISSN: 2005-4262 IJGDC

In minimizing this risk, fault tolerance should be applied effectively to identify failures accurately, support reliable execution in the presence of failures, and achieve great potential of computational grids [11, 12]. Fault tolerance is defined as a Nondeterministic Polynomial (NP)-complete problem [13], which means that there is no exact algorithm that can solve them in polynomial time [14, 15]. The most popular way to solve these problems is to use approximate (heuristic) algorithms such as Genetic Algorithm (GA) [16], Simulated Annealing (SA) [17], Tabu Search (TS) [18], and recently Ant Colony Optimization (ACO) [19]. GA, SA, and TS are some of the local search heuristic algorithms used to search a solution space by moving one solution to another and constructing the best solution for scheduling and load balancing algorithms. A feasible solution is quickly produced by using these methods, but it will not come close to the optimal solution. Fault tolerance should also be dynamic to compliment with the dynamic nature of a distributed system [20]. The most effective fault tolerance algorithm should be able to make decisions on when to apply preventive measures during runtime in order to reduce overhead.

This paper proposes dynamic ACO-based fault tolerance (DAFT) in grid computing extended from the Ant Colony System (ACS) scheduling algorithm. Related works are covered in Section 2 and followed by details on the proposed algorithm in Section 3. Experimental results are further explained and analyzed in Section 4, while the conclusion and future work are presented in the last section.

2. Related Works

Load balancing using enhanced ACO was proposed by [21] to effectively balance job allocation to all available resources. During initialization, the initial pheromone update will be calculated for each combination of job and resource and will be stored in a table called Pheromone Value (PV) matrix. A single ant will be populated for each job and will search for the highest pheromone value in the PV matrix to assign the task. Once the task execution is done, a pheromone update will be applied to all entries associated to the current resource. This approach has increased resource utilization and reduced average completing time per job.

Trust-based ACO (TACO) for grid resource scheduling was proposed by [22] with specific goals of job completion time reduction and workload balancing. They proposed a mechanism called trust as a form of incentive for success or penalty for failure and it is being applied during the pheromone update process. The higher the trust, the higher the resource reliability, and this will eventually lead to a lower possibility of failure during job execution. In terms of failure handling mechanism, job resubmission to other available resources is incorporated to ensure that all the failed jobs are completely executed in the end. However, all the failed jobs are required to be reprocessed from the beginning due to the lack of checkpoint technique.

[23] proposed a hybrid algorithm where ACS is combined with Directed Acyclic Graph (DAG) to enhance the job scheduling process. Using the DAG method, all tasks are sorted according to their dependency before undergoing the resource allocation process using ACS where ants will search for the optimal path for each combination of task and resource. The execution will take place in sequential order and both local and global pheromone updates are used to balance the load. The proposed algorithm was claimed to reduce the failure rate but without proper recovery action, it is not very effective in handling failure despite its ability to reduce the possibility of failure.

Hybrid ACO was proposed by [24], where they combined ACO with GA to overcome performance degradation due to the uncontrolled nature of metaheuristics of ACO. In their proposed algorithm, GA is used to determine whether to increase or decrease the pheromone update parameters. Resource selection is done randomly by ants before forming subsets. Each subset will be evaluated using the GA algorithm to spot the lowest

estimated error and will be sorted ascendingly. The subset with the lowest estimated error will be assigned with tasks and the pheromone update will be applied to the chosen subset once the execution is completed. Resources within the best subset will have higher probability of getting tasks assigned. The results showed that the proposed algorithm able to reduce possibility of failure but could be further extended by including the recovery technique and load balancing approach.

Fault tolerance ACO (FTACO) using checkpoint was proposed by [25] to solve fault and load balancing problems. The checkpoint technique is applied in the ACO-based scheduling algorithm to find the optimal resource as well as detect the occurrence of failure during job execution. Failure history is stored in the fault index manager, which is used as a reference in the next job assignment. Based on this information, jobs are rescheduled to other optimal resources from the last saved state instead of from the beginning. In addition to that, load balancing is also considered during the rescheduling process, where resources with low workload will have a higher possibility of getting selected as compared to resources with high workload. The amount of workload is determined by the pheromone value of each resource.

Job scheduling with fault tolerance in grid computing using ACO was proposed by [26] that combines resubmission and checkpoint techniques in addition to consideration of fault index manager to determine the checkpoint interval for each resource. The fault index manager is controlled by the status of job processing where either the success or failure count will be incremented accordingly. The results showed that controlling the occurrence and interval of checkpoints can enhance the performance of fault tolerance system based on checkpoint in terms of makespan, throughput and average turnaround time.

