Simulation-based Network Fault Injection in the CloudSim Plus Cloud Simulation Environment

Farida Asadova[1], Gabor Kertesz[1], Robert Lovas[2]

[1]Obuda University, Budapest, Hungary, asadovafarida@stud.uni-obuda.hu, kertesz.gabor@nik.uni-obuda.hu
[2]Eotvos Lorand Research Network (ELKH), Budapest, Hungary, robert.lovas@sztaki.hu

*Correspondence: Farida Asadova,Obuda University, Budapest, Hungary, asadovafarida@stud.uni-obuda.hu

Abstract

One effective method for assessing the dependability of computer systems is fault injection. This deliberate technique introduces faults into a system to assess its resilience and ability to handle abnormal conditions. Therefore, this study investigates and simulates the different network problems in the CloudSim Plus environment. CloudSim Plus is a simulation framework that enables the modeling and simulation of cloud computing environments, allowing researchers and practitioners to evaluate the performance and behavior of cloud-based systems and algorithms. Network fault detection and its management are essential duties in cloud systems. Moreover, the feasibility of manual monitoring and involvement has decreased as these infrastructures expand and change. This paper briefly introduces network problems and fault injection outcomes in CloudSim Plus nodes.

Keyword: CloudSim Plus, Cloud Systems, Network Faults, Fault Injection, Cloud Simulation.

1. Introduction

The primary objective of this research is to systematically identify network faults within the cloud environment, comprehensively analyze their principal functionalities, and subsequently introduce them into the (CloudSim Plus, 2022) environment. This study examines the procedural steps in injecting these faults into the cloud environment while concurrently collecting relevant data for statistical analysis.

A fault represents an anomaly in the system that causes it to behave unpredictably and unexpectedly. In a cloud environment, various types of network faults (Hsueh, M. C., Tsai, T. K., & Iyer, R. K., 1997) can occur. The most common network issues are listed:

1) Resource Allocation Faults (Mohan, N. R., & Raj, E. B., 2012, November)- Three resource allocation faults are listed.

a) Network Equipment Faults (Vishwanath, K. V., & Nagappan, N., 2010, June) - Network problems may arise within network hardware and devices such as switches, routers, firewalls, and wifi access points. These issues can stem from various factors, including improper settings, defective connections, and the occurrence of packet loss.

b) Extensive CPU Load (Mason, K., Duggan, M., Barrett, E., Duggan, J., & Howley, E., 2018) - This is caused by the cloud network becoming congested with a lot of traffic. CPU consumption may climb significantly when processes take longer to complete or when more network packets are delivered and received across the cloud network. High CPU utilization can slow down the network or leave insufficient CPU for other tasks, c) Extensive Bandwidth Load (Yu, R., Xue, G., Zhang, X., & Li, D. (2017, May) - Cloud network becomes congested when someone or something starts using all available capacity to download terabytes of data, possibly video. Users may start encountering issues like poor internet download speeds when the network becomes congested owing to high bandwidth utilization, leaving insufficient bandwidth for other network sections.

2) Faulty Cables or Connectors - Faults might be produced on the network equipment connected to by hardware issues like faulty cables or connectors. The quantity of data that can pass through a damaged copper, cable, or fiber-optic cable without packet loss will likely be reduced. The manuscript by Dantas, M. S. M., et al. (2022) examines missing or unplugged connectors and detects them in classification.

3) Equipment Operation - When hardware or devices are not operating as intended due to incorrect configuration or disabling, this could affect the network's performance.

4) Domain Name System (DNS) Faults - DNS faults generally occur when users cannot connect to an IP address, which is a symptom that they could not have network or internet connectivity anymore. Consequently, the application may concurrently appear online internally while offline to users.

5) Wireless Network Interference - Wireless interference occurs when something disrupts or weakens the wifi signal transmitted by your wireless router. This is especially true for 2.4GHz wireless routers.

The first obstacle for engineers is immediately identifying the events that can lead to breakdowns and the specific time, given that a network outage or failure can occur. Although consumers are usually prompt in reporting issues, it is obviously better to catch the issue early and fix it before it negatively impacts users.

