**Noname manuscript No.**
(will be inserted by the editor)

# Virtualised e-Learning on the IRMOS Real-time Cloud

**Tommaso Cucinotta · Fabio Checconi · George Kousiouris · Kleopatra Konstanteli · Spyridon Gogouvitis · Dimosthenis Kyriazis · Theodora Varvarigou · Alessandro Mazzetti · Zlatko Zlatev · Juri Papay · Michael Boniface · Sören Berger · Dominik Lamp · Thomas Voith · Manuel Stein**

the date of receipt and acceptance should be inserted later

**Abstract** This paper presents the real-time virtualized Cloud infrastructure that was developed in the context of the IRMOS European Project. The paper shows how different concepts, such as real-time scheduling, QoS-aware network protocols, and methodologies for stochastic modelling and runtime provisioning, were practically combined to provide strong performance guarantees to soft real-time interactive applications in a virtualized environment.The efficiency of the IRMOS Cloud is demonstrated by two real interactive e-Learning applications, an e-Learning mobile content delivery application and a virtual world e-Learning application.

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7 under grant agreement n.214777 IRMOS – Interactive Realtime Multimedia Applications on Service Oriented Infrastructures.

Tommaso Cucinotta · Fabio Checconi
Real-Time Systems Laboratory, Scuola Superiore Sant'Anna, Pisa, Italy
E-mail: {t.cucinotta,f.checconi}@sssup.it

George Kousiouris · Kleopatra Konstanteli · Spyridon Gogouvitis · Dimosthenis Kyriazis · Theodora Varvarigou
National Technical University of Athens, Greece
E-mail: {gkousiou,kkonst,spyrosg,dimos,dora}@mail.ntua.gr

Alessandro Mazzeti
eXact learning solutions S.p.A., Sestri Levante, Italy
E-mail: mazzetti@planasia.eu

Zlatko Zlatev · Juri Papay · Michael Boniface
University of Southampton, IT Innovation Centre, Southampton, UK
E-mail: {zdz,jp,mjb}@it-innovation.soton.ac.uk

Sören Berger · Dominik Lamp
University of Stuttgart, Germany
E-mail: {soeren.berger,dominik.lamp}@rus.uni-stuttgart.de

Thomas Voith · Manuel Stein
Alcatel Lucent, Stuttgart, Germany
E-mail: {thomas.voith,manuel.stein}@alcatel-lucent.com

**Keywords** Real-time scheduling · virtualised infrastructures · stochastic modelling · e-Learning.

## 1 Introduction

A current trend in the software engineering industry is to rely increasingly on the distributed computing paradigm, which is becoming mainstream and ubiquitous, especially due to the high market penetration of low-cost high-speed Internet connectivity. To this direction, more and more applications are developed in a distributed fashion to be hosted on dedicated infrastructures, such as the Clouds, that can be easily accessed by remote users, whether they are using local workstations, laptops, palmtop devices or mobile phones.

According to the Cloud computing model, distributed applications are developed by Software-as-a-Service (SaaS) providers to be hosted in Cloud infrastructures. During this process, the SaaS provider may use tools offered by Platform-as-a-Service (PaaS) providers that facilitate the design, development, and testing of their applications in a virtualized environment. The Infrastructure-as-a-Service (IaaS) providers use virtualization techniques to deploy these applications inside their Cloud infrastructure. By using such techniques, the IaaS providers are able to achieve a better server consolidation level by deploying multiple virtual machines (VMs) with different Operating Systems (and services therein) over the same physical resources. Furthermore, the IaaS providers utilize virtualization techniques at network level, such as live migration, to seamlessly migrate the VMs from one physical host to another one, for maintenance, fault-tolerance or load-balancing reasons.

However, as the number of VMs deployed over the same physical resources (e.g., links and CPUs) increases, the level of performance experienced by each VM becomes unstable. Indeed it depends heavily on the overall workload imposed by the other VMs competing for the shared resources, as has been observed, for example, in Amazon EC2 [10]. Therefore for a virtualised Cloud environment to provide proper accommodation to this increasing number of soft real-time distributed applications, proper CPU and network scheduling technologies, coupled with proper performance modelling and runtime provisioning techniques are needed.

In this paper we present the way these techniques have been combined into one virtualised Cloud Computing service-oriented infrastructure that has been developed in the context of the IRMOS European Project[1], and we show how these concepts have been practically applied to provide strong performance guarantees to each individual real-time interactive application that is hosted on the virtualized resources of the IRMOS Cloud. This is realised by allowing applications to be deployed in the form of a Virtual Service Network (VSN), a graph whose vertices represent Virtual Machine Units (VMUs) that encapsulate the different application components, and whose edges represent the network connections among them. In order for the VSN representation of the application to comply with real-time constraints as a whole, each VSN element is associated with precise requirements in terms of the computing resources and networking resources for each node and link respectably. These requirements are fulfilled thanks to the soft real-time scheduling mechanism and the QoS-aware network protocol in place. However, the accuracy of the estimations for the required computing and networking resources for each VMU may affect the performance of the latter.

To this end, the values of the performance parameters are obtained after a proper benchmarking and monitoring process of the service components on the IRMOS virtualized resources that allows for an accurate estimation of their behavior during the performance modelling of the application. However, the lack of information (e.g. source code) across the entities makes performance analysis a difficult task. To overcome this problem, ANNs were used to effectively model the application's performance when their internal structure is unknown.

The IRMOS PaaS layer consists of tools that facilitate the described above benchmarking, monitoring and performance modelling processes. Furthermore, in order to address cases of bad performance modelling, the IRMOS PaaS continuously collects and evaluates monitoring data at run-time to identify possible deviations between the agreed and achieved QoS levels, the source of the problem, and possible corrective actions, such as adjusting the allocated resources share, and reconfiguration of the running application. Apart from these corrective actions, the IRMOS PaaS also enable the SaaS provider to define a range of application-driven events that may automatically trigger certain actions by the IRMOS PaaS during the run-time of the application, such as automatic renegotiation/reconfiguration when certain usage terms are met.

This paper is organised as follows. Relevant related work is given in Section 2. Section 3 gives an overview of the way IRMOS achieves performance isolation by combining temporal isolation techniques at computing and networking level with proper performance modelling. A detailed description of the IRMOS PaaS is given in Section 4, whereas Section 5 presents in detail the two real e-Learning applications, a Mobile e-Learning application and a Virtual World e-Learning application, that were integrated in the IRMOS platform and were used to evaluate its performance. Section 6 presents the methods that are used for estimating the performance behavior of the applications to be hosted in IRMOS. In Section 7, the results gathered from a large set of experiments using the Mobile e-Learning application on the IRMOS platform under realistic settings are presented. A use case based on the Virtual World e-Learning application that highlights the runtime QoS provisioning capabilities of the platform is presented in Section 8. Finally, conclusions are drawn in Section 9.

## 2 Related Work

In this section, related works that have appeared in the research literature are briefly summarised.

The work by Lin and Dinda in [21] presents various similarities with the one presented in this paper. First, an EDF-based scheduling algorithm [23] for Linux is used on the host to schedule Virtual Machines (VMs). Furthermore, an analysis is conducted on the application performance, investigating the effects of scheduling decisions and concurrent virtual machines execution. The analysis is very thorough and interesting, however the major limitation of the work resides in the way low-level scheduling is achieved. In fact, the authors make use of a scheduler built into a proper user-space process (VSched), which exploits POSIX real-time priorities in order to achieve an EDF-based scheduling of VMs, and SIGSTOP/SIGCONT signals for realising optionally hard resource reservations. Such an approach presents high scheduling overheads due to the forcibly increased number of context switches, whilst our scheduler [5] is directly built into the kernel and

[1] More information is available at: `http://www.irmosproject.eu`.

does not introduce any additional context switch. Furthermore, VSched cannot properly react to those situations in which a VM blocks or unblocks, e.g. as due to I/O operations, something that is needed to guarantee a proper level of temporal isolation. In order to address this issue our scheduler uses the CBS algorithm [1].

