International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 6, Issue 3, March 2016)

Effective Fault Tolerance Management in Cloud Computing

Dr. Chandralekha[1], Satish Kumar Suman[2], Anupriya[3]
[1]Prof. Dept. of CSE, DRIEMS College, Cuttack, Odisha, India
[2,3]Student, Dept. of CSE, DRIEMS College, Cuttack, Odisha, India

***Abstract*- Cloud computing is an adoptable technology as it provides integration of software and hardware resources which are dynamically scalable. The dynamic environment of cloud results in various unexpected faults and failures. Fault tolerance enables a system to react gracefully to an unexpected equipment or programming malfunction. This paper explores the significance relationship between fault tolerance and system performance and develops metrics to measure fault tolerance within the context of system performance. We show the utility of designed metrics such as robustness by applying them to a sample mobile cloud computing environment consisting of multiple numbers of mobile nodes. Further robustness has been optimized using genetic algorithm and effective fault tolerance exhibited by the system has been evaluated.**

***Keywords*- Cloud computing, reactive/proactive fault tolerance, robustness, performance parameters, checks point.**

I. INTRODUCTION

The increasing demand for flexibility and scalability in dynamically obtaining and releasing computing resources in a cost-effective and device-dependent manner and in hosting applications without the burden of installation and maintenance has resulted in a wide adoption of the cloud computing paradigm. The increasing popularity of this paradigm as an attractive alternative to classic information processing systems has increased the importance of its correct and continuous operation even in the presence of faulty components. Among the many addressable issues [22] of cloud computing such as fault tolerance, workflow scheduling, workflow management, security etc, fault tolerance is one of the important issues which the researchers are supposed to look forward. Cloud computing is still vulnerable to a large number of system failures and in effect and there is an increasing concern among users regarding the reliability and availability of cloud computing services. In order to minimize the failure impact on system and application execution, failures should be anticipated and proactively handled. Fault tolerance techniques are used to predict these failures and take an appropriate action before failures actually occur.

Fault tolerance is concerned with all the techniques necessary to enable a system to tolerate different software faults remaining in the system after its development. It is carried out by error processing which have two constituent phases.

The phases are "effective error processing" which aimed at bringing the effective error back to a latent state, if possible before occurrence of a failure and "latent error processing" aimed at ensuring that the error does not become effective again. When a fault occurs, these techniques provide mechanisms to the system to prevent system failure occurrence .The main benefits of implementing fault tolerance in cloud computing include failure recovery, lower cost, improved performance metrics [1] etc. This paper aims to provide a better understanding of fault tolerance challenges and identifies various tools and techniques used for fault tolerance. When multiple instances of an application are running on several virtual machines (VM) and one of the server goes down, there is a need to implement an autonomic fault tolerance technique that can handle these types of faults.

Fault tolerance management in cloud platform can be analyzed in two different visions. The first one consists of giving both the detection and repair responsibilities to one cloud participant (exclusively) while the second is to harness the skills of the two types of participant. According to these two visions, different fault tolerance techniques have been presented [2] for the three types of failure namely hardware, VM and application.

II. FAULT TOLERANCE TECHNIQUES

Fault tolerance can be classified either as proactive or reactive in cloud computing. The proactive fault tolerance policy tries to[17] avoid recovery from fault, errors and failure by predicting them and proactively replace the suspected component means to detect the problem before it actually come. Reactive fault tolerance aims to remove the fault after it occurs and reduce the effort of failures when the failure effectively occurs. These can be further classified into two sub-techniques error processing and fault treatment. Error processing aims at removing errors from the computational state. Fault treatment aims at preventing faults from being re-activated.

A. Reactive Fault Tolerance Techniques

Reactive fault tolerance means to remove the fault after it occurs. Basically reactive fault tolerance policies reduce the effect of failures on application execution when the failure effectively occurs. These techniques are based on the following policies:

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 6, Issue 3, March 2016)

1) Check pointing/ Restart: In this scenario after doing every change in system a check pointing is done. When a task fails, rather than from the [21] beginning it is allowed to be restarted that job from the recently .It is an efficient task level fault tolerance technique for long running applications.

2) Replication: Various task replicas are run on different resources, for the execution to succeed till the entire replicated task is not crashed. It can be implemented using tools like HAProxy, Hadoop and Amazon Ec2 etc.

3) Job Migration: During failure of any task, it can be migrated to another machine. This technique can be implemented by using HAProxy.

