REVIEW OF RELIABILITY ENHANCEMENT STRATEGY USING CHECKPOINTING IN CLOUD COMPUTING

Kamalpreet Kaur , Kamaljit kaur

Mtech Student , Ass.Professor

Department of Computer Science And Technology

Guru Nanak Dev University

Amritsar , India

ABSTRACT

Cloud computing is a standard for conveying, arranging and getting access to substantial scale disseminated computing applications over the system. In cloud computing, fault tolerance is a drastic issue and one of the metric which consider being most critical because the resource failure affects work execution, response time, throughput and execution of framework and system. To overcome this fault tolerance the load is divided among free machines for the fulfilment of the job. Reliability and energy efficiency are also the two key difficulties in cloud computing systems (CCS) that need to be considered while handling with cloud. The current review articles are either centered on the energy efficiency and reliability systems techniques in cloud computing. This paper also reveals the key issues of reliability and energy efficiency and their exchange off in cloud computing. We also talk about the arrangements on resource allocation, adaptation to non-critical failure techniques and energy administration components in cloud systems. In addition, different difficulties and research gaps in exchange off amongst reliability and energy efficiency are distinguished for future research and improvements.

Keywords**:** CCS, reliability, energy efficiency, Jobs, fault, fault- tolerance, energy consumption

1. INTRODUCTION

Today cloud computing is utilises everywhere where virtualization is used in information and communication technology (ICT). The basic favourable position of cloud computing is rising as another computing worldview which expects to give solid, low costs, high accessibility, adaptability and flexibility for end-clients. Many research issues are completely tended to in cloud, for example, Fault tolerance, security, and so on. Fault tolerance is a vital key issue in cloud computing and it is worried about every one of the methods important to empower a framework to endure programming faults staying in the framework after its advancement. The fault tolerance method empowers dependability and robustness in cloud computing. The advantages of fault tolerance in cloud computing incorporate failure recovery bring down cost, enhanced execution measurements and so on.

In cloud computing the resources are accessed remotely sometimes which creates faults during that process. To correct the faulty components fault tolerance methods are used and correct them. Due to its remote access there are lots of chances of errors so to achieve reliability in real time cloud computing. Fault tolerance is achieved by error processing having two constituent phases. These phases are “effective error processes” i.e. before the occurrence of error and “latent error processing” means error will not occur again. Fault tolerance is one of the key issues among all. It is worried about every one of the strategies important to empower a framework to endure programming faults staying in the framework after its improvement [1]. At the point when a fault happens, these procedures give components to the product framework to avoid framework disappointment event [2].

In cloud situations, a few sorts of virtual machines are facilitated on an indistinguishable physical server from foundation [3]. In cloud, customers should pay according to utilization and does not pay for the unused resources. We have three kinds of cloud conditions: Public, Private, and Hybrid clouds [4]. Public cloud is standard model which suppliers make a few assets.

Public cloud administrations might be free or not. In the open clouds which they are running applications remotely by vast specialist co-ops and offer a few advantages over private clouds [5]. Private Cloud includes interior administrations of a business that isn't accessible for conventional individuals. Basically Private clouds are a promoting term for an engineering that gives facilitated administrations to specific gathering of individuals behind a firewall [6]. Likewise, Hybrid cloud is the combination of private and public cloud. In this compose, cloud supplier has an administration that has private cloud part which just opens by guaranteed staff and shielded by firewalls from outside getting to and an open cloud condition which outer clients can access to it [7]. There are three noteworthy sorts of administration in the cloud condition: SaaS, PaaS, and laaS. In cloud, like each proposed innovation, there are a few issues which included it and one of them is Restricted Access System(RAS) factor [8]. For having great and superior, cloud supplier must meet a few administration highlights to guarantee enhancing RAS (Restricted Access System) parameters of its administration, for example,

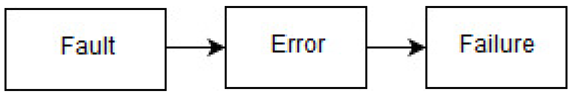
1. Accessibility administration
2. Access control administration
3. Vulnerability and issue administration
4. Patch and setup administration
5. countermeasure
6. Cloud framework utilizing and get to checking