Another very interesting work by the same authors is [22], where the users of a virtual machine are given the opportunity to adapt the allocated CPU through a simple interface, 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 the case of workflow applications. The degrading performance of a workflow application may be due to a bottleneck on various nodes executing a part of it. The users will most likely be unaware of the location of the bottleneck, especially if they are non-experts. Instead, the work by Nathuji et al. [26] focuses on automatic on-line adaptation of the CPU allocation in order to keep a stable performance of VMs. However, the framework does not treat a VM as a "black-box", and needs application-specific metrics to run the necessary QoS control loop, going beyond the common IaaS business model.

Gupta et al. investigated on the performance isolation of virtual machines [13], focusing on the exploitation of various scheduling policies available in the Xen hypervisor [6]. Furthermore, Dunlap proposed [9] various enhancements to the Xen credit scheduler in order to address various issues related to the temporal isolation and fairness among the CPU share dedicated to each VM. Instead, the IRMOS real-time scheduler is based on the KVM[2] hypervisor. Having already demonstrated the way temporal isolation of compute and network intensive VMs can be achieved using the KVM hypervisor in our previous work [7][8], the scheduling method proposed in this paper also addresses the modelling issues that arise during the deployment of an e-Learning application under proper QoS guarantees.

Shirazi et al. [28] proposed DynBench, a method for benchmarking infrastructures that support distributed real time applications under dynamic conditions. While promising, this framework is mainly oriented towards investigating the limits of the infrastructure and not towards understanding the application's behaviour in relationship to different scheduler configurations.

In [11], Germain et al. present DIANE for Grid-based user level scheduling. However, the focus is on controlling the execution end time of long processing applications, and not on real time interactive ones as done in this paper. The problem of optimum allocation of workflows of virtualised services on a set of physical resources under a stochastic approach has been investigated in [16], in the context of soft real-time interactive applications.

In terms of application performance modelling in distributed infrastructures a number of interesting works exist. A code analysing process that allows for the simulation of system performance is described in [14]. It models the application by a parameter-driven Conditional Data Flow Graph (CDFG) and the hardware (HW) architecture by a configurable HW graph. The execution cost of each task block in the application's CDFG is modelled by user-configurable parameters, which allows for highly flexible performance estimation. The CDFG of the application and the HW graph are used to perform a low-level simulation of the HW activities. While very promising, it needs the source code in order to provide the CDFG. In our work, we deal with VMs as black boxes, what allows for the deployment of applications where the source code is not available for confidentiality purposes.

Another interesting work is presented by Lee et al. in [20]. The application, whose performance must be measured, is run under a strict reservation of resources in order to determine if the given set of reservation parameters satisfies the time constraints for execution. If this is not the case, then these parameters are altered accordingly. If there is a positive surplus, the resources are decreased and if it is negative they are increased until a satisfying security margin is reached. While assuring high utilisation rates, the main disadvantage of this methodology is that this must be performed for every individual execution with the specific SLA parameters before the actual deployment.

Bekner et al. introduce the Vienna Grid Environment (VGE) [3], a framework for incorporating QoS in Grid applications. It uses a performance model to estimate the response time and a pricing model for determining the price of a job execution. In order to determine whether the client's QoS constraints can be fulfilled, for each QoS parameter a corresponding model has to be in place. However, VGE does not prescribe the actual nature of performance models. Instead VGE uses an abstract interface for the performance models, and it is assumed that these models become available through analytical modelling or historical data analysis. Although analytical performance modelling in general requires a thorough knowledge of the application, the method proposed in this paper allows for those parts of the application that are not visible to the external world to be handled as "black boxes".

[2] More information at: http://www.linux-kvm.org

Other works exist that address QoS assurance via performanc eprediction [18] and control via service selection [19]. While numerous promising solutions exist to the problem of performance analysis of VMs in presence of real-time scheduling, these are either not focused on critical parameters that are necessary for running real time applications on SOIs, or they lack of a proper low-level real-time scheduling infrastructure, which is needed for supporting temporal isolation among concurrently running VMs.

## 3 Performance Isolation – The IRMOS/ISONI Way

One of the core components which is being developed in IRMOS is the Intelligent (virtualised) Service-Oriented Networking Infrastructure (ISONI) [29]. It acts as a Cloud Computing[3] IaaS provider for the IRMOS framework and manages a set of physical computing, networking and storage resources available in form of multiple nodes/sites within a provider domain (see Fig. 1).

ISONI provides those virtualised resources over which IRMOS applications are deployed. One of the key innovations introduced by ISONI is its capability to ensure guaranteed levels of resource allocation for each individual application instance hosted within the ISONI domain.

This is realised by allowing applications to be deployed in form of a Virtual Service Network (VSN). This is a graph whose vertices represent individual Service Components (SCs) of an application which may be deployed in form of Virtual Machine Units (VMUs), and whose edges represent communications – the virtual links (VLs) – among them.

In order for the system represented by a VSN to comply with real-time constraints as a whole, QoS needs to be supported for all the involved resources, particularly for network links, computing hosts and storage resources. To this purpose, VSN elements are associated with precise resource requirements, e.g., in terms of the required computing performance (e.g. working memory, speed, scheduling) for each node and the required networking performance (e.g. bandwidth, latency, jitter) for each link. These requirements are fulfilled thanks to the allocation and admission control logic pursued by ISONI for instantiating VMs within the managed set of available physical resources, and to the low-level mechanisms shortly described in what follows (a comprehensive ISONI overview is out of the scope of this paper and can be found in [29]).

[3] More information at: http://www.cloudcomputing.org/.

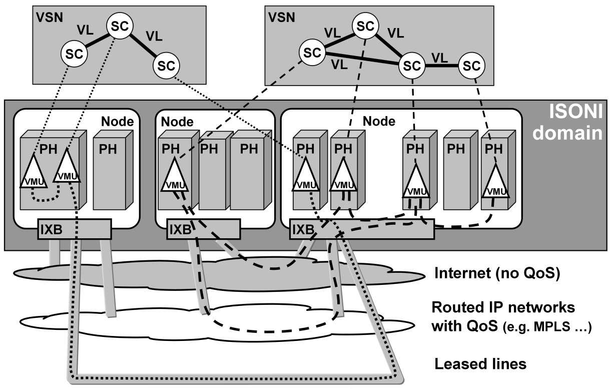


**Fig. 1** Deployment of Service Components (SCs) within Virtual Machine Units (VMUs) over IRMOS/ISONI.

### 3.1 Isolation of Computing

In order to provide scheduling guarantees to individual VMs scheduled on the same system, processor and core, IRMOS incorporates a hybrid deadline/priority (HDP) real-time scheduler [5] developed within the IRMOS consortium for the Linux kernel. This scheduler provides temporal isolation among multiple possibly complex software components, such as entire VMs (with the KVM hypervisor [4], a VM is seen as a process). It uses a variation of the Constant Bandwidth Server (CBS) algorithm [1], based on Earliest Deadline First (EDF), 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 running on different CPUs, according to the usual multi-processor real-time priority-based scheduling in Linux.

The scheduler exhibits an interface towards user-space applications based on the cgroups [24] framework, which allows for configuration of kernel-level parameters by means of a file system-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 budget Q and the period P, as explained above.

