Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 7.056

IJCSMC, Vol. 9, Issue. 12, December 2020, pg.1 – 10

Host Fault Injection Using Various Distribution Functions

Manoj Kumar Malik
Maharaja Surajmal Institute of Technology
manojmalik@msit.in

DOI: 10.47760/ijcsmc.2020.v09i12.001

Abstract-- Nowadays, people are connected to the cloud, i.e., back end servers to implement various tasks such as storage of important data, running applications of higher compatibility and so on. While performing such tasks, a number of hosts within the system may encounter various faults resulting in their downfall. Once a system failure occurs, it has an effect on the execution of the tasks performed by the failed components such as hosts or virtual machines. Subsequently, establishing a requirement for a cloud climate that helps in running various hosts or virtual machines effectively in a cloud computing framework independent of the fault or failure happening inside the parts of the framework.

Keywords-- Host Fault Injection, Mean-Time between failure, Cloud Computing, Datacenter, Broker, Virtual Machines, Hosts and Distributions

1. Introduction

The applications of computer systems have increased in various aspects over the past several years which have led to the overall progression & growth in the technological world. Such systems have successfully reduced the workload by completing the normal day-to-day tasks. Thus, making people dependable on these systems
However, the malfunctioning of a part or whole of these systems can lead to the loss of time & energy required for completing tasks. Therefore, it is important to check for the durability of such systems & ensure that the maintenance of these systems is done as soon as the faults are detected within them.

There have been several researches & studies over the past several decades which show various methods, techniques or strategies to check the durability of computer systems or for handling the faults within computer systems either proactively (before the fault occurs) or reactively (after the fault occurs).

Fault injection is a method which is used for injecting faults within computer systems (particularly within hosts or virtual machines). It is useful for checking the durability of hosts or the systems in which the faults are administered. It also helps to identify the strengths & weaknesses of computer systems.

In this paper, we describe about the working of a model that helps in running a system of hosts successfully &improve the overall performance of the cloud environment irrespective of the failure or faults occurring within the host or the untimely failure of the system during runtime. We have also analyzed several distribution functions for finding the probability of occurrence of faults &, thus, an optimal distribution function is selected from various distribution functions which are, then, used for reducing the faults occurring within the hosts in order to minimize the failure impact on the system.

2. Related Works

In request to test a framework's abilities and accessibility, his reaction in excellent circumstance is examined and checked [5]-[7]. Deficiency infusion is the vital activity for testing and making these anomalous circumstances for a framework, offering him as info defective states [8]. The inspiration around this sort of testing where the framework is purposefully presented to undesirable situations is that genuine occasions are difficult to gather and desirable over be dodged. With these strategies we can get disappointments and approve the framework accessibility to extraordinary situations. The most widely recognized disappointments are: crash, break of reaction, erroneous reaction message, subjective fizzles (byzantine fault) [9].

Adjacent to the above arrangement of usually disappointments we actually have the equipment bombs experiences, which are the vast majority of the occasions skirted by excess parts of worker's key things [10]. Here it's worth to make reference to plates disappointment, network availability issues (network over-burden or connector disappointment) and not least the climate occurrences (fire, floods, seismic tremors, and so forth) To characterize the worker accessibility and feasibility, the business utilizes the majority of the occasions the pointer Mean Time To Failure (MTTF). This boundary is characterized as the up-time partitioned by the quantity of disappointments [11].

Subsequently, numerous disappointments are connected with one another and can tie to a progression of basic occasions that can bring down the framework. The subjective fall flat is by a long shot the most troublesome inability to foresee because of its obviously haphazardness. The shortcoming lenient strategy, intended to forestall such kind of disappointments, is roused by the Byzantine Generals Problem [12]. For this situation, the framework's parts will fizzle in subjective manners and the general framework may react in a capricious manner except if is intended to be shortcoming open minded.

