DesktopCloudSim: Simulation of Node Failures in the Cloud

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*—Simulation tools are commonly used by researchers to simulate Clouds in order to study various research issues and test proposed solutions. CloudSim is widely employed to simulate Cloud computing by both academia and industry. However, it lacks the ability to simulate failure events which may occur to physical nodes in the infrastructure level of a Cloud. This paper proposes DesktopCloudSim tool as an extension developed to overcome this shortage. In order to demonstrate the effectiveness of this tool, we evaluate the throughput of simulating a private Cloud built on top of faulty nodes based on empirical data collected from NotreDame Desktop Grid.**

Keywords-Cloud; CloudSim; DesktopCloudSim; Failure; Nodes.

I. INTRODUCTION

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 [1]. However, a new direction of Cloud has recently emerged with an aim to exploit normal Desktop computers, laptops, etc. to provide Cloud services [2]. This kind of Cloud can be called Desktop Clouds [3]. 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 main advantage over Traditional Clouds. Researchers in Desktop Clouds can benefit from Cloud services at little cost, if not free. However, such feature comes with a price. 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 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 Desktop Cloud based on empirical data of failures in NotreDame 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 motivates the idea of harnessing idle resources 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 [4], Volunteer Cloud [2], Community Cloud [5] and Non-Dedicated Cloud [6]. The literature shows that very little work has been undertaken in this direction.

There are some differences between Desktop Clouds and Traditional Clouds. 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 utilized 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 distributed across the globe, whereas they are limited in Traditional Cloud to several locations in data centres. Furthermore, nodes in Desktop Cloud are highly volatile due to the fact that they can be down unexpectedly without prior notice. Node failures can happen 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 [7]. 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 similar distributed systems, specifically Desktop Grids. Both Desktop Clouds and Desktop Grids share the same goal 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 [10]. 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 [11]. 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 [12]. 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 to 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 [13]. 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 [14] and SimJava [15] simulation tools.

CloudSim is claimed to be more effective in simulating Clouds compared to SimGrid [16] and GroudSim [17] because CloudSim allows segregation of multi-layer service (Infrastructure as a Service, Platform as a Service and Software as a Service) abstraction [13]. 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 [18]. 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 service-level agreements. 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 [18]. Secondly, service failures are not simulated in CloudSim [19]. 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.

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.

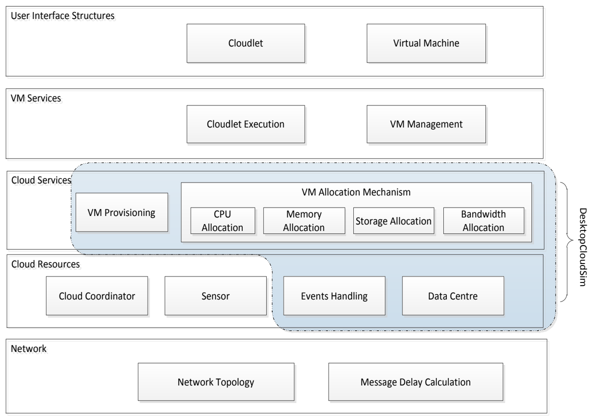

Figure 1. DesktopCloudSim Abstract

Figure 2. shows the components of DesktopCloudSim. The simulation starts by reading failure trace file(s). The trace files contain the specifications of the simulated nodes and failure events. The *Failure Analyser* component analyses the files of failures to send node specifications to *Create Nodes* component and failure events to *Failure Injection* component. Node specifications are the physical specifications of nodes, such as CPU, RAM. etc. *Create Nodes* component creates the nodes of a Desktop Cloud according to the given specification. *Failure Injection* component receives failure events from the *Failure* Analyser to inject failures into associated nodes during run time. The *Failure Injection* component informs the *VM Mechanism* unit if a node is failed to let it restart failed VMs on another alive node or nodes. *VM Provisioning* component provisions VMs to clients to execute tasks.

