International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-9 Issue-1, October, 2019

Proactive Fault Tolerance using Heartbeat Strategy for Fault Detection

Shelly Prakash, Vaibhav Vyas, Anup Bhola

Abstract: ***Failure is something which causes services on the cloud to go down for some time period. Most of the times instead of recovery and repair, we opt for virtual machine migration where failover of the failed service is done on some other running virtual server so that the service is revived. Virtual migrations and recovery mechanisms consume a lot of energy and many approaches are implemented to make them energy efficient. Failure Detection is a topic of equal importance and comes under fault tolerance. Failure detection if done properly can be more effective and energy/cost saving than fault recovery. Heartbeat strategy is one such failure detection approach where live processes send an "I am alive" message to the host device at some pre-defined fixed intervals which ensures that the process is running fine. In this paper, we propose to mark the nodes whose processes have failed to send the heartbeat message and prepare a count (confidence factor, α) for the same. In primary testing, if this confidence factor reaches a specific threshold then that particular node is sent for confidence testing (second level failure detection testing using a different time sequence of heartbeat message arrival) and later marked for failure recovery (if found faulty). Fault recovery techniques are then applied to it so that it can be corrected and reused and the current jobs can be migrated to the better node during the recovery period. If the confidence factor, α is below the threshold value then no action is taken and only network parameters and connections can be rechecked. This method will re-ensure the trust on heartbeat strategy for fault detection and save the device from failure.***

Keywords: Proactive, reactive fault tolerance, confidence factor, primary testing, confidence testing, virtual machine.

I. INTRODUCTION

Cloud computing refers to the delivery of computing resources as a service. Green Computing defines being environmentally responsible and eco-friendly while using computers and their resources. Green computing is also known as Green information technology (Green IT). Green IT should not only save energy but also reduce operating expenditures (OPEX) and capital expenditures (CAPEX). Data centres with huge data storage support have been promoting technologies such as data deduplication, storage virtualization and storage convergence, which helps in both decreases in carbon footprint and operational costs. This can be achieved by converging data centres, increasing power usage efficiency (PUE), recycling, proper disposal of e-waste, lowering gas emissions, and minimizing water usage in the cooling process of such devices. As the cloud services are so dynamic and scalable in nature, there can be unpredictable faults leading to failure of services. Therefore there is a need to closely monitor any occurrence of a fault and prepare the devices and software to handle the fault as and when it occurs. There are a lot of mechanisms and models which are built to enhance fault tolerance in cloud systems. Fault-Tolerant strategies are mainly focussed before a user agrees for Service Level Agreement (SLA) for obtaining a virtual server. This paper focuses on enhancing Heartbeat strategy used for fault detection by adding a confidence factor in the whole mechanism. Along with the heartbeat message, this confidence factor is also monitored and it decides which node is healthy which virtual machine should be removed and marked for recovery. The time duration of periodic checking is proposed to be decided by the machine learning algorithms and not to be based on previous patterns.

Revised Manuscript Received on October 30, 2019.

* Correspondence Author

Shelly Prakash, Ph.D. Scholar**,** Department of Computer Science, Banasthali Vidyapith, Tonk, Rajasthan. Email: shellyprakash.mpsi@gmail.com

Dr. Vaibhav Vyas ,Associate Professor, Department of Computer Science, Banasthali Vidyapith, Tonk, Rajasthan. Email: vvaibhav@banasthali.in

Dr. Anup Bhola, Assistant Professor, Department of Computer Science, Banasthali Vidyapith, Tonk, Rajasthan. Email: banup@banasthali.in

II. FAULT TOLERANCE

Software is judged on both its functional and non-functional properties. Functional properties include its working and usage, while, non-functional properties include its reliability, performance, fault tolerance and availability (Mylara Reddy Chinnaiah, N. N. 2018). Fault Tolerance means that when a process/service fails then server must be able to recover from it and take sufficient steps to ensure Quality of Service and service availability for the user. It is a technique to make the devices capable of handling faults on their own to some extent and continue to provide services to the users without any hindrance. To increase dependability and trust in cloud services, it is important to secure the device from failure. To incorporate this into a device, we need to have a better understanding of the error, bugs, faults, and failure. An error or bug in the system causes a fault, and which in turn leads to failure. Even a single error can cause the failure of the whole system. The error can be caused in either hardware or software or both. The various types of faults are:

- *Network-based:* Data loss, connectivity issue, network traffic, and data corruption, etc.