ACO has been one of the most effective solutions for dynamic scheduling in distributed systems. Due to its adaptability, it is possible to integrate other techniques such as job resubmission and checkpoint to further enhance its capability in providing fault tolerance. A consideration of execution history would complement the fault tolerance capability by providing more effective scheduling decision to reduce the turnaround time and failure rate without disregarding the performance of the system.

3. Dynamic ACO-Based Fault Tolerance in Grid Computing

DAFT is inspired by the ability of ants searching for new paths when the current path is blocked or is no longer an optimal path. This concept is useful when it comes to handling faults in grid computing especially when the system needs to make decisions dynamically to re-assign the task to another suitable resource. To compliment this functionality, the checkpoint technique as proposed by [25] is also considered so that the failed task does not need to be executed from the beginning. The failed task can be executed from the last saved state. Furthermore, the use of resource execution history as inspired by trust factor proposed by [22] is essential to influence the pheromone evaporation for each resource so that the resource that does not perform well will be less attractive during task assignment.

Figure 1 illustrates the high-level process workflow of DAFT. For each task, an ant will be generated to perform resource selection based on the resource pheromone value. The initial pheromone value will first be calculated to determine the state of all resources before the first task in queue can be submitted. The selection of resource will be based on the amount of pheromone value either from the initial pheromone calculation or pheromone update process [21]. Once a resource is assigned with task, global pheromone update will be applied by the ant to reduce the amount of pheromone. Each task will be divided into several checkpoints which will be executed in sequence to preserve the authenticity of the output. In each failed checkpoint or complete task execution, the local pheromone update will be applied to reduce the pheromone intensity based on execution history before releasing the resource for the next execution. In case of any failure during execution, the last checkpoint will be resubmitted to another suitable resource.

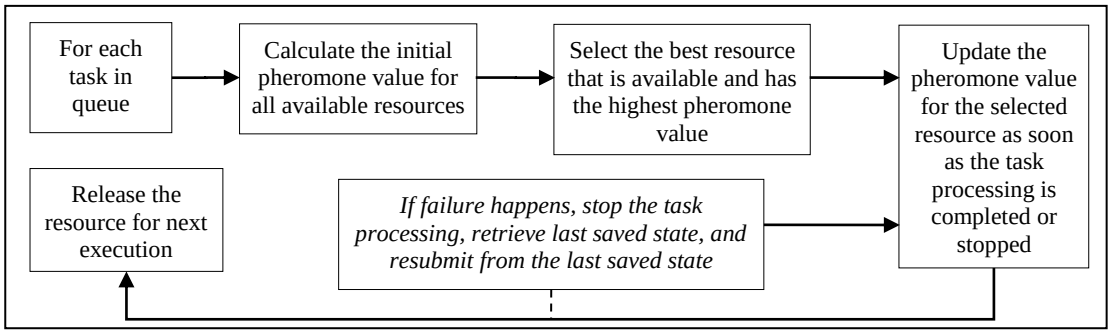

Figure 1. High-Level Workflow of DAFT

In the proposed algorithm, the local pheromone update is further extended to consider resource execution history so that less pheromone is evaporated should the resource complete task execution or more pheromone is evaporated otherwise. The extended global pheromone update formula (1) is given as follows:

$$\tau_{ij} = (1-\rho)\tau_{ij} + \rho.\tau_0(E_i) \tag{1}$$

where τ_{ij} is the current pheromone for combination of resource i and task j, ρ is the evaporation rate, τ_0 is the initial pheromone value of resource i, while E_i is the newly proposed parameter to represent execution history of resource i and is defined by (2):

$$E_i = \frac{\sum CP_{success}}{\sum \left(CP_{failed} + CP_{success}\right)} \tag{2}$$

where $CP_{success}$ indicates the current successful checkpoint call, while CP_{failed} is the current failed checkpoint at resource i respectively. The execution history (or defined as resource fitness) will be used to influence the quantity of pheromones to be deducted at a particular resource and eventually helps ants to make the best decision during task assignment; the better the execution history, the higher the number of task assigned by ants [27].

4. Experimental Result and Discussion

Experiments are conducted to measure the performance of the proposed algorithm in terms of execution time, success rate, and average turnaround time per task. The proposed DAFT algorithm is compared with TACO [22] and fault tolerance FTACO [25] because the proposed techniques are based from these two algorithms. During initialization, each resource will be assigned a fitness rate using the pseudo random algorithm within a defined range. However, this value is only used to simulate the occurrence of failure, while the task assignment by ants relies on the actual execution history. This approach is important because in the real situation, the initial fitness of a resource is not known in the beginning. As the execution goes on, the fitness will slowly be revealed in the form of execution history.

