Distributed Interactive Real-time Multimedia Applications: A Sampling and Analysis Framework

George Kousiouris, Fabio Checconi, Alessandro Mazzetti, Zlatko Zlatev, Juri Papay, Thomas Voith, Dimosthenis Kyriazis

***Abstract*—. The advancements in distributed computing have driven the emergence of service-based infrastructures that allow for on-demand provision of IT assets. However, the complexity of characterizing an application's behavior, and as a result the potential offered level of Quality of Service (QoS), introduces a number of challenges in the data collection and analysis process on the Service Providers' side, especially for real time applications. The aforementioned complexity is increased due to additional factors that influence the application's behavior, such as real time scheduling decisions, percentage of a node assigned to the application or application-generated workload. In this paper, we present a framework developed under the IRMOS EU-funded project that enables the sampling and gathering of the necessary dataset in order to analyze an application's behavior. Processing of the resulting dataset is also conducted in order to extract useful conclusions regarding CPU allocation and scheduling decisions effect on the QoS. We demonstrate the operation of the proposed framework and evaluate its performance and effectiveness using an interactive real-time multimedia application, namely a web-based eLearning scenario.**

I. INTRODUCTION

In the light of rising computing paradigms such as Cloud computing ([1]), new value chains are emerging for outsourced hosting and execution of interactive multimedia applications. The latter have strict requirements on quality of service in order to operate effectively (e.g. latency, bandwidth and jitter for video streaming, or processing power for interactive special effects rendering). Actors in the value chain emerge where value can be added, e.g. at the infrastructure level through virtualized storage, networking and compute resources (IaaS), at the application level through offering a specific software tool on a pay-per-use basis (SaaS) and in-between these two levels, comes the possibility of Platform as a Service (PaaS).

Manuscript received April 11, 2010. .This work is partially funded by the European Commission as part of the European IST 7th Framework Program through the project IRMOS under contract number 214777.

G. Kousiouris and D. Kyriazis are with the National Technical University of Athens 9, Heroon Polytechniou Str 15773 Athens, Greece, Tel.+302107722546;e-mail:gkousiou@mail.ntua.gr,dkyr@telecom.ntua.gr.

F. Checconi is with the Scuola Superiore S.Anna, Pisa, Italy. Mail: fabio@gandalf.sssup.it

A. Mazzetti is with Giunti Labs, Italy, Mail: a.mazzetti@giuntilabs.com.

Z. Zlatev and Juri Papay are with the IT Innovation Centre, University of Southampton, UK. Mail: {zdz, jp} @it-innovation.soton.ac.uk

T. Voith is with Alcatel Lucent Deutchland AG, mail: Thomas.Voith@alcatel-lucent.com

For the above value chain to support applications with real-time attributes, careful planning is required, so that neither under-provisioning (likely failure of the application to execute) nor massive over-provisioning (unnecessarily high costs) occur.

Furthermore, extensive use of techniques for incorporating applications with different characteristics in the infrastructure creates a burden with regard to the investigation of the application's behavior. Such techniques may include the use of virtualization, specialized scheduling and sharing of resources between different components. Conclusively, extensive data sets must be collected in an automated way so that conclusions regarding an application's behavior with varying resource allocations may be investigated.

In this paper, a process for gathering extensive datasets from applications inside the EU-funded project IRMOS ([11]) is described. This gathering incorporates state of the art components such as virtual machines (VMs) and real time schedulers, based on a variation of a number of parameters that are relevant to the investigated application and to the hardware configuration of the nodes of execution. Analysis and results regarding the effect of these parameters (such as changing scheduling granularity) to the application QoS levels are presented, in order to let the Service Provider (SP) use the fittest settings for the application under consideration. The resulting data sets will be made available to the general public for reusability purposes.

II. RELATED WORK

Similar work to the one presented in this paper is described in this section. In [2], DynBench is introduced, as a benchmark for distributed real time applications infrastructures. This creates dynamic conditions for the testing of the infrastructures. While promising, this framework is mainly oriented towards investigating the limits of the infrastructure and not towards understanding application behavior with different configurations.