4) SGuard: It is less disruptive [20] to normal stream processing and makes more resources available. It is based on rollback recovery and can be implemented in HADOOP, Amazon EC2.

5) Retry: It is the simplest task level technique that retries the failed task on the same cloud resource.

6) Task Resubmission: It is the most widely used fault tolerance technique in current scientific workflow systems. Whenever a failed task is detected, it is resubmitted either to the same or to a different resource at runtime.

7) User defined exception handling: In this user specifies the particular treatment of a task failure for workflows.

8) Rescue workflow: This technique allows the workflow to continue even if the task fails until it becomes impossible to move forward without catering the failed task.

B. Proactive Fault Tolerance

The principle of proactive fault tolerance policies is to avoid recovery from faults, errors and failures by predicting them and proactively replacing the suspected components from other working components. These techniques are based on these policies:

1) Software Rejuvenation: It is a technique that designs the system for periodic reboots. It restarts the system with clean state .

2) Proactive Fault Tolerance using self-healing: When multiple instances of an application are running on multiple virtual machines, it automatically handles failure of application instances.

3) Proactive Fault Tolerance using Pre-emptive Migration: Pre-emptive Migration relies on a feedback-loop control mechanism where application is constantly monitored and analyzed.

The fault tolerance techniques [12] can be measured by the parameters like, network load for delay and throughput, optimum resource utilization, response time and overload, job allocation, load balancing, scalability and security.

QoS in cloud is often characterized by using fault detection latency, replica launch latency and failure recovery latency, and other application-dependent metrics such as bandwidth, reliability, robustness, and loss rate etc.

III. Fault Tolerance Models

A failure represents the condition in which the system deviates from fulfilling its intended functionality or the expected behaviour. A failure happens due to an error that is, due to reaching an invalid system state. The hypothesized cause for an error is a fault which represents a fundamental impairment in the system. The notion of faults, errors and failures can be [4] represented using the following chain [S.2004, HH.1997]:

Fault → Error → Failure → Fault → Error → Failure

Several models that are [3] implemented based on previously defined fault tolerance techniques are as follows:

A. AFTRC

It is an Adaptive Fault Tolerance model in Real time Cloud Computing. In this proposed model system tolerates fault proactively and makes decisions on the basis of the reliability of the processing nodes.

B. LLFT

A propose model which contains a low latency fault tolerance (LLFT) middleware for providing fault tolerance for distributed applications deployed within the cloud computing environment. This middleware replicates application by the use of semi-active replication or semi-passive replication process to protect the application against various types of faults.

C. FTM

It is a model to overcome the limitation of existing methodologies and achieve the reliability and flexibility. An inventive is proposed perspective on creating and managing fault tolerance. By this particular methodology user can specify and apply the desire level of fault tolerance. FTM architecture can primarily be viewed as an assemblage of several web services components, each with a specific functionality.

D. FTWS

It is a Fault [19] Tolerant Work flow Scheduling algorithm for providing fault tolerance by using replication and resubmission of tasked based on based on the priority of the task. This model is based on the fact that workflow is a set of tasks processed in some order based on data and control dependency. Scheduling the workflow along with the task failure consideration in a cloud environment is very challenging. FTWS schedule and replicates the tasks to meet the deadline.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 6, Issue 3, March 2016)

E. Candy

Candy is a component based availability model. It is based on the high availability assurance of cloud service is one of the main characteristic of cloud service and also one of the main critical and challenging issue for cloud.

F. FT-Cloud

A component ranking based framework and its architecture for building cloud application. FT Cloud occupies the component invocation structure and frequency for identifying the component. Also, there is an algorithm to automatically govern fault tolerance safely.

IV. Fault Tolerant Strategies

There are many strategies in which three are exclusively included here.

A. Recovery Blocks (RB)

RB [3] is a means of structuring redundant program modules, where standby components will be invoked sequentially.

Failure probability f of a recovery block can be calculated by:

$$f = \pi^{n}_{i=1} f^{i} \qquad (1)$$

Where n is the number of redundant components and f_i is the failure probability of the i_{th} component.

B. N- Version Programming (NVP)

NVP [11] is multi-version programming where versions are independently generated. Failure probability f of a recovery block can be calculated by:

$$f = \sum^{n}_{i=n+1/2} F(i) \qquad (2)$$

Where n is the number of functionally equivalent components (n is odd) and F(i) is the probability that I alternative components from all the n components fail.

C. Parallel Strategy

Invokes all the n functional equivalent components in parallel and the first returned response will be employed as the final result. Failure probability f of a recovery block can be calculated by:

$$f = \pi^{n}_{i=1} f^{i} \qquad (3)$$

Where n is the number of redundant components and f_i is the failure probability of the i_{th} component.