- *Software-based:* Operating System Corruption, application/software performance issue, unexpected crash/errors or software update failure, disk space failure, etc.
- *Hardware-based:* Device failure, hardware aging, CPU, disk failure or excess heat generation, etc.

III. FAULT TOLERANCE APPROACHES

We follow 2 distinct approaches to faults in cloud computing. These are reactive fault tolerance and proactive fault tolerance approaches.

A. Reactive Fault Tolerance

This is the kind of approach that we follow when the failure has actually occurred and the system still should continue its processes without any hindrance. Some of the reactive approaches are:

- *Check pointing*
  A method in which checkpoints are created in the application so that it can be monitored in those suspected or vulnerable situations.
- *Retry*
  If an application fails to start or has some error while running then it can be made to retry to restart its process after recovering from the fault that had occurred.
- *Replication*
  The applications running on the server are replicated on other servers so as to increase resource utilization and to prevent any denial of service in case of server failure.
- *S Gaurd*
  It is a rollback and recovery strategy. In this process in case of failure, the processes are rolled back to a safe checkpoint and then recovered back to running state.
- *Task resubmission*
  If a subtask caused the whole process to fail, then we try to resubmit that task and assign a new set of resources to it to eliminate the chances of failure this time.
- *User-defined exception handling*
  In such a strategy, a server monitoring team decides what action has to be taken if a failure has occurred. Also, the exception handling schedules defined at the time of development are brought to action when a failure occurs.

HAProxy is reactive fault tolerance architecture. According to this system, we have two servers. If server1 fails, then the other server, server2 which has redundant data of server1 emerges as a backup system for entire server1 operations, and vice versa. HAProxy uses job migration and replication techniques.

B. Proactive Fault Tolerance

This approach works to predict or presume the failure that may likely occur according to the periodic system checks and thus prevents a failure.

- *Self-healing:*

The ability of the server to tolerate and rectify failures to some extent is self-healing.

- *Software rejuvenation :*

After some periodic instances of time, the server has to be restarted so that it starts fresh, eliminating all blocked, deadlocked states of processes that may cause failure later.

- *Pre-emptive migration:*

When a failure is detected, the crucial running applications from a virtual server are migrated to another virtual server to avoid failover.

IV. FAILURE METRICS

Fault Tolerance is based on analyzing the occurrence and timing of faults so that we are ready in our approach to handling it. For this, we have the following essential terms related to time instances in failure detection and recovery.

- *Mean Time To Recovery (MTTR):* It is the amount of time consumed in repairing the system, testing it and resuming processes after the failure has occurred. It is measured exactly after the first notification of failure is received. It is calculated as:

MTTR = total maintenance time/ total no. of repairs

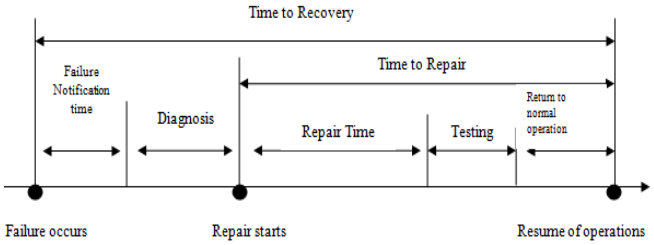

Fig.1. Pictorial representation of MTTR

- *Mean Time Between Failure (MTBF):* It is the time elapsed between the previous failure and the next failure occurred. It is calculated as:

MTBF = total operational time/ total number of failures.

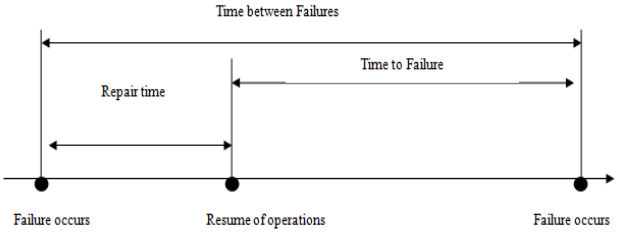

Fig.2. Pictorial representation of MTBF

- *Mean Time To Failure (MTTF):* It depicts the lifetime of a device where it is expected to work in that period until it fails. It is used to refer to non- repairable devices. It is collectively calculated for similar devices:

  MTTF = total hours of operation/ total number of units

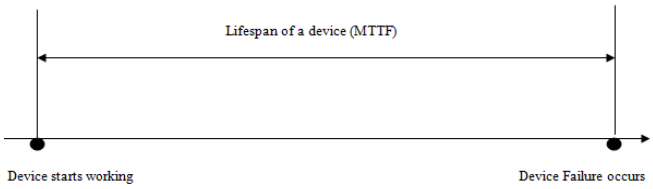

Fig.3. Pictorial representation of MTTF

IV. HEARTBEAT STRATEGY

Heartbeat strategy is a failure detection approach where live processes send an "I am alive" message to the host device at fixed intervals which ensures that the node is working fine. If the message arrives late, then that process is removed from the suspected processes list otherwise it remains suspected.

