Cloud-based Desktop Services for Thin Clients

Lien Deboosere, Bert Vankeirsbilck, Pieter Simoens, Filip De Turck, Bart Dhoedt and Piet Demeester

***Abstract*—Cloud computing and ubiquitous network availability has brought the thin client concept again under the attention. Executing applications in virtual desktops on servers in the cloud enables accessing any application from any location with any device. To be a successful alternative for traditional offline applications, important challenges have to be overcome. First of all, the performance of the thin client protocol is essential: the audiovisual output has to be displayed fluently. Second, the desktop should be executed on a server with sufficient resources, ideally close to the user's current location to limit the impact of network delay on the interactivity. In addition to delivering excellent user experience, reducing costs is important for service providers. This paper discusses these challenges from both the user's and the service provider's point-of-view and roads to solutions enabling cloud-based desktop services for thin clients.**

I. Introduction

RECENTLY, cloud computing [1] services have become widely available, offering an on-demand availability of computing resources (e.g. Amazon EC2). Thanks to these advances and ubiquitous network availability, the thin client computing paradigm is enjoying an ever increasing popularity. The paradigm, originally intended for wired local area network (LAN) environments [2], is repeating its success in the mobile context. Furthermore, a study from ABI Research forecasts a 20 B$ turnover associated with services directly associated with mobile cloud computing by the end of 2014. Clearly, when offloading is applied, the functionality of the terminal is limited to presenting audiovisual output to the user and to conveying user input to the remote servers, thereby considerably reducing the computational complexity on the client device. Consequently, these applications can be run as-is, without the need to provide (many) scaled-down versions for mobile devices.

Currently, popular applications are already executed on servers in the cloud (e.g., Google Docs, Microsoft Live). Accessing applications in the cloud is referred to as Software-as-a-Service (SaaS), while hosting a virtual desktop (VD) is referred to as Desktop-as-a-Service (DaaS). DaaS implementations can be categorized according to the location where the VD is executed (locally or remotely) or according to the method to access the output of the VD (browser or thin client protocol such as Microsoft's Remote Desktop Protocol (RDP) and Virtual Network Computing (VNC)). This paper specifically focuses on mobile users, and hence, due to resource constraints, VDs are executed remotely. To enable access to existing operating systems and applications, we will focus on using a thin client protocol to visualize the output of applications executed by a VD.

All authors are with the Ghent University - INTEC - IBBT Belgium. Simoens is also affiliated with INWE, Ghent University College.

Current DaaS deployments such as VMWare Virtual Desktop Infrastructure (VDI) are mainly concentrated in corporate environments. The availability of (virtual) computing resources distributed over the network makes cloud computing a big enabler for offering desktop services in mobile WAN environments.

In Fig. 1, we propose a system architecture for offering efficient desktop services in the cloud. Simplified OS image management (i.e. re-using an OS image among users and consequently reducing the storage per user) and application management are essential for the scalability of the service. In our system architecture, a VD is built from a shared *golden image* from the OS database, merged with personal settings, for example by using a copy-on-write solution with unionFS. Schwarzkopf et al. show how multi-layer VDs simplify the complexity of upgrading the golden image without causing broken dependencies and/or conflicts [3]. To improve the usability of DaaS, combining DaaS with application virtualization technologies such as Softricity and Microsoft App-V is very promising. The applications are then dynamically delivered to the user's VD without the need for installing, configuring and updating the applications. This approach further reduces the complexity of upgrading golden images since applications are not installed in the user's VD and thus cannot be broken.

Existing cloud platforms fulfill the hardware requirements to implement a DaaS service. However, we are witnessing the emergence of a new category of mobile applications (e.g., augmented reality, rich sensing and multimedia editing), posing stringent requirements on delays. Current cloud management systems are not able to meet user expectations from these applications, especially in terms of latency. There is a clear need for novel cloud management algorithms to take into account the specific requirements of mobile thin client computing. In our system architecture, these cloud management algorithms are implemented in the self management component of the service manager, which can be implemented as part of existing cloud management systems such as OpenNebula, OpenStack and Eucalyptus.

In this paper, we present and discuss solutions to adequately address the challenges for providers offering a cloud-based desktop service. First, we look at the service from the user's point-of-view and discuss how the user experience can be improved. Second, from the service provider's point-of-view, we discuss how to reduce the costs for offering the service.

