International Journal of Engineering Applied Sciences and Technology, 2020
Vol. 5, Issue 4, ISSN No. 2455-2143, Pages 610-615
Published Online August 2020 in IJEAST (http://www.ijeast.com)

FAULT TOLERATING MECHANISM IN DISTRIBUTED COMPUTING ENVIRONMENT

Lokendra Gour
Department of Computer Science
AKS University, Satna, Madhya Pradesh, India

Dr. Akhilesh A. Waoo
Department of Computer Science
AKS University, Satna, Madhya Pradesh, India

***Abstract*— Large scale distributed systems encompass heterogeneous computational machines, workloads and sub-systems dispersed diversely across the cloud environment. These sub-systems frequently encounter faults and failures due to different data structures, hardware/software malfunction, and communication delay. To speed up computation in such a situation a fault tolerating infrastructure is implemented by adopting a machine learning approach. Under machine learning, an artificial neural network (ANN) captures, manipulates, and updates the states and behaviors of the sub-systems in the servers and worker's machines. Multiple layers of neurons (i. e., deep learning) can handle large scale distributed systems with large datasets. Adopting the variants of a stochastic gradient descend algorithm on sub-systems (also known as computational nodes) the efficiency, and reliability of a distributed system are enhanced significantly. In high-performance computing (HPC) applications fault tolerance mechanisms must be embedded to recover from system failures.**

Keywords— **Distributed System, Cloud Environment, Fault Tolerance, Machine Learning, Artificial Neural Network**

I. INTRODUCTION

At present, the size of available data for training deep models has increased significantly. Exchanging the model parameters increases the communication overhead which causes the bottleneck problem in a distributed learning algorithm. For instance, calculating the sparse on the gradients to zero-out the non-important values will reduce the communication bit-rate. One of the great challenges in distributed computing is to identify the faults and failures that become the sever cause of failure of the system. There are so many algorithms are available to handle that problem, but most of them are not appropriate for large scale distributed systems. The machine learning model can manage such kind of problem easily and appropriately.

The Cloud computing system "Calheiros et al. (2009) suggested the cloud platform" has become the most versatile system under the umbrella of a distributed system. Cloud-centric applications are multifaceted multi-component software which can exhibit rich and complex behaviors [1-2]. Distributed systems are kinds of software system which exchange bits and bytes among various computing nodes [3, 5]. It provides infrastructures and services to the cloud users and both small and large business enterprises. The Cloud system is playing a very important role in our society in terms of sharing fundamental computing resources. Reliability and availability must be the prime priority of the cloud system. To achieve this objective the cloud system must embody the fault-tolerant infrastructure in its core system. Adopting a fault tolerating sub-system, "Kochar et al. (2017)", in a cloud environment allows the cloud system to function its targeted operations smoothly, even at a low-level efficiency "

Programs running on a centralized uniprocessor system are capable of tolerating faults due to the existence of many powerful solutions [6-8]. In contrast, programs running on a distributed computing environment with multiple multi-core processors face the greater challenges of faults and failures. Fault tolerance can be categorized as proactive fault management and reactive fault management, "Patil et al. (2011)". This paper gives a survey of pieces of work performed on the fault tolerance mechanism in a distributed system with a focus on the machine learning-based approach, "Hazan (2016), He et al. (2016)".

II. ANALYSIS OF DISTRIBUTED SYSTEM

Distributed systems may be homogeneous, or heterogeneous like Grid and Cloud. Several shortcomings occur in such types of systems, like the quality of service, resource selection, load balancing, and fault tolerance. Fault tolerance is a major concern concerning the design of distributed systems "Engelmann et al. (2009), Kakade et al. (2012) highlighted the design of distributed systems". Whenever the failures occur in the software system, it causes a partial or an entire breakdown in the operational system and we refer it, as a fault [9-10]. To allow the system to execute its functionalities, even in the occurrence of these faults, some sophisticated techniques must be implemented to tolerate the faults "Swartz et al. (2014), Zinkevich (2003)". The objective of these techniques is to detect, identify, and correct the errors. This paper introduces an overview of the basic framework of distributed systems and their associated failure types "Hatcher et al. (2018), Chen et al. (2016)". Java offers more options to realize distributed applications.

