Fault Tolerance using Adaptive Checkpoint in Cloud–An Approach

J. M. Nandhini
Research Scholar, Anna University
Assistant Professor, Sri Sai Ram Institute of Technology, Chennai

T. Gnanasekaran, PhD
Professor, Department of IT
RMK Institutions, Chennai

ABSTRACT
Cloud computing provides platform for improving the flexibility in designing applications through exploiting the different layers of virtualization. Cloud computing is a new technology used for large scale enterprise applications. Cloud computing provides platform for improving the flexibility in designing applications through exploiting the different layers of virtualization The requirements of the business processes are met in the cloud computing. Cloud computing provides very high scalability, reconfigurable resources and higher availability of the resources. A robust Fault Tolerance strategy is a very critical component of cloud computing to meet the Service Level Objectives in cloud. High level of cloud serviceability is achievable through fault tolerance. The widely used strategies of fault tolerance are Checkpointing and Replication. In this paper an overview has been provided on various techniques of fault tolerance, dimensional view Checkpoint classification and a dynamically adaptive checkpointing model has beed proposed.

Keywords
Cloud computing, Fault Tolerance, Checkpoint , Virtual Machine

1. INTRODUCTION
Cloud computing is a computing paradigm, which has a large pool of systems that refers to logical computational resources accessible via a computer network. Cloud computing relies on sharing of various resources such as networks, servers, storage, applications, and services to achieve coherence, scalability, economies of scale, and maximizes the effectiveness and utilization of the resources. According to NIST Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction.

Fault-tolerance includes all the techniques necessary for robustness and dependability for cloud resources. The fault tolerance helps in addressing the QOS requirements in terms of process failure, processor failure and network failures. In the absence of fault tolerance, the system encounters problems such as job/resource failure, violation of deadlines and Service Level Agreement (SLA) that leads to degraded QOS. System Dependability can be measured to signify the level of fault tolerance. Reliability and availability of the cloud resources are the metrics to quantify dependability. Reliability symbolizes the capability of a system to perform, on demand, its service correctly. Availability denotes that the system is up to perform this service when it is asked to do so. Fault Tolerance in cloud enhances the performance standards, reduces cost and speeds up the failure recovery.

The rest of the paper is organized as follows. Section II highlights the significance of Fault Tolerance, Section III summarizes Fault tolerance Techniques, Section IV describes the dimensional view of checkpointing, Section V carries the literature survey and Section VI consists of the proposed checkpoint model.

2. IMPORTANCE OF FAULT TOLERANCE
Fault tolerance [1] is the ability to preserve the delivery of expected services despite the presence of fault that caused errors within the system itself. The objective is to avoid failures even in the presence of faults. In a very heterogeneous computing environment, fault tolerance is critical to ensure reliable performance. Errors are detected and corrected but permanent faults are located and removed while the system continues to deliver acceptable services.

The Service Level Objectives (SLOs) in clouds has the fault tolerant service as an important component. A fault tolerant strategy is required to attain a high level of SLO metrics [3]. Achieving reliability is important in Cloud, as most of the systems are safety critical, this mandates the need for fault tolerance.

Virtual Machines modelled on the cloud, groups and configures various heterogeneous resources. This provides greater flexibility in meeting the on-demand user request as and when it is raised [4]. VM are prone to failure due to their heterogeneity and longer usage. Failure in VM impacts badly on scalability, performance, profit and the user satisfaction. To avoid such impacts, Fault tolerance is a method to ensure uninterrupted performance of the system, even during faults [5]**.** Fault tolerance is a challenging research area in cloud computing [6]. Large and complex infrastructure necessitates a robust fault tolerance [2].

3. FAULT TOLERANCE TECHNIQUES
Based on work flow and task flow, Fault tolerance in cloud computing can be classified into two categories. The Fault Tolerance techniques are shown in Figure 1.

