ISSN 2278-3091
Volume 10, No.3, May - June 2021
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse071032021.pdf
https://doi.org/10.30534/ijatcse/2021/071032021

A Review on Fault Tolerance in Distributed Database

Ahmad Shukri Mohd Noor[1], Auni Fauzi [2], Ainul Azila Che Fauzi [3]
Faculty of Ocean Engineering Technology and Informatics, Universiti Malaysia Terengganu, Malaysia, ashukri@umt.edu.my
[2]Faculty of Ocean Engineering Technology and Informatics, Universiti Malaysia Terengganu, Malaysia, ainul.auni@gmail.com
[3]Faculty of Computer Science and Mathematics, Universiti Teknologi MARA (UiTM) Cawangan Kelantan, Malaysia, ainulazila@uitm.edu.my

ABSTRACT

In this paper, we study about the different types of fault tolerance techniques which are used in various distributed database systems. The main focus of this research is about how the data are stored in the servers, fault detection techniques and the recovery techniques used. A fault can occur for many reasons. For example, system failure, resource failure, network between the server's failure and any other reasons. These faults must be emphasis in order to make sure the system can work smoothly without any problem. A proper failure detector and a reliable fault tolerance technique can avoid loss and at once save the system from fail.

Key words: Data recovery, Distributed database, Fault detection, Fault tolerance.

1. INTRODUCTION

In these days and age, data are very precious commodity Hence, it is important to make sure the data is well kept and secure. In order to keep the data safe, the usage of distributed database is more efficient compare to centralized database. Centralized database is a database that is located, stored and maintained in a single location. Thus, when the server is down or disaster happens, the whole data will be lost. Meanwhile, decentralized database or also known as distributed database is installed on systems that are geographically located at different location. Hence, in the event of disaster, there will always be another back up servers that have the same data. Therefore, distributed database system is more efficient in the term of preserving the data availability and reliability.

However, distributed database system cannot avoid from facing failures. This will result a faulty system. A faulty system can lead to a serious damage. A real time distributed database system is highly dependable on the reliability of the systems. If the failure is not detected and recovered properly at time, it can result to a system failure. For the critical systems that need frequently update such as such as flight control systems, banking system, nuclear systems and etc, they must be well functioning with high availability even under any failures. Hence, fault tolerance is very important technique to maintain system reliability and dependability.

Fault ought to be identified by applying a reliable fault detector followed by a recovery technique. Unreliable fault detector can commit errors by mistakenly trusting crashed process or suspecting the correct process. The rest of the paper is organized as Section 1 is introduction, Section 2 is fault tolerance techniques and Section 3 is literature review.

2. FAULT TOLERANCE TECHNIQUES

There are many fault tolerance techniques that have been in order to make sure all the systems can still function under a failure occurrence. Based on fault tolerance policies and techniques, it can be classified into two types; proactive and reactive as shown in Figure 1 [1].

2.2 Proactive Fault Tolerance

The concept of proactive fault tolerance policy is to predict faults and replace suspicious components in advance by avoiding recovery from faults, errors, and failures. This policy will detect the faults before the actual problems occur. This policy prevents the system from calculating the node failures by analyzing some of the applications (tasks, processes, or virtual machines) away from the failing nodes. In other word, this policy prevents the failures to affect the running parallel applications. Several technologies based on these policies such as software rejuvenation self-healing capabilities and preemptive migration are proposed.

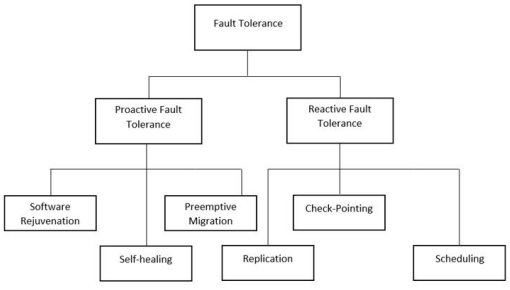

Figure 1: Fault tolerance types of group

- Software rejuvenation: A system with a method designed for periodic reboots. This method will reboot the system in a clean state and will be useful for a new startup [2].
- Self-healing: The basic idea is to make an auto-control of the failure of an application instance running on multiple virtual machines. This idea is done to improve the performance. If difference instances of the application are running on different virtual machines, then automatically handle the failure of the application instance.
- Preemptive migration: Preemptive migration is counted through the feedback loop control mechanism. Applications are constantly identified and analyzed.

