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. 6, June 2018, pg.76 – 93

Managing Configuration Errors in Cloud Computing Using Cluster & Node Management Techniques

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 technology of choice for most users globally due to the numerous advantages that it offers. This technology is offered to users with different levels of expertise who utilize the different services available in the cloud. At infrastructure level, however, configuration errors are some of the major causes of unavailability of services to users. The objective of this study was to evaluate whether cluster management and node management could effectively be used to manage configuration errors proactively at cloud infrastructure level. The literature indicates that most configuration errors occur due to human error, namely parametric configuration errors. Some human errors at infrastructure level were simulated using CloudSim simulation toolkit. The availability of the infrastructure was then measured using two availability parameters, namely, service availability and execution availability. The simulations showed that the injected errors resulted in a partial or complete loss of services to users. The correct configurations were then simulated to show the difference, and suggestions are made as to how these errors may be prevented proactively in the first place. The findings showed that cluster management is an effective tool in preventing configuration errors as long as those errors occur at cluster level; further node management is an effective tool in countering configuration errors, as long as the errors occurred at node level. It would thus be more desirable to use cluster management techniques to counter errors at both node and cluster level as the latter management technique presents management at a more abstract level.*

***Keywords:** cloud computing, infrastructure, configuration errors, cluster management, node management, service availability, execution availability.*

I. INTRODUCTION

Cloud computing is a technology that has been embraced in all corners of the globe. The technology is provided by Cloud Service Providers (CSPs) who offer their services using the XaaS (X-as –a – Service) paradigm, where X represents the different service categories offered by the CSP. The most common service categories are Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). The three services are offered at different layers and can be represented as follows:

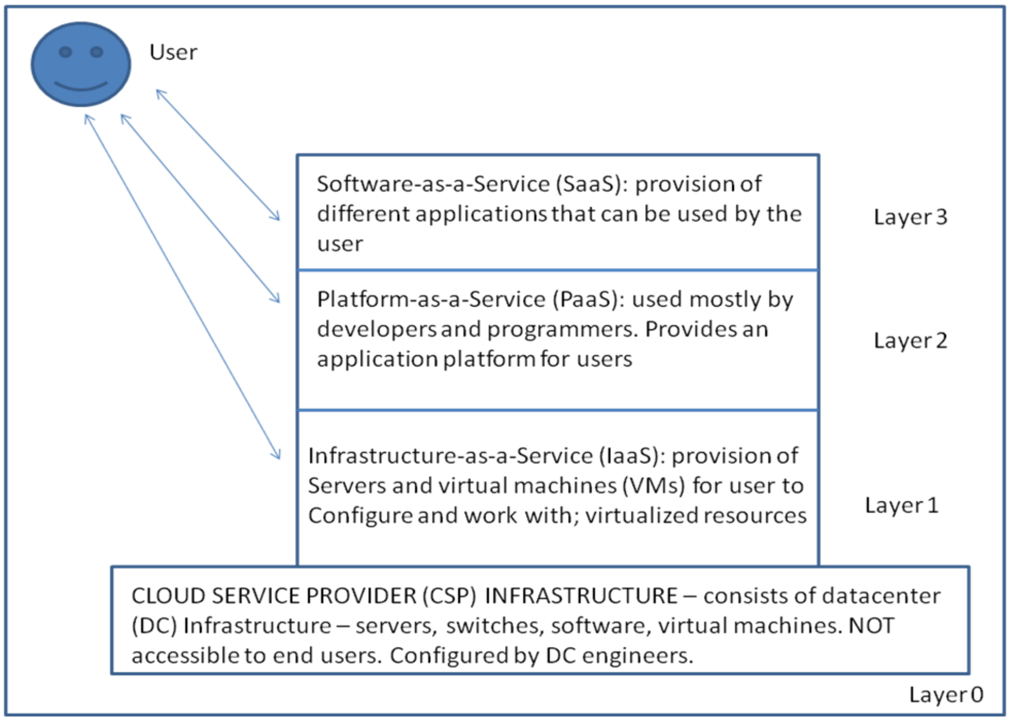
User
Software-as-a-Service (SaaS): provision of different applications that can be used by the user
Layer 3
Platform-as-a-Service (PaaS): used mostly by developers and programmers. Provides an application platform for users
Layer 2
Infrastructure-as-a-Service (IaaS): provision of Servers and virtual machines (VMs) for user to Configure and work with; virtualized resources
Layer 1
CLOUD SERVICE PROVIDER (CSP) INFRASTRUCTURE – consists of datacenter (DC) Infrastructure – servers, switches, software, virtual machines. NOT accessible to end users. Configured by DC engineers.
Layer 0

Fig. 1 Service categories in the Cloud

Each category offers different services to the user, depending on needs. Layer 0 is the DC itself and consists of the whole physical and logical infrastructure that provides and supports the services offered to cloud computing users; this layer is not available to end users and is configured at DC level by hardware, software, and network engineers. Layer 1 is for advanced users, and allows them to configure virtual resources that they can use for different services. Layer 2 is used by both advanced and users with sufficient skills, as an application platform. At this level users can develop and test programs and applications before using them in a live environment. Layer 3 is the category used by the greater Internet population and consists of different applications that are used on a daily basis by users (both individuals and organizations). At layer 3 one will find simple applications like spreadsheets and word processing packages provided by CSPs like Google. Further, cloud services are deployed using models, there being four main deployment models:

- Public clouds: these are clouds that are available to the general population for use. They are available for free, at a fee, or a mix of both, depending on the CSP providing the model. A good example of a public cloud SaaS that is available for free are the spreadsheet services offered on the Google cloud.

