Resource Management Policies for Cloud-based Interactive 3D Applications

Guido Marchetto, Riccardo Sisto and Luca Stanziano
Control and Computer Engineering Department, Politecnico di Torino
Corso Duca degli Abruzzi 24, 10129 Torino, Italy
E-mail:{name.surname}@polito.it

***Abstract*—The increasing interest for the cloud computing paradigm is leading several different applications and services moving to the "cloud". Those range from general storage and computing services to document management systems and office applications. A new challenge is the migration to the cloud of interactive 3D applications, especially those designed for professional usage (e.g., scientific data visualizers, CAD instruments, 3D medical modeling applications). Among the several hurdles rising from some specific hardware and software requirements, an important issue to address is the definition of novel management policies that can properly support these applications, namely, that ensure efficient resource utilization together with a sufficient quality perceived by users. This paper presents some preliminary results in this direction and discusses some possible future work in this field. Our work is part of a wider project aiming at developing a complete architecture to offer interactive 3D applications in a cloud computing environment. Hence, we refer to this particular solution in this study.**

Keywords*—*Cloud computing, 3D graphic applications, resource management.

I. INTRODUCTION

During the last past years, we have experienced an increasing interest toward the cloud computing paradigm, both from a business and an entertainment perspective [1]. In particular, companies perceive cloud computing as an effective instrument to increase business efficiency, while home users mainly see this paradigm as an interesting opportunity for ensuring ubiquitous access to their own content or novel services.

Cloud-based services currently focus on generic storage (e.g., distributed file systems, remote disks), CPU cycles (e.g., virtual machines totally available to remote users), content distribution (e.g., file sharing, video streaming), and office applications. However, the real potentialities of this approach are still many and actually not completely investigated, mainly due to the fact that some sets of applications cannot be easily migrated to the cloud due to their specific characteristics and requirements.

The majority of the interactive 3D applications, especially those designed for professional usage (e.g., scientific data visualizers, CAD instruments, 3D medical modeling applications), certainly fall into this category. Enabling interactive 3D applications on a cloud computing system is a real challenge from several points of view. First of all, the system should guarantee an efficient access to professional graphic cards, which should be virtualized just like a traditional cloud resource in order to ensure flexibility, scalability, and cost reduction. Moreover, effective remote visualization techniques should be adopted, so that users can remotely access 3D applications running on the cloud platform with an adequate quality perceived. Last but not least, user sessions have to be properly handled within the cloud platform in order to preserve the interactive nature of the offered service together with an efficient utilization of the available resources.

As part of a wider research project aiming at defining and developing a novel architecture for the provisioning of interactive 3D services over a cloud computing platform (generically referred to as *3D cloud* in the rest of the paper), this work focuses on the last issue discussed by presenting some preliminary results on the evaluation of possible policies for resource management (i.e., the rules for selecting the pool of resources to allocate to an incoming session) in a 3D cloud platform. In particular, we investigate by simulation how dedicated and shared resource allocation approaches perform in this context. The former guarantees performance isolation among active sessions, while the latter ensures high and efficient resource usage. We compare these two approaches in order to define which solution better fit with our specific scenario. For this purpose, we adapted the well-known CloudSim simulator [2] to this specific context by including graphic resources (i.e., GPU and graphic memory). Furthermore, we characterized the workload produced by two popular 3D applications in order to obtain meaningful input traces and, consequently, significant simulation results. The entire work is based on the 3D cloud architecture under development within the abovementioned research project.

After a brief analysis of the related work (Section II) and a description of the 3D cloud platform under development (Section III), the paper describes the resource management policies considered (Section IV). Then, Section V briefly describes the simulator and the input trace generation before presenting some simulation results. Section VI concludes the paper by also outlining some possible future work (for a large part already ongoing) on the topic.

II. RELATED WORK

Existing resource management solutions for cloud systems are usually though for general applications, with the target of maximizing the overall throughput of the data center (e.g., the Load Sharing Facility (LSF) commercial scheduler [3] developed by Platform Computing and recently acquired by IBM, but also the solutions discussed in [4]). Some of these approaches also consider interactive services (e.g., [5]), thus introducing mechanisms for guaranteeing this feature together with throughput maximization. However, they limit their scope to office applications, which do not require either large amount of resources or the interaction with a graphic card. On the contrary, 3D applications are characterized by an extensive and irregular usage of both computing and graphic resources, thus requiring specific management policies. To the best of our knowledge, the only existing proposal for integrating 3D graphic applications in a cloud system is [6], which however mainly focuses on the communication issues deriving from their distributed cloud architecture and does not consider resource management. Hence, an investigation on possible resource management solutions for an efficient support of 3D applications in a cloud infrastructure is required. Furthermore, the architecture proposed in [6] is quite different than that considered in this research project, as it will be discussed in the next section.