D. ACO

Ant colony optimization algorithm (ACO) is a probabilistic technique and is a member of the ant colony algorithms family, in swarm intelligence methods, and it constitutes some metaheuristic optimizations. It can be reduced to finding good paths through graphs.

In general, the ant moves from state to state with probability Where is the amount of pheromone deposited for transition, $0 \leq \alpha$ is a parameter to control the influence of ,is the desirability of state transition *xy* (*a priori* knowledge, typically , where *d* is the distance) , and represent the attractiveness and trail level for the other possible state transitions.

E. BC Strategy

The Artificial Bee Colony (ABC) [11] is an optimization algorithm based on the intelligent foraging behaviour of honey bee swarm. The main steps of the algorithm are given below:

1) Initial Food:

Initial food sources are produced for all employed bees.

2) Repeat:

a) Each employed bee goes to a food source and then evaluates its nectar amount and dances in the hive.
b) Each onlooker watches the dance of employed bees and chooses one of their sources depending on the dances, and then goes to that source.

V. Challenges For Implementation Of Fault Tolerance In Cloud Computing

Providing fault tolerance [1] requires careful consideration and analysis because of their complexity, inter-dependability and the following reasons:

- There is a need to implement autonomic fault tolerance technique for multiple instances of an application running on several virtual machines [5].
- The new approach needs to be developed that integrate these fault tolerance techniques with existing workflow scheduling algorithms [6].
- Different technologies from competing vendors of cloud infrastructure need to be integrated for establishing a reliable system [7].
- Autonomic fault tolerance must react to synchronization among various clouds [7].
- A benchmark based method can be developed in cloud environment for evaluating the performances of fault tolerance component in comparison with similar ones [8].
- To ensure high reliability and availability multiple clouds computing providers with independent software stacks should be used [9] [10].

VI. Related Work

The landmark paper on fault-tolerant matrix operation was published in 1984 by Huang and Abraham [13], in which an Algorithm-Based Fault Tolerance (ABFT) method was proposed.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 6, Issue 3, March 2016)

ABFT uses matrix or vector level checksum in row and column to detect a faulty processor in multiple processor systems. The method can be used to detect and correct errors in matrix operations such as addition, multiplication, scalar product, and LU-decomposition performed in multiple processor systems which may have one failed processor. In our work, however, we focus on the problem of achieving a certain reliability with the minimum cost in potentially faulty clouds. In [14], Mei et al. investigated the problems of dynamic computing service registration, large data storage and access, adaptability, and quality discovery in cloud computing. A comprehensive [15] high-level approach to shading the implementation details of the fault tolerance techniques to application developers and users by means of a dedicated service layer has been proposed. In particular, the service layer allows the user to specify and apply the desired level of fault tolerance, and does not require knowledge about the fault tolerance techniques that are available in the envisioned Cloud and their implementations. The proposed algorithm in [16] works for reactive fault tolerance among the servers and reallocating the faulty servers task to the new server which has minimum load at the instant of the fault.

VII. Proposed Work

Executing sophisticated applications like video, image and storage processing in personal mobile devices remains challenging due to their limited resources (e.g. computation, memory, energy etc.). As a result many applications rely on offloading all or part of their works to remote servers such as clouds and mobile devices. In a dynamic network, e.g. mobile cloud for disaster respose or military operations, when selecting remote servers they are often inaccessible because of node failures, unstable links or node mobility which raises reliability issue. The natures of these operating environments are such that faults often develop during the course of regular action. A fault can cause the node to lose functionality, which in turn may lead to drop the overall performance of the system.

In our proposed work, we evaluate the system performance based on the following terms:

A. Efficiency

Ability of the system to best utilize the available resources.

B. Robustness

Ability of the system to identify and recover from faults, Formally the problem can be defined as follows:

Given: A set of mobile nodes in cloud environment R = {R1, R2, R3... Rn}.

A pre-defined set of tasks to be executed by the mobile nodes T = {T1, T2, T3, ..., Tm}, where each task T_j is executed by a separate mobile node R_i.

