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: 6.017

IJCSMC, Vol. 7, Issue. 4, April 2018, pg.27 – 41

Addressing Node Failures Using Node Management & Cluster Management Techniques in Cloud Computing

Msagha J Mbogholi[1], Henry O Okoyo[2], Okoth Sylvester J McOyowo[3]

[1]Department of Information Technology, Maseno University, Kenya

[2]Department of Computer Science, Maseno University, Kenya

[3]Department of Computer Science, Maseno University, Kenya

[1] johnmsagha@gmail.com; [2] okoyo.ho@gmail.com; [3] oyowosilver@gmail.com

---

Abstract— Cloud computing has been the rave of users globally over the past few years. This has mainly been because of the many different services available in the cloud that has made it the technology of choice for many users. Whether knowingly or unknowingly once a user connects to the Internet and accesses a service be it an application or storage they are using a cloud service. Availability, however, has been a thorn in the flesh for most Cloud Service Providers (CSPs). One of the sources of outages at cloud infrastructure level has been node failures. In this context the nodes are the servers found within the datacenter (DC) that house the virtual machines (VMs), which in turn execute the tasks required by users. The objective of this study was to determine whether node management and cluster management are proactive effective availability Mechanisms (AMs) in countering node failures. To investigate this we simulated node failures in the DC using a random number generator. We then countered these failures using a node management technique and then secondly using cluster management, using CloudSim simulation toolkit. The availability of the infrastructure was then measured using two parameters namely service availability and execution availability. The findings showed that both techniques are effective techniques in countering node failures at infrastructure level; further effective management of the cluster results in effective management of the node at infrastructure level; the converse is also true.

Keywords— Availability, Cloud computing, infrastructure, node failure, node management, service availability, execution availability, CloudSim

---

I. INTRODUCTION

Cloud computing has grown steadily over the years as the technology of choice for many users globally. The number of users has grown steadily over the years, rising from 2.4 billion users in 2013 to 3.6 billion users in 2018 (projected).

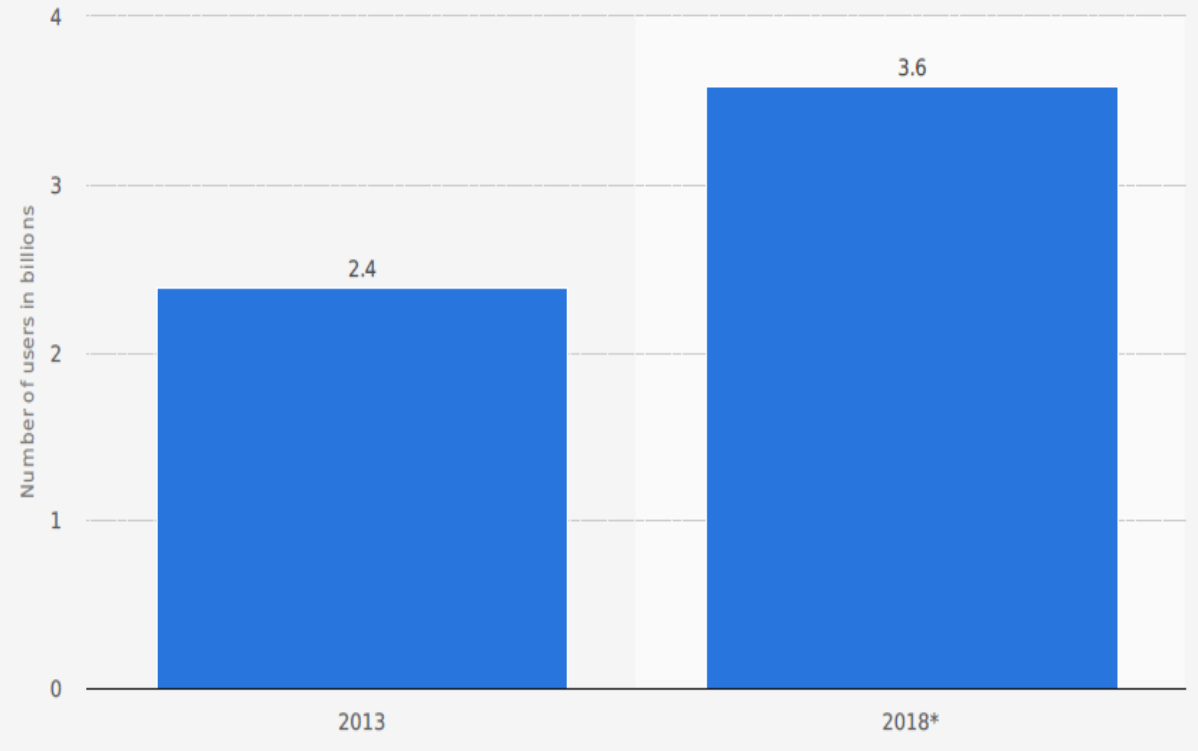

Fig 1 Number of consumers using cloud based services in 2013 and 2018 (*Source: Juniper research, © Statista, 2017*)

The benefits offered by the cloud include (Harding, C as cited by [13]):

- On-Demand Self Service: Cloud consumers must be capable of obtaining the cloud services at the infrastructure, platform or application level at any time without requiring any significant assistance.
- Measured Service: It is similar to utility computing where the used resources and obtained services are measured.
- Resource Pooling: It allows multiple users to make use of available physical and virtual resources.
- Rapid Elasticity: It provides flexible computing based on the demand which can expand or contract.
- Broad Network Access: Cloud services can be obtained in any kind network even it can remotely accessed from mobiles, Laptops, PDA's via internet.

The cloud offers four main deployment models and three services to its users. The deployment models are public, private, community and hybrid clouds. The difference between these models lies in the way they are availed to users:

- Public clouds are available for use by members of the public. These are services that may be provided free of charge or for a fee. An example of a free service would be storage services that are offered for free by Google (Google drive).
- Private clouds are exclusive to organizations. These are services that can only be accessed by members of the organization and outsiders do not have access to this cloud model; access levels even for the users themselves may be

defined in this model. The cloud in this instance may be run by the organization itself or by a third party on its behalf; it may be housed in-premise or at the third party site.

