2013 27th International Conference on Advanced Information Networking and Applications Workshops

Towards inter-cloud simulation performance analysis: Exploring service-oriented benchmarks of clouds in SimIC

Stelios Sotiriadis, Nik Bessis, Nick Antonopoulos
School of Computing & Maths, University of Derby, Derby, United Kingdom
(s.sotiriadis, n.bessis, n.antonopoulos)@derby.ac.uk

***Abstract* — On recent years, much effort has been put in analyzing the performance of large-scale distributed systems like grids, clouds and inter-clouds with respect to a diversity of resources and user requirements. A common way to achieve this is by using simulation frameworks for evaluating novel models prior to the development of solutions in highly cost settings. In this work we focus on the SimIC simulation toolkit as an innovative discrete event driven solution to mimic the inter-cloud service formation, dissemination, and execution phases; processes that are bundled in the inter-cloud meta-scheduling (ICMS) framework. Our work has meta-inspired characteristics as we determine the inter-cloud as a decentralized and dynamic computing environment where meta-brokers actas distributed management nodes for dynamic and real-time decision making in an identical manner. To this extend, we study the performance of service distributions among clouds based on a variety of metrics (e.g. execution time and turnaround) when different heterogeneous inter-cloud topologies are taking place. We also explore the behavior of the ICMS for different user submissions in terms of their computational requirements. The aim is to produce the results for a benchmark analysis of clouds in order to serve future research efforts on cloud and inter-cloud performance evaluation as benchmarks. The results are diverse in terms of different performance metrics. Especially for the ICMS, an increased performance tendency is observed when the system scales to massive user requests. This implies the improvement on scalability and service elasticity figures.**

Keywords: Inter-Cloud; SimIC; CloudSim; ICMS; Meta-broker resource management; cloud benchmarks.

INTRODUCTION

Recently, inter-cloud has been promoted as the next step of the Internet evolution with regards of realizing a wider service capability among collaborated computing clouds. This is based on the cloud attitude that implies an elastic service execution setting by giving the infinite resource impression to users. However, as the setting grows, the capacity of flexible cloud datacenters decreases due to their static-ness, thus making vital to evolve to an inter-connecting setting that continues to offer this endless service elasticity. This is the inter-cloud as proposed by [2] wherein, users utilize remote resources in a bespoke based model by submitting services that are executed within Virtual Machines (VMs); a process that is called sandboxing. Based on that, the inter-cloud meta-scheduling (ICMS) has been proposed in [8] as a novel framework that offers the fundamental capabilities for service distribution among inter-clouds on an analogous tactic to grid computing distributed management systems.

We utilize the meta-brokering model for achieving such functionality. This indicates that meta-components are placed on the top of cloud decision-making entities named as local-brokers. The novelty of our approach stands on the consideration of dynamic and real-time decision-making processes for allowing service and VM exchanging in order to achieve various performance criteria. Real-time decision making is considered as an important criterion for achieving realistic resource management [10]. Our implementation allows heterogeneous service distribution on spontaneous choices on scheduling and resource management policies. The ICMS is composed from a set of sub-scheduling heuristics that aim to request and accept connection with remote sites, distribute requests within that system, search for available resources, allocate the resource based on criteria, execute VMs within resources, and monitor the whole procedure for keeping performance measures. The entire functionality is based upon the meta-computing paradigm that suggests decentralized resource managers are interconnected to co-sites that have a corresponding architecture.

Therefore, this work presents a performance analysis study of the cloud service submission implemented in CloudSim and SimIC. The 'Simulating the Inter-Cloud' (SimIC) toolkit is a discrete event simulation framework that mimics an inter-cloud service dissemination setting. Specifically, the toolkit is developed using the SimJava package that allows event exchanging among components in terms of messages that are sending among the system entities at different time intervals as indicated by the users. This allows us to model inter-clouds wherein various users submit various requests on different phases. Fundamentally we implement a diversity of performance metrics including service makespan, VM execution times, request turnaround times, throughput of entities, resource utilization, response ratio, energy consumption of datacenters, VMs utilization cost, and service latency figures.

For demonstrating our performance benchmarks we compare identical cloud configurations that are implemented in the CloudSim [3] and SimIC [8] respectively. It should be mentioned that the original design decision of SimIC is based on CloudSim core classes. However, for achieving experimentation on dynamic and real-time multi-user submissions (e.g. in latency of setting) that includes meta-computing functionality (operations that are not supported by the available CloudSim version

978-0-7695-4952-1/13 $26.00 © 2013 IEEE
DOI 10.1109/WAINA.2013.196

presented in [3] the whole toolkit has been re-developed. This includes the development of the core cloud entities (e.g. clouds datacenter, hosts, VMs, as well as resource management policies) as well as meta-brokers and hypervisors of the system using SimJava enabled concepts. Based on that, the experimental analysis of the two settings shows that SimIC compliments CloudSim features by presenting comparable performance analysis on service execution times (e.g. VMs execution times). In addition, the consideration of meta-schedulers (meta-brokers) that dynamically control computational capacity of datacenters in an inter-cloud setting along with real-time scheduling and policy management of jobs and entities empowers the potentials of SimIC. Finally, the results produced from the SimIC will serve future research studies as benchmarks for clouds and inter-clouds performance analyses.