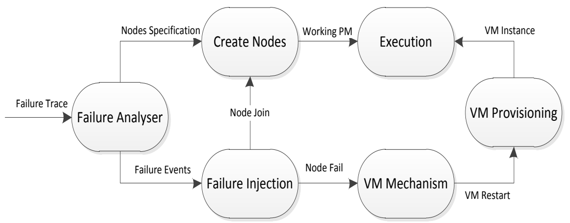

Figure 2. DesktopCloudSim Model

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. These mechanisms are implemented in open source Cloud management platforms, such as Eucalyptus [20], OpenNebula [21] and Nimbus [22].

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 [23]. The FCFS mechanism is expected to yield the lowest throughput among the aforementioned mechanisms because it assigns VMs to PMs in somehow random manner.

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 IV.A discusses further this archive. The second input set is the workload submitted to the Desktop Cloud during running time. Subsection IV.B talks about this workload.

A. Failure Trace Archive

Failure Trace Archive (FTA) is a public repository containing traces of several distributed and parallel systems [24]. 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 FTA is collected from the University of Notre Dame. The trace represents an archive of a pool of heterogeneous resources that have run for 6 months within the University of Notre Dame during 2007 [25]. 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.

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

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

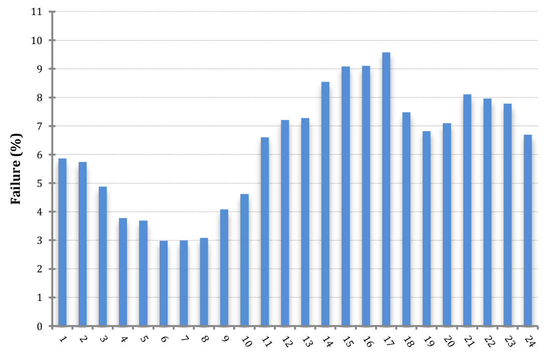

Figure 3. Average Hourly Failure

Figure 3 shows an average hourly failure percentage in 24 hour-period for analysis of 6 months run times of NotreDame 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. However, it was recorded that the percentage of node failures can reach up to 80% in some hours. This can demonstrate that failure events in Desktop Clouds are norms rather than exceptions.

B. *Experiment Setting*

The experiment is run for 180 times, each time representing a simulation of running NotreDame Desktop Cloud for one day. The run time was 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 540 (3 * 180). The workload was collected from the PlanetLab archive. The archive provides traces of real live applications submitted to the PlanetLab infrastructure [26]. 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 [27]. 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 rejoin 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 I shows a summary of descriptive results obtained when measuring the throughput output for each VM allocation mechanism implemented in NotreDame Cloud. N in the table means that the number of days is 180 days representing a six-month period. 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, $P < .05$. 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 one mechanism to another, $X_F^2(3) = 397.14, P < .001$. Mean, median, variance and standard deviation are reported in Table I.

TABLE I. THROUGHPUT RESULTS

Mechanism	N	Mean	Median	Var.	St. Dev.	K-S Test
FCFS	180	79.21 %	78.77 %	37.03	6.09	$P < .05$
Greedy	180	88.61 %	89.48 %	16.85	4.1	$P < .05$
RoundRobin	180	85.47 %	85.29 %	15.13	3.89	$P = .2$

Three Wilcoxon pairwise comparison tests were conducted to find out which mechanism had the 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 [28] method because there were 3 post-hoc tests required (.05/3 ≈ .017). The tests show that there is a significant difference between each mechanism with its counterpart. Therefore, it can be concluded that the Greedy mechanism yields the highest throughput since it has the median with highest value (median = 89.48%).

The median throughput of FCFS was about 79%, as being the worst mechanism among the tested mechanisms. Our findings confirm our expectation in section III.C. 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 89% when Greedy VM mechanism was employed. The mechanism aims at maximising utilisation by packing as many VMs as possible to the same PM, thus reducing the number of running PMs. The average failure rate in submitted tasks is about 12%, given the average node failure percentage is about 6% as section IV.A shows. Such figures demonstrate the importance to develop fault tolerant VM mechanisms.

