Comparative Study of Fault tolerant Strategy in Cloud

Savleen Kalra
Research Scholar
BCET, Gurdaspur

Sh. Rajeev Bedi
Assistant Professor
Dept. of CSE
BCET, Gurdaspur

S.K. Gupta
Associate Professor
Dept. of CSE
BCET, Gurdaspur

ABSTRACT

Most of fault tolerant strategies researched over rely on hardware or time redundancy. The techniques are used to mask faults. Time redundancy or replication indicates re-execution of failed component after failure is detected. The result can further be optimizing by the use of checkpoints. This strategy however consumes more time or in other words significant delay is introduced during the operation of fault tolerance. In many mission critical applications hardware redundancy is employed. This approach uses process replication to achieve fault tolerance. Both the approaches have drawbacks. System engineer hence is in dilemma in terms of approach to follow for fault tolerance. Shadow computing is the solution to dilemma which provides dynamic execution. Parameterized trade off between hardware and time redundancy is provided to ensure fault tolerance. This paper provides survey of time redundant, hardware redundant and shadow replication mechanism along with comparison of each.

KEYWORDS

Fault Tolerance, hardware, time redundancy, shadow replication

I. INTRODUCTION

Power consumption is the issue facing data centers and present in past and current techniques in cloud computing and high performance computing used to ensure fault tolerance[1][2]. Energy and power conservation is the need of the hour hence energy and power aware systems must be incorporated within future computational system. Consumption in energy vastly depends upon increase in number of computational nodes. It has adverse effect on reliability of the system. Even if individual node failure rate is low, still over all system failure rate goes beyond threshold value as all the node failure rate is considered in this situation. For example, consider the nodes 200,000, mean time between failure in this case is less than 1 hour but mean time between failure for 1 node is 5 years[3].

Future systems clearly should be energy aware fault tolerant in nature. Rest of the paper is organized as follows: section II provide the detailed description of strategies used in time redundant modelling for fault tolerance, section III provides description of hardware replication strategies used to monitor faults, section IV provides details of shadow replication mechanism. Section V give information regarding need for energy efficiency ,Section VI gives comparative analysis of fault tolerance mechanism where as section VII provides conclusion and future scope.

II. TIME REDUNDANT MECHANISMS FOR FAULT TOLERANCE

Technology advancement led to situation in which machines having least resources can execute more using data centers in cloud computing. Pay per use services are provided by cloud computing [4]and advanced computing mechanisms. As dependability increase so does a risk. This risk arises as more and more nodes starts to interact with advance computing model. Mean time between failure increases as more and more nodes interact with cloud and advance computing. Hence trade off exist between number of nodes and reliability. As process fail all the progress made by the node is lost. fault tolerant strategy is required to cope with this situation. Re-execution of process can lead to recovery and hence lost progress can be recovered. Mechanism following the listed approach is known as Time dependent mechanism[3][5]. Strategies under Time redundant approach are described in this section

- RECOMPUTING WITH SHIFTED OPERANDS

This is one of commonly used mechanism to ensure fault tolerance. Functions and operands are recomputed and compared against the previous results to ensure accuracy and faults present within the system. This mechanism is effective enough for short term faults that existed within the system. This approach however fails if faults is permanent or long lasting within the system[6]. For example, consider addition of two numbers X_1 and X_2. Numbers are added by shifting the digits of numbers to the left and then added again. This process continues until corrected answer is not obtained. Mechanism is reliable in case of short term or temporary faults.

- CHECKPOINTS

This is popular mechanism of establishing fault tolerance within advance computing schemes. In this approach save point is established at various points in time. As the node is using the resource reaches certain point in time where save point is established then progress is saved. This is a automated approach and has many advantages. One of the advantages is fault tolerant ability is introduced through recovery approach. The drawback of this approach is that it will consume more time to recover a system to normal stage[7][8].

III. HARDWARE DEPENDENT FAULT TOLERANCE

Most of faults in real time environment is due to hardware failure. This section tries to tackle these faults by considering hardware dependent fault tolerance mechanisms. Hardware which is sensitive to faults is made redundant in this approach. This section describes hardware redundancy to ensure fault tolerance.