The flaw infusion will help the cloud supplier by infusing into the framework a few occasions, following a numerical circulation, with the fundamental objective of bombing a few segments, for instance, the make virtual machine module. In an Adaptive Fault Tolerance progressively distributed computing is proposed [13]. This plan endures the shortcomings on the unwavering quality premise of each figuring hub. A virtual machine is chosen for calculation within the event that it's a better unwavering quality level and may be taken out if does not perform well for in-progress applications. There are unit 2 primary forms of hubs: a bunch of virtual machines, running on a cloud foundation, and a sinking hub. The virtual machine contains the constant application calculation Associate in nursing an acknowledgment take a look at for its smart legitimacy. On the individual, there's an amount checker, Associate in Nursing unwavering quality administrative official, and a few alternative system modules. The area of mediation hub relies upon the sort of the continuous applications and the situation where they are utilized. It very well may be a piece of the cloud foundation or can be a piece of the client framework.

The proposed Fault Injector Module will likewise help the above proposed versatile adaptation to non-critical failure module, by offering the setting of deciding the unwavering quality of an asset in a specific situation. A basic examination of existing apparatuses for executing adaptation to non-critical failure strategies is introduced [14]. Distributed computing, as the replacement of the network frameworks, has all the characteristics of the equal framework assembling an assortment of virtualized hubs, powerfully provisioned and introduced as one bound together processing asset. The assets are distributed through the principles of administration level arrangements and haggled between the specialist organization and customer. Investigating and testing the presentation of a disseminated framework, for example, a public cloud has gotten to a greater degree a test. Distributed computing conditions are offering a powerfully huge pool of assets, configurable and alternatively rebalanced. A full trial of a public cloud can bring about a huge expense and time, with the likelihood to go to a large number of preparing center included. The most attainable choice to test the administration disclosure execution, booking, observing, and so forth, of these frameworks without a versatile climate is a reproduction device [15]. This instrument should have the option to duplicate the applicable tests and conduct of a genuine framework.

cloudsim is a demonstrating and reproduction stage for distributed computing frameworks. The fundamental motivation behind the stage is to give to the client helpful data about the expense of given applications ran on the cloud explicit hard product and anticipate the compromises among cost and execution. The picked recreation apparatus for proposed arrangement is CloudSim. The seclusion of the device settled on it the ideal decision. Every segment is actualized as a Java class and can be broadened simple. CloudSim can give an extensible recreation structure summed up by the fundamental properties of the cloud idea [16].
This segment presents different measurable appropriations: the ones that as of now exist in CloudSim in addition to another – Poisson. It additionally presents the key ideas, usage and further subtleties for cloud recreation, adaptation to non-critical failure and different reenactment devices. We portray here just the Weibull, Poisson, ordinary, remarkable and Pareto disseminations. Different utilized distribution methods are [17]:

• Uniform distribution (utilized very well in circumstance of danger examination yet in addition in calculations for arbitrary age of numbers because of its appropriateness of given equivalent likelihood over a known reach for persistent circulation);
• Gamma distribution – model dramatically amounts of irregular factors;
• LogNormal distribution– Galton appropriation, utilized frequently for dependability demonstrating of the application to accomplish adaptation to internal failure situations)
• Lomax distribution – Pareto distribution, sed as an option in contrast to the remarkable circulation with information vigorously followed;
• Zipf distribution, a discrete circulation with numerous applications in semantics and displaying uncommon occasions.
Weibull distribution – forever information examination, it is the most utilized factual model. As consistent likelihood dissemination, it is utilized in ceaseless reproductions with application in financial determining, climate guaging and all issues dependent on the arrangement of time subordinate fractional differential conditions.
The Weibull distribution, specifically cases, it inserts between two known appropriations: remarkable conveyance and Rayleigh dissemination. On the off chance that we characterize the irregular Weibull variable as an ideal opportunity to-disappointment, at that point we will have dissemination where the pace of disappointment is relative to an intensity of time. Thusly, the Weibull dissemination changes drastically with the estimation of the shape boundary.
This boundary in the span (0,1) could be deciphered as follows:
• Failure rate diminishes fit as a fiddle boundary < 1,
• disappointment rate increments with time for shape boundary > 1, a model here could be a maturing cycle that is probably going to fall flat as time passes by,
• Constant disappointment rate fit as a fiddle boundary = 1 (arbitrary outer occasions are causing the disappointment).