- Private clouds: these are clouds that are for the exclusive use of an organization. They are not available to the public and are normally supported and configured in-house; in some instances they are hosted by a third party.

- Hybrid clouds: these are a mix of private and public cloud. Typically an organization will have developed a private cloud, and only reaches out to the public cloud as and when the need arises. The concept of reaching out to the public cloud is also known as cloud bursting.

- Community cloud: these are clouds developed for use by communities/users or organizations who share common interests. They collaborate on such a platform. The community agrees on who will host the cloud, either one of the members of the community or a third party.

The five pillars of information assurance were described by [14]. Of the five pillars, availability has been one of the major concerns for users. The availability of services in the cloud is a key factor in determining the success or failure of that CSP. However, availability over the years has not been satisfactory to end users, and the big CSPs are not spared either. The hours and costs involved with these outages are discussed in a separate paper; further, the different outage causes at layer 0 were also discussed [5]. In this study configuration errors were examined as an outage cause. Using CloudSim simulation toolkit six simulations were performed focused on layer 0 in fig 1. Four simulations showed the effect of parametric configuration errors on the infrastructure and task execution process, while two simulations had the errors corrected in their respective simulated environments. The results were then discussed and conclusions drawn there from. The rest of this paper is divided as follows: in section 2 the current literature on the different types of configuration errors is examined. Availability is also described in this section, as are availability mechanisms (AMs). Section 3 describes the methodology that was used to perform the simulations. Section 4 exhibits the results of the simulations, while section 5 analyses them. Section 6 presents the conclusions and directions for further research.

II. LITERATURE REVIEW

In this section availability is defined first, followed by an examination of the different types of configuration errors. Finally a discussion on how these errors have been managed hitherto.

A. *Availability*

Techopedia.com 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” [13]. Three different types of availability were defined by [8], namely operational, inherent and achieved. [3] went further to introduce a measure of availability called service availability (SA). Service availability is defined as the ratio of the resources allocated to the resources requested. In a separate paper a new measure for availability, namely execution availability, was introduced (EA) [4]. It was posited that SA alone cannot give a satisfactory measurement of availability of the infrastructure as a whole, but together with EA, they can both give a better understanding of the availability of the infrastructure. EA was defined as the ratio of the tasks executed to the tasks requested, with a component of time. These two measurement parameters were the ones used in measuring availability of the infrastructure in this particular study.

B. Configuration Errors

A configuration issue arises when any of the resources becomes unavailable due to a change in some policy or a failure of a component without a fault tolerant policy in place. Yin et al. (as cited in [15]) classified software configuration errors into the following three categories:

- Parameter: erroneous settings of configuration parameters (either an entry in a configuration file or a console command);
- Compatibility: configuration errors related to software incompatibility;
- Component: the other remaining configuration errors (for example, missing a specific software module).

Further [15] posited that most of the existing research efforts focus on parameter configuration errors, because they account for the majority of real-world configuration errors (the authors cited, for example, 70.0%–85.5% of the studied configuration error cases in Yin et al.). Most of these errors are caused by humans. [15] also pointed out that configuration errors are some of the most dominant causes of outages and downtime. Barroso and H¨olzle report that configuration errors were the second major cause of the service-level failures at one of Google's main services (as cited in [15]). [6] reported that configuration errors were the dominant cause of Hadoop cluster failures, in terms of both the number of customer cases and the supporting time.

C. Availability Mechanisms (AMs)

Availability mechanisms are the different approaches that have been used by the cloud computing industry (both academia and corporate) in increasing availability of the cloud. As was observed in [4] most service providers utilize either some form of redundancy, or build for fault tolerance in their infrastructure ([2], [9], [10], [11], [12]). Nonetheless, [15] go further to suggest that hardening systems against configuration errors and building configuration free systems would help ameliorate configuration errors. However, the biggest stumbling block has been that none of these AMs has been built to deal with specific outage causes in the cloud; they are at best, generic mechanisms. This study theorized that node management and cluster management may be used in handling parametric configuration errors in the cloud. This is described in detail in the next section.

