Review on Fault Tolerance Techniques in Cloud Computing

Zeeshan Amin
Lovely Professional University, Phagwara, Punjab, India

Nisha Sethi
Lovely Professional University, Phagwara, Punjab, India

Harshpreet Singh
Lovely Professional University, Phagwara, Punjab, India

ABSTRACT
With the immense growth of internet and its users, Cloud computing, with its incredible possibilities in ease, Quality of service and on-interest administrations, has turned into a guaranteeing figuring stage for both business and non-business computation customers. It is an adoptable technology as it provides integration of software and resources which are dynamically scalable. The dynamic environment of cloud results in various unexpected faults and failures. The ability of a system to react gracefully to an unexpected equipment or programming malfunction is known as fault tolerance. In order to achieve robustness and dependability in cloud computing, failure should be assessed and handled effectively. Various fault detection methods and architectural models have been proposed to increase fault tolerance ability of cloud. The objective of this paper is to propose an algorithm using Artificial Neural Network for fault detection which will overcome the gaps of previously implemented algorithms and provide a fault tolerant model.

Keywords
Cloud Computing, Fault Tolerance, Failure Detector.

1. INTRODUCTION TO CLOUD COMPUTING
Cloud computing alludes to the delivery of computing resources over the Internet. As opposed to keeping information on our own hard drive or upgrading applications for our needs, we can utilize a service over the Internet, at an alternate area, to store our data and Use its applications. The thought of cloud computing is focused around a basic idea of reusability of IT abilities [1, 3].
Cloud computing is built upon virtualization, distributed computing, utility computing, and more recently networking, web and software services. Individuals and organizations use hardware and software managed by third parties at remote location. Online file storage, social networking sites, webmail, and online business applications are some common cloud services. User can use these services without knowing the underlying hardware and software details [2].
A real time system can utilize the immense computing capabilities and virtualized environment of cloud for the execution of tasks. On the other side, most of these are safety critical systems which require high reliability and high level of fault tolerance for their execution.

The objectives of paper are as follows:

i. Introduction to various fault tolerance and fault detection techniques in cloud computing.
ii. Review of various types of fault detectors used for fault detection are reviewed.
iii. A theoretical foundation of artificial neural networkbased approach for detecting the faults in cloud computing.

1.1 Cloud Components
Cloud computing is made up of several elements. Each element has a purpose which plays specific roles which can be classified as clients, Distributed servers, data centers.

- **Clients:** These are typically the computers which are used by the end users i.e. the devices which can be used by the end user to manage the information on cloud (laptops, mobile phones, PADs etc.)
- **Data center:** These are collection of servers where the service is hosted. In order to create number of virtual server on one physical server in data center, virtualization is used.
- **Distributed servers:** These are servers which are located in different geographical place. It provides better accessibility, security to the user.

1.2 Characteristics of Cloud Computing
There are ten characteristics of cloud computing in their sum up: device and location independence, scalability, on-demand services, guaranteed Quality of Service (QOS), pricing, virtualization, multi-tenancy, security and fault tolerant [4].

- **Scalability and on-demand services:** users are given on-demand resources and services over cloud. Moreover the resources provided are scalable over several data centers
- **User-centric interface:**cloud interfaces are not dependent on location of user. They can be accessed by well-established interfaces such as web services and internet browsers.
- **Guaranteed Quality of Service (QOS):** Cloud computing assures Quality of service for users by guaranteed performance, bandwidth and memory capacity.
- **Autonomous system:**users can reconfigure and combine software and information according to their requirements.
- **Cost**: No capital expenditure or any up-form investment is required in cloud. Payment for services is made on the basis of need.

- **Virtualization:**Utilization of resources is increased by sharing the server and storage devices.
- **Multi-tenancy: S**haring of resource and cost among large number of users increase efficiency and allows for
  Centralization and peek lock capacity.
- **Loose coupling:**The resources are loosely coupled as one resource functionality hardly affects the functioning of another resource.
- **Reliable Delivery**: TCP/IP is used for delivery of information between resources. Private network protocols are used within the cloud infrastructure but most of the user are connected using HTTP protocol.
- **High Security:** This is maintained on the above discussed characteristics. Loose coupling enables the jobs to execute run well,even if part of cloud is destroyed. Virtualization and abstraction of cloud provider avoid exposing the detailing of implementation.