Table 1. Simulation Parameters

Parameter	Value
Number of Tasks	5000
Number of Resources	100
PE Rating	50 MIPS
Bandwidth	5000 B/S
Machine per Resource	1
PE per Machine	5

To accurately measure the effect of error range to the performance, the computation capability for each resource is set to be the same as shown in Table 1. However, in real case, the computation capability may differ in each resource and will influence the resource assignment process. For both TACO and FTACO, the evaporation rate is fixed at 0.5 which is a typical value of evaporation rate. However, for DAFT, the evaporation rate is set to 0.0004 which gives the most balanced resource allocation after thorough experiments. The error range defines the minimum and maximum resource fitness so that it does not fall outside the defined range in each experiment. It consists of 100% (0 error range) up to 50% (0.5 error range) fitness as assumed by [10] with 10% (0.1) incremental range to effectively measure the effect.

Figure 2 shows that the execution time is not affected by the increase of error range for DAFT as compared to TACO and FTACO with slight increment. Despite the good performance of DAFT algorithm, it does not prove that the execution time will be affected significantly by the increase of error range due to the small number of tasks and its size.

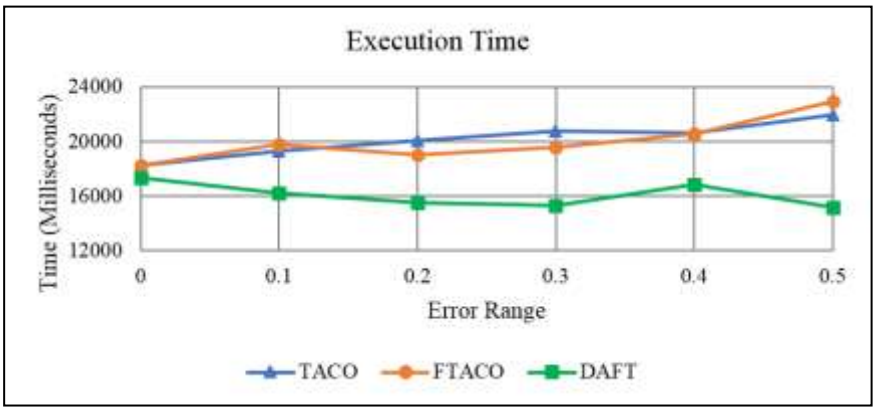

Figure 2. Results of Execution Time

The average turnaround time per task is equivalent to the average processing time per task. In this experiment, time refers to the simulator clock time and is not equivalent to real time. As shown in Figure 3, at 100% fitness rate (0 error range), the average turnaround time for all simulated algorithms is relatively similar. However, it increases significantly along with the error range increment. The results also suggest that the increment rate for DAFT is lower as compared to TACO and FTACO. It is proven that by combining trust factor from TACO and checkpoint technique from FTACO can lead to lower average turnaround time per task.

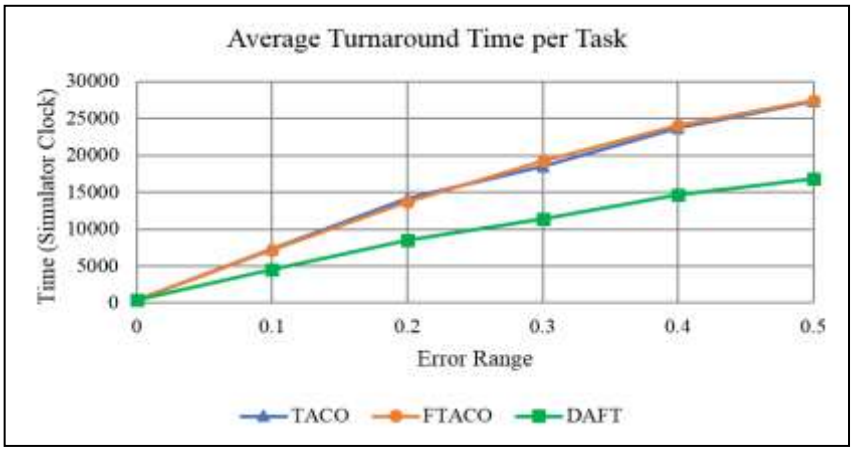

Figure 3. Results of Average Turnaround Time per Task

The execution success rate is measured by the total percentage of all successful checkpoints over total checkpoints. As shown in Figure 4, DAFT has the highest success rate, which is approximately 85% at 0.5 error range. Comparing the trend of success rate and average turnaround time per task in Figure 3, it can be hypothesized that the higher the success rate, the lower the average turnaround time. High success rate would reduce the possibility of execution failure and lead to lower time needed to process individual task. In contrast, low success rate would lead to higher possibility of execution failure and eventually affect the time to completely process individual task due to task resubmission process.