Fault injection is a broad and extensively researched area encompassing various implementation aspects. Hsueh et al. (1997) describe the concept of fault as a physical defect, imperfection, or flaw that manifests within hardware or software components. Engineers employ fault injection techniques to assess the resilience and reliability of fault-tolerant systems or components. Fault injection is the validation technique of the dependability of fault tolerant systems, which consists of the accomplishment of controlled experiments where the observation of the system's behavior in the presence of faults is induced explicitly by the writing introduction of faults in the system. The fault injection techniques have been recognized for a long time as necessary to validate the dependability of a system by analyzing the behavior of the devices when a fault occurs.

According to Ziade, H., Ayoubi, R. A., & Velazco, R. (2004), there are numerous fault injection techniques as hardware-based fault injection, software-based fault injection, simulation-based, emulation-based, and hybrid fault injection methods. This study tested injection techniques based on the simulation-based fault injection method. As an experimental setup, CloudSim Plus (2022) simulation environment has been used to inject collected network faults.

An innovative, generalized, and extendable simulation framework called CloudSim Plus makes it possible to model, simulate, and experiment with new Cloud computing infrastructures and application services (Silva Filho, et al., 2017, May). While CloudSim Plus builds upon the foundation of CloudSim (Buyya, R., Ranjan, R., & Calheiros, R. N., 2009, June), it introduces significant improvements, modularity, and new features, making it a preferred choice for researchers and developers seeking more advanced cloud simulation capabilities.

The stability and management of cloud systems have become increasingly important as cloud technologies advance quickly and more applications are moved to cloud environments. Even if cloud monitoring gets less attention, it is essential to conduct precise and ongoing monitoring efforts to spot errors and correctly run cloud processes. Cloud monitoring aids in reviewing, keeping track of, and managing the intricate operations of cloud infrastructure, as noted by Aceto, G., Botta, A., De Donato, W., & Pescapè, A. (2012, November).

2. Related Research

Numerous articles examine fault injection from a variety of perspectives. Gulenko, A., Wallschläger, M., Schmidt, F., Kao, O., & Liu, F. (2016, December) research paper investigates several anomalies in the experimental setup built in the open-source cloud computing system OpenStack. The results were evaluated by investigating multiple machine learning algorithms. This manuscript chooses seven common faults, including memory leaks, excessive CPU usage, disk write, package loss, increased latency, throttle bandwidth, and bandwidth usage. Some of these faults can be classified as network faults.

Fault injection is a wide area of research; therefore, many survey manuscripts are written to classify and clarify fault injection methodologies. In the survey paper, Ziade, H., Ayoubi, R. A., & Velazco, R. (2004), not only clarify the various types of fault injection methods and define their respective advantages and disadvantages but also categorize the concept of fault into hardware and software faults, with hardware faults encompassing permanent, transient, and intermittent faults. In contrast, software faults include function, algorithm, timing, checking, and assignment faults. Another survey paper by Kooli, M., & Di Natale, G. (2014, May) defines fault injection methodologies for fault-tolerant systems. Maxion, R. A., & Olszewski, R. T. (1993, June) use the experimental setup methodology on network fault injection in a campus network. The general experimenting concept could be applied to anomaly detection in the cloud.

The research written by Nita, M. C., Pop, F., Mocanu, M., & Cristea, V. (2014) presents the FIM-SIM module for the CloudSim environment. This module is helpful for developers to test and evaluate the cloud environment. The authors have developed a run-time, event-driven fault injection module for cloud simulation. At a randomly selected time frame, it will generate an event and simulate a failure in the cloud system. The interaction between the Broker and the Datacenter involves the transmission of cloudlets, followed by scheduling based on a designated Scheduling Policy. Within the CloudSim framework, entities can exchange specific events. Specifically, the Fault Injector module sends failure notifications to the Datacenter. Notably, the fault injector module utilizes statistical distribution (both discrete and continuous) to generate events. It operates as a continuous thread throughout the simulation period, attempting to introduce faults based on a statistical method for generating random numbers.