The next section II presents a discussion of the related works and the motivation of our study. The rest of the paper is organized as follows, section III demonstrates the algorithmic design of ICMS and the architectural strategy of SimIC in terms of inter-cloud connectivity, and section IV presents the experimental analysis by the inductive evaluation of functional scenarios of clouds and inter-clouds. Finally, section V illustrates a discussion on the contribution of our benchmarks for adding future directions as well as the conclusions of our study.

MOTIVATION AND RESEARCH BACKGROUND

The huge grow of distributed infrastructures (e.g. clouds) in terms of resources that are utilized remotely has created new demands of how to understand and evaluate the results produced from such high-performance settings. However, the performance as a meaning could include different perspectives of time and resource usage related with the origin of the system. This study aims of identifying experimental results produced from different simulation cases of clouds and inter-clouds in order to become the benchmarks for future evaluations by giving specific performance metrics analysis. A solution to achieve this, is through simulation of identical user submissions in different simulation settings for exploring the experimental features and analyzing results.

However, a general appreciation of all simulations is unrealistic due to the different requirements that the system is developed upon. This is realted with the requirements of the simulated systems [9]. Specifically, grid simulators like GridSim etc. as discussed in [8]have been determined as inappropriate to be used in cloud settings mainly because of the multi-layer structure of the cloud service abstraction [3]. In addition, virtualization is considered as one of the key cloud elements that it is not also included in grid simulation toolkits. This includes the elastic factor that facilitates a cloud system. Thus, the motivation of our study is focused on the cloud computing simulators and their performance results in order to meet the diverse requirements of user submissions in terms of time and resource usage.

So, for the case of clouds and inter-cloud none of the cluster or grid solutions can address the arising application level requirements. This is because of the application-oriented cloud emphasis and of the elasticity of services in the pay-on-demand model [5]. For that reason authors in [3] presented the CloudSim simulation framework that aims to simulate subscribed services that are delivered to the users in an elastic cloud setting. A different simulation case for clouds is the iCanCloud simulator [4] that is a solution for optimizing related performance criteria (e.g. VM performance).

Both toolkits offer capabilities in developing cloud service submissions in cloud datacenters. However, the design of our simulator was originally based on the CloudSim toolkit (architecture and multi-layered structure) thus in this work we explore performance analysis of SimIC over the CloudSim in order to capture results, support our design modeling decisions, and evaluate identical cloud settings when various users request for identical resources in both toolkits. It should be mentioned that the SimIC compliments SimCloud setting in terms of extending certain interoperable and dynamic functionalities. To this extend, we execute the same experimentation in the toolkits, including entities such as hosts and virtual machine (VM) as well as their configurations.

In advance, the study goes one step further by presenting simulation results of the inter-cloud service submission of the ICMS framework [8]. The last one offers the inter-cloud functionality by considering a large-scale resource management setting wherein real-time and dynamic scheduling is taking place. Both experimental cases aim on a) presenting the cloud benchmark analysis and b) the inter-cloud benchmark presentation to be valuable for future evaluations. Through this analysis, prospect experimental results are presented for the meta-brokering model. The ICMS vision is to design a total decentralized meta-broker based on our previous inter-cloud model presented in [7]. This will offer significant advantages, as it will support highly interoperability, flexibility and service heterogeneity (thus resource as well) while at the same time a job execution manner in a decentralized fashion. The next section illustrates the architectural analysis of the SimIC as well as the inter-cloud capability of the ICMS framework.

ARCHITECTURAL REQUIREMENTS OF CLOUDSIM AND ICMS OVER THE SIMIC

The ICMS is a set of algorithms for achieving service distribution in large-scale inter-clouds that aim to accomplish the following.

a) Request and accept connection with remote sites through meta-brokers.
b) Distribute service submission requests within an inter-cloud system and their entities.
c) Search for available resources on a real-time submission using dynamic workload management.
d) Allocate the resource based on performance criteria related with the required computational capacity.
e) Execute a VM within a selected resource by sandboxing the user requirements.

f) Monitor the whole procedure for keeping a diversity of performance measurements.