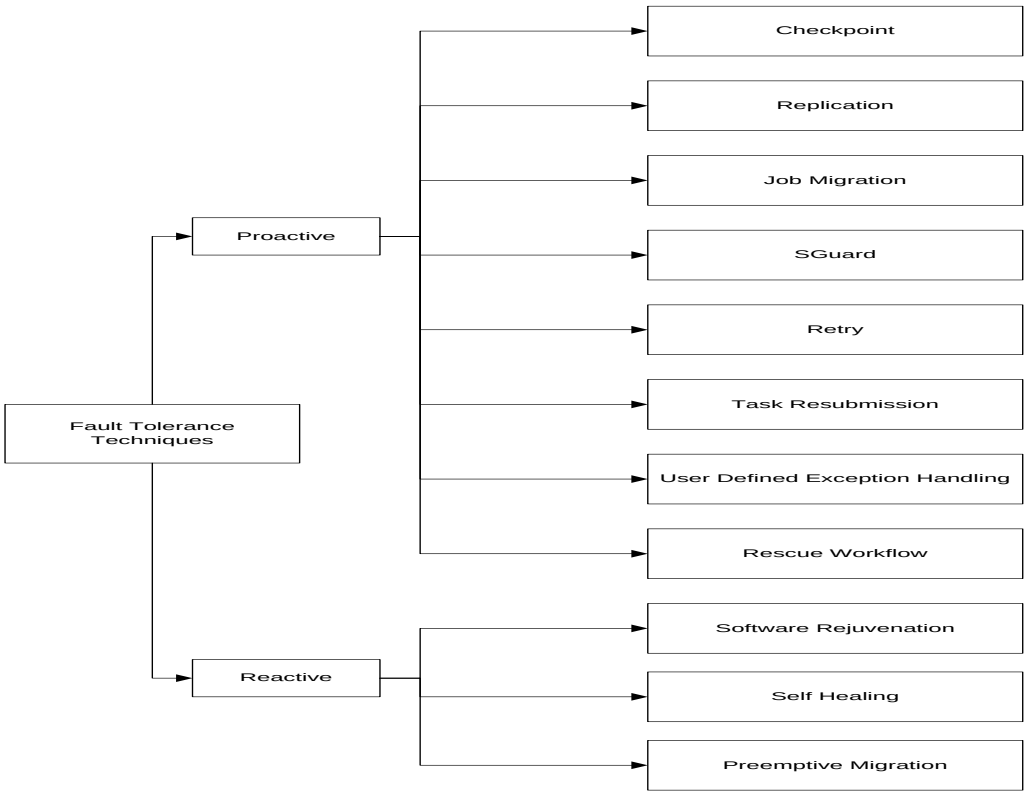

Fig 1: Fault Tolerance Techniques

A. Reactive fault tolerance

Reactive fault tolerance eliminates the faults after it has occurred. It minimizes the consequences of the failures on the system, when the failure occurs. This can be implemented by the following methods**.** [7].

- Checkpoint: In checkpoint method, the system can restart from the latest checkpoint by minimizing the number of re-computations involved from the previous failure.
- Replication: In replication method, the high prioritized data are replicated at multiple appropriate locations so that it can ease the user to access it from the close proximity of the fault free site.
- Job Migration: A resource failure or machine failure migrates the job to another VM machine where it resumes its execution. HAProxy can be used for implementing this method**.**
- SGuard: This method is based on rollback recovery. HADOOP and Amazon Ec2 uses this approach.
- Retry: This is simplest technique. It entails resubmitting the task on the same cloud resource**.**
- Task Resubmission: This approach is used when failed task is submitted to the same VM or to another VMduring runtime
- User defined exception handling: A workflow is predefined for execution in the event of task failure.
- Rescue workflow: In spite of the failure the system continues in its execution until it cannot proceed further without rectifying the fault.

B. Proactive Fault Tolerance

Proactive fault tolerance calculates the possible occurrences of faults in advance and averts the failures by substituting with working components. This can be implemented by following techniques

- Software Rejuvenation- It refers to the method of restarting the application as a clean state. The rejuvenation interval can also be defined periodically at various intervals where the application can be restarted as a clean interval state.
- Proactive Fault Tolerance using self-healing: This technique is widely used in the cases where many instances of a single application are running at different VMs. If a fault happens in any of the systems, self-healing automatically handles the failure in different VMs.
- Proactive Fault Tolerance using Preemptive Migration: Preemptive Migration of an application to a different node occurs when the current VM is about to fail.

4. DIMENSIONAL VIEW - CLASSIFICATION OF CHECKPOINT

The process of taking a snapshot of the current state of the running application on to a stable storage is called checkpoint. This is the most commonly used method of fault tolerance. When a fault is encountered the application can be restarted from the latest checkpoint state. This reduces the re-

computation time measurably. Checkpoint classification is based on several attributes [8] is represented in Fig 2.