2. FAULT TOLERANCE IN CLOUD COMPUTING

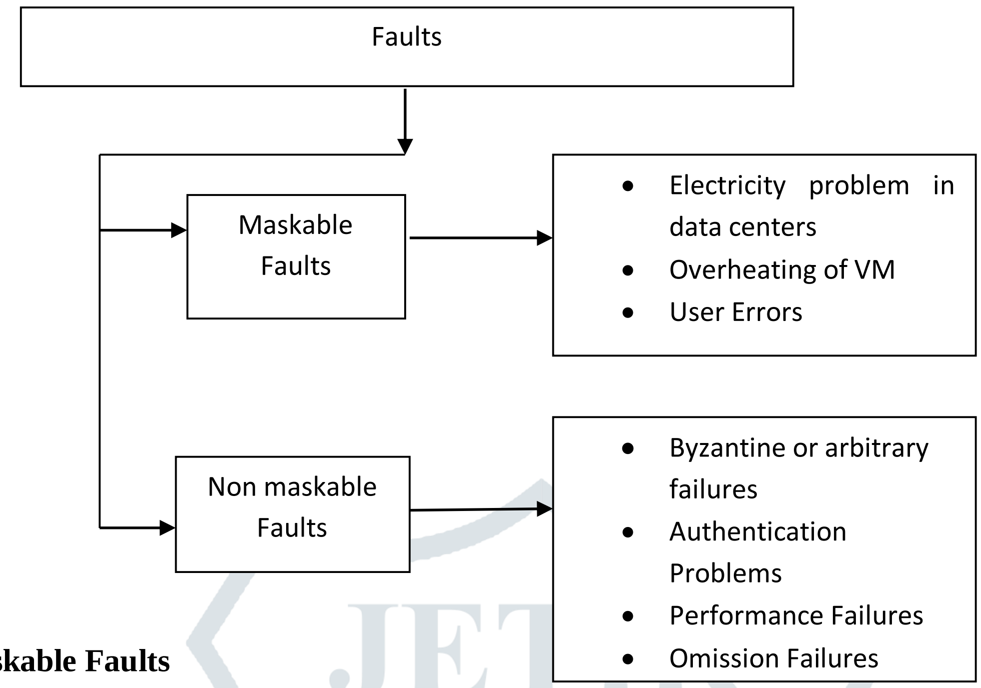
Fault tolerance includes to right and consistent operation even within the sight of faulty segments. It is the craftsmanship and exploration of building computing frameworks that keep on operating agreeably within the sight of faults. A fault tolerant framework might have the capacity to endure at least one fault composes including-transient, irregular or changeless equipment faults, programming and plan mistakes, administrator problem or physical damage. In ongoing cloud applications, preparing on computing hubs is done remotely which has a high likelihood of event of mistakes. These occasions increment the requirement for fault tolerance systems to accomplish unwavering quality for the ongoing computing on cloud framework.

Faults, Errors and Failures along with Relationship among them

Faults are abnormal external situations causing deviation from actual results. Deviation from actual result if persistent causes errors. In case, error last for longer period of time then it results in failure [19]. Broadly faults are categorised into recoverable and unrecoverable faults. Recoverable faults allow the recovery and normal operation to be reinitiated. Unrecoverable faults are those that stops the execution of the application program such as virtualization problem in cloud and they are irreversible faults. Regardless of types of fault, frequency of occurrence of faults effect performance of cloud [20].



2.1 TYPES OF FAULTS



- **Maskable Faults**

Maskable faults initiates due to temporary or transient situations occurring within the cloud [21]. These faults allow the reversible condition to be established which can be used to restore the system to its initial state [22]. Maskable faults are categorized as under:

1. **Electricity problems in Data centres**

Electricity problems in Data centres could be caused by equipments through which electrons are being transmitted. The transformers used in data center machines for step up or step down the voltage could cause the problem also. The phase down could lead to over or under flow of electricity causing damages to the electronic equipment [23].

2. **Overheating of virtual machines**

Overheating could be caused due to the resistance present within the medium used to generate virtual machines within cloud. In case resistance increases due to carbon over the medium of transfer, overheating is the result. Data will be transferred at much slower rate than normal in such situations [24].

3. **User Errors**

User errors could be due to wrong interaction of hardware with other hardware equipments used in cloud. This interaction of hardware with other hardware resources could result in disintegrated results [25].

- **Non Maskable Faults**

These faults occur due to the problems that persist and situation cannot be rectified. The node crash is an example of this fault. This fault is also known as fail stop fault. Restarting the node could be the solution of this fault but it causes loss of state and entire operation performed by the node required to be restarted [26].

1. **Byzantine or Arbitrary Failures**

This type of fault is caused due to cloud server misbehaviour. The cloud server in this case may send different responses to same query by different users of the cloud. The reliability of cloud is at stake considering such failures [9].

2. Authentication Problems

The cloud server in this situation may identify the problems that are caused by unauthorised user but cannot stop the operation performed by the unauthorised users. Cloud host data from millions of users but trust will be lost by the application of such problem [27].