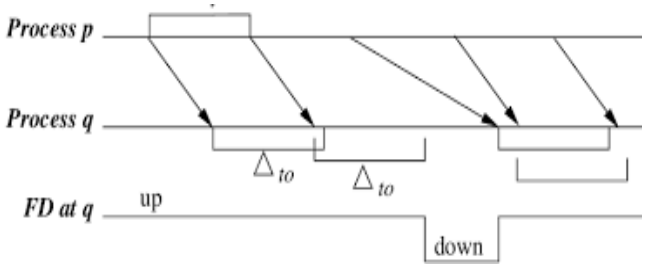

Fig. 4. Heartbeat arrival by Bertier, Marin(2002)

Here in fig., process p sends heartbeat signals in fixed time intervals to process q, If they are not received then the Fault detection mechanism (FD) declares that node to be down(not working).

V. LITERATURE SURVEY

(Mylara Reddy Chinnaiah, N. N. 2018) proposed fault-tolerant techniques which are based upon the frequency of interaction requests and all the configurations supported by the software systems. Configurations are differentiated into critical and noncritical; therefore, two parameters, frequency of configuration interactions (IFrFT) and characteristics and frequency of interactions (ChIFrFT) are introduced. IFrFT technique uses interaction values of configurations to measure the reliability and performance of software. ChIFrFT technique monitors critical operations of specific software such as payment transactions. It is then proved that a large number of interaction requests or failed critical requests both can cause software failure.

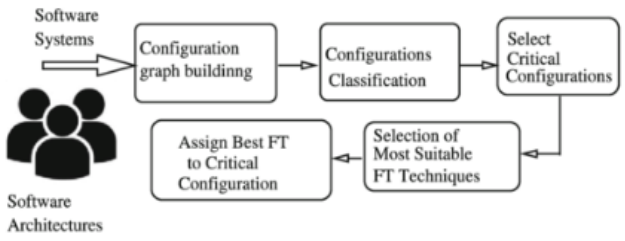

Fig.5. The system architecture of proposed work in (Mylara Reddy Chinnaiah, N. N. 2018).

(Sudha, 2013) proposed a model FTC, fault tolerance in cloud computing for real-time applications running on cloud. It starts with creating a variant of the virtual machine which is named as an adjudicator node here. The virtual machine has to pass the acceptance testing while the adjudicator acts as a time checker and is used in reliability test and decision making. A virtual machine is chosen for processing only if it passes the reliability test or otherwise it can be removed. It is a proactive fault tolerance model and it has good forward recovery methods.

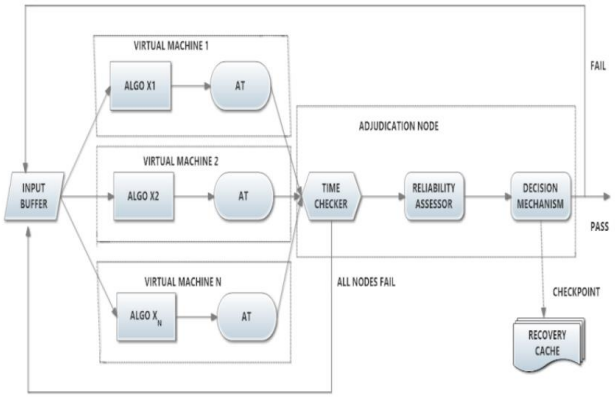

Fig.6. FTC model proposed by (Sudha, 2013)

(Jialei Liu, OCTOBER-DECEMBER 2018), proposed a PCFT approach that is proactive in nature and based on particle swarm optimization (PSO). It targets IaaS cloud structure employed on fat-tree topology architecture. The objective is to detect and monitor a deteriorating physical machine (PM) and then this system searches for optimal physical machine so that the Virtual machine(VM) from the defected PM can be migrated to this new PM. CPU temperature model is used here for fault prediction in PM.

