supercomputing manuscript No.
(will be inserted by the editor)

Efficient Resource Management for Virtual Desktop Cloud Computing

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

the date of receipt and acceptance should be inserted later

Abstract In virtual desktop cloud computing, user applications are executed in virtual desktops on remote servers. This offers great advantages in terms of usability and resource utilization; however, handling a large amount of clients in the most efficient manner poses important challenges. Especially deciding how many clients to handle on one server, and where to execute the user applications at each time is important. Assigning too many users to one server leads to customer dissatisfaction, while assigning too little leads to higher investments costs. We study different aspects to optimize the resource usage and customer satisfaction. The results of the paper indicate that the resource utilization can increase with 29% by applying the proposed optimizations. Up to 36.6% energy can be saved when the size of the online server pool is adapted to the system load by putting redundant hosts into sleep mode.

Keywords cloud computing · resource overbooking · resource allocation · resource reallocation · consolidation

1 Introduction

In virtual desktop computing, a major part of computation and storage components are shifted from the client device (e.g. desktop PC, laptop, PDA or smartphone) to the network. User applications are executed in a virtual desktop (VD) on a remote server and the client device only deals with user interaction and the presentation of the audiovisual output.

The concept of virtual desktop cloud computing is very attractive for several reasons, for example, lower client hardware investments are required, the

Ghent University - IBBT
Gaston Crommenlaan 8, bus 201, B-9050 Ghent, Belgium.
E-mail: Lien.Deboosere@intec.ugent.be
Tel: +3293314938
Fax: +3293314899

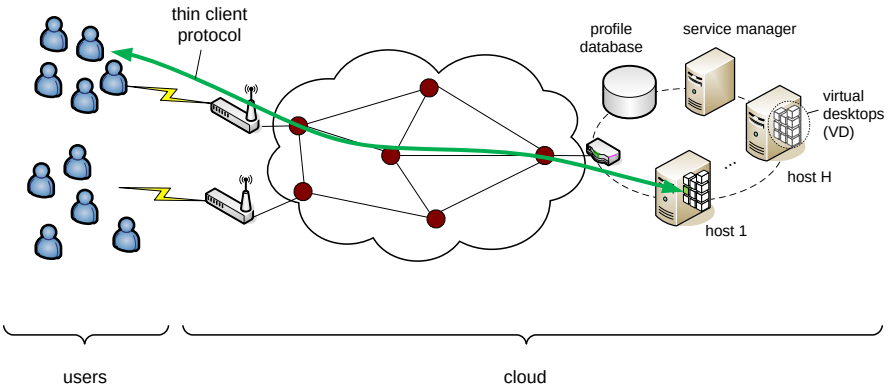


Fig. 1 System architecture to support remote desktops as a cloud service

end-user is no longer bothered with regular updates and often difficult installation and configuration of applications or anti-virus software, lower IT management costs for companies thanks to central management of desktops and applications. Furthermore, since the processing power of servers in the network is used, virtual desktop cloud computing enables access to advanced applications (e.g., computer-aided design (CAD) applications) from any device, for example from a tablet PC. Mobile users would no longer need to use restricted mobile versions of their applications.

Current virtual desktop computing deployments are typically operational in corporate local area network (LAN) environments, which are highly controlled environments offering fixed and stable bandwidth availability to a relative small, well-known user base. Extending virtual desktop computing to wide area network (WAN) environments, which comprise a large, geographically distributed customer base, where users are potentially connected through unreliable wireless network connections, involves a number of novel challenges. Strategies are needed to improve resource utilization and/or customer satisfaction in WAN environments in the most efficient manner.

Cloud computing [1] is an enabler for this kind of service. Unlike most current cloud services, the applications are not accessed through a web browser (e.g. Google Apps Cloud Service[2]), but through a thin client protocol (e.g., Microsoft Remote Desktop Protocol (RDP) [3] or Virtual Network Computing (VNC) [4]). This way, legacy applications must not be rebuilt to be offered by the envisioned service.

In Figure 1, the system architecture to support remote desktops as a cloud service is illustrated. When a user connects to the cloud service, the provider's service manager handles authentication, authorization and accounting (AAA) related functions and when access is granted, the user's virtual desktop is allocated to a host selected based on the available user profile information. Therefore, we assume that the cloud service provider has a database with user profiles, derived from monitoring information, describing the resource require-

ments of the users' virtual desktops. Finally, a thin client protocol session is started between the user and his virtual desktop.

For the envisioned service, it is extremely important to provide crisp interactivity of the service to all clients at all times, which is not the case in traditional supercomputing topics such as job/batch processing where it is possible to postpone a job until resources are available. The client's virtual desktop needs sufficient resources immediately to ensure quick responses; there is no possibility to delay the execution of a virtual desktop. The interactivity of the envisioned services brings important challenges not yet covered by existing algorithms. The focus in this paper is on the design of optimization algorithms for maximizing the quality and profits of the cloud service both for the customers and the provider. To evaluate the performance of the algorithms presented in this paper, an objective metric to represent the quality experienced by the user is needed. In the context of this paper, the metric is the probability that the user's virtual desktop requires more resources than it receives from the host, which we will refer to as the probability on *SLA violations*.

Note that an SLA violation in this context does not represent an SLA contract violation. In practice, an SLA contract will most probably contain a certain number of resources that is promised to the customer and/or a probabilistic guarantee that this number of resources is available for the customer. Only when this guarantee is not met, an SLA contract violation occurs and a potential penalty has to be paid to the customer.

The contributions of this paper are: *(i)* a resource overbooking technique, *(ii)* a resource allocation algorithm, *(iii)* a resource reallocation algorithm and *(iv)* a consolidation algorithm to reduce the energy consumption of the hosts. The remainder of this paper is structured as follows. Related work is discussed in section 2. The parameters of the model of the envisioned cloud service are described in section 3, together with a description of the simulation environment used to evaluate the performance of the proposed optimizations. Before discussing the optimization algorithms in detail, a resource overbooking technique to increase the average resource utilization of the system is introduced and evaluated in section 4. Next, the optimization algorithms, i.e. a resource allocation algorithm, a resource reallocation algorithm and a consolidation algorithm, are briefly discussed and evaluated respectively in section 5, 6 and 7. Finally, conclusions are drawn in section 8.

2 Related Work

2.1 Resource overbooking

Resource overbooking is a technique that can establish an increase of the average utilization of hosts in a data center by reserving less resources than required in worst case; at the cost of a larger probability that a virtual desktop does not receive all requested resources. Since more virtual desktops can be

allocated to a host, the cost for the service provider related to investment in hardware equipment, server maintenance cost and energy cost can be reduced.

The acquisition of the resource requirements of a virtual desktop, i.e. the user profile information, is an important aspect for resource overbooking but this is not the main topic of the paper; hence we assume profiling information is available in this paper. One approach to implement this functionality however, is to cluster the resource requirements of VDs offline into a finite number of profiles. At subscription time, a user could be assigned one of those predefined profiles. An online clustering algorithm such as the decentralized clustering algorithm presented by Quiroz et al. [5] could be used to map the current resource requirements (retrieved from monitoring information) 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.

Resource overbooking techniques have been studied in various contexts, e.g., in the field of video-on-demand servers [6] and ATM networks [7]. The importance of resource overbooking and application profiling in a shared internet hosting platform is demonstrated in [8]. The target is to maximize the revenue of the resource provider. The virtual machines never receive more resources than agreed on in the SLA because this does not increase the revenue for the resource provider. An important difference with our vision, is that in this paper, the resources that a virtual desktop can consume are not limited to the reserved resources. This can improve the user satisfaction when there are enough free resources available.

2.2 Resource allocation

Resource allocation algorithms for static workloads of web applications in a virtualized service hosting platform are presented in [9]. The metric used in these algorithms is the *yield* which is defined as the ratio between the resource fraction allocated and the maximum resource fraction potentially used. For example, if the service can be allocated maximum 60% of the host's CPU and the service consumes 30% of the CPU than the yield is 30%/60%=0.5. The algorithms assume static workloads, i.e., workloads that do not change throughout time. It is addressed in [9] that this assumption does not hold in practice and it is suggested to run the algorithms periodically to recompute the resource allocations. However, the new allocation scheme could differ a lot from the previous allocation scheme, leading to an enormous amount of reallocations which is not recommended in practice.