### 3.2 Isolation of Networking

Traffic isolation of independent VSNs within ISONI is achieved by provisioning each VSN deployment with an individual virtual address space and by policing the

[4] More information at: http://www.linux-kvm.org.

network traffic of each deployed virtual link. The automated deployment of policed virtual link overlays prevents unwanted crosstalk among services sharing physical network links and prohibits intrusion attempts from unnamed endpoints. The traffic policing in place ensures that the network traffic traversing the same network elements causes no overload which would lead to an unduly, uncontrolled growth of loss rate, delay and jitter for the network connections of other VSNs. A gapless policing ensures that the network multiplex stages always get a controlled load of traffic. Therefore, bandwidth policing is an essential building block of QoS assurance for the individual virtual links. It is important to highlight that ISONI allows for the specification of the networking requirements in terms of common and technology-neutral traffic characterisation parameters, such as the needed guaranteed average and peak bandwidth, latency and jitter.

Depending on the specified networking requirements, an adequate transport network is chosen in order to meet the application requirements (see Fig. 1). By default, non-critical traffic without estimable performance requirements can be deployed on Internet transit that does not provide guarantees on the delivery and performance of traffic. On the other hand, soft real-time distributed applications that rely on capacity guarantees can be mapped onto network resources for which ISONI controls the network utilization. The stronger the soft real-time application requirements on delay and jitter become for a virtual link, the shorter is the allowable network distance of the transport resource, which leaves either the site-local network or the use of leased lines as suitable transport resource. Since there exist transport network resources with classification, reservation and other technology-individual mechanisms to enforce QoS of traffic, an ISONI transport network adaptation layer abstracts from transport network technology-individual QoS mechanism like Diffserv [4], Intserv [30][31] and MPLS [27] (see Fig.1).

### 3.3 Performance Modelling

One of the key steps in deploying applications with precise real-time or generally QoS guarantees within IRMOS is the one of building a performance model of the application behavior. This means that, given application-specific configurable parameters (e.g., number of users, resolution of multimedia contents, etc.), and given possible performance levels that one may want to achieve, it should be possible to determine what allocation is needed on the physical resources in order to accomplish that. This is a core information needed by the SaaS provider in order to establish an accurate pricing policy for the customer(s).

Application performance in the cloud depends on many complex factors such as application workload, conditions of the network paths between the user(s) and the server(s) and the computing workload of the physical host(s). Computing workload factors are especially significant in multi-tenant clouds where single hosts are used to service multiple applications. However, as already described, the IRMOS real-time scheduler provides temporal protection, and thus the level of interference among different VMs sharing the same resources becomes negligible. Therefore the applications can be easily benchmarked using the facilities of the IRMOS PaaS (see Section 4) and their performance can be modelled as a pure function of application-specific parameters and allocated resources, independently of other applications that may be hosted in the IRMOS cloud.

## 4 The IRMOS PaaS

The IRMOS PaaS, namely the IRMOS Framework Services (FS), acts as a mediator between the SaaS and the ISONI provider. It consists of a family of services and tools that can be used by the SaaS provider for benchmarking, modelling and managing applications on the service-oriented virtualized environment that is offered by the ISONI provider. We can make the following categorization of the IRMOS FS:

– Engineering services: These include services for performing application modelling, benchmarking and performance modelling which are needed by the SaaS provider to make an application ready to be incorporated and hosted in the underlying ISONI cloud.
– Management services: These include services that support the management of the full life-cycle of the service-oriented application. Therefore, the FS include services for discovery, negotiation, reservation, enactment, monitoring and event handling during the execution of an application inside the ISONI virtualized environment.

Using the services above, a description of the VSN can be created and used by the FS for the purpose of negotiating resources with the ISONI provider. This description includes the required resources for the services that make up the application and their interconnections, including instances of some key FS responsible for real-time critical tasks, such as enactment and monitoring. Upon agreement, the ISONI provider is responsible for delivering a “just-in-time” deployment of the distributed application on the reserved physical resources according to the specified QoS requirements. From then on, the end user is able to use the application without any knowledge of the underlying Cloud infrastructure.

### 4.1 Monitoring and Benchmarking

By using the FS tools, the SaaS provider is able to model behavioral aspects of its application, which aid in the performance modelling analysis for deriving critical information about the fluctuation of resource utilization. To this direction, a number of important parameters when benchmarking the application needs to be defined. These include the critical inputs of the services that influence the produced workload, as well as parameters whose values can be chosen by the customer depending on its needs, e.g. the maximum number of clients that are allowed to connect to an e-Learning application.

Another factor is the QoS outputs of the services that comprise the application. These are collected and stored during benchmark runs by the FS monitoring system. The latter is a suite of components for monitoring, evaluating possible performance anomalies, and visualizing the performance of the application inside the VSN, along with the levels of the resources that are offered to the application by the ISONI provider. These components acquire this information through an extensible interface that is used to execute application-specific external programs to collect high-level data about the performance of the applications. Furthermore, the FS monitoring system establishes communication with ISONI's monitoring system for retrieving low-level information about computing and networking resources of the VSN. More information about the design and implementation of the FS monitoring system can be found in [15].

The monitoring information is aggregated and stored in the PaaS infrastructure, and is used both for benchmarking and performance modelling of an application to be deployed in the ISONI. Through this exchange of information, the FS are able to benchmark the application and generate the rules needed for the mapping of high level parameters (used by the SaaS provider for describing the QoS levels), to low-level parameters (used by the ISONI provider for the deployment of the VSN). These rules correlate the configurable workload inputs, the QoS outputs and the hardware assignments of the benchmark runs, and are used in conjunction with analytical approaches to generate an overall end-to-end performance model for the application. This model enables various trade-offs and configurations in order to select the most suitable combination of resources at the lowest cost, while guaranteeing the QoS levels needed by the SaaS (see Section 3.3).

### 4.2 Run-time QoS Provisioning

In a dynamic environment such as the Cloud, the number of the resources and their availability can change frequently. For example, new resources (compute servers, file servers, etc) may be added, old ones may be removed or become temporarily unavailable for maintenance purposes, etc. Furthermore, there is always the chance of bad performance modelling that may result in deviations between the agreed and achieved QoS levels at application run-time.

For these reasons, there is a need of continuous observation of the resources and the application's performance for purposes such as fixing problems and tracking down their origin. Because the business entities behind the SaaS, PaaS and IaaS, often have conflicting interests and different mechanisms and levels of fault tolerance, assessment on the source of the problem can be a significant undertaking. To this end, monitoring data are collected during run-time and are being evaluated by the FS monitoring system to identify possible deviations, the source of the problem, and corrective actions that could be undertaken. Typical corrective actions include adjustments of the resource allocation to the application resource utilization, reconfiguration of a running steering service of the application, notifying the end user and the SaaS provider when certain events that affect the availability of the application occur, among others [12].

Apart from these corrective actions, the FS also enable the SaaS provider to define a range of application-driven events that may automatically trigger certain actions by the FS during the run-time of the application. For example, it may be desirable for the SaaS provider that the FS automatically launch a renegotiation/reconfiguration process when certain usage terms are reached. Such scenarios can be found in a variety of soft real-time applications. In the specific case of a virtual world application such as the one described in Section 5.2, the number of users could be modelled into a trigger for launching automatic renegotiation and reconfiguration of the application by the FS (see Section 8).

## 5 E-Learning Applications

The IRMOS platform was especially designed to provide strong performance guarantees to distributed, interactive, soft real-time applications. In order to demon-

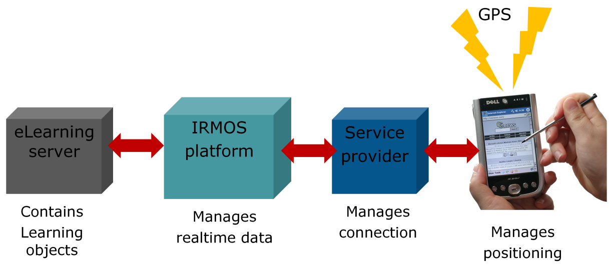


**Fig. 2** Components of the Mobile Outdoor e-Learning application.

strate the IRMOS platform functionality, the two interactive applications listed below were selected because of the soft real-time requirements that they impose on the Cloud infrastructure hosting them:

- Mobile Outdoor e-Learning, which is a mobile application, applied to outdoor usage, and
- Virtual World e-Learning, an ubiquitous application for enabling cooperative e-Learning.