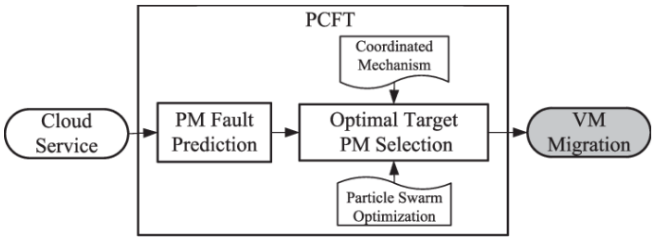

Fig.7. PCFT approach proposed by (Jialei Liu, OCTOBER-DECEMBER 2018)

(Kalanirnika G R, 2015)proposed a VM-µ Checkpoint mechanism that follows CoW-PC(Copy on Write- Presave in Cache) algorithm. It is a reactive technique in which after the occurrence of failure in a VM, it is saved and then the service is recovered from the last checkpoint marked in cache memory. It also follows memory exclusion technique to optimize performance of check pointing in memory.
(Zeeshan Amin, April 2015)used Heartbeat strategy along with Artificial Neural Network as a proactive fault tolerance mechanism. This algorithm proposes an arrival time of the next heartbeat message along with the safety margin, α. The equation formulated in this approach is:

$$TO = ET + \alpha ,$$

Where, TO: Time interval between two heartbeat messages, ET: Estimated time of arrival of next heartbeat message.

(Ghamdan Mohammed Qasem, 2018) proposed a Fuzzy min-max Neural Network (FMNN) approach based on proactive fault tolerance. It predicts if a virtual machine will fail or not depending upon its resource usage. The overlapping data from the FMNN classifier is then fed as input to KNN (K nearest neighbour algorithm) classifier which then tells whether it is failure or success. (Sam Goundar, July-September 2018) studied various fault tolerance techniques and addressed four different fault cases. These were: if the end-user is not able to log in and access the cloud service, or if the end user's request to the cloud server is lost or if the server has crashed after receiving the user's request and the last fault case studied is if the end-user loses the server reply.

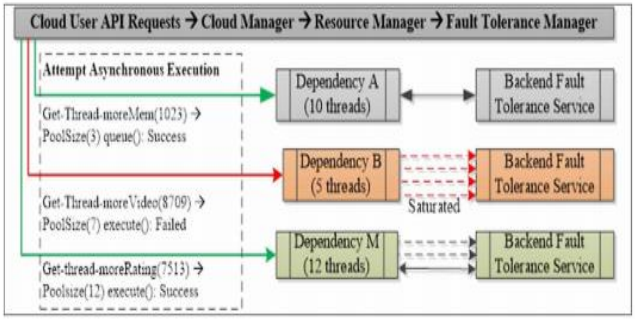

Fig.8. Fault Tolerance Cloud architecture proposed by (Sam Goundar, July-September 2018)

(A. Marahatta, 2018)proposed an energy-aware proactive fault-tolerant scheduling scheme for cloud data centres. Firstly they used a machine learning prediction method to learn to differentiate between failure-prone tasks and non-failure prone tasks so that the failure rate can be predicted. After this vector reconstruction method is used to reconstruct failure-prone tasks and project both corrected failure-prone tasks (called super tasks here) and non-failure-prone tasks to most appropriate host.

(Mohd Noor, 2010) proposed an extended heartbeat mechanism for fault detection in which index server and pinging service for each unindexed process are implemented to prove that node has failed and that the fresh time interval is too long to realize this.

VI. METHODOLOGY

- Primarily, a detailed study is carried out to understand the existing strategies for fault tolerance and figure out their execution and energy analysis.
- To enhance the Heartbeat strategy, a simple track, called confidence factor, α, has to be maintained that how many times a node has been marked as suspected and how many times it has been recovered. This simple tracker will keep count of a node's accessibility and reliability over the network.

If a node is marked as suspected then its network parameters can be studied and monitored so that it can be identified that whether a network fault has occurred and how to recover from it. This will help the fault tolerance units in real-time fault localization and fault detection.

If the confidence factor rises above a certain threshold then that node is escalated and passed for confidence testing or in some specific cases, marked for fault recovery and its services need to be migrated to a more suitable host.

