Study the Throughput Outcome of Desktop Cloud Systems Using DesktopCloudSim Tool

Abdulelah Alwabel, Robert Walters, Gary Wills
School of Electronics and Computer Science
University of Southampton
Southampton, UK
e-mail: {aa1a10, rjw1, gbw}@ecs.soton.ac.uk

Abstract— **Desktop Cloud computing is a new type of Cloud computing that aims to provide Cloud services at little or no cost. This ambition can be achieved by combining Cloud computing and Volunteer computing into Desktop Clouds, harnessing non-dedicated resources when idle. However, Desktop Cloud systems suffer from the issue of node failures. Node failure can happen without prior notification, which may affect the throughput outcome of these systems. This paper studies the impact of node failures using a simulation tool. Simulation tools are commonly used by academics and researchers to simulate Clouds in order to investigate various research issues and examine proposed solutions. CloudSim is a well-known and widely employed tool to simulate Cloud computing by both academia and industry. However, CloudSim lacks the ability to simulate failure events, which may occur to physical nodes in the infrastructure level of a Cloud system. In order to show the effectiveness of DesktopCloudSim, we evaluate the throughput of two types of Desktop Clouds: private and public Desktop Clouds that are built on top of faulty nodes based on empirical data sets. The data sets are analysed and studied in this paper to reflect the number of node failures in these two Cloud types. The evaluation process serves two purposes: the first is that it validate the working of the proposed tool. The second is to show that throughput of Desktop Cloud systems is affected badly by node failures.**

Keywords-Cloud; CloudSim; DesktopCloudSim; Failure; Nodes; Throughput; VM Allocation.

I. INTRODUCTION

DesktopCloudSim [1] is proposed in our previous paper as an extension tool that can simulate node failures in Cloud system. Cloud computing has emerged with a promise to improve performance and reduce running costs. The services of Cloud computing are provided by Cloud service providers (CSPs). Traditionally, CSPs use a huge number of computing resources in the infrastructure level located in datacentres. Such resources are claimed to have a high level of reliability, which makes them resilient to failure events [2]. However, a new direction of Cloud has recently emerged with an aim to exploit normal Desktop computers, laptops, etc. to provide Cloud services [3]. This kind of Cloud can be called Desktop Clouds [4]. In contrast to the traditional way of CSP, which uses a huge number of computing resources that are dedicated to be part of the Cloud. Throughout this paper, the term Traditional Cloud refers to this traditional way of Clouds.

The cost-effectiveness of Desktop Clouds is the key advantage over Traditional Clouds. Researchers in Desktop Clouds can use Cloud services at little cost, if not free. However, such feature suffers from an issue. The nodes of a Desktop Cloud are quite volatile and prone to failure without prior knowledge. This may affect the throughput of tasks and violate the service level agreement. The throughput is defined as the number of successful tasks submitted to be processed by virtual machines (VMs). Various VM allocation mechanisms can yield different variations of throughput level in the presence of node failures.

VM allocation mechanism is the process of allocation requested VMs by Cloud's users to physical machines (PMs) in the infrastructure level of a Cloud. The main goal of this paper is to study the impact of node failures on the outcome of Desktop Cloud systems. The contribution of this paper can be summarised into: (i) it proposes and describes the DesktopCloudSim as being an extension for CloudSim simulation toolkit; (ii) it investigates the impact of failure events on throughput and (iii) three VM mechanisms: FCFS, Greedy and RoundRobin mechanisms are evaluated in terms of throughput using DesktopCloudSim. The reminder of this paper is organised as follows: Section II discusses Desktop Cloud as being a new direction of Cloud computing. Section III proposes the simulation tool that extends CloudSim. The section starts by reviewing CloudSim to show the need to extend it. The section, then, reviews some VM allocation mechanisms. Next section demonstrates experiments conducted to evaluate the impact of node failures in a private Desktop Cloud based on empirical data of failures in NotreDame nodes. Another simulation of public Desktop Cloud is conducted using data of SETI@home nodes. The results are then analysed and discussed in Section V. Several related works are reviewed in Section VI. Finally, a conclusion and future work insights are given in the last section.

II. DESKTOP CLOUD

The success of Desktop Grids stimulates the idea of harnessing idle computer machines to build Desktop Clouds. Hence, the term Desktop comes from Desktop Grids because both of Desktop Clouds and Desktop Grids are based on Desktop PCs and laptops etc. Similarly, the term Cloud comes from Cloud as Desktop Cloud aims to provide services based on the Cloud business model. Several synonyms for Desktop Cloud have been used, such as Ad-hoc Cloud [5], Volunteer Cloud [3], Community Cloud [6] and Non-Dedicated Cloud [7]. The literature indicates that very little work has been undertaken in this direction.

Table I. Traditional Clouds vs. Desktop Clouds

Feature	Traditional Clouds	Desktop Clouds
Resources	Dedicated	Non-dedicated and volatile
Cost	Relatively high	Cheap
Location	Limited to a number of data centres	Distributed across the globe
Services	Reliable and available	Low availability and unreliable
Heterogeneity	Heterogeneous	Very heterogeneous

Desktop Clouds differ from Traditional Clouds in several things, as it is depicted in Table I. Firstly, the infrastructure of Desktop Cloud consists of resources that are non-dedicated, i.e., not made to be part of Cloud infrastructure. Desktop Cloud helps in saving energy since it utilises already-running undedicated resources, which would otherwise remain idle. Some studies show that the average percentage of local resources being idle within an organisation is about 80% [8]. It is shown that an idle machine can consume up to 70% of the total power consumed when it is fully utilised according to [9]. On the contrary, the infrastructure of Traditional Clouds is made of a large number of dedicated computing resources. Traditional Clouds have a negative impact on the environment since their data centres consume massive amounts of electricity for cooling these resources.

Secondly, resources of Desktop Clouds are quite scattered across the globe, whereas they are limited in Traditional Cloud to a number of locations in data centres. Furthermore, nodes in Desktop Cloud are highly volatile because nodes of Desktop Clouds can be down unexpectedly without prior notice. Node failures can occur for various reasons such as connectivity issues, machine crashing or simply the machine becomes busy with other work by its owner takes priority. High volatility in resources has negative impact on availability and performance [10]. Although, resources in both Traditional Cloud and Desktop Cloud are heterogeneous, they are even more heterogeneous and dispersed in Desktop Cloud. Traditional Clouds are centralised, which leads to the potential that there could be a single point of failure issue if a Cloud service provider goes out of the business. In contrast, Desktop Clouds manage and offer services in a decentralised manner. Virtualisation plays a key role in both Desktop Clouds and Traditional Clouds.

Desktop Clouds can be confused with other distributed systems, specifically Desktop Grids. Both Desktop Clouds and Desktop Grids share the same concept that is exploiting computing resources when they become idle. The resources in both systems can be owned by an organisation or denoted by the public over the Internet. Both Desktop Grids and Desktop Clouds can use similar resources. Resources are volatile and prone to failure without prior knowledge. However, Desktop Grids differ from Desktop Clouds in the service and virtualisation layers. Services, in Desktop Clouds, are offered to clients in an elastic way. Elasticity means that users can require more computing resources in short term [11]. In contrast, the business model in Desktop Grids is based on a 'project oriented' basis, which means that every user is allocated a certain time to use a particular service [12]. In addition, Desktop Grids' users are expected to be familiar with details about the middleware used in order to be able to harness the offered services [13]. Specific software needs to be installed to computing machines in order to join a Desktop Grid. Clients in Desktop Clouds are expected to have little knowledge to enable them just use Cloud services under the principle ease of use. Desktop Grids do not employ virtualisation to isolate users from the actual machines while virtualisation is highly employed in Desktop Clouds to isolate clients from the actual physical machines.