II. User experience

Two aspects are important for the user experience: *(i)* high performance of the thin client protocol, i.e. crisp interactivity and fluent audiovisual output, and *(ii)* sufficient allocated resources on the server-side so the applications respond quickly.

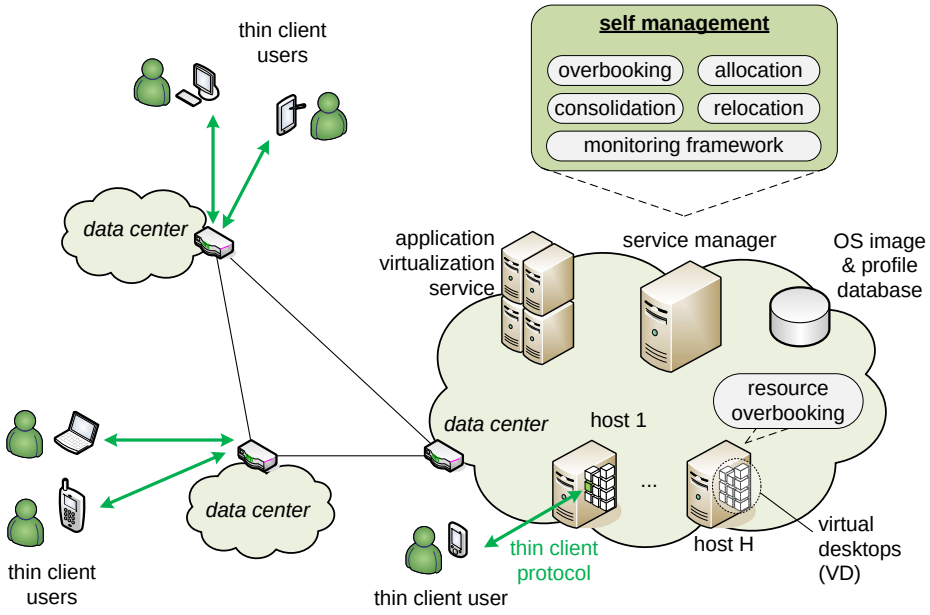


Fig. 1: System architecture to enable cloud-based desktop services for thin clients. Users connect via a thin client device (e.g., smartphone, tablet PC, PDA, netbook, minimal state or zero-state thin client device) to their remote applications executed in a virtual desktop. The self management component of the service manager covers optimizations to increase the user experience and decrease the costs of the service provider.

For mobile users, it is also important to reduce the energy consumption on the client device.

A. *Crisp interactivity*

Acceptable interaction delay bounds depend on the application at hand. For office automation applications, delays up to 150ms can be tolerated [4], while for multimedia applications such as video games, users are already susceptible for interaction delays higher than 80 ms. Because the result of user input can only be seen after at least one round-trip time (RTT), delay for critical applications should be addressed through the use of a proximate server. Every time a user connects to the service, a data center has to be selected that can be reached fast enough from the user's current location. Inside the data center, the selection of an appropriate server is based on the expected resource requirements of the user's applications based on the user's profile and the current load on the servers (see section II-C).

Due to user mobility, guaranteeing delay bounds can imply that a VD is to be migrated to another server. In practice, the desktop service is continuously monitored and, for example, when the RTT exceeds a predefined boundary based on the type of active applications, the user's VD is relocated to a more suitable host. The relocation is performed by *live migration* of the VD [5].

By storing the unionFS delta filesystem of the VDs on network storage equipment, the cost of relocating VDs from the service provider's point-of-view is reduced to copying the current active memory and in worst-case (i.e. migration across data centers) also copying the delta filesystem of the VDs. During the migration process, resources are required on both the original and target host. In a heavily loaded system, these double resource reservations can lead to rejecting new user requests while also causing substantial network traffic for the memory copying process. Therefore, relocating VDs should only be performed when valuable improvements for the customer and/or the service provider can be achieved.