3. Performance failures

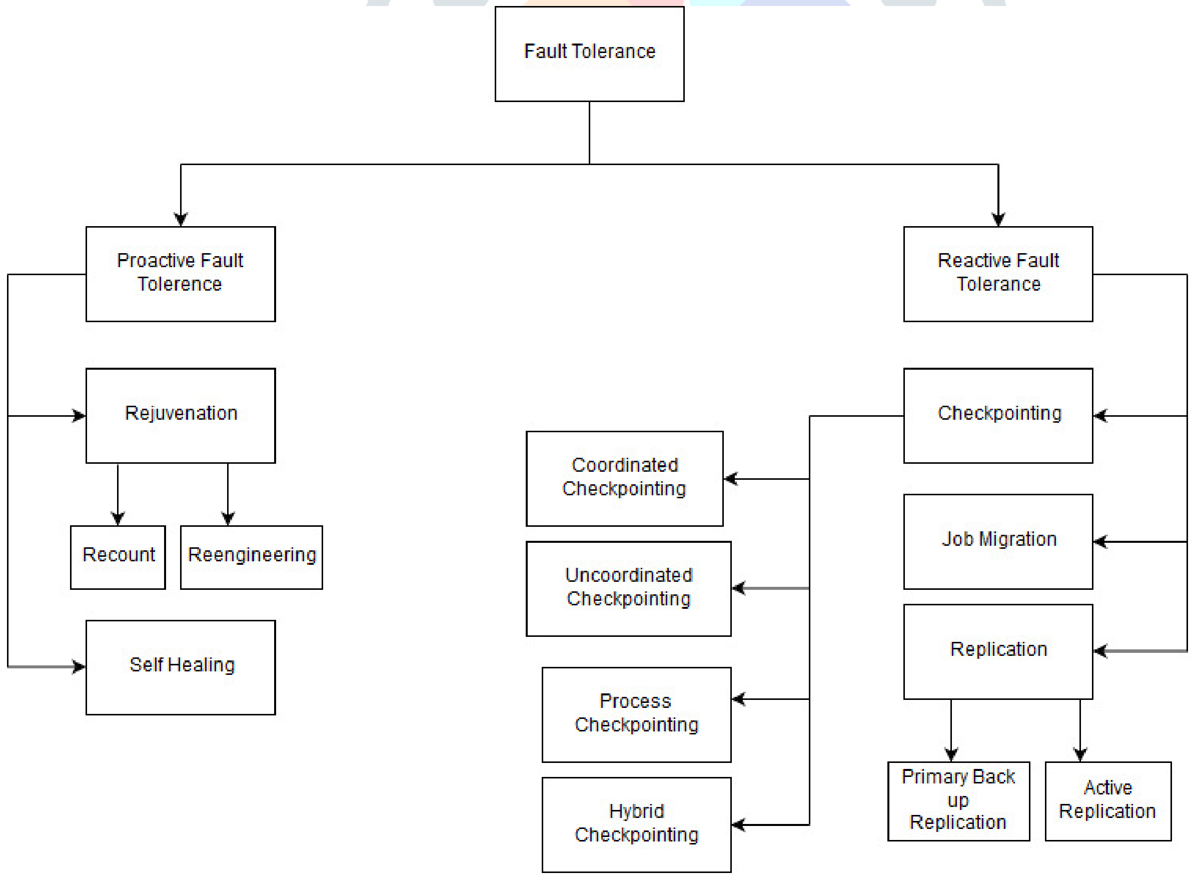
This problem is caused due to timing of delivery of service. The server in this situation is producing correct result but it is not delivered to the user on time. It is either delivered early or late to the user. Performance degradation greatly affects the QoS of the cloud [28].

4. Omission Fault

In this case services by the cloud server is infinite late. In other words services are not delivered at all to the user. Service level agreement although bound both user as well as cloud service provider but cloud user will pay even though they are not getting the services of the cloud in this case. Trust level degrades due to this problem [29].

2.2 FAULT TOLERANCE TECHNIQUES IN CLOUD COMPUTING

To increase the performance and reliability of cloud it is required to manage the faults occurring during job processing. For this purpose some fault tolerance techniques can be used.



- **CHECKPOINTING**

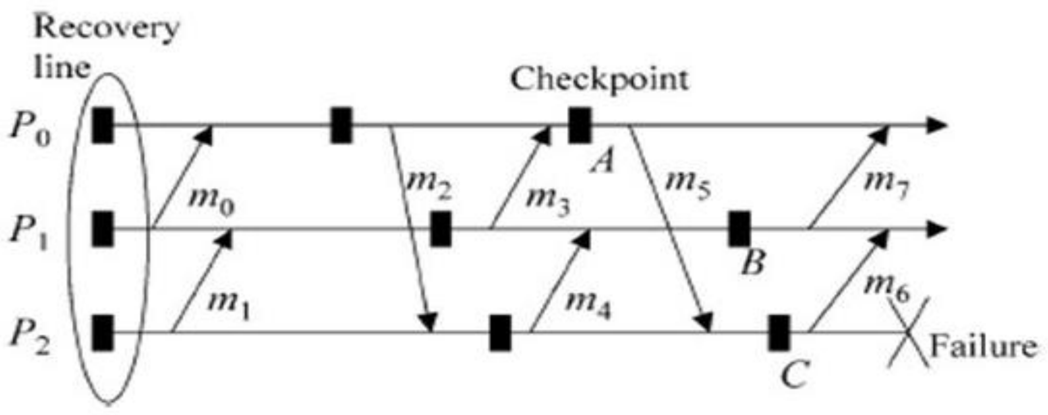
1. **Coordinated checkpointing:**

It utilises the technique of message passing and process checkpointing that design the rollback recovery system frameworks at the concurrent application level associated with cloud. The target of this convention is to assemble a consistent flowed depiction of the appropriated system [30]. A flowed review is an accumulation of process checkpoints (one for each technique), and a social event of in-flight messages [31]. [32, 33].