1.3 Cloud Computing Benefits

Cloud computing reduces the response time and running time of job, also minimizes the risk in deploying application, lowered cost of deployment, and decreasing the effort and increasing innovation [5].

- **Increased Throughput:** Cloudmakes use of thousands of servers to finish an assignment in reduced time unit verses the time required by a solitary server.
- **Minimize infrastructure risk**: Cloud can be used by organizations to reduce the load of purchasing physical servers. The issues of highinvestment and deployment of servers depending upon the workload can be resolved considering investment on infrastructure for those application's whose attainment is short-lived.
- **Lower cost of entry:** Various characteristics outlined in previous section of cloud reduces the cost for organizations to enter new markets:
  - The capital investment is reduced to zero by renting the infrastructure instead of purchasing it and hence controls the cost.
  - The rapid application development helps to reduce the time to market, possibly giving organizations an edge against the competition.
- **Focus on innovation:**Organization relived with the issue to infrastructure deployment can focus in innovating items.

1.4 Cloud Computing Model

Services offered by the cloud providers can be grouped into three categories [6]:

- **Software as a Service (SaaS):** In this model, a complete application is provided on demand to the user.
  Multiple end users are serviced while at the back end a single instance of service is executed. Customers need not to go for any upfront investments, since just a single application is to be facilitated & kept up. Google, Salesforce, Microsoft, Zoho etc are the providers of Saas.
- **Platform as a Service (Paas):**In this model, software or development environment is offered as a service. The customer is given with the option to construct his own particular applications, which run on the suppliers' base. A predefined combination of OS and application servers is provided to the user.Google's App Engine, Force.com are providing a platform to users.
- **Infrastructure as a Service (Iaas):** Standardized services that are provided are Fundamental storage and computing capabilities. Various resources are made available and shared among users in order to manage workload. The customer has to deploy his own software on the infrastructure. Amazon, GoGrid, 3 Tera are examples of Iaas.

1.5 Types of Cloud

On the bases of access to clouds, they can be classified into following types [6]:

- **Public Cloud:** Usersconnected to internet and having access to the cloud space can use public cloud. It refers to availability of computing resources to anyone on "Pay As You Go Basis". Public clouds are owned and operated by third parties; they deliver superior economies of scale to customers. All customers share the same infrastructure pool with limited configuration, security protections, and availability variances.
- **Private Cloud:** A private cloud in an organization is specific and limited access to a particular group. It can be referred as computing services delivered exclusively for the use of a particular organization or a group. It utilizes the same architecture for scalability and availability as the public cloud but it is limited to a single organization. Two major concerns on data security and control are addressed which are not there in public cloud.
- **Hybrid Cloud**: A combination of public and private cloud is named as hybrid cloud. With a Hybrid Cloud, service providers can expand the adaptability of computing byutilizingother Cloud Providers in full or partial manner. The Hybrid cloud environment is capable for providing on-demand, externally provisioned scale with the capacity to enlarge a private cloud to deal with any sudden surges in workload.
- **Community Cloud:** The organization with common prerequisites share the cloud functionality making it a hybrid cloud.It reduces the capital consumption by imparting the cost among the associations. The operation may be in-house or with an outsider on the premises.

2. FAULT TOLERANCE IN CLOUD COMPUTING

Fault Tolerance alludes to a methodology to system design that permits a system to keep performing actually when one of its parts falls flat or it can be defined as capacity of a system to react nimbly to an unexpected equipment or

programming break down. If not fully operational, fault tolerance solutions may allow a system to continue operating at reduced capacity rather than shutting down completely following a failure [7].

2.1 Metrics for Fault Tolerance in Cloud Computing

The existing fault tolerance technique in cloud computing considers various parameters: throughput,response-time, scalability, performance, availability, usability,reliability, security and associated over-head. [8]

- **Throughput**–It defines the number of tasks whose execution has been completed. Throughput of a system should be high.
- **Response Time-** Time taken by an algorithm to respond and its value should be made minimized.
- **Scalability**– Number of nodes in a system does not affect the fault tolerance capacity of the algorithm.
- **Performance**– This parameter checks the effectiveness of the system. Performance of the system has to be enhanced at a sensible cost e.g. by allowing acceptable delays the response time can be reduced.
- **Availability**: Availability of a system is directly proportional to its reliability. It is the possibility that an item is functioning at a given instance of time under defined circumstances.
- **Usability**: The extent to which a product can be used by a user to achieve goals with effectiveness, efficiency, and satisfaction.
- **Reliability**: This aspect aims to give correct or acceptable result within a time bounded environment.
- **Overhead Associated**: It is the overhead associated while implementing an algorithm. Overheads can be imposed because of task movements, inter process or inter-processor communication. For the efficiency of fault tolerance technique the overheads should be minimized.
- **Cost effectiveness**: Here the cost is only defined as a monitorial cost.