- Community clouds are clouds that are shared between two or more organizations. This allows the organizations to share data between them, i.e. the community shares common interests in one or more areas. This model may also be managed by a third party or by one of the organizations in the community; further it may be managed in-house or by the third party.

- Hybrid clouds are a mix of a private cloud and a public cloud. This enables the organization to use the public cloud only when it needs it services, a concept also known as cloud bursting.

The services offered on the cloud are varied depending on the layer at which the service is offered. There are three main services offered:

- Infrastructure-as-a-Service (IaaS): this service is at the lowest layer and offers to the user servers, storage and network services.

- Platform-as-a-Service (PaaS): this is the intermediate layer and is designed mostly for developers. It offers the tools necessary for development of different applications.

- Software-as-a-Service (SaaS): this is the highest layer and offers to users different applications that they can use on a day to day basis. These applications may include spreadsheets, storage and word processing. The layer also offers enterprise applications.

The services offered by the cloud are provided by Cloud Service Providers (CSPs) who are like Internet Service Providers (ISPs) only difference being that the former provides cloud services while the latter provide Internet connectivity. Availability has been a thorn in the flesh for CSPs and users alike over the years. Further statistics show that outages incidents have been on the increase for the past few years ([4], [5], [12]). These problems (lack of availability) at infrastructure level have by extension led to losses for the clients using cloud services to their chagrin. Indeed availability has now become the leitmotif of cloud computing alongside reliability and security. Whereas outages have been the cause of lack of availability there is little literature that has been availed as to the cause of these outages. In this paper we examined node failure as an outage cause. We then examine how researchers and engineers have been building for availability using different availability mechanisms (AMs). Node management as an AM for node failures is then discussed. A pair of simulations was done where we first injected node failure into the datacenter using CloudSim; next we used a node management technique followed by a cluster management technique to counter the node failure and observed the results. The rest of this paper is divided as follows: in section 2, node failures are discussed as an outage cause; current literature on AMs related to node failures is then discussed. Availability is also defined in this section. Section 3 describes the methodology that was used to perform the simulations. Section 4 presents the results of the simulations, while section 5 discusses them. Section 6 presents the conclusion and directions for further research.

II. LITERATURE REVIEW

In this section availability is first defined; then an examination of node failures and related AMs follows.

A. Availability

Techopedia.com identifies availability as one of the five pillars of information assurance (IA) together with integrity, authentication, confidentiality and non-repudiation. Further, it states that "Availability, in the context of a computer system, refers to the ability of a user to access information or resources in a specified location and in the correct format." Dictionary.com goes further to define availability, *inter alia*, as "readily obtainable, accessible, suitable or ready for use, at hand". "Highly available characterizes a system that is designed to avoid the loss of service by reducing or managing failures as well as minimizing planned downtime for the system. We expect a service to be highly available when life, health, and well-being, including the economic well-being of a company, depend on it" [19]. Ref [15] defined three different types of availability namely inherent, achieved and operational. In this study, however, we used two availability parameters namely service availability (SA) and execution availability (EA). SA was defined by [9] as the ratio of the resources allocated (R_A) to the resources requested (R_R):

Service Availability (SA) = Resources Allocated/Resources Requested = R_A / R_R

We define execution availability (EA) as the ratio of the tasks executed to the tasks requested, with a component of time.

B. Node Failures

Node failure is a major outage cause at CSP level. This is since failure of a node directly implies failure of the VMs running in it due to unavailability of processing power; this in turn means unavailability of the cloud to the users. In a typical datacenter there will be several nodes running in it, with a variety of configurations in place. The most common configuration is to have the nodes all running in a cluster (or several clusters) and sharing workloads between them. A cluster is a configuration whereby several nodes work together as one such that they appear to be one node but in reality there are several nodes sharing workloads based on some workload balancing algorithm. In a heterogeneous setup the nodes are all of different physical configurations while in a homogeneous one they are all of the same configuration and running software from a specific vendor. There are many arguments for and against either of these configurations, with opponents of homogeneous setups citing vendor lock-in as the most significant drawback. Reasons for node failure in the datacenter (DC) vary and these may include failure of the hardware components in the server. Ref [2] also pointed out another cause of node failure when an organization opts to buy cheap and buys servers that do not have the capacity to handle the operations in the datacenter. Other causes of failure may be related to hypervisor failure [11], electricity overloads, virus attacks and resource exhaustion.

C. Availability Mechanisms (AMs)

AMs are the different mechanisms that different authors have used to increase availability in the cloud computing environment. Most service providers build for fault tolerance in their infrastructure. Fault tolerance is generally defined as the ability of a system to remain in operation even if some of the components used to build the system fail [1]; examples of fault tolerance at VM level have been described by [6], [12], [17]. Notably most of the AMs are based on some form of redundancy, for example [18], [20] and the Linux-HA (http://www.linux-ha.org) project. Ref [20] solution is based on clusters as is the CloudDisco solution [16]. These AMs have not been designed specifically to deal with node failures *per se* but they are suitable in dealing with this type of outage; however, most of the current AMs do not address any specific outage cause and this presents a *magna quaestio* as far as the study of the subject of availability in cloud computing is concerned. We discuss this quagmire in a different paper. In this paper we use the concept of virtualization and more precisely VM migration (at node level) and VM workload balancing (at cluster level) as AMs for node failure. This is discussed in detail in the next section.

III. RESEARCH METHODOLOGY

In this section we describe the configuration of the machine that was used and the theory behind the simulations. We also tabulate the configurations that were used for the different simulations. The simulations were done using CloudSim simulation toolkit a discrete, deterministic simulator [3]. The CloudSim architecture consists of different classes that can be extended in order to customize scenarios according to the investigator's needs. This requires knowledge of the Java language as the simulator is written in that language. In the simulator different classes can pass messages to each other and one may use the inbuilt policies or create their own policies to suit the scenario they wish to configure.