In [10], a framework is presented to offer remote desktop sessions in utility or service grids. When a request to launch a remote desktop session arrives, the *site admission control* system checks which servers have enough free resources to host the session. At runtime, a *session admission control* system checks if there are enough resources on the host to execute an additional application inside the remote desktop session. If not enough resources are available, the

application is put in a queue and can only start when there are enough resources on the host. Instead of waiting until resources become available on the current host, we propose to start the application immediately and to reallocate the remote desktop session to another host which does has enough resources.

2.3 Resource reallocation

In the context of object load balancing, the impact of monitoring the objects to collect data and rebalancing the problem cannot be neglected. In [11], a flexible method to invoke load balancers has been presented to achieve adaptivity and to avoid interferences from the load balancer at fixed intervals, which may diminish the performance. The target of their load balancer is to reduce the execution time of a job by taking into account the localization of the objects. The difference with our system is that we cannot reduce the execution time of a virtual desktop because it is an interactive user session.

Charm++ is a framework for parallel computing using object-oriented programming language [12]. Based on object migration, efficient dynamic load balancing is supported. The framework also deals with external factors that cause load imbalance. However, these load balancing strategies are mainly based on the dependency and locality of objects and the methods, which is fundamentally different from the independent execution of the virtual desktops.

In [13], a system is presented that monitors virtual machines, detects overloaded servers and initiates necessary migrations. Two strategies for migrations or reallocations are discussed: a black-box approach which is completely independent of the OS and applications, and a gray-box approach which exploits OS and application level statistics. In the black-box approach, the resource requirements of the virtual machine are observed externally. In case of an overloaded virtual machine, it is hard to decide how many resources should be allocated to the virtual machine to solve the overloaded state. In the gray-box approach, monitoring details on application level (e.g. application response time) are available which allows a more accurate estimation of the resource needs of the virtual machine. To avoid needless migrations due to small transient spikes, migrations are only initiated when thresholds or SLAs are exceeded for a sustained time. It is suggested that this sustained time approximates the time period of about three migrations before migrations are initiated. While this approach focuses on migrating VMs due to overloaded servers, we also focus on migrating VMs to reduce the energy consumption in the data center.

2.4 Energy-efficiency

Energy-efficient resource management in cloud data centers is very important [14–18], not only to reduce the cost of the energy consumption of the data centers, but also to reduce the carbon dioxide footprint of a data center.

This includes increasing the power efficiency at the architectural level of microprocessor design with technologies like asymmetric multicores [19] and the ability to scale the performance of the CPU according to the needs and as such save energy when possible (e.g., Intel SpeedStep).

In [16], heuristics are presented for dynamic reallocation of VMs with workloads originating from web applications and online services. The main idea of the policies is to set upper and lower utilization thresholds and keep the total CPU utilization of a node between these bounds. How adaptive upper and lower bounds can further improve the energy-efficiency of the cloud data center is investigated in [20]. When the upper bound is exceeded, VMs are reallocated for load balancing and when the utilization of a host drops below the lower bound, VMs are reallocated for consolidation. Nevertheless, a significant part of the consolidation algorithm, namely the decision when hosts should be powered on again to cope with increasing load, is missing in [16,20]. In section 7, we show that our consolidation algorithm can save even more energy than the approach described in [20].

In [21], a trace-based workload placement controller uses historical information to periodically and proactively reassign workloads to servers subject to their QoS objectives. A reactive migration controller is introduced that detects server overload and underload conditions. This controller initiates the migration of workloads when appropriate. In our work, we detect even more opportunities to shutdown hosts by combining the virtual desktops of hosts with low load without waiting for the total system load to drop below a predefined threshold.

3 System model

The different parameters of the system model are introduced in this section. For maintaining the overview, table 1 summarizes the abbreviations and parameters used through this paper. After describing the model of the virtual desktop cloud service, details on the simulation environment used to evaluate the performance of the algorithms are discussed and the scenario of the simulations is described.

3.1 Model description

We assume that there are M hosts in the data center and N users are subscribed to the virtual desktop cloud service. All hosts in the cloud are considered identical and have limited processing power, which is modeled by the number of FLOPS (Floating Point Operations Per Second) F a host can execute.

We assume that two user types can be distinguished: *normal* users and *heavy* users. Based on the planning guide described in [22], a single host is able to execute on average 10 *normal* virtual desktops or 4 *heavy* virtual

Symbol	Description
B_l	Bandwidth of network link l
F	Number of FLOPS available in a host
H_i	Set of virtual desktops assigned to host i
M	Number of hosts
m_j	Memory assigned to virtual desktop of user j
N	Number of users
n_i	Number of virtual desktops on host i
$P_{dyn}(u)$	Dynamic power consumption of a host as a function of the utilization u of the host
P_{stat}	Static power consumption of a host
R	Minimum amount of additional resources that a VD can receive from RP
r_j	Number of FLOPS reserved for user j
$req_j(t)$	Number of FLOPS requested by user j on time t
RP	Resource pool
t_{online}	Time to have an additional host in online mode
$t_{offline}$	Time to put a host in offline mode
T_s	Parameter of the exponential distribution of virtual desktop service times
VD_j	Virtual Desktop of user j
μ_j	Average number of FLOPS requested by user j taken from: for *normal* users the folded normal distribution $N(10000, 3500)$ for *heavy* users the folded normal distribution $N(25000, 5000)$
σ_j	Standard deviation of the number of FLOPS requested by user j taken from: for *normal* users the folded normal distribution $N(3500, (2/3 \times 3500)^2)$ for *heavy* users the folded normal distribution $N(5000, (2/3 \times 5000)^2)$

Table 1 An overview of the parameters and abbreviations used in the model description and in the algorithms

desktops. The ratio of heavy users to normal users is assumed to be 25%. We assume that every client has been assigned a personal profile based on historical data. The average memory consumption for a normal user's virtual desktop varies between 768MB and 4GB, while for a heavy user's virtual desktop it varies between 2GB and 4GB [22]. We can assume that current server equipment has at least 16GB of memory installed; thus the memory consumption is not considered to be a bottleneck. Therefore, the user profile only contains a FLOPS requirement distribution. Based on the observation that the FLOPS requirement of a virtual desktop depends on many factors such as several active applications and events in the operating system, we assume that the central limit theorem [23] can be applied and the FLOPS requirement distribution for a virtual desktop VD_j can be approximated by a normal distribution $N(\mu_j, \sigma_j^2)$. The service time of user's virtual desktop is assumed to be exponentially distributed with parameter T_s.

The following parameters are introduced concerning the power consumption of a host. The power consumption model of a host is assumed to consist of a static power consumption component P_{stat}, i.e. the power consumption of a host in idle mode, and a dynamic power consumption $P_{dyn}(u)$ which is assumed to be a linear function of the utilization u of the host [24]. In idle mode, the power consumption of a host is on average 70% of the power P_{max}

consumed when the host is fully utilized [20]. The power consumption of a host can thus be defined as

$$\begin{aligned} P &= P_{stat} + P_{dyn}(u) \\ &= 0.7 \times P_{max} + 0.3 \times P_{max} \times u \\ &= P_{max} \times (0.7 + 0.3 \times u) \end{aligned}$$

The time needed to have an additional online host, either wakening up from sleep mode or booting from standby mode, is represented by t_{online}. The time needed to put a host in offline state, either in sleep mode or in standby mode, is represented by $t_{offline}$. Experiments learned that the power consumption of a host in sleep mode is only slightly higher compared to a host in standby mode. Therefore, the power consumption of a host in sleep mode is neglected.