III. METHODOLOGY

In this section we describe the methodology used in the study. The tool that was used is CloudSim, a discrete, deterministic simulation toolkit developed by [1]. The simulator allows for configuration of different scenarios, and has few limitations depending on the design of the simulation. In this case simulations and scenarios were developed at layer 0 (fig 1). CloudSim allows for configuration of scenarios in Java programming language; this means that knowledge of Java is required in order to properly configure required scenarios. The simulator has inbuilt classes some of which can be extended in order to build scenarios. It also has inbuilt policies and users can opt to use these or create their own policies to suit their needs. Users configure their scenarios and a broker sends the tasks (called cloudlets) to the DC which will act according to the programmed configuration and policies. The sequences of events that occur when a user sends a request (cloudlet) to the DC are described in [4].

A. *Host Configuration*

The simulator was run on a host machine, in this case a HP laptop. The configuration of the host environment is given below:

TABLE I
HOST MACHINE CONFIGURATION

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

Each simulation was done in line with [7] who stated that in a deterministic simulation only one simulation is enough to generate valid predictions. There were six scenarios that were configured.

B. *Simulations*

There were two groups of simulations done based on six different configurations; essentially, each group had three simulations. Three different configurations were used for node management; similarly, three different configurations were used for cluster management. There is need to differentiate between the different levels of configuring a simulation scenario when using the CloudSim simulation toolkit. Configurations can be done at three different levels:

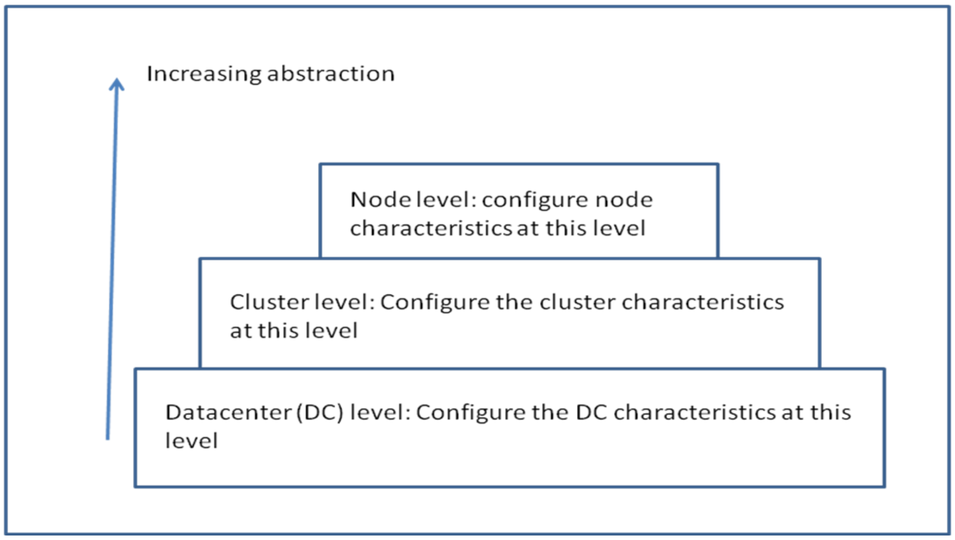
Increasing abstraction
Node level: configure node characteristics at this level
Cluster level: Configure the cluster characteristics at this level
Datacenter (DC) level: Configure the DC characteristics at this level

Fig. 2 CloudSim configuration layers

Each layer is more abstract than the layer below it. At DC level the configuration of the datacenter as a whole can be configured, while at cluster level the characteristics of the cluster are similarly configured; ditto node level.

1) *Cluster level simulations:* There were two simulations performed that showed the effect of configuration errors at CSP infrastructure level, while a third simulation was run where the error was corrected. The simulations were based on configuration parameters at the cluster level.

Simulation 1: in this simulation an erroneous setting of PE(Processing Element)configuration settings in the machines in the datacenter was done by intentionally leaving out two zeros in the MIPS (Millions of Instructions Per Second) processing power such that the MIPS = 10 in the private static Datacenter createDatacenter(String name) method. This is the method that is used to create the machines in the cluster and their characteristics in the datacenter; all other configurations were left unchanged. Tasks were then sent to the datacenter and the number of cloudlets executed was observed and recorded.

```
//Configuration error pseudocode
Create datacenter
Configure datacenter characteristics, //configure the characteristics for the datacenter
Hosts = 10, VMs = 30, PE = 2 (dual core servers in the datacenter), RAM = 16384 MB,
MIPS = 10 //this is the configuration error at cluster level
Complete configuration
Broker: send cloudlets
```

TABLE II
SIMULATION 1 CONFIGURATION

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

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

Method: createDatacenter (String name); VMAllocation Policy: time – shared

Simulation 2: in this procedure the configuration error was adjusted and MIPS set to 250 and cloudlet execution observed and recorded while maintaining the remaining configurations of simulation 1.

```
//Configuration error pseudocode
Create datacenter
Configure datacenter characteristics, //configure the characteristics for the datacenter
Hosts = 10, VMs = 30, PE = 2 (dual core servers in the datacenter), RAM = 16384 MB,
MIPS = 250 //this is the configuration error at cluster level
Complete configuration
Broker: send cloudlets
```