In [3], VSched is presented, an EDF-based scheduling algorithm. In this work, an analysis is conducted on application performance with the scheduler in question, investigating the effect of scheduling decisions and concurrent virtual machines execution. The analysis is very thorough and interesting, however no framework is presented for obtaining the necessary data sets.

In [4], DIANE is presented for Grid-based user level

scheduling with a focus on applications. However these applications are more centered around execution end time and not on real time interactivity.

A very interesting work is presented in [5] , where the users of a virtual machine are given the opportunity to increase through a simple interface their allocated CPU, based on their experience with the application. The cost of the increase is shown, so that the user may decide on the fly. While it is a very promising approach and would eliminate a vast number of issues with regard to application QoS levels, its main drawback is in cases of workflows. Inside a workflow, a degradation in performance may be due to a bottleneck on various nodes executing a part of it. The user will most likely be unaware of the location of the bottleneck, especially in cases of non experts.

Another interesting work with regard to real time scheduling and virtualization appears in [12]. In this case, the schedulability of concurrent virtual machines is investigated, in relation to the application deadlines met. Our work differs from this due to the fact that in this paper one of the major goals is to investigate application behavior with regard to changing scheduler assignments. The same applies for the work presented in [13], which compares different scheduling algorithms. The framework presented here is more application centric but can also be used for comparison purposes of the effect of these schedulers on application performance.

The remainder of the paper is structured as follows. In Section III, the role of the proposed framework for application sampling and analysis inside the IRMOS Framework is presented, along with details regarding the parameters of concern. In Section IV, the testbed used for the automatic collection of data is described, while in Section V the description of the process is presented. Finally, we present an analysis on the created data set, with a focus on the effect of the altered parameters on the application QoS level (Section VI) and conclusions (Section VII).

III. ROLE OF SAMPLING IN IRMOS

The major goal of IRMOS (Interactive Real-Time Applications on Service Oriented Infrastructures) is to enable the utilization of distributed infrastructures such as SOIs for interactive soft real time applications. In order for this to be accomplished, the most significant challenge is to offer guaranteed levels of QoS to the applications running inside the framework. However, these software components may be in many cases proprietary. Acquisition of sufficient information in order to deduce conclusions for their behavior can only be achieved through a macroscopic view. What is more, it is assumed that the applications are not written specifically for operation as IRMOS services, but rather, software components already in general use wrapped up as SaaS applications. As a consequence the actual internal operation of the application will be very difficult to be ascertained and used for the purposes of performance modeling. One way to collect this type of information is through executing the application for a variety of different parameters and determine their effect on the QoS output.. Through this information the IRMOS provider will be able to have an idea regarding what kind of resources should be allocated in order to meet the QoS levels requested by the client. Processing of these data for interpolation may be performed in a variety of ways (analytical modeling, statistical analysis, queueing theory, artificial intelligence etc.) however we consider this part out of scope for this paper.

For real time applications, the basic aim is to provide QoS guarantees. These may be either extremely strict, with no possibility to fall below the specified levels (hard real time constraints) or more relaxed, allowing for a predefined percentage of the QoS output to be above the wanted levels (soft real time constraints) ([8]) . In IRMOS, the second case is considered. So, what is critical, is to have a probabilistic approach that covers the needed levels.

The data sets needed for the creation of these probability distributions are obtained by general experimentation and sampling activities that are described in this work. Application runs can be performed for a series of workloads, with different application setup, on a variety of hardware configurations. More details regarding the modelling approach followed in the project can be found in [9].

A. Sampling Parameters

The parameters for which these sampling tests will be conducted depend on both the hardware on which the application is executed and the application parameters that will be toggled during real life executions.

For the hardware parameters, different CPU percentage assignments can be given with varying granularity. The granularity concerns the time period in which this percentage is assigned and can vary from a few milliseconds to seconds typically. This affects the performance of an application (for the same percentages of CPU allocation) in many ways since different needs must be met in each occasion. For example, for interactive real time cases, the application must be able to be activated in specific time intervals in order to give the user the notion of interactivity. This period may be different from case to case. However frequent task switching in the CPU results in increased overheads for the switching process and the restoration of each task's status. So a trade-off must be achieved between the two cases. On the other hand, on applications such as scientific simulations this effect may be different. These applications are typically time consuming and have no need for interactivity. Thus, the larger this granularity is, the better the application behavior in terms of overall execution time will be, since with reduced task switching, cache utilization will be improved.