III. DESKTOPCLOUDSIM

DesktopCloudSim is an extension tool proposed to simulate failure events happening in the infrastructure level based on CloudSim simulation tool. Therefore, this section starts by a brief discussion of CloudSim. The extension tool, DesktopCloudSim, is presented next. DesktopCloudSim is used to evaluate VM allocation mechanisms, thus the last subsection in this section discusses traditional mechanisms that are used by open Cloud middleware platforms.

A. CLOUDSIM

CloudSim is a Java-based discrete event simulation toolkit designed to simulate Traditional Clouds [14]. A discrete system is a system whose state variables change over time at discrete points, each of them is called an event. The tool was developed by a leading research group in Grid and Cloud computing called CLOUDS Laboratory at The University of Melbourne in Australia. The simulation tool is based on both GridSim [15] and SimJava [16] simulation tools.

CloudSim is claimed to be more effective in simulating Clouds compared to SimGrid [17] and GroudSim [18] because CloudSim allows segregation of multi-layer service (IaaS, PaaS and SaaS) abstraction [14]. This is an important feature of CloudSim that most Grid simulation tools do not support. Researchers can study each abstraction layer individually without affecting other layers.

CloudSim can be used for various goals [19]. First, it can be used to investigate the effects of algorithms of provisioning and migration of VMs on power consumption

and performance. Secondly, it can be used to test VM mechanisms that aim at allocating VMs to PMs to improve performance of VMs. It is, also, possible to investigate several ways to minimise the running costs for CSPs without violating the SLAs. Furthermore, CloudSim enables researchers to evaluate various scheduling mechanisms of tasks submitted to running VMs from the perspective of Cloud brokers. Scheduling mechanism can help in decreasing response time and thus improve performance.

Although CloudSim is considered the most mature Cloud simulation tool, the tool falls short in providing several important features. The first is that does not simulate performance variations of simulated VMs when they process tasks [19]. Secondly, service failures are not simulated in CloudSim [20]. The service failures include failures in tasks during running time and complex overhead of complicated tasks. Furthermore, CloudSim lacks the ability to simulate dynamic interaction of nodes in the infrastructure level. CloudSim allows static configuration of nodes, which remain without change during run time. Lastly, node failures are not included in CloudSim tool. DesktopCloudSim enables the simulation of dynamic nodes and node failures while performance variations and service failures are simulated by other tools. Section VI discusses those tools.

Several simulators have been published to simulate Grid computing. SimGrid [17] is one of the early simulation tools to simulate Grid environment. GridSim [15] is another tool fits within the same goal. CloudSim is built on top of GridSim. Donassole et al. [21] extended SimGrid to enable simulating Desktop Grids. Their work enables building a Grid on top of resources contributed by the public. The simulation tool is claimed to be of high flexibility and enable simulating highly heterogonous nodes. GroudSim [18] is a scalable simulation tool to simulate both Grid and Cloud platforms. The tool lets researchers to inject failures during running time. However, all of these tools fall in short to provide virtualisation feature, which is essential to evaluate VM allocation mechanisms.

MDCSim [22] is a commercial, discrete-event simulation tool developed at Pennsylvania State University to simulate multi-tier data centres and complex services in Cloud computing. It has been designed with three-level architecture, including a user-level layer, a kernel layer and communication layer for modelling the different aspects of a Cloud system. MDCSim can analyse and study a cluster-based data centre with in-depth implementation of each individual tier. The tool can help in modelling specific hardware characteristics of different components of data centres such as servers, communication links and switches. It enables researchers to estimate the throughput, response times and power consumption. However, as the simulation tool is a commercial product, it is unsuitable to run experiments.

GreenCloud [23] is another cloud simulation framework, implemented in C++ and focused on the area of power consumption and its measurement. The tool was developed on top of Ns2, a packet-level network simulation tool [24]. Having the tool implemented in C++ makes it feasible to simulate a large number of machines (100,000 or more), while Java is assumed to be able to handle only 2GB memory on 32 bit machines. However, CloudSim was able to simulate and instantiate 100,000 machines in less than 5 minutes with only 75 MB of RAM, according to Sakellari and Loukas 2013. Although GreenCloud can support a relatively large number of servers, each may have only a single core. In addition, the tool pays no attention to virtualisation, storage and resource management.

iCanCloud [26] is a C++ based open source Cloud simulation tool based on SIMCAN [27], a tool to simulate large and complex systems. It was designed to simulate mainly IaaS Cloud systems, such as instance-based clouds like EC2 Amazon Cloud. iCanCloud offers the ability to predict the trade-off between performance and cost of applications for specific hardware to advise users about the costs involved. The tool has a GUI feature and can be adapted to different kinds of IaaS cloud scenarios. However, iCanCloud does not enable researchers to study and investigate energy efficiency solutions.

There are several extensions of CloudSim that have been developed to overcome the limitations of CloudSim tool. The extensions are NetworkCloudSim [28], WorkflowSim [20], DynamicCloudSim [19], FederatedCloudSim [29] and InterCloud [30]. NetworkCloudSim is an extension simulation tool based on CloudSim to enable the simulation of communication and messaging aspects in Cloud computing. The focus of the tool is on the network flow model for data centres and network topologies, bandwidth sharing and the network latencies involved. It also enables the simulation of complex applications such as scientific and web applications that require interconnections between them during run time. Such features can allow further accurate evaluation of scheduling and resource provisioning mechanisms in order to optimise the performance of Cloud infrastructure.

WorkflowSim is a new simulation extension that has been published recently as an extension for CloudSim tool. The tool was developed to overcome the shortage of CloudSim in simulating scientific workflow. The authors of WorkflowSim added a new management layer to deal with the overhead complex scientific computational tasks, arguing that CloudSim fails in simulating the overheads of such tasks such as queue delay, data transfer delay, clustering delay and postscripts. This issue may affect the credibility of simulation results. They also point out the importance of failure tolerant mechanisms in developing task scheduling techniques. WorkflowSim focuses on two types of failures: tasks failure and job failure. A task contains a number of jobs, so failure in a task causes a series of jobs to fail. However, our work differs from WorkflowSim in the failure event and its impact. The focus of this research is on the infrastructure level, containing nodes hosting VMs, whereas its authors were interested in the service level, that is, tasks and applications. It can be argued that service providers should consider developing failure-tolerant mechanisms to overcome such events in the infrastructure level.

DynamicCloudSim is another extension for CloudSim tool. Its authors were motivated by the fact that CloudSim lacks the ability to simulate instability and dynamic

performance changes in VMs during runtime. This can have a negative impact on the outcome of computational intensive tasks, which are quite sensitive to the behaviour of VMs. The tool can be used to evaluate scientific workflow schedulers, taking into consideration variance in VM performance. In addition, the execution time of a given task is influenced by the I/O-bound such as reading or writing data. Its authors extended instability to include task failure. Performance variation of running VMs is an open research challenge, but beyond the scope of this study.

FederatedCloudSim [29] is an extension tool in the CloudSim toolkit to enable the simulation of federated Clouds using difference federation scenarios, while respecting SLAs. According to Goiri et al. [31], Cloud Federation is the idea of bringing many CSPs together in order to avoid the case of over-demand for Cloud services by letting a CSP rent out CSPs to other computing facilities. FederatedCloudSim enables researchers to simulate and study various ways to standardise interfaces and communications between CSPs in a federated Cloud. Such a tool can help to study optimisation solutions for exchanging Cloud services between CSPs without violation of SLAs. InterCloud is another simulation tool that has been developed to simulate Cloud federation, based on the CloudSim tool. However, InterCloud falls short of providing sufficient simulation capabilities of SLAs, compared to FederatedCloudSim.