Continuous improvements in cloud simulation environments have led to the developing of a new version of the CloudSim framework, known as CloudSim Plus. In the paper, Silva Filho, et al. (2017, May) discuss the enhanced functionalities and improvements made to the existing classes within CloudSim. The introduction of CloudSim Plus provides a more advanced and flexible architecture, allowing for greater customization and development of cloud simulations. In the CloudSim Plus simulation environment, Malik, M. K. (2020) have conducted research based on host fault injection. The simulation of host injection used in this study was tested using several distribution techniques. Analysis and evaluation of the faults were done appropriately. Similar to this, available bandwidth in the simulation data centers has been examined in the article Bosilca, A., Nita, M. C., Pop, F., & Cristea, V. (2014, September). Another research work describes (Zhang, H., Dong, F., Shen, D., Xiong, R., & Jin, J., 2017, April) a Virtual network fault diagnosis mechanism based on fault injection.

3. Experimental Setup in Cloudsim

CloudSim Plus, an open-source framework, is widely utilized to simulate cloud computing services and infrastructure. It is a powerful tool for modeling and simulating cloud computing environments, enabling researchers to assess hypotheses and replicate tests before software development. The adoption of CloudSim Plus brings forth numerous advantages and benefits across various dimensions of cloud computing research and development.

- Utilizing a simulation tool like CloudSim Plus entails no capital investment, as it involves neither installation expenses nor maintenance costs.
- CloudSim Plus offers ease of use and scalability, allowing users to modify resource requirements effortlessly by making minor changes to a few lines of code.
- Using simulation in cloud computing enables the evaluation of risks at an earlier stage, circumventing the limitations posed by real testbeds, which restrict experiments to the scale of the test environment and impede result reproducibility.

• CloudSim Plus eliminates the necessity for trial-and-error approaches, replacing them with a repeatable and controlled environment that allows for testing services without incurring costs. This alternative obviates the reliance on theoretical and imprecise evaluations, which can result in suboptimal service performance and revenue generation.

The experimental setup has been built for fault injection simulation and faulty data collection. Figure 1 represents the architecture of the simulation environment. This architecture comprises one Datacenter, ten hosts, ten Virtual Machines, and ten Cloudlets.

1) Datacenter - used for modeling the foundational hardware equipment of the cloud environment. This class provides methods to specify the functional requirements of the Datacenter as well as methods to set the allocation policies of the VMs.

2) DatacenterBroker - an entity acting on behalf of the user or customer. It is responsible for functions of VMs, including VM creation, management, destruction, and submission of cloudlets to the Virtual Machine(VM).

3) Host - executes actions related to the management of virtual machines. It also defines policies for provisioning memory and bandwidth to the virtual machines and allocating CPU cores to the virtual machines.

4) VM - represents a virtual machine by providing data members defining a VM's bandwidth, RAM, and size while providing setter and getter methods for these parameters.

5) Cloudlet - represents any task run on a VM, like a processing task, a memory access task, a file updating task, etc. This class stores parameters defining the characteristics of a task, such as its length and size. It provides methods similar to the VM class while also providing methods that define a task's execution time, status, cost, and history.

CloudSim Core Simulation Engine provides interfaces for managing resources such as VM, memory, and bandwidth of virtualized Datacenters.

CloudSim layer manages the creation and execution of core entities such as VMs, Cloudlets, Hosts, etc. It also handles network-related execution along with the provisioning of resources and their execution and management.

User Code is the layer controlled by the user. The developer can write the requirements of the hardware specifications in this layer according to the scenario.

CloudSim uses a Virtual machine allocation policy in VMs distribution to resolve resource allocation drawbacks (Iyengar, N. C. S., 2015).

4. Injected Faults

CloudSim Plus simulation tool contains the class HostFault- Injection, which generates random failures for the Processing elements(PE) (Interface Pe, 2023) of Hosts inside a given Datacenter. A Fault Injection object usually has to be created after the VMs are created to make it easier to define a function to clone failed VMs. The events happen in the following order:

1) Random Number Generator creates time to inject a Host failure.

2) A Host is randomly selected to fail at that time using an internal Uniform Random Number Generator with the same seed as the given generator;

3) The number of Host PEs to fail is randomly generated using the internal generator;

4) failed physical PEs are removed from affected VMs, VMs with no remaining PEs and destroying and clones of them are submitted to the DatacenterBroker of the failed VMs;

5) Another failure is scheduled for the next time using the given generator;

6) the process repeats until the end of the simulation.