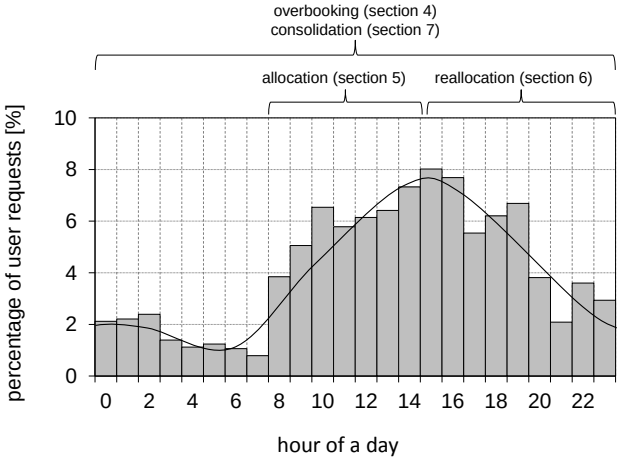


Fig. 2 Percentage of arrivals during a day according to the Lublin model [25]. The regions with the highest impact of the optimization algorithms presented in this paper are indicated in the graph.

The arrival process is modeled by a Lublin model [25] with interactive job types. The Lublin model takes the daily cycle of arrivals into account as can be seen in Figure 2. Each optimization algorithm proposed in this article tackles a specific part of the complete scenario with arrivals according to the Lublin model. In Figure 2, for each algorithm, the part of the scenario with the strongest impact is illustrated. During the morning, a lot of users want to start their virtual desktop and the system load transits from low load to high load. The *allocation* algorithm selects for every virtual desktop an appropriate host from the active host group and reserves an amount of resources based on the user's profile and on the applied overbooking degree (see section 4). When convenient, the *consolidation* algorithm can decide to wake up additional hosts

to serve and satisfy more clients. To the end of the day, the amount of active users decreases. When users depart, the system becomes in a state with some hosts still heavily loaded while other hosts are (almost) idle. To increase the quality experienced by the users, the *reallocation* algorithm rebalances the load of the virtual desktop by moving load from high loaded hosts to low loaded hosts. Finally, the consolidation algorithm comes again into play to regroup the remaining virtual desktops on a smaller amount of hosts and put redundant hosts into sleep mode to save energy.

3.2 Simulation environment

Our simulation environment is an extension of the CloudSim 2.0 toolkit [26]. CloudSim is a discrete event simulator for modeling and simulating cloud computing infrastructures and services. In the context of this paper, virtual desktops are migrated regularly. CloudSim contains a realistic model to migrate virtual desktops from one host to another. The duration of a migration in CloudSim takes as long as it needs to migrate the memory assigned to the virtual desktop, m_j, over the network links with bandwidth B_l between the original and the target host. In our simulations, all links are 1 Gbps network links.

In the CloudSim toolkit, a simple VM scheduler is implemented to distribute the available resources of a host among the involved VMs. The resources of a host are simply distributed in a first-come, first-served manner. In other words, the first VM on the host's list receives the requested resources, then the second VM is served and so on. In this paper, an advanced scheduler is proposed to distribute the host's resources. For each VM, a number of FLOPS is *reserved* based on the user profile and the applied overbooking technique. It is guaranteed that the VM always has at least this amount of FLOPS at its disposal. When the VM consumes less FLOPS than reserved, the remaining FLOPS are collected in a resource pool (RP) as shown in Figure 3. The resources available in the resource pool can be shared in a second phase among virtual desktops requesting more FLOPS than reserved.

When enough resources are available to give all virtual desktops the resources they request, there is no issue. However, when not all resource requests can be met by exhausting the resource pool, a choice has to be made on how to distribute the resources among the remaining virtual desktops. Both for the users and the service provider, it is most profitable to minimize the amount of virtual desktops encountering an SLA violation in such case. Therefore, the following policy is applied (a flowchart of the resource sharing policy can be found in Figure 4).

The minimum amount of additional resources that every involved virtual desktop could get, R, is calculated as

$$R = \frac{\#\text{resources in RP}}{\#\text{additional resources requested by remaining VMs}}.$$

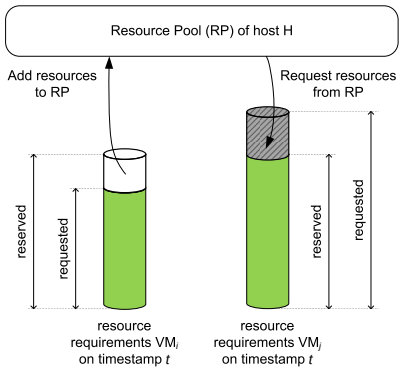


Fig. 3 Non consumed reserved resources are collected in the host's resource pool. These resources can be shared among virtual desktops requesting more resources than reserved.

To distribute the resources among the virtual desktops, the virtual desktops are sorted by ascending amount of additional requested resources.

When the first virtual desktop in the list requests less additional resources than R, it receives all the resources it needs and this VD does not encounter an SLA violation. The remaining resources are again collected in the resource pool. The value of R is updated. This approach is continued, until the first virtual desktop in the row requests more additional resources than R. At that point, none of the remaining virtual desktops can receive all of its requested resources. Instead, all remaining virtual desktops are given the same amount of additional resources, R, and thus encounter an SLA violation. This approach results in fewer SLA violations compared to adopting a simple scheduler as will be shown in the simulation results in the next section.

4 Resource overbooking

The target of the resource overbooking technique presented in this section is to increase the system utilization with little impact on the user experience. This can be achieved by reserving a certain amount of resources for the virtual desktop based on its profile. First, the term *overbooking degree* is defined to depict how many resources should be reserved based on the profile of a virtual desktop.The impact of resource overbooking on the user satisfaction is discussed and a realistic scenario is evaluated by means of simulations.

4.1 Definition overbooking degree

The overbooking degree is defined as the percentage of resources that are *not* reserved for a virtual desktop. For example, when the 90-*th* percentile of the resource consumption distribution from the virtual desktop profile is reserved, the overbooking degree is 10%.

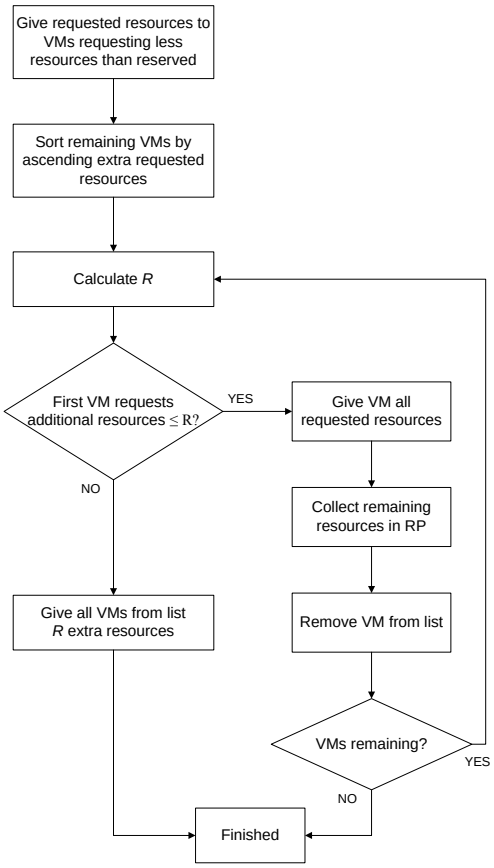


Fig. 4 Flowchart of the advanced resource sharing policy: an extension of the CloudSim simple resource sharing policy