We assume that the task assignment is pre-defined by means of a vector < R_i, T_j >. An individual task Tj is executed by the specific mobile node Ri. Faults can occur naturally during task execution or can be artificially introduced into the system. Faults are broadly categorized into three types: known, which are faults the designer can anticipate based on experience, application type and operating environment unknown, which are faults not anticipated by the designer, but which can be diagnosed by the system based on experience and sparse information available during execution and undiagnosable, which are faults that cannot be classified autonomously and need specific techniques to handle. The number of faults in each category is represented as f_{known}, $f_{unknown}$, and $f_{undiagnosable}$. The mobile nodes have three functionally significant operating states: Normal state, in which a mobile node focuses all its system resources and operating time towards completing the assigned task. Fault state, in which a mobile node spends all available time and resources in attempting to identify the source of the encountered fault and Recovery state, in which a mobile node spends its resources and operating time in executing the recovery action for the diagnosed fault. Once assigned to a mobile node, a task can have two possible outcomes: task success or task failure. Task success is defined as the ability of the robot to successfully complete its assigned task in the presence of faults. A task failure is defined as the inability of the mobile node to complete its assigned task in the presence of faults. If a mobile node (R_i) fails to complete a task (T_j) , then based on the system design, the system can either assign task T_j to a different mobile node R_i, re-assign T_j to the task queue of R_j , remove task T_j from the system task list. Every task assignment <R_i,T_j>, is considered a task attempt and is evaluated separately towards overall system performance. Based on the importance of the task the designer builds a task-utility table, such as that shown in Table I, in which the summation of the terms (∑u and ∑c) are normalized between ranges of [0, 1].

VIII. Measuring System Performance

We first define the total number of faults for the i_{th} attempt of task T_j as the summation [18] of all encountered faults during the course of task execution. That is, $F^i_j = f^i$ known$_j$+ f^i unknown$_j$+ f^i undiagnosable$_j$. Successful completion of task Tj is measured by means of a success metric, A_j. An award is associated with every successfully completed task, given by the utility component uj.

$$A_j = u_j \qquad (1)$$

Then, the system level measure of success (A) is calculated as:

$$A = \sum_{J:Tj \in X} u_j \qquad (2)$$

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 6, Issue 3, March 2016)

Where X = {T_j | Task T_j € T was successfully completed}, Similarly, we associate a task failure metric, B^i_j ,for each unsuccessful attempt of task T_j by a mobile node. The punishment associated with a failed task attempt is given by the cost component for task failure c_j . On the other hand, as the performance is closely tied with the mobile node's ability to recover from faults, every failed task has a robustness component associated with it. The effect of the task failure metric towards performance is discounted by the extent of the robustness in the task, i.e., the higher the robustness, the lower is the value of the task failure. We define ρ^i_j as the measure of robustness for the i^{th} attempt of task Tj and is given by

$$\rho^i_j = (f^i \text{known}_j + f^i \text{unknown}_j) / F^i_j \quad (3)$$

Based on "(3)", the task failure metric for the i^{th} attempt of task Tj is:

$$B^i_j = c_j * (1 - \rho^i_j) \quad (4)$$

Grouping all failed attempts of a task T_j ,we get task failure metric B_j for a task T_j.

$$B_j = (c_j * q_j) * \sum_{i=1}^{q_j} (1 - \rho^i_j) \quad (5)$$

where q_j is total number of failed attempts of task T_j. The upper bound of q is application specific and needs to be determined by the designer before implementation.

Simplifying,

$$B_j = (c_j * q_j) * (q_j - \sum_{i=1}^{q_j} \rho^i_j) \quad (6)$$

Extending equation "(3)" across all task failures, gives:

$$B = \sum_{J:Tj \in X} (c_j * q_j) * (q_j - \sum_{i=1}^{q_j} \rho^i_j) \quad (7)$$

Where Y= {T_j | Task T_j € T failed}.

Now the measure of performance can be obtained by subtracting the cost associated with a task failure from the utility for successful task completion.

$$P = A - B \quad (8)$$

P provides the designer with a measure of the system's effective performance whose value will be in the range of [-∞,1].If P=1, then the system performs optimally and if P approaches -∞ then it indicates a total system failure.

IX. Measuring Fault-Tolerance

By combining individual task robustness measures, system robustness can be represented as

$$\rho_s = \sum_{J:Tj \in Y} \sum_{i=1}^{q_j} \rho^i_j \quad (1)$$

A high value of ρ_s indicates a highly robust system and a ρ_s value 0 indicates a system with no robustness to faults. In order to define the system efficiency metric (e), we need to measure the total time (t_j) spent by a mobile node on a successfully completed task, T_j . This is given by the summation of time spent in Normal (t_{Normal}), Fault (t_{Fault}) and Recovery ($t_{Recovery}$) states for that attempt, i.e.,

$$t_j = t_{Normal\ j} + t_{Fault\ j} + t_{Recovery\ j} \quad (2)$$

Then, we can define e as:

$$e = \sum_{j:Tj \in X} (t_{Normal\ j}) / t_j \quad (3)$$

Similar to the robustness measure, a more efficient system has a higher value of e and an inefficient system has e near 0. The influence of learning towards system performance can be measured as an empirical quantity.