The processes discussed above have been implemented within the SimIC simulation toolkit in order to achieve the meta-brokering and dynamic service dissemination among interoperable clouds. In general the SimIC offers a flexible and elastic service submission environment with the following characteristics.

a) It achieves large-scale distribution of job requests among meta-brokers that are inter-connected in random topologies.
b) It offers decentralized topologies of meta-brokers for realistic large-scale scheduling.
c) It includes static and dynamic management policies of dynamic workload management.
d) It includes static and dynamic service level agreement (SLA) matchmaking policies among meta-brokers.
e) It offers static and dynamic instantiation of VMs with regards to history records.
f) It achieves real-time job scheduling in VMs according to a variety of heuristic scheduling criteria (e.g. preemptive and non-preemptive cases).
g) It includes dynamic queuing of VMs according to selected schedulers (e.g. shortest job first etc.).
h) It addresses VM migration according to cloud provider requirements.
i) It offers re-active management of heterogeneous services submission in the form of VMs.

The CloudSim [3] involves a set of cloud entities in order to simulate the VM allocation according to specific requirements. Particularly, during the simulation construction the modeler determines the VM parameters e.g. cpu size, ram etc., the job parameters, named as cloudlet, that includes the length of the job, the number of cores etc., the datacenter configuration that includes the number of datacenters, hosts and computational power. In addition, datacenter characteristics involve system architecture and costs. Finally, the setting contains various policies for sharing host capacity among VMs (e.g. the cloudlet space sharing policy). The simulation starts when a modeler specifies the required parameters and decides the number of users (determined by cloudlets) that are about to enter the simulation setting and to be submitted to a broker that forwards the request fro resource allocation.

The SimIC [8] involves a similar preliminary configuration pattern however; this involves additional requirements such as the number of users, the latency of user submissions and the number of jobs that a user submits. In addition, the host and datacenter parameters (along with user requirements) are defined in text files that can be easily adapted and loaded into the simulator. The user is also represented by an SLA matchmaking policy that includes dynamic workload management based on current workloads of clouds. SimIC involves the reflection of the local policies in a hypervisor component that collectively manages the whole service submission of the cloud, e.g. VM scheduling and deferred queues formations. Finally, in the simulation configuration file the user could design the number of users and jobs, the delay on submissions, the number of clouds their inter-relationships, the latency of each entity to respond to requests and the loading of user and host requirements into the toolkit.

The great advantage of SimIC includes the profiling of user requirements that is shared among entities without exposing all user information but only the parameters that are required from each entity. In addition, policies on VM instantiation, message exchanging information, local resource management policies and energy-aware-ness experimentation are also included. The next section presents the experimental analysis of a hybrid dataset that is submitted to both simulation settings in order to measure the performance of cloud service execution in CloudSim and SimIC that will serve as benchmarks.

EXPERIMENTAL ANALYSIS AND BENCHMARK RESULTS OF CLOUDSIM AND SIMIC

This section presents the experimental analysis of CloudSim and SimIC when identical user submissions are taking place. We execute a basic experiment initially in CloudSim and we present the results produced when the system encompasses the configuration of tables I and II.In particular, table I demonstrates the cloud host and datacenter configuration while table II shows the number of users and the requested computational performance measurements.

TABLE I: CLOUD CONFIGURATION PARAMETERS FOR INPUT IN CLOUDSIM AND SIMIC

Host Requirements	***Experiment parameters***
Mips:	1000
RAM:	2048
Storage:	1000000
Bandwidth:	10000
Host number:	2
Host 1:	4
Host 2:	2
Datacenters:	2

TABLE 2: USER CONFIGURATION PARAMETERS FOR INPUT IN CLOUDSIM AND SIMIC

VM Requirements	***Experiment parameters***
CPU size:	1000
RAM:	512
Mips:	1000
Bandwidth:	1000
Cores:	1

Further to this, we present the results as produced by the SimIC and we compare both cases for the average execution time of the service submission (presented as VMs). The performance measures are given by formula (1) that calculates the millions of instructions per second (mips) as a rate for operations per unit used by the CloudSim and SimIC.

$$mips = \frac{clock\ rate}{cpi} * 10^{-6} \quad (1)$$

Similarly, (2) calculates the execution time of job in VM in terms of instructions count submitted and cycles per instruction as used by SimIC. The *h* parameter demonstrates the time duration of the VM leasing by the cloud user.

$$ExecTime_{VM} = \left(\frac{Instruction}{Program}\right)_{user} \times CPI_{user} \times \frac{1}{CPU_{VM}} \times \frac{1}{CPUCores_{VM}} \times h \quad (2)$$

The execution time of the service is given by formula (3) that contains the total latency of the system for the service to reach the destination VM as follows.

$$ExecTime_{service} = Latency_{user-vm} + PerformanceTime_{VM} \quad (3)$$

For identifying the performance benchmark of cloud submission we execute a number of user submissions for VMs in CloudSim. This includes 1 user to 1 VM request towards 100 users with 100 VM requests. It should be mentioned that CloudSim shares the computational power of cpu cores in space sharing policy, thus the VM execution time is increased for high workloads. In parallel, SimIC utilizes an identical feature that dynamically allocates more resources in order to fulfill the requests. Figure 1 demonstrates the experimental analysis of CloudSim and SimIC in which the same input configuration of users and requests has been submitted.