Other parameters have to do with the workload produced by the application. Different executions may produce different amount of work for the processor to handle. The effect of these factors must also be investigated in the context of service oriented infrastructures. For example, in a

server based application, this parameter is determined by the number of users that produce requests towards the server. The higher this value is, the more requests are generated and the more strenuous to the resources the application will be.

IV. SAMPLING TEST-BED

The sampling test-bed consists of a number of necessary components in order to alter the aforementioned parameters. These include the Virtual Machine inside which the application resides and is executable on all nodes of the infrastructure. The second necessary component is the real time scheduler used, that allocates the CPU share to the VM process and alters the granularity of this assignment. Finally, the application that is installed inside the VM is needed in addition to an external simulator. The latter is used in order to create application workload. More details regarding the role of each component are given in the following sections.

A. Virtualization Approach

Expanding network connectivity and the growing bulk of data demands for larger infrastructures, which are able to react dynamically on computing, networking and storage needs. The concept, which provides "on demand" services by sharing infrastructure resources and maintaining reliability and scalability, is associated with the term "cloud". On the 32th IETF meeting in 1995 ([10]) the term cloud has been already used for the telecommunication infrastructure – now known as telco cloud – dealing with IP routing over large "shared media" networks. Sharing the computing resources over time (perceived already by John McCarthy in 1961) is experiencing now a renaissance due to the virtualization technologies. Virtualization of computing resources allows running multiple operating systems time-shared on a single computer in so called virtual machines. The independence of the virtual machine from the real hardware allows it to provide the computing as an infrastructure service on demand on any real host with enough free computing power. The virtual machine needs to be light-weight for movement and for instant availability. The three main pillars computing, data storage and networking can be provided as a service on demand as long as there are some guarantees associated with it. The cloud infrastructure service - Infrastructure as a Service (IaaS) means that the infrastructure can be utilized as a service without expertise or control over the technology infrastructure ensuring certain guarantees. The guarantees for computing belong to virtual machines experiencing certain CPU time over the complete service time. This is an inevitable prerequisite for enabling a real-time application inside a virtual machine with certain CPU time guarantees. A real-time capable OS of the virtual machine makes it possible to run real-time tasks inside.

Inside the IRMOS framework, the Kernel-based Virtual Machine tool is used. For each application a VM is created that covers its functional requirements (OS, specific internal tools etc.) and which then can be executed on all nodes of a distributed infrastructure. Through the use of VMs, other parameters may easily be altered such as number of underlying cores used, memory size, CPU model etc.

B. Host Scheduler Description

In order to provide scheduling guarantees to the VMUs, we used a hybrid deadline/priority (HDP) real-time scheduler ([6]) developed within the IRMOS consortium for the Linux kernel. This scheduler provides temporal isolation among multiple possibly complex software components, such as entire VMUs. It uses a variation of the Constant Bandwidth Server (CBS) algorithm, ([7]) based on Earliest Deadline First, for ensuring that each group of processes/threads is scheduled for Q time units (the budget) every interval of P time units (the period). The CBS algorithm has been extended for supporting multi-core (and multi-processor) platforms, achieving a partitioned scheduler where the set of tasks belonging to each group may migrate across the associated CBS scheduler instances across processors, according to the usual load-balancing heuristic of Linux. Furthermore, whenever each (partition of a) reservation is scheduled on each core, the associated tasks are scheduled according to their real-time priorities.

The scheduler exhibits an interface towards user-space applications based on the cgroups framework, which allows for configuration of kernel-level parameters by means of a filesystem-based interface. This interface has been wrapped within a Python API, in order to make the real-time scheduling services accessible from within the IRMOS platform. The parameters that are exposed by the scheduler are the C and D values, where C is the amount of computing time assigned to the VM every D interval.