7) When Host's PEs fail, if there are more available PEs than those required by its running VMs, no VM will be affected.

They consider X as the number of failed PEs, lower than the total available PEs. In this case, the X PEs will be removed cyclically, one by 1, from running VMs. In this way, some VMs may continue execution with fewer PEs than they requested initially. On the other hand, if after the failure, the number of Host working PEs is lower than the required to run all VMs, some VMs will be destroyed.

If all PEs are removed from a VM, it is automatically destroyed, and a snapshot (clone) from it is taken and submitted to the Broker so the clone can start executing into another host. In this case, all the cloudlets inside the VM will be cloned to and restart executing from the beginning.

If a cloudlet running inside a VM affected by a PE failure requires Y PEs, but the VMs do not have such PEs anymore, the Cloudlet will continue executing, but it will take longer to finish. For instance, if a Cloudlet requires 2 PEs, but after the failure, the VM is left with just 1 PE, the Cloudlet will spend double the time to finish.

Host PEs failures may happen after all its VMs have finished executing. This way, the presented simulation results may show that the number of PEs in a Host is lower than its VMs require. In this case, the VMs shown in the results finished executing before some failures happened. Analyzing the logs is easy to confirm that. Failures interarrivals are defined in minutes since seconds is a too small time unit to define such value. Furthermore, defining the number of failures per second does not make sense. This way, the generator of failure arrival times given to the constructor considers the time in minutes, despite the simulation time unit being seconds. Since commonly Cloudlets take some seconds to finish, mainly in simulation examples, failures may happen just after the cloudlets have finished. This way, one should ensure that Cloudlets' lengths are large enough to allow failures to happen before the end.

Certain network faults cannot be injected into the cloud simulation environment. These faults are related to the natural environment and can fail for unpredictable reasons. Network Equipment faults, Faulty Cables or Connectors, and Wireless Network Interference faults cannot be tested in the CloudSim Plus environment. Some network faults can occur due to natural circumstances. As the CloudSim Plus simulates

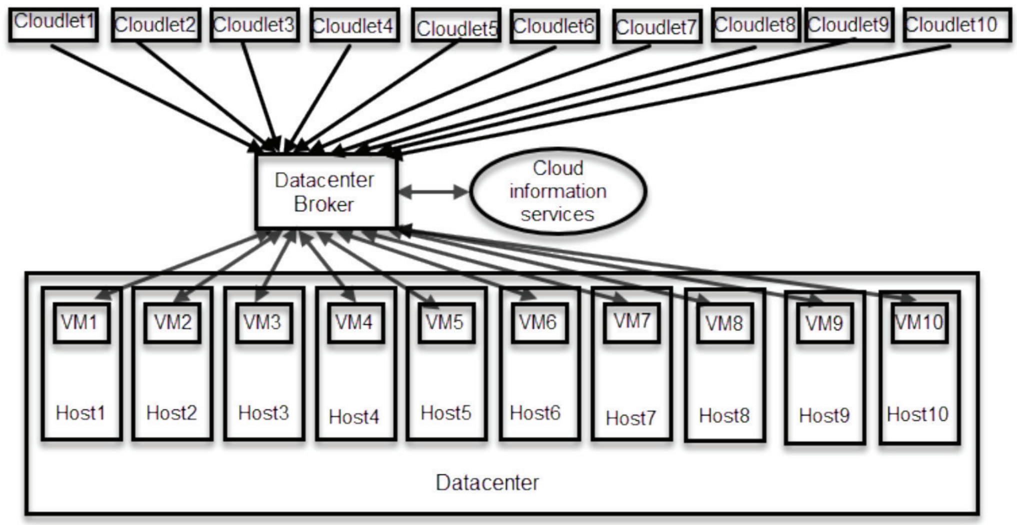


Fig. 1: Experimental Setup in CloudSim Plus

the cloud environment, it does not contain equipment, cables, or connectors.

This research mainly injects the faults classified as Extensive CPU Load, Extensive Bandwidth Load, and DNS faults into the CloudSim Plus. The faults are injected into the Hosts in the testbed.