The Poisson appropriation could be a valuable and used circulation within the investigation in light-weight of the actual fact that varied irregular occasion's area unit following the instance of this distribution. The Poisson distribution is separate probability dissemination that may be used to determine the probability of sure occasion numbers to happen in a very fastened timeframe and house. The occasions thought of ought to be free and with a realized traditional event rate.
The probability parts of the Poisson distribution for a given distinct irregular variable. The boundary utilized by the dispersion is that the complex quantity of progress that happened due to the Poisson analysis, a standard range of victories that happens during a specific well-known stretch. within the Poisson explore, the probability of a triumph to happen is relative to the dimensions of the span/area, and also the decreased is that the period or scene the probability are going to be close to zero.
Pareto Distribution – the appropriation is known as once the specialist economic expert and accustomed portray discernible occasions in varied fields of ability. The measurable investigation of the circulation will uncover the key occasions, that impact altogether the occasion's chain a part of the appropriation. once thorough examination in internal control measures, outlining the occasions captivated with the dispersion, the Vilfredo Pareto decide was characterized locution that eightieth of the problems (occasions) are caused by 2 hundredths of key occasions/activities fouled up. The endurance work is given by the probability of the Vilfredo Pareto capricious variable to be additional noteworthy than some variety.
The circulation can be utilized to depict numerous circumstances for harmony found in enormous/little things or occasions and mention objective facts about the adequacy of the cycle steps.

2.1 Components of the Working Model

1) **Fault Injector:** fault injectors are utilized for assessing the steadfastness of PC frameworks. Designers use fault injectors to test issue lenient frameworks or parts. A Fault injector tests for flaw location, issue disengagement, reconfiguration and recuperation capacities [18]. It might likewise be utilized for estimating the inclusion and inactivity boundaries, for considering mistake engendering, and for investigating the connection between the remaining burden of the framework and its deficiency taking care of capacities.

2) **Processing Element:** A processing element (PE) performs the basic computations within a system. It may either run only one process or it may be shared among several processes [19].
A processing element is also used for improving performance and reducing the number of memory ports by eliminating the no. of ports to one or more functional units or by providing data paths to several forwarded results from functional unit outputs directly to other functional unit inputs.

3) **Cloudlet:** A cloudlet is responsible for various application services & tasks such as content delivery, social networking, and business workflow. Most of the application services have a pre-assigned instruction length and data transfer overhead that needs to be undertaken during their life cycle. A cloudlet can be extended to support modeling of other performance and composition metrics for applications such as transactions in database-oriented applications.

4) **Datacenter:** A datacenter is like the center framework level administrations which are offered by Cloud suppliers. An average datacenter includes a bunch of figuring has and virtual machines. Likewise, each datacenter part starts up a summed up application provisioning segment which actualizes a bunch of approaches for memory, transfer speed allotment, and capacity gadgets to have and virtual machines.

5) **Datacenter Broker or Cloud Broker:** A datacenter or cloud broker acts as a mediator & coordinates the negotiation taking place between SaaS and Cloud providers. These negotiations are driven by QoS requirements [20]. The broker acts on behalf of SaaS providers. It is responsible for the allocation of various resources or services that can meet the application's QoS needs.

6) **Virtual Machine (VM):** A virtual machine (VM) is managed and hosted by a Cloud host component. Every virtual machine has access to components which store the data inaccessible memories &processors depending upon factors such as the storage size of a VM, the internal provisioning policy of a VM & so on.