2. **Uncoordinated checkpointing:**

The problem corresponding to the checkpoint failure recovery is solved using uncoordinated checkpointing strategy proposed by [32, 34]. The enhanced reliability is achieved using the uncoordinated checkpointing strategy as discussed by [16, 35, and 36]. To lessen the common costs of encouraged checkpointing, uncoordinated checkpointing conventions have in this way been handled within virtual machines. On the fault free bit of the execution, the crucial idea is to clear the co-arrangement of checkpointing, concentrating on a decreasing of the I/O weight when checkpoints are secured on shared space within cloud, and the diminishment of deferrals or extended framework utilize while arranging the enrolment, uncoordinated conventions go for obliging the restart of a unimportant game plan of strategies when a failure happens [37, 38 and 39]. Uncoordinated checkpointing provides following features:

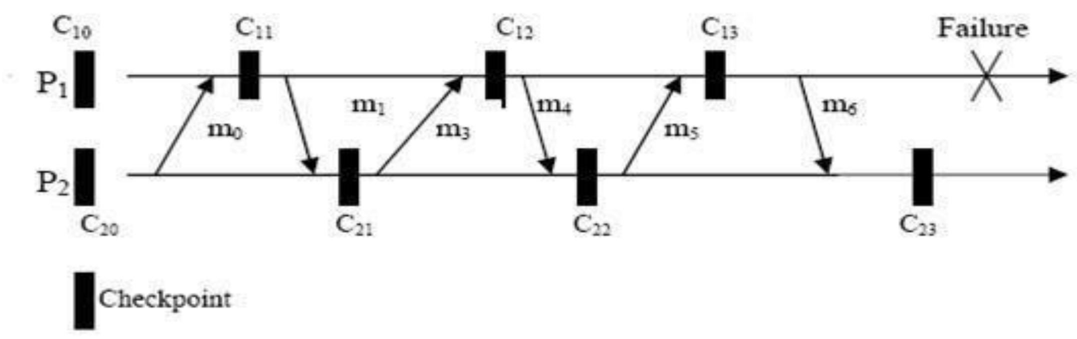
- It allows each process having maximum autonomy for deciding when to consider checkpointing.
- Recovery is based on the selection of recovery server which is accomplished using recovery line.
- To determine a consistent global checkpoint during recovery each process records the dependencies among their checkpoints during fail free operation.
- Uncoordinated checkpointing is not useful to reconstruct a consistent global state.



3. **Processes checkpointing:** The target of process checkpointing in cloud computing is to spare the current state of a strategy. In current High Performance Computing applications, a procedure involves various customer level or system level strings, making it a concurrent application without any other person. Process checkpointing systems for by and large use a locking strategy named coarse grain to prevent rapidly the execution of the impressive number of strings of the method, giving them an overall point of view of its present state, and decreasing the issue of sparing the methodology state to a progressive issue [40, 41]. Processes checkpointing provides following features:

- In this checkpointing coordinator takes the information of checkpoint and broadcast the message to all processes.
- This checkpointing is application specific.
- In this checkpoint every process has to be participating in every checkpoint.

- The main role of this checkpointing is to prevent a process from receiving application messages that could make the checkpoint consistent.



4. **Hybrid Checkpointing:** The likelihood of Hierarchical Checkpointing is genuinely essential: frames are coursed in social affairs; shapes having a place with a comparative get-together for organizing the rollbacks, It utilized the message logging for uncoordinated checkpointing between get-togethers within data centers [42, 43]. Regardless, the state of a singular system depends on the correspondences between social affairs, yet moreover upon the coordinated efforts with various methods inside the get-together [37, 40].

- **Replication**

1. **Primary Backup Replication:** This technique uses one copy, the basic that has an extraordinary impact: it gets invocations from client methodology and returns responses [47, 48]. Server x's basic imitation is shown prim(x); diverse duplicates are fortifications Backups work together straightforwardly just with the fundamental duplicate, not the client shapes within cloud computing [46].
2. **Active Replication:** Also called the state-machine approach, this system gives all imitations a similar part without the concentrated control of the essential reinforcement procedure [49, 50]. Active replication requires non crashed reproductions to get the summons of customer forms in a similar request. This requires correspondence crude that fulfils the request and the atomicity properties presented before [50, 51].