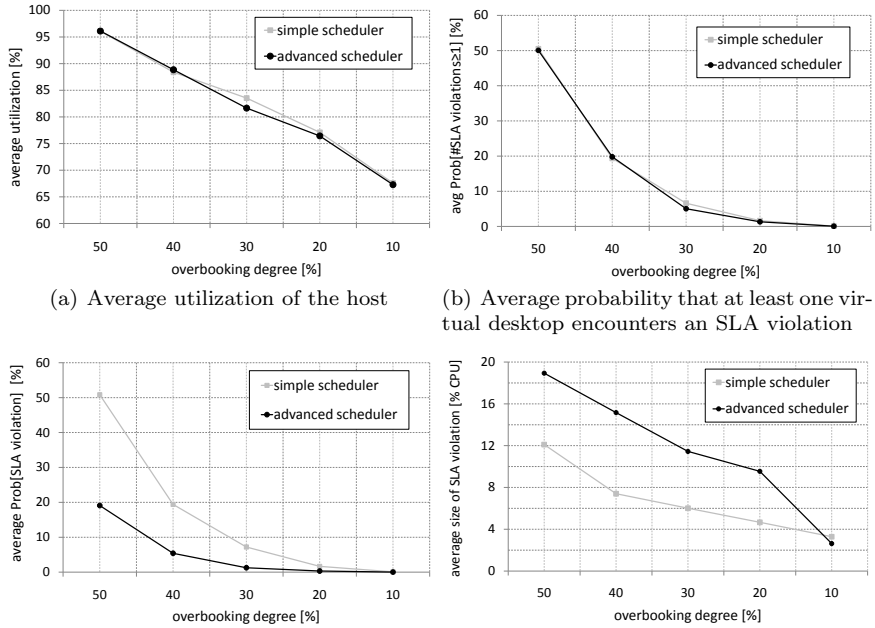
4.2 Impact of overbooking on user satisfaction

For the customer, it is important that the negative impact of overbooking on the amount of SLA violations experienced is acceptable. Applying a certain overbooking degree and the advanced resource scheduler essentially means that a virtual desktop *never* encounters an SLA violation when the requested resources do not exceed the reserved resources. When the virtual desktop requests more resources than reserved, it *can* encounter an SLA violation depending on the resource requests of other virtual desktops executed on the same host.

The impact of resource overbooking on the user experience can be represented by the number of SLA violations experienced by the user's virtual desktop. When a host does not have sufficient resources available to fulfill all requests, at least one virtual desktop will experience an SLA violation. Deducing the exact amount of virtual desktops experiencing an SLA violation - when

the advanced resource scheduler is applied - is complicated. An analytical deduction for a simplified use case can be found in appendix A. However, when the number of VDs on a host increases, it becomes hard to analytically deduce the probability that a virtual desktop experiences SLA violations. Therefore, simulations are used to further discuss the impact of overbooking in a realistic scenario.

In the simulations, only one host is considered, which is fully reserved. It does not really matter how the host got fully loaded. Therefore, we do not take into account a specific arrival process but only assure that the host is fully reserved with VDs from *normal* users. Every timestamp, a virtual desktop requests a number of FLOPS picked from the associated FLOPS requirement distribution of the user profile of this virtual desktop. This is repeated until the measurement of the averages does not significantly change between two timestamps. The reader is referred to section 3 and Table 1 where the parameters of VDs belonging to normal users were described. The main results of the average of 15 simulations can be found in Figure 5.



(a) Average utilization of the host

(b) Average probability that at least one virtual desktop encounters an SLA violation

(c) Average probability that a virtual desktop encounters an SLA violation

(d) Average size of an SLA violation expressed in relative percentage of CPU load not obtained

Fig. 5 Simulation results to evaluate the performance of the overbooking strategy for both the simple scheduler delivered with CloudSim and the advanced scheduler introduced in section 3.2.

In Figure 5(a), the average utilization of the host is shown for different overbooking degrees. For smaller overbooking degrees, less virtual desktops are accepted for execution on the same host and hence the average utilization decreases from 96% to 67.3% when the overbooking degree decreases from 50% to 10%. Because the same virtual desktops are accepted for execution on the host, independent of the applied scheduler type, there is no significant difference in average utilization of the host. The graph depicted in Figure 5(b) shows that the average probability at least one of the virtual desktops on the host encounters an SLA decreases exponentially from 50% to 0.07% when the applied overbooking degree decreases from 50% to 10%. Again, there is no significant difference between the two studied schedulers for the same reason as above: the same virtual desktops are executed on the host, thus the same probability exists that the sum of the resource requirements of those virtual desktops exceeds the available resources of the host and at least one of the virtual desktops encounters an SLA violation.

Nevertheless, in Figure 5(c) an important difference between the simple and advanced scheduler can be noticed. By adopting the advanced scheduler and thus assuring that every virtual desktop always receives at least the reserved amount of resources, a virtual desktop encounters a significantly lower amount of SLA violations. For example, when an overbooking degree of 50% is applied, the probability that a virtual desktop encounters an SLA violation can be decreased from 50% to 19.1% by adopting the advanced scheduler instead of the simple scheduler. On the other hand, as can be seen in Figure 5(d), the size of the SLA violation (i.e., the ratio of unallocated resources to the total requested resources) is larger when the advanced scheduler is applied: 18.9% compared to 12.1% for an overbooking degree of 50%. Larger SLA violations mean lower quality experienced by the user and could lead to less revenues for the service provider depending on the agreement between the user and the service provider.

For a user, high QoS means few SLA violations which can be reached by adopting a small overbooking degree. For the service provider, small overbooking degrees are less attractive, because they lead to an overdimensioning of the equipment. To accept the same number of VDs, the service provider would have to invest a lot more in hardware equipment and pay a lot more for energy consumption. Therefore, it is likely that a service provider uses higher overbooking degrees and adopt an SLA with the users to guarantee a minimum service level. Our simulations can be used by a service provider to set a target on the adopted resource overbooking degree depending on the maximum probability on SLA violations that a service provider can tolerate.

5 Allocating virtual desktops

When there N online hosts available in the system, it has to be decided on which host the virtual desktop of an arriving user request should be allocated.

After describing the allocation algorithm, its performance is evaluated with our simulator.

5.1 Allocation algorithm

The allocation algorithm is a classical bin-packing problem [27] which is known to be NP-hard. Therefore, a cost-based heuristic is proposed to select an appropriate host when a user request to start VD_j arrives. The allocation algorithm *(i)* calculates for every host i the associated cost c_i and *(ii)* selects the host with the best cost.

We propose a cost that takes into account both the objectives of the customers (i.e., few SLA violations) and the objectives of the service provider (i.e., serve many customers) and is calculated as:

$$\begin{aligned} c_i = \alpha \times \text{Prob}[\#\text{SLA violations on host } i \geq 1 \mid VD_j \in H_i\ ] \\ +\beta \times \text{Prob}[\text{next user rejected by host } i \mid VD_j \in H_i] \end{aligned} \tag{1}$$

with $\alpha, \beta \geq 0$.

The host for which the cost c_i is minimal, is selected to execute the user's virtual desktop.

The first term in (1) can be calculated as the probability that the sum of the resource requirements of virtual desktops executed on the host is larger than the total amount of resources available on the host:

$$\begin{aligned} \text{Prob}[\#\text{SLA violation(s) on host } i \geq 1] &= \text{Prob}\left[\sum_{j=0}^{n_i} req_j \geq F\right] \\ &= \int_F^{\infty} N(\mu_h, \sigma_h^2) \end{aligned}$$

$$\text{with } \mu_h = \sum_{j=0}^{n_i} \mu_j \text{ and } \sigma_h = \sqrt{\sum_{j=0}^{n_i} \sigma_j^2}$$

The second term in (1) actually depends on several parameters: on the arrival rate, on the amount of resources the users request and on the current distribution of virtual desktops among the available hosts. The first two parameters are not controlled by the allocation algorithm. Therefore, we propose to distribute the resources according to the best-fit strategy. This heuristic selects the host for which the amount of available resources is closest to the requested amount of resources. This way, virtual desktops are gathered as much as possible on a single host, leaving other hosts more or less idle. In case the system is not overloaded, the probability that at least one host can be found with enough unreserved resources to execute a user's virtual desktop is higher compared to, for example, a random allocation algorithm.

