An Efficient Fault Tolerance Mechanism using TCSA to Ensure Reliabilty in Cloud

Abhilash Prasad, Santosh Anand, Somnath Sinha

ABSTRACT--- Distributed computing gives on-request benefit in computing innovation. This innovation offers a combination of programming and assets which displays dynamic adaptability in nature. These frameworks are advantageous; however, there is likewise a burden for the framework. In the event that the framework has side effects of disappointment, it doesn't give its usefulness. So, the arrangement is to blame expectation and relief. It gives the ability to the framework to respond easily when the framework has turned out badly or any surprising equipment or programming disappointment in the association. The paper portrays a higher objective behind faulty speculations and mitigation techniques, schemes, and how the cloud analyst uses the new novel method to solve these problems. In this document, fault prediction will be overcome with the schemes speculated blame for faults and mitigation will be clarified and achieved.

Keywords - TCSA, Fault Tolerance, Cloud Computing

INTRODUCTION

"Cloud computing" as another worldview in broadly appropriated figuring in the world of infrastructure developed for cyber infrastructure in the areas of virtualization, Cloud computing, utility processing in networks, web administration and the software services for actualizing a service-oriented architecture [1,2]. On the request to diminish generally speaking it is solely dependent on the end user i.e. the Client, a mutual set for processing assets is required to give vigorous adaptability and decrease the absolute expense of possession for the services which demands high intense results [3,4]. Obviously Cloud computing can associate with the Internet and broad system so as to utilize assets that are accessible remotely, and thus, pay-per-use helps to produce the present effective arrangements [5, 6]. It becomes the supreme factor in cloud computing due to its diverse and rapid exponential growth and fast development fulfilling all the demands. As there will be a huge traffic of data to ensure reliability fault tolerance plays on among the major thoughts.

Adapting to the fault-resolving mechanism involves all the necessary steps for strict system reliability and heartiness as well. The probability of the system to terminate internally or fail could be reduced to a large extend using fault tolerance considering the fact that it produces better results in dynamic improvements in the execution time of the system, failure recovery and an economically lower budget cost [7]. Meanwhile unmistakably cloud environment has diversified in short term run which was able to develop a circulated application which was severely developed to improve the layers of virtualization where a designing application where able to reflect effective adaptability [8]. The framework highlights the work which consists of accessibility and reliability of course as mentioned which is depicted by the frameworks QoS [9].

The user input the jobs to the scheduler through cloud services. The TCSA algorithm gets invoked within the resource mediator at once, when the scheduler receives the input jobs. The coordination between resources and resource broker is established by Cloud Information Server (CIS) and fault handler. The fault handler identifies the faulty resources in cloud. CIS also enables frequent updating of the status of the resources. The copy of the status will be sent to resource allocator.

Once the user submits jobs to the scheduler, it splits the job into small tasks and send to task controller. The task distribution among available idle resources is coordinated by the task controller with the help of resource allocator. Task controller takes the information regarding the idle resources with the help of resource allocator from CIS and allocates these tasks to the resources in the order of their arrival. The TCSA allocate tasks to resources in a dynamic manner for reducing the time of execution and maximizing the utilization of resources.

The resource allocator contains all information and updated status regarding grid resources sent by CIS. The function of the timer is to allocate the time to each task assigned to grid resources. When the assigned task is finished, timer will calculate the time taken for the completion of that particular task and pass the results back to the scheduler. Once any fault is identified among any one of the assigned resources, scheduler analyses the load of the remaining available resources and rescheduling is performed by reallocating the task of faulty resource to the resource that have the least load. Timer keeps identifying the faulty instance along with the maintenance of execution time.

TYPES OF FAULT

A framework's powerlessness is brought to play as an irregular position or as a player from a bug in at least one system to play their essential work. The shortcomings are considered to be the root cause of a mistake or a lack of fault, which gives the basic driver [10] a false belief.

Revised Manuscript Received on August 14, 2019.