International Journal of Engineering Applied Sciences and Technology, 2020
Vol. 5, Issue 4, ISSN No. 2455-2143, Pages 610-615
Published Online August 2020 in IJEAST (http://www.ijeast.com)

From the Java perspective, the bottom layer is represented by sockets. A socket facilitates transmitting un-interpreted data streams from one computer system to another. All others build on this mechanism. Java's java.net package provides the infrastructure needed for the direct use of sockets. From the programmer's point of view, another abstraction is more appropriate: sending messages to remote objects. This technique is called Remote Method Invocation (RMI) and is a part of Java's java.rmi package. Despite this, RMI is limited to Java1 and you have to know the location of a remote object or the registry's location.
The Jini(Java Intelligent Networking Infrastructure) manifests a basic structure which provides, register, and obtains distributed services associated with its specification. A Jini system has of the following parts:

- A set of components that provides the basic infrastructure for federating services in a distributed system
- A set of programming model that enhances the production of reliable distributed services

The Jini technology infrastructure is centric to Java technology. The Jini sub-system of Java gains its accessibility by considering that the Java programming language is the language for potential components.

III. Overview of Cloud Computing

The Cloud indicates to a Network or the Internet. In other words, the Cloud computing hierarchy provides various services over private and public networks, i.e., LAN, WAN, MAN, or VPN. Applications such as e-mail, customer relationship management (CRM), and web conferencing execute on a cloud platform. Cloud computing provides platform independence, as the software is not required to be installed locally on the PC at users' end. Now the Cloud system is creating and boosting our business applications [11-12]. Cloud Computing refers to organizing, manipulating, configuring, and accessing the software and hardware resources remotely. It offers on-demand online data storage, infrastructure, and software services "Yuan et al. (2015), Zhu et al. (2017) and Ujjwalkarm (2016)".

Cloud computing is extended under the scaling of distributed computing. The Cloud system, "Nielsen (2018), Blanchard et al. (2017) and Li et al. (2014) presented scaling distributed machine", offers computing resources, infrastructure, and services. Various technologies are available to contribute to Cloud Computing. Some of the state-of-the-art techniques are:

- Virtualization technology: Virtualization refers to executing multiple virtual computers or virtual machines into a single physical machine. Cloud virtualization is a technique for creating a virtual platform for an operating system, storage, network, data, and server "McMahan et al. (2017)". Virtual machine techniques, such as VMware, and AWS offer virtualized computational infrastructures on demand [13-16]. Virtualization "Shaw et al. (2017), Singh et al. (2003) highlighted the virtualization", is the basic framework for cloud computing.
- Coordination of cloud nodes: For smooth functioning of Cloud Computing there must be proper coordination among the various computing nodes "Bokhari et al. (2016) attempt to address cloud computing services". In the cloud system, every small cloud also known as cloudlet shares computing recourses. These cloudlets run various virtual machines. Therefore coordination and synchronization must be implemented among these cloudlets for the smooth functioning of cloud computing "Chen et al. (2012), Dong et al. (2011)".
- Web service: Computing Cloud services are normally exposed as Web services, which follow the industry standards such as WSDL (Web Services Description Language), SOAP (Simple Object Access Protocol), and UDDI (Universal Description, Discovery, and Integration). WSDL is a protocol for exchanging or sharing information in a distributed computing environment [17-20]. It is an XML-based language. SOAP is a protocol for exchanging information over the Internet. UDDI protocol is applicable for publishing the network-oriented software components. Amazon Web Services (AWS) "Garzon et al. (2008) proposed network based process", is a form of web services offers various IT services in the global market. AWS technology is constructed via server clusters spread all over the world.

IV. FAULT-TOLERANT APPROACHES IN CLOUD COMPUTING

The objective of creating a fault-tolerant system is to prevent faults arising from a single point of failure, ensuring the high availability and business continuity.

Cloud computing offers numerous services and various computing resources via the internet "Gomez et al. (2006)". On the service provider's side, a data center (DC) provides facility to keep computer systems as well as their associated components, like networking, storage, uninterruptible power supply, etc.

Primary Backup Replication (PBR):

Primary backup applies several replications to enhance system reliability. Active replication does not assign any replica as the primary replica, so it removes the centralized control of primary backup. All replicas receive the system's activation and then reply to the result. So it sustains a high cost for keeping all replicas synchronized. The fault-tolerant controlling system generally replicates the constituent components to recover from the failure "Guo et al. (2008)". Primary-backup replication protocols are very common in distributed computing [21-22].