TABLE III
SIMULATION 2 CONFIGURATION

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

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

Method: createDatacenter (String name); VMAllocation Policy: time – shared

Simulation 3: in this procedure the configuration error was corrected and MIPS set to 1000 and cloudlet execution observed and recorded while maintaining the remaining configurations of simulation 1.

```
//Configuration error pseudocode
Create datacenter
Configure datacenter characteristics, //configure the characteristics for the datacenter
Hosts = 10, VMs = 30, PE = 2 (dual core servers in the datacenter), RAM = 16384 MB,
MIPS = 1000 //this is the configuration error corrected at cluster level
Complete configuration
Broker: send cloudlets
```

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;
				
				//Kbit/sec
					30
					
					size = 10000; //image size (MB)
					
					RAM = 512; //vm memory (MB)
					
					MIPS = 250;
					
					BANDWIDTH = 1000;
					
					//Kbit/sec
					
					pesNumber = 1; //number of cpus
					
					VMM = "Xen";
					
					//VM Manager
						500
						
						length = 40000;
						
						//in MI
						
						fileSize = 300;
						
						//in bytes
						
						outputSize = 300
						
						//in bytes

Method: createDatacenter (String name); VMAllocation Policy: time – shared

2) *Node level simulations:* There were three simulations performed to show the effect of configuration errors at node level. In the first two simulations parametric configuration errors were configured, while in the third simulation the configuration error was corrected.

Simulation 1: In this simulation an erroneous setting of RAM configuration settings in the machines in the datacenter was done by setting RAM = 163 in the private static Datacenter createDatacenter (String name) method. This is the method that is used to create the machines and their characteristics in the datacenter; all other configurations were left unchanged. Tasks were then sent to the datacenter and the number of cloudlets executed was observed and recorded.

```
//Configuration error pseudocode
Create datacenter
Configure datacenter characteristics, //configure the characteristics for the datacenter
Hosts = 10, VMs = 30, PE = 2 (dual core servers in the datacenter), MIPS = 1000,
RAM = 163 MB//this is the configuration error at node level
Complete configuration
Broker: send cloudlets
```

TABLE V
SIMULATION 1 CONFIGURATION

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

1	5	2 - for dual core machines	1000	10
				
				ram = 163; //host memory (MB)
				
				long storage = 1000000; //host storage
				
				bw = 10000;
				
				//Kbit/sec
					30
					
					size = 10000; //image size (MB)
					
					RAM = 512; //vm memory (MB)
					
					MIPS = 250;
					
					BANDWIDTH = 1000;
					
					//Kbit/sec
					
					pesNumber = 1; //number of cpus
					
					VMM = "Xen";
					
					//VM Manager
						500
						
						length = 40000;
						
						//in MI
						
						fileSize = 300;
						
						//in bytes
						
						outputSize = 300
						
						//in bytes

Method: createDatacenter (String name); VMAllocation Policy: time – shared

Simulation 2: The configuration error was adjusted and RAM set to 512 MB for each machine and the resulting simulation results recorded.

```
//Configuration error pseudocode
Create datacenter
Configure datacenter characteristics, //configure the characteristics for the datacenter
Hosts = 10, VMs = 30, PE = 2 (dual core servers in the datacenter), MIPS = 1000,
RAM = 512 MB//this is the configuration error at node level
Complete configuration
Broker: send cloudlets
```

TABLE VI
SIMULATION 2 CONFIGURATION

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

1	5	2 - for dual core machines	1000	10
				
				ram = 512; //host memory (MB)
				
				long storage = 1000000; //host storage
				
				bw = 10000;
					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;
						
						fileSize = 300;
						
						outputSize = 300

Method: createDatacenter (String name); VMAllocation Policy: time – shared

Simulation 3: The configuration error was adjusted and RAM set to 16384 MB for each machine and the resulting simulation results recorded.

```
//Configuration error pseudocode
Create datacenter
Configure datacenter characteristics, //configure the characteristics for the datacenter
Hosts = 10, VMs = 30, PE = 2 (dual core servers in the datacenter), MIPS = 1000,
RAM = 16384 MB//this is the where configuration error had been set at node level
Complete configuration
Broker: send cloudlets
```

TABLE 7
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;
					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;
						
						fileSize = 300;
						
						outputSize = 300

Method: createDatacenter (String name); VMAllocation Policy: time – shared

IV. RESULTS

This section presents the results of the simulations. There were six simulations performed in total, divided between two groups. The first three simulations were with regard to cluster management, while the last three were with regard to node management.