- PASSIVE FAULT TOLERANCE

Passive fault tolerance do not necessarily remove fault from the system rather it hide the fault and hence fault become invisible. Mechanism is also termed as fault masking. Major problem with the passive fault tolerance is small problem in inputs can yield large deviation in output. Voting mechanism is used in order to select resources in this approach. The voting could either be hardware or software in nature. Hardware voting is fast but expensive. Software voting is flexible but slow[9][10].Mid value select approach is used in order to rectify voting problem.

- ACTIVE FAULT TOLERANCE

Active fault tolerance mechanisms are actually used to rectify the faults that occurred within the system. Hence faults are actually removed rather than hiding as in passive fault tolerance. Active fault tolerance mechanism involve the following techniques

a. DUPLICATION WITH COMPARISON

In this approach two identical modules performing the same task are placed in parallel. The results produced by two identical modules are compared with each other. The threshold value is established. The module producing result satisfying threshold is accepted. The problem with this approach is that it can only detect the faults but cannot tolerate. Recovery mechanism is hence absent in this case[11][10].

b. STAND BY SPARING

This approach utilizes spare modules along with workable module. One module in this case is operational while other modules are spare. Error detection mechanisms are implemented to detect when the fault is occurring within the system along with detection of module in which fault occurred. The faulty module is removed from the system and is replaced by the spare module. Special switch is used to monitor the errors. The module if error frees then selection is made on the basis of priority. The module having error is eliminated from the system[12][10].

Hot standby sparing is also available in which spare modules are active all the time and are ready to change place with running module in case of errors. This switching operation is fast and downtime is close to zero. This downtime is the problem with offline standby sparing techniques[10].

c. PAIR AND SPAIR TECHNIQUE

In this technique hybrid approach is followed by combining duplication with comparison and stand by sparing. Two modules are operated in parallel rather than one module. Their results are compared and error is detected if any. Both error detection and correction mechanism is available under this methodology[13], [14].

d. WATCHDOG TIMER

Watchdog timer is effective mechanism used in order to detect crash, infinite loop and failure. Watchdog timer is reset many times during its operation. As the fault is detected timer is reset. If timer is not reset then system is turned off[15].

The hardware redundancy techniques are efficient enough to handle faults present within the system. But these techniques are expensive. The faults that can be detected are limited in nature. In order to tackle the problem of hardware and time redundant techniques hybrid approach is utilized rather than individual approach.

IV. SHADOW REPLICATION

This is the mechanism which follows both hardware and time redundant techniques. the process level replication is performed in this case. Large scale distributed system requires self configuration of resources with minimum human interference. Fault tolerance mechanisms for mobile agents to cope with the mobile server crash can be accomplished through shadow replication. The approach required in this case is known as dynamic shadow replication[16].

The basic concept of shadow replication is to associate with each main process a shadow. The size of shadow depends upon criticality of application. Shadow replication mechanism is described through the following steps.

- A process is given a responsibility to execute a task by allocating it exclusive core processor.
- Many Shadows are associated with the main process.
- Every shadow has equivalent speed and process allocation space as the original process.
- In case of failure main process is shifted to the shadow and shadow becomes main processor on which process executes[17].

Shadow replication suffers from the drawback of failure in shadow cores. But this problem can be rectified by the use of checkpoint in this scheme.

V. NEED OF ENERGY EFFICIENCY WITHIN FAULT TOLERANCE

Technology advancement led to the situation that more and more users are participating in advanced computing. As node increases, energy consumption also increases within advanced and cloud computing. Energy consumption and reliability has trade off associated with them. Hence reliability decreases as energy consumption increases. Sensor nodes have limited energy along with less storage capacity. High energy consumption makes lifetime of sensors limited hence working of sensors is greatly affected[18].

Energy consumption directly linked with the cost. Most of the services provided by cloud and advance computing models is on the basis of pay per use. As energy consumption increases so does utilization of resources. This causes high costing to be encountered. Reliability is also at stake during high energy consumption. Energy consumption and reliability is inversely proportional to each other. This is the prime reason to decrease energy consumption so that reliability can enhance and which ultimately increase efficiency[19][20].