In order to understand the simulations it is important to understand the sequence of events that occur in CloudSim when a user requests for a task (called a cloudlet) to be executed:

(1) Datacenter (DC) registers with Cloud Information Service (CIS)
(2) Broker queries CIS for available DCs
(3) CIS allocates DC to broker (in this case DC2)
(4) Broker requests DC to create Virtual Machines (VMs)
(5) Users send cloudlets to broker
(6) Cloudlets assigned to VMs
(7) VMs execute cloudlets according to policy
(8) VMs return results to broker
(9) Broker requests DC to destroy VMs
(10) Broker returns results of cloudlet execution to users

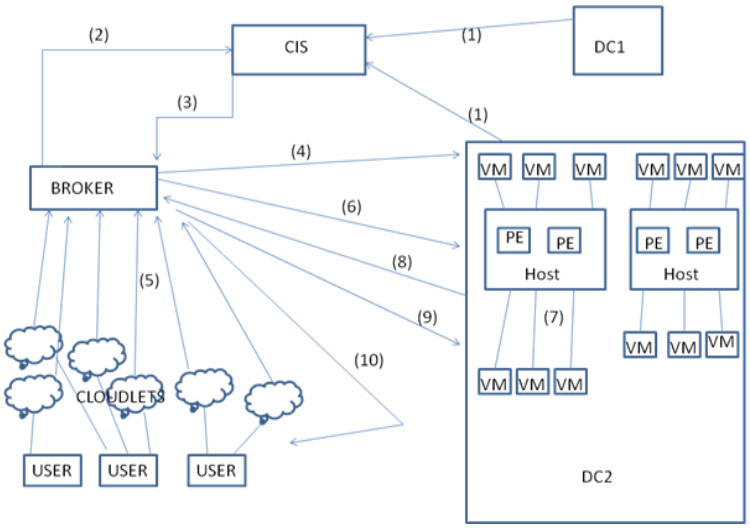

Fig 2 Cloudlet execution events in CloudSim

A. Host Configuration

The configuration of the host machine that was used to perform the simulations was as follows:

Table I

HOST CONFIGURATION DETAILS

Parameter	**Detail**
Manufacturer	Hewlett-Packard
Model	HP ProBook 4340s
Processor	Intel ® Core ™ i5 – 2450M CPU @ 2.50 GHz
Installed Memory (RAM)	8.00 GB
Operating System	Windows 7 Professional
System Type	64-bit Operating System
Integrated Development Environment (IDE)	Netbeans 8.1
Platform	JDK 1.8

Each simulation was done once in line with [14] who stated that in a deterministic simulation only one simulation is enough to generate valid predictions. Each simulation in the different scenarios had different configurations for datacenter, host, Processing Elements (PEs), RAM, VMs, users and cloudlets as specified in the specific simulations.

B. Simulations

1) Simulation 1: Node Failure: This was done by creating a HostFaultInject entity that extends the SimEntity class. In the first simulation the random number generator randomly failed several hosts which ensured that the entity was working. This was done by scheduling an event (schedule(getId(), delay, HOST_FAILURE), with delay based on the random number generator, and HOST_FAILURE being the customized tag that described how the host would fail. When failing hosts it was necessary to ensure that not only the host fails but also the VMs within the host and the PEs. There was no VM migration allowed such that if a host failed then the broker would not request more VMs to be created on other hosts or request VM migration for that matter. When a cloudlet completed execution on a VM then it was sent back to the broker who then requested for the VMs to be destroyed once all cloudlets had been executed in the simulation. At the user end the measure of availability of the cloud was in terms of the tasks (cloudlets) that had been executed. Thus the number of hosts that failed and the number of cloudlets executed were recorded in the simulation.

This sequence of events can be captured in simple form as follows:

HostFaultInject extends SimEntity;
Broker: send cloudlets;

While Random_Number_Generator ON,
//Fail hosts randomly,
Do {//HOST_FAILURE
Fail host,
Fail VMs, //in the host
}
Send cloudlets to broker

Table II

SIMULATION 1 CONFIGURATION

Datacenter	**Users**	**PEs**	**MIPS**	**Hosts**	**VMs**	**Cloudlets**

1	5	Mixed (2 for dual core and 4 for quad core)	1000	10
				ram = 16384; //host memory (MB)
				long storage = 1000000; //host storage
				bw = 10000; //bw in Kbit/s
					30
					size = 10000; //image size (MB)
					RAM = 512; //vm memory (MB)
					MIPS = 250;
					BANDWIDTH = 1000; //bandwidth in Kbit/s
					pesNumber = 1; //number of cpus
					VMM = "Xen"; //VM Manager
						500
						length = 40000; //in MI
						fileSize = 300; //in bytes
						outputSize = 300 //in bytes
						pesNumber = 1;

CloudSim tag: HOST_FAILURE; Event: (schedule(getId(), delay, HOST_FAILURE)

2) Simulation 2: Introducing node management: Nodes are managed according to some provisioning policy. It is the cores in the nodes that provide the VMs with the execution environment and thus when a node fails it goes along with the VMs that had been created in it. The question arises then as to what happens to the cloudlets being executed in it? Without node management the cloudlets will not be executed and thus they will return with status INCOMPLETE or will not be seen in the cloudlet list that the broker produces showing the executed cloudlets. Node management then means that the datacenter should allow VMs to be migrated to other hosts according to some policy. Once the VMs have been migrated to other hosts then cloudlet execution can continue and the user should receive the results of the execution. In this simulation VM migration was enabled using the addMigratingInVm(Vm vm) method in the host, and by using the processCloudletMove(int[] receivedData, int type) method in updateVMProcessing() method at both datacenter and host level the status of tasks could be monitored during the simulation, and thus the cloudlets executed. The same policy, i.e. space-shared policy for the VMs in the host objects was maintained. With the policy enabled Simulation 1 was repeated and the number of hosts failed and cloudlets executed were recorded.
This sequence of events can be captured in simple form as follows:
//Node management
HostFaultInject extends SimEntity;
Broker: send cloudlets
While Random_Number_Generator ON,
//Fail hosts randomly,
Do { //HOST_FAILURE
Set VM Migration ON, //activate VM migration
Fail host,
If HostFailure = = TRUE;
addMigratingInVM //call/invoke the VM migration method

migrate VMs //according to some policy
UpdateVMProcessing //record where the VMs have been moved to and continue processing
}
Send cloudlets to broker

Table III

SIMULATION 2 CONFIGURATION

Datacenter	Users	PEs	MIPS	Hosts	VMs	Cloudlets

1	5	Mixed (2 for dual core and 4 for quad core)	1000	10
				ram = 16384; //host memory (MB)
				long storage = 1000000; //host storage
				bw = 10000; //bw in Kbit/s
					30
					size = 10000; //image size (MB)
					RAM = 512; //vm memory (MB)
					MIPS = 250;
					BANDWIDTH = 1000;
					pesNumber = 1; //number of cpus
					VMM = "Xen"; //VM Manager
						500
						length = 40000; //in MI
						fileSize = 300; //in bytes
						outputSize = 300 //in bytes

Methods:

- addMigratingInVm(Vm vm) – in host
- processCloudletMove(int[] receivedData, int type) – in datacenter and host
- updateVMProcessing() – in datacenter and host

Policy: space – shared.