- *Confidence testing:* If the output from primary testing indicates node failure, then that node is checked for its overload. If it has some major crucial processes running over then instead of node escalation, node has to pass another single step of primary testing, executed for only one confidant time interval. If node fails this last time then Failure recovery and immediate job migration are done.

This mechanism follows a proactive fault tolerance strategy. The physical machine can be hardcoded to keep only a fixed number of past reports, only for comparison purposes.

Even after the load is removed from the faulty node, it should continue preparing and sending the parameters so that it can be monitored and recovered.

This method relatively supports Energy efficiency as the proactive failure detection method is simple and learns and heals itself. User intervention is least needed in this strategy. If we reduce the number of failures and correct our practices by learning from this report then we can increase hardware life and reduce the use of further fault recovery and repair methods, thus saving energy. Energy efficiency leading to fewer carbon footprints eventually supports Green Cloud Computing. The following algorithm describes this process in brief:

Algorithm 1: Primary and confidence testing of nodes.

1: Begin.
2: Initialization:
3: T(d) –Set the time interval of heartbeat message arrival
4: α – Set the confidence factor
5: T(h) – Set the initial threshold value
6: H(n) – Set the heartbeats (per T(d)) counter
7: n – number of time slots per T(d)
8: *Step 1:* Collect relevant data.
9: If Heartbeat is received for time T(d)
10: Then α = 1, H(n)= H(n) + 1
11: *Step 2:* Create sampled data for each H(n)
12: After a specific time,
13: If α > T(h)
14: Then check for node overload and start confidence testing.
15: *Step 3:* Initiate Fault recovery.
16: After n consecutive T(d), analyze sampled data collected in step 2 and after that decrease the delay for even time slots and increase the delay for odd time slots and vice versa
17: Repeat from step 5.
18: End.

Using this algorithm, we can ensure that our tool will be able to detect node failure or a smaller network connectivity issue. Fault detection is completely dependent here on heartbeats. Decreasing time of arrival of heartbeats will increase frequency of heartbeats received, which is good. But it will also cause overheads.

Analyzing confidence factor here and executing confidence testing, along with changing time delays will improve energy consumption and brings accuracy in fault localization.

VII. ANALYSIS OF FAULT DETECTION TECHNIQUE

After analysis of fault detection, proactive fault tolerance tools and strategies, it is observed that the heartbeat strategy is used in some of them and we have enhanced it to a certain limit. We have analyzed the current techniques, refer table-I. Its intervals of message arrival are altered, indexing is done, its frequency is optimized, and the focus is to minimize the energy and resource wastage at the time of failure. Therefore, in this paper, a tool is organized which will efficiently detect failure in nodes, by analysing the frequency of failed processes. Confidant failure detection reduces energy wastage.

1. In the analysis we found that the fault detection models using heartbeat mechanism only focussed on heartbeat messages and little attention was paid towards its energy efficiency factor.
2. Some of the fault detection methods are bulky and do not contribute towards green computing.
3. Some of the fault prediction models focus on only one cause of failure such as Fuzzy min-max Neural Network (FMNN) approach proposed by (Ghamdan Mohammed Qasem, 2018) where only resource usage is focussed to predict failure.
4. Some of the failure prediction techniques are used only to select a proper working virtual machine before assigning load to it. Although failure prediction should be an ongoing strategy even after the correct virtual machine is chosen.

VIII. CONCLUSION

After analysing various fault tolerance techniques, we felt a need to improve it. More precisely, fault prediction strategy needs improvement because if a fault is predicted and prevented earlier then a lot of energy, time and cost can be saved and thus our cloud will become greener. The algorithm followed here will improve the efficiency of heartbeat mechanism (a fault prediction strategy). It is easy to be implemented in cloud. It will not only detect failed processes but can also recognise a virtual machine which may fail soon. Thus failure prediction for nodes is made efficient and simple using this approach.

Table- I: Comparison of few fault tolerance techniques.