Abhilash Prasad, Amrita school of arts and sciences, Mysuru, Amrita Vishwa Vidyapeetham, India. (E-mail: abprsd6657@gmail.com)
Santosh Anand, Amrita school of arts and sciences, Mysuru, Amrita Vishwa Vidyapeetham, India. (E-mail: santoshanand.jha@gmail.com)
Somnath Sinha, Amrita school of arts and sciences, Mysuru, Amrita Vishwa Vidyapeetham, India. (E-mail: sosin.mca@gmail.com)

Hardware Fault: The vast majority of the fault tolerant procedures have centered towards organizing frameworks which can undertake and recoup by their own which generally exists in the modules of equipment, it includes part a computational framework which then converts into modules. When a particular module is getting filled, its usefulness can be taken over by another module assigned dynamically [11].

Software Fault: In fact, it is just like Hardware issues but it is given more priority to endure the issues over it [12]. Different dynamic and static repetition Methods have been used to overcome and accomplish for its effective utilization [11].

A. Fault Prediction and Mitigation Techniques in Cloud Computing

Proactive Fault Tolerance: This approach seeks to keep a strategic distance from the extra attempt to restore failed job and axes by anticipating previous mistakes and replacing them with other components of the job. Predictable error frameworks can meet the time constraints established by real-time frameworks [13].

Pre-emptive tolerance of error: This approach seeks to keep a strategic distance from the extra attempt to restore failed job and axes by anticipating previous mistakes and replacing them with other components of the job. Predictable error frameworks can fulfill the real-time time limitations [13].

Interactive error tolerance: When a real error occurs in the frame, at this point this strategy helps reduce the impact of disappointments on the operating frame. These strategies give a large fault-tolerant response to the application of general computing; however, there is a problem with this strategy that it cannot meet the time requirements set by continuous registration frameworks [13].

B. Types of Proactive Fault Tolerance

Software Renewal: Software renewal is the way an application is terminated immediately and later rebooted as a flawless case. There is a remarkable reclamation between the time and the application restarts at each break with a clean internal state. In this technique, unprepared reboot operations become ready for the structure [14].

Using Self-Healing: While many parts of a single framework work on different VMs, and when blame is taken by taking advantage of self-recreation, disappointment can be dealt with in different application situations [14].

Using the Proactive Migration: By using this method, moving the parts of the framework that runs on an arithmetic hub will fail to multiple axes. By using a preventive motion, the application is moved to an alternate hub before a real error occurs [14].

C. Types of Reactive Fault Tolerance

Check pointing/Restart: During job planning, this approach is used. To differentiate the rate of blame, the recording is included. Due to the re-running of the promise at the stage last inspected, these techniques take less time and less time. No need to restart the entire job [14].

Replication: Replication is a strong method for fault tolerance. There are distinct variants on distinct resources of any application or assignment. While any error occurs in the frame, implementation continues until all re-created tasks are eliminated. [14]

Job Migration: At the point where the task of neglect or Machine negligence is due to a negligent job, the job is transmitted to another device by default. Wherever the method remains to be implemented. [14]

Sguard: Requires less project time for standard operation and more resources available for free. Depends on restoration. Requires less investment for the normal operation and makes a lot of assets here [14].

Retry: Most importantly, the unused task will again be performed on a comparable identical asset from the starting point on the basis of the strategy the discarded approach. On off chance that the undertaking keeps on changing on various executing focuses, when they dispose off the extra overheads, they will process the normal execution time. The limit value is set to limit the amount of retries on a comparable machine for neglected jobs [14].

Task Resubmission: Task resubmission suggests to recommit or to find Similar machine or alternate machine must be assigned when the work fails. Client characterized special case taking care of the point when any assignment/work flops the consumer then provides efficient solution to the unsuccessful job process employments. In special case, taking care of client can compose code into the attempt, get lastly square to tosses exemption and it handles the special case [14].

Rescue Workflow: This strategy allows the workflow to continue to pay little attention to the possibility that the quest will miss the mark until you have the opportunity to be difficult to move forward without cooking the failed mission.

FAULT-TOLERANCE MODELS IN CLOUD COMPUTING

AFTRC: fault tolerant failure display to continuous distributed computing dependent on the way that an ongoing Framework can take advantage of the computing restriction and the versatile virtualized situation of distributed computing to make the continual implementation more effective. In this suggested model, the structure proactively endures the fault and expresses the unwavering value of the hubs that are being prepared [15].