Message Logging:

Message logging protocol is used for building a fault tolerating system. The message logging scheme is applicable in the model of message passing distributed system. This policy registers custom messages. Most users exploit the message logging facility because of its usefulness for analyzing network simulations [23]. In this scheme, each message received by a process must be recorded in the message log and the process's state is saved as a checkpoint "Jialei et al. (2016)". The logged messages are saved properly to recover the system from faults or failures. In high-performance computing (HPC) every process logs all the messages sent to any other process. It creates potential storage overhead. This scheme works as: A request is forwarded to the API then the messages are registered. After that, the API response is returned and finally, the message appears on the application log "Kalyani et al. (2016)". There are two kinds of message passing protocol:

- Pessimistic Message Logging Protocol
- Optimistic Message Logging Protocol

A pessimistic message logging scheme is the synchronous event logging scheme. In the pessimistic message logging protocol, each message is recorded in the machine's local memory [41].

An optimistic message logging system guarantees to obtain the recoverable system state. However, it has a drawback that it is less efficient than a pessimistic logging scheme.

Scheduling:

Scheduling is a decision-making process that a distributed system incorporates to determine the execution order of the available resources [23-26]. Scheduling is important for managing incoming task requests and determining which task to execute next. Scheduling is also one of the techniques to tolerate fault in a distributed system "Lebiednik et al (2016)". It is used to reduce the drawback of check-pointing in a distributed environment. It is categorized as time-sharing scheduling and space-sharing scheduling. There are three approaches to scheduling such as space, time, and hybrid [27-30].

Check-Pointing:

The Checkpointing technique provides fault tolerance for a distributed computing system. It saves a snapshot of the application program state, therefore application can resume from the point where the fault occurred. Checkpoints must be coordinated for recovery from the faults and obtaining optimal stable storage requirements "Li et al. (2015)".
There are two kinds of checkpointing:

- Coordinated
- Uncoordinated

In a coordinated checkpointing scheme, the process must confirm that their checkpoints are consistent. It is achieved by two-phase commit protocol algorithms. It has two advantages: 1. Recovery is simple and 2. Garbage collection is easy. It has some major drawbacks: 1. It is expensive due to energy consumption 2. All processes are competing for writing their checkpoints at the same point.

In the uncoordinated checkpointing protocol, there is no requirement for synchronization between the processes at checkpoint time [31]. It has some major drawbacks: 1. If no checkpoint forms a global state, the application has to resume from the starting of the event of a failure. 2. The recovery cost is not acceptable and 3. Garbage collection becomes more complex to implement.

K-Modular Redundancy (KMR):

KMR is a widely used fault tolerance mechanism in software engineering. KMR is a kind of version programming, hence it is also known as N-Version Programming (NVR) [32-36]. It is based on the principle of function ranking. Higher K^{th} significant functions are recognized and selected for invocation. This strategy performs parallel executions that are functionally equivalent and then take priority voting to calculate the final output. Triple Modular Redundancy (TMR), a kind of KMR, is a fault tolerance form. In which three subsystems execute a process and the final result is obtained by the majority voting subsystem [37-40].

The advantage of this system is if any one of the three subsystems fails, the other two subsystems can correct the error and mask or remove the faults. TMR system contains three similar logic circuits to compute the basic Boolean function. The output is obtained by combining the three intermediate results by using another logic circuit [41-43]. The concept of TMR can apply to many forms of redundancy which are found in many fault-tolerant computer systems "Patidar et al. (2011)". The TMR is used in space satellite systems.

V. CONCLUSION

The objective of fault tolerating a distributed system is to make a distributed system capable of defending against the faults and failures. Fault tolerance strategies are very crucial in the distributed system, especially cloud-centric applications. In large scale distributed system failures lead to the collapse of the entire system. A fault may occur at any constituent computational node or machine. This becomes the cause of a partial breakdown in the system therefore the throughput and performance of the system degrade severely.

A plethora of research has been going on the direction of the fault-tolerant system. Recently machine learning especially deep learning is emerged as a promising approach to enhance fault tolerance in the distributed system. By principle, a deep learning approach incorporates multiple processing units to handle voluminous heterogeneous computing resources scattered over distinct geographical locations. A distributed deep machine learning algorithm has become a promising approach to implement fault-tolerant systems.