The probability that the next user is rejected can be calculated as the average amount of unreserved resources on the host after accepting the current user's virtual desktop. The cost c_i (equation (1)) becomes then:

$$c_i = \alpha \times \int_F^\infty N(\mu_h, \sigma_h^2) + \beta \times \frac{1}{F}\Big(F - \sum_{VD_j \in H_i} r_j\Big).$$

Note the trade-off between both objectives in the cost function: the first term in (5.1) aims to equally distribute virtual desktops to prevent overloaded hosts which would involve higher probabilities on SLA violations, and the second term in (5.1) aims to collect the virtual desktops as much as possible on the same host(s) to maximize the probability that a host can be found for the next arriving user request.

The cost-based allocation algorithm has to calculate the cost c_i for every host i and select the host for which c_i is minimal. Therefore, the complexity of the cost-based allocation algorithm is $O(\sum_{i=0}^{N} \#H_i)$ with $\#H_i$ the cardinality of the set H_i or in other words the number of VDs allocated to host i.

5.2 Simulation results

To evaluate the performance of the presented cost-based allocation algorithm, the following simulations have been conducted by studying the transition from a non utilized system to a high utilized system. An overbooking degree of 50% has been chosen. The performance of the allocation algorithm should not be measured when the system is fully loaded, because in that case, it does not matter how the system got in this state, it is simply overloaded and the performance would be approximately independent of the adopted allocation algorithm. Rather, it is interesting to evaluate the performance of the algorithms for a nearly overloaded system. Therefore, in the simulations, the transition from a not-utilized system to a system with an average utilization of 90% is studied. The allocation algorithm allocates VDs from both normal and heavy users, in a ratio of respectively 3 to 1, to a host selected out of 10 available hosts. The inter-arrival time is not important for the current performance evaluation because the evaluation of the probability on SLA violations of the final placement of the VDs is important.

The performance of the cost-based allocation algorithm for relevant ratios of α/β is compared with the performance of a random allocation algorithm. The average results of 15 simulations are shown in Figure 6.

The average utilization of the hosts is approximately independent of the allocation algorithm and reaches 88.5% in the simulations. However, the probability on SLA violations clearly differs depending on the adopted allocation algorithm as can be seen in Figure 6. For small ratios of $log(\alpha/\beta)$, the cost is dominated by the second term in (1), which means that the cost-based allocation algorithm tries to gather the virtual desktops as much as possible on the same host. In other words, the allocation algorithm first fills host 1 with virtual desktops, then host 2 and so on. This leads, already for a small amount

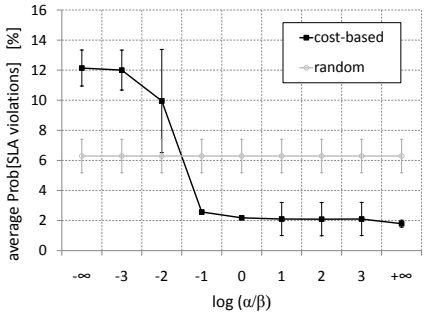


Fig. 6 Average probability that a virtual desktop experiences an SLA violation for different allocation algorithms and interesting ratios α/β

of active virtual desktops, to a system with overloaded hosts on which the probability that SLA violations occur is high. On the other hand, the virtual desktops of all users in the simulations were accepted for execution. When the first term of the cost function (1) becomes more and more important – in other words when α/β increases – the average probability that SLA violations occur decreases from 12% to 2%, at the cost of 0% to 0.6% user requests being rejected. When the random allocation algorithm is applied, the average probability SLA violations occur is 6.3% and the average blocking probability is 0.3%. For well-chosen ratios of α/β, the cost-based allocation algorithm can thus outperform the random allocation algorithm.

The service providers can set the values of α and β to a meaningful value, namely the cost associated to an SLA violation (α) and the cost associated to a rejected user (β). By tuning the ratio between these costs based on our simulations, the service providers can increase their revenues by minimizing the costs related to violating SLA agreements.

6 Reallocating virtual desktops

Virtual desktops can be reallocated during operation to optimize the quality experienced by the users. To implement this optimization, two problems need to be tackled. First, the reallocation algorithm itself has to be designed taking into account the cost of reallocating a virtual desktop. The reallocation algorithm has to decide which virtual desktops should be reallocated; the allocation algorithm presented in the previous section can be used to decide where to reallocate those virtual desktops. Secondly, it has to be defined when the reallocation algorithm should be activated.

6.1 Reallocation Algorithm

Reallocating a virtual desktop involves live migrating the virtual desktop from the original host to the target host. There is a cost associated to this migration

process [28]. The user can experience a small downtime depending on the kind of applications executed inside the virtual desktop [29]. The duration of the migration process depends on the memory used by the virtual desktop. It is thus important to reallocate as less virtual desktops as needed.

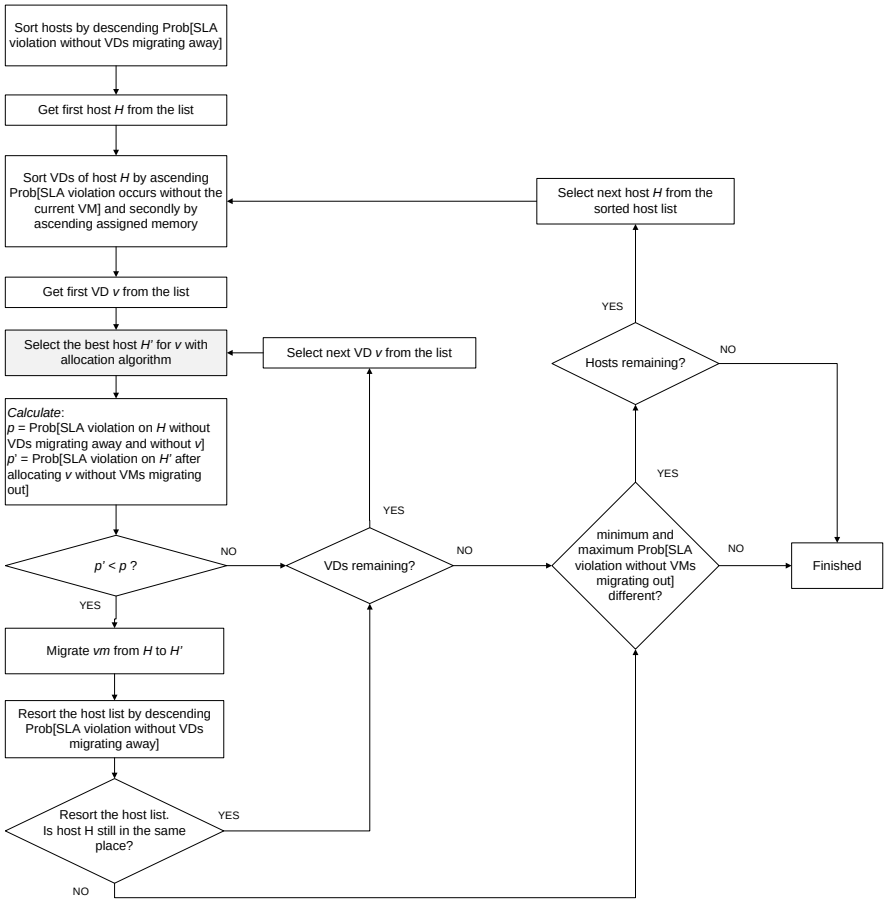


Fig. 7 Flowchart reallocation algorithm to reduce the Prob[SLA violation] on the hosts in the data center

A detailed flow-chart of the reallocation algorithm is presented in Figure 7.

By rebalancing virtual desktops among the available hosts, the probability on SLA violations can be reduced. In this algorithm, we do not adapt the number of FLOPS reserved for the virtual desktops, we only migrate them to hosts which has more free resources than the current host. This means that the target hosts have more free resources that can be shared among virtual desktops requesting more resources than reserved. This can result in even

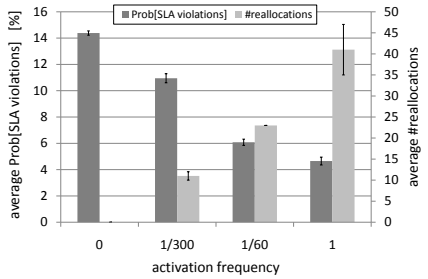
faster responses of the applications executed in the virtual desktop and hence a better user experience.