B. The Architecture of DesktopCloudSim

Simulation is necessary to investigate issues and evaluate solutions in Desktop Clouds because there is no real Desktop Cloud system available on, which to run experiments. In addition, simulation enables control of the configuration of the model to study each evaluation metric. In this research, CloudSim is extended to simulate the resource management model. CloudSim allows altering the capabilities of each host machines located in the *data centre* entity in the simulation tool. This feature is very useful for experimentations, as it is needed to set the infrastructure (i.e., physical hosts) to have an unreliable nature. This can be achieved by extending the *Cloud Resources* layer in the simulation tool. Figure 1 Depicts the layered architecture of CloudSim combined with an abstract of the DesktopCloudSim extension.

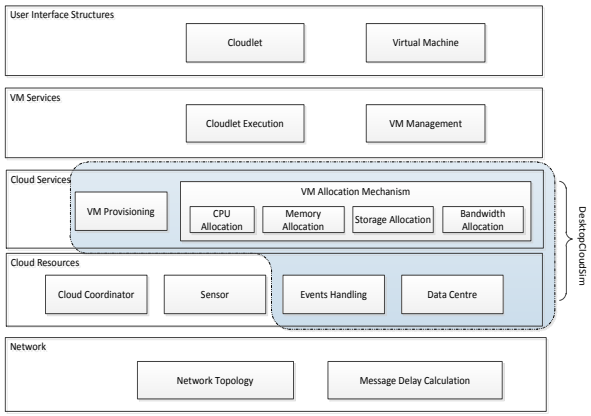

Figure 1. DesktopCloudSim Abstract

Figure 2 illustrates the components of DesktopCloudSim that read FTA trace files, as explained later in this paper. The trace files contain the failure events of PMs. The *Failure Analyser* component analyses the files of failures to send failure events to *Failure Injection* component. The *Failure Injection* component receives failure events from the *Failure Analyser* and inject failures into associated PMs during run time by sending events to *Available PMs* component. The *Available PMs* contains a list of PMs that are ready to be used, so if a PM fails then it is removed or, if a PM joins, it is added. The *Failure Injection* component informs the *VM Mechanism* unit if a PM fails, to let it restart the failed VMs on another live node or nodes. The *VM Provisioning* component provisions VMs instances to be allocated to PMs selected by *Select PM*. The *VM Mechanism* controls, which PM hosts a VM instance. The *VM Mechanism* creates restart VM instances. In addition, the *VM Mechanism* can replicate a running VM instance, if required.

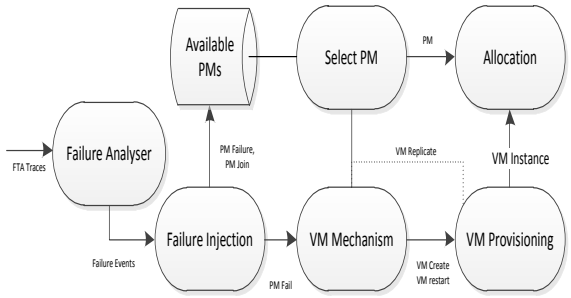

Figure 2. DesktopCloudSim Model

C. VM Allocation Mechanisms

Several VM allocation mechanisms that are employed in open Cloud platforms are discussed in this subsection. VM allocation mechanisms are: (i) Greedy mechanism, which allocates as many VMs as possible to the same PM in order to improve utilisation of resources; (ii) RoundRobin mechanism allocates the same number of VMs equally to each PM; and (iii) First Come First Serve (FCFS) mechanism allocates a requested VM to the first available PM that can accommodate it. This paper is limited to these mechanisms because they are implemented in open source Cloud management platforms such as Eucalyptus [32], OpenNebula [33] and Nimbus [34].

When a VM is requested to be instantiated and hosted to a PM, the FCFS mechanism chooses a PM with the least used resources (CPU and RAM) to host the new VM. The Greedy mechanism allocates a VM to the PM with the least number of running VMs. If the chosen PM cannot accommodate the new VM, then the next least VM running PM will be allocated. RoundRobin is an allocation mechanism, which allocates a set of VMs to each available physical host in a circular order without any priority. For example, suppose three VMs are assigned to two PMs. The RoundRobin policy will allocate VM1 to PM1 then VM2 to PM2 then allocate VM3 to PM1 again. Although these

mechanisms are simple and easy for implementation, they have been criticised for being underutilisation mechanisms, which waste energy [35]. The FCFS mechanism is expected to yield lowest throughput among the aforementioned mechanisms because it assigns VMs to PMs in somehow random manor.

IV. EXPERIMENT

The experiment is conducted to evaluate VM mechanisms mentioned in Section III.C. There are two input types needed to conduct the experiment. The first input is the trace file that contains failure events happening during the run time. Failure trace files are collected from an online archive. Subsection A discusses further this archive. The second input set is the workload submitted to the Desktop Cloud during running time. Subsection B talks about this workload.

A. Failure Trace Archive

Failure Trace Archive (FTA) is a public source containing traces of several distributed and parallel systems [36]. The archive includes a pool of traces for various distributed systems including Grid computing, Desktop Grid, peer-to-peer (P2P) and High Performance Computing (HPC). The archive contains timestamp events that are recorded regularly for each node in the targeted system. Each event has a state element that refers to the state of the associated node. For example, an event state can be unavailable, which means this node is down at the timestamp of the event. The unavailable state is considered a failure event throughout this report. The failure of a node in an FTA does not necessarily mean that this node is down. For example, a node in a Desktop Grid system can be become unavailable because its owner decides to leave the system at this time.

The Notre Dame and SETI@home FTAs were retrieved from Failure Trace Archive website. The NotreDame FTA represents an archive of a pool of heterogeneous resources that have run for 6 months within the University of Notre Dame during 2007 [37]. The nodes of this archive can be used to simulate the behaviour of nodes in a private Desktop Cloud system. Each month is provided separately representing the behaviour of nodes located in the University of Notre Dame. The FTA contains 432 nodes for month 1, 479 nodes for month 2, 503 nodes for month 3, 473 nodes for month 4, 522 nodes for month 5 and 601 nodes for month 6. The second trace archive is SETI@home FTA. The FTA has a large pool of resource (more than 200 thousand nodes) that have been run for a year in 2008/09 [38]. The nodes in SETI@home are highly heterogeneous because most of these computing nodes are denoted by the public over the Internet. A random sample of 875 nodes has been selected from SETI@home FTA for six months. The selected PMs are those who have trace files with sufficient failure events to simulate SETI@home Desktop Cloud, which is considered a public Desktop Cloud system.

We calculated the average percentage failure of nodes on every hour basis. Such study can help in evaluating the behaviour of VM mechanisms. The failure percentage is calculated as:

$$failure\ (h) = \frac{numbr\ of\ failed\ nodes\ at\ \ h}{total\ number\ of\ nodes} * 100$$

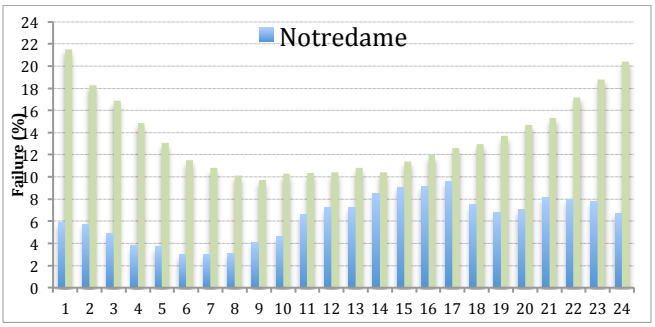

Figure 3. Average Hourly Failure

Figure 3 shows an average hourly failure percentage in 24 hour-period for analysis of 6 months run times of NotreDame and SETI@home nodes. The period is set to 24 hours because this is the running time set for our experiments. NotreDame failure analysis shows that failure percentage is about 3% as minimum in hour 6. Hour 17 recorded the highest failure percentages at about 10%. It is worth mentioning that on average about 6.3% of running nodes failed in an hour during the 6-month period. For SETI@home nodes, the highest failure percentage was about 21% in hour 1 while lowest was about 10%. However, it was recorded that the percentage of node failures can reach up to 80% in some hours. Overall, the average hourly failure rate of SETI@home is about 13.7%. This can demonstrate that failure events in Desktop Clouds are norms rather than exceptions.