VI. Related work

Several simulators have been published to simulate Grid computing. SimGrid [16] is one of the early simulation tools to simulate Grid environment. GridSim [14] is another tool that fits within the same goal. CloudSim is built on top of GridSim. Donassole et al. [29] 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 [17] 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 short in providing virtualisation feature which is essential to evaluate VM allocation mechanisms.

WorkflowSim [19] 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 the scientific workflow. The authors add a new management layer to deal with overhead of complex scientific computational tasks. The authors argue that CloudSim fails in simulating the overhead of such tasks. The overhead may include queue delay, data transfer delay clustering delay and postscript. This issue may affect the credibility of findings and 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 a failure in a task causes a series of jobs to fail. However our work differs from WorkflowSim in the failure event and its impact. We focus on infrastructure level which contains nodes that host VMs whereas the authors are interested in the service level, i.e., the tasks and applications. We argue that service providers should consider developing failure tolerant mechanisms to overcome failures events in the infrastructure level.

DynamicCloudSim [18] is another extension for CloudSim tool. The authors are 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 behaviour of VMs. The tool can be used to evaluate scientific workflow schedulers taking in consideration the variance of VM performance. In addition, execution time of a given task is influenced by I/O-bound such as reading or writing data. The authors extend instability to include tasks failure. Performance variation of running VMs is an open research challenge, but it is out of this scope.

VII. Conclusion and future work

Desktop Cloud represents a new direction in Cloud computing. Desktop Cloud aims at exploiting idle computing resources to provide Cloud services mainly for research purposes. The success of Desktop Grids in providing Grid capabilities has stimulated the idea 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 a negative impact 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 real traces. We showed that Greedy VM mechanism yielded better throughput in the presence of failures compared to the FCFS and RoundRobin mechanism.

The results of experiments demonstrate that node failures affect negatively the throughput outcome of Desktop Clouds. This opens a new direction to design a fault tolerant mechanism for Desktop Cloud. We intend to develop such a 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] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, 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, pp. 599–616, Jun. 2009.

[2] 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.

[3] A. Alwabel, R. Walters, and G. Wills, “A view at desktop clouds,” in *ESaaSA 2014*, 2014.

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

[5] A. Marinos and G. Briscoe, “Community Cloud Computing,” pp. 472–484, 2009.

[6] A. Andrzejak, D. Kondo, and D. P. Anderson, “Exploiting non-dedicated resources for cloud computing,” *2010 IEEE Netw. Oper. Manag. Symp. - NOMS 2010*, pp. 341–348, 2010.

[7] A. Marosi, J. Kovács, and P. Kacsuk, “Towards a volunteer cloud system,” *Futur. Gener. Comput. Syst.*, Mar. 2012.

[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] C. Gong, J. Liu, Q. Zhang, H. Chen, and Z. Gong, “The Characteristics of Cloud Computing,” *2010 39th Int. Conf. Parallel Process. Work.*, pp. 275–279, Sep. 2010.

[11] I. Foster, Y. Zhao, I. Raicu, and S. Lu, “Cloud computing and grid computing 360-degree compared,” in *Grid Computing Environments Workshop, 2008. GCE’08*, 2008, pp. 1–10.

[12] S. Choi, H. Kim, E. Byun, M. Baik, S. Kim, C. Park, and C. Hwang, “Characterizing and Classifying Desktop Grid,” in *Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid ’07)*, 2007, pp. 743–748.

[13] 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.

[14] 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. …*, vol. 14, no. 13–15, pp. 1175–1220, Nov. 2003.

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

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

[17] 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 Workshops*, no. 261585, pp. 305–313, 2011.

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

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

[20] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, D. Zagorodnov, and S. Barbara, “The Eucalyptus Open-Source Cloud-Computing System,” *2009 9th IEEE/ACM Int. Symp. Clust. Comput. Grid*, pp. 124–131, 2009.

[21] 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*.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

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

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

[28] A. Field, *Discovering statistics using SPSS*, Third. SAGE Publications Ltd, 2009, p. 856.

[29] 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.