In this paper, we compare dedicated and shared approaches for resource provisioning to incoming sessions. The former is usually adopted in QoS-aware cloud systems to avoid interference among different sessions [7], [8]. In essence, a set of available resources (e.g., one or more processor cores, some memory or disk space) are exclusively allocated to a given virtual machine, so that a user can perceive the same service as on a dedicated local workstation. However, this approach may result in a large waste of resources if users experience long inactivity periods as their own resources cannot be used by any other session. This may be the case for interactive applications due to their particular workload and resource utilization. These issues will be discussed in the following.

III. 3D CLOUD ARCHITECTURE

This work is part of a wider research project that aims to define a novel architecture for cloud-based 3D applications. We briefly discuss this ongoing work in order to both highlight some possible critical constraints that also apply to our resource scheduling problem and describe the deployment scenario to which we refer our policies and results.

As in a traditional cloud computing environment, the overall system is based on a data center, where applications are installed and run, and the user's machine, from which the user accesses the service offered in the cloud. Whenever a user asks the system to start a new session, this is associated to a virtual machine instance in the data center.

Given the 3D-graphic nature of the applications we are dealing with, a key point to consider is the availability of graphic resources at the data center. Technologies such as NVIDIA SLI [9] and ATI/AMD Crossfire [10] enabled the utilization of multiple graphic cards on the same physical host, but what is missing is an effective technique for sharing these resources among several virtual machines. Some existing solutions are based on software emulation [11], which does not offer adequate performance to professional applications. Others limit each graphic card to be exclusively assigned to one of the virtual machines [12], which is not efficient. In [6], the solution proposed is based on hosting a large number of graphic cards on separate machines (the so called *graphics rendering servers*) and remotely access them from the application servers. Our architecture, instead, will rely on the hardware virtualization of the GPU (e.g., [13]), which enables the concurrent access of several virtual machines to graphic resources locally deployed in the application server. This ensures performance improvements with respect to the remote approach. It is worth noticing that current graphic cards do not allow distinguishing between status information of different virtual machines that concurrently access this resource. Hence, virtual machine migration is not possible in this kind of architecture as the graphic status information cannot migrate together with the session. This significantly affects our work on resource management as resource selection has to be good enough to guarantee adequate performance for the entire duration of the session.

Another important aspect of the 3D cloud architecture is the communication between the user's client and the data center. In particular, efficient remote visualization techniques have to be adopted in order to guarantee an adequate Quality of Experience (QoE) to the user. Several solutions exist (e.g., [14]–[17]). Work is ongoing to improve these approaches, especially in a WAN scenario.

IV. RESOURCE MANAGEMENT POLICIES

Graphic 3D applications may be characterized by five fundamental resource components, which have to be considered in a resource management module of a 3D cloud platform for performing session allocation:

- *CPU cycles* on the physical host, required to run the virtual machine associated to the user and, in particular, to execute the 3D application.
- *Memory* on the physical host, i.e., the memory resources associated to the virtual machine.
- *GPU cycles* on the graphic card, required for graphic processing.
- *Graphic memory* on the graphic card for hosting graphic data to be processed.
- *Bandwidth availability* between the data center hosting the application and the user location.

The latter is actually the most significant as poor bandwidth availability may invalidate any attempt to improve resource scheduling on the data center. However, in this paper we would like to focus on resource scheduling on a given data center, where the bandwidth availability may be considered constant for any possible physical host selected. Hence, we do not include bandwidth availability is our scheduling criteria. Other resource components are clearly required, e.g., some disk space on the cloud system, but are not critical for the application and hence not considered in this work for simplicity.