2.2 Fault Taxonomy

Cloud is prone to faults and they can be of different types. Various fault tolerance techniques can be used at either task level or workflow level to resolve the faults [9].

i) Reactive fault tolerance

Reactive fault tolerance techniques are used to reduce the impact of failures on a system when the failures have actually occurred. Techniques based on this policy are checkpoint/Restart and retry and so on.

- **Check pointing/Restart**- The failed task is restarted from the recent checkpoint rather than from the beginning. It is an efficient technique for large applications.
- **Replication:** In order to make the execution succeed, various replicas of task are run on different resourcesuntil the whole replicated task is not crashed. HAProxy, Haddop and AmazonEc2 are used for implementing replication.
- **Job migration:** On the occurrence of failure, the job is migrated to a new machine. HAProxy can be used for migrating the jobs to other machines.

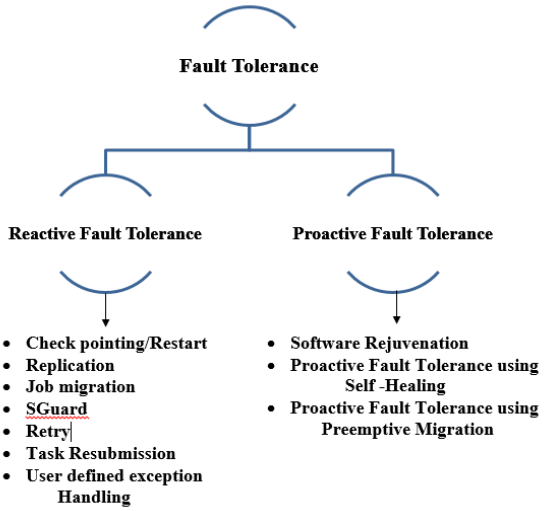

Fig 1 Fault Tolerance Techniques

- **SGuard:**It isbased on rollback recoveryand can be executed in HADOOP, Amazon Ec2.
- **Retry:** This task level technique is simplest among all. The user resubmits the task on the same cloud resource.
- **Task Resubmission:**The failed task is submitted again either to the same machine on which it was operating or to some other machine.
- **User defined exception handling:** Here the user defines the specific action of a task failure for workflows.
- **Rescue workflow**: It allows the system to keep functioningafter failure of any task until it will not be able to proceed without rectifying the fault.

ii) Proactive Fault Tolerance:

Proactive fault tolerance predicts the faults proactively and replace the suspected components by other working components thus avoiding recovery from faults and errors. Preemptive migration, software rejuvenation etc. follow this policy.

- **Software Rejuvenation-**the system is planned for periodic reboots and every time the system starts with a new state.
- **Proactive Fault Tolerance using self-healing:** Failure of an instance of an application running on multiple virtual machines is controlled automatically.
- **Proactive Fault Tolerance using Preemptive Migration:**In this technique an application is constantly observed and analyzed. Preemptive migration of a task depends upon feed-back-loop control mechanism.

2.3 Failure Detector

A failure detector is an application or a system that is used to detect node failures or crashes. Failure detector can be classified as reliable or unreliable on the basis of result it produces. If the output of failure detector is always accurate it is called as reliable failure detector. An unreliable failure detector is one that provides information that is not necessarily accurate and it may take very long time for detection of faulty process and produce false results by suspecting the processes that have not crashed. Most of the failure detectors fall in this category.