3) Simulation 3: Introducing cluster management: It is important to note that the datacentercharacteristics class is the one used by CloudSim in managing the cluster. Cluster management in this instance would then imply following the same configuration and approach as in simulation 2 since it would mean moving tasks among the nodes in the cluster such that the user will only see the results of the execution itself and not which node actually performed the task; essentially the difference here being that the migration is in form of load balancing policy across the cluster, while in simulation 1 the migration was based on migrating the VMs to the next available node. In order to achieve this node failure was configured such that the VMs in each node would not fail but be migrated to other working nodes; thus this would ensure the cloudlets being executed in them would receive the PEs required to complete execution. Further the allocation policy was changed to time shared to accommodate the same MIPS rating as in simulation 1 of this scenario (retaining space shared resulted in a VM creation error due to MIPS). The same parametric results as in simulation 1 were recorded.

//Cluster management

HostFaultInject extends SimEntity;
Broker: send cloudlets
While Random_Number_Generator ON,
//Fail hosts randomly,
Do { //HOST_FAILURE
Set VM Migration ON, //activate VM migration
Fail host,
If HostFailure = = TRUE;
addMigratingInVM //call/invoke the VM migration method according to load balancing algorithm
migrate VMs //according to some policy
UpdateVMProcessing //record where the VMs have been moved to and continue processing
}
Send cloudlets to broker

Table IV

SIMULATION 3 CONFIGURATION

Datacenter	Users	PEs	MIPS	Hosts	VMs	Cloudlets
1	5	2 - for dual core machines	1000	10 ram = 16384; //host memory (MB) long storage = 1000000; //host storage bw = 10000; //bw in Kbit/s	50 size = 10000; //image size (MB) RAM = 512; //vm memory (MB) MIPS = 250; BANDWIDTH = 1000; pesNumber = 1; //number of cpus VMM = "Xen"; //VM Manager	500 length = 40000; //in MI fileSize = 300; //in bytes outputSize = 300 //in bytes

CloudSim tag: HOST_FAILURE
Event: (schedule(getId(), delay, HOST_FAILURE)
Cluster characteristics: in datacentercharacteristics class
Attribute: resource architecture
VMAllocation Policy: time – shared

IV. RESULTS

This section discusses the results of the simulations. There were three simulations that were performed; the first was simulating node failure, the second was simulating node management while the third was simulating cluster management, respectively as counters to node failure.

A. Simulations

The output of the simulations was captured in the figures below:

1) Simulation 1: (no node management):