It is apparent that both systems execute jobs in a parallel execution trend, however by operating in different For example, CloudSim shares computational power of hosts within a datacenter in order to fulfill all the requests, while SimIC considers a policy for dynamic CPU sharing by considering a latency that increases the VM execution time as presented in [8]. In any case, the fundamental benchmark analysis shows that for high workloads (greater than 50 users) both simulators offer corresponding parallel increased trend line of the VM execution times. Accordingly, based on figure 1 we propose that SimIC operates in corresponding with CloudSim, wherein a slightly optimization of execution time could be observed (mainly due to the low latency of dynamic allocations). Nevertheless, the experiment illustrates that benchmark analysis in both toolkits offer identical result output.

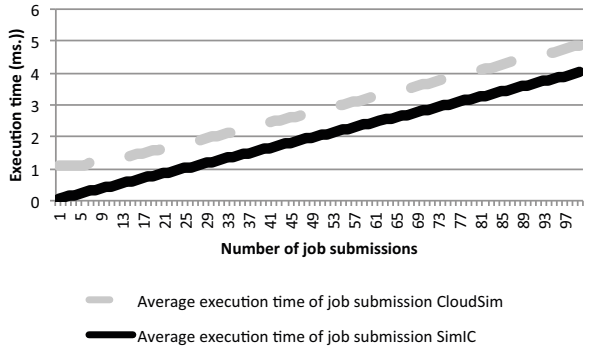

Figure 1: The comparison of CloudSim and SimIC for one cloud specification of 1-100 user submissions for 1-100 VMs.

In order to present the novelty of the SimIC we address a more complex objective that allows job distribution among different clouds that accept the SLAs posed by users. In that way we implement various realistic scenarios wherein collaborated clouds exchange information on behalf of the user request on run-time by always checking the SLA specification with regards to current execution workload and capacity of sub-clouds to execute certain requests. We implemented an inter-cloud of 8 clouds wherein various set of users (16 and 32) submit jobs (10 per user). The distribution algorithm aims of allocating jobs to clouds that fulfill specific requirements that are sandboxed in VMs. For our use case we implement a scenario with regards to a) run-time decision-making and b) dynamic workload management. The next sections present four experimental cases that are executed in SimIC within an inter-cloud setting. It should be mentioned that the inter-cloud meta-brokering topology implies that each meta-broker is interconnected with the next one (e.g. meta-broker 1 to meta-broker 2, meta-broker 2 to meta-broker 3 etc.).

Case 1: 160 jobs submitted by 16 users(partial SLAs)

In first case each user submits an identical job specification that can be served only from the clouds 1, 4, 5 and 8. This is because of the heterogeneity factor of the service submissions that require matching with the competency of the system to execute the requests (SLA matchmaking). In addition, the dynamic workload management allows services that cannot be executed locally to be forwarded to capable resources that can offer the computational capacity. In this setting we implement the execution time of the VM that reflects the current system delay (measured as turnaround time) given by formula (4). It should be mentioned that the time interval of VM usage is not determined to affect performance (h is set to 1).

$$TurnTime_{cloud} = \left\{\frac{instructions_{VM} * CPI_{VM}}{CPU_{VM} * CPUCores_{VM} * 10^5}\right\}_{cloud} + CurrentTime \quad (4)$$

The second metric is the makespan that demonstrates the sum-up of the VM execution time plus the total delay due to service dissemination given by formula (5).

$$Makespan_{job} = VMexecutionTime + totalDelayTime \quad (5)$$

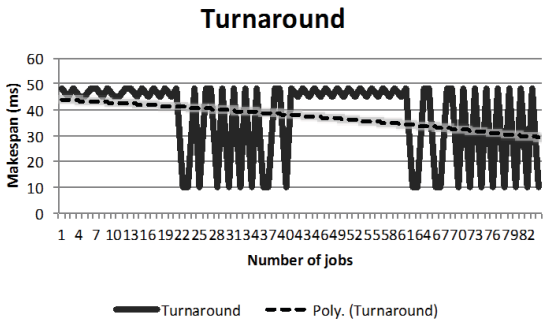

Figure 2: The turnaround and the polynomial trend line of turnaround timesof SimIC for 160 identical jobs submitted by 16 users (10 jobs per user) -Jobs can be executed from clouds 1, 4, 5 and 8 (SLA matchmaking).