The fault injection techniques can be grouped into invasive and noninvasive techniques. The problem with sufficiently complex systems, particularly time dependant ones, is that it may be impossible to remove the footprint of the testing mechanism from the system's behavior, independent of the fault injected.

• Invasive techniques are those which leave behind such a footprint during testing.

• NoninvasiveNoninvasive techniques can mask their presence not to affect the system other than the faults they inject.

The injection agents installed on the data center nodes perform the injections. The agents inject faults by changing the infrastructure configuration at the hypervisor level: for example, by deallocating a resource or setting the network interface's loss or corruption rate.

5. Erroneous Data Collection

Fault injection techniques yield seven benefits for erroneous data collection:

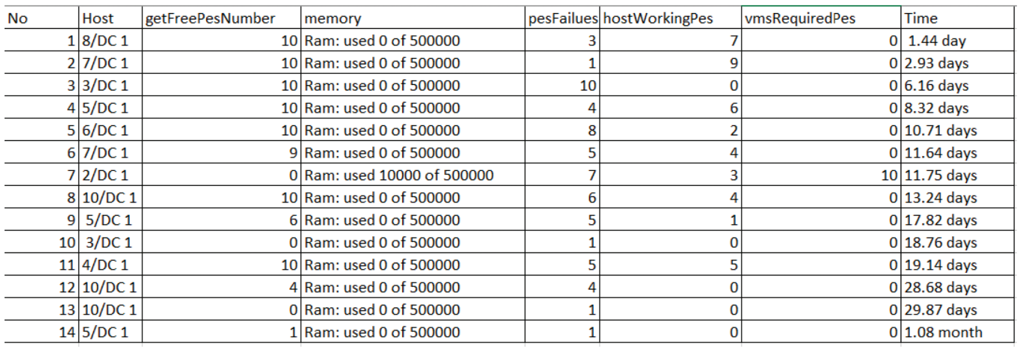
• An understanding of the effects of real faults and, thus, of the related behavior of the target system in terms of functionality and performance.

• An assessment of the efficacy of the fault tolerance mechanisms included in the target system and thus feedback for their enhancement and correction (e.g., for removing design faults in the fault tolerance mechanisms).

• They are forecasting the faulty behavior of the target system, in particular, encompassing a measurement of the efficiency (coverage) provided by the fault tolerance mechanisms.

• Estimating fault-tolerant mechanisms' failure coverage and latency (i. e timing).

• Exploring the effects of different workloads (different input profiles and environments) on the effectiveness of fault-tolerant mechanisms.



No	Host	getFreePesNumber	memory	pesFailues	hostWorkingPes	vmsRequiredPes	Time
1	8/DC 1	10	Ram: used 0 of 500000	3	7	0	1.44 day
2	7/DC 1	10	Ram: used 0 of 500000	1	9	0	2.93 days
3	3/DC 1	10	Ram: used 0 of 500000	10	0	0	6.16 days
4	5/DC 1	10	Ram: used 0 of 500000	4	6	0	8.32 days
5	6/DC 1	10	Ram: used 0 of 500000	8	2	0	10.71 days
6	7/DC 1	9	Ram: used 0 of 500000	5	4	0	11.64 days
7	2/DC 1	0	Ram: used 10000 of 500000	7	3	10	11.75 days
8	10/DC 1	10	Ram: used 0 of 500000	6	4	0	13.24 days
9	5/DC 1	6	Ram: used 0 of 500000	5	1	0	17.82 days
10	3/DC 1	0	Ram: used 0 of 500000	1	0	0	18.76 days
11	4/DC 1	10	Ram: used 0 of 500000	5	5	0	19.14 days
12	10/DC 1	4	Ram: used 0 of 500000	4	0	0	28.68 days
13	10/DC 1	0	Ram: used 0 of 500000	1	0	0	29.87 days
14	5/DC 1	1	Ram: used 0 of 500000	1	0	0	1.08 month

Fig. 2: CloudSim Plus Model simulation data overview

• Identifying weak links in the design: For example parts of the system within which a single fault could lead to severe consequences.