C. Application Description and Preparation

The application under investigation is an eLearning mobile instant content delivery, in which real-time requirements are combined with service oriented architecture. In this scenario a user can receive on his/her mobile phone some eLearning contents relevant to the position where she is (e.g. walking near to historical monument). It consists of a Tomcat based e-learning application that incorporates a MySQL database (Figure 2). The application is able to receive queries with GPS data from a client, search internally in the database and respond with an elearning object identifier that corresponds to the provided GPS coordinates (Figure 1). It is provided as a war file and installed inside the VM. The real-time requirement is mainly the response time in each request and depends on the number of concurrent users and the size of the downloaded contents.

Furthermore, it must provide a way for the sampling framework to gather the reported data with regard to the values of interest. In the examined application, this transition of information was implemented with two potential ways. The first one was an XML report available through a URL. The sampling framework polled this URL with a given frequency and the XML report was stored and processed afterwards. The second option was for the application to

store on its own the reports inside a MySQL database, from which the sampling framework could retrieve them.

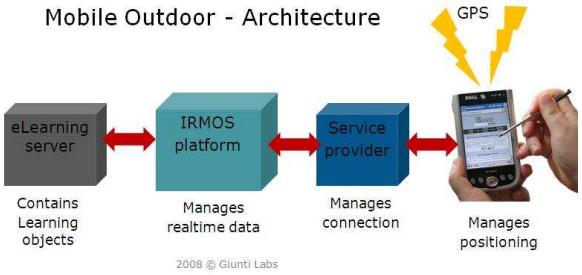

Figure 1

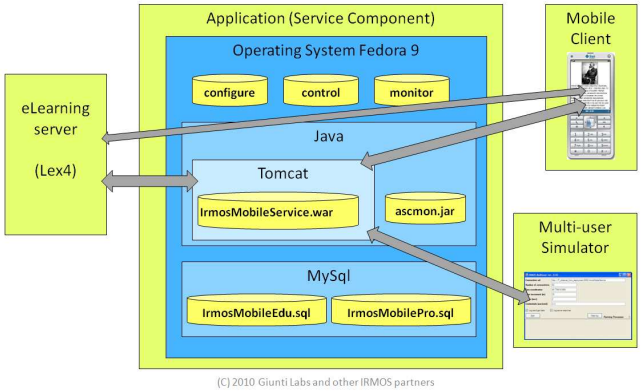

Figure 2: Application Design

D. Application Client Simulation Description s

In order to simulate application parameters, a client simulator is also necessary. This simulator can toggle the number of users performing queries on the server, thus varying the server load. By having different server loads and different hardware configurations we can have an analysis of their effect on the expected QoS output (in this case the response time of the server to the users).

One significant advantage of the test-bed is that the components described are decoupled from one another. This makes it flexible, so that these components (like schedulers, different virtualization tools or applications) can be replaced with different versions, thus making comparisons between them easy.

V. COLLECTION AND PROCESSING FRAMEWORK

In order for the collection of the samples to be conducted as automatically as possible, a number of actions have been implemented. First of all, the application client simulator is started, with a fixed number of users, whose created traffic is simulated. Afterwards a Java-based program resides on the physical host level of the infrastructure. This code is responsible for retrieving the reported monitoring data from the application. This can be done with two ways. The first case is to call the URL provided by the application and described above in order to collect the XML reports produced by the latter. The reports from every sample of one configuration are appended in a single XML file, whose name is indicative of the scheduling parameters used for the execution (C and D). Each sample is taken in a specific period, expressed through a parametric delay inserted between consecutive calls to the URL. This sampling frequency could be adjusted in case of periodic applications in order to obey to the Nyquist-Shannon theorem so that from the samples collected the entire distribution can be created. The second case is through the MySQL database, in which the reports from the application are timestamped and stored. For every configuration the start and end time are saved, and based on this information the application data that were stored during this interval are retrieved.

Furthermore, in the same code, a Java-system interface is implemented in order to be able to change the configuration of the scheduling parameters through the interface script described in Section IV.B. This way, consecutive configurations are tested automatically and their result in the QoS parameter of the application (response time) is recorded. During the time of each configuration, the previously examined retrieval framework takes the necessary measurements. The change in the scheduling parameters is two-fold, it involves the C/D value, which is the percentage of CPU assigned to the VMU (which can be considered in a way as a simulation of different CPU speeds) but also the granularity of D. This granularity is expected to affect application performance, as stated in Section III. Even with the same %CPU assignment, a very large value of D would result in a highly non-responsive service, especially for interactive applications, due to the large interval of deactivation. For other applications with no interactivity, e.g. scientific simulations, a large number of D could prove to be useful, due to reduction in task switches and better cache utilization.