B. Experiment Setting

The experiment is run for 180 times once for NotreDame Desktop Cloud and another for SETI@home Desktop Cloud, each time represent a simulation of running NotreDame Desktop Cloud for one day. The run time set to one day because the FTA provides a daily trace for NotreDame nodes as mentioned above. Each VM allocation mechanism is run for 180 times representing traces of 6 months from the FTA. This makes the total number of runs is 540 (3 * 180). The workload was collected from the PlanetLab archive. The archive provides traces of real live applications submitted to the PlanetLab infrastructure [39]. One day workload was retrieved randomly as input data in this experiment. Each task in the workload is simulated as a Cloudlet in the simulation tool. The workload input remains the same during all the experiment runs because the aim of this experiment is to study the impact of node failures on throughput of Desktop Clouds.

The FTA files provide the list of nodes along with timestamps of failure/alive times. However, the specifications of nodes are missing. Therefore, we had set specification up randomly for physical machines. The missing specifications are technical specifications such as CPU power, RAM size and hard disk size.

Clients requested that 700 instances of VMs to run for 24 hours. There are four types of VM instances: *micro, small, medium and large*. They are similar to VM types that are offered by Amazon EC2. The type of each requested VM instance is randomly selected. The number of requested VMs and types remain the same for all run experiment sets. Each VM instance receives a series of tasks to process for a given workload. The workload is collected from PlanetLab archive, which is an archive containing traces. PlanetLab is a research platform that allows academics to access a collection of machines distributed around the globe. A one day workload of tasks was collected using CoMon monitoring tool [40]. The same workload is submitted in every one day run.

In the experiment, if a node fails then all hosted VMs will be destroyed. The destruction of a VM causes all running tasks on the VM to be lost, which consequently affect the throughput. The lost VM is started again on another PM and begins receiving new tasks. During running time, a node can become alive and re-join the Cloud according to the used failure trace file. The simulation was run on a Mac i27 (CPU = 2.7 GHz Intel Core i5, 8 GB MHz DDR3) running OS X 10.9.4. The results were analysed using IBM SPSS Statistics v21 software.

V. RESULTS AND DISCUSSION

Table II shows a summary of descriptive results obtained when measuring the throughput output for each VM allocation mechanism implemented in NotreDame Cloud. Kolmogorov-Smirnov (K-S) test of normality shows that the normality assumption was not satisfied because the FCFS and Greedy mechanisms are significantly non-normal,$P < 0.05$. Therefore, the non-parametric test Friedman's ANOVA was used to test which mechanism can yield better throughput. Friedman's ANOVA test confirms that throughput varies significantly from mechanism to another, $X_F^2(2) = 276.6, P < 0.001$. Mean, median, variance and standard deviation are reported in Table II.

Table II. Throughput Results for NotreDame Desktop Cloud

Mechanism	Mean (%)	Median (%)	Var.	St. Dev.	K-S Test
FCFS	82.66	82.2	40.32	6.35	P = 0.034
Greedy	92.47	93.1	18.34	4.28	P < 0.001
RoundRobin	89.14	89	16.47	4.06	P = 0.2

Three Wilcoxon pairwise comparison tests were conducted to find out which mechanism with highest throughput. Note that three tests are required to compare three pairs of mechanisms, which are FCFS Vs. Greedy, FCFS Vs. RoundRobin and Greedy Vs. RoundRobin mechanisms. The level of significance was altered to be 0.017 using Bonferroni correction [41] method because there were 3 post-hoc tests required ($0.05/3 \approx 0.017$). The tests show that there is a significant different between each mechanism with its counterpart. Therefore, it can be concluded that Greedy mechanism yield highest throughput since it has the median with highest value (median = 92.47%).

The median throughput of FCFS was about 83%, as being the worst mechanism among the tested mechanisms. The RoundRobin came second in terms of throughput because the mechanism distributes load equally. So, node failures are ensured to affect the throughput. The median throughput was about 92% when Greedy VM mechanism was employed. The mechanism aims at maximising utilisation by packing as many VMs as possible to the same PM, thus reduce the number of running PMs. The average failure rate in submitted tasks is about 8%, given the average node failure percentage is about 6% as Section IV.A shows.

Table III shows a summary of descriptive results obtained for throughput output for the FCFS, Greedy and RoundRobin VM allocation mechanisms employed in SETI@home Desktop Cloud. Kolmogorov-Smirnov (K-S) test of normality shows that the normality assumption was violated because the FCFS and RoundRobin mechanisms are significantly non-normal, $P < 0.05$. Therefore, the non-parametric test Friedman's ANOVA was used to test, which mechanism can yield better throughput. Friedman's ANOVA test confirms that throughput varies significantly from mechanism to another, $X_F^2(2) = 86.63, P < 0.001$. Mean, median, variance and standard deviation are reported in Table III.

Table III. Throughput Results for SETI@home Desktop Cloud

Mechanism	Mean (%)	Median (%)	Var.	St. Dev.	K-S Test
FCFS	82.04	83.28	20.23	4.5	P < 0.001
Greedy	81.80	81.93	16.1	4.01	P = 0.2
RoundRobin	80.45	81.04	16.11	4.01	P = 0.004

Three Wilcoxon pairwise comparison tests were conducted to find out which mechanism yielded highest throughput. As explained before, three tests are required to compare three pairs of mechanisms which are FCFS Vs. Greedy, FCFS Vs. RoundRobin and Greedy Vs. RoundRobin mechanisms. The level of significance was altered to be 0.017 using Bonferroni correction [41] method because there were 3 post-hoc tests required ($0.05/3 \approx 0.017$). The tests show that there is a statistically significant different between RoundRobin vs. Greedy mechanisms and RoundRobin vs. FCFS mechanisms. However, Greedy vs. FCFS mechanisms did not show a significant difference. Therefore, it can be which mechanisms yielded highest throughput.

The throughput results of employed mechanism for SETI@home Desktop Cloud showed that the difference between throughput of results were quite limited, by less than 2%. The FCFS and Greedy mechanisms yielded highest throughput at about 82% and 81% respectively. RoundRobin came the last with throughput of about 80% only. The mean reason behind the drop of throughput results of mechanisms in SETI@home Desktop Cloud compared to NotreDame Desktop Cloud is the average failure rate of nods in

SETI@home is almost the double of average failure rate of NotreDame nodes. We can conclude that based on the results of experiments there is a potential to develop fault-tolerant VM mechanism for Desktop Cloud systems.

VI. Related work

The literature shows that the focus is on how to minimise the power consumed by physical nodes in order to maximise revenue for CSPs. Researchers are motivated to tackle the issue because power in data centres accounts for a large proportion of maintenance costs [42]. The idea is that better utilisation leads to more servers that are idle, so can be switched to power saving mode (e.g., sleep, hibernation) to reduce their energy consumption. According to Kusic et al. an idle machine uses as much as 70% of the total power consumed when it is fully utilised [9].

Srikantaiah et al. studied the relationship between energy consumption, resource utilisation and performance in resource consolidation in Traditional Clouds [43]. The researchers investigated the impact of resource high utilisation on performance degradation when various VMs are consolidated at the same physical node, introducing the notion of optimal points. They argued that there is a utilisation point that allows placement of several VMs at the same physical node without affecting performance. Once this point is reached in a PM, no new VMs are placed, and the proposal is to calculate this optimal point of utilisation then to employ a heuristic algorithm for VM placement, since the authors defined the consolidation problem as a multi-dimensional Bin Packing problem and showed that the consumption of power per transaction results in a 'U'-shaped curve. They found that CPU utilisation at 70% was the optimal point in their experiment, but that it varied according to the specification of the PMs and workload. The approach is criticised because the technique adopted depends heavily on the type of the workload and the nature of the targeted machines [44].