Both applications focus on e-Learning and were offered as SaaS, and they were executed on the IRMOS IaaS layer (ISONI) after being modelled using the tools of the IRMOS PaaS layer (IRMOS FS).

## 5.1 Mobile Outdoor e-Learning

We focus on an e-Learning mobile instant content delivery application, developed to take advantage of a service-oriented architecture paradigm, in which real-time requirements play an important role. In this scenario a user can receive on a mobile phone some e-Learning contents relevant to the current geographical position (e.g., when approaching historical monuments). It consists of a Tomcat[5]-based e-Learning server that exploits a MySQL database for content management (see Fig. 3). The application is able to receive queries with GPS data from multiple clients, search the database and respond with e-Learning contents corresponding to the provided GPS coordinates (see Fig. 2). The application server is provided as a Web Application Archive (war) file, installed on Tomcat, and made available as a Virtual Machine image within the IRMOS infrastructure. Using ISONI, each instance of the application can be assigned precise computing and networking resources to ensure that the high-level requirements defined within an application provider's Service-Level Agreement (SLA) can be met with an agreed level of reliability.

The timing requirements of the application are mainly related to the response times of individual requests sub-

[5] More information at: http://tomcat.apache.org

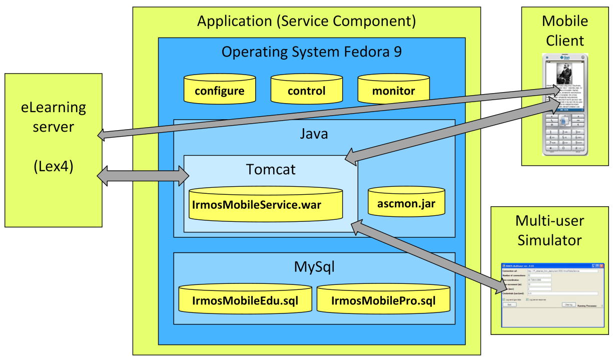


**Fig. 3** Software architecture of the Mobile Outdoor e-Learning application.

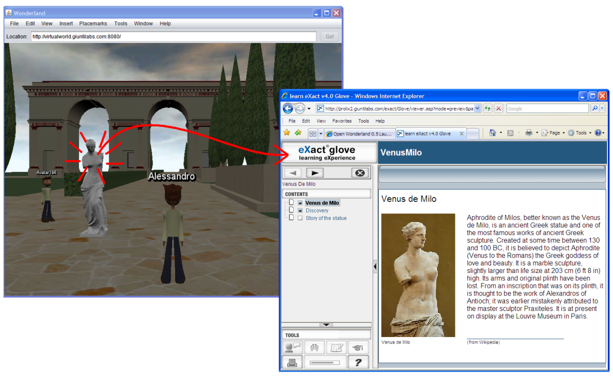
mitted by the multitude of users.The response times are gathered by 'ascmon.jar'and communicated to the FS by the 'monitor' script (see Fig. 3). As discussed in Section 3.3, thanks to the deployment within IRMOS, these response times depend merely on high-level application-specific parameters, i.e., on the number of concurrent users querying the same e-Learning instance and the size of the downloaded contents.

### *5.1.1 Application Client Simulation Description*

In order to investigate application performance, we developed a multi-user client simulator. This is capable of simulating the random movements of a certain number of users walking around given GPS coordinates. Then, the simulator mimics the behaviour of the real mobile client associated with the application: whenever the monitored GPS coordinates move sufficiently away from the position of the last queried content, a new request is submitted to the server with the new user position. The number of users and a few parameters governing how each emulated user exactly moves (e.g., the user speed) determine the exact pattern of requests submitted to the server, thus strongly impacting on the imposed server load.

## 5.2 Virtual World e-Learning

In the Virtual World e-Learning application we consider remote users interacting through avatars in a virtual world system. The Virtual World reproduces a museum, where works of art are associated to e-Learning lessons (see Fig. 4). The avatars can move independently from each other and can communicate through a chat or a voice line. When an avatar comes close to a work of art, he or she can invite the other avatars, through chat, to download the corresponding lesson . Each user can play the lesson by themselves, while communicating via chat.



Venus de Milo

Aphrodite of Milos, better known as the Venus de Milo, is an ancient Greek statue and one of the most famous works of ancient Greek sculpture. Created at some time between 130 and 100 BC, it is believed to depict Aphrodite (Venus to the Romans) the Greek goddess of love and beauty. It is a marble sculpture, slightly larger than life size at 203 cm (6 ft 8 in) high. Its arms and original plinth have been lost. From an inscription that was on its plinth, it is thought to be the work of Alexandros of Antioch; it was earlier mistakenly attributed to the master sculptor Praxiteles. It is at present on display at the Louvre Museum in Paris.

Venus de Milo

(from Wikipedia)

**Fig. 4** The Virtual World e-Learning application's GUI

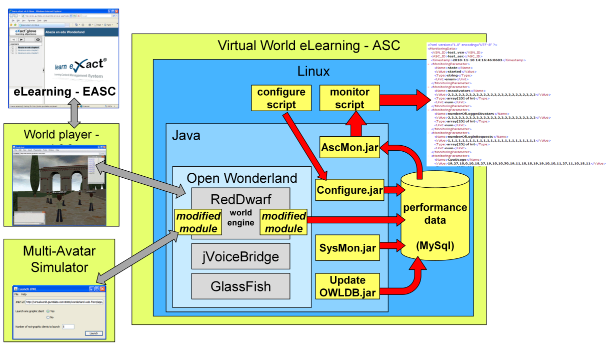


**Fig. 5** Software architecture of the Virtual World e-Learning application.

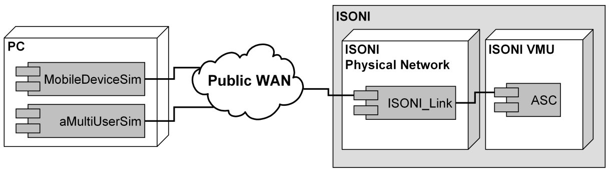


**Fig. 6** Modelled elements of the e-Learning application.

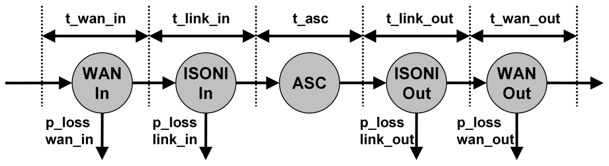


**Fig. 7** Stochastic performance model: t_wan_in and t_wan_out are modelled as exponential distributions, the other delays as Erlang ones.

This collaborative virtual world application is composed by two main elements: the application service component (ASC), dealing with user's communication, and the application client component (ACC or World Player), dealing with the graphic rendering. The ASC builds on top of an Open Wonderland server with modified modules for IRMOS adaptation and a MySQL database for managing performance data. A detailed architecture of the Virtual world application is shown in Fig. 5, in which the parts that have been created or adapted to the IRMOS specifications, such as modules for configuration and monitoring, are marked with yellow color. It should also be noted that the Virtual World application provides a multi-avatar simulator for benchmarking purposes. This tool allows the simulation of a large number of users, which are moving continuously to generate the maximum amount of traffic between the clients and the server.