Correctness properties of failure detectors:

- **Completeness:** when a process fails that process is eventually detected by at least one other non-faulty process.Completeness describes the capability of failure detector of suspecting every failed process permanently.
- **Accuracy:** There are no mistaken failure detections i.e. when a process is detected as failed, it has actually failed. Less number of false positives result in high accuracy.
  It is impossible to build a failure detector over a realistic network that is 100% accurate and complete.
  Real life failure detectors guarantee 100% completeness but the accuracy is either partial or probabilistic. There is a trade-off between completeness and accuracy
- **Speed:** Time for the detection of failure should be as less as possible. In other words, time between occurrence of a failure and its prediction must be small.
- **Scale:**There should be low and equally distributed load on each process in a group and also low overall network load.
  A failure detector should guarantee all of these properties in spite of the fact that there can be arbitrary simultaneous multiple process failures. In addition to these properties Chandra and Toug[15] proposed a set of metrics that specify Quality of service (QOS) of a failure detector.
  - **Detection time (TD):** Time that elapses from crashing of a process p to the time when another process q starts suspecting process p permanently.
  - **Mistake recurrence time (T_{MR}):** Time between two successive mistakes.
  - **Mistake Duration (T_M):** Time taken by a failure detector to correct the mistake.

Failure detectors that adapt themselves to the changing network conditions and application requirements are named as adaptive failure Detectors [11]. Most adaptive failures are based on heartbeat protocol where previous information is used for the prediction of arrival time of next heartbeat.

2.4 Heartbeat Strategy for failure detection

Heartbeat isa widely implemented strategy for failure detectors. After a fixed interval of time every process p send "I am alive" message to a process q. q waits for the message from p till the expiration of timeout from p and if the message is not received it adds p to list of suspected processes. If q later receives "I am alive" message from p, it will remove the process p from list of suspected processes.

Chandra and Toug [15] proposed an improvement of this classic heartbeat implementation. In the proposed algorithm, the process q (monitoring process) uses a sequence of fixed time points $\mathcal{T}_1$, $\mathcal{T}_2$, $\mathcal{T}_3$…called freshness points in order to determine whether to suspect the process p. The freshness point $\mathcal{T}_i$is an estimation of arrival time of heartbeat from p.

The advantage of this algorithm is that detection time is independent from the last heartbeat message, thus increasing accuracy of the failure detector as it avoids premature timeout.

2.5 Existing Strategies using Heartbeat

Chen FD [12]:

To figure out the estimation of the arrival time of the next heartbeat, Chen FD uses arrival times sampled in the recent past [15]. The expected time is set according to this estimation along with a safety margin and the value is recomputed for every interval.

Bertier FD[13, 14]:

Bertier introduced a failure detector principally intended for LAN environments. Their proposed algorithm uses the same mechanism as Chen for estimating expected arrival times, but a dynamic way of computing freshness points based on Jacobson's estimation [17, 18]. Bertier FD adapts the safety margin every time it receives a message. The adaptation of the margin α is based on the variable *error* in the last estimation.

The φ FD [15]:

Instead of providing information having a conventional binary nature i.e. true or suspect, the φ-FD gives a suspicion level on a continuous scale which makes it different from Chen and Bertier- FD[11]. InφFD, the suspicion level is given by a value called φ, expressed on a scale that is dynamically adjusted to reflect current network conditions.

3. RELATED WORK

WENBING ZHAO et al. (2010) proposed Low Latency Fault Tolerance (LLFT) Model that utilizes leader/follower replication approach and provides fault tolerance for distributed applications deployed within a cloud computing environment. The novel commitments of the LLFT middleware incorporate the low Latency Messaging Protocol, the leader-determined membership protocol and the virtual determinizer Framework.

DAWEI SUNet al. (2013) put forward a dynamic adaptive fault tolerance strategy (DAFT) that is focused around the standards and semantics of cloud fault tolerance. An analysis on relationship between different failure rates and two different fault tolerance techniques, check-pointing and replication has been carried out. A dynamic adaptive model has been built by combining the two fault tolerance models which helps to increase the serviceability.

ANJU BALA et al. (2014) put forward an idea of designing an intelligent task failure detection models for facilitating proactive fault tolerance by predicting task failures for

scientific workflow applications. The working of model is distributed in two modules. In first module task failures are predicted with machine learning approaches and in second module the actual failures are located after executing workflow execution in cloud test-bed. Machine learning approaches such as naïve Bayes, ANN, logistic regression and random forest are implemented to predict the task failures intelligently from the dataset of scientific workflows.

HWAMIN LEE et al. (2009) proposed a fault tolerant and recovery system called FRAS system (Fault Tolerant and recovery Agent System). This is an agent based system consisting of four types of agents. Recovery agent performs roll back recovery after occurrence of failure. Information agent hypothesis domain knowledge and information during a failure free operation. Facilitator controls the communication between agents and garbage collection agent performs garbage collection of data.Agent recovery algorithm is proposed to maintain a consistent state of a system and prevent domino effect.