Verma et al. presented 'pMapper', a power-aware framework for VM placement and migration in virtualised systems, where the monitoring engine collects current performance and power status for VMs and PMs in case migration is required [45]. The allocation policy in pMapper employs *mPP,* an algorithm that places VMs on servers with the aim of reducing the power they consume. The algorithm has two phases. The first is to determine a target utilisation point for each available server based on their power model. The second is to employ a First Fit Decreasing (FFD) heuristic solution to place VMs on servers with regard to the utilisation point of each. The optimisation in the framework considers reducing the cost of VM migration from one server to another. The migration cost is calculated by a migration manager for each candidate PM in order to determine which node is chosen. The work is criticised as it does not strictly comply with SLA requirements [46]; the proposed allocation policy deals with static VM allocation where specifications of VMs remain unchanged. This is not the case in Cloud computing, where clients can scale up or down dynamically. In addition, it requires prior knowledge of each PM in order to compute the power model.

Meng et al. proposed a VM provisioning approach to consolidate multiple VM instances for the same PM in order to improve resource utilisation and thus reduce the energy consumed by under-utilised PMs [47]. A VM selection algorithm was developed to identify compatible VM instances for consolidation. Compatible VM instances are those with similar capacity demand, defined as their application performance requirement, and these are grouped into sets allocated to the minimum number of PMs. It can be argued that consolidating compatible VM instances to the same PMs will have a small negative effect on applications assigned to each VM instance and thus keep SLA requirements from being violated. The study found an improvement of 45% in resource utilisation.

The authors in [48] and [32] devised an algorithm to allocate VM instances to PMs at data centres with the goal of reducing power consumption in PMs without violating the SLA agreement between a Cloud provider and users. The researchers argued that assigning a group of VMs to as few PMs as possible will save power [49]. The energy-aware resource algorithm [46] has two stages: VM placement and VM optimisation. The VM placement technique aims to allocate VMs to PMs using a Modified Best Fit Decreasing (MBFD) algorithm. This is based on the Best Fit Decreasing (BFD) algorithm that uses no more than 11/9 * OPT + 1 bins (OPT is the optimal number of bins) [50].

The MBFD algorithm sorts VMs into descending order of CPU utilisation in order to choose power-efficient nodes first. The second stage is the optimisation step responsible for migrating VMs from PMs that are either over- or under-utilised. However, VM migration may cause unwanted overheads, so should be avoided unless doing so reduces either power consumption or performance, so the authors set lower and upper thresholds for utilisation. If the total utilisation of the CPU of a PMs falls below the lower threshold, this indicates that the host might consume more energy than it needs. Similarly, if the utilisation exceeds the upper threshold then the performance of the hosted VMs may deteriorate. In this case, some VMs should migrate to another node to reduce the level of utilisation. The authors concluded that the Minimisation of Migrations (MM) policy could save up to 66% of energy, with performance degradation of up to 5%. It was found that the MM policy minimised the number of VMs that have to migrate from a host in the event of utilisation above the upper threshold.

Graubner et al. proposed a VM consolidation mechanism based on a live migration technique with the aim of saving power in Cloud computing [44]. They developed a relocation algorithm that periodically scans available PMs to determine which PM to migrate VM instances from, and which PM to migrate them to. The approach was found to save up to 16% of power when implemented in the Eucalyptus platform, however the relocation process was unclear, with no further explanation of when it is triggered during run time [51].

The authors in [52] proposed GreenMap, a power-saving VM-based management framework under the constraint of multi-dimensional resource consumption in clusters and data centres. GreenMap dynamically allocates and reallocates VMs to a set of PMs within a cluster during runtime. There

are four modules in the framework: clearing; locking; trade-off; and placement. The clearing module is responsible for excluding VMs inappropriate for dynamic placement, for instance those with unpredictable or rapid variation in demand. The locking module monitors SLA violations caused by the workload, in which event the module will switch to a redundant VM for execution. The trade-off module evaluates the potential of a new placement generated by the placement module in respect of performance and cost trade-off. The placement module performs a strategy for reallocating live VMs to another physical resource to save power, based on a configuration algorithm. The algorithm starts by randomly generating a new placement configuration. The placement module then delivers the configuration to the trade-off module. The experiment showed that it is possible to save up to 69% of power in a cluster, with some performance degradation, but it did not consider the overheads of the placement module.

The authors in [51] proposed an energy-saving mechanism developed and implemented for a private Cloud called Snooze, tested using a dynamic web workload. The authors argued that it differed from other power-aware VM mechanisms in two aspects, in that it was applied and tested in a realistic Cloud environment, and that it takes dynamic workload into consideration. A monitor unit was introduced periodically to check running PM; any under- or over-utilised nodes were reported to a general manager module to issue a migration command. There are four VM allocation policies: placement; overload relocation; underload relocation; and consolidation.

The placement policy allocates new VM instance requests to PMs using RoundRobin scheduling, which distributes the load to PMs in a balanced way. The overload policy scans PMs to check if a PM is overloaded with VM instances and, if so, searches for a PM that is only moderately loaded to accommodate these VM instances in all-or-nothing way (i.e., migrate all running VMs or none). The migration command is sent to the migration policy for straightforward execution. Similarly, the underload policy issues a migration command to migrate VMs from under-utilised PM in an all-or-nothing way. The mechanism managed to save up to 60% of power, the experiment concluded, but it was conducted in a homogenous infrastructure, that is, it assumed that all PMs have the same computing capacity. In addition, the all-or-nothing method may be a drawback as it leads to PMs being overloaded, which may cause performance degradation in instances of hosted VM.

Van et al. proposed a virtual resource manager focused on maintaining service levels while improving resources utilisation via a dynamic placement mechanism [53]. The manager has two levels: a local decision module and a global decision module. The first is concerned with applications, as the manager deals with complex N-tier levels in, for instance, online applications that require more than one VM instance to process. The global decision module has two stages: the VM placement stage, concerned with allocating a VM to a specific PM with the goal of improving resource utilisation; and the VM provisioning stage of scheduling applications to VMs (i.e., sending applications to be processed by VM instances).

The authors in [54] proposed a novel VM placement approach of two phases: candidacy and placement. The former elects a list of PMs eligible to accommodate VM instances, choosing the candidate PM on the basis of migration capability, network bandwidth connectivity and user deployment desire, which should be available beforehand. Available PMs have a four-level hierarchy representing an ordering system of PMs available to be candidates. The latter phase selects one of the candidate PMs from the first phase to host a VM instance on the basis of low-level constraints. The authors argue that the first phase can help to reduce the time spent choosing the most suitable PM. However, this work requires prior knowledge of user deployment of VM instances, which is not supported in CSPs. CSPs usually offer different classes of VM instances for end users to choose between. Asking further questions regarding user preferences is not economically viable.

The authors in [55] proposed a VM placement technique that employs the FF heuristic solution to maximise revenue for CSPs under performance constraints, expressed as an SLA violation metric measuring performance degradation of VM instances caused by using the FF mechanism to improve resource utilisation. The proposed system has two managers: the global manager decides which PM hosts a VM instance; and the local manager is concerned with scheduling VM instances within the hosted PM. The global manager employs a decision-making policy for each candidate PM's viability for hosting a VM instance in such a way as to improve resource utilisation.