In conclusion, the Java class is responsible for altering the C, D parameters (both ratio and absolute values), for connecting to the application interface (URL available XML reports or MySQL DB), for extracting the reported values, for processing them in order to produce the necessary statistics (in this case mean response time and standard deviation) and for creating the final output. This output is CSV files, that contain matrices that can directly be used by performance estimation methods. These include columns with the different number of users, different C, D parameters and the extracted statistics.

Finally, the number of users in the client simulator is changed and the process is initialized again. Due to the elastic form of the testbed, other parameters may also be investigated easily, such as the memory assignment to the VM, configured at the VM startup. In this particular application memory requirements were not extensive that is why it was decided not to investigate this parameter.

The structure of the sampling framework appears in Figure 3.

In order to extract the necessary information that is needed in the modeling approach followed inside IRMOS as described in Section III, the sampled response times of the eLearning server are gathered for each execution and statistical metrics are extracted. These can be used for the construction of the PDFs of the QoS output in consideration, for use in the next stages of modeling. The basic metrics that

are extracted are the mean value and the standard deviation of the response times.

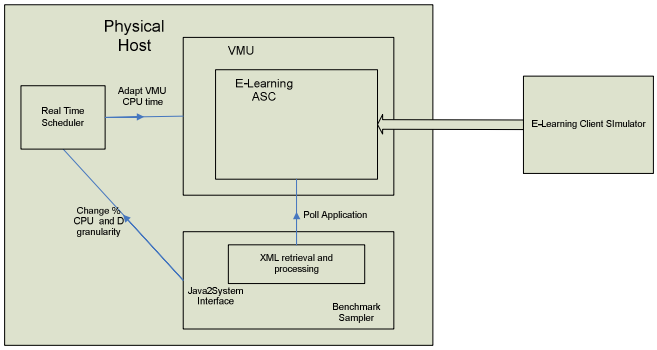

Figure 3: Sampling Framework

VI. RESULTS

In this section, the results from the performed experiments are depicted. The range of values that were altered is:

- Number of Users: 30-150
- C/D (CPU share) : 20-100% with a step of 20
- D: 10000- 560000 (μsec) with a step of 50000

Measurements were taken and the gathered values for each configuration were collected. An average of 800 response times was collected for each different setup, in order to extract their mean and standard deviation values. An indicative set of these measurements is depicted in the following figures, from which useful conclusions can be drawn.

The effect of changing granularity on the deviaton of the response time values can be observed in Figure 4. This is expected since with high values of D, the service has long active and inactive periods. If the requests fall in the active interval, they will be satisfied quickly but if they fall in the inactive one then they will have to wait until this has finished. This effect decreases as allocated CPU share increases, since in these cases the CPU is almost dedicated to the application and whenever a request arrives it is served. The mean response time, as shown in Figure 5, seems not to be affected greatly given that the percentage of CPU assigned is the same.