B. *Fluent audiovisual output*

Multimedia content has been a stumbling block for thin client computing for many years, especially in mobile WAN environments where bandwidth availability is limited and expensive, mainly due to the fact that the same coding is applied to static (e.g., text editing) and to dynamic (e.g., video game) content. Recently, several bandwidth optimizations for thin client protocols have been proposed. An important innovation implements a channel to redirect multimedia in its original format to the client (e.g. Citrix SpeedScreen), at least when the appropriate codec is available on the client device. This approach is only valid for playing multimedia streams and not for displaying high-motion output from an application (e.g., a video game). We have evaluated a thin client protocol optimization that encodes the high-motion output of applications with a video codec and switches to a thin client protocol to encode low-motion output [6] and show its feasibility for popular mobile devices in Table I. In these experiments, a full-screen video was played on the server and streamed to the client. Since the bottleneck of the live encoding process is the CPU of the server, higher framerates are reached for smaller screen resolutions. Using the processing power of a Graphical Processing Unit (GPU) on the server-side could improve the framerate.

device	screen resolution	available codecs	streaming framerate [fps]
iPhone 4	640 x 960	H264, MPEG-4, M-JPEG	27
Samsung Galaxy S	800 x 480	H263, H264, MPEG-4, WMV, VC-1	23
iPad	1024 x 768	H264, MPEG-4, M-JPEG	20
laptop	1280 x 1024	depends on OS/applications	12

TABLE I: Comparison of the performance of thin client computing on popular mobile devices. The output of a graphical-intensive application (i.e., a full-screen video) is live encoded with H264 before streaming to the client device. The bottleneck of the live encoding process is the CPU of the server. Therefore, a higher framerate can be reached for smaller screen resolutions

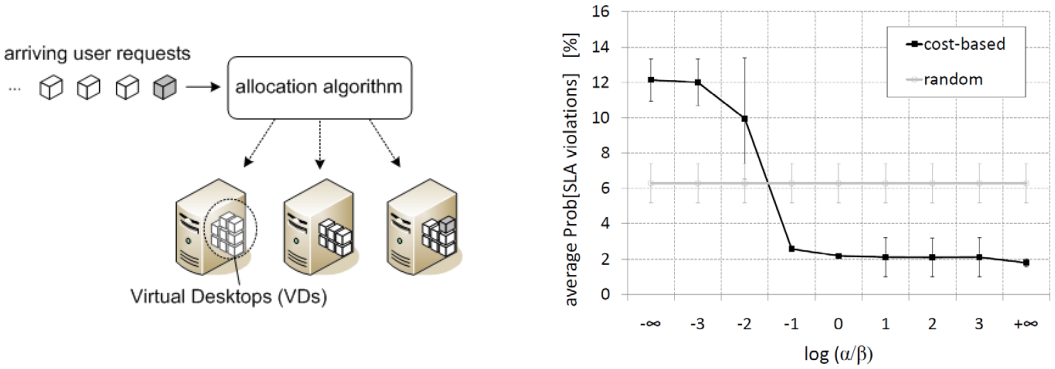


Fig. 2: Simulation results of the proposed allocation algorithm in a scenario with 10 hosts and an average utilization of 90%. An SLA violation means that the user applications receive less resources than requested.

Vankeirsbilck et al. propose to cache important output sequences such as the desktop view and menu items to reduce both the required bandwidth and the interaction delay [7]. Simoens et al. present a complete overview of recent thin client protocol optimizations in [8].

C. Resource allocation

In the data center, a suitable host to satisfy an arriving user request must be found. From the customer's point-of-view, the least utilized host is preferable, while from the provider's point-of-view the host resulting in the least resource fragmentation (i.e. the *best-fit* host) is preferable since this can reduce the energy consumption. The resources needed by a user are specified in a Service Level Agreement (SLA). To observe the balance expressed above, the allocation algorithm attributes a penalty of α for each request receiving too few resources, and a penalty β related to resource fragmentation, i.e. to the amount of non-reserved resources on this host. The host with the lowest penalty is selected to handle the user request. In Fig. 2, the influence of the ratio α/β on the probability of SLA violations is shown for a simulation with 10 hosts and an average utilization of 90%. In this context, an SLA violation implies that the user applications receive less resources than requested. When α/β increases, i.e. when SLA violations are expensive, the allocation algorithm is able to reduce the probability of SLA violations with 10%. An SLA violation as defined here might not be noticeable or obstructive for the user experience as it might just take some longer for the user applications to execute a task.

For scalability, we cannot assume that every user has a dedicated profile. Rather, the resource requirements of VDs should be clustered offline into a finite number of profiles. At subscription time, a user is assigned one of those predefined profiles. An online clustering algorithm such as the decentralized clustering algorithm presented by Quiroz et al. [9] could be used to map the current resource requirements of a user's VD to one of the cluster profiles. This online mapping can be used to adapt the current resource allocation or even the user's profile when appropriate.