VI. ACKNOWLEDGMENTS

I would like to express my deep gratitude to Dr. Akhilesh A. Waoo, Head of the Department, AKS University Satna, and my research supervisor, for their patient guidance, enthusiastic encouragement, and useful suggestions of my entire research work. I would also like to thank Professor Dr. Rakesh Kumar Katare, Dr. Navita Shrivastava, APS University Rewa, for their advice and assistance in keeping my progress in the right direction. This research paper would not have been possible without the exceptional assistant of my fellow Ms. Sonali Singh. My special thanks to the academic and technical staff of AKS University and RGCCAT Satna, for their encouragement and valuable suggestions.

Finally, I wish to thank my parents and brothers for their continuous support and encouragement throughout my research.

VII. REFERENCE

[1] Calheiros, R.N., Ranjan, R., De Rose, C.A.F., Buyya, R. (2009). CloudSim: A Novel Framework for Model and Simulation of Cloud Computing Infrastructures and Services, (pp. 1-9).

[2] Kocher, D., Hilda, A.K.J. (2017). An approach for faults tolerance in cloud computing using machine learning technique. Int. J. Pure Appl. Math. 117(22), (pp. 345-351).

[3] Bekkerman, R., Bilenko, M., and Langford, J. (2011). Scaling up machine learning: Parallel and distributed approaches. Cambridge University Press.

[4] Bernstein, J., Xiang Wang, Y., Azizzadenesheli, K. and Anandkumar, A. (2018). Signsgd: Compressed optimization for non-convex problems. In International Conference on Machine Learning, (pp. 559-568).

[5] Bijral, A. S., Sarwate, Anand D., and Srebro N. (2016). On data dependence in distributed stochastic optimization. arXiv preprint arXiv:1603.04379.

[6] Chaturapruek, S., John, C. D. and C. R´e, C. (2015). Asynchronous stochastic convex optimization: the noise is in the noise and sgd don't care. In Advances in Neural Information Processing Systems, (pp. 1531-1539).

[7] Patil, A., Shah, A., Gaikwad, S., Mishra, A.a., Kohli, S.S., Dhage, S. (2011). Fault Tolerance in Cluster Computing System. In: 2011 Int. Conf. P2P, Parallel, Grid, Cloud Internet Comput., (pp. 408-412).

[8] Hazan, E. Introduction to online convex optimization. Foundations and Trends in Optimization (2016). 2(3-4): (pp. 157–325).

[9] He, K., Zhang X., Ren, S. and Jian S., Deep residual learning for image recognition. (2016). In Proceedings of the IEEE conference on computer vision and pattern recognition, (pp 770-778).

[10] Engelmann, C., Vallée, G.R., Naughton, T., Scott, S.L. (2009). Proactive fault tolerance using preemptive migration. In: Proc. 17th Euromicro Int. Conf. Parallel, Distrib. Network-Based Process. PDP 2009, (pp. 252-257).

[11] Kakade, S. M., Shwartz, S. S. and Tewari, A. (2012). Regularization techniques for learning with matrices. ournal of Machine Learning Research, 13(Jun):1865-1890.

[12] Shwartz, S. S. and David, S. B. (2014). Understanding machine learning: From theory to algorithms, Cambridge University press.

[13] Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. In International Conference on Machine Learning, (pp. 928-936).

[14] HATCHER, W. G., and YUA, W. (2018). Survey of Deep Learning: Platforms, Applications and Emerging Research Trends, IEEE Access, May 24.

[15] Chen X.W. and Lin X. (2016). Big data deep learning: Challenges and perspectives, IEEE Access, vol. 2, 2014. 14. Y. Ding, S. Chen, and J. Xu, Application of deep belief networks for opcode based, (pp. 514-525)

[16] Malware detection, in Proc. Int. Joint Conf. Neural Netw. (IJCNN), (pp. 3901-3908).

[17] Yuan, Y. and Jia K. (2015). A distributed anomaly detection method of operation energy consumption using smart meter data, in Proc. Int. Conf. Intell. Inf. Hiding Multimedia Signal Process. (IIH-MSP), (pp. 310-313).

[18] Zhu, D., Jin, H., Y, Y., Wu, D. and Chen, W. (2017). DeepFlow: Deep learning based malware detection by mining Android application for abnormal usage of

sensitive data, in Proc. IEEE Symp. Comput. Commun. (ISCC), (pp. 438-443).