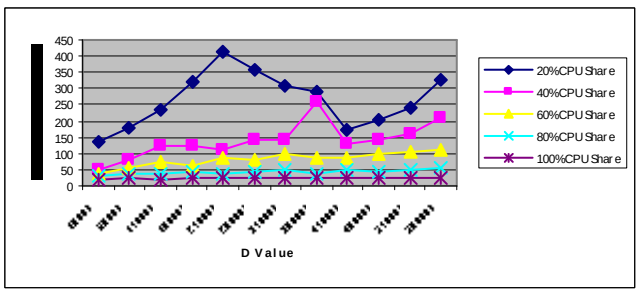

Figure 4: Standard Deviation with regard to changing D for 90 users

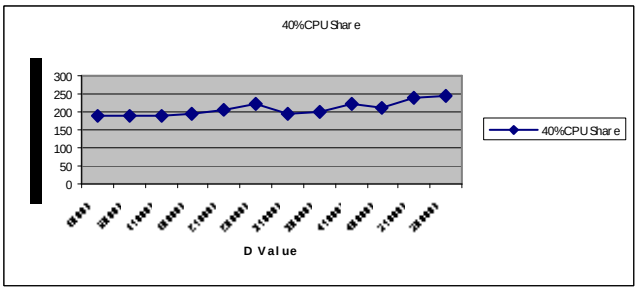

Figure 5: Mean value with regard to changing D for 70 users and 40% CPU share

In Figure 6, the comparison between the collected values of response times is shown for two different numbers of users. The difference especially in the maximum values of the distributions depicts the effect of the application workload in the response times.

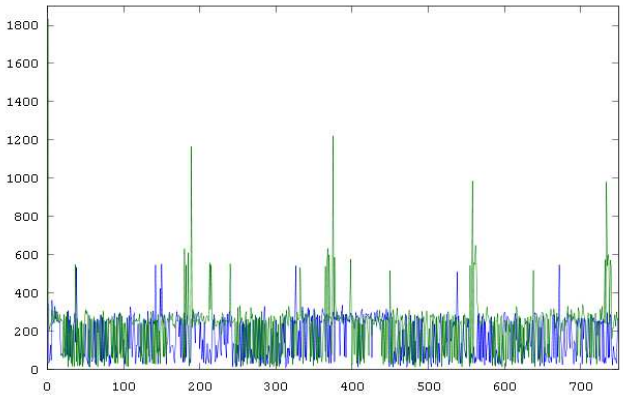

Figure 6: Comparison of different number of users (blue 30, green 50) for the same resources (40% of the CPU) and same D

In Figure 8 all the different configurations are shown for two different numbers of users. In this case, each group of columns (the first high one followed by 4 lower ones) represents one D configuration for different percentages. The high bar is for low utilization and while the utilization increases the response time decreases. In the horizontal axis the different D configurations represent increasing D values.

From these measurements it seems interesting that the fittest granularity (D) selected depends also on the percentage of the CPU assigned to the application. In this occasion, for low percentages of utilization it is best to assign values near the middle of the investigated interval (10000-560000), as is depicted in Figures 7 and 8. For higher percentages of utilization, lower values of D are more beneficial for the response times of the application. Furthermore, from Figure 7 the effect of the increased CPU share allocation to the response time can be observed.

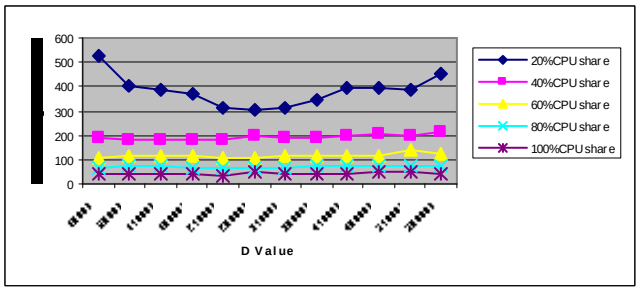

Figure 7: Mean Response Time for different D's and CPU shares for 110 users

The data set that was produced from the process described in this paper and that was the basis for the above figures will be made available to the community, following the initiative for reusable data sets.

VII. SUMMARY

In this paper, a sampling and analysis framework, used within the IRMOS project has been described. The aim of this framework is to easily gather extensive datasets regarding an e-Learning application and its real time requirements, in relation to characteristics such as the number of users of the application and the hardware allocation to it. This framework utilizes state of the art techniques in virtualization and real time scheduling, and the corresponding analysis of the results aids Service Providers in understanding the application's behavior. It is also flexible in order to be used in distributed infrastructures with no need for alterations for the deployment in a variety of nodes. This in turn can lead to enhanced allocation strategies. For the future, one interesting aspect to investigate would be the interference between co-scheduled VMs.

ACKNOWLEDGMENT

This research is partially funded by the European Commission as part of the European IST 7th Framework Program through the project IRMOS under contract number 214777.