Table I
UTILITY-COST TABLE

Task	Utility	Cost For Task Failure
t1	u1	c 1
t2	u 2	c2
t3	u3	c3
.....	.....	.....
t10	u10	c10

Consider a sample multi-node application comprised of 10 individual tasks to be completed by a team of 10 functionally similar mobile application.. We make the assumptions that the robots encounters one failure per task and the task/utility weights are evenly distributed. Then we define these measures as follows:

For all i, $u_i = c_i$

$$\sum_{i=1}^{10} u_i \sum_{i=1}^{10} c_i = 1 \quad (4)$$

The time spent by a mobile node in normal operation mode is assumed to be t secs. Also, as it takes a very small fraction of time to diagnose task failure from the time a fault is discovered, we assume this time to be negligible and ignore it.

The variations in the values can be evaluated by considering the following three cases:

A. Best cases, where the system encounters no failures.
B. Average case, where the system encounters at least one failure in half the number of executed task.
C. Worst-case, where there is at least one failure in all cases.

The Table I illustrate the values obtained for two different architectural implementation-one with no built-in fault tolerance (F1) and another with some redundancy-based fault-tolerance (F2).

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 6, Issue 3, March 2016)

When a fault is encountered during task execution in the first architecture, mobile nodes do not have the capability to recover and report a failed task. In case of the second architecture, if and when a failure occurs, the task is assumed to have failed and is reassigned to another team member for execution. The task reassignment continues until all mobile nodes in the team have had an opportunity to complete the task. We further make the assumption that at an average it takes n /2 attempts to successfully recover from an encountered fault. Finally, we assume there is 50% probability of the system successfully recovering from an encountered error.

X. Result Analysis

Firstly the performance of the system is measured in a system with faults. We inferred that on average, the performance of the system with zero fault-tolerance (F1) is better than that of the system with some fault-tolerance (F2) which is represented in Table II. However, with increasing number of faults in the system increases, F2 edges F1 in terms of performance. On the other hand, a system with a higher task completion rate will have a higher value for efficiency, which is reflected in Table II.

Table II
EVALUATION TABLE

System	Case	P	ρ	e
F1	Best case	1	0	10
	Average case	0.5	0	5
	Worst Case	-1	0	0
F2	Best case	1	0	10
	Average case	0.4	0	7.5
	Worst Case	-4.5	0	5

Secondly, robustness of the system is optimized using Genetic algorithm which is reflected in Fig1.

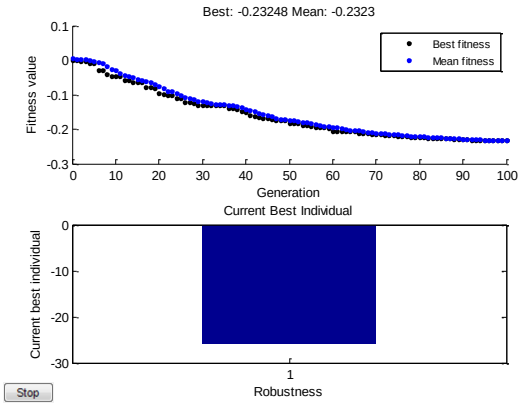

Fig 1

XI. Conclusion And Future Work

Fault tolerance is concerned with all the techniques which are necessary to enable a system to tolerate system faults remaining in the system after its development. This paper discussed the fault tolerance techniques covering its research challenges, used for implementing fault tolerance techniques in cloud computing. In this paper, we present an evaluation metric to measure the extent of fault-tolerance towards system improvement over a period of time .Specifically this research provides a qualitative measure for identifying system fault-tolerance in terms of efficiency, robustness. However we have not considered some parameters like availability, reliability and extent of fault management in dynamic environment for measuring the system performance which we will evaluate in our future research work.

REFERENCES

[1] Anju Bala, Inderveer Chana," Fault Tolerance- Challenges, Techniques and Implementation in Cloud Computing" IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 1, No 1, January 2012.

[2] Alain Tchana, Laurent Broto, Daniel Hagimont,"Fault Tolerant Approaches in Cloud Computing Infrastructures ,ICAS 2012 : The Eighth International Conference on Autonomic and Autonomous Systems.