NAIXUE XIONG et al. (2007) Given that networks are dynamic and unexpected, Naixue-Xiong, investigates Failure detector properties with connection to real and programmed fault-tolerant cloud based network systems, in order to discover a general non-manual investigation strategy to self-tune corresponding parameters to fulfill user requirements. Based on this general self-tuning method, they propose a dynamic and programmed Self-tuning Failure Detector scheme, called SFD, as an improvement over existing schemes.

ANJALI MESHRAM et al. (2013) proposed fault tolerance model for cloud (FTMC). This model accesses the reliability of computing nodes and choses the node for the computation on the basis of reliability. The node can be removed if it does not perform well.

RAVI JAWAHAR et al. (2012): provided a new dimension for applications deployed in a cloud computing infrastructure which can obtain required fault tolerance properties from a third party. The model straightforwardly work fault tolerance solution to user's applications by combining selective fault tolerance mechanisms and discovers the properties of a fault tolerance solution by method of runtime monitoring.

SAGAR C JOSHI et al (2014) proposed a fault tolerance mechanism to handle server failures by migrating the virtual machines hosted on the failed server to a new location. Virtualization has been applied for data centers giving rise to the concept of virtual Data Centers (VDC) which have virtual Machine (VM) as the basic unit of allocation. Using appropriate resource allocation algorithms, multiple VDCs can be hosted on a physical data center.

SHIVAM NAGPAL et al (2013) proposed a fault tolerant model that takes decisions. Reliability of a node is estimated on the basis of 2 parameters; accuracy and time. If any of the nodes does not achieve the level then backward recovery is performed by the system. This model focuses on adaptive behavior of processing nodes and the nodes are removed or added on the basis of reliability.

SHUN-SHENG et al (2010) proposed Dual Agreement Protocol of Cloud Computing (DAPCC), keeping in consideration the scalable and virtual nature of cloud. DAPCC is proposed to tackle the agreement problem caused by faulty nodes which send wrong messages, it tells how the system achieves agreement in a cloud computing environment.

HIEP NGUYEN (2013) proposes that one of the biggest challenges for diagnosing an abnormal distributed application is to pinpoint the faulty components. Black-Box online fault localization system called F-chain has been presented that can pinpoint faulty components immediately after a performance anomaly is detected. F-chain is presented as: a practical online fault localization system for large scale Iaas clouds. This system does not depend upon prior knowledge i.e. previously seen and unseen anomalies, and is practical for Iaas clouds. To achieve higher pinpointing accuracy, an integrated fault localization scheme has been introduced that consider both fault propagation patterns and inter component dependencies.

FABIO LIMA et al (2004) proposed adaptive failure detectors that are adjustable to the changing communication loads and use artificial neural networks for predicting the arrival time of next heartbeat from a virtual machine.

4. SCOPE OF STUDY

As per as the research gaps analyzed there is a potential need for implementing autonomic fault tolerance by using different parameters in cloud environment. During the literature review the various challenges faced by academicians in incorporating fault tolerance in cloud computing is as follows:

- The heterogeneity of the cloud is the biggest hindrance to localize the faults. There is a need to implement efficient techniques for locating the faults.
- There are more chances of errors because processing is done on remote computers.
- Failures occurring in the data centers are not in the scope of the user's organization necessitating the implementation of an autonomous fault tolerance technique for applications computing on cloud environment.
- It is difficult to interpret the changing system state because cloud environment are dynamically scalable, unexpected and often virtualized resources are provided as a service.
- Limited information is provided to the users because of high system complexity, so it is difficult to design an optimal fault tolerance solution.
- Fault Prediction and Monitoring framework needs to be developed for real time applications that execute in cloud environment.

5. METHODOLOGY

In order to achieve the objective "study and analyze various fault tolerance and fault detection techniques in cloud computing" a comprehensive literature survey was carried out for cloud computing and various fault detection and fault tolerance techniques implemented in cloud

computing. An extensive literature review was carried out for various models of artificial neural networkswhich can be used for fault detection.

Our proposed failure detector is based on Heartbeat strategy which uses Artificial Neural Network for the estimation of expected arrival time from a virtual machine.