Calcavecchia et al. proposed the Backward Speculative Placement as a novel VM placement technique [56]. The VM placement technique has two phases: continuous deployment and ongoing optimisation. The continuous deployment phase allocates a VM instance to the PM with the highest demand risk, a scoring function to measure the level of dissatisfaction with a PM at the final unit of time. It is, however, not clearly explained how this is awarded. The ongoing optimisation phase migrates VM instances hosted to a PM with high risk demand to another PM with a low score, as long it is able to accommodate the VM instances. The Backward Speculative Placement technique was able to decrease the execution time of submitted tasks.

The authors in [57] proposed a VM placement and migration approach to minimise the effect of transfer time of data between VM instances and data storage. In Cloud computing, a CSP can provide VM instances to end users to process data while these data are stored in different locations, for example Amazon EC2 and Amazon S3. Therefore, the approach developed takes network I/O requirements into consideration when VM placement is applied. In addition, the VM migration policy is triggered when the time required to transfer data exceeds a certain threshold. Network instability is the main reason for this increase of time, and the threshold is stated in the SLA agreement. The study showed that the time taken to complete the task fell, on average, due to the placement of VM, depending on location.

A novel traffic-aware VM placement technique was developed by [58] with the goal of improving network scalability. The mechanism employs a two-tier approximate algorithm to place VM instances with PMs in such a way that significantly reduces the aggregate traffic in datacentres. The two-tier algorithm partitions VMs and PMs separately into clusters. The VMs and PMs are matched individually in each cluster. The partitioning step is achieved using a classical min-cut graph algorithm that assigns each VM pair with a high mutual traffic rate to the same VM cluster. Having VM instances with a high traffic rate in the same cluster of PMs means that traffic is exchanged only through that cluster, which can reduce the load upon switches at a data centre.

Purlieus [59] is a resource allocation tool developed to improve the performance of MapReduce jobs and to reduce network traffic by paying attention to the location of resources. MapReduce enables the analysis and processing of large amount of data in a quick and easy way [60]. Purlieus employs VM placement techniques that allocate VM instances to PMs according to their location. Purlieus was able to reduce the execution time of jobs by 50% for a variety of types of workload.

The authors in [61] studied the VM allocation problem from the network perspective [61]. They proposed a novel VM placement mechanism that considers network constraint, which is the variation in traffic demand time. Its goal is to minimise the load ratio across all network cuts by implementing a novel mechanism, the two-phase connected component-based recursive split, to choose the PM with which to place a VM instance. It exploits the recursive programming technique to formulate a ranking table of each VM instance that is connected. The PM with the least connected ranks of associated VMs is selected to host a new VM instance, but the proposed mechanism is for static VM placement only, thus it does not consider moving VM instances around during run time to reduce the cut load ratio.

The authors in [62] introduced S-CORE, a scalable VM migration mechanism to reallocate VM instances to PMs dynamically with the goal of minimising traffic within a datacentre. They showed that S-CORE can achieve cost reductions in communication of up to 80% with a limited amount of VM migration. S-CORE assigns a weight for each link in a datacentre, taking into consideration the amount of data traffic routed over these links. If the line weight exceeds a certain threshold, then some VM instances with high traffic load have to migrate to another PM using a different link. Such an approach avoids traffic congestion on core links at data centres to prevent any degradation in the performance of a Cloud system.

The aforementioned studies investigated various VM allocation mechanisms with the aim of minimising power consumption, improving performance or reducing the traffic load in Cloud systems. However, they all fell short of providing a mechanism tolerant of failure events in Clouds' PMs. Therefore, these VM allocation techniques are neither practical to employ nor to implement in a Desktop Cloud system. The following subsection reviews several studies that have tackled the issue of node failure.

A wide range of techniques and approaches has been developed to tackle node failure issues in Desktop Grid systems, because a node within a Desktop Grid system can voluntarily join or leave the system, increasing the probability of node failure, heightening the risk of losing results. For example, the authors in [63] developed a fault-tolerant technique in Desktop Grid systems that employs replication of applications to avoid losing them in failure events. Another approach was proposed by [64], based on the mechanism of application migration. This checks applications periodically during runtime, and in the event of node failures all associated application are restored and migrated to another node. However, this is not practical in this study because it is concerned with the applications level and violates the concept of the Cloud computing paradigm that isolates the infrastructure layer from the service layer to prevent CSPs from having control over services run by end users.

Machida et al. proposed a redundancy technique for server consolidation [65]. The focus was complex online applications requiring several VM instance for each application, and the technique offers k fault tolerance with the minimum number of physical servers required for application redundancy [65]. It relies on replicating an application a times and running it for k number of VM instances. The number of VM instances is calculated on the basis of the requirements of application a, but requires full knowledge of and access to the applications and services that run on VM instances in order to replicate them. This, again, violates the concept of Cloud computing whereby CSPs are prevented from being able to access and control the applications of end users. Furthermore, the approach assumes that all physical servers have the same computing capacity, impractical in the era of Cloud computing where PMs are usually quite heterogeneous.

The authors in [66] proposed the BFTCloud, a fault-tolerant framework for Desktop Cloud systems that tackles the specific malicious behaviour of nodes known as Byzantine faults: machines that provide deliberately wrong results. The framework employs a replication technique with a primary node by $3 * f$, where f is the number of faulty nodes at run time. The framework considers failure probability as the mean to choose primary nodes and their replicas in respect of QoS requirements. Byzantine faults are identified by comparing the results reported by a primary node with those of its replica; if the results are inconsistent then they will be sent to another node to process and compared to detect which machine is behaving suspiciously. However, the calculation of failure probability is not clearly given. In addition, although the framework was said to be for Desktop Cloud systems, it does not possess the essential feature of employing virtualisation to keep the service layer isolated from the physical layer; in fact, the technique is to replicate tasks by sending one to a primary node and its $3* f$ replicas of nodes. Another issue worth mentioning about the BFTCloud mechanism is the notion of f, which means that the number of faulty nodes should be known before run time. However, this technique is impractical since the number of

node failures in such distributed systems is unpredictable and difficult to calculate [67].

The authors in [68] addressed the issue of node failure in hybrid Clouds, that is, private and public Clouds. The problem is formulated as follows: a private Cloud with limited resources (i.e., PMs) has a certain number of nodes with a high failure rate. The question is how to minimise the dependency of public Clouds to achieve better QoS, given that sending workload to a public Cloud costs more. The authors proposed a failure-aware VM provisioning for hybrid Clouds, a 'time-based brokering strategy', to handle failure of nodes in private Clouds by redirecting tasks required long term into a public Cloud. The decision to forward a task to a public Cloud is based on the duration of the request; if longer than the mean request duration of all tasks, then it will be forwarded. Although the proposed strategy considers that a public Cloud solves the issue of node failure in private Clouds, the issue is not answered unless the reliability of this public Cloud can be guaranteed.

The review of VM mechanisms in this section shows that the design of a fault-tolerant VM allocation mechanism remains an open research problem that needs to be tackled in Cloud environments with faults, such as in Desktop Cloud systems.

VII. Conclusion

Desktop Cloud can be seen as a new direction in Cloud computing. Desktop Cloud systems exploit idle computing resources to provide Cloud services mainly for research purposes. The success of Desktop Grids in providing Grid capabilities stimulated the concept of applying the same concept within Cloud computing. However, Desktop Clouds use infrastructure that is very volatile since computing nodes have high probability to fail. Such failures can be problematic and cause negative on the throughput of Desktop Clouds.

This paper presented a DesktopCloudSim as an extension tool CloudSim, a widely used Cloud simulation tool. DesktopCloudSim enables the simulation of node failures in the infrastructure of Cloud. We demonstrated that the tool can be used to study the throughput of a Desktop Cloud using NotreDame and SETI@home FTA traces. We showed that the average failure rate of nodes in NotreDame and SETI@home FTAs. Such study can help to show that node failure in Desktop Cloud is quietly expected.