As discussed in Section II, isolation is a viable solution to offer service guarantees to incoming sessions. Given the amount of resources required by an incoming session, these may be exclusively reserved to that session in a physical host with sufficient availability. In our context, this might be extremely inefficient. First of all, current technology does not allow efficient isolation of graphic resources. Different sets of GPU cores, as well as different portions of the graphic memory, cannot be statically assigned to different processes. The only available solution [12] is the association of the graphic card as a whole to the virtual machine related to a specific session, which hence has exclusive access to it. Clearly, this solution does not scale to a cloud computing environment, where tens of sessions may be hosted in a single host. A second problem is related to CPU core reservation. It is possible to dedicate one or more CPU cores to a single section. However, since the number of cores currently available on modern CPU is still relatively low, this approach would significantly reduce the maximum number of sessions that can be hosted on a single physical host.

In addition to that, it is worth considering the specific workload of interactive 3D applications. First of all, the task of graphic resources is usually the generation of image flows at a given frame rate that are sent to a display. Given the target frame rate, the graphic card exploits the portion of its own resources to reach that target, thus being potentially underutilized. Furthermore, typical interactive sessions are based on sequences of commands that users send to the system, spaced out by inactivity periods usually referred to as "thinking times". During thinking times, a large portion of resources are actually unutilized and might be redistributed among other active sessions. For these reasons, a resource provisioning approach based on resource sharing might be interesting in this context for ensuring efficient resource utilization while preserving the quality perceived by end-users. In particular, this may hold in small private clouds, where cost reduction and efficient utilization of available resources is crucial for revenue maximization.

This considered, we define and compare the following resource provisioning approaches for the resource management module of a 3D cloud platform:

- *Dedicated resource reservation*: the requested amount of resources is statically assigned to every incoming session, thus implementing the isolation principle.
- *Shared resource utilization*: resources are freely shared among sessions and new sessions can be added until a defined threshold is reached.

In this second approach, for example, all the cores of a given CPU are seen as a unique processing unit and incoming sessions are associated to that CPU without strict core reservation. Furthermore, resources on a graphic card (i.e., GPU and graphic memory) are shared by several sessions, thus increasing efficiency. This is obtained thanks to the GPU hardware virtualization technology adopted in our cloud architecture.

V. Simulation results

A. CloudSim toolkit

The evaluation of the resource management policies presented in the previous section has been made in a simulation environment developed within CloudSim, a Java-based toolkit for modeling and creating virtual machines over simulated datacenter. CloudSim implements scheduling, allocation, and migration policies and also supports the creation of customized ones. A lack of this toolkit was the absence of graphic resources among the possible resource components offered by simulated data centers. Hence, part of our work has been the development of proper classes for simulating also the behavior of these specific physical resources, which are of key importance in our context. In particular, the new resource graphic card, geared with proper memory and processing functionalities, has been added to the toolkit. One or more simulated graphic cards can be associated to a simulated physical host, thus reproducing the real environment of a 3D cloud data center.

Whenever a new session setup request is triggered, the simulator checks if one or more hosts have enough resources (i.e., CPU cycles, memory, GPU cycles, and graphic memory, as discussed in Section IV) to accommodate the virtual machine according to the specific resource provisioning model adopted. Then, the hosting node is selected among the eligible ones by applying the configured scheduling algorithm.

B. User session profiling

The overall workload on a 3D cloud significantly varies depending on the kind of applications deployed and the specific user behavior. This results in resource requests that are generally unpredictable in number, frequency, and size. Long inactivity periods may be interrupted by sudden requests of a huge amount of resources due to specific commands launched by users. Otherwise, the amount of required resources may be almost constant for the entire session duration. This is the case, for example, for the request pattern at the GPU when it is used for runtime rendering.

For this reason, the adoption of proper input traces is fundamental for obtaining a significant evaluation of the resource management policies described. Hence, we monitor and sample the resource utilization data of two different 3D applications, which we consider significant and representative of the wide 3D graphic scenario: *Blender* [18], a popular modeling and 3D animation software, and *The Elder Scroll V: Skyrim* [19], a recent videogame. The former is a typical instance of interactive application based on sequences of commands and inactivity periods. The latter requires a more constant resource utilization pattern as resources are used to elaborate continuous flows of frames at a given rate. To be exhaustive, we monitor the occupancy of both processing (i.e., CPU and GPU, in percentage) and dynamic memory (i.e., RAM and GRAM, in byte) units. The monitored data are used as a basis for constructing the input workload of our simulator. 10 different sessions has been monitored for this two softwares and in our tests we randomly extract an input trace with 50% of probability to use one from Blender or one from the videogame.