- **Software rejuvenation**

Software rejuvenation is the component by which software component which is failed required restarting [52, 53]. Once in a while this fault is long lasting however generally this fault is small [54, 55]. Software rejuvenation is additionally arranged as under:

1. Re-engineering

On the off chance that software part faults forever then software segment utilized inside cloud required substitution. Procedure of changing or replacing the current software part of the cloud is known as Re-engineering [10].

2. Re-Count

In this approach whole virtualized software important to deal with cloud requires change. The software part is created from scratch for this situation. Re-building is financially savvy as contrast with recreation [23, 56].

- **Self Healing**

The VM if over-burden required relocation. After the movement VM is re-established to its underlying state. The way toward re-establishing VM to its unique state is known as Self Healing [16, 57].

- **Job Migration**

This process includes exchanging VM assets to ideal VM inside or outside the extent of current datacenter. Such movement falls under the class of pre-emptive relocation in which inter or intra cloud movement is the source [58, 59].

2.3 Impact of Failures on Energy Consumption

In literature the evaluation of various techniques for energy consumption and impact of utilization has been done. But it remains unclear that how the occurrence of failure will effect on energy consumption and reliability. So it is necessary to utilize the fault tolerance technique that reduces the failure in CCS and also remains optimized in terms of reliability. Also it is important to study the relationship between the failure and energy consumption.[60].

Next section explores the existing work which is done to achieve fault tolerance in cloud based system.

2.4 ANALYSIS OF VARIOUS FAULT TOLERANCE TECHNIQUES AND THEIR APPLICATIONS

According to researchers and companies recently study fault-tolerance techniques are divided in two different environments of cloud computing: environments that are reliable and on demand based, and environments that are unreliable that are with spot. The fault-tolerance techniques are more required in unreliable environments. Our study was performed in the latter category of the environments to provide the cost-effectiveness of task execution.

The unreliable environment uses Spot instance, and the main aims of the various low cost spot instances focus on performing tasks. The spot instances in the Amazon Elastic Compute Cloud (EC2) offer lower price but reduced reliability. In the spot instances environment, there are numerous studies on resource allocation, SLA and fault tolerance.

Resource allocation: the main problem is to execute tasks that are pooled on intermittent VMs. This could be solved by using runtime estimation mechanism and the given technique achieves it [61]. It also achieves the user's satisfaction during scheduling and increases the supplier's income. [62].

SLA: It gives the decision model that is probabilistic based and gives minimum cost according to the SLA. It uses the model based on probability to ensure the optimized cost, performance and reliability. [63]

Fault Tolerance: It utilises the check pointing schemes for considering costs and occurrence of failure. It execute job by assuming the transfer cost fixed. It uses three fault tolerance scheme check pointing, replication and migration[30].

2.5 Comparative Study of Distinct Parameters is Presented in This Section

Reference No.	Energy Consumption	Execution Time	Accuracy	Time Redundancy	Down Time	Migration Time	Technique Used	Remarks
8.	Less energy consumption	Improved by 5%	Not consider	Redundancy is not consider	Must be improved	Improved by 5%	Hybrid load and checkpointing strategy	Multiple fault tolerance strategies such as check pointing and load balancing is combines together to achieve the desired result.
35.	5% improved	Enhanced by 4%	More accurate than any other technique	Improved by 7%	No improvement is shown	Must be enhanced by 2%	Reducing Energy consumption based on distance between the migrating vms	This technique enhances the level of accuracy, software reliability metrics and provides efficient energy consumption
64.	Less energy consumption	Consider as parameter and improved by 7%	Parameter not consider	Must be consider	Parameter not considered	Time is improved by 7%	Computational complexity reduction through checkpointing	The technique improved fault tolerance with minimizing computational overhead and uses checkpointing to achieve optimal result.

65.	Energy is consumed by the margin of 18%	Enhanced by 3%	Must be further enhanced	Not considered	Improved by 8%	Enhanced by 3%	System level fault tolerance by using replication mechanism	System level faults are handled using backup even operating system failure is also manage
4.	Not considered this parameter	Execution time is better than any other technique	Accuracy is enhanced by 3%	Redundancy is improved	Must be improved further	Not consider the parameter	Time slots are formed to reduce the redundancy within the data to reduce execution tine	It utilized time redundancy based fault-tolerance techniques

3. CONCLUSION

Today cloud computing is widely used but there are still has research gap that should be addressed. Because of widely usability of cloud computing infrastructure the main concerns are energy efficiency, reliability and scalability. In this paper the analysis of various failure that has been occurred in CCS system and also the mechanism to handle these failure to make system reliable. It also describes the various methods that increase the reliability of CCS system along with their limitations. Because of the increase in usage of cloud services the energy consumption is more and design becomes complex. It also gives comprehensive review of the various techniques that are used for energy conservation in cloud. By analysing these technique we observed that if any technique is adopt for reliability in cloud computing services than it must be impact on energy usage of system. By utilizing various methods like backing up resource, replicate the system or storing logs fault must be tolerated but it may increase energy consumption. So we have concluded that the reliability and energy consumption are critical points thus there is need for a policy to improve them.

