Survey on Fault-Tolerance-Aware Scheduling in Cloud Computing

Chesta Kathpal and Ritu Garg

Abstract Nowadays, to a large extent, clients look at cloud not just as service provider but also as partner. So, they want cloud to deliver timely and accurate services. Cloud nodes must be reliable in order to provide quality of services as per the customer requirements. Further, physical size of high-performance computing environment is also increasing day by day. Larger the system, more failures are likely to occur that eventually results in the poor reliability of the system which is highly undesirable for the time-critical applications. To deal with the reliability, service provider must know the failure characteristics of the cloud computing nodes in order to better handle the failure using fault-tolerance-aware techniques at the time of scheduling the application tasks. Thus, in this paper, we presented the survey of fault-tolerance-aware techniques which are classified as proactive and reactive fault tolerance. This survey provides the foundation for the researchers to work in the area of fault-tolerance-aware scheduling in order to have better scheduling decisions with the aim to enhance the performance and reliability of application execution.

Keywords Reliability · Fault tolerance · Virtualization

1 Introduction

Cloud is an Internet-based computing paradigm that provides basic services as Infrastructure as a Service (IaaS), Software as a Service (SaaS), Platform as a Service (PaaS) [1]. Different types of cloud providers, i.e., public, private, or hybrids, are responsible for providing above services to user. Nowadays, usage of

C. Kathpal (✉) · R. Garg
Department of Computer Engineering, National Institute of Technology, Kurukshetra, Kurukshetra, Haryana, India
e-mail: Chestakathpal93@gmail.com

R. Garg
e-mail: ritu.59@gmail.com

© Springer Nature Singapore Pte Ltd. 2019

S. Fong et al. (eds.), *Information and Communication Technology for Competitive Strategies*, Lecture Notes in Networks and Systems 40,
https://doi.org/10.1007/978-981-13-0586-3_28

cloud computing is increasing because of its enormous features such as sharing of resources, on-demand resource provisioning and virtualization. As users need to pay for required resources on basis of pay as you go model, cloud service provider must provide reliable services as per the QoS [2] requirements of the users. Although cloud computing makes the computing very reliable, dynamic, fast, and easy, it is still facing numerous challenges due to its large-scale and complex architecture. Reliability is the key challenge that needs careful attention.

Reliability and performance of applications in cloud environment depend upon many factors like scheduling of application tasks on resources and occurrence of failure of resources. Aim of scheduling algorithm is to assign the tasks to the available resources with the objective to maximize the execution performance and reliability of an application. To achieve these goals, numerous research works have been carried out in the past on scheduling problems. As we know, scheduling problem is NP-hard in nature, different approaches like heuristics, meta-heuristics, or approximations are the possible solutions. Because of failure of resources, reliability of application execution is decreasing that affects the financial losses to both service providers and users. For performance improvement, it is essential to maximize the reliability of a system. For the same purpose, understanding of foundering and rectifying attributes of cloud resources is very crucial. Garraghan et al. [3] put efforts in knowing the failure characteristics of wide-scale cloud applications. They investigated on the Google Cloud Trace Log for measuring the failure characteristics of cloud servers and tasks, comprising more than 12,500 servers that took 29 days to operate. They observed that the failures' rate of application tasks follow different theoretical distribution like Lognormal or Weibull. Further, analysis on distributed systems [4–6] indicates that the computing resources have Poisson failure distribution.

2 Fault-Tolerance-Aware Techniques in Cloud

To improve the reliability of application execution, failures [7] can be handled either in proactive manner or reactive manner. In case of proactive fault tolerance, it is handled before the occurrence of fault.

2.1 Proactive Fault Tolerance

In cloud computing environment, to minimize the effect of failure in proactive manner, tasks are scheduled to reliable virtual machines according to prior failure information. Various reliability-aware scheduling techniques are proposed in literature for scheduling the tasks in order to maximize the reliability of application execution as shown in the Fig. 1.

Fig. 1 Taxonomy of fault tolerance aware scheduling techniques

Predictive-Based Reliability-Aware Techniques.