2.3 Reactive Fault Tolerance

The reactive fault tolerance policy is also known as on-demand fault tolerance. It reduces the impact of failures on application execution when the failure occurs effectively. There are various methods based on this policy such as replication, check-pointing and scheduling.

- Replication: Storing several copies of the same data in different servers is the basic idea of data replication. This obviously increases the performance by decreasing remote access latency and failure [3]. Providing reliable services along with high data availability and the performance are the important requirements that need to be essentially met. The concept of replication is used to ensure these requirements. The main idea of replication is to manage large volumes of data in a distributed manner, speeds up data access, reduces access latency and increases data availability [4], [5], [6]. There are three fundamental questions that must be answered in managing replica placement strategy [7]. The three questions are: When must the copies be produced? What data must be copied? Where the copies must be allocated?
- Check-pointing: It is the process to saving from complete execution a task [8]. Check-pointing approach balances the load of processors in a distributed system; processes are moved from heavily loaded processors to lightly loaded ones. Check-Pointing process periodically provides the information necessary to move it from one processor to another [9]. Check-Pointing can be initiated from within grid systems or within applications.
- Scheduling: It is used to overcome the drawback of check-pointing in distributed environment [8]. It is categorized as time-sharing scheduling, space sharing scheduling, and hybrid (combination of both). Scheduling is used for load balancing as well as fault tolerance in distributed system on the basis of space or time sharing [8], [10]. There are three approaches of scheduling such as space, time and hybrid. Space scheduling is used to tolerate permanent or hardware type of fault from a system. The Primary-Backup approach is applied in space redundancy. Time redundancy is used when there is intermittent type of fault in the system and hybrid redundancy is used when both are required.

3. LITERATURE REVIEW

The existing fault tolerance techniques consider various parameter which are the platform used, simulation or real time, technique used to detect, recovery technique used, advantages and drawbacks are being compared in Table 1.

Table 1: The Comparison Among Existing Techniques

Source	Recovery technique used	Advantage	Drawback
Minimizing Overheads Checkpoints [11]	Check pointing	Minimize overheads of checkpoints	not reliable for critical system
Fault tolerance in distributed database [12]	Replication with parity calculation	System can run properly even when two sites are down	Need to have minimum of six sites
Dmap [9]	Check pointing and replication	Support consistent long running read operations	The failure site cannot be used during the failure occurrence
Active-active Virtual Machine [13]	Check pointing and replication	The performance for online service with i/o intensive is faster compare to COLO	What if the secondary database is crashed?
An efficient fault tolerance framework [14]	Replication with parity calculation	Faster recovery	Minimum sites are four
Hadoop Mapreduce [15]	Re-executed and re-scheduling	Provides reasonable resistance to failures.	High execution time
Fault tolerance evaluation of a new SQL database [16]	Replication	Can tolerate two failure nodes	The failure node cannot be used during the failure occurrence

A new checkpoint technique is proposed to minimize the checkpoint overhead in distributed system [11]. This technique will reduce the number of checkpoints by

increasing the checkpoint interval. The study of the rate of system failure will be conducted first in order to decide the increasing of checkpoint interval. If the system failure rate is high and almost all systems crashed during the interval time, then the checkpoint schedule will remain the same as traditional checkpoint. Otherwise, if the system failure rate is low and no system shows failure during the specific time, then it will increase the checkpoint interval. The result shows that the time consumed when the checkpoint is increased is lesser than the normal checkpoint.