By executing this scenario case, we extract results that are presented in figures 2, 3 and 4. Specifically, figure 2 demonstrates the turnaround time and the polynomial trend

line of this value of the SimIC for 160 identical jobs submitted by 16 users (10 jobs per user). As initially discussed jobs can be executed from clouds 1, 4, 5 and 8 (SLA matchmaking). The trendline denotes that as the number of user submissions increase the system tends to offer improved turnaround times by achieving job executions for all the set of jobs.

Figure 3 shows the makespan values of SimIC when 160 identical jobs submitted by 16 users (10 jobs per user) enter the simulators. Again the same constraint includes that jobs can be executed from clouds 1, 4, 5 and 8. In particular, the turnaround times are tend to increase due to the distribution of jobs among meta-brokers in order to achieve job execution of the whole input set. In general, the makespan times of the last jobs have been showing an increased tendency to a factor of 0.658 (this is calculated as the division of the average value of result set by 1000) that we consider as an acceptable rate mainly because of the large submission number. Specifically, this will serve as a metric for comparison with next scenario cases.

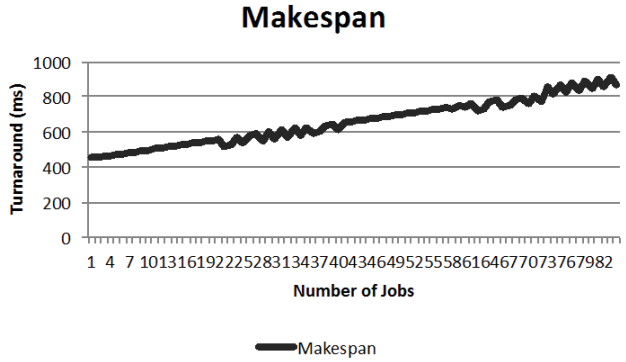

Figure 3: The makespan values of SimIC when 160 identical jobs submitted by 16 users (10 jobs per user) -Jobs can be executed from clouds 1, 4, 5 and 8 (SLA matchmaking).

Figure 4 shows the cloud allocation numbers of SimIC when 160 identical jobs submitted by 16 users (10 jobs per user) that can be executed from clouds 1, 4, 5 and 8 (SLA matchmaking). It is apparent that clouds 4 and 8 served the most of the service submissions mainly because of the meta-brokering topology.

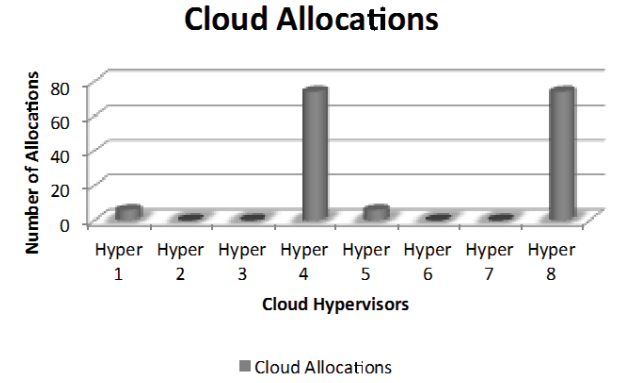

Figure 4: The cloud allocation numbers of SimIC when 160 identical jobs submitted by 16 users (10 jobs per user) – Jobs can be executed from clouds 1, 4, 5 and 8 (SLA matchmaking).

Case 2: 160 jobs submitted by 16 users(full SLAs)

This case scenario involves the experimental input of 160 identical jobs submitted by 16 users (10 jobs per user) wherein all clouds can offer job execution. However, in this case the dynamic workload management defines the dissemination functionality. This involves that if a cloud cannot execute the job due to limited resources then it sends the job back to the meta-broker for further dissemination. Figure 5 shows the turnaround and the polynomial trend line of the turnaround times of SimIC for 160 identical jobs submitted by 16 users (10 jobs per user) where all clouds can offer job execution. It is apparent that the turnaround polynomial trend line shows an increasing trend for 50 to 100 job submissions; however for 100 to 160 the line shows a decreasing rate. That is considered as an improvement because the system tends to offer better performance (decrease turnaround time trends) for peak workloads.

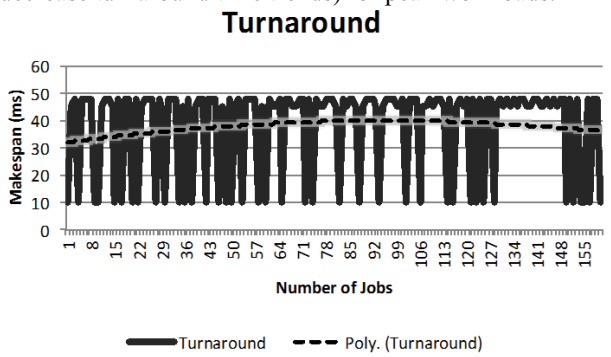

Figure 5: The turnaround and the polynomial trend line of the turnaround of SimIC for 160 identical jobs submitted by 16 users (10 jobs per user) - all clouds can offer job execution (SLA matchmaking).