LLFT: is a propose model which contains a low latency fault-tolerance (LLFT) middleware for giving adaptation to internal failure to appropriated applications sent with in the distributed computing condition as an administration offered by the proprietors of the cloud. This model depends on the way that one of the principle difficulties of distributed computing is to guarantee that the application which is running on the cloud without a break in the administration they gave to the client. This middleware imitates application by the utilizing of semi-dynamic replication or semi-latent replication procedure to secure the application against different kinds of issues [16].

FTWS: Is a suggested model containing a fault-tolerant job process calculation for indoor failure by using replication and resubmission of undertakings depending on the need for tasks in a heuristic matrix. Such a model is based on how many tasks in some requests are prepared based on the information and control reliance. In a cloud domain, the reservation of the work method included in the assignment deception is being evaluated. FTWS duplicates and schedules the undertakings to fulfill the deadline [17].

FTM: is a proposed model to beat the confinement of existing approaches of the on-request benefit. To accomplish the dependability and flexibility they propose an imaginative viewpoint on making and overseeing adaptation to non-critical failure. By this specific system client can determine and apply the craving dimension of adaptation to non-critical failure without requiring any learning about its execution. FTM engineering this can basically be seen as a collection of a few web administrations segments, each with a particular usefulness [18].

CANDY: is a segment base accessibility displaying outline work, which develops an exhaustive accessibility demonstrate semi naturally from framework particular depict by frameworks demonstrating dialect. This model depends on the way that high accessibility confirmation of cloud benefit is one of the fundamental normal for cloud benefit and furthermore one of the primary basic and testing issues for cloud specialist provider [19].

Vega-warden: The board structure is a standardized client, providing a global client room for multiple virtual foundations and implementation advantages in distributed computing conditions. This model is created for the distributed computing situation of the virtual group base to overcome the two problems: comfort and safety result from frame sharing [20].

FT-Cloud: Are a segment-based casing work and its design for cloud application construction. In order to acknowledge the section, FT-Cloud uses the part summon framework and recurrence. Thus, a calculation for deciding naturally to adapt to inner failure comes into being securely [21].

MAGI-CUBE: High strong and small storage design for distributed computing. Which assemble the framework on the highest point Use HDFS as a capability structure for recording managers reading / composing and metadata. They also manufactured a record scripting and fixed section for autonomous job in the back floor. This model depends on how the 3-clashing section of the capability structure is elevated unwavering quality and implementation and ease (space). To offer a particular Magi 3D shape to these offices is proposed [22].

TCSA: proposes a new method of fault tolerance with the help of TCSA algorithm. The primary step is resource allocation which helps to reduce the make span and load balancing among resources effectively. Job allocation is done dynamically along with minimized job execution time and maximized resource utilization time. During the job allocation, if any faulty resources are identified, then the tasks allocated to those faulty resources are reassigned to some other resources with the least load.

RELATED WORK

Amal Ganesh et al. (2014) suggests that the advent of cloud computing has given the IT world a fresh dimension. While cloud computing provides benefits such as agility, on-demand resource allocation, cost reduction, and various rentals, hazards and pitfalls remain. One of the major research challenges of cloud computing is ensuring continuous reliability and availability of resources. Cloud computing therefore needs a powerful error tolerance scheme (FT). You need to know the distinct kinds of mistakes to better comprehend FT in cloud computing. In this document, we illustrate the fundamental concepts of fault tolerance by knowing various FT approaches (such as interactive FT strategy and proactive FT strategy) and FT methods linked to various kinds of fault tolerance. Research was carried out on multiple error-tolerant techniques, algorithms and frameworks created and enforced in this sector by research specialists. This is an ongoing study area that will guide us in constructing powerful cloud-based FT methods. In this paper, they concentrate on cloud computing's notion of normative tolerance.

Alan Chana et al. (2012) Proposed an error-tolerant strategy in which cloud clients and vendors share responsibility for providing the necessary tolerance for failures. Application failures can be identified at the customer stage, according to Tchana. However, it is possible to detect a virtual machine (VM) and to disrupt and repair systems at the cloud provider stage. Applications running on VM computers for recovery / recovery may be asked to be renewed and performed at client level. To generate restore points for the restored VM, a checkpointing method is used [23]. In this article, cloud provider level mistakes can be identified and fixed. Applications running on VM computers for recovery/recovery may be asked to be renewed and performed at client level.