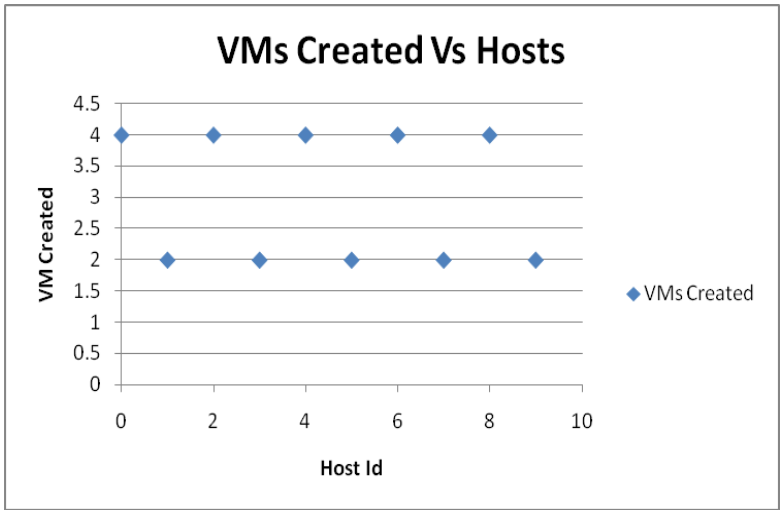

Fig 3 Hosts and VMs created

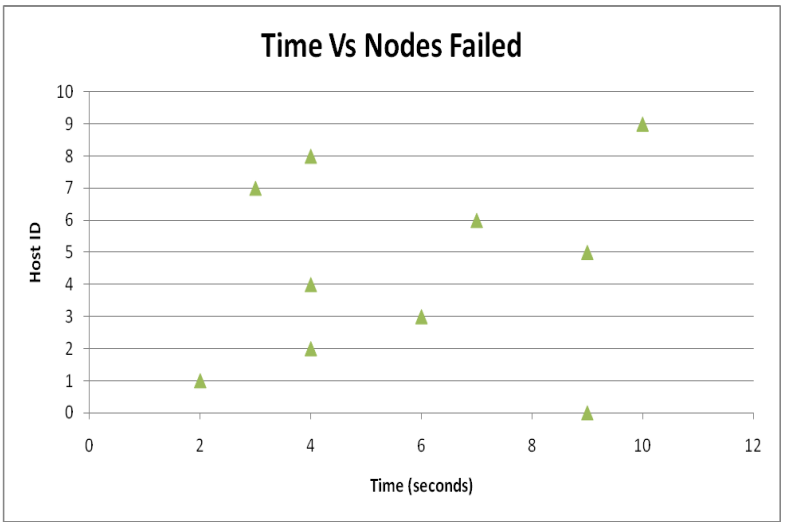

Fig 4 Nodes failed during simulation

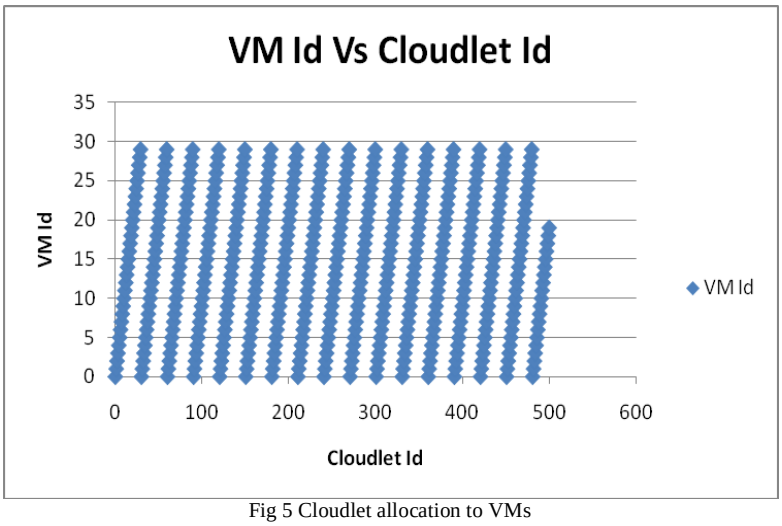

Fig 5 Cloudlet allocation to VMs

2) Simulation 2 (node management):