[3] Virendra Singh Kushwahi, Sandip Kumar, Priyush Narwariya, "A Survey on various Fault Tolerant Approaches for cloud Environment during Load Balancing, International Journal of Computer Networking.

[4] Ravi Jhawar and Vincenzo Piuri, "Fault Tolerance and Resilience in Cloud Computing Environments", in Computer and Information Seurity handbook, 2nd Edition.

[5] Gang Chen, Hai Jin, Deqing Zou, Bing Bing Zhou,Weizhong Qiang, Gang Hu, "SHelp: Automatic Selfhealing for Multiple Application Instances in a Virtual Machine Environment", IEEE International Conference on Cluster Computing, 2010.

[6] [6] Yang Zhang1, Anirban Mandal2, Charles Koelbel1 and Keith Cooper, "Combined Fault Tolerance and Scheduling Techniques for Workflow Applications on Computational Grids" in 9th IEEE/ACM international symposium on clustering and grid, 2010.

[7] Imad M. Abbadi, "Self-Managed Services Conceptual Model in Trustworthy Clouds' Infrastructure", 2010.

[8] S. Hwang, C. Kesselman, "Grid Workflow: A Flexible Failure Handling Framework for the Grid",12th IEEE International Symposium on High Performance Distributed Computing (HPDC'03), Seattle, Washington,USA. IEEE CS Press, Los Alamitos, CA, USA, June 22 - 24, 2003.

[9] Michael Armbrust, Armando Fox, Rean Griffith, "Above the Clouds: A Berkeley View of Cloud Computing", Electrical Engineering and Computer Sciences University of California at Berkeley, 2009.

[10] Wenbing Zhao, P. M. Melliar-Smith and L. E. Moser, "Fault Tolerance Middleware for Cloud Computing", 2010 IEEE 3rd International Conference on Cloud Computing.

[11] http://haproxy.1wt.eu/download/1.3/doc/configuration.txt.

[12] Seyyed Mansur Hosseini and Mostafa Ghobaei Arani, "Fault Tolerance Techniques in Cloud Storage: A Survey", International Journal of Database Theory and Application Vol.8, No.4 (2015), pp.183-190.

[13] K.-H. Huang and J. A. Abraham, "Algorithm-based fault tolerance for matrix operations," IEEE Transactions on Computer, vol. 33, no. 6, pp. 518–528, 1984.

[14] L. Mei, W. Chan, and T. Tse, "A tale of clouds: Paradigm comparisons and some thoughts on research issues," Asia-Pacific Conference on Services Computing. 2006 IEEE, vol. 0, pp. 464–469, 2008.

[15] R. Jhawar , Univ. degli Studi di Milano, Crema, Italy , V. Piuri ,M. Santambrogio, "Fault Tolerance Management in Cloud Computing: A System-Level Perspective**",** IEEE System Journal, Volume 7 Issue 2,2013.

[16] Jasbir Kaur, Supriya Kinger , "Efficient Algorithm for Fault Tolerance in Cloud Computing",International Journal of Computer Science and Information Technologies, Vol. 5 (5) , 2014, 6278-6281.

[17] T.K. Singh, G.T. RaviTeja, P.S.Pappala," Fault Tolerance-Challenges, Techniques and Implementation in Cloud Computing" International Journal of Scientific and Research Publications, Volume 3, Issue 6, June 2013.

[18] Balajee Kannan and Lynne E. Parker," Fault-Tolerance Based Metrics for Evaluating System Performance in Multi-Robot Teams", Proc. of Performance Metrics for Intelligent Systems Workshop, 2006.

[19] Sweta Patel, Ajaya Kumar Singh," Fault Tolerance Mechanisms and its Implementation in Cloud Computing –A Review",IJARCSSE, Volume 3, Issue 12,December 2013.

[20] Sonal Rana ,Aditi Sharma,Amandeep Kaur," Fault Tolerance in Cloud Computing by Efficient Load Balancing Mechanism Based on Ant Colony", IJARCSSE,Volume 5, Issue 1,January 2015 .

[21] Vikas Kumar,Shiva Prakash," A Load Balancing Based Cloud Computing Techniques and Challenges", IJSRM,Volume,Issue 5, May 2014

[22] K.Ganga,,Dr S.Karthik,, A.Christopher Paul, "A Survey on Fault Tolerance in Work flow Management and Scheduling ",IJARCET, Volume 1, Issue 8, October 2012