2.2 Working Model

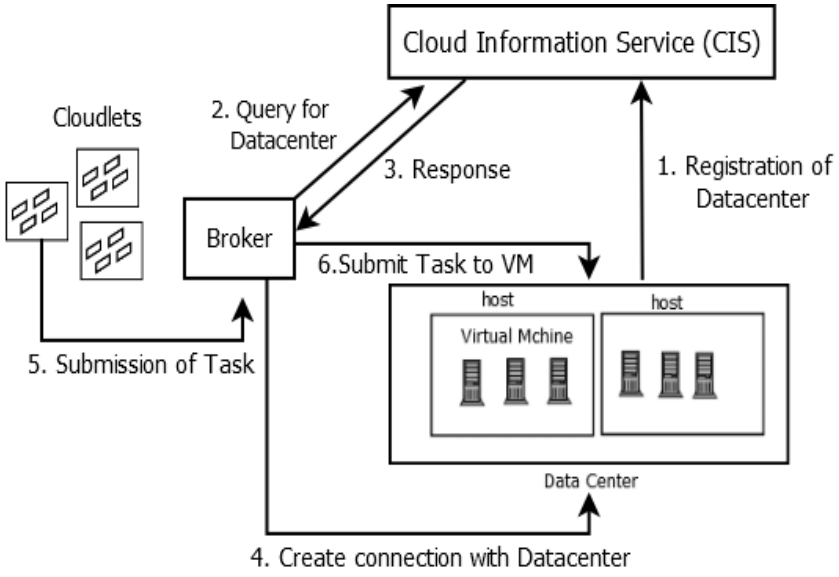

Fig. 1.System Setup for Host Fault Injection

Fig. 1. Shows the various components along with the working of the system, the system consists of a fault injector, processing element, broker, datacenter, cloudlets, virtual machines (VM) & hosts.

The Broker will send at least one cloudlets to the Datacenter and will plan it, as per a Scheduling Policy, on a host. Every element of CloudSim plus can send a specific occasion to another. For this situation, the Fault Injector will make an impression on the Datacenter and it will inform it about any failures that have happened in the framework.

3. Results

3.1. Graph Analysis

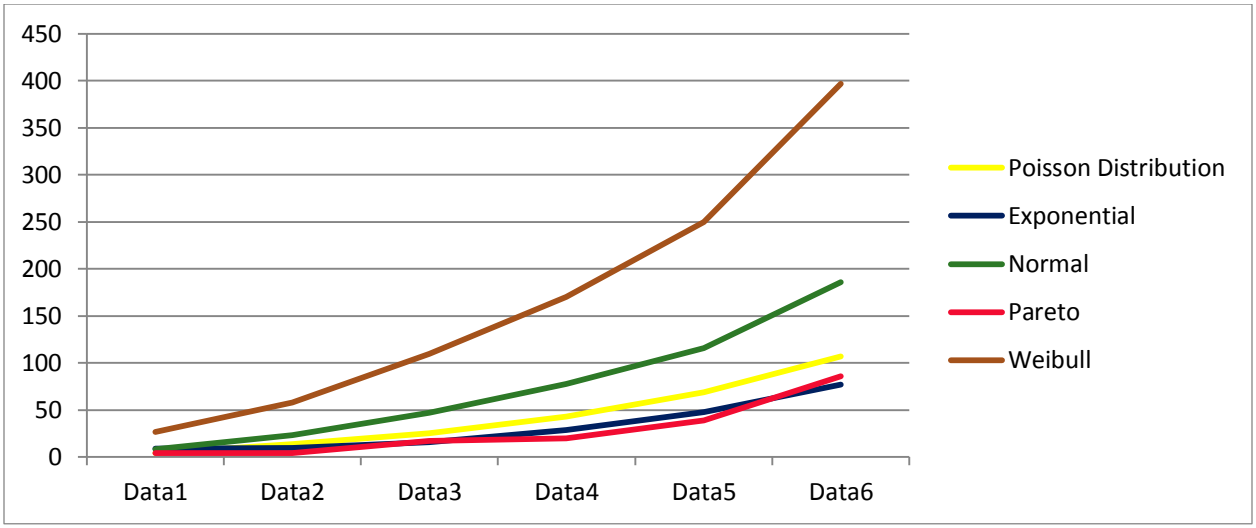

Fig. 2. Number of faults occurring in hosts for various distribution functions

Fault Analysis												
Distribution Function	No of Fault (Cloudet ,Core)		No of Fault (Cloudet ,Core)		No of Fault (Cloudet ,Core)		No of Fault (Cloudet ,Core)		No of Fault (Cloudet ,Core)		No of Fault (Cloudet ,Core)	
	2	2	4	3	6	4	8	5	10	6	12	8
	Data1		Data2		Data3		Data4		Data5		Data6	
Poisson Distribution	4		14		25		43		69		107	
Exponential	9		10		16		29		48		77	
Normal	8		23		47		78		116		186	
Pareto	4		4		17		20		39		86	
Weibull	27		58		110		170		250		397	

Table 1: Fault Analysis

The above graph shows the number of faults occurring in hosts for various distribution functions. There are 6 Data for each distribution function which are represented by Data 1, Data 2 up to Data 6. Each data has different values for both parameters: Cloudlet & Core. Also, the no. of cloudlets & no. of cores for each data is increased in the next data set. For each data, there is no. of faults occurring for each distribution function.