Figure 6 presents the makespan values of SimIC when 160 identical jobs submitted by 16 users in an identical case as previously. The chart shows an increasing trend of the makespan with ratio factor of 0.59 that it is marginally lowest compared with case 1.

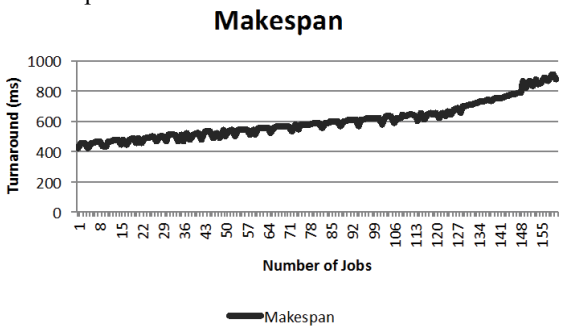

Figure 6: The makespan values of SimIC when 160 identical jobs submitted by 16 users (10 jobs per user) - all clouds offer job execution.

Figure 7 presents the job allocation among clouds and their hypervisor that are responsible for creating and allocating VMs. It is apparent that clouds 2 and 6 served most of the services. This is again mainly because of the inter-cloud meta-brokering topology.

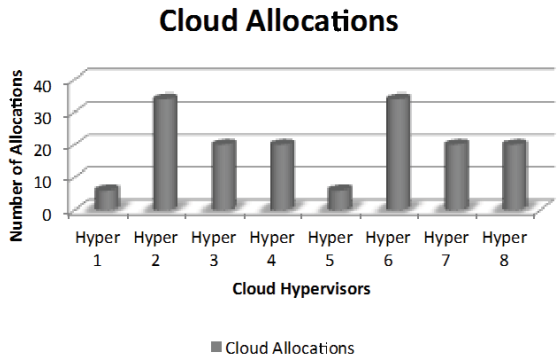

Figure 7: The cloud allocation numbers of SimIC when 160 identical jobs submitted by 16 users (10 jobs per user) –Jobs can be executed from the whole collection of clouds.

Case 3: 320 jobs submitted by 32 users (full SLAs)

This case scenario involves the experimental input of 320 identical jobs submitted by 32 users (10 jobs per user) wherein 8 clouds can offer job execution (SLA matchmaking) for all of the services.

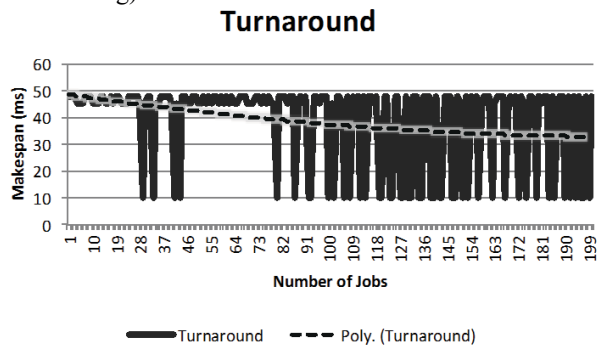

Figure 8: The turnaround and the polynomial trend line of turnaround of SimIC for 320 identical jobs submitted by 32 users (10 jobs per user) – Jobs can be executed from the whole collection of clouds.

This is considered a massive submission wherein the delay among each user 10 set submission is determined to 10 ms.

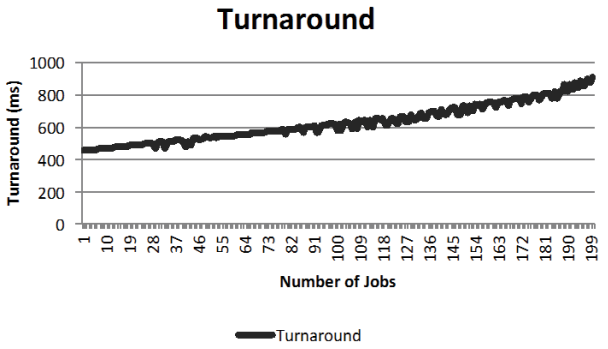

Figure 9: The makespan values of SimIC when 320 identical jobs submitted by 32 users (10 jobs per user) – all clouds offer job execution.