1. Abstraction Level
2. Message coordination
3. Checkpoint Initiator
4. Checkpoint Granularity
5. Checpoint Scope
6. Storage Space

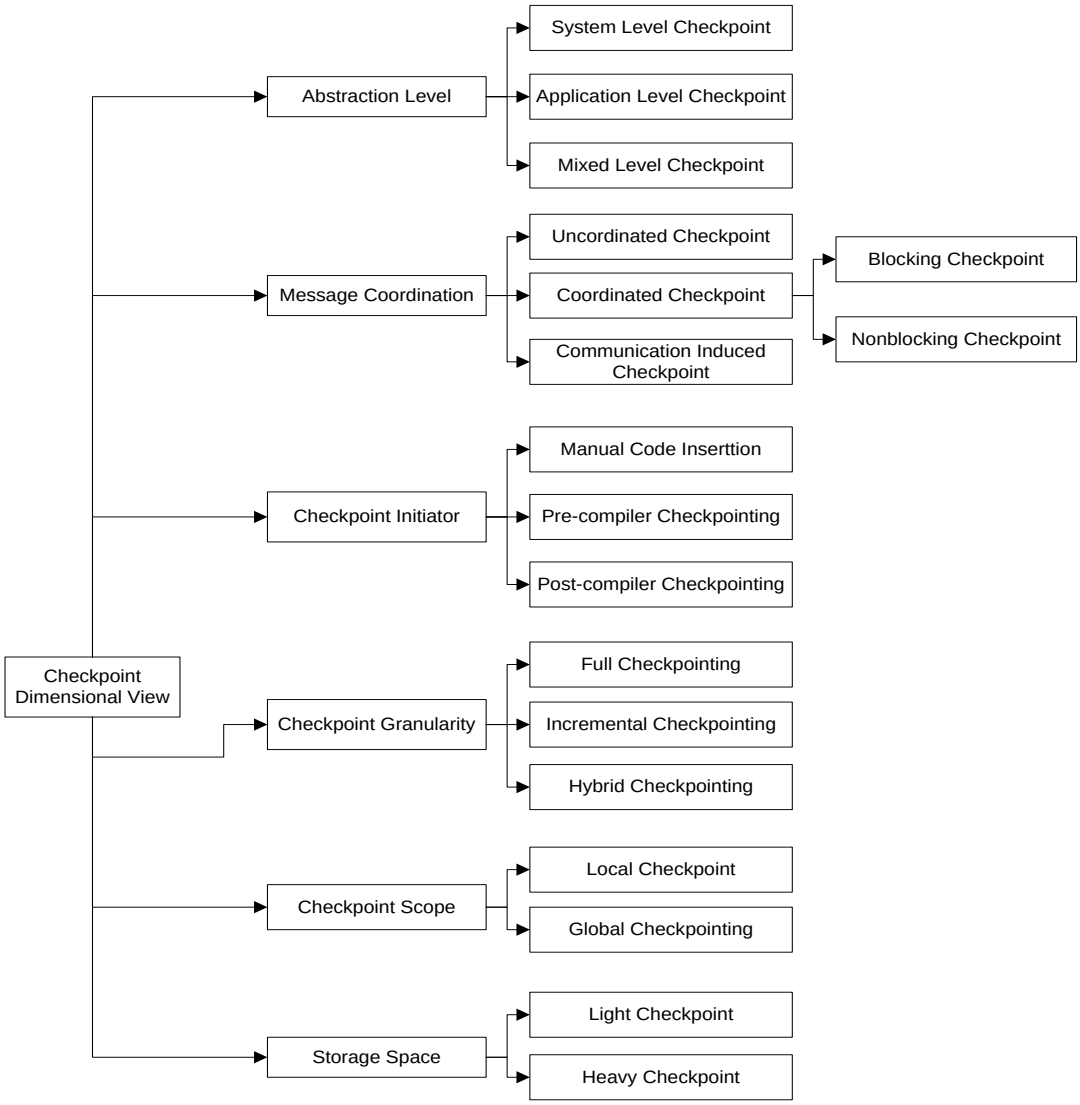

Fig 2: Classification of Checkpoint

A. Abstraction Level

The level of abstraction at which the current state of the application is saved is the criteria for classification. Under this classification there are three types

1. System Level Checkpoint: Automatic and Transparent check-pointing of applications at the operating system or middleware level is provided using this method. This mechanism has no knowledge about the characteristics of the application. The complete process image of the application is captured. It comprises of the attributes of process state such as program counter, registers and memory saved on the stable storage.