VI. COMPARATIVE STUDY OF VARIOUS TECHNIQUES FOR FAULT TOLERANCE

S.no.	Title	Year	Published By	Type of Fault handled
1	Fault Tolerant Approaches in Cloud Computing Infrastructures Alain[21]	2012	ICAS	Handled fault at different level application level, virtualization level and hardware level.
2	A Family of Fault-Tolerant Efficient Indirect Topologies[22]	2016	IEEE	Fault tolerance in network for high performance computing are done and it uses simple indirect topology to handle the faults
3	Fault Tolerance Management in Cloud Computing: A System-Level Perspective[23]	2016	IEEE	Relies on generic fault tolerance mechanisms that handle the fault at server end
4	Fault tolerance techniques and algorithms in cloud computing[24]	2014	IJCSCN	Handle the faults that enter in the system or software.
5	Optimising Fault Tolerance in Real-time Cloud Computing IaaS Environment[25]	2016	IEEE	Handle faults in real time computing on the cloud infrastructure.
6	Fault Tolerance in Cloud Using Reactive and Proactive Techniques 1Kalanirnika[26]	2015	IJCSEC	Manage faults in memory and perform recovery using checkpoints
7	Fixed-Priority Allocation and Scheduling for Energy-Efficient Fault Tolerance in Hard Real-Time Multiprocessor Systems[27]	2008	IEEE	Manage faults in hard real time systems using optimistic fault tolerance algorithms
8	On Fault Tolerance in Data Centre Network Virtualization Architectures [28]	2013	IEEE	Handle faults in Virtual Data Centres using the address handling techniques at server end.
9	Fault tolerance and QoS scheduling using CAN in mobile social cloud computing[29]	2013	Springer	Handle faults in mobile devices in computing environment and use CAN structure for fault management
10	A Performance	2007	IEEE	Manage the faults that

	Study of Deployment Factors in Wireless Mesh Networks[30]			occurred in mesh topology during the connectivity and also gives the mechanism

Table 1: Existing literature on fault tolerance

Legion of techniques are available to ensure fault tolerance within cloud and advance computing models. Energy efficiency however can further be enhanced using hybridization.

VII. CONCLUSION AND FUTURE SCOPE

Fault tolerance along with energy efficiency is the need of the hour. The proposed work provides detailed description of various approaches used to establish the same. Energy and reliability has trade off. This means that when energy consumption is high reliability is low and vice versa. Described approaches including time redundant and hardware redundant approaches are not efficient enough to tackle energy efficiency along with fault tolerance. In future hybrid approach could be area of analysis including shadow replication approach. Hybridization with check pointing can provide efficient energy efficiency fault tolerance within advance computing model.

VIII. REFERENCES

[1] S. Ashby, P. Beckman, J. Chen, P. Colella, B. Collins, D. Crawford, J. Dongarra, D. Kothe, R. Lusk, P. Messina, and others, “The opportunities and challenges of exascale computing,” *Summ. Rep. Adv. Sci. Comput. Advis. Comm. Subcomm.*, pp. 1–77, 2010.

[2] L. A. Barroso and U. Hölzle, “The Case for Energy-Proportional Computing,” *Computer (Long. Beach. Calif).*, vol. 40, no. 12, pp. 33–37, Dec. 2007.

[3] B. Mills, T. Znati, and R. Melhem, “Shadow Computing: An energy-aware fault tolerant computing model,” *2014 Int. Conf. Comput. Netw. Commun.*, pp. 73–77, 2014.

[4] Z. Zhu, G. Zhang, M. Li, and X. Liu, “Evolutionary Multi-Objective Workflow Scheduling in Cloud,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 5, pp. 1344–1357, May 2016.

[5] D. Burlyaev, P. Fradet, and A. Girault, “Time-redundancy transformations for adaptive fault-tolerant circuits,” in *2015 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, 2015, pp. 1–8.

[6] “10.1109@FTCS.1989.105616.pdf.” .

[7] J. Devale, “Checkpoint / Recovery Overview : Checkpointing - Recovery,” *Memory*, 1999.

[8] J. Hursey, J. M. Squyres, T. I. Mattox, and A. Lumsdaine, “The Design and Implementation of Checkpoint / Restart Process Fault Tolerance for Open MPI ∗.”

[9] P. A. Lee and T. Anderson, “Fault Tolerance,” Springer Vienna, 1990, pp. 51–77.