Figure 8 shows the turnaround times and the polynomial trend line of turnaround of SimIC for the aforementioned specification. It is apparent that for high workloads the polynomial trend line of turnaround shows a decrease tendency that is considered a significant improvement of our setting. On the other hand, the turnaround time is increased due to the highly number of users that utilize resources. The ratio factor in this case is the 0.63 that again it is decided as an acceptable rate if we consider that for case 2 (with half number of users) the rate was 0.59. Finally, figure 9 shows the cloud allocation map, wherein clouds 2 and 6 receive the most number of job requests. This is similar to case 2 due to the meta-brokering topology.

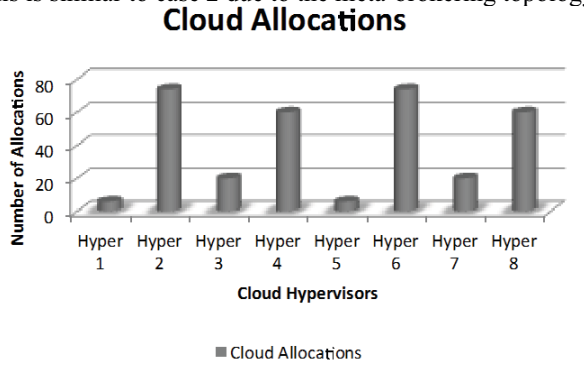

Figure 10: The cloud allocation values of SimIC when 320 identical jobs submitted by 32 users (10 jobs per user) – all clouds offer job execution.

Case 4: 320 jobs submitted by 32 (pattern submission)

This experimental case includes the turnaround, makespan and the polynomial trend line of turnaround times of jobs executed in SimIC. This time the specification is altered wherein 8 clouds can execute certain requests. This involves a more complex setting where SLAs and user submission follows a submission pattern as follows: user 1-8 to cloud 1-8, user 9-16 to cloud 1-8, user 17-24 to cloud 1-8 and user 25-32 to cloud 1-8. To this extend, the system considers a highly number of failures mainly because of the mismatching of SLAs that denotes the non-competency of the cloud to execute certain requests. Figure 11 shows the turnaround, makespan and the polynomial trend line of turnaround of jobs executed in SimIC for the aforementioned specification. The rate factor in this case, (for the number of successful jobs), is determined in 0.56 a values similar to the previous cases.

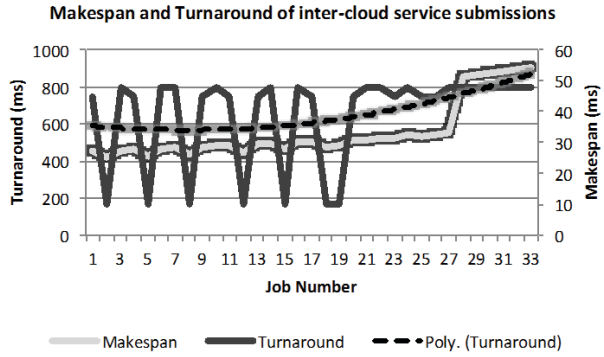

Figure 11: The turnaround, makespan and the polynomial trend line of turnaround of jobs executed in SimIC (16 clouds where 1-8 can execute SLAs, and user submission pattern as follows user 1-8 to cloud 1-8, user 9-16 to cloud 1-8, user 17-24 to cloud 1-8 and user 25-32 to cloud 1-8).

Finally figure 12 shows the job allocation and failures for demonstrating the positioning of jobs within the bucket class. This class keeps a log of non-executed jobs and contains a trigger to release the jobs in the inter-cloud in regular intervals. However, in our experimental case we have considered impractical to release the queue as the system status on SLA matchmaking remains static. This means that jobs will not be executed till the setting extends to include capable for SLA matchmaking clouds.

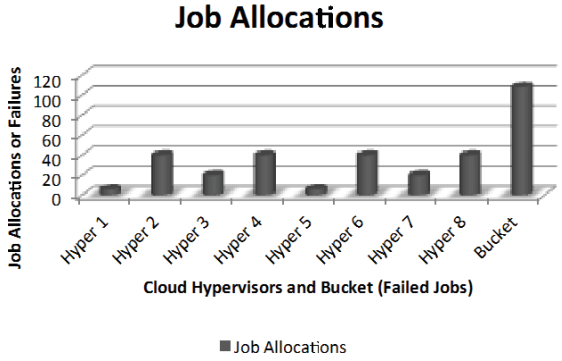

Figure 12: The job allocation numbers and job failures (bucket) of SimIC for 16 clouds where 1-8 can execute SLAs, and user submission pattern.