2 User or Application Level Checkpoint: Fault Tolerance is achieved by the application within itself by providing self-containing code. The application is designed in such a manner that it restarts automatically using the information in the restart file.

3. Mixed Level Checkpoint: It is the combination of System Level Checkpoint and User Level Checkpoint.

B. Message Coordination:

The way the system manages the in-transit and orphan messages is the criteria for this classification. Under this head, the following are the types [9]

1. Asynchronous/ Uncoordinated Checkpoint: Each process of the application takes the checkpoint independently without coordinating with the other process. Lack of synchronization makes the points inconsistent and during rollback, the points have to be searched for consistent global checkpoint.

Merits:

1. Control message exchange is avoided.
2. Process can perform checkpoint individually.

Demerits:

1. Possibility of Domino effect.
2. Many checkpoints for a process may lead to storage overhead.
3. A process may take a check-point that need not ever contribute to a consistent global check-point.

2. Synchronous/ Coordinated Checkpoint: In this approach, the processes maintain the consistent global checkpoint. It follows two phase commit. The tentative checkpoints taken in the first phase are made permanent in the second phase. On fault, the processes will roll back to the permanent checkpoint.
Merits:

1. Single permanent checkpoint lowers the stable storage overhead.
2. Does not suffer from domino effect
3. Simple rollback procedure

Demerits:

1. Involves exchange of multiple communication on exchanging messages.

There are two types of coordinated check-pointing

1. **Blocking Check-pointing**: To prevent orphan messages, the process remains blocked, until the entire check-pointing activity is complete after taking a local checkpoint. The process is allowed to resume its execution as soon as it finishes its local checkpoint. The disadvantage is the computation is blocked during the check-pointing

2. **Non-blocking Check-pointing**: The in transit and orphan messages may exist at the time of local checkpoint. The processes need not stop their execution while taking checkpoints. Preventing a process from receiving an application message that would result in inconsistent checkpoint in one of the issues encountered in this type.

3. **Communication Induced/Hybrid/Quasi-synchronous Checkpoint:** This method enforces at the creation of global checkpoint that is uncoordinated. The local checkpoints are created independently. However, domino effect is avoided by forcing additional checkpoints so as to ensure the eventual progress of the global checkpoint**.**

5. RELATED WORK

In [3] a dynamic adaptive fault tolerance strategy called DAFT is put forward. Two different fault tolerant strategies namely check-pointing and data replication is proposed by analyzing the mathematical relationship between different failure rates. This dynamic adaptive strategy check-pointing and data replication provides maximum serviceability there by ensures the cloud Service Level Objectives. It is evaluated under various conditions on metrics such as degree of fault tolerance, overhead of fault tolerance and response time. The DAFT strategy provides enhanced fault tolerance as conclusively demonstrated by the outcome of the experiments.

In [10] existing strategies on check-pointing were analyzed and compared to provide adequate work flow characteristics to the cloud computing environment. A light weight check-pointing model is proposed. A strong consistency is confirmed by the strategy called Adaptive Time based Coordinated Checkpointing (ATCCp). Checkpoint performance is improved by VIOLIN topology and soft check-pointing minimizes storage time. Checkpoint overhead and SLA violations are greatly reduced are shown in experimental results.

A reactive fault tolerance technique is proposed in [11] using check-pointing. The strategy called VM- μ Checkpoint framework protects VMs against transient errors. CoW-PC (Copy on Write – Presave in cache) algorithm is used. The cache memory is used in saving the checkpoints of the tasks running in the VMs in advance. In this algorithm in-memory incremental checkpoints are taken so that restoration can be done in-place. This greatly improves the efficiency of the restoration process.

In [12] a scheme has been implemented that uses software rejuvenation technique of fault tolerance. Two approaches have been employed. An adaptive failure detection and aging degree evaluation is done in the first stage which forecasts the cloud services to be rejuvenated. The second stage comprises of a component with applications. The designed architecture consists of various cloud services that comprises tightly coupled components and loosely coupled components executing in different VMs.

In [13] to avoid delays in task completion, a price history based check-pointing scheme based on SLA (Service Level Agreement) is proposed. This is achieved by reducing the number of checkpoints and there by improves the performance of the tasks. Total cost and number of checkpoints are effectively reduced in this scheme. A coordinator component in this scheme supports and manages the SLA between users and instances. The checkpoints are taken at two places. One at the raising edge where the price exceeds the threshold and the other is taken at the failure time foreseen by the average failure time and failure possibility.