A. Cluster level simulations

Simulation 1 (MIPS = 10):

In this simulation requests for VMs were not fulfilled, i.e. VM creation failed in all the hosts
Allocation of VM failed by MIPS

Simulation 2 (MIPS = 250):

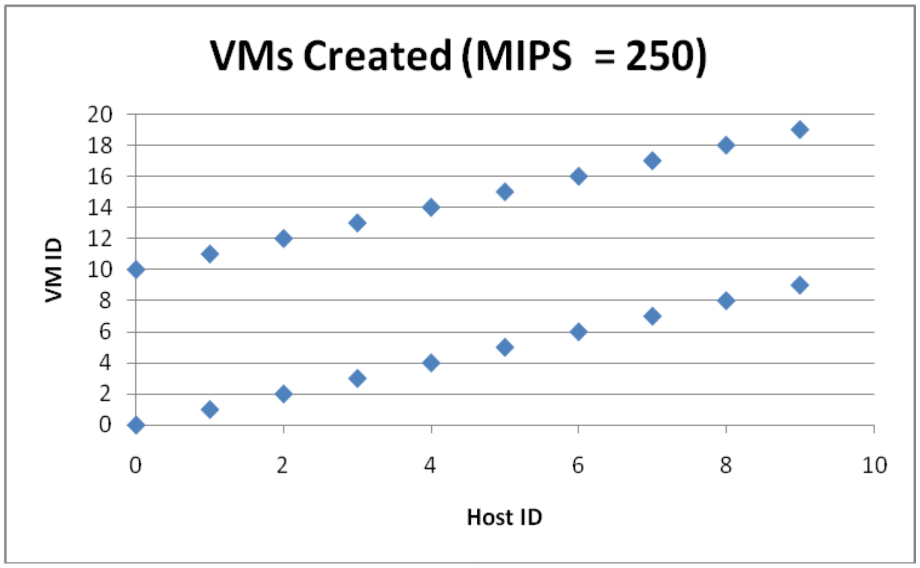

Fig. 3 VMs created (MIPS = 250)

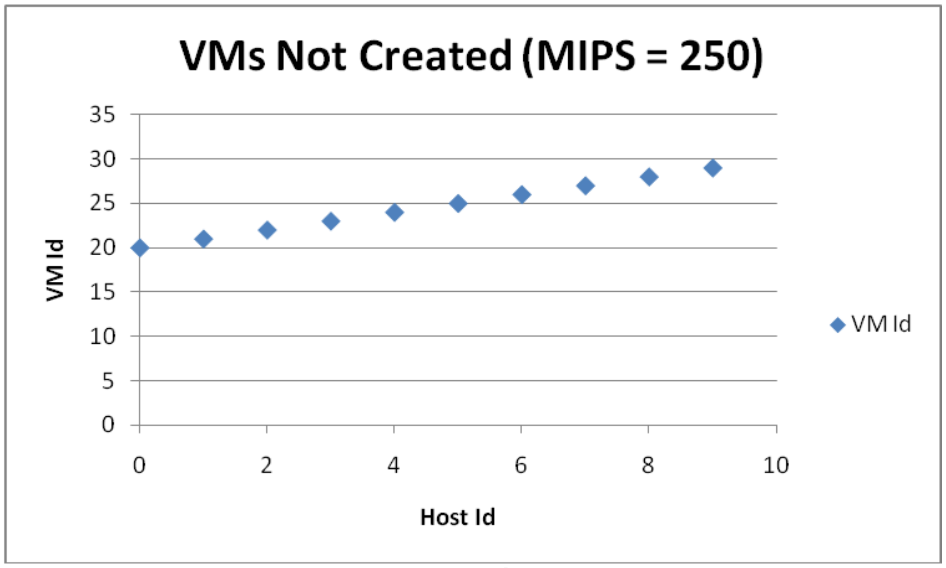

Fig. 4 VMs not created (MIPS = 250)

- VMs created 20
- VMs failed: 10
- Cloudlets executed: ALL
- Average execution time: 4000 seconds

Simulation 3 (MIPS = 1000):

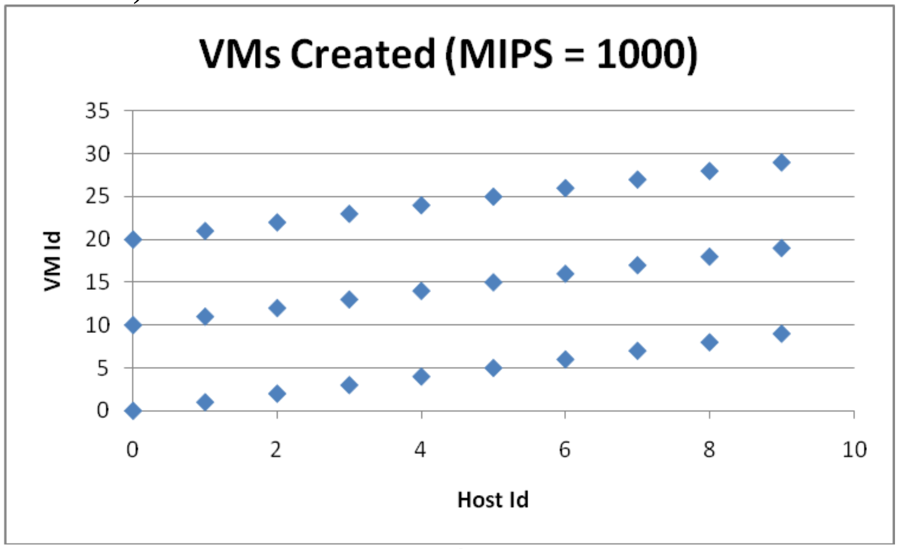

Fig. 5 VMs created (MIPS = 1000)