Historic Log Basic. In Ref. [5], authors proposed the scheduling of tasks considering reliability of processors and communication links under the *Poisson failure distribution* [8] of networks. In Ref. [9], proactive approach is proposed for resource reliability considering three parameters: CPU (MIPS), memory (RAM), and bandwidth (BW). The system model finds out the reliability of each virtual machine and then assigns the cloudlets to best reliable VM using prior information like available memory, available MIPS ratio, and available bandwidth ratio. *Weibull failure distribution* is considered in Ref. [10] for maximizing the reliability by scheduling the tasks to reliable VM.

Monte Carlo Estimation Technique. Monte Carlo Failure Estimation algorithm is developed to investigate the future patterns for scheduling the tasks by estimating the failure of virtual machines using Weibull failure distribution in cloud. In Ref. [11], to schedule the tasks to various virtual machines, Failure-Aware Resource Scheduling (FARS) algorithm is proposed in which the reliability is taken into

consideration to map the tasks in workflow application. This mechanism reduces the schedule length of FARS because more reliable virtual machines are always selected for execution.

Mapping Strategy-based Reliability-Aware Techniques. As the problem of scheduling in cloud belongs to the class of NP-Hard problems, heuristic and meta-heuristics are the preferred choices. The heuristic-based approaches [10] are problem specific and may not find near-optimal solution. Meta-heuristic approaches are used to handle problem of local optima.

Heuristic Approach. In Ref [12], a heuristic algorithm is developed to achieve one objective, i.e., maximizing reliability under the constraint of another objective, i.e., minimizing end-to-end delay (EED) for distributed computing systems. The reliability is maximized under time bound constraint for mapping the task to processor. Mapping scheme is different from previous centralized mapping scheme where only one central server collects all the information related to networks and decides about mapping, but in this approach all the distributed severs participate in taking the decision of mapping to achieve scalability while keeping the status of failure rates. In Ref. [4], a simulated annealing algorithm is developed for scheduling the tasks to processors to maximize the system reliability and evaluated its performance in comparison with branch-and-bound technique with satisfactory results.

Meta-Heuristic Approach. In Ref. [5], authors developed a scheduling algorithm using meta-heuristic approach along with the maximization of the reliability. NSGA-II approach is used with ENLU technique to avoid applying the non-domination sorting from scratch, and ENLU helps in sorting the solution by taking the advantages of existing knowledge of current population for scheduling the applications on system. Also maintain the energy consumption and reliability of system. The main advantage of the proposed algorithm is that if we scale up then it improves the performance of the system.

Criteria-Based Reliability-Aware Technique.

Single Objective. In Ref. [10], hill-climbing heuristic approach is applied within particle swarm optimization (PSO) in order to maximize the system reliability. This algorithm outperforms the genetic algorithm (GA) because particles can communicate with each other which is the main feature of PSO.

Multiobjective. Various multiobjective algorithms [13, 14] have been developed based on multiple QoS parameters that include reliability, energy consumption, makespan, deadline, scalability. In Ref. [15], the bi-objective genetic algorithm (BOGA) is developed for heterogeneous systems to minimize the energy consumption and maximize the reliability of system by executing the tasks concurrently as a combinatorial optimization problem. In Ref. [16], a double strategy was developed to minimize the Euclidean distance between the generated solutions to a set of user-specified constraints for four-objective problem, i.e., makespan, economic cost, energy consumption, and reliability optimization. In Ref. [17],

authors developed a scheduling algorithm using meta-heuristic approach to optimize the three conflicting objectives, i.e., minimization of energy consumption and makespan of tasks along with the maximization of the reliability.

2.2 Reactive Fault Tolerance

Reactive fault tolerance policies handle the effect of failures on application execution when the failure effectively occurs. Reactive fault tolerance policies reduce the effect of failures on application execution when the failure effectively occurs. Various reactive techniques are proposed in the literature for handling the failure of application tasks.