Compared to the Mobile Outdoor e-Learning application, the Virtual World application is more real-time intensive and thus requires a greater number of high level performance parameters such as the avatar speed, the chatting quality, etc. The Virtual World ASC must satisfy real-time constraints (e.g. response time and processing time), in order to support realistically fluid movement of the avatars. The performance of this component strongly depends on the number of connected users, because the quantity of dispatched messages increases polynomially with the number of avatars.

## 6 Performance Estimation

In order to estimate what QoS level can be achieved using different resource configurations, we use performance modelling techniques. Many times, the application internal software structure may be too complex to be modelled. Or, it may be unknown because developers are reluctant to share detailed information about their application internals, for confidentiality purposes. In other cases, the use of external libraries or components whose internals are unknown makes it impossible to build an exact model. So, from a modelling point of view, it is critical to be able to identify the expected QoS output using a black-box approach.

Therefore, we use a combination of a stochastic model for predicting statistics over the expected run-time networking performance [2], and an Artificial Neural Network (ANN) for identifying the dependency of a component QoS from factors like application-level parameters (e.g., number of clients) or scheduling parameters (e.g., allocated budget and period). These two models, put together, allow for a precise estimate of the overall end-to-end QoS experienced by end-users.

### 6.1 Stochastic Performance Model

We built a Matlab model for simulating, by means of Monte-Carlo type discrete event simulation, a system composed of (see Figure 6): a request generator (modelling the end users), a Public Wide Area Network (WAN), a Private Network internal to an ISONI domain and the VMU hosting the actual Application Service Component (ASC). In order to account for interactivity, we modelled both paths from the user to the ASC and the other way round. The model uses a mix of exponential and Erlang probability distributions (see Figure 7) for modelling the latencies of application requests while traversing the involved networks, and it may also simulate packet loss due to buffer saturation in the various networks (particularly useful for UDP-based communications).

The individual parameters of the model need to be tuned by resorting to proper benchmarking techniques. The behaviour of the latencies inside the ISONI internal network may be accurately estimated thanks to the ISONI networking isolation, and they depend merely on the requested network-level QoS parameters specified in the VSN, and the expected application request pattern. On the other hand, parameters relative to the QoS-unaware WAN must be estimated based on available statistics on the overall network workload foreseen at the time of usage of the application. However, a widespread usage of ISONI would reduce the need for traversing QoS-unaware networks. The behaviour of the ASC was also estimated as an Erlang distribution. However, due to the non-trivial dependency of the performance from application-level parameters, in addition to the resource allocation ones, the Erlang parameters were tuned by resorting to an ANN model (see below).

The described simulator is capable of providing, for each configuration, the full probability distribution of the end-to-end response-times, as well as simpler statistics that may be easily leveraged at design-time, such as the average or a given percentile of the distribution. For example, this allows for finding the configuration parameters granting a given end-to-end response-time with a given probability.

## 6.2 Artificial Neural Networks in the Model

As already described in Section 6, the ASC part of Figure 7 is modelled through Erlang distributions. However, the characteristics (mean value and standard deviation) of these distributions are affected by application level parameters (such as the number of users) and hardware level parameters (such as the choice of scheduling periods or CPU percentages). Thus, a way of identifying and predicting these variations in the distribution characteristics must be inserted, for each different execution of the service instance on the Cloud infrastructure. We have chosen ANNs due to the fact that they can model this relationship (both linearly and non-linearly) and interpolate it for cases that have not been met before. Furthermore, they represent a black box approach, thus they do not need information regarding the internal structure of the software component. This structure (e.g. source code) is not available as seen in the introduction.

An ANN model is used for modelling the time the server needs in order to retrieve the results from the internal database. The factors that are taken under consideration are mainly the number of connected clients and the scheduling decisions (Q and P parameters). Following the black-box approach, the use of ANNs allows for an easy addition of further inputs (or outputs) as needed (e.g., hardware-specific parameters like the reserved memory or processor speed), once the necessary training data sets are collected.

The investigation of the effect of parameters like the allocated CPU time Q over a period P is critical due to its influence in the QoS output. The choice of P is mainly driven by the time granularity for the allocation needed by the application. For example, for interactive applications, with fast response times and relatively light computations, the granularity must be kept small (in the order of 10–100 ms). For scientific applications performing long and heavy computations, large periods will result in lower overheads (500 ms and beyond).

The outputs of the ANN have been chosen to represent the average and the standard deviation of the expected response times, as due to the configuration represented at the ANN inputs. These outputs are easily mapped to the Erlang parameters needed for modelling the ASC temporal behaviour in the general model in Figure 7. The ANN model structure is described in V.C., while the data gathered as training set is presented in VI.B.

## 6.3 ANN Structure and Design

In order to implement the ANN, a standard form of network was selected. The type of operation that was desired was function approximation, in order to determine the effect of the input parameters (number of clients, Q, P) on the predicted output (mean value of inner server response time and standard deviation). The collection of the data set was performed with the process described in [17]. Two more inputs were included, CPU speed and VM memory size, but the main focus was on the initial 3 parameters. The resulting network for the mean time prediction was a 3-layer, feed-forward, back propagation network, created through the GNU Octave tool[6]. It was trained with the Octave 'trainlm' func-

[6] More information is available at: http://www.gnu.org/software/octave/.

**Table 1** Structure of Mean Response Time and Standard Deviation Prediction ANN.

| Layer | Transfer Function | | Size (Neurons) |
|---|---|---|---|
| | Mean response time | Standard deviation | |
| Input | Tansig | Logsig | 5 |
| Hidden | Tansig | Logsig | 2 |
| Output | Linear (Purelin) | Linear (Purelin) | 1 |

| Mean Absolute Error | |
|---|---|
| Mean response time | 2.51% |
| Standard deviation | 2.75% |

tion, using the Levenberg-Marquardt algorithm [25]. The structure of the network is shown in Table 1. All the inputs and outputs are normalized in the (-1,1) interval. A standard form of function approximation network was used, with one hidden layer and Tansig transfer functions for the input and hidden layer and linear transfer function for the output layer.

For the standard deviation, a similar process was followed (but with the Logsig transfer function) and the resulting network also appears in Table 1.

## 7 Experimental Results

This section presents experimental results that validate the presented approach, in terms of achieving temporal isolation and performance estimation accuracy of the ANN model.

First, the assumptions of temporal isolation over which the modelling technique relies are validated. To this direction.....

Then, some experimental data used for training the ANN models is described, and finally the accuracy of the ANN-based estimations is discussed. For the ANN section, the main motivation was to validate the accuracy of the predicted expected outputs in terms of mean response time and standard deviation. Furthermore, in order to validate why ANNs were used, graphs depicting the variation of the metrics with regard to scheduling parameters are used to demonstrate the effect of the scheduling parameters. Different scheduling periods may have an effect on some of the metrics like standard deviation. Furthermore, the results highlight the difference in the distributions for varying number of users, which is a high level application parameter. This difference is predicted by the ANNs for cases that may or may not been included in the data set, through interpolation.

All experiments were conducted using instances of the Mobile Outdoor e-Learning application that has been presented in detail in Section 5.1.

### 7.1 Temporal Isolation by Real-Time Scheduling

We ran an experiment for the purpose of validating our approach to the temporal isolation of VMs concurrently running on the same CPU based on real-time scheduling. To this purpose, we considered two instances of the e-Learning application deployed on the same host and physical core. Two instances of the Mobile Outdoor e-Learning multi-user simulator, deployed on a second machine in an isolated networking context, were used to simulate the requests from 10 different users. We collected the response-times experienced by the two multi-user simulator instances under various conditions in terms of the scheduling parameters configured for the two VMs on the server host.