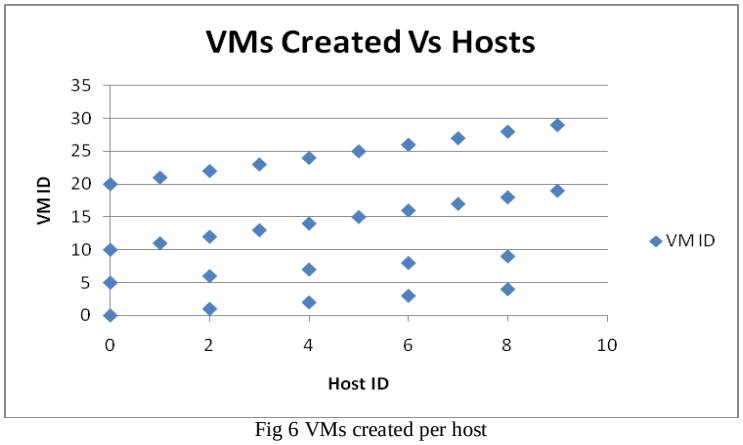

Fig 6 VMs created per host

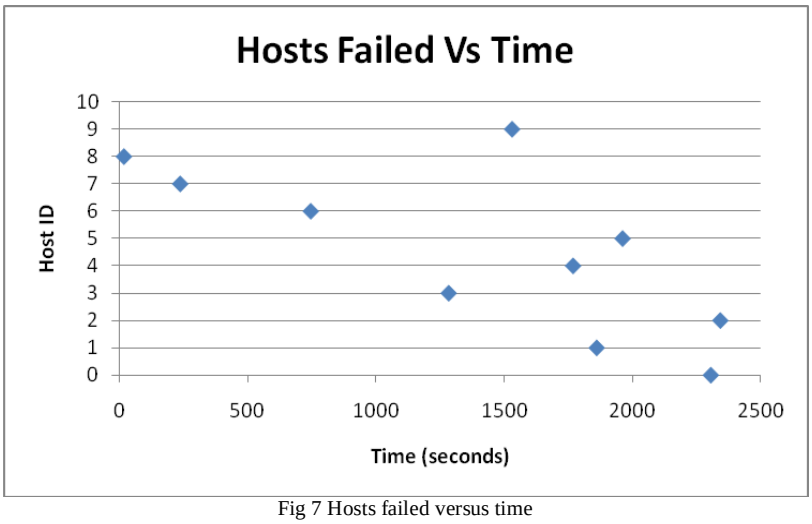

Fig 7 Hosts failed versus time

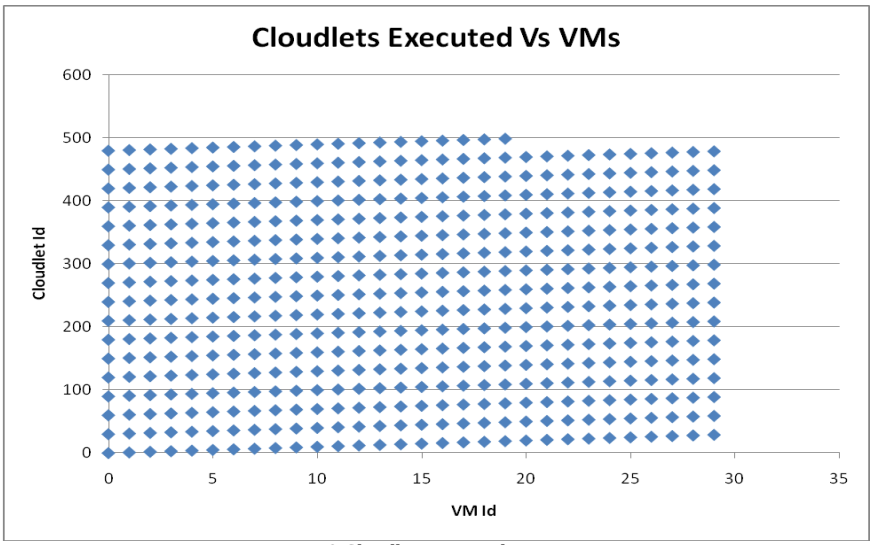

Fig 8 Cloudlets executed per VM

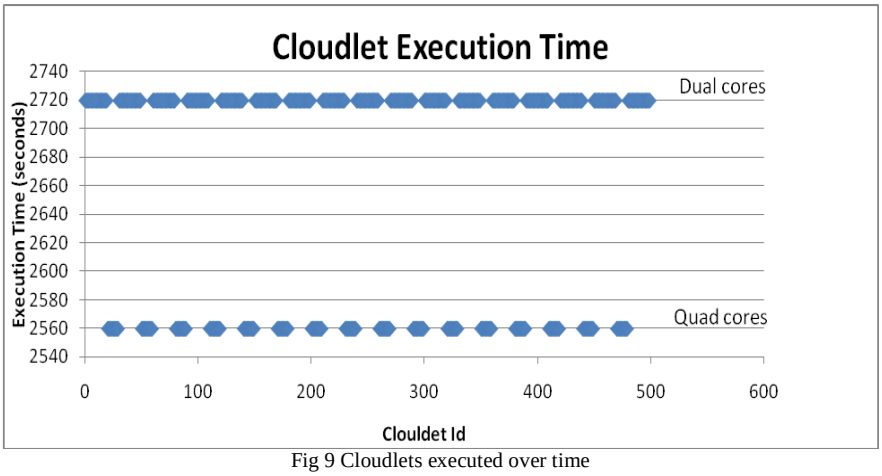

Fig 9 Cloudlets executed over time

3) Simulation 3 (cluster management):*

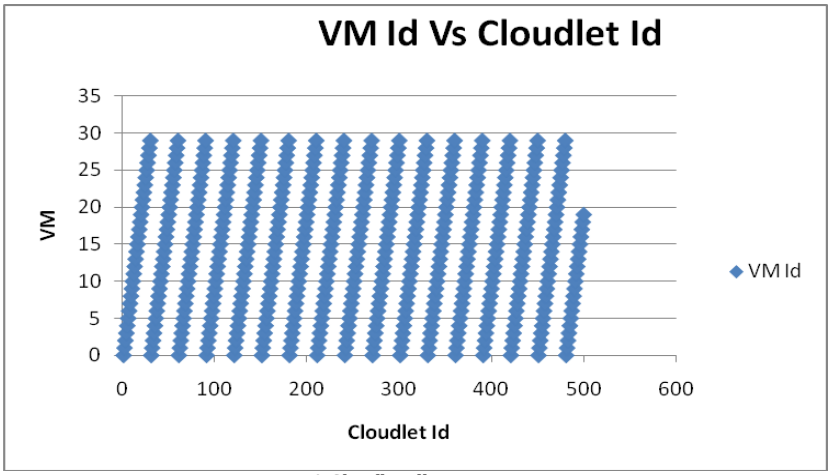

Fig 10 Cloudlet allocation to VMs

*The rest of the results for this simulation were similar to *Fig* 6, 7 and 8.

V. DISCUSSION

In simulation 1 of this scenario there was no node management and hosts were failed using a random number generator. As was expected during the simulation the hosts failed randomly over a 10 second period, for example at the 4 second mark three hosts all failed at the same time, i.e. host ids 2, 4 and 8, while at the 10 second mark only host id 9 failed (Fig 4). This was done purposely to reflect the dynamic nature of the cloud. In a real world environment it would be expected that hosts would not fail in a linear or some preordained manner, but rather randomly. The results also show that despite VMs being created in the host (Fig 3) this did not have any effect on cloudlet execution, or lack thereof. All 30 VMs were created during the simulation. The simulation also shows that the broker did send all the 500 cloudlets to the VMs that had been created, and allocation was done (Fig 5). However, the hosts started failing at the 2.0 second mark (Fig 4). This resulted in no cloudlet being returned to the broker from that point on to the end of the simulation. The reason for this is that while the VMs were created in the hosts, there was no mechanism to move them as their respective hosts failed. Essentially this means that the cloudlets all terminated in the VMs that they had been assigned to. The assignment of cloudlets to the VMs occurs according to the assigned policy, in this case a simple round robin policy that allocates first cloudlet to the first VM (VM #0), next one to the next VM (VM #1), and so on, till all cloudlets have been assigned. This explains why VM Id 19 had fewer cloudlets allocated to it than the others. The node failure process can be demonstrated in four steps:

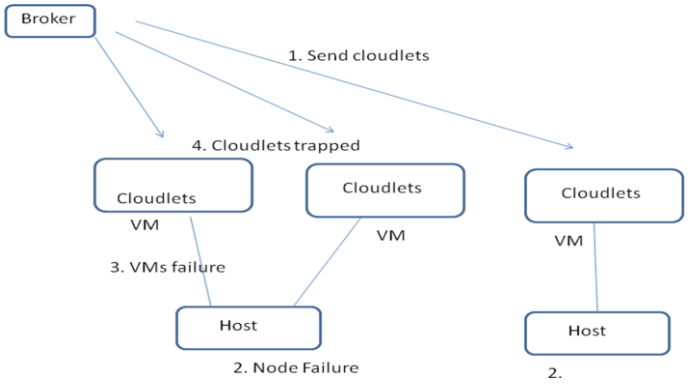

Fig 11 Node failure

In step 1 the broker sends the cloudlets to the the VMs in the hosts. In step 2, the nodes are failed. This causes the VMs to fail as well in step 3. Finally the cloudlets get trapped in the VMs in step 4 and thus can not be returned to the broker. As the simulation indicated node failure resulted in zero availability of the datacenter to the broker and cloudlets; though the cloudlets were sent none of them were executed by the VMs in the datacenter.

Let the number of cloudlets executed = μ;
Let the time taken to execute the tasks = t;
Let the number of cloudlets requested = λ;
Execution availability (EA) = μ / λt = 0/(50 * 0) = 0; {t = 0, in this instance since no tasks were executed}

In an ideal situation an EA of 1 would be an ideal score, i.e. all cloudlets requested are executed in a maximum time t of 1 second. Nonetheless higher values of EA are more desirable as this would mean less time to execute which in turn means more tasks executed in a given time. Understandably this is not achievable at this time due the nature and number of tasks; further in this study it is not possible to determine the exact time each cloudlet takes to be executed should the cloudlets all be of different sizes (which would be the ideal case in a real world scenario) and thus state unequivocally that the average time it takes to execute a single task is say Ω seconds.

Further, using the definition of service availability [9] we find that this type of availability can only be measured either over a period of time or at a discrete moment in time. In this scenario it changes over the whole period of the simulation. For the purposes of this simulation we assume the resource being measured is the node itself:

Service Availability (SA) = Resources Allocated/Resources Requested = R_A / R_R
At the 500 second mark it is 7/9 = 77.78 %
At the 1000 second mark it is 6/9 = 66.67%
At the 1500 second mark it is 5/9 = 55.56%
At the 2000 second mark it is 2/9 = 22.22%
And finally by the 2500 second mark it is 0%

Thus the service availability ratio is variant depending on at what time it was being measured. Nonetheless as the nodes were failing together with the VMs then the service availability percentages will still remain the same even when measuring the VM as the requested resource since the ratio of the resource requested versus the resource allocated is 100%. For the purposes of the study we posit that this would be an erroneous measure to use since no execution occurs and thus from a user's point of view the service was not available to complete the requested tasks (cloudlets).