To implement the proposed algorithm using Cloudsim the work is encompassed as:

i. The monitoring process q uses an estimated value (TO) which conveys q how much time it has to wait for the next heartbeat message from a process p.
ii. If after TO, q does not receive the heartbeat message from p, it will start suspecting p.
iii. TO is allowed to change over time to make it adaptive with actual communication loads.
iv. Time interval (TO) comprise of two values: the estimated time for the arrival of the next heartbeat message (ET) and the safety margin (α). The safety margin computed by ANN will help the detector to avoid false detections.

$$TO=ET+\alpha$$

6. EXPECTED OUTCOMES

Expected outcomes of the algorithm will be:

- Pro-active fault tolerance mechanism designed for dynamic clouds using Artificial Neural Network for fault detection can prove more beneficial than traditional models.
- The algorithm will provide detection time that is independent from the last heartbeat message, thus making the failure detector adaptive and increasing its accuracy.

7. CONCLUSIONS

Cloud environment is dynamic which leads to unexpected system behavior resulting in faults and failures. In order to improve reliability and achieve robustness in cloud computing, failures should be assessed and handled effectively. Fault detection is one of the biggest challenges in making a system fault tolerant.

This thesis proposes the use of artificial Neural Networks for detecting the faults in cloud environment. The faults are first detected and then suitable fault tolerance technique (pre-emptive migration/ check-pointing) is applied to make the system fault tolerant. The faults will be handled proactively and this will help to resolve the problems associated with fault tolerance techniques.

8. ACKNOWLEDGEMENTS

The author expresses his gratitude to Mr. HARSHPREET SINGH Assistant Prof., School of Computer Science & Engineering, LPU, India for his generous guidance, continuous encouragement and estimated supervision.

9. REFERENCES

[1] Foster, I., Zhao, Y., Raicu, I., & Lu, S. (2008, November). Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop, 2008. GCE'08* (pp. 1-10). Ieee.

[2] Mell, P., & Grance, T. (2009). The NIST definition of cloud computing. *National Institute of Standards and Technology*, *53*(6), 50.

[3] Plummer, D. C., Cearley, D. W., & Smith, D. M. (2008). Cloud computing confusion leads to opportunity. *Gartner Report*.

[4] Gong, C., Liu, J., Zhang, Q., Chen, H., & Gong, Z. (2010, September). The characteristics of cloud computing. In *Parallel Processing Workshops (ICPPW), 2010 39th International Conference on* (pp. 275-279). IEEE.

[5] Carolan, S. J. (2009). Introduction to cloud computing. *Architecture. White Paper, 1st edn. Sun Microsystems (June 2009)*.

[6] Furht, B. (2010). Cloud computing fundamentals. In *Handbook of cloud computing* (pp. 3-19). Springer US.

[7] Kaushal, V., & Bala, A. (2011). Autonomic fault tolerance using haproxy in cloud environment. *Int. J. of Advanced Engineering Sciences and Technologies*, *7*(2), 54-59.

[8] Patra, P. K., Singh, H., & Singh, G. (2013). Fault Tolerance Techniques and Comparative Implementation in Cloud Computing. *International Journal of Computer Applications*, *64*(14).

[9] Bala, A., & Chana, I. (2012). Fault Tolerance-Challenges, Techniques and Implementation in Cloud Computing. International Journal of Computer Science Issues (IJCSI), 9(1).

[10] Tchana, A., Broto, L., & Hagimont, D. (2012, March). Fault Tolerant Approaches in Cloud Computing Infrastructures. In *ICAS 2012, The Eighth International Conference on Autonomic and Autonomous Systems* (pp. 42-48).

[11] Hayashibara, N., Defago, X., Yared, R., & Katayama, T. (2004, October). The φ accrual failure detector. In Reliable Distributed Systems, 2004. Proceedings of the 23rd IEEE International Symposium on (pp. 66-78). IEEE.

[12] Gupta, I., Chandra, T. D., & Goldszmidt, G. S. (2001, August). On scalable and efficient distributed failure detectors. In *Proceedings of the twentieth annual ACM symposium on Principles of distributed computing* (pp. 170-179). ACM

[13] Maier, G., Sommer, R., Dreger, H., Feldmann, A., Paxson, V., & Schneider, F. (2008, August). Enriching network security analysis with time travel. In *ACM SIGCOMM Computer Communication Review* (Vol. 38, No. 4, pp. 183-194). ACM.

[14] Bahl, P., Chandra, R., Greenberg, A., Kandula, S., Maltz, D. A., & Zhang, M. (2007, August). Towards highly reliable enterprise network services via

inference of multi-level dependencies. In *ACM SIGCOMM Computer Communication Review* (Vol. 37, No. 4, pp. 13-24). ACM.