In case the current resource requirements do not correspond with the user's profile (e.g., bursts of SLA violations are detected), the cloud management can decide, based on the user's SLA contract, to adapt the resource allocation to the current needs. In case more resources are required and sufficient resources are available on the current host, these additional resources are simply allocated. A problem arises when the current host cannot update the resource reservation to the desired level. Two actions can be taken: relocating the user's VD to a host with sufficient free resources, or relocating other VDs from the current host until sufficient resources are freed. The preferred choice depends on several factors such as the users' SLA contracts and the memory consumption of the VDs which is known to determine the time required to finish the live migration of a VD.

D. Battery autonomy

Limited battery drain is important for mobile users. Since the computing power is shifted to the network, one could expect a small battery drain, but on the other hand, the continuous wireless network connection is a huge battery consumer.

Several approaches exist to reduce the energy consumption of the wireless network connection, for example, Simoens et al. propose a cross-layer optimization [8]. Even with this

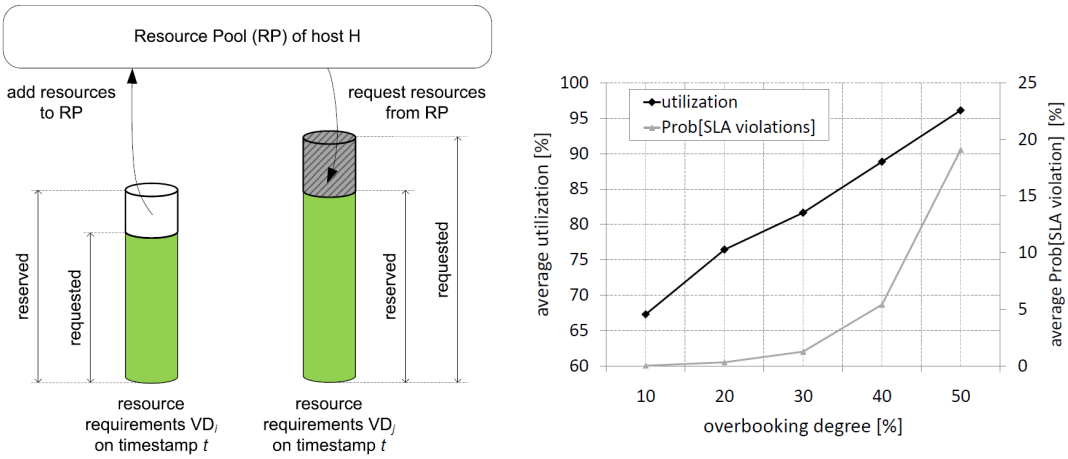


Fig. 3: Non-consumed reserved resources are collected in the host's resource pool to be shared among VDs requesting more resources than reserved. The presented simulation results (averaged over 15 simulations) concern a fully reserved host with *normal* VDs requesting resources (based on the planning guide by Citrix Inc. [11]) according to a normal distribution $N(\mu, \sigma^2)$ with μ taken from $N(10, 3.5)$ and σ^2 taken from $N(3.5, (2/3 \times 3.5))$.

adaptation, offloading all applications cannot be justified in terms of reducing energy consumption. Therefore, we propose to weigh the advantages of offloading an application to a remote server versus local execution of the application. Lu et al. propose a solution between these two extremes: offloading parts of the applications and rendering to remote servers and executing the other parts locally, which could also reduce the interaction delay [10].

III. Service provider costs

The most important challenge for a service provider is satisfying the customers while minimizing the costs. We focus on optimizing the number of users served by a single host and minimizing the energy consumption of the hosts in the cloud.

A. Number of users served by one host

Depending on the targeted user experience, resources should be reserved on the infrastructure. Of course, reserving worst-case resource needs will lead to an over-provisioning of cloud resources. The planning guide by Citrix Inc. [11] suggests assigning at most 10 *normal* VDs or 4 *heavy* VDs to a single host. Since resolutions of mobile devices are becoming closer to resolutions of regular screens, the difference in resource requirements for hosting a VD for a mobile or for a fixed user is negligible. Therefore, the study is also valid in nowadays mobile context. Berryman et al. show that, when more VDs are assigned to a host, the performance degradation depends on the type of applications executed in the VDs [12]. Therefore, the number of allocated resources should be dependent on the applications expected to be executed, as specified in the user's profile.