4. REFERENCES

[1] V. M. Sivagami, “Survey on Fault Tolerance Techniques in Cloud Computing Environment,” International Journal of Scientific Engineering and Applied Science (IJSEAS) vol. 1, no. 9, pp. 419–425, 2015.

[2] A. Bala and I. Chana, “Fault Tolerance-Challenges, Techniques and Implementation in Cloud Computing,” International Journal of Computer Science Issues(IJCSI), vol. 9, no. 1, pp. 288-293, January 2015.

[3] C. N. Höfer and G. Karagiannis, “Cloud computing services: Taxonomy and comparison,” Journal of Internet Services and Applications, Springer, vol. 2, no. 2, pp. 81–94, Sept. 2011.

[4] A. E. El-desoky, “Improving Fault Tolerance in Desktop Grids Based On Incremental Checkpointing,” IEEE International Conference on Computer Engineering and Systems, ICCES’06, pp. 386–392, Nov. 2006.

[5] A. Kumar, "An Efficient Checkpointing Approach for Fault Tolerance in Time Critical Systems wi ith Energy Minimization," IEEE International Conference on computing, communication and automation, pp. 704–707, May 2015.

[6] S. Choi, K. Chung, and H. Yu, "Fault tolerance and QoS scheduling using CAN in mobile social cloud computing," Cluster Computing, springer, vol. 17, no. 3, pp. 911–926, Sept. 2014.

[7] A. Kumar, "Improved EDF Algorithm for Fault Tolerance with Energy Minimization," IEEE International Conference on computational Intelligence and Communication Technology(CICT), pp. 370-374, Feb. 2015.

[8] A. Kumar and B. Alam, "Real-Time Fault Tolerance Task Scheduling Algorithm with Minimum Energy Consumption," Proceedings of the Second International Conference on Computer and Communication Technologies. Advances in Intelligent Systems and Computing, Springer, vol. 380, pp. 441-448, Sept. 2015.

[9] Y. Zhang, Z. Zheng, and M. R. Lyu, "BFTCloud: A Byzantine Fault Tolerance framework for voluntary-resource cloud computing," in Proc. IEEE 4th International Conference on Cloud Computing, pp. 444–451, July 2011.

[10] B. Mills, T. Znati, and R. Melhem, "Shadow Computing : An Energy-Aware Fault Tolerant Computing Model," IEEE International Conference on Computing, Networking and Communications(ICNC), pp. 73–77, Feb. 2014.

[11] M. E. M. Diouri, O. Glück, L. Lefevre, and F. Cappello, "Energy considerations in checkpointing and fault tolerance protocols," in Proc. IEEE International Conference Dependable Systems and Networks workshops, pp. 1–6, June 2012.

[12] M. Salehi, M. K. Tavana, S. Rehman, M. Shafique, A. Ejlali, and J. Henkel, "Two-State Checkpointing for Energy-Efficient Fault Tolerance in Hard Real-Time Systems," IEEE Transaction on Very Large Scale Integration (VLSI)Systems, vol. 24, no. 7, pp. 2426–2437, July 2016.

[13] C. A. Chen, M. Won, R. Stoleru, and G. G. Xie, "Energy-efficient fault-tolerant data storage and processing in mobile cloud," IEEE Transaction on Cloud Computing, vol. 3, no. 1, pp. 28–41, Jan. 2015.

[14] C. Colman-Meixner, C. Develder, M. Tornatore, and B. Mukherjee, "A survey on resiliency techniques in cloud computing infrastructures and applications," IEEE Commun. Survey Tutorials, vol. 18, no. 3, pp. 2244–2281, 2016.

[15] S. Wang, A. Zhou, C. Hsu and X. Xiao, "Provision of Data-intensive Services through Energy- and QoS-aware Virtual Machine Placement in National Cloud Data Centers," IEEE Transactions on Emerging Topics in Computing, vol. 4, no. 2, pp. 290-300, Dec. 2015.

[16] M. Amoon, "Adaptive Framework for Reliable Cloud Computing Environment," IEEE Access, vol. 4, pp. 9469–9478, Nov. 2016.

[17] B. An, V. Lesser, D. Irwin, and M. Zink, "Automated negotiation with decommitment for dynamic resource allocation in cloud computing," in Proc. ACM 9th International Conference on Autonomous Agents and Multiagent Systems, vol. 1, pp. 981–988, 2010.

[18] Sherif Abdelwahab, Bechir Hamdaou, "R EPLISOM : Disciplined Tiny Memory Replication for Massive IoT Devices in LTE Edge Cloud", IEEE Internet of Things, vol. 3, no. 3, pp. 327-338, June 2016.