The duration of each monitored session is about 1 hour and the selected sampling period is 1 second. The machine used for this data collection is a notebook PC with an Intel i7-2640M - 2.8 GHz processor, 8 GB RAM, and a NVIDIA GeForce 610M graphic card with a 1GB graphic memory. Fig. 1 shows the results of this measurement campaign. Notice the different resource usage of the two applications.

C. Results

In the following we evaluate by simulation the two resource provisioning models discussed in Section IV. In particular, we compare the dedicated resource provisioning model implementing the "isolation" concept (currently adopted in QoS-aware cloud architectures) with a more efficient shared ap-

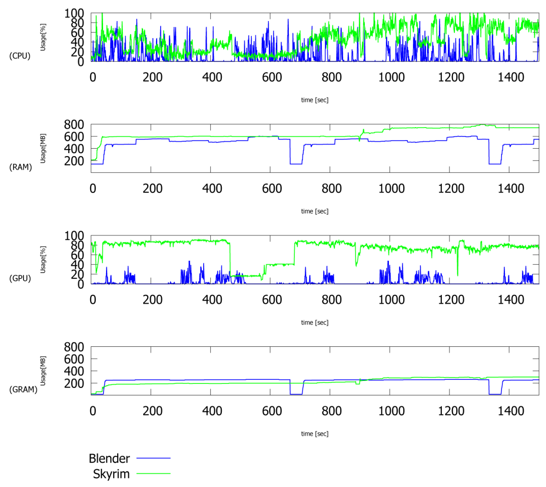

Fig. 1. Example of resource usage of two popular 3D graphic applications.

proach. The simulated cloud architecture is based on 10 physical hosts, each one geared with a 6-core Intel i7 Extreme 980X processor with HyperThreading technology (i.e., 12 virtual processor cores), 64 GB RAM, and two NVIDIA Telsa 2070q graphic cards. User session workloads are defined according to the above described profiling, i.e., each session perfectly maps the resource usage of either a 3D modeling applications or a 3D videogame. CPU and GPU usage percentages are renormalized according to the computational capacity of the processors available in the cloud. 3D modeling and videogame sessions are supposed to be equally distributed. Furthermore, a simple *greedy worst-fit scheduling* (i.e., the load is balanced among the available hosts) is used. We also recall that sessions cannot migrate in our context due to graphic card technology limitations.

Given the interactive nature of our applications, the performance metric of main interest in our context is the delay perceived by users during their normal operation (e.g., a delay between a click and the actual visualization on the screen, as well as a movement command on the joystick and the actual movement observed). To be more precise, in our context we are interested in the additional delay perceived by users compared to the delay observed in a local dedicated workstation, where the amount of requested resources is always available for the user. However, the exact identification of both a given command (e.g., a click or a joystick movement) and the visualization instant of the corresponding result on the screen is not practicable. Hence, we adopt the *yield* as a performance metric, introduced in [20] for the same purpose. Compared to [20], we slightly modify the yield definition. Here it is defined as the fraction of resources that can be used on a given node by a given session at a given instant of time, divided by the session's resource request at that instant. The additional delay strictly depends on the yield value. The lower the yield, the larger is the additional delay.

It is worth noticing that the advent of 64-bit computing, together with the lowering of RAM costs, significantly increased the memory availability on a physical host, which therefore is now larger than graphic applications needs. Hence, we can assume that strict reservation of memory is a feasible approach in our context and a shared approach is not necessary. On the contrary, it is evident that a dedicated reservation of each graphic card to a given virtual machine would result in an accommodation of only two user sessions in our exemplifying cloud system, which clearly provides service guarantees but is extremely inefficient. This is a first straightforward result of our analysis and represents a lower bound of the number of graphic sessions that can be supported in a 3D cloud without affecting the quality of the offered service.