The results of experiments demonstrate that node failures affect negatively the throughput outcome of Desktop Clouds. However, the related works lack the ability to solve the problem of throughput decrease as a result of node failures.

This opens a new direction to design a fault tolerant mechanism for Desktop Cloud. We intend to develop such mechanism and evaluate it using the proposed tool. In addition, several metrics such as power consumption and response time should be used to evaluate VM mechanism.

References

[1] A. Alwabel, R. Walters, and G. B. Wills, "DesktopCloudSim : Simulation of Node Failures in The Cloud," in *The Sixth International Conference on Cloud Computing, GRIDs, and Virtualization CLOUD COMPUTING 2015*, pp. 14-19, 2015.

[2] R. Buyya, R. Buyya, C. S. Yeo, C. S. Yeo, S. Venugopal, S. Venugopal, J. Broberg, J. Broberg, I. Brandic, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Futur. Gener. Comput. Syst.*, vol. 25, no. 6, p. 17, Jun. 2009.

[3] B. M. Segal, P. Buncic, D. G. Quintas, D. L. Gonzalez, A. Harutyunyan, J. Rantala, and D. Weir, "Building a volunteer cloud," *Memorias la ULA*, 2009.

[4] A. Alwabel, R. Walters, and G. B. Wills, "A View at Desktop Clouds," in *Proceedings of the International Workshop on Emerging Software as a Service and Analytics*, 2014, pp. 55–61.

[5] G. Kirby, A. Dearle, A. Macdonald, and A. Fernandes, "An Approach to Ad hoc Cloud Computing," *Arxiv Prepr. arXiv1002.4738*, 2010.

[6] A. Marinos and G. Briscoe, "Community Cloud Computing," pp. 472–484, 2009.

[7] A. Andrzejak, D. Kondo, and D. P. Anderson, "Exploiting non-dedicated resources for cloud computing," in *Proceedings of the 2010 IEEE/IFIP Network Operations and Management Symposium, NOMS 2010*, 2010, pp. 341–348.

[8] A. Gupta and L. K. L. Awasthi, "Peer enterprises: A viable alternative to Cloud computing?," in *Internet Multimedia Services Architecture and Applications (IMSAA), 2009 IEEE International Conference on*, 2009, vol. 2, pp. 1–6.

[9] D. Kusic, J. O. Kephart, J. E. Hanson, N. Kandasamy, and G. Jiang, "Power and performance management of virtualized computing environments via lookahead control," *Cluster Comput.*, vol. 12, no. 1, pp. 1–15, Oct. 2008.

[10] A. Marosi, J. Kovács, and P. Kacsuk, "Towards a volunteer cloud system," *Futur. Gener. Comput. Syst.*, vol. 29, no. 6, pp. 1442–1451, Mar. 2013.

[11] C. Gong, J. Liu, Q. Zhang, H. Chen, and Z. Gong, "The characteristics of cloud computing," in *Proceedings of the International Conference on Parallel Processing Workshops*, 2010, pp. 275–279.

[12] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud Computing and Grid Computing 360-degree compared," in *Grid Computing Environments Workshop, GCE 2008*, 2008, pp. 1–10.

[13] S. Choi, H. Kim, E. Byun, M. Baik, S. Kim, C. Park, and C. Hwang, "Characterizing and classifying desktop grid," in *Proceedings - Seventh IEEE International Symposium on Cluster Computing and the Grid, CCGrid 2007*, 2007, pp. 743–748.

[14] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities," *High Perform. Comput. Simulation, 2009. HPCS'09*, pp. 1–11, Jun. 2009.

[15] R. Buyya and M. Murshed, "Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing," *Concurr. Comput. Pract.and Exp*, vol. 14, no. 13–15, pp. 1175–1220, Nov. 2003.

[16] F. Howell and R. McNab, "SimJava: A discrete event simulation library for java," *Simul. Ser.*, 1998.

[17] H. Casanova, "Simgrid: a toolkit for the simulation of application scheduling," *Proc. First IEEE/ACM Int. Symp. Clust. Comput. Grid*, pp. 430–437, 2001.

[18] S. Ostermann, K. Plankensteiner, R. Prodan, and T. Fahringer, "GroudSim: an event-based simulation framework for

computational grids and clouds," *Euro-Par 2010 Parallel Processing Workshop*, no. 261585, pp. 305–313, 2011.

[19] M. Bux and U. Leser, "DynamicCloudSim: simulating heterogeneity in computational clouds," *SWEET '13 Proc. 2nd ACM SIGMOD Work. Scalable Work. Exec. Engines Technol.*, 2013.

[20] W. Chen and M. Rey, "WorkflowSim : A Toolkit for Simulating Scientific Workflows in Distributed Environments," 2012.

[21] B. Donassolo, H. Casanova, A. Legrand, and P. Velho, "Fast and scalable simulation of volunteer computing systems using SimGrid," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing - HPDC '10*, 2010, p. 605.

[22] S. H. Lim, B. Sharma, G. Nam, E. K. Kim, and C. R. Das, "MDCSim: A multi-tier data center simulation platform," in *Cluster Computing and Workshops, 2009. CLUSTER '09. IEEE International Conference on*, 2009.

[23] D. Kliazovich, P. Bouvry, Y. Audzevich, and S. U. Khan, "GreenCloud: A Packet-Level Simulator of Energy-Aware Cloud Computing Data Centers," in *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, 2010, vol. 62, no. 3, pp. 1–5.

[24] S. McCanne, S. Floyd, K. Fall, and K. Varadhan, "Network simulator ns-2." 1997.

[25] G. Sakellari and G. Loukas, "A survey of mathematical models, simulation approaches and testbeds used for research in cloud computing," *Simul. Model. Pract. Theory*, vol. 39, pp. 92–103, Dec. 2013.

[26] A. Núñez, J. L. Vázquez-Poletti, A. C. Caminero, G. G. Castañé, J. Carretero, and I. M. Llorente, "ICanCloud: A Flexible and Scalable Cloud Infrastructure Simulator," *J. Grid Comput.*, vol. 10, no. 1, pp. 185–209, Apr. 2012.

[27] A. Núñez, J. Fernández, J. D. Garcia, L. Prada, and J. Carretero, "SIMCAN : A SIMulator Framework for Computer Architectures and Storage Networks," in *Simutools '08 Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, 2008.

[28] S. K. Garg and R. Buyya, "NetworkCloudSim: Modelling parallel applications in cloud simulations," in *Proceedings - 2011 4th IEEE International Conference on Utility and Cloud Computing, UCC 2011*, 2011, no. Vm, pp. 105–113.

[29] A. Kohne, M. Spohr, L. Nagel, and O. Spinczyk, "FederatedCloudSim: a SLA-aware federated cloud simulation framework," in *Proceedings of the 2nd International Workshop on CrossCloud Systems - CCB '14*, 2014, pp. 1–5.

[30] R. Buyya, R. Ranjan, and R. N. Calheiros, "InterCloud: Utility-oriented federation of cloud computing environments for scaling of application services," in *Algorithms and architectures for parallel processing*, 2010, vol. 6081 LNCS, no. PART 1, pp. 13–31.

[31] Í. Goiri, J. Guitart, and J. Torres, "Characterizing cloud federation for enhancing providers' profit," in *Proceedings - 2010 IEEE 3rd International Conference on Cloud Computing, CLOUD 2010*, 2010, pp. 123–130.

[32] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The eucalyptus open-source cloud-computing system," in *2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGRID 2009*, 2009, pp. 124–131.

[33] J. Fontán, T. Vázquez, L. Gonzalez, R. S. Montero, and I. M. Llorente, "OpenNEbula: The open source virtual machine manager for cluster computing," in *Open Source Grid and Cluster Software Conference – Book of Abstracts*.