It is clearly shown in the above graph that no. of faults occurring for Poisson distribution has a value of 4 for 2 cloudlets & 2 cores. Whenever the no. of cloudlets & cores are increased for the next data set, the no. of faults also increases for each distribution function. The increase in the no. of faults may vary depending upon the performance of each distribution function.

For each data set, the no. of faults occurring is the minimum for Pareto distribution functions but maximum for Weibull distribution function. According to Data 1 (2 cloudlets & 2 cores), the no. of faults occurring are the least for Pareto & Poisson distribution functions. However, for Data 3 (6 cloudlets & 4 cores) & Data6 (12 cloudlets & 8 cores), the no. of faults occurring is the least for Exponential distribution function. Thus, it is clear that for the least no. of cloudlets & cores, Pareto distribution function gives the best performance. However, for a large no. of cloudlets & cores, Exponential distribution function gives the best performance.

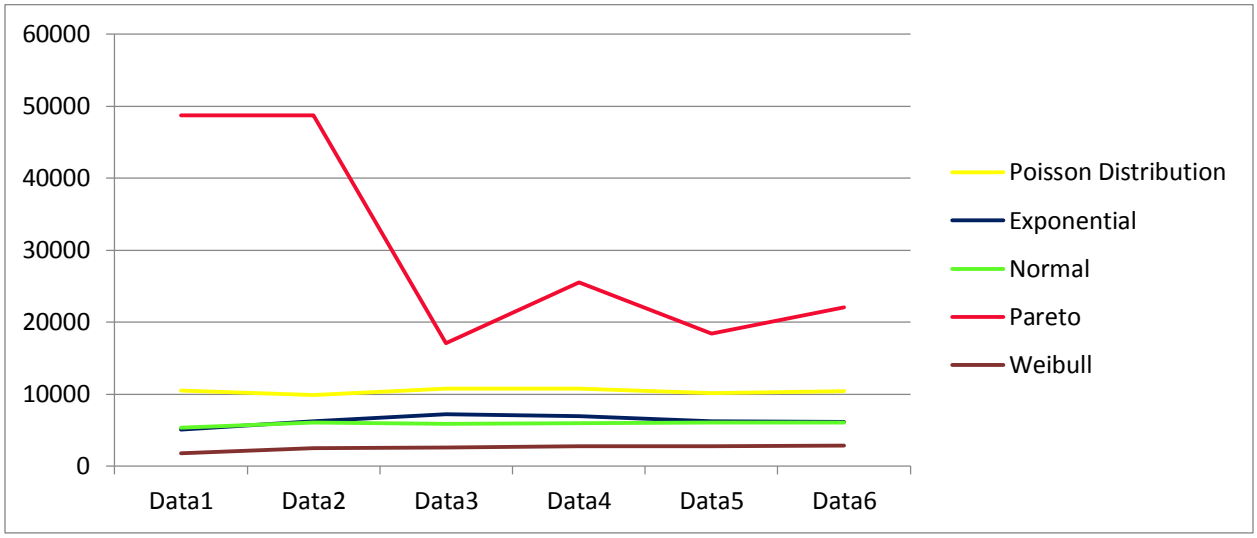

Fig. 3. Mean time between failures for various distribution functions (MTBF)

MTBF (In Minutes)												
Distribution Function	MTBF (Cloudet ,Core)		MTBF (Cloudet ,Core)		MTBF (Cloudet ,Core)		MTBF (Cloudet ,Core)		MTBF (Cloudet ,Core)		MTBF (Cloudet ,Core)	
	2	2	4	3	6	4	8	5	10	6	12	8
	Data1		Data2		Data3		Data4		Data5		Data6	
Poisson Distribution	10524		9912		10806		10729		10125		10432	
Exponential	5106		6209		7248		6949		6223		6182	
Normal	5308		6010		5888		5965		6006		6006	
Pareto	48703		48703		17117		25544		18442		22046	
Weibull	1812		2452		2587		2745		2802		2860	

Table 2: Mean Time between Failures (MTBF)

The above graph shows the mean time between failures (MTBF) for various distribution functions. MTBF is defined as the average time taken by all the faults occurring in the host. The higher the MTBF value, the lesser is the no. of faults occurring in the hosts & vice versa.