In simulation 2, the same configuration was used only this time the VM migration was allowed by configuring a VM migration policy at datacenter level. As can be observed all VMs were created (Fig 6) with 2 VMs being created in the hosts residing in the dual core machines and 4 VMs in the hosts residing in the quad core machines. Next the hosts were failed using the same random number generator (Fig 7); the failures were this time spread over a longer time period but still randomly. Nonetheless, as seen from Fig 6 all the hosts failed during the simulation period. All the cloudlets were executed (Fig 8), while Fig 9 shows cloudlets executed over time. It was observed that in this instance all the cloudlets were executed and returned to the broker during the simulation; further, hosts that were running on quad cores took shorter time to execute cloudlets (2560 seconds) compared to those running on dual cores (2720 seconds). In this simulation VM migration was enabled at datacenter level. VM migration allows the VM to be migrated to a different host in the event of failure by some configured policy. As discussed in the literature [20] have used this technique in development of their VMware HA solution. The solution enables VMs to be migrated from a host in case it fails, to another host. There are two types of VM migration that are used: hot (live) migration and cold migration. When the former is used service will not be interrupted while with the latter users are likely to notice the service interruption. In live migration the VM is transferred together with its state [10]; this is the type of migration used in this simulation. This ensures that the jobs (cloudlets) in it are not affected and thus they are executed. The migration policy at datacenter level can be demonstrated as follows:

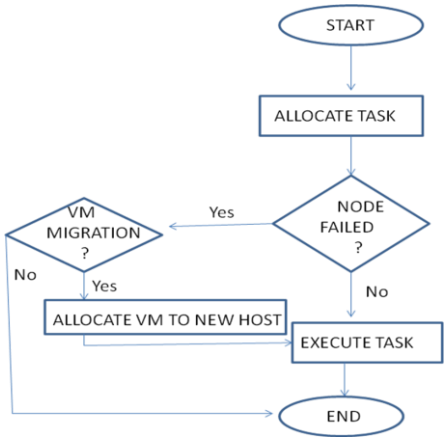

Fig 12 VM migration

The engine behind the cloud computing paradigm is actually virtualization more than anything else. This is because users are offered virtualized services at their level (SaaS), developers are offered the same (PaaS), and the infrastructure runs on hosts where VMs perform the actual job execution. While [10] describe the details of the two types of VM migration this study's focus was on whether node management and cluster management are indeed proactive approaches to countering node failures in the cloud. There are many ways to implement node management but for the purposes of this study we chose VM migration for the following reasons: when a node fails it is the VMs in it that will cause unavailability of the resources; VMs are the driving engines as far as execution of cloudlets is concerned and both VM migration policies as well as load balancing policies can be set proactively. Another approach to consider in managing nodes would be to investigate what causes the nodes to fail in the first place and build for availability from this point; however, this is beyond the scope of this study. Nodes may also be managed by using an active-passive node approach similar to the Linux HA approach mentioned in the literature. In terms of service availability if the nodes are defined as the resources then the pattern would be similar to that in simulation one; decreasing percentages of service availability. However, since the broker requests for VM creation depending on the tasks at hand, then the resource requested in this instance is the VM itself. Further since all VMs requested were available throughout the simulation then service availability is at 100%. In terms of execution availability all cloudlets sent were all executed and sent back to the broker.

Service availability = 100%; Execution Availability = 100%

Simulation 3 examined whether cluster management may also increase availability in cases of node failures. This simulation intended to examine whether cluster management can be used in place of node management to counter node failures. In this simulation the cluster was configured using dual core machines only as a cluster would require all machines to be of the same configuration (similar characteristics and operating systems); this makes it easier to manage and also to make it easier to distribute the workload among the machines that make up the cluster. The simulation run showed that while cloudlets were sent to the datacenter and random nodes failed the nodes completed cloudlet execution since they were communicating with each other to enable load balancing. The allocation policy was changed to time shared since running the simulation using same parameters as in simulation 2 resulted in errors in creation of VMs due to MIPS rating. The difference between time-shared and space-shared was described by [8] and these are both policies that are used to allocate tasks to VMs in the hosts. The main difference is that whereas both use the same round robin policy of task allocation described earlier, the latter schedules all tasks at the same time and allocates time slices to each task, while the former simply executes one task after another regardless of time. The errors that occurred were due to the fact that the machines used in the simulation were dual core and not quad core machines; thus by changing the allocation policy the VMs could be created. This study's scope was focused on whether cluster management does indeed increase availability proactively. By adjusting the policy the VMs could be created and the simulation was run successfully. The results indicate service availability and execution availability measures the same as for node management. The literature highlighted the high availability solution in High Availability Open Source Cluster Application Resource (HA-OSCAR) [18]; their solution using an enhancement of HA-OSCAR produced an availability of 0.99999. In HA-OSCAR the clusters kept executing tasks by ensuring that there was redundancy from the head node all the way down to the switches sending tasks to the clusters. This ensured that the tasks reached the clusters and that they were executed by the clusters but did not examine the possibility of nodes in the cluster failing. Further, even the HA solution provided by heartbeat described in the literature only envisions a redundant solution should a failure occur; the strength of this being in how quickly the changeover occurs. The cloud takes cluster computing to a new level by the introduction of cluster virtualization. This allows configuration of a cluster at both physical and virtual levels. The configuration of the cluster at both levels is referred to as cluster management. In the cloud however at infrastructure level this study focused on cluster management at the virtual level; the physical level configuration would involve parameters outside the scope of the study. From a user perspective the execution of tasks is done by VMs and what lies beyond them is not of concern to the user save for the fact that their tasks are executed. Virtualization is indeed one of the biggest strengths of the cloud.

The simulations suggest that proper node management does increase both service availability and execution availability. Further the results of the simulation using cluster management also indicate that this is an effective AM against node failure. Interestingly the results suggest that effective management of the individual node does result in effective management of the cluster itself, be it in a shared-nothing or shared-everything environment; further the converse also appears to be true, that is, effective management of the cluster results in effective management of the individual nodes as a whole. This implies that node failures can be effectively countered from an availability perspective using either node management or cluster management.