[10] E. Dubrova, “System-on-Chip - Hardware redundancy.”

[11] K. Echtle, “Fault Tolerance based on Time-Staggered Redundancy,” Springer Berlin Heidelberg, 1987, pp. 348–361.

[12] K. E. Grosspietsch, “Schemes of dynamic redundancy for fault tolerance in random access memories,” *IEEE Trans. Reliab.*, vol. 37, no. 3, pp. 331–339, 1988.

[13] W. Krings and F. S. Sequence, “Hardware redundancy –,” pp. 1–25, 2007.

[14] M. S. Adeghi, H. S. Oltani, and M. K. Hayyambashi, “The study of hardware redundancy techniques to provide a fault tolerant system,” vol. 36, 2015.

[15] J. R. Barnes, “Watchdog Timers,” in *Robust Electronic Design Reference Book*, Boston, MA: Springer US, 2004, pp. 860–867.

[16] J. Xu and S. Pears, “A Dynamic Shadow Approach to Fault-Tolerant Mobile Agents in an Autonomic Environment,” *Real-Time Syst.*, vol. 32, no. 3, pp. 235–252, Mar. 2006.

[17] X. Cui, B. Mills, T. Znati, and R. Melhem, "Shadow replication: An energy-aware, fault-tolerant computational model for green cloud computing," *Energies*, vol. 7, no. 8, pp. 5151–5176, 2014.

[18] M. Singh and V. K. Prasanna, "Energy-efficient and fault-tolerant resolution of topographic queries in networked sensor systems," *Proc. Int. Conf. Parallel Distrib. Syst. - ICPADS*, vol. 1, pp. 271–280, 2006.

[19] A. Banerjee, P. Agrawal, and N. C. S. N. Iyengar, "Energy Efficiency Model for Cloud Computing," *Int. J. Energy, Inf. Commun.*, vol. 4, no. 6, pp. 29–42, 2013.

[20] Y. Sharma, B. Javadi, W. Si, and D. Sun, "Reliability and energy efficiency in cloud computing systems: Survey and taxonomy," *J. Netw. Comput. Appl.*, vol. 74, pp. 66–85, 2016.

[21] A. Tchana, L. Broto, and D. Hagimont, "Fault Tolerant Approaches in Cloud Computing Infrastructures," no. c, pp. 42–48, 2012.

[22] D. F. Bermudez Garzon, C. G. Requena, M. E. Gomez, P. Lo, and J. Duato, "A Family of Fault-Tolerant Efficient Indirect Topologies," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 4, pp. 927–940, Apr. 2016.

[23] A. S. Perspective, R. Jhawar, G. S. Member, and V. Piuri, "Fault Tolerance Management in Cloud Computing :," pp. 1–10, 2012.

[24] Y. L. Devi, "FAULT TOLEREANE TECHNIQUES AND ALGORITHMS IN CLOUD Cloud system," vol. 4, no. 1, pp. 1–8.

[25] B. Mohammed, M. Kiran, I. Awan, and K. M. Maiyama, "Optimising Fault Tolerance in Real-time Cloud Computing IaaS Environment," pp. 363–370, 2016.

[26] G. R. Kalanirnika and V. M. Sivagami, "Fault Tolerance in Cloud Using Reactive and Proactive Techniques," vol. 3, no. 3, pp. 1159–1164, 2015.

[27] R. M. Systems, T. Wei, P. Mishra, K. Wu, and H. Liang, "Fixed-Priority Allocation and Scheduling for Energy-Efficient Fault Tolerance in Hard," vol. 19, no. 11, pp. 1511–1526, 2008.

[28] S. C. Joshi and K. M. Sivalingam, "On Fault Tolerance in Data Center Network Virtualization Architectures," pp. 1–6, 2013.

[29] S. Choi, K. Chung, and H. Yu, "Fault tolerance and QoS scheduling using CAN in mobile social cloud computing," 2013.

[30] J. Robinson and E. W. Knightly, "A Performance Study of Deployment Factors in Wireless Mesh Networks," *IEEE INFOCOM 2007 - 26th IEEE Int. Conf. Comput. Commun.*, pp. 2054–2062, 2007.