• Studying the system's behavior in the presence of faults, for example propagation of fault effects between system components or the degree of fault

In practice, fault removal and fault forecasting are frequently not used separately, but one follows the other. For instance, after rejecting a system by fault forecasting testing, several fault removal tests should be applied. These new tests provide actions that will help the designer to improve the system. Then, it will be applied to another fault forecasting test, and so on (Hsueh, M. C., Tsai, T. K., & Iyer, R. K., 1997).

According to the simulation of the model, the data has been collected in Figure 3. The data in this Table shows that the high number of PEs usage caused a rapid increase in memory utilization. In this table number of failed hosts is represented. Due to the failure of the PEs, VMs are removed from the Hosts. By the end of the simulation, no working VMs are left for simulation. The execution of this simulation is performed once, yielding varying results with each iteration. However, conducting a significantly larger number of simulations, preferably in the hundreds, is necessary to achieve a reliable statistical analysis. Figure 4 shows the spike on DC 2 when all hosts were unavailable, and it could no longer request the VMs. The most striking observation to emerge from the data comparison was when all hosts and PEs were in use, and there were no available memory and VM to prevent an outage.

Conclusion and Future Work

This paper investigates the general network faults and classifies them. Also, the fault detection concept is explained by referring to various publications.

In future work, utilizing the extensive dataset obtained from hundreds of simulations would facilitate the execution of statistical analyses and predictive modeling. This, in turn, would enable the expansion of fault detection and Prediction capabilities encompass a wider array of fault types, as well as their corresponding specifications, alongside the incorporation of diverse monitoring technologies.

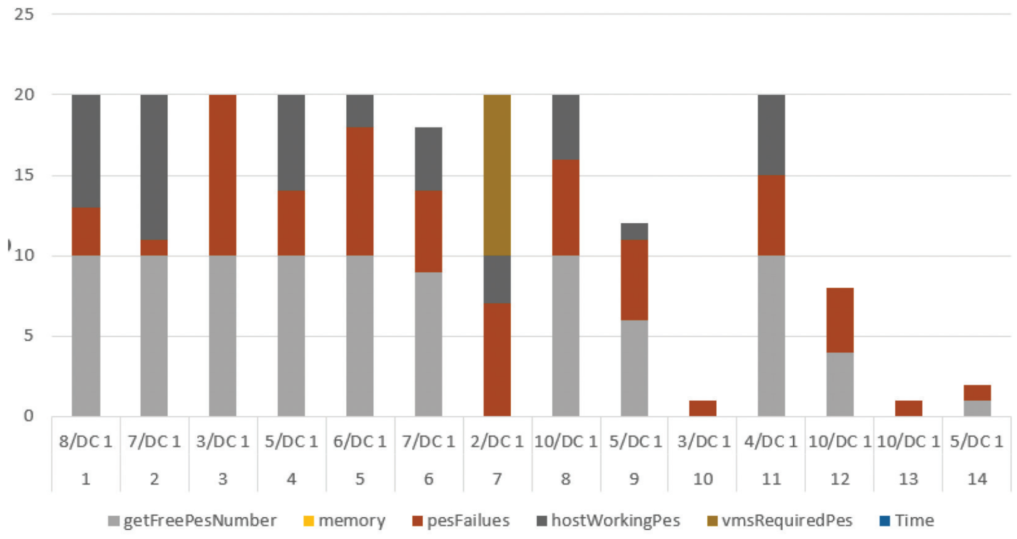


Fig. 3: Chart based on the collected data

This paper focused on the broad overview by referring to various materials and documentation. The components of the CloudSim Plus simulation environment are learned and implemented.

Acknowledgment

The authors thankfully acknowledge the support of the Doctoral School of Applied Informatics and Applied Mathematics, Obuda University. The New National Excellence Program of the Ministry For Innovation And Technology supported the presented research from the source of the National, Research, Development and Innovation Fund. We gratefully acknowledge the financial support of the Hungarian Scientific Research Fund. The Janos Bolyai Research Scholarship of the Hungarian Academy of Sciences also supported the presented work of R. Lovas.

References

Aceto, G., Botta, A., De Donato, W., & Pescapè, A. (2012, November). Cloud monitoring: Definitions, issues and future directions. In *2012 IEEE 1st International conference on cloud networking (CLOUDNET)* (pp. 63-67). IEEE.