Checkpointing-Based Fault Tolerance Techniques. Checkpointing is a mechanism that records the system state periodically to establish recovery points. At the time of failure, computation restarts from the last saving state. For example, an application needs to access 100 files to complete the execution, and after reading of 99 files, there is some failure and all the computation gets lost. Just to avoid the restarting of application from the scratch, restart it from last saving state. Checkpointing is the most popular technique for recovery of unreliable systems, but some questions also arise: (i) How to optimize the number of checkpoints for each task? (ii) Since the task failure rate is not fixed so how to dynamically tune the optimal solution with the checkpoint/restart mechanism at run-time? (iii) How to know the size of checkpointing interval, etc.? In checkpointing mechanism, storage of snapshot of system state can be done locally or globally. On storing, checkpointing in shared disk achieves higher reliability and degrades the implicit process migration; in contrast, storing process status in local disk increases the cost of process migration. Different kinds of checkpointing techniques have been introduced by researchers like:

Full Checkpointing. It is a mechanism in which complete state of the process is saved to some media after a fixed interval of time. It should be considered that checkpoints should be applied after optimal interval so that it does not cost much overhead that result in maximization of application execution time. In Ref. [18], author developed an algorithm for optimal checkpointing by efficiently selecting checkpointing storage and recovery server.

Incremental Checkpointing. This mechanism helps in reducing the checkpoint overhead by saving those pages in which there have been any changes instead of saving the whole process. In the scheme [19], firstly, save the state of system as full checkpoint, and after that, some mechanism is applied to update the pages which have been modified since the last checkpoint. In Ref. [20], they develop an approach of smart checkpointing based on read-only and read–write part in VM image. During the time of first checkpoint, it saves the state of read-only part only once while rest of checkpoints will save modifications for read–write part. Checkpoints are logged into Hadoop

Distributed File System and replication is made to each node of the service provider. This mechanism is efficient when several tasks are recovering concurrently rather than one task recovery because checkpoints can concurrently recover from different nodes.

Uncoordinated Checkpointing and Coordinated Checkpointing. Each process takes its checkpoint independently; there is no synchronization among the processes to form a consistent global checkpoint. Hence, there is a main drawback called the domino effect. Coordinated checkpointing [21] is an approach for fault tolerance in distributed applications because all the processes come to an agreement to synchronize their checkpoints to make system consistent. This consistent set can be used to limit the rollback.

User-Level Checkpointing. The application programmer identifies program points at which state, status of task must be captured from within the application. In user-level checkpointing, explicit linking is required with the user-level library which is responsible for recovery from failure. In Ref. [22], the algorithm user specifies the different reliability requirements of each task. Each task synchronously checkpointing all its VMs using coordinated checkpointing scheme. One advantage of the algorithms is that it provides equal reliability to multiple users at same time using *peer-to-peer checkpointing*.

Replication-Based Fault Tolerance Techniques. As compared to failure of multiple resources simultaneously, a single resource is much more vulnerable to failure. Replication works as a guard against single point of failure and widely used as a fault-tolerant technique. The primary concern here is to decide the number of replicas. In replication, as the number of replicas increases, the fault coverage increases due to which management of backup systems is very costly. Various types of replication approaches have been studied in cloud computing environment, i.e., job replication, component replication, and data replication.

Active Replication. Each job is replicated on several processors and all the redundant processors are invoked simultaneously and the job will succeed if there is at least one processor which executed the job completely. This technique handles the f arbitrary faults by assigning the tasks to $3f + 1$ replica. In Ref. [23], an algorithm is developed for a Byzantine fault tolerance framework for a reliable system. In BFTCloud, a BFT group is chosen which consists of one primary and $3k$ replications of a task and response to current request will be committed or not committed are judge by cloud module. In case of 'k' faulty nodes in BFT group, execution of tasks will be done again on newly selected primary node and replicas. After identification of k faults, faulty resources will be substituted with other resources. BFTCloud achieves higher throughput where probability of failure of BFT group members is very low. In Ref. [24], MaxRe algorithm is developed for handling the fault during execution. The number of replicas of application tasks depends upon execution time.

Passive Replication. Once there is a crash in primary processor, the task will be scheduled on backup processor for completing their execution. In Ref. [25],

algorithm belongs to the category of rescheduling strategy. Compared to MaxRe [24] algorithm, the number of replicas depends upon number of failures but in former depends upon execution time. Problem with the algorithm is that it is not suitable for multiple failures and costly also, because once the failure is detected multiple replicas of tasks are executed. In Ref. [26], algorithm is developed using double strategy. The backup copies are overlapped with the backup copy of their precedence task on the same processors to further reduce schedule length and eventually maximize the performance.