The target of the reallocation algorithm is to achieve a smaller probability on SLA violations in the data center with as less reallocations as possible. Therefore, the algorithm starts with sorting the hosts by descending probability of SLA violations on the host. The first host in the list, H, is the host with the highest probability on SLA violations. To reduce the load on this host, VDs should be moved to other hosts with as few reallocations as possible. Therefore, for every virtual desktop on host H, the probability on SLA violations without that virtual desktop is calculated. The virtual desktop that can reduce the load on the host the most when it would be moved to another host, is selected for reallocation. Next, the cost-based allocation algorithm introduced in the previous section is used to select a new host for this virtual desktop. Since the target is to minimize the probability on SLA violations, a high ratio for α/β is advised. It only makes sense to reallocate the virtual desktop, when the probability on SLA violations on the target host appears smaller than the probability on SLA violations on the current host. When comparing these probabilities, the virtual desktops migrating away from the hosts are not counted, because in the near future, these virtual desktops disappear from the hosts and will not cause any SLA violations on that host anymore. When it is decided not to migrate this virtual desktop, the next virtual desktop in the list is considered for reallocation. When it was decided to migrate the virtual desktop, the migration process is initiated and the sorted host list is updated. The algorithm continues until either all hosts reach the same probability on SLA violations or until no more reallocations are possible that can improve the quality of the service.

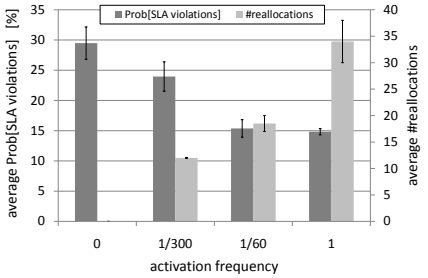
The complexity of the presented reallocation algorithm can be calculated as follows. We assume that the actual values for the probability on SLA violations with VDs migrating out are available in memory. The complexity of the algorithm used to sort an array is $O(n \times log(n))$ with n the size of the array to sort. Therefore, the complexity of the first step in the reallocation algorithm is $O(N \times log(N))$. The second step involves another sorting step which is repeated at most once for every host, thus the complexity of this step is $O(\sum_{i=0}^{N} \#H_i \times log(\#H_i))$. At most once for every VD allocated to a host, the allocation algorithm is executed, which has a complexity of $O(\sum_{i=0}^{N} \#H_i)$ as calculated in the previous section. So the complexity of this step is $O\left(\sum_{i=0}^{N} \#H_i \times \left(\sum_{i=0}^{N} \#H_i\right)\right)$. In the next step, the probability on SLA violations is recalculated for the original host i and the selected target host i'. The complexity for this step is $O(\sum_{i=0}^{N} \#H_i \times (\#H_i + \#H_{i'}))$. To resort the host list, only the position of two involved hosts must be checked, because only for the original and target host of the reallocated VD, the probability on SLA violations can be changed. Therefore, the complexity of this resorting is $O(\sum_{i=0}^{N} \#H_i \times N)$. The total complexity of the reallocation algorithm is the sum of the complexities calculated above and can be simplified to $O\left(N \times log(N) + \sum_{i=0}^{N} \left[\#H_i \times \left(\sum_{i=0}^{N} \#H_i\right) + \#H_i \times N\right]\right)$.

6.2 Simulation results

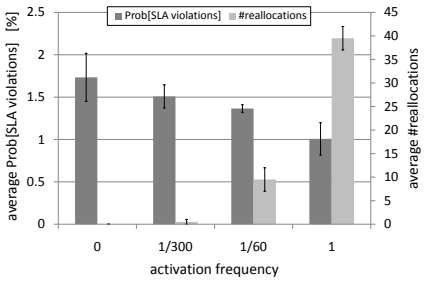
In the scenario of the simulations conducted in this section, the transition from a fully utilized system to a low utilized system is studied. The simulation starts with a fully reserved system of 10 hosts. The service times of the accepted virtual desktops are exponentially distributed with parameter $T_s = 600$. The results of the simulations for an overbooking degree of 50% and 60% are measured from the start of the simulation to a third of the total simulation time because the reallocation algorithm has the largest impact in that part of the scenario. Indeed, after a while the system is not overloaded anymore and thus the probability on SLA violations decreases automatically so the reallocation algorithm cannot further optimize the quality of the service.



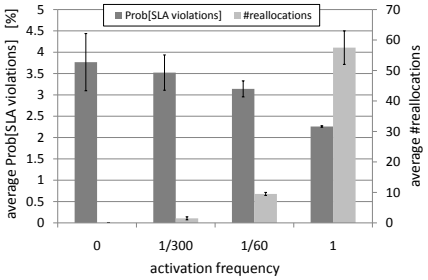
(a) Overbooking degree of 50% with only normal VDs allocated



(b) Overbooking degree of 60% with only normal VDs allocated



(c) Overbooking degree of 50% with 75% normal VDs and 25% heavy VDs allocated



(d) Overbooking degree of 60% with 75% normal VDs and 25% heavy VDs allocated

Fig. 8 Improvement of the quality of the service through reallocating virtual desktops once every 5 minutes, once every minute or every second

As stated in the introduction of this section, besides the reallocation algorithm itself, the activation frequency of the reallocation algorithm is evaluated. When the activation frequency of the reallocation algorithm increases, more opportunities are created to increase the quality of the service by reallocating virtual desktops. The results of the simulations can be found in Figure 8. By activating the reallocation algorithm every timestamp, a reduction in the probability on SLA violations from 14.4% to 4.65% and from 29.5% to 14.8%

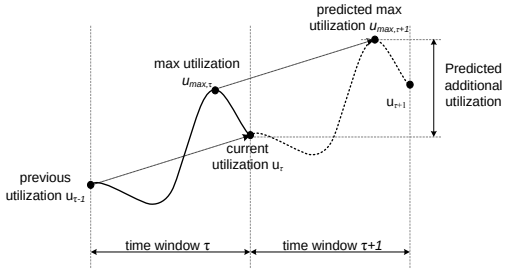


Fig. 9 Prediction of the load during time window $\tau + 1$ based on the variation of the load during the previous time window τ

can be reached for an overbooking degree of respectively 50% and 60% when only normal VDs are allocated (see Figures 8(a) and 8(b)). To achieve this reduction, respectively 34 and 41 reallocations are required. Because the probability on SLA violations is higher at the start of the simulation for higher overbooking degree, a larger reduction in the probability on SLA violations can be reached. The results when a mix of normal and heavy VDs is allocated can be found in Figures 8(c) and 8(d) for an overbooking degree of respectively 50% and 60%. The average probability on SLA violations is a lot smaller compared to the case where only normal VDs were allocated. The relative increase in performance is then also smaller compared to the case where only normal VDs were allocated.

Depending on the cost of migrations, the service provider can tune the frequency of activating this optimization algorithm.

7 Consolidating virtual desktops

The number of online hosts should be dynamically adapted to the system load to reduce the energy cost and carbon dioxide footprint of the virtual desktop cloud service, naturally with minimal impact on the user experience. In this section, a consolidation algorithm is presented and evaluated.

7.1 Consolidation algorithm

An important part of the consolidation algorithm, is to determine how the system load is varying over time. Based on a prediction of the system load, the consolidation algorithm has to decide whether *(i)* additional hosts are required to cope with an increasing system load, or *(ii)* redundant hosts can be put offline to save energy, or *(iii)* the current amount of hosts is sufficient.