Wenping Chao et al. (2012) proposed an error-tolerant Low Error Tolerance Framework (LEFT) that uses the Replication / Replication method to replicate the application process. To guarantee accurate communication between replication procedures, the frame is fitted with a LEFT message protocol. The LEFT membership protocol guarantees that a coherent member display is available for the entire replication process group [24]. In this article, you use the Replication method for the follower / followers to handle the tasks that bear the errors.

Ravi Jahawar et al. (2012) pointed out a way to use the virtualization layer to provide the application with the required features for error tolerance as a service upon request. This is achieved by adding a service layer that acts as an intermediate error-tolerant (FTM) program that offers the characteristics needed to facilitate its application conversion [25]. To manage fault-tolerant employment, they use on-demand services.

S Gokuldev et al. (2018) focuses on performing job scheduling using TCSA while tolerating failures that occur during the process. Job assignment is done with the help of a resource agent and a grid system consisting of GIS. Once

the failed resource is identified, the scheduler analyzes the load of the remaining idle resources and rearranges the failed job to the new set of resources with the least current load. The focus of this work is to tolerate failures that occur during task assignment and to perform rescheduling by redistributing tasks to different resources, which helps the system achieve significant results when the system is exposed to high loads.

Problem Statement

The problem of the handling fault jobs is corrected by a approach of the defected prediction and mitigation methods and rescheduling the job to queue for reprocessing. The system proposes a new method of fault tolerance with the help of TCSA algorithm. The primary step is resource allocation which helps to reduce the make span and load balancing among resources effectively. Job allocation is done dynamically along with minimized job execution time and maximized resource utilization time. During the job allocation, if any faulty resources are identified, then the tasks allocated to those faulty resources are reassigned to some other resources with the least load.

Proposed System

The primary reason for the proposed calculation is to enhance the execution of the cloud through limiting both the time spent by the application in the cloud and the impact of disappointment whenever happened. The calculation relies upon choosing VMs and Cloud Analyst that possess the most punctual completing energy for client applications. Likewise, it relies upon utilizing the replication system to produce numerous duplicates of a similar application to be executed on various VMs and Cloud Analyst, at the same instant.

The segments incorporate the expedite, the VM checking server, the replication chief and the cloud VMs and Cloud Analyst. Customers or clients present their applications or job or task alongside their QoS prerequisites to the cloud through the cloud entryway. The occupations will be embedded in the representative line. In addition to its necessary QoS, the expert will receive a vocation from the representative line. At that stage, a rundown of appropriate VMs and Cloud Analyst to execute the activity will approach the VM Monitoring Server. The server will answer for the application of the client with a rundown of VMs that can play out the application in addition to their normal completion times. The officer will sort this rundown on each VM as per the application's completion moment. The first VM is selected as the fundamental VM to perform the activity in the arranged rundown, which is the VM with the timeliest completion time.

The completion time of a VM i for an occupation j is characterized as:

$$FT\ (j, i) = t\ (j, i) + ST\ (j, i), \qquad (1)$$

Where t (j,i) is the moment of execution of occupation j on VM I and ST (j,i) is the moment of execution of job j on VM i. The estimate of the ST (j, i) is the summation of the execution time of the considerable number of occupations allocated to the VM I and executed or to be executed before task j.:

Where t (j, i) is the time when j is executed on VM I and ST (j, i) is the time that j will start on VM i. The estimate of ST (j, i) is a summary of the implementation time for a large number of VM I assigned occupations and implemented or performed before task j and can be well characterized by

$$ST\ (j,i) = \sum_{n=1}^{j-1} t\ (n,i) \qquad (2)$$

Since the chose principle VM may come up short, the framework will pick some different VMs from the rundown on which duplicates of the activity will be executed.