In [14], a FTM is proposed as a middle layer to handle VM failures. Replication manager and Checkpoint managers are implemented to help the Recovery Overseer component to take preventive measures when a failed VM is detected by the Fault Detector. The fault detector identifies the failed VM reviewing the performance of the same tasks running at different machines at the same time and identifying the VM that deviates from the normal behavior. Replication Manager Component determines the number and the type of replicas using MaxRe algorithm. The saving of recovery information during execution is done periodically by the Checkpoint Manager and uses few resources when compare to replication. It uses coordinated checkpoint. The Recovery Overseer determines the usage of checkpoints whose main purpose is to recover from failure with minimum response time and waiting time. The system provides a Communication System with an intra messaging method.

An overview of workflow temporal checkpoint selection is presented in [15]. A temporal checkpoint selection strategy to deal with business workflows is proposed. Several consistency models for business and scientific workflows such as Throughput based temporal consistency model, Response-time based temporal consistency model, probability based temporal consistency have been discussed. The experimental results show the efficiency of the models above models.

In [16] Earlier strategy included a non-preemptive scheduling with task migration algorithm. This method had a major drawback of starting again the task in another virtual machine. This greatly increases the execution time of the migrated task. The proposed solution here includes an algorithm that migrates the aborted task and starts the execution at a point where the

latest checkpoint was saved. This leads to better performance and achieves QoS

In [17] the strategy involves computing the optimal number of checkpoints based on failure event distribution. This is generic in application. Various parameters like check-pointing overload, time delay have an impact on the cloud system. To optimize the impact and for better performance an adaptive algorithm is designed. The parameters that are considered for implementing the model are number of jobs, user request of multiple task, probability of failure event, checkpointing cost, checkpointing position, execution time and wall clock time. Dynamic optimization of checkpointing positions and local disk vs shared disk checkpointing are the main concepts used in Adaptive optimization of Fault Tolerance. Experimental results show the better suitability of the system for large scale applications.

In [18], fault tolerance module includes multilevel checkpoint functionality along with load balancing algorithms to decrease the checkpoint overheads. Checkpointing efficiency is improved and checkpoint overhead is reduced by considering various metrics such as computation time, migration time, latency, overhead, checkpoint ratio and response time. By considering the above parameters the algorithm finds an optimal checkpoint interval which provides high performance. The main areas addressed in this paper are checkpoint overhead and checkpoint latency.

In [19],a fault tolerant architecture of BFTCloud based on a Byzantine fault tolerance approach is proposed with five major operations such as the primary node selection, replica selection, request execution, primary node updating, and replica updating. The primary node selection is based on the Qos requirements, which handles the request. The replicas performs the required operation and the results are updated to the primary node. The BFTCloud provide high consistency and fault tolerance along with better performance.

6. PROPOSED MODEL

In cloud computing the hypervisor sits on the hardware providing abstraction for the above layers. The VMM provides a virtual environment for the virtual machine which has multiple operating systems running concurrently to service the user request. A dynamically adaptive checkpoint model is proposed to reduce the checkpoint counts and checkpoint overhead. The fault tolerance layer handles the checkpointing process. The numbers of checkpoints are ascertained dynamically within the process itself based on the rate of VM failures adaptively. The proposed model is shown in Fig 3.

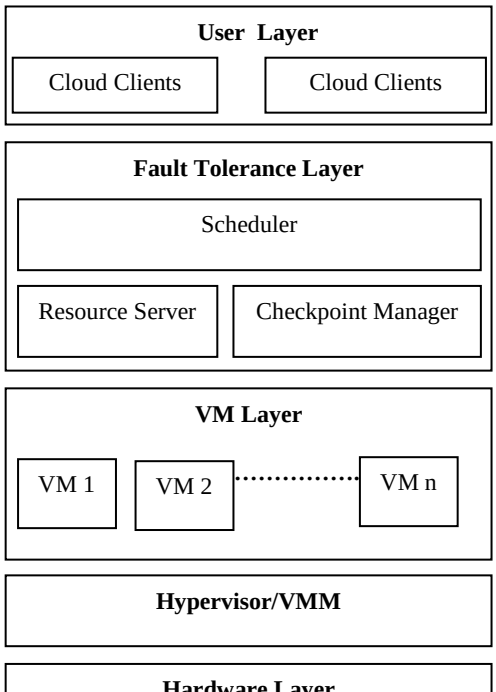

Fig 3: Checkpoint Model

The success of the cloud computing environment primarily depends on high level of cloud serviceability and to meet the requirements of SLO. An efficient fault tolerance model proposed in this paper helps to provide a dynamically adaptive checkpoint method minimizing checkpoints overhead and trails.