[19] M. Zhang, H. Jin, X. Shi, and S. Wu, "VirtCFT : A Transparent VM-Level Fault-Tolerant System for Virtual Clusters," IEEE Access, pp. 147–154, 2010.

[20] A. S. Perspective, R. Jhawar and V. Piuri "Fault Tolerance Management in Cloud Computing," IEEE Systems Journal, vol. 7, no. 2, pp. 288-297, 2013.

[21] G. Yao, Y. Ding, S. Member, and K. Hao, “Using imbalance characteristic for fault - tolerant workflow scheduling in Cloud systems,” IEEE Transactions on Parallel and Distibuted Systems, vol. 28, no. 12, pp. 3671-3683, 2017.

[22] Mohammed el Mehdi Diouri, Olivier Gluck, Laurent Lefevre and Franck Cappelo, “ECOFIT: A framework to estimate energy consumption of fault tolerance protocols for HPC applications,” in Proc. 13th IEEE International Symposium on Cluster, Cloud and Grid Computing, pp. 522–529, 2013.

[23] A. Zhou, S. Wang, B. Cheng, Z. Zheng, F. Yang, R. N. Chang, “Cloud Service Reliability Enhancement via Virtual Machine Placement Optimization,” IEEE Transactions on Services Computing, vol. 10, no. 6, pp. 902-913, 2017.

[24] M. Kalra and S. Singh, “REVIEW A review of metaheuristic scheduling techniques in cloud computing,” Elsevier, Egyptian Informatics Journal, vol. 16, no. 1, pp. 275–295, 2015.

[25] C. Cheng and H. Liao, “A Malicious-Resilient Protocol for Consistent Scheduling Problem in the Cloud Computing Environment,” IEEE Access, vol. 58, no. 2, 2015.

[26] T. Wang and S. C. Liew, “Frequency-Asynchronous Multiuser Joint Channel-Parameter Estimation, CFO Compensation and Channel Decoding,” IEEE Transactions on Vehicular Technology, vol. 65, no. 12, pp. 9732-9746, 2016.

[27] D. Song, E. Shi, I. Fischer, and U. Shankar, “Cloud Data Protection for the Masses,” IEEE Access, vol. 45, no. 1, pp. 39–45, 2012.

[28] D. F. Bermudez Garzon, C. G. Requena, M. E. Gomez, P. Lo, and J. Duato, “A Family of Fault-Tolerant Efficient Indirect Topologies,” IEEE Transactions on Parallel and Distributed System, vol. 27, no. 4, pp. 927–940, Apr. 2016.

[29] F. Li, Z. Li, W. Huo, and X. Feng, “Locating Software Faults Based on Minimum Debugging Frontier Set,” IEEE Access, pp. 1–18, 2016.

[30] J. T. Goiri, F. Julia, J. Guitart, “Checkpoint-based fault tolerant infrastructure for virtualized service providers,” Proc. IEEE Network Operations and Management Symposium(NOMS), pp. 455–462, 2010.

[31] K. B. Ferreira, R. Riesen, P. Bridges, D. Arnold, and R. Brightwell, “Accelerating incremental checkpointing for extreme-scale computing,” Future Generation Computer Systems, Science Elsevier, vol. 30, no. 1, pp. 66–77, 2014.

[32] \Y. Jin, C. Jiannong, and W. Weigang, “Efficient global checkpointing algorithms for mobile agents,” Proc. IEEE Second International Conference on Semantics, Knowledge and Grid, vol. 20, no. 7, pp. 825–838, 2008.

[33] A. Zhou, Q. Sun, and J. Li, “Enhancing Reliability via Checkpointing in Cloud Computing Systems,” IEEE China Communications, vol. 14, no. 7, pp. 1-10, 2017.

[34] G. B. Bakhta Meroufeland, “Adaptive time-based coordinated checkpointing for cloud computing workflows,” Scalable Computing: Practice and Experience, vol. 15, pp. 153–168, 2014.

[35] X. Cui, B. Mills, T. Znati, and R. Melhem, “Shadows on the cloud: An energy-aware, profit maximizing resilience framework for cloud computing,” Proc.IEEE 4th International Conference on Cloud Computing Services, pp. 15–26, 2014.

[36] S. Ghemawat, H. Gobioff, and S.-T. Leung, “The Google file system,” in proc. 19th ACM symposium on Operating systems principles(SOSP), pp. 29-43, 2003.

[37] J. E. M. S. Agarwal, R. Garg, M. S. Gupta, “Adaptive incremental checkpointing for massively parallel systems,” Proc. ACM 18th Annual International Conference on Supercomputing, pp. 277–286, 2004.

[38] G. C. J. Ansel, K. Arya, "DMTCP: Transparent checkpointing for cluster computations and the desktop," Proc. IEEE International Symopsium on Parallel and Distributed Processing, pp. 1–12, 2009.