The time between two iterations of the consolidation algorithm is called the *time window*. During a time window, monitoring information of the service is collected. Based on this information and the assumption that the system load during the next time window will vary in a similar way, the system load

is predicted by means of linear extrapolation. This is illustrated in Figure 9. The difference between the maximum utilization of time window τ ($u_{max,\tau}$) and the utilization at the start of that time window ($u_{\tau-1}$) determines the expected extra load in time window $\tau+1$. The expected maximum utilization for the next time window, $u_{max,\tau+1}$, can be calculated as:

$$u_{max,\tau+1} = u_{\tau} + (u_{max,\tau} - u_{\tau-1}).$$

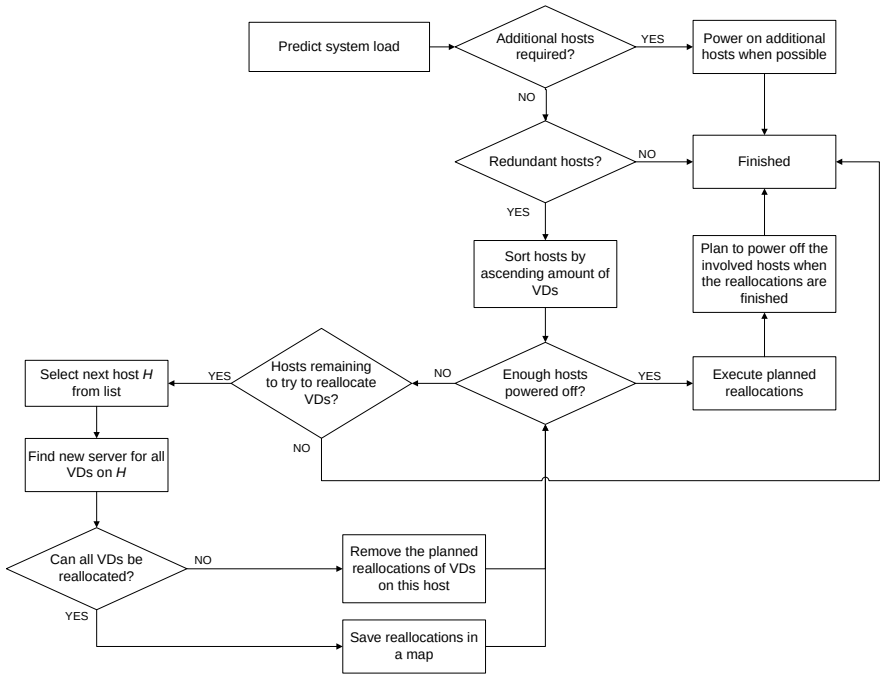


Fig. 10 Flowchart of the consolidation algorithm

Based on the predicted system utilization, the required amount of hosts is deduced by dividing the required FLOPS of the predicted system utilization by the amount of FLOPS available on a single host. The details of the consolidation algorithm are presented in the flowchart in Figure 10. When additional hosts are required to handle the predicted system utilization, the hosts are woken up by means of for example Wake-on-LAN or through a management interface available in the hosts. When no additional hosts are required nor redundant hosts can be put offline, the algorithm finishes. When the algorithm decides there are redundant hosts, more elaboration is required to decide which hosts should be put offline. When there are idle hosts, those are the best choice to put offline since no virtual desktops have to be reallocated before the hosts can be put offline. Otherwise, to minimize the amount of reallocations, the hosts are sorted by ascending amount of virtual desktops. The algorithm tries

to reallocate the virtual desktops from the hosts in the sorted list until enough hosts are put offline, or until no hosts are remaining in the list to try to reallocate its virtual desktops. When not all virtual desktops can be reallocated to other hosts, it makes no sense to reallocate any of the virtual desktops from that host and the algorithm should continue with the next host from the list.

Besides executing the consolidation algorithm on a regular basis, an optimization is implemented to react fast to a wrong prediction of the system utilization. When a user is rejected, an urgent request for an additional host is fired to the consolidation component, which will immediately try to wake up an additional host, without waiting for the next time window.

The complexity of the presented consolidation algorithm is deduced as follows. When additional hosts are required based on the predicted system load, the run time of the algorithm can be neglected compared to the complexity of the algorithm when redundant hosts should be put offline. In that case, the algorithm starts with sorting the host list by ascending amount of VDs. The complexity of the sorting algorithm is $O(N \times log(N))$. For every redundant host i, it has to be checked whether all VDs allocated to host i can be reallocated to another host by means of the allocation algorithm which has a complexity of $O(\sum_{i=0}^{N} \#H_i)$. The complexity of this step is thus $O(\sum_{i=0}^{N} \#H_i \times \sum_{i=0}^{N} \#H_i)$. The total complexity of the consolidation algorithm is then $O(N \times log(N) + \sum_{i=0}^{N} \#H_i \times \sum_{i=0}^{N} \#H_i)$.

7.2 Simulation results

To evaluate the performance of the consolidation algorithm, the complete scenario of user requests arriving according to the Lublin model as described in section 3.1 is used. This scenario takes into account the daily cycle of many user requests during the day and less user requests during the night. This is a realistic scenario which introduces opportunities to put hosts online/offline according to the system load. The allocation algorithm presented in section 5 is applied in the simulations with $log(\alpha/\beta) = 3$.

To evaluate the performance of the consolidation algorithm presented in this article, the results of the simulations are compared with the results of the *adaptive thresholds* algorithm proposed in [20]. In the adaptive thresholds algorithm, an adaptive upper and lower bound for the utilization of host is calculated. The aim is to keep the utilization of the hosts in the data center between these boundaries. When the utilization exceeds the upper bound, the algorithm tries to reallocate virtual desktops to other hosts as needed. When the utilization is below the lower bound, the algorithms tries to reallocate all virtual desktops of the host to be able to put the host offline. In [20], it is not described when to wake up hosts. In our implementation of the adaptive thresholds algorithm, we have decided to wake up hosts when a user is rejected.

The results of the simulations can be found in Figure 11 and in Table 2. When no optimization algorithms are enabled, all 40 hosts are continuously online and no reallocations are executed. The trade-off between maximizing

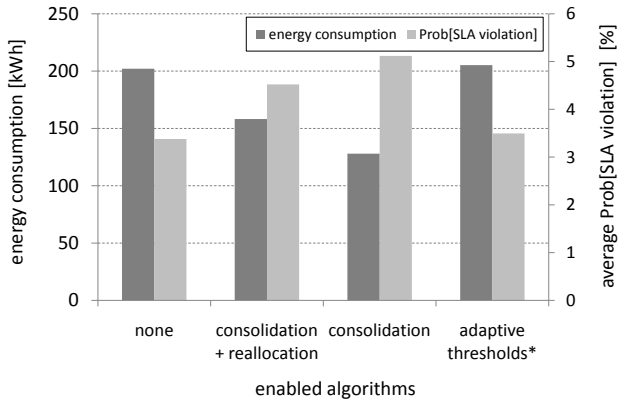


Fig. 11 Average energy consumption during a day and probability SLA violations occur on a VD for several combinations of enabled algorithms. The reference case is when no optimization algorithms are enabled.
* From literature: adaptive thresholds algorithm as proposed in [20]

Enabled algorithms	#migrations	average #online hosts	blocking probability [%]
none	0	40	5.24%
adaptive thresholds	178687	34.2	34.35%
consolidation and reallocation	4237	24.1	5.72%
consolidation	1405	24.1	5.66%

Table 2 Performance evaluation for consolidation algorithms

the quality experienced by the users – in other words minimizing the probability on SLA violations – and minimizing the energy consumption of the system can be seen in Figure 11. By enabling the adaptive thresholds algorithm, no savings in energy consumption are noticeable. Nevertheless, for this algorithm, the average number of online hosts is 34.2 instead of 40. Due to the enormous amount of reallocations there is no reduction in the energy consumption of the hosts because during the reallocation process, a virtual desktop consumes resources on both the original host and on the target host. When there are a lot of reallocations, the overhead of the reallocations negates the target of reducing the energy consumption of the system. When the consolidation presented in this paper is enabled, a significant reduction of 36.6% of the energy consumption of the system can be reached compared to the reference case where no optimization algorithms are enabled. The average amount of online hosts is 24.1 in this case. By enabling both the consolidation algorithm and the reallocation algorithm presented in the previous section, still a significant reduction of the energy consumption of 21.8% can be reached. The reduction of the energy consumption is smaller when both optimization algorithms are enabled, because more reallocations are scheduled: 4237 compared to 1405. As explained above, more reallocations lead to temporarily higher utilization and thus temporarily higher energy consumption of the system. On the other

hand, as the target of the reallocation algorithm is to increase the quality experienced by the users, a smaller probability on SLA violations can be reached when both algorithms are enabled. In the reference case, the probability on SLA violations is 3.38%. When the adaptive thresholds algorithm is enabled, the probability on SLA violations is 3.49%, but the blocking probability (i.e. the probability that an user request is refused) is unacceptably high: 34.35% compared to 5.24% in the reference case. The reason is again the enormous amount of reallocations and the fact that during a reallocation, two reservations are required for a single VD. Therefore, less resources are available for new user requests and the probability a client has to be rejected increases. When the consolidation algorithm is enabled, the probability on SLA violations is 5.12%, while the blocking probability is only 5.66%. When also the reallocation algorithm is enabled, a decrease of 0.6% of the probability on SLA violations can be reached, while the blocking probability increases with 0.06% because of the increase in performed reallocations.