Techniques	Availability	Downtime	Failure Prediction	Policy	Limitations
Interaction and configuration Frequency (Mylara Reddy Chinnaiah, N. N. 2018)	Yes	Yes	No	Reactive	Focus is on fault tolerance and recovery.
FTC model based on adjudicator node (Sudha, 2013)	Yes	No	Yes	Proactive	Failure prediction is applied to select virtual machine for computation. After processes start, failure prediction is not defined.
Extended heartbeat mechanism (Mohd Noor, 2010)	Yes	Yes	Yes	Proactive	Indexing and then pinging may cause overhead, which will deteriorate energy efficiency
Fault Tolerance Manager (Sam Goundar, July-September 2018)	No	Yes	Yes	-	Fault tolerance method is only based on four cases of Cloud User API Requests. Fault prediction beyond this is not defined.
PCFT approach using Particle Swarm Optimization(PSO) (Jialei Liu, OCTOBER-DECEMBER 2018)	Yes	No	Yes	Proactive	Detects failure of deteriorating Physical machine using CPU temperature model. Generalized faults that can occur on any physical machine are not focused.
Fuzzy min-max Neural Network (FMNN) approach (Ghamdan Mohammed Qasem, 2018)	No	Yes	Yes	Proactive	Failure detection completely depends on resource usage of virtual machine. Only Resource usage may not accurately predict failure.
Heartbeat strategy using confidence factor (our technique)	Yes	Yes	Yes	Proactive	Only Network parameters are considered for fault recovery after fault prediction is completed.

1. Proactive fault tolerance strategy using Heartbeat mechanism will prove to be efficient in fault localization and fault detection method.
2. In heterogeneous clouds, fault localization is the biggest hurdle; this mechanism will be efficient in detecting the fault in virtual machines in such clouds.
3. In the era of HPC (High-Performance Computing), where the cloud is dynamic, scalable and resources are virtualized, this mechanism will record each heartbeat of the system and maintain a consistent close watch on virtualized servers.

4. This fault prediction system will work even for Real-time applications running on virtual servers as we are continuously keeping a log of virtual server and its dynamic services.