CONCLUSIONS AND FUTURE RESEARCH STEPS

In this work we compare identical cloud configurations that are implemented in the CloudSim [3] and SimIC [8] respectively in order to show the parallel performance tendency of both toolkits. For achieving experimentation on dynamic and real-time multi-user submissions within an inter-cloud we present an extended experimental analysis of four cases that offers prosperous results. In particular, for high workload submissions with a partial capability of service execution of clouds the system shows decrease values of selected benchmarks. Similarly, for the case of a complete distribution of service submissions the system shows again improvement on turnaround times. In particular for the third case (320 job submissions) the turnaround times are decreased during the simulation time elapses. At last, the makespan values for the first three cases increased due to the latency value that it is added between the service submissions. But then again the ratio that represents the increasing rate remains in similar levels for all the cases.

Lastly, case 4 demonstrates the failures of a system due to the incompetence of the local clouds to execute the requested jobs. It should be mentioned that this is non-related with the dynamic workload management but with an opportunistic SLA requirement regarding specific resources (e.g. specific software). In general this study presents that by using the SimIC [8] a modeler could configure a diversity of inter-clouds in terms of datacenter hosts and software policies wherein desired number of users could send single or multiple requests for computational power (cores, CPU, memory, storage, bandwidth), software resources (measured empirically in clocks per instruction and millions of instructions per second) and duration of VM utilization. The toolkit contains a selection of meta-scheduling inspired characteristics for achieving job dissemination, resource discovery services, dynamic workload management, real time scheduling of jobs in VMs, static and dynamic VM deployment policies and VMs migration cases. In the future we aim of improving the quality of the SimIC and extend the experimental analysis to contain various numbers of cloud and user submissions as well as different meta-brokering . In addition, It should be mentioned that former works for optimizing the message exchanging as presented in [1] among SimIC entities as well as the resource discovery scheme of [6], will add to the overall optimization of the selected metrics.

REFERENCES

[1] Bessis, N., Sotiriadis, S., Pop, F. And Cristea, V. (2012). Optimizing the Energy Efficiency of Message Exchanging for Job Distribution in Interoperable Infrastructures, 4th IEEE International Conference on Intelligent Networking and Collaborative Systems (INCoS-2012), September 19-21, Bucharest, pp. 105-112

[2] Bessis, N., Sotiriadis, S., Xhafa, F., Pop, F. and Cristea, V. (2012). Meta-scheduling Issues in Interoperable HPCs, Grids and Clouds, International Journal of Web and Grid Services, Volume 8, Issue 2, Inderscience, pp. 153-172.

[3] Calheiros, R., N., Ranjan, R., Beloglazov, A., De Rose, C., A., F., and Buyya, R. (2011). CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exper.* 41, 1 (January 2011), p.p.: 23-50.

[4] Núñez, A., Vázquez-Poletti, L., J., Caminero, A. C., Castañé G. G., Carretero, J., and Llorente, M. I., (2012) storage networks. iCanCloud: A Flexible and Scalable Cloud Infrastructure Simulator. Journal of Grid Computing, Volume 10, Number 1. p.p.: 185-209. Springer.

[5] Sotiriadis, S., Bessis, N., Xhafa, F., and Antonopoulos, N. (2012). From Meta-computing to Interoperable Infrastructures: A Review of Meta-schedulers for HPC, Grid and Cloud. In Proceedings of the 2012 IEEE 26th International Conference on Advanced Information Networking and Applications (AINA '12). IEEE Computer Society, Washington, DC, USA, pp. 874-883.

[6] Sotiriadis, S., Bessis, N. And Kuonen, P. (2012). Advancing Inter-cloud Resource Discovery based on Past Service Experiences of Transient Resource Clustering, 3rd International Conference on Emerging Intelligent Data and Web Technologies (EIDWT-2012), September 19-21, Bucharest pp. 38-45

[7] Sotiriadis, S., Bessis, N. and Antonopoulos, N. (2012). Decentralized Meta-brokers for Inter-Cloud: Modeling Brokering Coordinators for Interoperable Resource Management, 9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD'12), May 29-31, Chongqing, May 29 – 31 2012, pp. 2475-2481.

[8] Sotiriadis, S., Bessis, N., Antonopoulos, N. (2012), SimIC: An Inter-Cloud Simulator for large scale resource management, The 27th IEEE International Conference on Advanced Information Networking and Applications (AINA-2013), Barcelona, Spain, March 25-28, 2013

[9] Sotiriadis, S., Bessis, N., Huang, Y., Sant, P. And Maple, C. (2010). Defining Minimum Requirements of Inter-collaborated Nodes by Measuring the Weight of Node Interactions, 4th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS-2010), 15th-18th February, Krakow, pp: 291-298.

[10] Stavrinides, G., L., and Karatza, D., H., (2010). Scheduling multiple task graphs with end-to-end deadlines in distributed real-time systems utilizing imprecise computations. J. Syst. Softw. 83, 6 (June 2010), pp. 1004-1014.