- All VMs were created
- Average execution time: 2640 seconds
- Cloudlets executed: ALL

B. Node level simulations

Simulation 1 (RAM = 163 MB):
In this simulation requests for VMs were not fulfilled, i.e. VM creation failed in all the hosts
Allocation of VM failed by RAM

Simulation 2 (RAM = 512 MB)

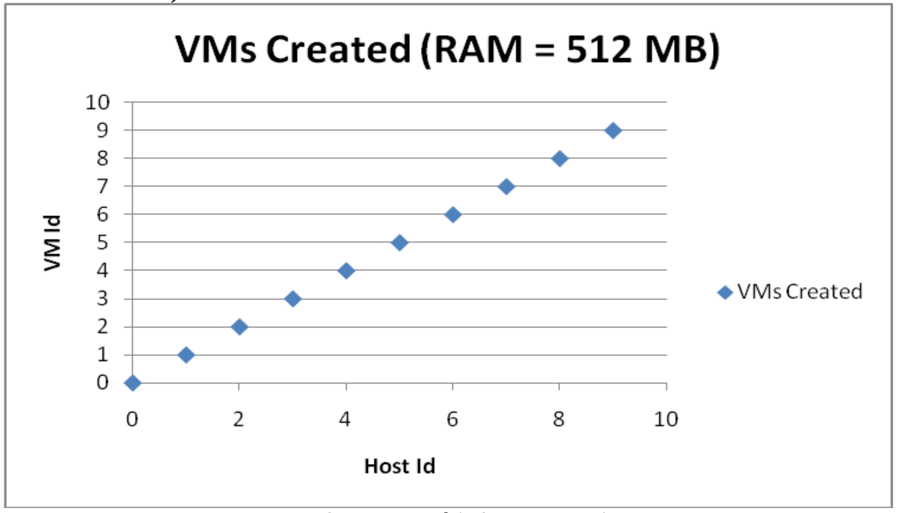

Fig. 6 VMs created (RAM = 512 MB)

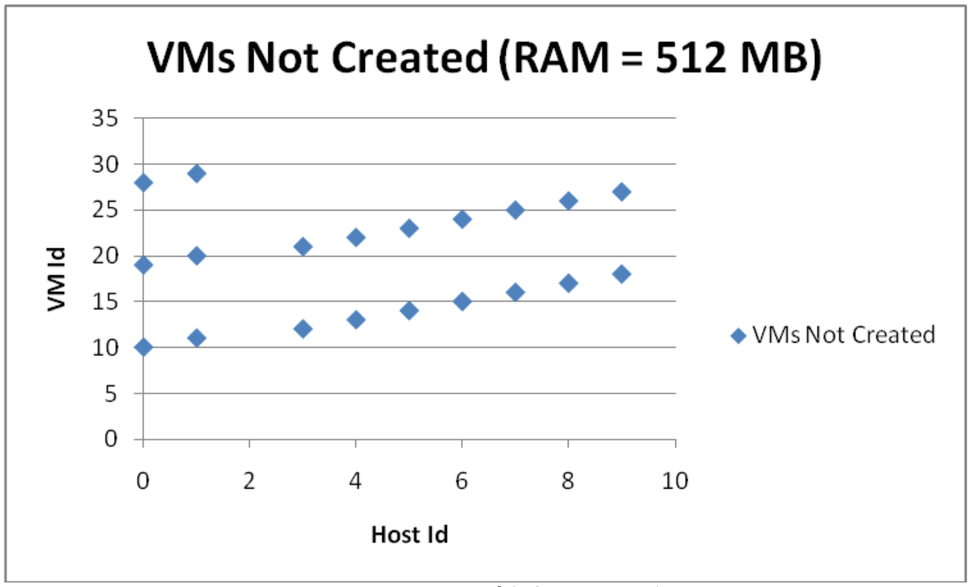

Fig. 7 VMs not created (RAM = 512 MB)

- VMs created 10
- VMs failed: 20
- Cloudlets executed: ALL
- Average execution time: 8000 seconds

Simulation 3 (RAM = 16384 MB)