It is clearly shown in the above graph that MTBF (in min) for Pareto distribution has a value of 48703 min for Data 1 (2 cloudlets & 2 cores). At Data 1, Pareto distribution gives the best MTBF value. The MTBF value for Pareto

distribution remains steady from Data 1 to Data 2, then shows a drop from Data 2 to Data 3 followed by various rises & drops between the succeeding data sets.

When the no. of cloudlets & cores are increased from Data 1 to Data 3, there is a steady increase in the MTBF value for Exponential distribution. However, from Data 3 to Data 6 there is a slight decrease in MTBF value for Exponential distribution.

However, there are very slight changes in the MTBF values for Poisson, Normal & Weibull distributions between each succeeding data set.

3.3. Result snapshots and evaluation matrix of the results

The simulation results and outputs for the various distributions are given as follows:

1) Pareto Simulation Results

```
Cloudlet|Status |DC|Host|Host PEs |VM|VM PEs   |CloudletLen|CloudletPEs|StartTime|FinishTime|ExecTime
    ID|      |ID|  ID|CPU cores|ID|CPU cores|        MI|  CPU cores|  Seconds|   Seconds| Seconds
------------------------------------------------------------------------------------------------------
     1|SUCCESS| 1|   0|       3| 1|       2| 2800000000|          3|        0|  8400000| 8400001
     3|SUCCESS| 1|   0|       3| 1|       2| 2800000000|          3|        0|  8400000| 8400001
     2|SUCCESS| 1|   1|       4| 2|       2| 2800000000|          3|        0|  8400000| 8400001
     4|SUCCESS| 1|   1|       4| 2|       2| 2800000000|          3|        0|  8400000| 8400001
------------------------------------------------------------------------------------------------------

# Mean Number of Failures per Hour: 0.010 (1 failure expected at each 0.01 hours).
# Number of Host faults: 4
# Number of VM faults (VMs destroyed): 0
# Time the simulations finished: 3299.4861 hours
# Mean Time To Repair Failures of VMs in minutes (MTTR): 0.00 minute
# Mean Time Between Failures (MTBF) affecting all VMs in minutes: 0.00 minutes
# Hosts MTBF: 48703.00 minutes
# Availability: 100.00%

HostFaultInjectionParetoDistr finished!
```

2) Exponential Simulation Results

```
Cloudlet|Status |DC|Host|Host PEs |VM|VM PEs   |CloudletLen|CloudletPEs|StartTime|FinishTime|ExecTime
    ID|      |ID|  ID|CPU cores|ID|CPU cores|        MI|  CPU cores|  Seconds|   Seconds| Seconds
------------------------------------------------------------------------------------------------------
     1|SUCCESS| 1|   0|       3| 1|       3| 1800000000|          3|        0|  3600000| 3600001
     3|SUCCESS| 1|   0|       3| 1|       3| 1800000000|          3|        0|  3600000| 3600001
     2|SUCCESS| 1|   1|       4| 2|       3| 1800000000|          3|        0|  3600000| 3600001
     4|SUCCESS| 1|   1|       4| 2|       3| 1800000000|          3|        0|  3600000| 3600001
------------------------------------------------------------------------------------------------------

# Mean Number of Failures per Hour: 0.010 (1 failure expected at each 100.00 hours).
# Number of Host faults: 10
# Number of VM faults (VMs destroyed): 0
# Time the simulations finished: 1105.0924 hours
# Mean Time To Repair Failures of VMs in minutes (MTTR): 0.00 minute
# Mean Time Between Failures (MTBF) affecting all VMs in minutes: 0.00 minutes
# Hosts MTBF: 6209.00 minutes
# Availability: 100.00%

HostFaultInjectionExponentialDistr finished!
```


3) Poisson Simulation Results

Cloudlet ID	Status	DC ID	Host ID	Host PEs CPU cores	VM ID	VM PEs CPU cores	CloudletLen MI	CloudletPEs CPU cores	StartTime Seconds	FinishTime Seconds	ExecTime Seconds
1	SUCCESS	1	0	3	1	2	2800000000	3	0	8400000	8400001
3	SUCCESS	1	0	3	1	2	2800000000	3	0	8400000	8400001
2	SUCCESS	1	1	4	2	2	2800000000	3	0	8400000	8400001
4	SUCCESS	1	1	4	2	2	2800000000	3	0	8400000	8400001

Mean Number of Failures per Hour: 0.010 (1 failure expected at each 100.00 hours).
Number of Host faults: 14
Number of VM faults (VMs destroyed): 0
Time the simulations finished: 2478.8906 hours
Mean Time To Repair Failures of VMs in minutes (MTTR): 0.00 minute
Mean Time Between Failures (MTBF) affecting all VMs in minutes: 0.00 minutes
Hosts MTBF: 9912.00 minutes
Availability: 100.00%

HostFaultInjectionPoissonDistr finished!