Bosilca, A., Nita, M. C., Pop, F., & Cristea, V. (2014, September). Cloud simulation under fault constraints. In *2014 IEEE 10th International Conference on Intelligent Computer Communication and Processing (ICCP)* (pp. 341-348). IEEE.

Buyya, R., Ranjan, R., & Calheiros, R. N. (2009, June). Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities. In *2009 international conference on high performance computing & simulation* (pp. 1-11). IEEE.

CloudSim Plus (2022). https://cloudsimplus.org, [Online] Accessed 15 September 2022.

Dantas, M. S. M., et al. (2022). Faulty RJ45 connectors detection on radio base station using deep learning. *Multimedia Tools and Applications, 81*(21), 30305-30327.

Gulenko, A., Wallschläger, M., Schmidt, F., Kao, O., & Liu, F. (2016, December). Evaluating machine learning algorithms for anomaly detection in clouds. In *2016 IEEE International Conference on Big Data (Big Data)* (pp. 2716-2721). IEEE.

Hsueh, M. C., Tsai, T. K., & Iyer, R. K. (1997). Fault injection techniques and tools. *Computer, 30*(4), 75-82.

Interface Pe (2023). https://www.javadoc.io/doc/org.cloudsimplus/ cloudsim-plus/4.3.2/org/cloudbus/cloudsim/resources/Pe.html, [Online] Accessed 3 May 2023.

Iyengar, N. C. S. (2015). Virtual machine allocation policy in cloud computing using cloudsim in java. *Int J Grid Distrib Comput, 8*(1), 145-158.

Kooli, M., & Di Natale, G. (2014, May). A survey on simulation-based fault injection tools for complex systems. In *2014 9th IEEE International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS)* (pp. 1-6). IEEE.

Malik, M. K. (2020). Host fault injection using various distribution functions. *Int J Comput Sci Mob Comput, 9*(12), 1-10.

Mason, K., Duggan, M., Barrett, E., Duggan, J., & Howley, E. (2018). Predicting host CPU utilization in the cloud using evolutionary neural networks. *Future Generation Computer Systems, 86,* 162-173.

Maxion, R. A., & Olszewski, R. T. (1993, June). Detection and discrimination of injected network faults. In *FTCS-23 The Twenty-Third International Symposium on Fault-Tolerant Computing* (pp. 198-207). IEEE.

Mohan, N. R., & Raj, E. B. (2012, November). Resource Allocation Techniques in Cloud Computing--Research Challenges for Applications. In *2012 fourth international conference on computational intelligence and communication networks* (pp. 556-560). IEEE.

Nita, M. C., Pop, F., Mocanu, M., & Cristea, V. (2014). FIM-SIM: fault injection module for CloudSim based on statistical distributions. *Journal of telecommunications and information technology,* (4), 14-23.

Silva Filho, et al. (2017, May). CloudSim plus: a cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness. In *2017 IFIP/IEEE symposium on integrated network and service management (IM)* (pp. 400-406). IEEE.

Silva Filho, et al. (2017, May). CloudSim plus: a cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness. In *2017 IFIP/IEEE symposium on integrated network and service management (IM)* (pp. 400-406). IEEE.

Vishwanath, K. V., & Nagappan, N. (2010, June). Characterizing cloud computing hardware reliability. In *Proceedings of the 1st ACM symposium on Cloud computing* (pp. 193-204).

Yu, R., Xue, G., Zhang, X., & Li, D. (2017, May). Survivable and bandwidth-guaranteed embedding of virtual clusters in cloud data centers. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications* (pp. 1-9). IEEE.

Zhang, H., Dong, F., Shen, D., Xiong, R., & Jin, J. (2017, April). Virtual network fault diagnosis mechanism based on fault injection. In *2017 IEEE 21st International Conference on Computer Supported Cooperative Work in Design (CSCWD)* (pp. 384-389). IEEE.

Ziade, H., Ayoubi, R. A., & Velazco, R. (2004). A survey on fault injection techniques. *Int. Arab J. Inf. Technol., 1*(2), 171-186.

Submitted: 03.04.2023
Accepted: 29.05.2023