Re-establishing a career can help manage disappointment. At this point, when the VM comes short, the uncooperative effect can be reduced to the implementation Application that passes without disappointment if the tradition is finished. The number of entries should not be too high to stay away from the cloud. The number of copies depends on the VM disappointment rate and the quality of the service. The disappointment rate of VM is a depiction of the history of disappointment from it. Accept fi are the occasions in which VM j is neglected to finish functions and ni is the absolute number of occupations assigned to VM j. The VM i's disappointment rate is as follows:

$$fr_i = \frac{f_i}{n_i}, \quad \text{where} \quad 1 \leq \text{fr}_i \leq 0.$$

While giving his QoS prerequisites, the client chooses whether or not the customer wants to replicate the request. This can be achieved by a factor called rep, the client's esteem for the cloud server provider. On the off chance that the estimation of rep = 0, at that point the client does not have to recreate his application. On the off chance that the client needs to repeat his application he will set rep = 1.

On an opportunity to exit from rep = 1, Replication Manager uses the VM default disappointment rate to determine the number of constraints for client applications. It checks the fr rating of the VM principle. If the estimate of fr is not equal to k, it will include additional cloning of the application. If the estimate of fr is more pronounced than the k level, it will contain two additional versions of the application. The k estimate is determined by a cloud-specific cooperative as demonstrated by the abundance of VMs that can execute the application without affecting the allocation of different applications.

Tolerating the faults that are occurred during task allocation and performs rescheduling by reallocating the tasks to different resources which helps system to handle high loads to obtain very significant results.

Steps to follow

Step 1 - Fetch the job list.
Step 2 – Calculate the execution time.
Step 3 – Identify the VMS
Step 4 – Allocate VM resources
Step 5 – Identify the disappointment level
Step 6 – Reallocate to queue
Got to step 2 until the disappointment level is 0
Step 7 – Stop

Algorithm: FT-JS (Fault Tolerant –Job Scheduler)
Input: Tasks and resources.
Output: Process span and resource utilization. Initialize the resource-list [No. of resources] Initialize the job-list [No. of tasks]
while job -list is not empty do task 'J' from front of the job-list is taken for each task 'Ji'
doVMi random nodes are selected uniformly.
VM0(J0), VM1(J1),)… VMn(Jn) and jobs are allocated.
Set rep=0 for all jobs.
for current load Allocate Job Ji to the VMi with the less load
if there is a tie for the least loaded node then Allocate Job J to randomly chosen among them
end if Fault-list is initialized [No. of faulty resources]
Allocated Job 'Ji' is taken from faulty resource.
Set rep=1;
Send Jobs back to queue
if idle resources available Query nodes for current load
Reallocate Job J to the node with less load
if there is a tie for the least loaded node then Reallocate Job J and split and allocate VM to randomly chosen among them
end if
else
Wait until idle resources with least load are available.
end if
end for
end while

When the user submits the job, the primary step done by the scheduler is splitting the job in to small tasks. Task controller will be having a task list from where, each task, which is ready for execution, is taken from front of the list. The resource allocator will be maintaining a resource list from where 'd' grid nodes are randomly selected from available nodes. Scheduler sends a query message to each of the 'd' grid nodes. As a result, scheduler gets the current load information of each of the d nodes. Finally, the scheduler allocates the task to the least loaded of d randomly selected nodes.

During the process of task allocation, there will be situations where the resources will not be able to execute the tasks that are assigned to it. This can be because of the reason that the assigned resources are already loaded with tasks. In such a case, the occurred fault prevents the scheduler from proper task scheduling. So, task allocator identifies the faulty resources and a list of faulty resources is made with the help of fault handler. Scheduler checks for the idle resource in resource list. Once the resource is available, the task is taken from the faulty resource of the fault-list and fault tolerance is performed by reallocating that particular task to the available resource with the least load. This helps in proper scheduling and all the tasks are executed successfully with minimum execution time.

Tolerating the faults that are occurred during task allocation and performs rescheduling by reallocating the tasks to different resources which helps system to handle high loads to obtain very significant results.

RESULT

The execution time of TCSA algorithm in both Cloud and Grid Environment have been verified and it has been observed that Cloud is more efficient and reliable based on the graph generated below.