In Figure 8 we report the average response-time of the first VM as a function of the CPU share (on the x-axis) assigned to it, at varying CPU shares assigned to the second competing VM (corresponding to the various curves). Under the ideal conditions of perfect temporal isolation, we would like the second competing VM workload, ranging from completely idle (continuous curve) to having a 50% of load on the system (dashed curve tagged with little triangles), to have no impact at all on the performance of the first VM. This would correspond to having all the curves perfectly superimposed.

As it can be seen from the experimental results, the soft real-time scheduler achieves a nearly good approximation of such a condition, realising a set of curves which are quite close to each other, where the increase of computing resources granted to the second VM corresponds to a slight decrease of the performance of the first VM. This may be mainly attributed to an increased contention on the cache, and constitutes a minimum of interference which cannot be removed. Other factors of interference which are not trivial to keep under control are due to shared resources on the host OS, like the networking stack and interrupts. For example, see [8] for a discussion of the interference due to network-intensive VMs, and the extent to which it can be controlled by real-time scheduling of the CPU.

### 7.2 Experimental Performance of the e-Learning Application

In this section, experimental performance data gathered for various configurations of both application-level and resource allocation parameters is presented. The range of values that were altered for the configuration parameters are:

– Number of Users: 30-150
– Q/P (CPU share) : 20-100% with a step of 20

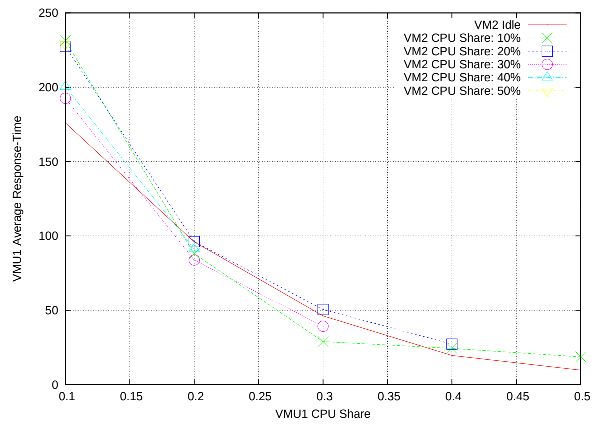


**Fig. 8** Average response-time of the first VM as a function of the CPU share (on the x-axis) assigned to it, at varying CPU shares assigned to the second competing VM (corresponding to the various curves).

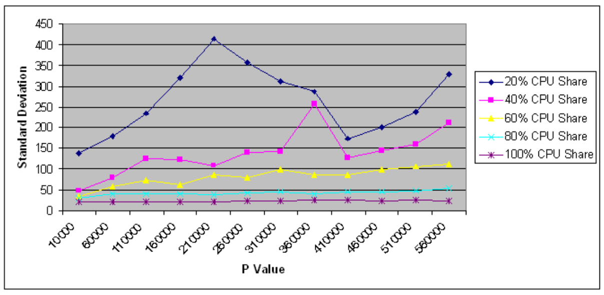


**Fig. 9** Standard deviation of response time with regard to changing P for 90 users.

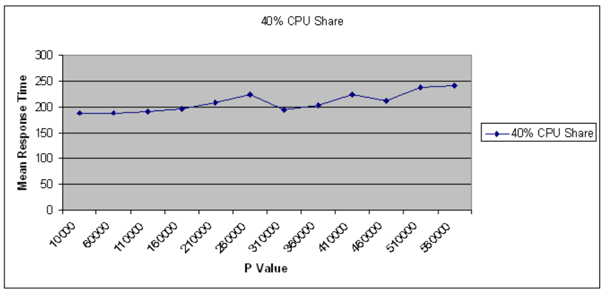


**Fig. 10** Mean response time with regard to changing P for 70 users and 40% CPU share.

– P: 10000–560000 ($\mu$sec) with a step of 50000

For each configuration, about 800 response times were collected, and the corresponding average and standard deviation figures computed. An indicative set of these measurements is discussed below.

The effect of changing granularity on the deviation of the response time values can be observed in Figure 9. This is expected since with high values of P, 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 the next active period begins. This effect decreases

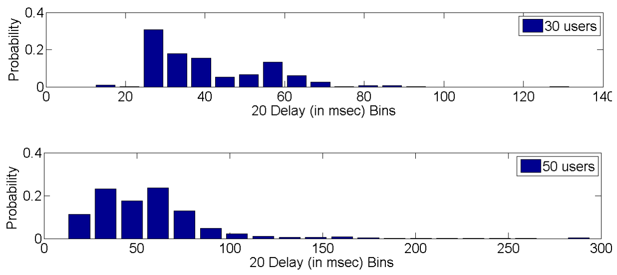


**Fig. 11** Comparison of normalised probability distribution of delay times for different number of users.

at increasing allocated CPU shares, since then the CPU is almost dedicated to the application and whenever a request arrives, it is served. The mean response time, as shown in Figure 10, seems not to be affected greatly given that the percentage of CPU assigned is the same.

In Figure 11, the comparison of the normalised probability distribution of delay times is shown for two different numbers of users (30 and 50 users). The difference especially in the maximum values of the distributions depicts the effect of the application workload in the response times. Figure 12 shows the cumulative distribution function and the confidence intervals for the same combination of number of users, given that the CPU share allocated to the VMU remains the same.

In Figure 13 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 period configuration (P) for different CPU share percentages. The upper line is for low utilization. While the utilization increases the response time decreases. In the horizontal axis, the different P configurations represent increasing period values.

From these measurements it seems interesting that the best granularity (P) should depend 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 us), as is depicted in Figure 13. For higher percentages of utilisation, lower values of P are more beneficial for the response times of the application. Furthermore, Figure 13 highlights the effect of the increased CPU share allocation to the response time.

## 7.3 Prediction accuracy of the ANN Model

For the estimation of the ANN accuracy, about 30% (87 test executions) of the data set was used only for validation. After the training of the model with the 70% of the test cases, we applied the according inputs of

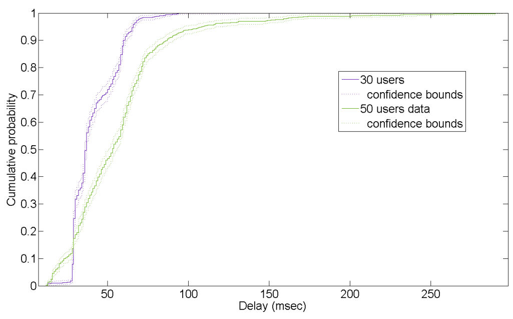


**Fig. 12** Cumulative distribution function and confidence interval for different number of users.

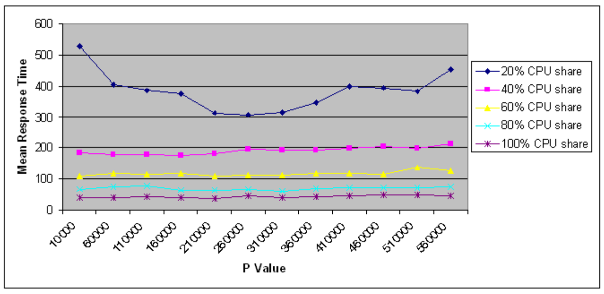


**Fig. 13** Different P's and CPU shares for 110 users.

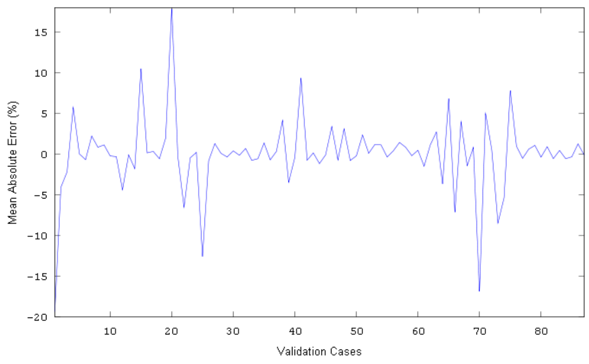


**Fig. 14** Accuracy of the ANN for Mean Response Time Prediction.

the validation cases and compared the estimated output with the observed one. The overall accuracy was around 2.5% and the error of the network for each individual test case appears in Figure 14. For each validation case, the network error appears in Figure 15.