An enhancement technique from Redundant Array of Inexpensive Disks 5 (RAID 5) is proposed [12]. During the first half of this technique, it will be similar to RAID 5 technique. It will divide the data into blocks (databases) and also will calculate the parity blocks. During the experiment all databases are assumed located in different locations. Instead of 4 blocks that has been used in RAID 5, this new technique required of 6 minimum blocks. This will improve the fault tolerance because it can handle 2 databases failure that occurred at the same time. The surviving database will not only have to handle the recovery, but it also needs to deal with the requested task from client for itself as well as the failure database.

A checkpointing and replication technique is proposed in order to handle system failure [9]. Each replica will periodically checkpoint onto its stable state. Upon resuming after failure, the recovering replica will retrieve the last checkpoint state and will install the latest checkpoint. It will also request the most recent data on all replicas to ensure data consistency. Furthermore, this code can be used to scale any existing Java applications.

An active-active technique called GANNET is proposed [13]. This paper critiqued the usage of active-passive technique paper. Active-passive technique will cause high performance overhead due to release outputs of services. It is because of large amount of states to be transferred. Even though active-active systems have greatly reduced the load in transferring the data, it will get worse in case of performance for online services when intensive I/O workload exist. Thus, this paper presented a new replication fault tolerance technique for active-active virtual machine, GANNET. GANNET will only have 36 and 49 code respectively and can handle single point of VM failure. The result shows huge contribution in storage checkpoint duration compared to existing technique only when the present of intensive I/O workload exist.

An efficient fault tolerance technique in distributed in-memory caching system is proposed [14]. This technique adopts Row-Diagonal Parity (RDP) codes in improving efficiency Memcached system. RDP array is defined by a controlling parameter p which is should be a prime number and must be greater than 2. The minimum nodes for this technique would be four based on the equation for total nodes p+1, where p=3. Moreover, the parity block will follow the equation p-2. For the data recovery, this technique uses RDOR-based Data Recovery which acquires less transmissions during reconstruction. For example, the standard decoding process to recover all the data block where p=5 it will need 16 blocks in total for RDP. By using RDOR it will need to have 12 blocks only. Thus, recovery time will be faster compared to RDP.

A fault tolerance technique of Hadoop Mapreduce under failure occurrence is discussed [15]. Fault tolerance in Hadoop Mapreduce uses timeout as a failure detection. When there is a task failure occur, the worker node or also called as slave node will sends heartbeat to notify the master node about the failure. Next, the master node will try to re-execute from scratch on another healthy node. This process may be taking a long recovery time and will lead to unpredictable tasks execution time and resources wastage. In order to detect the failure, the master node will check if any slave node has not sent the heartbeat message within 10 minutes. Then, if the master node fails, MapReduce will automatically restart the master node and rescheduling all the failure tasks.

This proposed fault tolerance technique is quite same with the technique that discussed above. It uses a heartbeat failure detection technique to detect any failure from Region Server (RS) or Data Node [16]. Both nodes are containing their own data. From the experiment, it shows even when a node fails and process the maximum load for the configuration the system is able to recover and continue processing with reasonable latency.

4. CONCLUSION

Fault tolerance is very important in order to handle the occurrence of fault especially in distributed database system or critical data system. Several existing fault detection and fault recovery techniques and models has been reviewed and been compared in this paper. In the present scenario, there are number of challenges which need some concern hence fault tolerance will need to evolve to be able to solve the design fault problem.

ACKNOWLEDGEMENT

This research is funded by Fundamental Research Grant Scheme (FRGS) with the Ref: FRGS/1/2018/ICT04/UMT/02/2. FRGS is a research grant from the Ministry of Higher Education (MOHE) Malaysia.