More interesting is the case of graphic cards shared among several user sessions. We first consider a dedicated reservation of CPU cores, currently adopted as a form of isolation among the existing virtual machines. If we suppose that each session requires one core, we can accommodate a maximum of 120 sessions in our cloud (a virtual machine for each virtual core of each physical host). In this scenario, the cloud can perfectly support the service (the observed yield never distances itself from its target value of 1 for the entire duration of the simulation) but resources are not used efficiently (the observed average utilization of processing and memory resources is: CPU= 31%, RAM= 10%, GPU= 28%, GRAM= 20%). Resource utilization efficiency may be improved by deploying a shared utilization of the available CPU cores, where a larger number of sessions can be accommodated in each physical host. This may degrade the perceived quality, as several sessions compete for CPU cycles, GPU cycles, and graphic memory (we recall that central memory is always reserved in our tests) on the host. However, Fig. 2 shows how the particular workload of graphic applications well fits with this approach, which ensures an average yield degradation lower than 1% on any of the considered resources until an average value of 24 sessions per host are accommodated in the cloud. Hence, a shared resource utilization model allows supporting about a 100% workload increase with respect to the previous case without experiencing any service degradation. In order to better investigate the real performance of the system, Fig. 3 plots the evolution of the CPU yield (which Fig. 2 demonstrates to be the one that first experiences a degradation) over time. The reason why CPU is the bottleneck can be found by the fact that the GPU load is partitioned among multiple graphics cards as opposed to the CPU that receives the load of all sessions alone. We consider the boundary scenario of 25 sessions per host, i.e., the first workload experiencing an average CPU yield degradation larger than 1%. The figure shows how this slight degradation is mainly due to temporary service disruptions observed for about 6% of the total session time. Such a performance result, together with this outage duration, is even better than the values that may characterize free 3D cloud services without guarantees. Hence, this may be another interesting deployment scenario for the shared resource utilization model, as ensures significant cost reduction.

To conclude our analysis, we evaluate the potential of the shared resource utilization model in a possible power saving mode. We change the scheduling policy to a *greedy best-fit* approach, i.e., sessions are accommodated on the smallest number of hosts. The threshold used to consider a host full in the shared resources approach is the degradation of 1% of the average yield of one of the involved resources. We consider a target scenario of 60 sessions submitted to the cloud system. The schedule of the sessions in the dedicated CPU core scenario is again trivial: 5 hosts will be used and each of them will accommodate 12 sessions. In this scenario, yields clearly

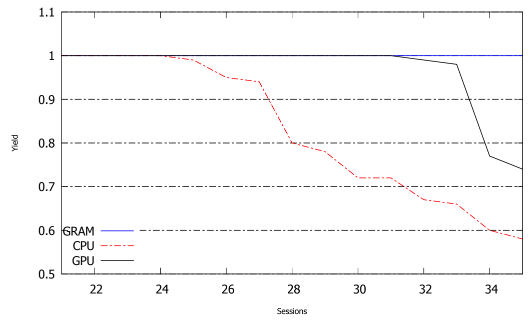

Fig. 2. Observed yield as a function of the number of accommodated sessions per host.

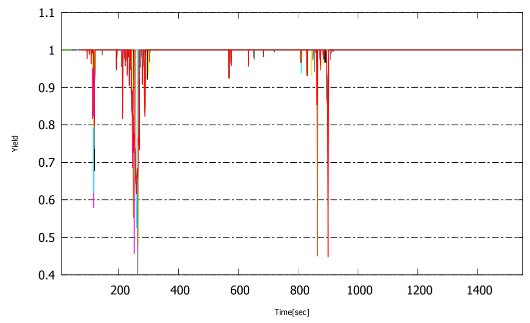

Fig. 3. Observed yield as a function of the session time.

TABLE I. BEST-FIT SCHEDULING WITH 60 SESSIONS

HostID	Load(Mean)	Yield(Mean)	Yield(Variance)
Host0	68.82	0.99	0.01
GraphicCard00	60.32	1	0
GraphicCard01	66.02	1	0
Host1	72.82	0.99	0.01
GraphicCard10	63.02	1	0
GraphicCard11	65.17	1	0
Host2	28.22	1	0
GraphicCard20	24.24	1	0
GraphicCard21	26.54	1	0

do not experience any degradation but resource utilization is very low. Instead, the same sessions can be accommodated in only 3 hosts when CPU cores can be arbitrarily shared among them. This results in a 40% power saving, which gains a great significance in our green computing era. Results on average CPU and GPU load conditions and on the average CPU and GPU yield values (whose degradation is lower than 1% by construction) are reported in Table I.

VI. Conclusion

This paper presents some preliminary results on some possible policies for resource management in a 3D cloud platform. In particular we showed how a shared resource utilization model may outperform a traditional resource reservation approach based on isolation thanks to the particular workload of a 3D cloud system.