[34] B. Sotomayor, R. R. S. Montero, I. M. Llorente, and I. Foster, "Virtual infrastructure management in private and hybrid clouds," *IEEE Internet Comput.*, vol. 13, no. 5, pp. 14–22, Sep. 2009.

[35] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers," *Concurr. Comput. Pract. Exp.*, vol. 24, no. 13, pp. 1397–1420, Sep. 2012.

[36] B. Javadi, D. Kondo, A. Iosup, and D. Epema, "The Failure Trace Archive: Enabling the comparison of failure measurements and models of distributed systems," *J. Parallel Distrib. Comput.*, vol. 73, no. 8, pp. 1208–1223, Aug. 2013.

[37] B. Rood and M. J. Lewis, "Multi-state grid resource availability characterization," *2007 8th IEEE/ACM Int. Conf. Grid Comput.*, pp. 42–49, Sep. 2007.

[38] B. Javadi, D. Kondo, J.-M. Vincent, and D. P. Anderson, "Mining for statistical models of availability in large-scale distributed systems: An empirical study of SETI@home," *2009 IEEE Int. Symp. Model. Anal. Simul. Comput. Telecommun. Syst.*, pp. 1 – 10, 2009.

[39] L. Peterson, S. Muir, T. Roscoe, and A. Klingaman, "PlanetLab Architecture : An Overview," no. May, 2006.

[40] K. Park and V. S. Pai, "CoMon," *ACM SIGOPS Oper. Syst. Rev.*, vol. 40, no. 1, p. 65, Jan. 2006.

[41] A. Field, *Discovering statistics using SPSS*, Third. SAGE Publications Ltd, 2009.

[42] L. A. Barroso and U. Hölzle, *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*, vol. 4, no. 1. 2009.

[43] S. Srikantaiah, A. Kansal, and F. Zhao, "Energy aware consolidation for cloud computing," in *HotPower'08: Proceedings of the 2008 conference on Power aware computing and systems*, 2008.

[44] P. Graubner, M. Schmidt, and B. Freisleben, "Energy-efficient management of virtual machines in Eucalyptus," in *2011 IEEE 4th International Conference on Cloud Computing*, 2011, pp. 243–250.

[45] A. Verma, P. Ahuja, and A. Neogi, "pMapper: power and migration cost aware application placement in virtualized systems," in *Middleware '08 Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*, 2008, pp. 243–264.

[46] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing," *Futur. Gener. Comput. Syst.*, vol. 28, no. 5, pp. 755–768, May 2012.

[47] X. Meng, C. Isci, J. O. Kephart, L. Zhang, E. Bouillet, and D. Pendarakis, "Efficient resource provisioning in compute clouds via VM multiplexing," *Proceeding 7th Int. Conf. Auton. Comput. - ICAC'10*, p. 11, 2010.

[48] A. Beloglazov and R. Buyya, "Energy efficient resource management in virtualized cloud data centers," in *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, 2010, pp. 826–831.

[49] R. Buyya, A. Beloglazov, and J. Abawajy, "Energy-Efficient Management of Data Center Resources for Cloud Computing : A Vision, Architectural Elements, and Open Challenges," *arXiv Prepr. arXiv1006.0308*, no. Vm, pp. 1–12, 2010.

[50] V. Vazirani, *Approximation algorithms*, Second. New York, New York, USA: Springer, 2003.

[51] E. Feller, C. Rohr, D. Margery, and C. Morin, "Energy management in IaaS clouds: A holistic approach," in *Proceedings - 2012 IEEE 5th International Conference on Cloud Computing, CLOUD 2012*, 2012, pp. 204–212.

[52] X. Liao, H. Jin, and H. Liu, "Towards a green cluster through dynamic remapping of virtual machines," *Futur. Gener. Comput. Syst.*, vol. 28, no. 2, pp. 469–477, Feb. 2012.

[53] H. N. Van, F. D. Tran, and J. M. Menaud, "SLA-aware virtual resource management for cloud infrastructures," in *Proceedings - IEEE 9th International Conference on Computer and Information Technology, CIT 2009*, 2009, vol. 1, pp. 357–362.

[54] K. Tsakalozos, M. Roussopoulos, and A. Delis, "VM placement in non-homogeneous IaaS-clouds," in *Service-Oriented Computing*, vol. 7084 LNCS, G. Kappel, Z. Maamar, and H. R. Motahari-Nezhad, Eds. Springer Berlin Heidelberg, 2011, pp. 172–187.

[55] W. Shi and B. Hong, "Towards Profitable Virtual Machine Placement in the Data Center," *2011 Fourth IEEE Int. Conf. Util. Cloud Comput.*, pp. 138–145, Dec. 2011.

[56] N. M. Calcavecchia, O. Biran, E. Hadad, and Y. Moatti, "VM placement strategies for cloud scenarios," in *Proceedings - 2012 IEEE 5th International Conference on Cloud Computing, CLOUD 2012*, 2012, pp. 852–859.

[57] J. T. Piao and J. Yan, "A Network-aware Virtual Machine Placement and Migration Approach in Cloud Computing," *2010 Ninth Int. Conf. Grid Cloud Comput.*, pp. 87–92, Nov. 2010.

[58] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *Proceedings - IEEE INFOCOM*, 2010.

[59] B. Palanisamy, A. Singh, L. Liu, and B. Jain, "Purlieus: Locality-aware resource allocation for MapReduce in a cloud," *2011 Int. Conf. High Perform. Comput. Networking, Storage Anal.*, pp. 1–11, 2011.

[60] J. Dean and S. Ghemawat, "MapReduce: Simplied Data Processing on Large Clusters," in *Proceedings of 6th Symposium on Operating Systems Design and Implementation*, 2004, vol. 103, no. 34, pp. 137–149.

[61] O. Biran, A. Corradi, M. Fanelli, L. Foschini, A. Nus, D. Raz, and E. Silvera, "A stable network-aware VM placement for cloud systems," in *Proceedings - 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid 2012*, 2012, pp. 498–506.

[62] F. P. Tso, G. Hamilton, K. Oikonomou, and D. P. Pezaros, "Implementing scalable, network-aware virtual machine migration for cloud data centers," in *IEEE International Conference on Cloud Computing, CLOUD*, 2013, pp. 557–564.

[63] A. Litke, D. Skoutas, K. Tserpes, and T. Varvarigou, "Efficient task replication and management for adaptive fault tolerance in Mobile Grid environments," *Futur. Gener. Comput. Syst.*, vol. 23, no. 2, pp. 163–178, 2007.

[64] H. L. H. Lee, D. P. D. Park, M. H. M. Hong, S.-S. Y. S.-S. Yeo, S. K. S. Kim, and S. K. S. Kim, "A Resource Management System for Fault Tolerance in Grid Computing," *2009 Int. Conf. Comput. Sci. Eng.*, vol. 2, pp. 609–614, 2009.

[65] F. Machida, M. Kawato, and Y. Maeno, "Redundant virtual machine placement for fault-tolerant consolidated server clusters," in *Proceedings of the 2010 IEEE/IFIP Network Operations and Management Symposium, NOMS 2010*, 2010, pp. 32–39.

[66] Y. Zhang, Z. Zheng, and M. R. Lyu, "BFTCloud: A Byzantine Fault Tolerance Framework for Voluntary-Resource Cloud Computing," *2011 IEEE 4th Int. Conf. Cloud Comput.*, pp. 444–451, Jul. 2011.

[67] B. Javadi, D. Kondo, J. Vincent, and D. P. Anderson, "Discovering Statistical Models of Availability in Large Distributed Systems: An Empirical Study of SETI@home," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 11, pp. 1896–1903, 2011.

[68] B. Javadi, J. Abawajy, and R. Buyya, "Failure-aware resource provisioning for hybrid Cloud infrastructure," *J. Parallel Distrib. Comput.*, vol. 72, no. 10, pp. 1318–1331, Oct. 2012.