The accuracy of the ANN models is evident from these measurements, giving sufficient reliability for this part of the overall model. The maximum deviation from the validation cases is very satisfying and so is the mean error for all the experiments. Furthermore, the predictions are not biased, a factor that is critical for the merging of different modelling approaches like in this paper.

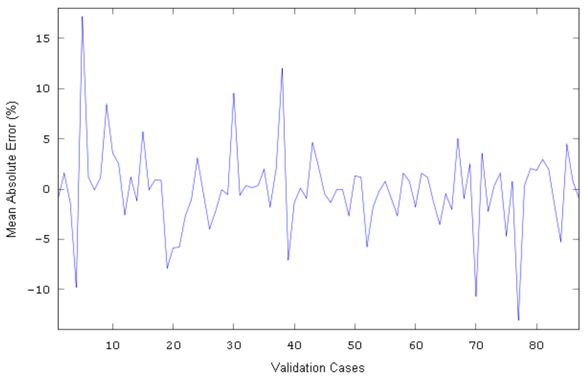


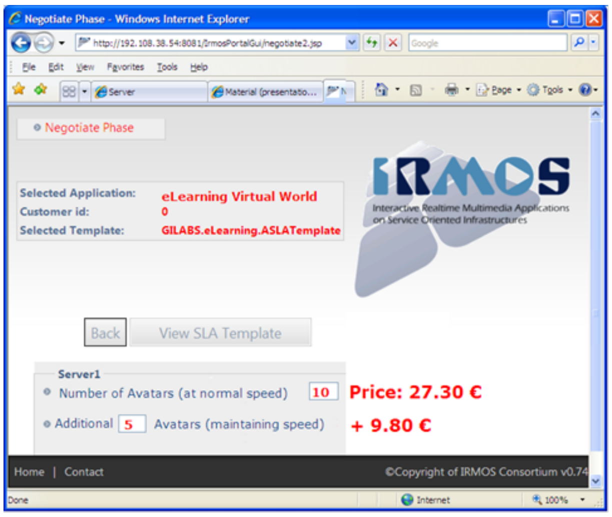
**Fig. 15** Accuracy of the ANN for Standard Deviation Prediction.

## 8 The Virtual World Use Case

Within the IRMOS context, the Virtual World e-Learning application, as presented in Section 5.2, was used to promote collaborative education through an immersive experience. In more detail, several people (learners, teachers, tutors, organizers, etc.) were allowed to meet in an interactive three-dimensional virtual world environment and interact with 3D-objects that were linked to e-Learning contents (Fig. 4).

This particular use case was used to demonstrate the run-time QoS adaption capabilities behind the IRMOS platform as described in Section 4.2. The negotiation of the Virtual World application is based on a “pay-as-you-reserve” model, i.e. the cost fluctuates according to the amount of resources that are reserved to achieve the requested performance which heavily depends on maximum number of avatars. Therefore in order to maintain the given QoS level, any extra requests for entrance should be rejected when the number of avatars already in the Virtual World is maximum.

However, the IRMOS ISONI is able to support the seamless extension of the resources, by either extending the reserved CPU, memory, network bandwidth share of the reserved resources or by performing other actions such as live-migrating the application onto other physical resources. Furthermore, the IRMOS FS Monitoring system, as already described in Section 4, is continuously collecting high-level monitoring data during run-time, such as the number of logged avatars and requests for entrance, and can combine them into rules that may trigger certain actions. To this end, the IRMOS customer is given the ability to satisfy a demand for extra resources at run-time, in order to maintain the overall performance without interruption. This has been concretized by a structured negotiation in which the customer can enable the IRMOS FS to automatically re-negotiate an extension of the resources by pre-

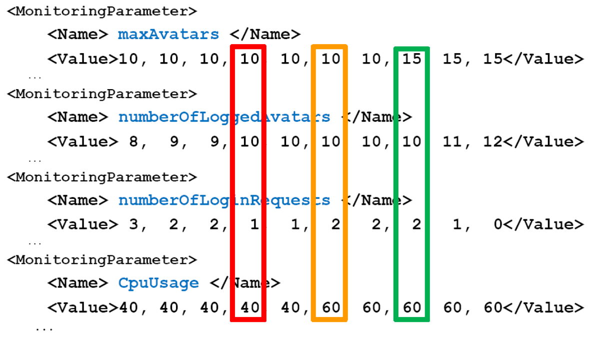

Negotiate Phase - Windows Internet Explorer
http://192.108.38.54:8081/IrmosPortalGui/negotiate2.jsp
File Edit View Favorites Tools Help
Server
Material (presentatio...
Page
Tools
IRMOS
Interactive Realtime Multimedia Applications on Service Oriented Infrastructures
Back
View SLA Template
Home | Contact
©Copyright of IRMOS Consortium v0.74
Done
Internet
100%

Negotiate Phase

Selected Application: eLearning Virtual World
Customer id: 0
Selected Template: GILABS.eLearning.ASLATemplate

Server1
Number of Avatars (at normal speed) 10 Price: 27.30 €
Additional 5 Avatars (maintaining speed) + 9.80 €

**Fig. 16** Negotiation of automatic resource extension.



```
<MonitoringParameter>
    <Name> maxAvatars </Name>
    <Value>10, 10, 10, 10  10, 10  10, 15  15, 15</Value>
 ...
<MonitoringParameter>
    <Name> numberOfLoggedAvatars </Name>
    <Value> 8,  9,  9, 10  10, 10  10, 10  11, 12</Value>
 ...
<MonitoringParameter>
    <Name> numberOfLoginRequests </Name>
    <Value> 3,  2,  2,  1   1,  2   2,  2   1,  0</Value>
 ...
<MonitoringParameter>
    <Name> CpuUsage </Name>
    <Value>40, 40, 40, 40  40, 60  60, 60  60, 60</Value>
  ...
```

**Fig. 17** Illustrative example of the QoS adaption process using snapshots (enclosed in a box). Moving left to right: at the first snapshot the renegotiation is triggered, at the second snapshot the reserved CPU share is increased, and at the last snapshot the Virtual World application is reconfigured to accept 15 avatars. The blocked avatars are allowed to log-in afterwards.

agreeing to pay an additional cost. Figure. 16 shows the application negotiation in which 5 additional avatars are foreseen.

At the end of the negotiation, the resources for the basic number of avatars (i.e. 10 avatars) are reserved and the application is deployed. Thus up to 10 avatars can enter the Virtual World with the guarantee that the requested QoS level is respected. With 10 already logged avatars, any request after that is being blocked, while the IRMOS FS are re-negotiating with the ISONI provider for resources that are able to sustain the 15 avatars under the same QoS level. In the background, the allocated resources (CPU share, memory, network bandwidth) are being increased according to the output of the performance model for the new maximum number of avatars. At the end of the re-negotiation, a VSN able to support additional users (15 avatars in total) is instantiated and the Virtual World application is reconfigured to accept 15 avatars. The blocked avatar requesting access is now allowed to enter, and the customer is charged with the agreed additional fee. The entire process of renegotiation and reconfiguration, including the addition of the blocked avatar in the Virtual World is complete in less than a minute ($\approx$ 50 secs) with the monitoring polling rate set to 10 secs for keeping the monitoring overhead absolutely negligible. However, if appropriate, one can increase the monitoring rate, to achieve a more responsive infrastructure at the cost of higher monitoring overheads imposed on computing and networking resources.