4) Lognormal Simulation Results

Cloudlet ID	Status	DC ID	Host ID	Host PEs CPU cores	VM ID	VM PEs CPU cores	CloudletLen MI	CloudletPEs CPU cores	StartTime Seconds	FinishTime Seconds	ExecTime Seconds
1	SUCCESS	1	0	3	1	2	2800000000	3	0	8400000	8400001
3	SUCCESS	1	0	3	1	2	2800000000	3	0	8400000	8400001
2	SUCCESS	1	1	4	2	2	2800000000	3	0	8400000	8400001
4	SUCCESS	1	1	4	2	2	2800000000	3	0	8400000	8400001

Mean Number of Failures per Hour: 0.010 (1 failure expected at each 0.01 hours).
Number of Host faults: 4
Number of VM faults (VMs destroyed): 0
Time the simulations finished: 2437.8555 hours
Mean Time To Repair Failures of VMs in minutes (MTTR): 0.00 minute
Mean Time Between Failures (MTBF) affecting all VMs in minutes: 0.00 minutes
Hosts MTBF: 36121.00 minutes
Availability: 100.00%

HostFaultInjectionLognormalDistr finished!

5) Normal Simulation Results

Cloudlet ID	Status	DC ID	Host ID	Host PEs CPU cores	VM ID	VM PEs CPU cores	CloudletLen MI	CloudletPEs CPU cores	StartTime Seconds	FinishTime Seconds	ExecTime Seconds
1	SUCCESS	1	0	3	1	2	2800000000	3	0	8400000	8400001
3	SUCCESS	1	0	3	1	2	2800000000	3	0	8400000	8400001
2	SUCCESS	1	1	4	2	2	2800000000	3	0	8400000	8400001
4	SUCCESS	1	1	4	2	2	2800000000	3	0	8400000	8400001

Mean Number of Failures per Hour: 0.010 (1 failure expected at each 100.00 hours).
Number of Host faults: 23

Number of VM faults (VMs destroyed): 0
Time the simulations finished: 2368.5161 hours
Mean Time To Repair Failures of VMs in minutes (MTTR): 0.00 minute
Mean Time Between Failures (MTBF) affecting all VMs in minutes: 0.00 minutes
Hosts MTBF: 5950.00 minutes
Availability: 100.00%

HostFaultInjectionNormalDistr finished!

4. Conclusions

Host fault injection is a process of injecting faults within a cloud environment system comprising of a number of hosts including several virtual machines. Numerous studies have been conducted in order to prevent system failure during the execution of tasks in the cloud environment.

This paper described the working model & its various components proposed for the cloud computing environment. The given model explains how the components interact with each other to perform the execution of tasks on their respective hosts & virtual machines.

This paper also provides the comparison between several distribution functions based on the number of faults occurring in the hosts or the mean-time between failures &, thus, represents the comparison with the help of graphs which provide the optimal distribution function to be used for the given scenario.
The performance of distribution functions based on the both the factors is also summarized which is represented in the later sections.

4.1. Performance of distribution functions based on the number of faults injected in hosts

To sum up, the performance of distribution functions based on the number of faults injected in hosts is given as follows:

Best Case Scenario: Pareto & Exponential distribution
Average Case Scenario: Poisson & Normal distribution
Worst Case Scenario: Weibull distribution

4.2. Performance of distribution functions based on mean time between failures

To sum up, the performance of distribution functions based on mean time between failures is given as follows:

Best Case Scenario: Pareto distribution
Average Case Scenario: Poisson, Exponential & Normal distribution
Worst Case Scenario: Weibull distribution

References

[1]. Baughman, Aaron K., et al. "Workload adaptive cloud computing resource allocation." U.S. Patent No. 8,793,381. 29 Jul. 2014.