Checkpointing along with replication. In Ref. [27], algorithm dynamically selects the most suitable reactive fault tolerance technique for task execution. The proposed algorithm adaptively determines the length of checkpoint interval and number of replicas of tasks based on previous history. In Ref. [28], an algorithm Replication-based scheduling for Maximizing System Reliability (RMSR) is suggested while considering the task communication. If the number of tasks increases, reliability decreases, but the algorithm RMSR increases the reliability by dynamically replicating the tasks according to threshold value (λ) determined by the user.

Primary-backup replication. In Ref. [29], an algorithm FASTER is developed for real-time applications in virtualized cloud. The author has extended the primary-backup (PB)-based scheduling by incorporating the cloud features like elasticity and virtualization. In Ref. [30], algorithm considers response time and replication cost for dependent tasks and independent tasks in the scheduling process with primary-backup approach. For dependent tasks, backup copies are scheduled with consideration of precedence tasks, and for independent tasks, backup copies can be scheduled with any backup copy of another task on same processor with minimum replication time and cost. In Ref. [6], the given algorithm first selects the number of host servers for placing the replicas of application and then selects the VM by considering the proximity of primary VM and backup VM, and if there is requirement of recovery, then selects the best virtual machine for completion of task as backup VM in the same subnet of hosts. The major aim is to moderate the consumption of network resources by using fat-tree network structure when there is a failure in primary virtual machine.

3 Conclusion

Among all the major challenges in cloud computing, fault tolerance is significant issue. Fault tolerance deals with errors and faults occurred in the system by ignoring, tolerating, and fixing. This paper identified various types of fault-tolerance-aware scheduling techniques. As we know, scheduling problem is NP-hard in nature, so different approaches like heuristics, meta-heuristics, or approximations are the possible solutions that are discussed in the survey. Nowadays, there are several mechanisms for fault tolerance but still there are number of challenges which need to be considered.

References

1. Sadiku, M.N., Musa, S.M., Momoh, O.D.: Cloud computing: opportunities and challenges. IEEE Potent. **33**(1), 34–36 (2014)
2. Patel, P., Ranabahu, A.H., Sheth, A.P.: Service Level Agreement in Cloud Computing (2009)
3. Garraghan, P., Townend, P., Xu, J.: An empirical failure-analysis of a large-scale cloud computing environment. In: 2014 IEEE 15th International Symposium on High-Assurance Systems Engineering (HASE), pp. 113–120. IEEE (2014)
4. Attiya, G., Hamam, Y.: Task allocation for maximizing reliability of distributed systems: a simulated annealing approach. J. Parallel Distrib. Comput. **66**(10), 1259–1266 (2006)
5. Rehani, N., Garg, R.: Meta-heuristic based reliable and green workflow scheduling in cloud computing. Int. J. Syst. Assur. Eng. Manag. 1–10
6. Zhou, A., Wang, S., Cheng, B., Zheng, Z., Yang, F., Chang, R., Buyya, R.: Cloud service reliability enhancement via virtual machine placement optimization. IEEE Trans. Serv. Comput. (2016)
7. Heddaya, A., Helal, A.: Reliability, Availability, Dependability and Performability: A User-Centered View. Boston University Computer Science Department (1997)
8. Qin, X., Jiang, H.: A dynamic and reliability-driven scheduling algorithm for parallel real-time jobs executing on heterogeneous clusters. J. Parallel Distrib. Comput. **65**(8), 885–900 (2005)
9. Charity, T.J., Hua, G.C.: Resource reliability using fault tolerance in cloud computing. In: 2016 2nd International Conference on Next Generation Computing Technologies (NGCT), pp. 65–71. IEEE (2016)
10. Zhou, A., Wang, S., Cheng, B., Zheng, Z., Yang, F., Chang, R., Buyya, R.: Cloud service reliability enhancement via virtual machine placement optimization. IEEE Trans. Serv. Comput.
11. Rehani, N., Garg, R.: Reliability-aware workflow scheduling using monte carlo failure estimation in cloud. In: Proceedings of International Conference on Communication and Networks, pp. 139–153. Springer, Singapore (2017)
12. Cao, F., Zhu, M.M.: Distributed workflow mapping algorithm for maximized reliability under end-to-end delay constraint. J. Supercomput. **66**(3), 1462–1488 (2013)
13. Dongarra, J.J., Jeannot, E., Saule, E., Shi, Z.: Bi-objective scheduling algorithms for optimizing makespan and reliability on heterogeneous systems. In: Proceedings of the nineteenth annual ACM symposium on Parallel algorithms and architectures, pp. 280–288. ACM (2007)
14. Wang, X., Yeo, C.S., Buyya, R., Su, J.: Optimizing the makespan and reliability for workflow applications with reputation and a look-ahead genetic algorithm. Fut. Generat. Comput. Syst. **27**(8), 1124–1134 (2011)
15. Zhang, L., Li, K., Li, C., Li, K.: Bi-objective workflow scheduling of the energy consumption and reliability in heterogeneous computing systems. Inf. Sci. **379**, 241–256 (2017)
16. Fard, H.M., Prodan, R., Barrionuevo, J.J.D., Fahringer, T.: A multi-objective approach for workflow scheduling in heterogeneous environments. In: Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012), pp. 300–309. IEEE Computer Society (2012)
17. Zhang, L., Li, K., Xu, Y., Mei, J., Zhang, F., & Li, K.: Maximizing reliability with energy conservation for parallel task scheduling in a heterogeneous cluster. Inf. Sci. **319**, 113–131 (2015)
18. Zhou, A., Sun, Q., Li, J.: Enhancing reliability via checkpointing in cloud computing systems. China Commun. **14**(7), 1–10 (2017)
19. Paun, M., Naksinehaboon, N., Nassar, R., Leangsuksun, C., Scott, S.L., Taerat, N.: Incremental checkpoint schemes for Weibull failure distribution. Int. J. Foundat. Comput. Sci. **21**(03), 329–344 (2010)
20. Goiri, Í., Julia, F., Guitart, J., Torres, J.: Checkpoint-based fault-tolerant infrastructure for virtualized service providers. In: 2010 IEEE Network Operations and Management Symposium (NOMS), pp. 455–462. IEEE (2010)