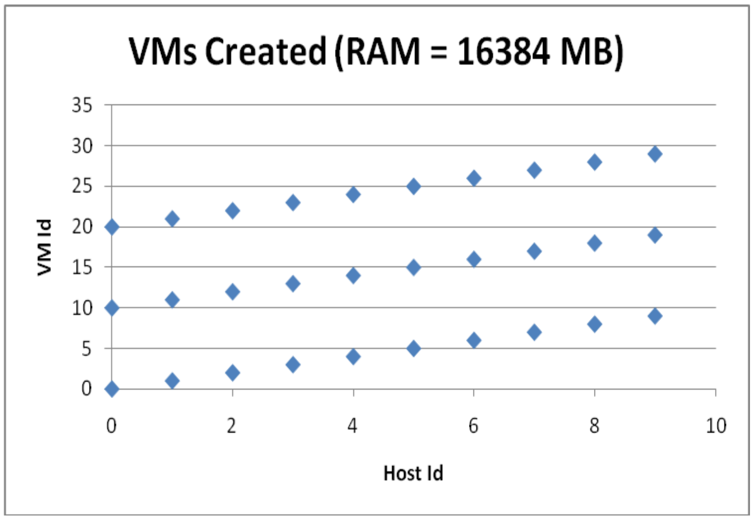

Fig. 8 VMs created

- VMs created 30
- VMs failed: 0
- Cloudlets executed: ALL
- Average execution time: 2640 seconds

V. DISCUSSION

This section analyses the results of the simulations from section four. This study sought to investigate the effect of configuration errors on the infrastructure's ability to provide services, i.e. availability. In this section an analysis of the results is offered, specifically from an availability perspective. Before examining the two sets of simulations, the availability parameters used to measure the availability of the infrastructure were defined:

As described in the literature, [3] defined service availability (SA) as the ratio of the resources allocated to the resources requested:

$SA = R_A/R_R$, where R_A = Resources allocated, and R_R = Resources requested.

A second availability measurement parameter, execution availability (EA) was defined as the ratio of the tasks executed to the tasks requested, with a component of time [4]:

Let the number of cloudlets executed = μ;
Let the time taken to execute the tasks = t;
Let the number of cloudlets requested = λ;

$EA = \mu / \lambda t$, where t is measured in seconds.

A. Cluster level simulations

Configuration errors were identified in the literature as outage causes. Human errors inevitably produce these configuration errors. The simulations show the extent to which configuration errors may cause lack of availability of the cloud infrastructure.

In the first simulation simply by configuring an MIPS of 10 both the service and execution availability becomes zero percent. This is because the processing power of the machines making up the cluster is too low to create any VM.

In the second simulation the MIPS was increased to 250. The following was observed:

$SA = R_A/R_R = 20/30 = 66.67\%$ {Resource = VM in this case, measured at the end of the simulation}
$EA = \mu/\lambda t = 500/ (500 * 4000) = 0.00025$ {t = 4000};

In the third simulation the MIPS was further increased to 1000. The following was observed:

$SA = R_A/R_R = 30/30 = 100\%$ {Resource = VM in this case, measured at the end of the simulation}
$EA = \mu/\lambda t = 500/ (500 * 2640) = 0.00038$ {t = 2640};

As is observed with the configuration error in simulation 2, the service availability by the end of the simulation was 66.67%; however, it is inferential that the SA was 66.67% from the point when the broker sent the cloudlets since the 10 VMs were not created in the first place. The EA for simulation 2 was also measured at 0.00025; as proposed earlier this measure is comparative though the higher it is the better. In the third simulation the SA was measured at 100% as all the VM resource was available throughout the simulation since all VMs were created. Further a significant improvement on EA was observed, rising to 0.00038. The significance of this to a user is in the time s/he takes to receive results of their tasks. The service provider needs to ensure that the infrastructure is fast if they are to retain users who would not like to tap their fingers on a desk waiting for the results of a task.

In this instance the configuration issue has been caused by a typo due to human error, as most configuration errors are. The contention that arises is whether the error was part of the cluster configuration, and if so how could it be prevented in the first instance? As it stands, the error was only observed during the output in both simulations 1 and 2 and correcting it was a reactive rather than a proactive measure. The study purposed to increase availability from a proactive perspective and therefore proactive measures are more desirable. However, it is worth noting that configuration errors are some of the hardest errors to deal with since a configuration error in one setting may destabilize more than one area of the infrastructure; for example, performance degradation may require investigating more than the MIPS settings of the infrastructure. In the literature [15] had suggested the use of configuration free systems where users do not have to key in any configurations; this is rather restrictive and does not allow infrastructure engineers and designers to make their own configurations. The most practical approach to this would be in restricting input when configuring the cluster during installation, thus making it impossible for erroneous configuration entries. This may be done by use of input masks and/or validation alerts in the configuration scripts and programs at user interface level. Practically, an input mask would restrict values that the user enters when configuring the cluster, for example, a code may be put in place that restricts the MIPS rating for the cluster to be not less than say, 1000. The validation alert would tell the user that that the value they have entered is invalid, and suggest the range of values valid for that particular parameter setting. To make it even more restrictive infrastructure engineers may also opt not to allow users to enter certain parameters and make these settings part of an input script to be picked directly by the server; unfortunately, this too isn't free of human error completely, making the input mask a more practical solution.

It is not conclusive as to whether cluster management is an effective AM for configuration issues; however, the simulations show that it is an effective AM if the configuration error occurred in the cluster at cluster level. It is cognizable that the majority of configuration errors will occur somewhere in the cluster as this is where the machines, hosts and VMs are to be found and these are the key components of the infrastructure. Still, if the error occurs outside of the cluster say, at the datacenter level then cluster management becomes ineffective as an AM.