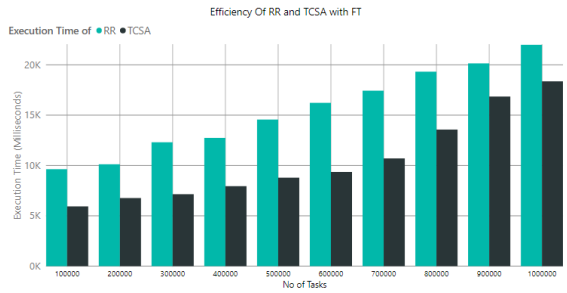

Fig. 1 Comparison of RR and TCSA With FT in Grid

To predict errors, the Cloud Analyst tool is used. Different errors can vary depending on execution parameters, for example, execution time for short time, CPU utilization cost, RAM processing cost, shallowness of the assignment. With the mentioned parameter mistake which will arrive can be betokened with the help of choice tree. This very well may be anticipated which mistake will be coming in order for its rate proportion. By choice tree the characterized the assignment definitely end up progress nor fizzled. On these off chance that the project stumbles, and that the error will occur. Each error can be moderate through special relief measures.

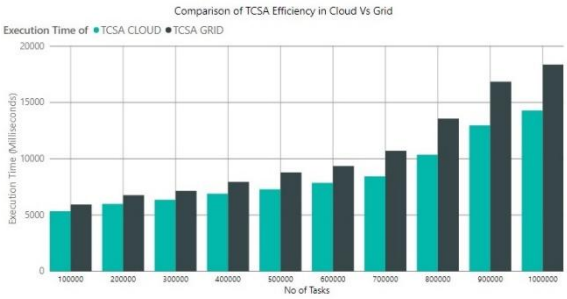

Fig. 2 Comparison of TCSA Efficiency in Cloud vs Grid

By blame expectation and relief result utilizing choice tree it can examine that specific blame can be anticipated by its FTA. For specific mistake there is specific moderation strategy characterized by ETA for one of them. By this we can predicate the blame by FTA and alleviate the blame by moderation systems.

CONCLUSION

Speculated fault prediction and novel mitigation is used to give framework accessibility, also heartiness when framework has equipment or programming shortcoming. The writing survey concentrated on the different blame forecast and relief methods and instruments utilized for executing it and look at this procedure and give the best answer for blame expectation and alleviation.