[15] Chen, W., Toueg, S., & Aguilera, M. K. (2002). On the quality of service of failure detectors. *Computers, IEEE Transactions on*, *51*(5), 561-580.

[16] Bertier, M., Marin, O., & Sens, P. (2002). Implementation and performance evaluation of an adaptable failure detector. In *Dependable Systems and Networks, 2002. DSN 2002. Proceedings. International Conference on* (pp. 354-363). IEEE.

[17] Bertier, M., Marin, O., & Sens, P. (2003, June). Performance analysis of a hierarchical failure detector. In *2013 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)* (pp. 635-635). IEEE Computer Society.

[18] Défago, X., Urbán, P., Hayashibara, N., & Katayama, T. (2005, March). Definition and specification of accrual failure detectors. In *Dependable Systems and Networks, 2005. DSN 2005. Proceedings. International Conference on* (pp. 206-215). IEEE.

[19] A Vouk, M. (2008). Cloud computing–issues, research and implementations.*CIT. Journal of Computing and Information Technology*, *16*(4), 235-246.

[20] Jhawar, R., Piuri, V., & Santambrogio, M. (2013). Fault tolerance management in cloud computing: A system-level perspective. *Systems Journal, IEEE*, *7*(2), 288-297.

[21] Huth, A., & Cebula, J. (2011). The Basics of Cloud Computing. *United States Computer*.

[22] Brian, O., Brunschwiler, T., Dill, H., Christ, H., Falsafi, B., Fischer, M., ... & Zollinger, M. (2012). Cloud Computing. *White Paper SATW*.

[23] Youssef, A. E. (2012). Exploring Cloud Computing Services and Applications.*Journal of Emerging Trends in Computing and Information Sciences*, *3*(6), 838-847.

[24] Xiong, N., Vasilakos, A. V., Yang, Y. R., Qiao, C., & Andy, Y. P. (2012). A class of practical self-tuning failure detection schemes for cloud communication networks. *IEEE/ACM Transactions on Networking (ToN), submitted*.

[25] Deng, J., Huang, S. H., Han, Y. S., & Deng, J. H. (2010, December). Fault-tolerant and reliable computation in cloud computing. In GLOBECOM Workshops (GC Wkshps), 2010 IEEE (pp. 1601-1605). IEEE.

[26] Zhao, W., Melliar-Smith, P. M., & Moser, L. E. (2010, July). Fault tolerance middleware for cloud computing. In Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on (pp. 67-74). IEEE.

[27] de Araújo Macêdo, R., & e Lima, F. R. L. (2004). Improving the quality of service of failure detectors with SNMP and artificial neural networks. In Anais do 22o. Simpósio Brasileiro de Redes de Computadores (pp. 583-586)

[28] Sun, D. W., Chang, G. R., Gao, S., Jin, L. Z., & Wang, X. W. (2012). Modeling a dynamic data replication strategy to increase system availability in cloud computing environments. *Journal of computer science and technology*, *27*(2), 256-272.

[29] Meshram, A. D., Sambare, A. S., & Zade, S. D. (2013). Fault Tolerance Model for Reliable Cloud Computing.

[30] Nguyen, H., Shen, Z., Tan, Y., & Gu, X. (2013, July). FChain: Toward black-box online fault localization for cloud systems. In *Distributed Computing Systems (ICDCS), 2013 IEEE 33rd International Conference on* (pp. 21-30). IEEE.

[31] Joshi, S. C., & Sivalingam, K. M. (2014). Fault tolerance mechanisms for virtual data center architectures. *Photonic Network Communications*, *28*(2), 154-164.

[32] Wang, S. S., Yan, K. Q., & Wang, S. C. (2011). Achieving efficient agreement within a dual-failure cloud-computing environment. *Expert Systems with Applications*, *38*(1), 906-915.

[33] Malik, S., & Huet, F. (2011, July). Adaptive Fault Tolerance in Real Time Cloud Computing. In *Services (SERVICES), 2011 IEEE World Congress on* (pp. 280-287). IEEE.

[34] Bala, A., & Chana, I. (2014). Intelligent failure prediction models for scientific workflows. *Expert Systems with Applications*.

[35] Chandra, T. D., & Toueg, S. (1996). Unreliable failure detectors for reliable distributed systems. *Journal of the ACM (JACM)*, *43*(2), 225-267