REFERENCES

1. P. K. Patra, H. Singh and G. Singh. **Fault Tolerance Techniques and Comparative Implementation in Cloud Computing,** *International Journal of Computer Applications,* vol. 64, no. 14, pp. 37-41, 2013.
2. A. Ledmi, H, Bendjenna and S. M. Hemam. **Fault Tolernace in Distributed Systems: A Survey,** in *3rd International Conference on Pattern Analysis and Intelligent Systems (PAIS),* Tebessa, 2018.
3. D. Boru, D. Kliazovich, F. Granelli, P, Bouvry and A. Y. Zomaya. **Energy-Efficient Data Replication in Cloud Computing Datacenters,** in *IEEE Globecom Workshops,* Atlanta, 2018, pp. 446-451.
4. N. Ahmad, A. A. C. Fauzi, S. H. S. A. Ubaidillah and B. Alkazemi. **BVAGQ_AR for fragmented database replication management**, *IEEE Access*, vol. 9, pp. 56168-56177, 2021.
5. B. A. Milani and N. J. Navimipour. **A comprehensive review of the data replication techniques in the cloud environments: Major trends and future directions**, *J. Netw. Comput. Appl,* vol. 64, pp. 229-238, 2016.
6. J. Wang, H. Wu and R. Wang. **A new reliability model in replication-based big data storage systems**, *J. Parallel Distrib. Comput,* vol. 108, pp. 14-27, 2017.
7. N. Ahmad, A. A. C. Fauzi, W. M. W. Mohd, M. Amer and T. Herawan. **Managing fragmented database replication for Mygrants using binary vote assignment on cloud quorum,** *Applied Mechanics and Materials,* vol. 490, pp. 1342-1346, 2014.
8. A. Dagur, R. S. Yadav and A. J. Ranvijay. **Fault Tolerance in Real Time Distributed System,** *International Journal on Computer Science and Engineering (IJCSE), vol. 3, no. 2, pp. 933-938, 2018.*
9. S. Benz and F. Pedone. **DMap: A fault-tolerant and scalable distributed data structure,** in *IEEE 37th International Symposium on Reliable Distributed Systems,* Salvador, 2018, pp. 153-160.
10. S. Krishnan and D. Gannon. **Checkpoint and restart for distributed components in XCAT3,** in *Proceedings of the Fifth IEEE/ACM InternationalWorkshop on Grid Comp,* Pittsburgh, 2004, pp. 281-288.
11. S. M. A. Akber, H. Chen, Y. Wang and H. Jin. **Minimizing Overheads of Checkpoints in Distributed Stream Processing Systems,** in *IEEE 7th International Conference on Cloud Networking (CloudNet),* Tokyo, 2018, pp. 1-4.
12. S. Pareek, N. Sharma and G. M. A. **Fault Tolerance in Distributed Database Management Systems – Improving reliability with RAID,** in *Innovations in Power and Advanced Computing Technologies (i-PACT),* Vellore, 2019, pp. 1-4.
13. C. Wang, X. Chen, Z. Wang, Y. Zhu and H. Cui. **A Fast, General Storage Replication Protocol for Active-Active Virtual Machine Fault Tolerance,** *in IEEE 23rd International Conference on Parallel and Distributed Systems (ICPADS),* Shenzhen, 2017, pp. 151-160.
14. S. Zhao, L. Shen, Y. Li, R. J. Stones, G. Wang and X. Liu. **An Efficient Fault Tolerance Framework for Distributed In-memory Caching Systems,** in *IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS),* Singapore, 2018, pp. 553-560.
15. S. Yassir, Z. Mostapha and C. Tadonki. **Analyzing fault tolerance mechanism of Hadoop Mapreduce under different type of failures,** in *4th International Conference on Cloud Computing Technologies and Applications (Cloudtech),* Brussels, 2018.
16. A. Azqueta- Alzúaz, M. P. Martinez, V. Vianello and R. J. Péris. **Fault-tolerance Evaluation of a New SQL Database,** *14th European Dependable Computing Conference (EDCC),* in Iasi, 2018, pp. 81-86.