Figure 4. Results of Execution Success Rate

5. Conclusion

The consideration of execution history during pheromone update process and the application of checkpoint technique in the proposed DAFT algorithm provide significant improvement to the processing time as well as execution success rate as compared to TACO and FTACO. By having the execution history as a parameter to influence the pheromone intensity in each resource, the load balancing aspect is indirectly controlled through effective task assignment based on resource fitness. On the other hand, checkpoint technique allows the failed task to be reprocessed from the last saved state instead of from the beginning which will lead to lower time needed to completely process each task. Despite the good performance of DAFT, it can still be further improved by considering other factors such as fatal resource failure, priority of submitted task, and timeout of task processing.

Acknowledgment

The authors wish to thank the Ministry of Higher Education Malaysia in funding this study under the Trans Disciplinary Research Grant Scheme (TRGS), S/O code 13164, and Research and Innovation Management Centre, Universiti Utara Malaysia, Kedah for the administration of this study.

References

[1] A. Natrajan, M. Crowley, N. Wilkins-Diehr, M.A. Humphrey, A.D. Fox, A.S. Grimshaw and C.L. Brooks, "Studying protein folding on the grid: Experiences using CHARMM on NPACI resources under Legion", Concurrency and Computation: Practice and Experience, vol. 16, no. 4, **(2004)**, pp. 385–397.
[2] K.A. Dill and J.L. MacCallum, "The protein-folding problem, 50 years on", Sci., vol. 338, no. 6110, **(2012)**, pp. 1042–1046.