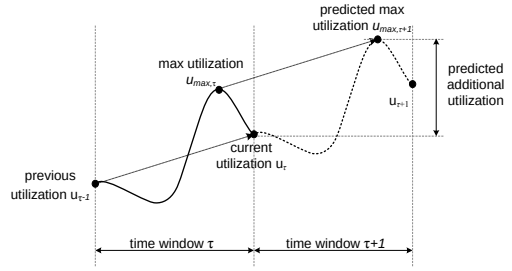
Urgaonkar et al. demonstrate the importance of resource overbooking to avoid over-provisioning of resources in the context of shared internet hosting platforms [13]. Based on the observation that the resource requirement of a VD varies a lot and depends on many factors such as multiple active applications, there is an opportunity to use a resource overbooking technique in the context of virtual desktop computing.

In Fig. 3, we propose a novel overbooking technique that takes advantage of the host's shared resource platform used to execute the VDs. In our approach, the provider reserves a part of the expected resource requirements according to the adopted overbooking degree. The overbooking degree is defined as the probability of *not* being able to satisfy a user's request. The host's resource scheduler assures that a VD can always consume *at least* the reserved resources. Non-consumed resources are collected in the host's resource pool. VDs requesting more resources than reserved can receive additional resources from the resource pool. The figure shows that when the overbooking degree increases, also the utilization of the host increases. In that case, less resources are reserved and hence the probability of SLA violations increases.

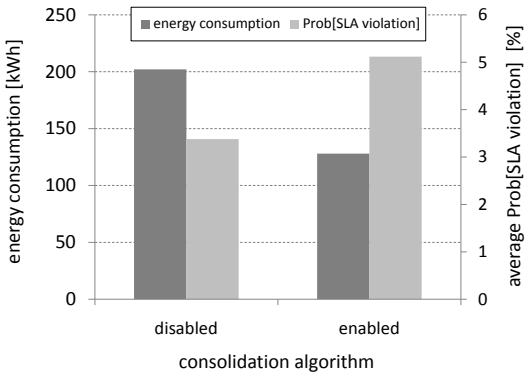
VDs with different profiles or SLA contracts can be assigned different overbooking degrees. As emphasized before, the user experience is not only determined by the resource allocation for her VD, but also by the audiovisual quality and the interaction delay with the application. To globally optimize the user experience and resource allocation, future research should be devoted to couple the resource allocation strategy with the thin client protocol settings in a global framework.

B. Energy cost

To achieve a green cloud-based desktop service, a consolidation algorithm should be implemented to adapt the online host pool to the current system load. The consolidation algorithm has to predict the (near) future system load in order to determine the required number of hosts. The time between two iterations of the consolidation algorithm is called the *time window*. During a time window, monitoring information is collected. Based on this information and the assumption that the system load during the next time window will vary in a



(a) Prediction of the system load in the next time window



(b) In the simulation, a daily-cycle of arrivals of two types of users are considered in a ratio of respectively three to one: *normal* users and *heavy* users with an average resource request distribution of respectively $N(10, 3.5)$ and $N(25, 5)$.

Fig. 4: A consolidation algorithm aims at reducing the energy consumption of servers in the cloud by adapting the number of online servers to the system load. The cost of the energy saving is a small increase in SLA violations.

similar way, the system load is predicted by means of linear extrapolation (see Fig. 4(a)).

When additional hosts are required, they are simply put online. When there are redundant hosts, more elaboration is required to decide which hosts should be put offline. Idle hosts are of course the best choice to put offline since no VDs have to be relocated before the hosts can be put offline. In case there are not enough idle hosts to put offline, the hosts are sorted by ascending amount of VDs. To minimize the number of relocations, the algorithm tries to relocate the VDs from the hosts in the order of the sorted list. When not all VDs on a host can be relocated to other hosts, it makes no sense to relocate any of the VDs from that host and the algorithm should continue with the next host from the list, until sufficient hosts are put offline, or until no hosts are remaining in the list.

When the real system load appears to be higher than expected, the monitoring framework notices this unfavorable situation and requests the cloud management component to take appropriate actions.