7. REFERENCES

[1] Y. Li, Z. Lan, “Exploit failure prediction for adaptive fault-tolerance in cluster”. Proceedings of the sixth IEEE International symposium on cluster computing and the grid, Vol 1, May 2006, pp. 531-538.

[2] P. Das, P. Mohan Khilar, "VFT: A Virtualization and Fault Tolerance Approach for Cloud Computing," Proceedings of the IEEE Conference on Information and Communication Technologies (ICT 2013), Kanyakumari, Tamil Nadu, in press, 2013, pp. 473-478.

[3] Dawei Sun, Guiran Chang, Changsheng Miao, XingweiWang,” Analyzing, modeling and evaluating dynamic adaptive fault tolerance strategies in cloud computing environments”, in JSupercomputer March 2013

[4] R. Buyyaa et al,” Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility,” Future Generation Computer Systems, vol. 25, 2009, pp. 599-616.

[5] D. Singh, J. Singh and A. Chhabra, “High availability of Clouds: Failover Strategies for Cloud Computing using Integrated Checkpointing algorithms”, IEEE International Conference on Communication Systems and Network Technologies, 2012.

[6] P. Gupta and S. Banga, “Review of Cloud Computing in Fault Tolerant Environment with Efficient Energy Consumption,” International Journal of scientific research and management, Vol. 1, Issue 4, 2013, pp. 251-254.

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

[8] S. Siva Sathya, S. Kuppuswami, K. Syam Babu, “Fault tolerance by check-pointing mechanisms in grid computing”, Proceedings of the International Conference on Global Software Development, Coimbatore, July 2007

[9] Raman Kumar, Dr. Parveen Kumar ,”Review of Some Checkpointing Schemes for Distributed and Mobile Computing”, Int. J. Advanced Networking and Applications Vol: 06 Issue: 06 (2015) ISSN: 0975- 0290.

[10] Bakhta Merofuel and Ghalem Belalem, "Adaptive time based coordinated checkpointing for cloud computing workflows", Scalable Computing, Practice and Expert

[11] Kalanirnika G R,V.M.Sivagami, Fault Tolerance in Cloud Using Reactive and Proactive Techniques International Journal of Computer Science and Engineering Communications Vol.3, Issue 3, 2015, Page.1159- 1164 ISSN: 2347–8586

[12] Jing Liu, Jiantao Zhou and Rajkumar Buyya, Software Rejuvenation based Fault Tolerance Scheme for Cloud Applications .2015 IEEE 8th International Conference on Cloud Computing.

[13] Daeyong Jung, SungHo Chin, KwangSik Chung, HeonChang Yu, JoonMin Gil, An Efficient Checkpointing Scheme Using Price History of Spot Instances in Cloud Computing Environment.

[14] L. Arockiam and Geo Francis E ,FTM- A Middle Layer Architecture for Fault Tolerance in Cloud Computing.,Special Issue of International Journal of Computer Applications (0975 – 8887) on Issues and Challenges in Networking, Intelligence and Computing Technologies – ICNICT 2012, November 2012.

[15] Wang Fu Tian, Liu Xiao & Yang Yun, Necessary and sufficient checkpoint selection for temporal verification of high-confidence cloud workflow systems, SCIENCE CHINA Information Sciences, May 2015, Vol. 58 052103:1–052103:16

[16] R. Santosh, T. Ravichandran, Non-Premptive Real Time Scheduling using checkpointing Algorithm for cloud computing. International Journal of computer Applications (0975-8887) Volume 80-No.9, October 2013

[17] Sheng Di, Yves Robert, Frédéric Vivien, Derrick Kondo, Cho-Li Wang, Franck Cappello, Optimization of cloud task processing with checkpoint-restart mechanism, , published in "SC13 Supercomputing-2013(2013)"DOI 10.1145/2503210.2503217.

[18] Dilbag Singh, Jaswinder Singh, Amit Chhabra, Evaluating Overheads of Integrated Multilevel Checkpointing Algorithms in Cloud Computing Environment, I. J. Computer Network and Information Security, 2012, 5, 29-38.

[19] Zhang, Y, Zheng, Z, & Lyu, M R 2011, 'BFTCloud:A byzantine fault tolerance framework for voluntary-resource cloud computing', In IEEE International Conference on Cloud Computing