REFERENCES

1. M. A. Vouk, "Cloud Computing – Issues, Research and Implementations", Department of Computer Science, North Carolina State University, Raleigh, North Carolina, USA, Journal of Computing and Information Technology - CIT 16, vol. 4, **(2008)**, pp. 235–246.
2. B. S. Taheri, M. G. Arani and M. Maeen, "ACCFLA: Access Control in Cloud Federation using Learning Automata", International Journal of Computer Applications, vol. 107, no. 6, **(2014)**, pp. 30-40.
3. M. Fallah and M. G. Arani, "ASTAW: Auto-Scaling Threshold-based Approach for Web Application in Cloud Computing Environment", International Journal of u- and e-Service, Science and Technology (IJUNESST), vol. 8, no. 3, **(2015)**, pp. 221-230.
4. A. Fereydooni, M. G. Arani and M. Shamsi, "EDLT: An Extended DLT to Enhance Load Balancing in Cloud Computing", International Journal of Computer Applications, vo. 108, no. 7, **(2014)**, pp. 6-11.
5. Sun Microsystems, Inc., "Introduction to Cloud Computing Architecture", White Paper 1st Edition, **(2009)**.
6. M. Fallah, M. G. Arani and M. Maeen, "NASLA: Novel Auto Scaling Approach based on Learning Automata for Web Application in Cloud Computing Environment", International Journal of Computer Applications, vol. 113, no. 2, **(2015)**, pp. 18-23.
7. A. Bala and I. Chana, "Fault Tolerance- Challenges, Techniques and Implementation in Cloud Computing", IJCSI International Journal of Computer Science Issues, vol. 9, no. 1, **(2012)**, pp. 1694-0814.
8. B. Lussier, A. Lampe, R. Chatila, J. Guiochet, F. Ingrand, M. O. Killijian and D. Powell, "Fault Tolerance in Autonomous Systems: How and How much?", LAAS-CNRS 7 Avenue du Colonel Roche, F-31077 Toulouse Cedex 04, France.
9. P. Latchoumy and S. A. Khader, "Survey on fault tolerance in grid computing", IJCSI International Journal of Computer Science Issues, vol. 2, no. 4, **(2011)**.
10. H. Agarwal and A. Sharma, "A comprehensive survey of fault tolerance techniques in cloud computing," pp. 408–413, 2015.
11. P. D. Kaur and K. Priya, "Fault tolerance techniques and architectures in cloud computing-a comparative analysis," pp. 1090–1095, 2015.
12. K. Chauhan and V. Prasad, "Distributed denial of service (ddos) attack techniques and prevention on cloud environment," International Journal of Innovations & Advancement in Computer Science, pp. 210–215, 2015.
13. A. Tchana, L. Broto, and D. Hagimont, "Approaches to cloud computing fault tolerance," pp. 1–6, 2012.
14. R. Jhawar, V. Piuri, and M. Santambrogio, "Fault tolerance management in cloud computing: A system-level perspective," IEEE Systems Journal, vol. 7, no. 2, pp. 288–297, 2013.
15. Sun Microsystems, Inc., "Introduction to Cloud Computing Architecture", White Paper 1st Edition, **(2009)**.
16. W. B. Zhao, P. M. Melliar and L. E. Mose, "Fault Tolerance Middleware for Cloud Computing", IEEE 3rd International Conference on Cloud Computing, **(2010)**.
17. S. K. Jayadivya, J. S. Nirmala and M. S. Bhanus, "Fault Tolerance Workflow Scheduling Based on Replication and Resubmission of Tasks in Cloud Computing", International Journal on Computer Science and Engineering (IJCSE).
18. R. Jhawar, V. Piuri and M. Santambrogio, "A Comprehensive Conceptual System Level Approach to Fault Tolerance in Cloud Computing", IEEE.
19. F. Machida, E. Andrade, D. S. Kim and K. S. Trivedi, "Candy: Component-based Availability Modeling Framework for Cloud Service Management Using Sys-ML", 30th IEEE International Symposium on Reliable Distributed Systems, **(2011)**.
20. J. Lin, X. Y. Lu, L. Yu, Y. Q. Zou and L. Zha, "Vega Warden: A Uniform User Management System for Cloud Applications ", Fifth IEEE International Conference on Networking, Architecture, and Storage, **(2010)**.
21. Z. B. Zheng, T. C. Zhou, M. R. Lyu and I. King, "FT-Cloud: A Component Ranking Framework for Fault-Tolerant Cloud Applications", IEEE 21st International Symposium on Software Reliability Engineering, **(2010)**.
22. Q. Q. Feng, J. Z. Han, Y. Gao and D. Meng, "Magi-cube: High Reliability and Low Redundancy Storage
23. A. Tchana, L. Broto, and D. Hagimont. "Approaches to cloud computing fault tolerance." In Computer, Information and Telecommunication Systems (CITS), 2012 International Conference on, pp. 1-6. IEEE, 2012.
24. W. Zhao, P. M. Melliar-Smith, and L. E. Moser. "Fault tolerance middleware for cloud computing." In Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on, pp. 67-74. IEEE, 2012.
25. R. Jhawar, V. Piuri, and M. D. Santambrogio, "Fault tolerance management in IaaS clouds." In Satellite Telecommunications (ESTEL), 2012 IEEE 1st AESS European Conference on, pp. 1-6. IEEE, 2012.

AUTHORS PROFILE

Abhilash Prasad, completed BCA from STEMS College, Kannur University. Currently pursuing MCA in Amrita Vishwa Vidyapeetham Mysuru campus India.

Santosh Anand, completed his BE(CSE) from MIT- Manipal, MTech (CSE) from VTU-University and currently doing PhD in Amrita Vishwa Vidyapeetham. He is working as Asst. Professor in the Department of Computer Science, Amrita Vishwa Vidyapeetham, Mysuru campus, India. His research area is Wireless Sensor Network.

Somnath Sinha, completed his PhD in Computer Science from Pacific University, Udaipur, India. He also completed his Master's in Computer Application in IIEST, West Bengal, India. His research interest is in Security and different types of attack detection in MANET. He is presently working as an Assistant Professor in the Amrita Vishwa Vidyapeetham, Mysuru.