The simulation results in Fig. 4(b) from a scenario with realistic user behaviour (i.e. a daily cycle of user requests according to the Lublin model [14]) show that there is a large potential to save energy at the cost of a small increase in SLA violations. In this scenario, up to 36.6% energy can be saved at the cost of an additional 1.7% SLA violations.

IV. Conclusion

Accessing a desktop in the cloud by means of a thin client protocol enables accessing any application from any device and any location. In this paper, an overview of challenges and approaches is given to offer efficient desktop services in the cloud.

Existing optimizations of thin client protocols and desktop services each focus on a specific part of the user experience. Currently, the user experience of thin client based virtual desktops can only be quantified offline by means of the slow-motion benchmarking technique [12]. There is a clear need for a novel, objective metric representing the global user experience and online measurement methodologies. Future research should be devoted to integrate relevant thin client protocol optimizations with resource allocation strategies to achieve the best user experience. To further increase the user experience, the cloud management algorithms presented in this paper should be extended to operate on interconnected data centers, e.g., by relocating virtual desktops from overloaded to less loaded data centers.

REFERENCES

[1] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, 2009.

[2] A. Lai and J. Nieh, "On the performance of wide-area thin-client computing," *ACM Transactions on Computer Systems*, vol. 24, no. 2, pp. 175–209, 2006.

[3] R. Schwarzkopf, M. Schmidt, N. Fallenbeck, and B. Freisleben, "Multi-Layered Virtual Machines for Security Updates in Grid Environments," in *proceedings of 35th EUROMICRO Conference on Internet technologies, quality of service and applications*, pp. 563–570, 2009.

[4] N. Tolia, D. G. Andersen, and M. Satyanarayanan, "Quantifying Interactive User Experience on Thin Clients," *Computer*, vol. 39, no. 3, pp. 46–52, 2006.

[5] F. Travostino, P. Daspit, L. Gommans, C. Jog, C. de Laat, J. Mambretti, I. Monga, B. van Oudenaarde, S. Raghunath, and P. Y. Wang, "Seamless live migration of virtual machines over the MAN/WAN," *Future Generation Computer Systems*, vol. 22, no. 8, pp. 901–907, 2006.

[6] P. Simoens, P. Praet, B. Vankeirsbilck, J. De Wachter, L. Deboosere, F. De Turck, B. Dhoedt, and P. Demeester, "Design and implementation of a hybrid remote display protocol to optimize multimedia experience on thin client devices," in *proceedings of the Australasian Telecommunications Networks and Applications Conference*, 2008.

[7] B. Vankeirsbilck, P. Simoens, J. De Wachter, L. Deboosere, F. De Turck, B. Dhoedt, and P. Demeester, "Bandwidth optimization for mobile thin client computing through graphical update caching," in *proceedings of the Australasian Telecommunications Networks and Applications Conference*, 2008.

[8] P. Simoens, F. De Turck, B. Dhoedt, and P. Demeester, "Remote display solutions for mobile cloud computing," in *Computer*, vol. 44, no. 8, pp. 46–53, 2011.

[9] A. Quiroz, H. Kim, M. Parashar, N. Gnanasambandam and N. Sharma, "Towards autonomic workload provisioning for enterprise Grids and clouds," in *proceedings of IEEE/ACM International Conference on Grid Computing*, pp. 50–57, 2009.

[10] Y. Lu, S. Li, and H. Shen, "Virtualized Screen: A Third Element for Cloud-Mobile Convergence", in *IEEE Multimedia*, vol. 18, no. 2, pp. 4–11, 2011.

[11] Citrix Inc., "XenDesktop Planning Guide - Hosted VM-Based Resource Allocation," white paper (CTX12277), 2010.

[12] A. Berryman, P. Calyam, M. Honigford, A. Lai, "VDBench: A Benchmarking Toolkit for Thin-client based Virtual Desktop Environments," in *proceedings of the 2nd IEEE International Conference on Cloud Computing Technology and Science*, pp. 480–487, 2010.

[13] B. Urgaonkar, P. Shenoy, and T. Roscoe, "Resource overbooking and application profiling in a shared internet hosting platform," *ACM Transactions on Internet Technology*, vol. 9, no. 1, pp. 1–45, 2009.

[14] U. Lublin and D. G. Feitelson, "The workload on parallel supercomputers: modeling the characteristics of rigid jobs," *Journal of Parallel and Distributed Computing*, vol. 63, no. 11, pp. 1105–1122, 2003.