Figure 17 is an illustrative example of the renegotiation process showing also the XML-based syntax of the gathered monitoring data (see yellow box labeled 'monitor script' in Fig. 5). The monitoring parameters have 10 values, with each of them collected 10 seconds after its previous one.

## 9 Conclusions

In this work, we discussed how two real e-Learning distributed applications have been deployed with predictable and stable QoS levels within the IRMOS platform. We showed how we flanked the temporal isolation mechanisms available within the platform with proper performance analysis, modelling and benchmarking techniques, in order to investigate the performance levels achievable by the application under the various possible configurations. Furthermore, we demonstrated the run-time QoS adaption capabilities behind the IRMOS platform and the way they build proper monitoring and evaluation of application performance on top. In the future, we plan to leverage the black-box approach for performance estimation, so as to apply the described technique to other applications that are already being adapted for deployment within IRMOS, such as distributed editing of professional-quality video and a virtual reality application. Also, we plan to extend the used performance models by accounting for possibly heterogeneous hardware within an ISONI domain. Finally, we plan to extensively compare the predicted performance and the actually realised one, in presence of a variety of other deployed workload types, from compute-intensive to network-intensive ones.

## References

1. L. Abeni and G. Buttazzo. Integrating Multimedia Applications in Hard Real-Time Systems. In *Proceedings of the IEEE Real-Time Systems Symposium*, Madrid, Spain, 1998.

2. M. Addis, Z. Zlatev, W. Mitchell, and M. Boniface. Modelling Interactive Real-time Applications on Service Oriented Infrastructures. In *Proceedings of NEM Summit - Towards Future Media Internet*, September 2009.
3. S. Benkner and G. Engelbrecht. A Generic QoS Infrastructure for Grid Web Services. In *Proceedings of the International Conference on Internet and Web Applications and Services*, Guadeloupe, French Caribbean, February 2006.
4. S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. *RFC2475.* An Architecture for Differentiated Service, December 1998.
5. F. Checconi, T. Cucinotta, D. Faggioli, and G. 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)*, Dublin, Ireland, June 2009.
6. L. Cherkasova, D. Gupta, and A. Vahdat. Comparison of the three CPU schedulers in Xen. *SIGMETRICS Perform. Eval. Rev.*, 35:42–51, September 2007.
7. T. Cucinotta, G. Anastasi, and L. Abeni. Respecting temporal constraints in virtualised services. In *Proceedings of the 2nd IEEE International Workshop on Real-Time Service-Oriented Architecture and Applications (RTSOAA)*, Seattle, Washington, July 2009.
8. T. Cucinotta, D. Giani, D. Faggioli, and F. Checconi. Providing Performance Guarantees to Virtual Machines using Real-Time Scheduling. In *Proceedings of the 5th Workshop on Virtualization and High-Performance Cloud Computing (VHPC)*, Ischia (Naples), Italy, August 2010.
9. G. Dunlap. *Scheduler development update.* Xen Summit Asia, Shanghai, 2009.
10. D. Durkee. Why Cloud Computing Will Never Be Free. *Queue*, 8:20:20–20:29.
11. C. Germain-Renaud, C. Loomis, J. Moscicki, and R. Texier. Scheduling for Responsive Grids. *Journal of Grid Computing*, 6:15–27, 2008. 10.1007/s10723-007-9086-4.
12. S. Gogouvitis, K. Konstanteli, S. Waldschmidt, G. Kousiouris, G. Katsaros, A. Menychtas, D. Kyriazis, and T. Varvarigou. Workflow management for soft real-time interactive applications in virtualized environments. *Future Generation Computer Systems*, In Press:–, 2011.
13. D. Gupta, L. Cherkasova, R. Gardner, and A. Vahdat. Enforcing performance isolation across virtual machines in Xen. In *Proceedings of the ACM/IFIP/USENIX International Conference on Middleware*, pages 342–362, New York,USA, 2006. Springer-Verlag New York, Inc.
14. Z. He, C. Peng, and A. Mok. A Performance Estimation Tool for Video Applications. In *Proceedings of the 12th IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 267–276, Washington, DC, USA, 2006. IEEE Computer Society.
15. G. Katsaros, G. Kousiouris, S. V. Gogouvitis, D. Kyriazis, and T. A. Varvarigou. A service oriented monitoring framework for soft real-time applications. In *SOCA'10*, pages 1–4, 2010.
16. K. Konstanteli, T. Cucinotta, and T. Varvarigou. Optimum allocation of distributed service workflows with probabilistic real-time guarantees. *Service Oriented Computing and Applications.*, 4:68:229–68:243, December 2010.
17. G. Kousiouris, F. Checconi, A. Mazzetti, Z. Zlatev, J. Papay, T. Voith, and D. Kyriazis. Distributed interactive real-time multimedia applications: A sampling and analysis framework. In *Proceedings of the 1st International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS)*, Brussels, Belgium, July 2010.
18. G. Kousiouris, D. Kyriazis, K. Konstanteli, S. Gogouvitis, G. Katsaros, and T. Varvarigou. A Service-Oriented Framework for GNU Octave-Based Performance Prediction. In *Proceedings of the IEEE International Conference on Services Computing (SCC)*, Miami, Florida, August 2010.
19. D. Kyriazis, K. Tserpes, A. Menychtas, I. Sarantidis, and T. Varvarigou. Service selection and workflow mapping for Grids: an approach exploiting quality-of-service information. *Concurr. Comput. : Pract. Exper.*, 21:739–766, April 2009.
20. J. W. Lee and K. Asanovic. METERG: Measurement-Based End-to-End Performance Estimation Technique in QoS-Capable Multiprocessors. In *Proceedings of the 12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2006.
21. B. Lin and P. Dinda. Vsched: Mixing batch and interactive virtual machines using periodic real-time scheduling. In *Proceedings of the IEEE/ACM Conference on Supercomputing*, November 2005.
22. B. Lin and P. Dinda. Towards Scheduling Virtual Machines Based On Direct User Input. In *Proceedings of the 2nd International Workshop on Virtualization Technology in Distributed Computing*, Washington, DC, November 2006.
23. C. L. Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *J. ACM*, 20:46–61, January 1973.
24. P. Menage. *CGROUPS*, 2008. Available on-line at: http://www.mjmwired.net/kernel/Documentation/cgroups.txt.
25. J. More. The Levenberg-Marquardt algorithm: Implementation and theory. In *Numerical Analysis*, volume 630 of *Lecture Notes in Mathematics*, pages 105–116. Springer Berlin / Heidelberg, 1978. 10.1007/BFb0067700.
26. R. Nathuji, A. Kansal, and A. Ghaffarkhah. Q-Clouds: Managing Performance Interference Effects for QoS-Aware Clouds. In *Proceedings of the 5th European Conference on Computer systems (EuroSys)*, Paris, France, April 2010.
27. E. Rosen, A. Viswanathan, and R. Callon. *RFC3031, Multiprotocol Label Switching Architecture.* IETF, January 2001.
28. B. Shirazi, L. Welch, B. Ravindran, C. Cavanaugh, B. Yanamula, R. Brucks, and E. Huh. Dynbench: A dynamic benchmark suite for distributed real-time systems. In *Proceedings of IPDPS Workshop on Embedded HPC Systems and Applications*, S. Juan, Puerto Rico, 1999.
29. T. Voith, M. Kessler, K. Oberle, D. Lamp, A. Cuevas, P. Mandic, and A. Reifert. *ISONI Whitepaper*, September 2008.
30. J. Wroclawski. *RFC2210, The Use of RSVP with IETF Integrated Services.* IETF, September 1997.
31. J. Wroclawski. *RFC2211, Specification of the Controlled Load Quality of Service.* IETF, September 1997.