Additional work is ongoing to refine and enrich the presented results. In particular, proper load prediction techniques based on past load observations are under investigation for improving scheduling decisions. Furthermore, we are studying some yield balancing methodologies for avoiding unbalanced performance among sessions on the same host. Finally, the shared resource utilization model has to be implemented in the cloud architecture under development within the ongoing research project of which this work is part.

References

[1] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: State-of-the-art and research challenges," *Journal of Internet Services and Applications*, vol. 1, pp. 7–18, May 2010.

[2] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, Jan. 2011.

[3] IBM, "Load sharing facility (lsf)." [Online]. Available: http://www-03.ibm.com/systems/technicalcomputing/platformcomputing/solutions/workloadmanagement.html

[4] O. Khalid, I. Maljevic, R. Anthony, M. Petridis, K. Parrott, and M. Schulz, "Dynamic scheduling of virtual machines running hpc workloads in scientific grids," in *Proc. IEEE Int. Conf. on New Technologies, Mobility and Security (NTMS)*, Dec. 2009, pp. 1–5.

[5] Y. Song, H. Wang, Y. Li, B. Feng, and Y. Sun, "Multi-tiered on-demand resource scheduling for vm-based data center," in *Proc. IEEE/ACM Int. Symp. on Cluster Computing and the Grid (CCGRID)*, May 2009, pp. 148–155.

[6] W. Shi, Y. Lu, Z. Li, and J. Engelsma, "Scalable support for 3d graphics applications in cloud," in *Proc. IEEE Int. Conf. on Cloud Computing (CLOUD)*, July 2010, pp. 346–353.

[7] R. Nathuji, A. Kansal, and A. Ghaffarkhah, "Q-clouds: Managing performance interference effects for qos-aware clouds," in *Proc. ACM European Conf. on Computer Systems (EuroSys)*, Apr. 2010, pp. 237–250.

[8] S. Govindan, J. Liu, A. Kansal, and A. Sivasubramaniam, "Cuanta: Quantifying effects of shared on-chip resource interference for consolidated virtual machines," in *Proc. ACM Symp. on Cloud Computing (SOCC)*, 2011, pp. 22:1–22:14.

[9] NVIDIA, "Scalable link interface (sli)." [Online]. Available: http://www.geforce.com/hardware/technology/sli

[10] AMD, "Crossfire." [Online]. Available: http://sites.amd.com/us/game/technology/Pages/crossfirex.aspx

[11] ORACLE, "Virtual box." [Online]. Available: http://www.virtualbox.org/

[12] CITRIX, "Multi-gpu passthrough." [Online]. Available: http://www.citrix.com/products/xenserver/overview.html

[13] NVIDIA, "Vgx hypervisor." [Online]. Available: http://www.nvidia.com/object/vgx-hypervisor.html

[14] B. De Vleeschauwer, F. De Turck, B. Dhoedt, P. Demeester, M. Wijnants, and W. Lamotte, "End-to-end qoe optimization through overlay network deployment," in *Proc. IEEE Int. Conf. on Information Networking (ICOIN)*, Jan. 2008, pp. 1 –5.

[15] F. Lamberti, C. Celozzi, A. Sanna, and G. Paravati, "A feedback-based control technique for interactive live streaming systems to mobile devices," *IEEE Transactions on Consumer Electronics*, vol. 56, no. 1, pp. 190–197, 2010.

[16] A. Sanna, F. Lamberti, G. Paravati, and L. Ciminiera, "An adaptive control system to deliver interactive virtual environment content to handheld devices," *Journal on Special Topics in Mobile Networks and Applications*, vol. 16, no. 3, pp. 385–393, 2011.

[17] L. Ciminiera, A. Sanna, F. Lamberti, and G. Paravati, "An open and scalable architecture for delivering 3d shared visualization services to heterogeneous devices," *Concurrency and Computation: Practice and Experience*, vol. 23, no. 11, pp. 1179–1195, 2011.

[18] "Blender." [Online]. Available: http://www.blender.org

[19] "The elder scroll v: Skyrim." [Online]. Available: http://www.elderscrolls.com

[20] M. Stillwell, F. Vivien, and H. Casanova, "Dynamic fractional resource scheduling for hpc workloads," in *Proc. IEEE Int. Symp. on Parallel Distributed Processing (IPDPS)*, Apr. 2010, pp. 1 –12.