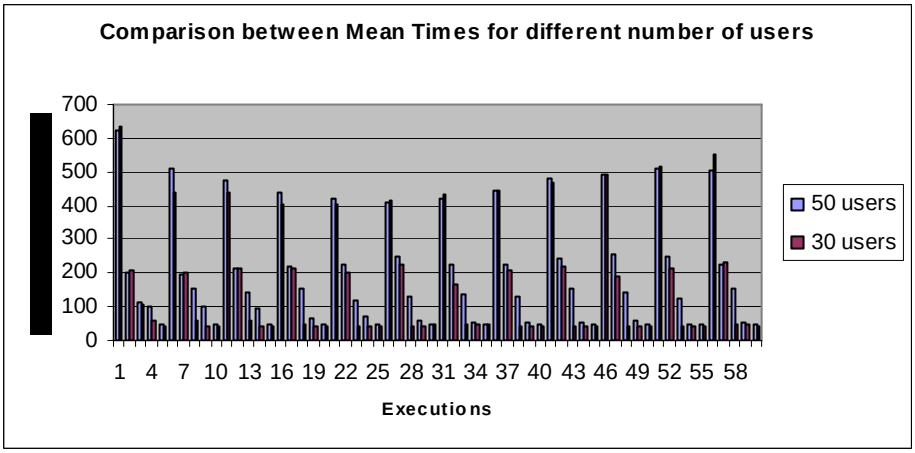

Figure 8

REFERENCES

[1] http://www.cloudcomputing.org/

[2] B. Shirazi, L. Welch, B. Ravindran, C. Cavanaugh, B. Yanamula, R. Brucks, E. Huh. DynBench: A Dynamic Benchmark Suite for Distributed Real-Time Systems. IPDPS Workshop on Embedded HPC Systems and Applications. S. Juan, Puerto Rico, 1999

[3] Lin, B., Dinda, P.: Vsched: Mixing batch and interactive virtual machines using periodic real-time scheduling. In: Proc. of the IEEE/ACM Conf. on Supercomputing, p. 8, Nov. 2005

[4] Germain, C., Loomis, C., Mo´scicki, J.T., Texier, R.: Scheduling for responsive Grids. J. Grid Computing 6(1), 15–27 (2008)

[5] Lin, B. and Dinda, P. A. 2006. Towards Scheduling Virtual Machines Based On Direct User Input. In Proceedings of the 2nd international Workshop on Virtualization Technology in Distributed Computing (November 17 - 17, 2006). Virtualization Technology in Distributed Computing. IEEE Computer Society, Washington, DC, 6. DOI= http://dx.doi.org/10.1109/VTDC.2006.15

[6] Fabio Checconi, Tommaso Cucinotta, Dario Faggioli, Giuseppe Lipari, "Hierarchical Multiprocessor CPU Reservations for the Linux Kernel," in Proceedings of the 5th International Workshop on Operating Systems Platforms for Embedded Real-Time Applications (OSPERT 2009), Dublin, Ireland, June 2009

[7] Luca Abeni and Giorgio Buttazzo, "Integrating Multimedia Applications in Hard Real-Time Systems," in Proc. IEEE Real-Time Systems Symposium, Madrid, Spain, 1998

[8] .Liu, C. L. and Layland, J. W. 1973. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. J. ACM 20, 1 (Jan. 1973), 46-61. DOI= http://doi.acm.org/10.1145/321738.321743

[9] Matthew Addis, Zlatko Zlatev, Bill Mitchell, Mike Boniface, Modelling Interactive Real-time Applications on Service Oriented Infrastructures, Proceedings of 2009 NEM Summit, ISBN 978-3-00-028953-8

[10] http://www.ietf.org/proceedings/32/charters/rolc-charter.html

[11] http://www.irmosproject.eu/

[12] Cucinotta, T., Anastasi, G., Abeni, L.: Real-time virtual machines. In: Proceedings of the 29th IEEE Real-Time System Symposium (RTSS 2008) – Work in Progress Session, Barcelona (December 2008)

[13] B. Brandenburg and J. Anderson. A comparison of the M-PCP, D-PCP, and FMLP on LITMUSRT. In Proc. of the 12th International Conference On Principles Of Distributed Systems, pp. 105–124, 2008.