[39] Y. Zhang and K. Chakrabarty, "Energy-aware adaptive checkpointing in embedded real-time systems," Proc IEEE Design, Automation and Test in Europe Conference and Exhibition, pp. 918–923, 2003.

[40] S. Yi, J. Heo, and Y. Cho, "Adaptive Page-level Incremental Checkpointing based on Expected Recovery Time," Proc. ACM symposium on Applied computing, pp. 1472–1476, 2006.

[41] R. R. Chandrasekar, A. Venkatesh, K. Hamidouche, and D. K. Panda, "Power-check: An energy-efficient checkpointing framework for HPC clusters," Proc. IEEE 15th International Symposium on Cluster, Cloud and Grid Computing, pp. 261–270, 2015.

[42] J. L. B. Egger, Y. Cho, C. Joe, E.Park, "Efficient Checkpointing of Live Virtual Machine Migration", IEEE Transactions on Computers, vol 65, no. 10, pp. 3041–3054, 2016.

[43] X. Ni, E. Meneses, N. Jain, and L. V Kale, "ACR : Automatic Checkpoint / Restart for Soft and Hard Error Protection," Proc. ACM International Conference on High Performance Computing, Networking, Storage and Analysis 2013.

[44] Q. Wei, B. Veeravalli, B. Gong, L. Zeng, and D. Feng, "CDRM: A cost-effective dynamic replication management scheme for cloud storage cluster," Proc. IEEE International Conference on Cluster Computing, pp. 188–196, 2010.

[45] K. Ranganathan and I. Foster, "Identifying dynamic replication strategies for a high-performance data grid," Proc. Springer International Grid Computing Workshop, vol. 2242, pp. 75–86, 2001.

[46] W. Lang, J. M. Patel, and J. F. Naughton, "On energy management, load balancing and replication," ACM Special Interest Group on Management of Data, vol. 38, no. 4, pp. 35-42, 2010.

[47] D. Jung, J. Lim, H. Yu, and T. Suh, "Estimated Interval-Based Checkpointing (EIC) on Spot Instances in Cloud Computing," ACM, Journal of Applied Mathematics, 2014.

[48] Y. Xiang, H. Liu, T. Lan, S. Subramaniam, and H. Huang, "Optimizing Job Reliability via Contention-Free , Distributed Scheduling of VM Checkpointing," ACM Special Interest Group on Data Communication(SIGCOMM), workshop on Distributed cloud computing, pp. 59–64, 2015.

[49] W. Li, Y. Yang, and D. Yuan, "A novel cost-effective dynamic data replication strategy for reliability in cloud data Centers," Proc. - IEEE 9th Int. Conf. Dependable, Auton. Secur. Comput. (DASC) , pp. 496–502, 2011.

[50] A. Z. D. Boru, D. Kliazovich, F. Granelli, P. Bouvry, "Energy-Efficient Data Replication in Cloud Computing Datacenters," Cluster Computing, Springer, vol 18, no. 1, pp. 385-404, 2015.

[51] Y. Lin and H. Shen, "EAFR: An Energy-Efficient Adaptive File Replication System in Data-Intensive Clusters," IEEE Trasactions on Parallel and Distributed System, vol. 28, no. 4, pp. 1017–1030, 2017.

[52] S. C. K. Kourai, "A Fast Rejuvenation Technique for Server Consolidation with Virtual Machines," IEEE International Conference on Dependable Systems and Networks, 2007.

[53] K. S. Trivedi and K. Vaidyanathan, "Software Rejuvenation - Modeling and Analysis," Springer International Federation for Information Processing(IFIP), vol. 157, pp. 151–182, 2004.

[54] C. S. Shih and T. K. Trieu, "Shadow phone: Context aware device replication for disaster management," Proc. 5th IEEE International Conference on Service-Oriented Computing and Applications (SOCA) , pp. 1-5, 2012.

[55] M. P. H. Goudarzi, "Energy-Efficient Virtual Machine Replication and Placement in a Cloud

Computing System," IEEE 5th International Conference on Cloud Computing, pp. 750–757, 2012.

[56] J. T. L. Silva, J. Alonso, "Using Virtualization to Improve Software Rejuvenation," IEEE Transactions on Computers, vol. 58, no. 11, pp. 1525–1538, 2009.

[57] E. Pinheiro, W. Weber, and L. Barroso, "Failure trends in a large disk drive population," Proc. ACM 5th USENIX conference on File and Storage Technologies, pp. 2-2, Feburary 2007.

[58] D. Jung, S. Chin, K. S. Chung, and H. Yu, "VM Migration for Fault Tolerance in Spot Instance Based Cloud Computing," Springer International Conference on Grid and Pervasive Computing, vol. 7861, pp. 142–151, 2013.

[59] A. Gupta, B. Acun, O. Sarood, and L. V. Kale, "Towards realizing the potential of malleable jobs," IEEE 21st International Conference High Performance Computing(HIPC), pp. 1-10, 2014.