[19] Nielsen, M. (2018). Neural Networks and Deep Learning. [Online]. Available: 19. Dan Alistarh, Zeyuan Allen-Zhu, and Jerry Li.

[20] Byzantine stochastic gradient descent. In Advances in Neural Information Processing Systems, (pp. 4613-4623).

[21] Blanchard, P., Guerraoui, R., and Stainer, J. (2017). Machine learning with adversaries: Byzantine tolerant gradient descent. In Advances in Neural Information Processing Systems, (pp. 119-129).

[22] Li, M., G., D., Andersen, Park, J. W., Smola, A. J., Ahmed, Josifovski, A., Long, J., J., I., Shekita, and Yiing B. Su. (2014). Scaling distributed machine learning with the parameter server. In proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation, OSDI'14, Berkeley, CA, USA, 2014. USENIX Association, (pp. 583-598)

[23] McMahan B. and Ramage, R. (2017). Federated learning: Collaborative machine learning without centralized training data. Google Research Blog, 3.

[24] Alistarh, D., Allen-Zhu, Z., Li, J. (2018): Byzantine stochastic gradient descent. In: Advances in Neural Information Processing Systems, (pp. 4613-4623).

[25] Blanchard, P., Guerraoui, R., Stainer, J., et al. (2017): Machine learning with adversaries: byzantine tolerant gradient descent. In: Advances in Neural Information Processing Systems, (pp. 119-129).

[26] Chen, L., Wang, H., and Papailiopoulos, D. (2016): Draco: robust distributed training against adversaries. In: 2nd SysML Conference. Obermeyer, Z., and Emanuel, E. J.: Predicting the future-big data, machine learning, and clinical medicine. New Engl. J. Med. 375(13), 1216.

[27] Wang, X., Han, Y., Wang, C., Zhao, Q., Chen, X., Chen, M., 2019: In-edge AI: intelligentizing mobile edge computing, caching and communication by federated learning. IEEE Network 33(5), (pp. 156-165).

[28] Yu, H., Yang, S., Zhu, S. (2019): Parallel restarted SGD with faster convergence and less communication: demystifying why model averaging works for deep learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, (pp. 5693-5700).

[29] M. Deisenroth, D. Fox, and C. Rasmussen. (2015). Gaussian processes for data-efficient learning in robotics and control. IEEE Transactions on Pattern Analysis & Machine Intelligence, 37(2): (pp. 408-423).

[30] Pandey, S., Wu, L., Guru, S.M., Buyya, R. (2010). A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In: 24th IEEE International Conference on Advanced Information Networking and Applications, (pp. 400-407).

[31] Prathiba, S., Sowvarnica, S. (2017). Survey of failures and fault tolerance in cloud. In: Proc. 2017 2nd Int. Conf. Comput. Commun. Technol. ICCCT 2017, (pp. 169-172).

[32] Rimal, B.P., Choi, E., Lumb, I. (2009). A taxonomy and survey of cloud computing systems. In: NCM 2009 - 5th Int. Jt. Conf. INC, IMS, IDC, (pp. 44-51).

[33] Savu, L. (2011). Cloud computing: deployment models, delivery models, risks and research challenges. In: 2011 Int. Conf. Comput. Manag. CAMAN 2011.

[34] Shaw, R., Howley, E., Barrett, E. (2017). An Advanced Reinforcement Learning Approach for Energy-Aware Virtual Machine Consolidation in Cloud Data Centers, (pp. 61-66).

[35] Shwe, T., Aye, W. (2008). A fault tolerant approach in cluster computing system. In: 5th Int. Conf. Electr. Eng. Comput. Telecommun. Inf. Technol. ECTI-CON 2008, vol. 1, (pp. 149-152).

[36] Singh, P., Cabillic, G. (2003). A Checkpointing Algorithm for Mobile Computing Environment, (pp. 65–74).

[37] Agbaria, A., Sanders, W.H. (2004). Distributed snapshots for mobile computing systems. In: Proc. - Second IEEE Annu. Conf. Pervasive Comput. Commun. PerCom, no. Cic, (pp. 151-186).

[38] Alomari, E., Manickam, S., Gupta, B.B., Karuppayah, S., Alfaris, R. (2012). Botnet-based distributed denial of service (DDoS) attacks on web servers: classification and art. arXiv preprint arXiv:1208.0403.