[2]. Jhawar, Ravi, Vincenzo Piuri, and Marco Santambrogio. "Fault tolerance management in cloud computing: A system-level perspective." IEEE Systems Journal 7.2 (2012): 288-297.

[3]. Ganesh, Amal, M. Sandhya, and Sharmila Shankar. "A study on fault tolerance methods in cloud computing." 2014 IEEE International Advance Computing Conference (IACC). IEEE, 2014.

[4]. Nita, M-C., et al. "FIM-SIM: fault injection module for CloudSim based on statistical distributions." Journal of telecommunications and information technology 4 (2014): 14-23.

[5]. P. Das and P. M. Khilar, "VFT: A virtualization and fault tolerance approach for cloud computing", in Proc. IEEE Conf. Inform. Commun. Technol. ICT 2013, Jeju Island, South Korea, 2013, pp. 473–478.

[6]. S. Malik and F. Huet, "Adaptive fault tolerance in real time cloud computing", in Proc. IEEE World Congr. Serv. SERVICES 2011, Washington, DC, USA, 2011, pp. 280–287.

[7]. Y. He et al., "A simulation cloud monitoring framework and its evaluation model", Simul. Modell. Pract. and Theory, vol. 38, pp. 20–37, 2013.

[8]. S. Vilkomir, "Cloud testing: A state-of-the-art review", Inform. &Secur.: An Int. J., vol. 28, no. 2, pp. 213–222, 2012.

[9]. A. Boteanu, C. Dobre, F. Pop, and V. Cristea, "Simulator for fault tolerance in large scale distributed systems", in Proc. IEEE 6th Int. Conf. Intell. Comp. Commun. and Process. ICCP 2010, ClujNapoca, Romania, 2010, pp. 443–450.

[10]. A. Costan, C. Dobre, F. Pop, C. Leordeanu, and V. Cristea, "A fault tolerance approach for distributed systems using monitoring based replication", in Proc. IEEE 6th Int. Conf. Intell. Comp. Commun. and Process. ICCP 2010, Cluj-Napoca, Romania, 2010, pp. 451–458.

[11]. D. Ford et al., "Availability in globally distributed storage systems", in Proc. 9th USENIX Conf. Operat. Sys. Design and Implemen. OSDI'10, Berkeley, CA, USA, 2010, pp. 1–7. USENIX Association.

[12]. Y. Zhang, Z. Zheng, and M. R. Lyu, "BFTCloud: A byzantine fault tolerance framework for voluntary-resource cloud computing", in Proc. IEEE 4th Int. Conf. Cloud Comput. CLOUD '11, Washington, DC, USA, 2011, pp. 444–451.

[13]. F. Cappello, "Fault tolerance in petascale/ exascale systems: current knowledge, challenges and research opportunities", Int. J. High Perform. Comput. Appl., vol. 23, no. 3, pp. 212–226, 2009.

[14]. A. Nu´nez˜ et al., "A flexible and scalable cloud infrastructure simulator", J. Grid Comput., vol. 10, no. 1, pp. 185–209, 2012.

[15]. L. Liu et al., "Greencloud: A new architecture for green data center", in Proc. 6th Int. Conf. Industry Session on Autonomic Comput. andCommuni. Industry Session ICAC-INDST '09, Barcelona, Spain, 2009, pp. 29–38.

[16]. R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms", Softw. Pract. Exper., vol. 41, no. 1, pp. 23–50, 2011.

[17]. Long, Wang, LanYuqing, and Xia Qingxin. "Using cloudsim to model and simulate cloud computing environment." 2013 Ninth International Conference on Computational Intelligence and Security. IEEE, 2013.

[18]. Calheiros, Rodrigo N., et al. "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms." Software: Practice and experience 41.1 (2011): 23-50.

[19]. Thanakornworakij, Thanadech, et al. "A reliability model for cloud computing for high performance computing applications." European Conference on Parallel Processing. Springer, Berlin, Heidelberg, 2012.

[20]. Humane, Pravesh, and J. N. Varshapriya. "Simulation of cloud infrastructure using CloudSim simulator: A practical approach for researchers." 2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM). IEEE, 2015.