VI. CONCLUSION

This study purposed to address node failure in cloud computing environments at infrastructure level. The study proposed that by using node management or cluster management node failures can be addressed proactively. The node management technique that was used was VM migration; for cluster management VM migration was enabled such that the load could be

distributed using some policy so that when a node failed in the cluster its workload could be migrated to other nodes within the cluster. A series of simulations was performed using CloudSim simulator. The findings confirmed that both VM migration at node level, as well as migration vide load balancing across nodes, are effective techniques to use in countering node failures at infrastructure level.

Interestingly the results suggest that effective management of the individual node does result in effective management of the cluster itself, be it in a shared-nothing or shared-everything environment; further the converse also appears to be true, that is, effective management of the cluster results in effective management of the individual nodes as a whole. The study also averred that VM migration is not the only technique that can be used as a node management technique; for future research other techniques can also be investigated that lead to the nodes being managed more effectively and proactively in order to counter node failure. The study introduced a new availability parameter called execution availability which measures availability in terms of the actual tasks executed versus the tasks requested, and incorporated a component of time. We encourage the use of execution availability as a measurement parameter as was observed in this study that service availability by itself does not give an accurate measure of the availability of the infrastructure in terms of the actual tasks executed. For future research it is recommended to study what causes the node to fail in the first instance; further specific studies on AMs that will deal with node failures specifically will help ameliorate this outage cause, since it was observed in the literature that there is no writing at this time that specifically relates outage causes to specific AMs. Further as far as cluster management is concerned future studies can investigate how to increase availability using the cluster controllers themselves by more studies on intra-cluster relationships in the datacenter. This study is part of a wider one we are currently undertaking that aims to explore these relationships further.

REFERENCES

[1] Barr, J., & Narin, A. (2010). Building Fault-Tolerant Applications on AWS Failures shouldn't t be THAT Interesting. Retrieved from http://d36cz9buwru1tt.cloudfront.net/AWS_Building_Fault_Tolerant_Applications.pdf

[2] Bigelow, S. (2011). The causes and costs of data center system downtime: Advisory Board Q&A. In http://searchdatacenter.techtarget.com/feature/The-causes-and-costs-of-data-center-system-downtime-Advisory-Board-QA, accessed 11/08/14

[3] Buyya, R., Ranjan, R., & Calheiros, R. N. (2009). Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities. *Proceedings of the 2009 International Conference on High Performance Computing and Simulation, HPCS 2009*, 1–11. https://doi.org/10.1109/HPCSIM.2009.5192685

[4] Christophe, C., France, C. C., France, P. D., France, M. G., France, Q. G., & Guillaume, N. (2013). Downtime statistics of current cloud solutions, (June). *International Working Group on Cloud Computing Resiliency*, 2–4.

[5] Christophe, C., France, C. C., France, P. D., France, M. G., France, Q. G., & Laurent, S. (2014). Downtime Statistics of Current Cloud Solutions, (March). *International Working Group on Cloud Computing Resiliency* ,1–5.

[6] Das, P., & Khilar, P. (2013). LBVFT: A Load Balancing Technique for Virtualization and Fault Tolerance in Cloud Computing. *International Journal of Computer Applications*, *69*(28), 14–18

[7] Gagnaire, M., Diaz, F., Coti, C., & Cerin, C. (2012). Downtime statistics of current cloud solutions. *International Working Group on Cloud Computing Resiliency*, 2–3. Retrieved from http://iwgcr.org/wp-content/uploads/2012/06/IWGCR-Paris.Ranking-002-en.pdf

[8] Himani, H.S., & Sidhu, H. S. (2014). Comparative Analysis of Scheduling Algorithms of Cloudsim in Cloud Computing. *International Journal of Computer Applications*, *97*(16), 29–33.

[9] Jose, A.T. (2013). Benchmarking Service Availability for Cloud Computing. *IOSR Journal of Engineering*, *3*(8), 01–03. https://doi.org/10.9790/3021-03860103

[10] Kaur, P., & Rani, A. (2015). Virtual Machine Migration in Cloud Computing. *International Journal of Grid Distribution Computing*, *8*(5), 337–342.

[11] Myerson, J. M. (2013). Mitigate risks of cloud resource exhaustion outages Use service level agreements and other proactive tools to avoid, IBM developerworks, 1–9.

[12] Nagpal, S., Shivam and Kumar, P. (2013). A Study on Adaptive Fault Tolerance in Real Time Cloud Computing. *International Journal of Advanced Research in Computer Science and Software Engineering*, *3*(3), 246–248.

[13] Ranjithprabhu, K., & Sasirega, D. (2014). Eliminating Single Point of Failure and Data Loss in Cloud Computing. *International Journal of Science and Research (IJSR)*, *3*(4), 335–337.

[14] Ritter, E.F., Schoelles, M.J., Quigley, K.S., Klein, L. C. (2011). Determining the number of simulation runs: Treating simulations as theories by not sampling their behavior. *Human-in-the-Loop Simulations: Methods and Practice*, 97–116. https://doi.org/10.1007/978-0-85729-883-6

[15] Rohani, H & Roosta,A.K.(2014). Calculating Total System Availability, *Information Services Organization Amsterdam*, 2014.

[16] Stanik, A., Hoger, M., & Kao, O. (2013). Failover Pattern with a Self-Healing Mechanism for High Availability Cloud Solutions. *2013 International Conference on Cloud Computing and Big Data*, 23–29. doi:10.1109/CLOUDCOM-ASIA.2013.63

[17] Tchana, A., Broto, L., & Hagimont, D. (2012). Fault Tolerant Approaches in Cloud Computing Infrastructures. *ICAS 2012, The Eighth International Conference on Autonomic and Autonomous Systems pp 42-48*

[18] Thanakornworakij, T., Sharma, R., Scroggs, B., Leangsuksun, C., Greenwood, Z. D., Riteau, P., & Morin, C. (2012). High availability on cloud with HA-OSCAR. Euro-Par 2011: Parallel Processing Workshops, 7156 LNCS(PART 2), 292–301. https://doi.org/10.1007/978-3-642-29740-3-33

[19] Weygant, P.S. (2001). *Clusters for High Availability: A Primer of HP Solutions*. 2001: Prentice Hall Professional.

[20] VMware. (2007). *VMware High Availability: Concepts, Implementation and Best Practices*, VMWare, Inc.