REFERENCES

1. Amin, Z., Sethi, N., & Singh, H. (April 2015). Review of Fault Tolerance Techniques in Cloud Computing. *International Journal of Computer Applications*, 11-17.
2. Bertier, Marin & Marin, Olivier & Sens, Pierre. (2002). Implementation and Performance Evaluation of an Adaptable Failure Detector. 354-363. 10.1109/DSN.2002.1028920.
3. Bertier, Marin & Marin, Olivier & Sens, Pierre. (2004). Performance Analysis of a Hierarchical Failure Detector. 10.1109/DSN.2003.1209973.
4. Bhardwaj, Akashdeep & Goundar, Sam. (2018). Efficient Fault Tolerance on Cloud Environments – A Survey. International Journal of Computers and Applications. 7.
5. Cui, Xiaoqing & Ma, Zhifeng. (2019). Dynamic heartbeat detection algorithm based on RBFNN. The Journal of Engineering. 10.1049/joe.2019.0050.
6. Garraghan, Peter & Moreno, Ismael & Townend, Paul & Xu, Jie. (2014). An Analysis of Failure-Related Energy Waste in a Large-Scale Cloud Environment. Emerging Topics in Computing, IEEE Transactions on. 2. 166-180. 10.1109/TETC.2014.2304500.
7. Ghamdan Mohammed Qasem, D. M. (2018). A Classification Approach for Proactive Fault Tolerance in Cloud Data Centers. *International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 22*, 15762-15765.
8. Guo-jian, Peng & Ze-hua, Liu & Guo-min, Chen & Chen-hui, Luo. (2013). A novel self-regulatory failure detection algorithm for distributed storage systems. 674-678. 10.1109/IMSNA.2013.6743366.
9. Haider, S., &Nazir, B. (2016). Fault tolerance in computational grids: perspectives, challenges, and issues. *SpringerPlus*, *5*(1),1991. doi:10.1186/s40064-016-3669-0
10. Hayashibara, Naohiro & Défago, Xavier & Yared, Rami & Katayama, Takuya. (2004). The φ accrual failure detector. 10.1109/RELDIS.2004.1353004.
11. Hosseini, Seyyed&Ghobaei-Arani, Mostafa. (2015). Fault-Tolerance Techniques in Cloud Storage: A Survey. International Journal of Database Theory and Application. 8. 183-190. 10.14257/ijdta.2015.8.4.19.
12. https://limblecmms.com/blog/mttr-mtbf-mttf-guide-to-failure-metrics/
13. Jan Seeger, Arne Br¨oring, and Georg Carle. 2016. Optimally Self-Healing IoT Choreographies. 1, 1, Article 1(January 2016), 18 pages.DOI: 10.1145/
14. Jialei Liu, S. W. (OCTOBER-DECEMBER 2018). Using Proactive Fault-Tolerance Approach to Enhance Cloud Service Reliability. *IEEE TRANSACTIONS ON CLOUD COMPUTING, VOL. 6, NO. 4*, 1191-1202.
15. Kalanirnika G R, V. (2015). Fault Tolerance in Cloud Using Reactive and Proactive. *International Journal of Computer Science and Engineering Communications*, 1159-1164.
16. Kaur, S., & Singh, G. (2017). Review of Fault Tolerance Techniques in Cloud Computing. *International Journal of Engineering and Management Research*, 69-71.
17. Kenga Mosoti Derdus, Vincent Oteke Omwenga, Patrick Job Ogao, "Causes of Energy Wastage in Cloud Data Centre Servers: A Survey ", *International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT)*, ISSN: 2456-3307, Volume 5 Issue 3, pp. 416-430, May-June 2019. Available at doi: https://doi.org/10.32628/CSEIT1953139 Journal URL: http://ijsrcseit.com/CSEIT1953139
18. Kumari, P., & Kaur, P. (2018). A survey of fault tolerance in cloud computing. *Journal of King Saud University - Computer and Information Sciences*.
19. Madani, S. S., & Jamali, S. (2018). A comparative study of fault tolerance techniques in cloud computing. *International journal of research in computer applications and robotics*, 7-15.
20. Marahatta, C. C. (2018). Energy-aware Fault-tolerant Scheduling Scheme based on Intelligent Prediction Model for Cloud Data Center. 2018 Ninth International Green and Sustainable Computing Conference (IGSC), Pittsburgh, PA, USA, 1-8.
21. Mohamed, S., & Hemayed, E. (2015). Fault tolerance in cloud computing - a survey. *11th International Computer Engineering Conference (ICENCO).* Cairo, Egypt: IEEE.
22. Mohd Noor, Ahmad Shukri & Mat Deris, Mustafa. (2010). Extended Heartbeat Mechanism for Fault Detection Service Methodology. 10.1007/978-3-642-10549-4_11.
23. Mylara Reddy Chinnaiah, N. N. (2018). Fault-tolerant software systems using software configurations for cloud computing. *Journal of Cloud Computing volume 7, Article number: 3*.
24. NazariCheraghlou, Mehdi & Khadem-Zadeh, Ahmad & Haghparast, Majid. (2015). A Survey of Fault Tolerance Architecture in Cloud Computing. Journal of Network and Computer Applications. 61. 10.1016/j.jnca.2015.10.004.
25. Sudha Lakshmi S. Fault tolerance in cloud computing.International Journal of Engineering Sciences Research-IJESR, Vol 04, Special Issue 01, (ACICE) 2013.
26. Tian, Dong & Chen, Shuyu & Chen, Feng. (2006). A Dynamic Fault Detection Algorithm under Grid Environments. Journal of Computer Research and Development. 43. 1870-1875.
27. Tomsic, Alejandro & Sens, Pierre & Garcia, João & Arantes, Luciana & Sopena, Julien. (2015). 2W-FD: A Failure Detector Algorithm with QoS. 885-893. 10.1109/IPDPS.2015.74.
28. Turchetti, Rogerio & Duarte Jr, Elias & Arantes, Luciana & Sens, Pierre. (2016). A QoS-configurable failure detection service for internet applications. Journal of Internet Services and Applications. 7. 9. 10.1186/s13174-016-0051-y.
29. Zang, Xiuhuan & Chen, Wei & Zou, Jing & Zhou, Sheng & Lisong, Huang & Ruigang, Liang. (2018). A Fault Diagnosis Method for Microservices Based on Multi-Factor Self-Adaptive Heartbeat Detection Algorithm. 1-6. 10.1109/EI2.2018.8582217.

AUTHORS PROFILE

Shelly Prakash has completed Master in Computer applications and is a research scholar at present in Department of Computer Science, Banasthali Vidyapith. Her research is based on efficient Fault tolerance techniques for greener clouds.

Dr Vaibhav Vyas has done his Doctral from Banasthali Vidyapith. His research areas are Data Analytics, Cloud Computing, Software Engineering and Programming languages. Currently wokring as Associate Proessor in Department of Computer Science, Banasthali Vidyapith.

Dr. Anoop Kumar has done his Ph.D. from Banasthali Vidyapith,Rajasthan Currently he is working as Assistant Professor of Computer Science, Engineering and IT department, Banasthali Vidyapith