B. Node level simulations

The first two simulations show how configuration errors affect availability of the cloud:

In simulation 1 by erroneously setting RAM at 163 instead of 16384 at node level it is observed that no allocation of VMs occurs and thus there is no execution of tasks at all. This is due to the inability of the cores to generate enough memory for the VMs to be created by the hosts in the datacenter. User tasks are thus not executed and the broker is not able to send any of the tasks to the hosts.

In simulation 2 there is partial creation of VMs when the RAM at the cores is set at 512 MB. Availability parameters are observed as follows:

$SA = R_A/R_R = 10/30 = 33.33\%$ {Resource = VM in this case, measured at the end of the simulation}

$EA = \mu/\lambda t = 500/\ (500 * 8000) = 0.000125$ {$t = 8000$};

In the third simulation the RAM was further increased to 16384 MB. The following was observed:

$SA = R_A/R_R = 30/30 = 100\%$ {Resource = VM in this case, measured at the end of the simulation}

$EA = \mu/\lambda t = 500/\ (500 * 2640) = 0.00038$ {$t = 2640$};

As is observed in simulation 2 the SA at the set RAM of 512 MB is very low at 33.33% while the EA is also very low at 0.000125. When the correct RAM configuration of 16384 MB is set then SA rises to 100% while there is a great improvement in EA at 0.00038; suffice to say that if the RAM parameters are set higher at the nodes then it would be expected that EA would improve significantly (the higher the better). The observations of the previous section apply here. It is desirable to prevent configuration errors from occurring in the first place. Node management may be used to counter configuration errors, but only at node level. The parameter settings at node level are restricted to the cores themselves and the hosts created in each core. This is a more abstract level of the infrastructure compared to the cluster level. In a typical datacenter configuration settings would typically be at three levels: the datacenter, the clusters and finally the nodes in the clusters (fig 2). In the previous section it had been concluded that cluster management may counter configuration issues but only up to the cluster level. From the simulations above it may also be inferred that node management may be used to counter configuration errors, but only at node level. Further as nodes make up the cluster it is more desirable to use cluster management to proactively manage configuration errors as opposed to node management; we may thus conclude the node management is not a viable alternative to cluster management in proactively countering configuration errors.

VI. CONCLUSIONS

It is not conclusive as to whether cluster management is an effective AM for configuration issues; however, the simulations show that it is an effective AM if the configuration error occurred in the cluster at cluster level. Further, node management may be used to counter configuration errors, but only at node level. It is therefore more desirable to tackle configuration errors at the cluster level as opposed to the node level, since the latter is a more abstract level. At cluster level configuration errors that may possibly occur at node level can be prevented proactively.

The study also observed that the results indicate that cluster management is an effective AM if the configuration errors occurred at cluster level; this was the same case for node level. Further investigation will be needed in order to study the effect of cluster level configuration errors (at both node and cluster level) on other parts of the infrastructure, as this is what has made the study not hundred percent conclusive. These errors at both levels can then be classified even further and experiments performed on each classification in order to identify the best AM(s) for each identified area.

REFERENCES

[1] 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

[2] Hauck, M., Huber, M., & Klems, M. (2010). Challenges and opportunities of cloud computing. Karlsruhe Institute of Technology Technical Report, 2010, 31. Retrieved from http://digbib.ubka.uni-karlsruhe.de/volltexte/documents/1978786

[3] 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

[4] Mbogholi, M. J., Okoyo, H. O., & McOyowo, O.S.J. (2018). Addressing Node Failures Using Node Management & Cluster Management Techniques in Cloud Computing. International Journal of Computer Science and Mobile Computing, 7(4), 27–41.

[5] Msagha, M. J., Okoyo, H.O., & McOyowo, O.S.J. (2015). A Review of Availability Mechanisms in Dynamic Cloud Computing Environments. International Journal of Engineering Research & Technology, 4(5), 653–658.

[6] Rabkin, A., & Katz, R. H. (2013). How Hadoop Clusters Break. IEEE software, 30(4), 88-94.

[7] 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

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

[9] 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

[10] Thanakornworakij, T., Sharma, R, Blaine. S, Chokchai, L., Zeno, D. G., 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

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

[12] Weissman, J., & Ramakrishnan, S. (2009). Using Proxies to Accelerate Cloud Applications. HotCloud '09 Workshop in Conjunction with USENIX Annual Technical Conference, 20. Retrieved from http://portal.acm.org/citation.cfm?id=1855553

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

[14] Wilson, K.S. (2013). Conflicts among the Pillars of Information Assurance. IT Professional , 15(4), 44-49.

[15] Xu, T., & Zhou, Y. (2015). Systems Approaches to Tackling Configuration Errors: A Survey. ACM Computing Surveys, 47(4), 1–41. https://doi.org/10.1145/2791577