[39] Ataallah, S.M.A., Nassar, S.M., Hemayed, E.E. (2015). Fault tolerance in cloud computing Survey. In: 11th Int. Comput. Eng. Conf., no. 1, (pp. 241-245).

[40] Bokhari, M.U., Shallal, Q.M., Tamandani, Y.K. (2016). Cloud computing service models: a comparative study. In: IEEE Int. Conf. Comput. Sustain. Glob. Dev. INDIACom, (pp. 16-18).

[41] Chen, W., Deelman, E. (2012). WorkflowSim: A toolkit for simulating scientific workflows in distributed environments. In: 2012 IEEE 8th Int. Conf. E-Science, e-Science 2012.

[42] Chen, Z., Son, S.W., Hendrix, W., Agrawal, A., Liao, W.K., Choudhary, A. (2014). NUMARCK: machine learning algorithm for resiliency and check pointing. In: Int. Conf. High Perform. Comput. Networking, Storage Anal. SC, vol. 2015-Janua, no. January, (pp. 733-744).

[43] Dong, Z., Rojas-Cessa, R. (2011). Non-blocking memory-memory-memory Close network packet switch. In: 2011 34th IEEE Sarnoff Symp. SARNOFF 2011, (pp. 1-5).

[44] Frincu, M.E., Craciun, C. (2011). Multi-objective meta-heuristics for scheduling applications with high availability requirements and cost constraints in multicloud environments. In: Proc.- 2011 4th IEEE Int. Conf. Util. Cloud Comput. UCC 2011, (pp. 267-274).

[45] Garzon, D.B., Gomez, C., Gomez, M.E., Lopez, P., Duato, J. (2014). FT-RUFT: a performance and fault-tolerant efficient indirect topology. In: Proc. - 2014 22nd Euromicro Int. Conf. Parallel, Distrib. Network-Based Process. PDP 2014, (pp. 405-409).

[46] Gomez, C., Gomez, M., Lopez, P., Duato, J. (2006). A Dynamic and Compact Fault Tolerant Strategy for Fat-tree. In: Proc. IFIP Int'l Conf. Netw. Parallel Comput., no. January.

[47] Gomez, C., Gilabert, F., Gomez, M.E., Lopez, P., Duato, J. (2008). RUFT: Simplifying the fat-tree topology. In: Proc. Int. Conf. Parallel Distrib. Syst. - ICPADS, (pp. 153-160).

[48] Greenberg, A. et al. (2009). VL2: A Scalable and Flexible Data Center Network. In: ACM SIGCOMM Conf. Data Commun. (pp. 51-62).

[49] Guo, C., Wu, H., Tan, K., Shi, L., Zhang, Y., Lu, S. (2008). DCell: a scalable and fault tolerant network structure for data centers. In: Proc. ACM SIGCOMM 2008 Conf. Data Commun. - SIGCOMM '08, (pp. 75-86).

[50] Guo, C., Lu, G., Li, D., Wu, H., Zhang, X. (2009). BCube: a high performance, server centric network architecture for modular data centers. In: SIGCOMM '09 Proc. ACM SIGCOMM 2009 Conf. Data Commun., (pp. 63-74).

[51] Jialei Liu, I., Wang, Shangguang, Senior Member, IEEE, Zhou, Ao, Kumar, Sathish A.P., Yang, Fangchun, Senior Member, IEEE, Buyya, Rajkumar, Fellow. (2016). Using proactive fault-tolerance approach to enhance cloud service reliability. IEEE Trans. Cloud Comput., (pp. 1-1).

[52] Kalyani, Z., Amune, A., Chas, M., Chas, M. (2016). Review on Secure Distributed Deduplication Systems with Improve, 5(1).

[53] Lebiednik, B., Mangal, A., Tiwari, N. (2016). A survey and evaluation of data center network topologies, 1–12. arXiv preprint arXiv:1605.01701.

[54] Li, J., Chen, X., Huang, X., Tang, S., Xiang, Y., Member, S. (2015). Secure Distributed Deduplication Systems with Improved Reliability, 64(12), (pp. 3569-3579).

[55] Patidar, S., Rane, D., Jain, P. (2011). A survey paper on cloud computing. In: Proc. - 2012 2nd Int. Conf. Adv. Comput. Commun. Technol. ACCT 2012, (pp. 394-398).