21. Cao, G., Singhal, M.: On coordinated checkpointing in distributed systems. IEEE Trans. Parallel Distrib. Syst. **9**(12), 1213–1225 (1998)
22. Zhao, J., Xiang, Y., Lan, T., Huang, H.H., Subramanian, S.: Elastic reliability optimization through peer-to-peer checkpointing in cloud computing. IEEE Trans. Parallel Distrib. Syst. **28** (2), 491–502 (2017)
23. Zhang, Y., Zheng, Z., Lyu, M.R.: BFTCloud: a byzantine fault tolerance framework for voluntary-resource cloud computing. In 2011 IEEE International Conference on Cloud Computing (CLOUD), pp. 444–451. IEEE
24. Zhao, L., Ren, Y., Xiang, Y., Sakurai, K.: Fault-tolerant scheduling with dynamic number of replicas in heterogeneous systems. In: 2010 12th IEEE International Conference on High Performance Computing and Communications (HPCC), pp. 434–441. IEEE (2010)
25. Mei, J., Li, K., Zhou, X., Li, K.: Fault-tolerant dynamic rescheduling for heterogeneous computing systems. J. Grid Comput. **13**(4), 507–525 (2015)
26. Chen, C.Y.: Task scheduling for maximizing performance and reliability considering fault recovery in heterogeneous distributed systems. IEEE Trans. Parallel Distrib. Syst. **27**(2), 521–532 (2016)
27. Amoon, M.: Adaptive framework for reliable cloud computing environment. IEEE Access **4**, 9469–9478 (2016)
28. Wang, S., Li, K., Mei, J., Xiao, G., Li, K.: A Reliability-aware task scheduling algorithm based on replication on heterogeneous computing systems. J. Grid Comput. **15**(1), 23–39 (2017)
29. Zhu, X., Wang, J., Guo, H., Zhu, D., Yang, L.T., Liu, L.: Fault-tolerant scheduling for real-time scientific workflows with elastic resource provisioning in virtualized clouds. IEEE Trans. Parallel Distrib. Syst. **27**(12), 3501–3517 (2016)
30. Zheng, Q., Veeravalli, B., Tham, C.K.: On the design of fault-tolerant scheduling strategies using primary-backup approach for computational grids with low replication costs. IEEE Trans. Comput. **58**(3), 380–393 (2009)