[3] G. Lecca, M. Petitdidier, L. Hluchy, M. Ivanovic, N. Kussul, N. Ray and V. Thieron, "Grid computing technology for hydrological applications", J. Hydrology, vol. 403, no. 1-2, **(2011)**, pp. 186–199.
[4] E. Pajorova and L. Hluchý, "Visualization the natural disasters simulations results based on grid and cloud computing", in Emerging Informatics – Innovative Concepts and Applicat., S.J. Miah, Ed., InTech, **(2012)**, pp. 85–100.
[5] K. Qureshi, F.G. Khan, P. Manuel and B. Nazir, "A hybrid fault tolerance technique in grid computing system", J. Supercomputing, vol. 56, no. 1, **(2011)**, pp. 106–128.
[6] I. Foster and C. Kesselman, "The grid in a nutshell", in Grid Resource Management, J. Nabryski, J. M. Schopf, and J. Weglarz, Eds., Springer US, **(2004)**, pp. 3–13.
[7] A. Moallem and S.A. Ludwig, "Using artificial life techniques for distributed grid job scheduling", Proc. of the 2009 ACM Symp. on Appl. ComputingHawaii, United States of America, **(2009)**.
[8] S. Sotiriadis, N. Bessis, F. Xhafa and N. Antonopoulos, "From meta-computing to interoperable infrastructures: A review of meta-schedulers for HPC, grid and cloud", IEEE 26th Int. Conf. on Advanced Inform. Networking and Applicat., Fukuoka, Japan, **(2012)**.
[9] D. Rakheja, P. Kaur and A. Rkheja, "Performance evaluation of resource scheduling and fault tolerance in grid", Int. J. Comput. And Commun. Syst. Eng., vol. 1, no. 1, **(2014)**, pp. 15–19.
[10] M. Amoon, "A fault tolerance scheduling system based on checkpointing for computational grids", Int. J. Advanced Sci. and Technology, vol. 48, **(2012)**, pp. 115–124.
[11] M. Nandagopal and V.R. Uthariaraj, "Fault tolerant scheduling strategy for computational grid environment", Int. J. Eng. Sci. and Technology, vol. 2, no. 9, **(2010)**, pp. 4361–4372.
[12] P. Keerthika and N. Kasthuri, "An efficient fault tolerant scheduling approach for computational grid", Amer. J. Appl. Sci., vol. 9, no. 12, **(2012)**, pp. 2046–2051.
[13] C. Glaßer, A. Pavan and S. Travers, "The fault tolerance of NP-hard problems", in Language and Automata Theory and Applicat., A.H. Dediu, A.M. Ionescu, and C. Martin-Vide, Eds., Springer Berlin Heidelberg, **(2009)**, pp. 374–385.
[14] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison", J. ACM Computing Surveys, vol. 35, no. 3, **(2003)**, pp. 268–308.
[15] Z. Pooranian, M. Shojafar, J.H. Abawajy and M. Singhal, "GLOA: A new job scheduling algorithm for grid computing", Int. J. Artificial Intell. and Interactive Multimedia, vol. 2, no. 1, **(2013)**, pp. 59–64.
[16] F. Werner, "Genetic algorithms for shop scheduling problems: A survey", Preprint, vol. 11, no. 31, **(2011)**, pp. 1–66.
[17] S.W. Lin and F.Y. Vincent, "A simulated annealing heuristic for the team orienteering problem with time windows", European J. Operational Research, vol. 217, no. 1, **(2012)**, pp. 94–107.
[18] F. Glover and M. Laguna, "Tabu Search*", in Handbook of Combinatorial Optimization, P. M. Pardalos, D.Z. Du, and R.L. Graham, Eds., Springer New York, **(2013)**, pp. 3261–3362.
[19] K.R. Ku-Mahamud and M.M. Alobaedy, "New heuristic function in ant colony system for job scheduling in grid computing", in Mathematical Methods for Information Science and Economics, N. Mastorakis, E. Zaitseva, D. Randjelovic, K.K.F. Yuen, C.G. Carstea, S. Capusneanu and A. Larion, Eds., Montreux: WSEAS, **(2012)**, pp. 47–52.
[20] J. Balasangameshwara, "Survay on job scheduling, load balancing and fault tolerance techniques for computational grids", Global J. Technology and Optimization, vol. 6, no. 1, **(2014)**, pp. 169.
[21] H.J.A. Nasir, K.R. Ku-Mahamud and A.M. Din, "Load balancing using enhanced ant algorithm in grid computing", 2nd Int. Conf. on Computational Intell., Modelling and Simulation, Bali, Indonesia, **(2010)**.
[22] H. Wenming, D. Zhenrong and W. Peizhi, "Trust-based ant colony optimization for grid resource scheduling", 3rd Int. Conf. on Genetic and Evolutionary Computing, Guilin, China, **(2009)**.
[23] V. Modiri, M. Analoui and S. Jabbehdari, "Fault tolerance in grid using ant colony optimization and directed acyclic graph", Int. J. Grid Computing and Applicat., vol. 2, no. 1, **(2011)**.
[24] S. Mandloi and H. Gupta, "Adaptive job scheduling for computational grid based on ant colony optimization with genetic parameter selection", Int. J. Advanced Comput. Research, vol. 3, no. 9, **(2013)**, pp. 66–71.
[25] T. Prashar, Nancy and D. Kumar, "Fault tolerant ACO using checkpoint in grid computing", Int. J. Comput. Applicat., vol. 98, no. 10, **(2014)**, pp. 44–49.
[26] H. Idris, A.E. Ezugwu, S.B. Junaidu and A.O. Adewumi, "An improved ant colony optimization algorithm with fault tolerance for job scheduling in grid computing systems", PLOS ONE, vol. 12, no. 5, **(2017)**, pp. 1–24.
[27] S. Bukhari, K.R. Ku-Mahamud and H. Morino, "Fault tolerance grid scheduling with checkpoint based on ant colony system", J. Comput. Sci., vol. 13, no. 8, **(2017)**, pp. 363–370.

Authors

Saufi Bukhari, he received his BIT (Hons.) Security Technology from Multimedia University, Malaysia. He is currently pursuing his postgraduate study in Computer Science at Universiti Utara Malaysia. Besides his study, he has been working for several years as IT Systems Analyst at a semiconductor chip maker. His research interest includes parallel and distributed computing, bio-inspired algorithm, load balancing and fault tolerance.

Ku Ruhana, she holds a Bachelor in Mathematical Sciences and a Master's Degree in Computing, both from Bradford University, United Kingdom in 1983 and 1986 respectively. Her PhD in Computer Science was obtained from Universiti Pertanian Malaysia in 1994. As an academic, her research interests include computer systems performance modeling, ant colony optimization and intelligent agent.

Hiroaki Morino, he received B.E, M.E. and Ph.D. degrees from the University of Tokyo in 1994, 1996 and 1999, respectively. He is currently an associate professor of Shibaura Institute of Technology, Japan. His research interests include the MAC protocols and routing protocols of wireless multihop networks, reliable peer-to-peer networks and mobile computing.