8 Conclusion

The concept of virtual desktop cloud computing, i.e. executing applications in virtual desktops on remote servers, is very interesting because it enables access to any kind of application from any device. Current virtual desktop systems are mainly installed in LAN environments. Extending this to WAN environments involves important challenges to efficiently handle the typical large amount of geographically distributed and potential mobile users.

These challenges include to optimize the number of customers that can be served by a single host and thus minimize the investment costs for the service provider, to optimize the quality experienced by the customers by optimizing the distribution of the customers among the available hosts and to optimize the energy consumption of the hosts by shutting down redundant servers during quiet periods and thus saving energy costs for the service provider.

First, an optimization has been introduced to increase the average utilization on a single host. It was shown that the proposed overbooking approach, together with an advanced scheduler, can increase the average utilization of the resources with 29% at the cost of a probability that virtual desktops on the host cannot receive the requested resources (i.e. a probability on SLA violations) of 19%.

In practice, several hosts are available to execute the users' virtual desktops. A cost-based allocation algorithm has been presented that aims to maximize the quality of the service both for the customers and the service provider. Depending on the ratio of the two cost parameters, the probability on SLA violations varies between 2% and 12%, while the probability on SLA violations when applying a random allocation algorithm is 6.3%.

To further optimize the quality of the service, a reallocation algorithm has been proposed to rebalance the virtual desktops among the available hosts after a busy period. After a busy period, some hosts could still be fully loaded while

other hosts are almost not loaded and therefore, reallocating virtual desktops from fully loaded hosts to not loaded hosts can minimize the probability on SLA violations. By activating the reallocation algorithm every timestamp, in the simulations, the probability on SLA violations can be reduced from 14.4% to 4.65%.

The last optimization presented in this paper concerns an optimization of the energy consumption (and thus also an optimization of the costs for the service provider) by dynamically adapting the amount of powered-on hosts to the actual system load. By powering off hosts during quiet periods and powering on additional hosts during busy periods, the energy consumption can be decreased with 36.6% while the probability on SLA violations increases from 3.38% to 5.12%.

In this manuscript, we have shown that the probability on SLA violations can be influenced by our optimization algorithms. When a target or an objective is placed on the maximum probability on SLA violations, the parameters of our algorithms can be tuned so the target user experience can be delivered to the customers while optimizing the energy efficiency and utilization of the server infrastructure. Future work includes subjective user acceptance tests to investigate which probability on SLA violations is still acceptable for the customers.

Acknowledgements Lien Deboosere and Bert Vankeirsbilck are funded by a Ph.D grant from the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen). Part of the research leading to these results was done for the MobiThin Project and has received funding from the European Community's Seventh Framework (FP7/2007-2013) under grant agreement nr 216946.

References

1. Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25(6):599 – 616, 2009.
2. Google. Google apps cloud service. http://www.google.com/apps.
3. Microsoft Corporation. Windows Remote Desktop Protocol (RDP). http://www.microsoft.com/ntserver/ProductInfo/terminal/tsarchitecture.asp.
4. T. Richardson, Q. Stafford-Fraser, K.R. Wood, and A. Hopper. Virtual network computing. *Internet Computing, IEEE*, 2(1):33 –38, 1998.
5. A. Quiroz, Hyunjoo Kim, M. Parashar, N. Gnanasambandam, and N. Sharma. Towards autonomic workload provisioning for enterprise grids and clouds. In *Grid Computing, 2009 10th IEEE/ACM International Conference on*, pages 50–57, oct. 2009.
6. H. Vin, P. Goyal, and A. Goyal. A statistical admission control algorithm for multimedia servers. In *Proceedings of the second ACM international conference on Multimedia*, pages 33–40, 1994.
7. R.R. Boorstyn, A. Burchard, J. Liebeherr, and C. Oottamakorn. Statistical service assurances for traffic scheduling algorithms. *Selected Areas in Communications, IEEE Journal on*, 18(12):2651 –2664, December 2000.
8. Bhuvan Urgaonkar, Prashant Shenoy, and Timothy Roscoe. Resource overbooking and application profiling in a shared internet hosting platform. *ACM Transactions on Internet Technology*, 9:1:1–1:45, February 2009.

9. Mark Stillwell, David Schanzenbach, Frédéric Vivien, and Henri Casanova. Resource allocation algorithms for virtualized service hosting platforms. *J. Parallel Distrib. Comput.*, 70:962–974, September 2010.
10. Vanish Talwar, Bikash Agarwalla, Sujoy Basu, Raj Kumar, and Klara Nahrstedt. Resource allocation for remote desktop sessions in utility grids: Research articles. *Concurrency and Computation: Practice & Experience*, 18:667–684, May 2006.
11. Jose Alvarez-Bermejo and Javier Roca-Piera. A proposed asynchronous object load balancing method for parallel 3d image reconstruction applications. In *Algorithms and Architectures for Parallel Processing*, volume 6081 of *Lecture Notes in Computer Science*, pages 454–462. Springer Berlin / Heidelberg, 2010.
12. Manish Parashar, Xiaolin Li, and Sumir Chandra. *Advanced computational infrastructures for parallel and distributed adaptive application.* John Wiley & Sons, 2010.
13. Timothy Wooda, Prashant Shenoya, Arun Venkataramania, and Mazin Yousif. Sandpiper: Black-box and gray-box resource management for virtual machines. *Computer Networks*, 53:2923–2938, December 2009.
14. Laurent Lefvre and Anne-Ccile Orgerie. Designing and evaluating an energy efficient cloud. *The Journal of Supercomputing*, 51:352–373, 2010.
15. Andreas Berl, Erol Gelenbe, Marco Di Girolamo, Giovanni Giuliani, Hermann de Meer, Minh Quan Dang, and Kostas Pentikousis. Energy-efficient cloud computing. *The Computer Journal*, 53:1045–1051, 2010.
16. Anton Beloglazov and Rajkumar Buyya. Energy-efficient consolidation of virtual machines in cloud data centers. In *Proceedings of the IBM Collaborative Academia Research Exchange Workshop*, I-CARE 2010, October 2010.
17. Young Lee and Albert Zomaya. Energy efficient utilization of resources in cloud computing systems. *The Journal of Supercomputing*, pages 1–13, 2010. 10.1007/s11227-010-0421-3.
18. Anton Beloglazov and Rajkumar Buyya. Energy efficient resource management in virtualized cloud data centers. In *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid),*, pages 826–831, May 2010.
19. T.Y. Morad, U.C. Weiser, A. Kolodnyt, M. Valero, and E. Ayguade. Performance, power efficiency and scalability of asymmetric cluster chip multiprocessors. *Computer Architecture Letters*, 5(1):14–17, 2006.
20. Anton Beloglazov and Rajkumar Buyya. Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers. In *Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science*, MGC '10, pages 4:1–4:6, New York, NY, USA, 2010. ACM.
21. Daniel Gmacha, Jerry Roliaa, Ludmila Cherkasovaa, and Alfons Kemper. Resource pool management: Reactive versus proactive or lets be friends. *Computer Networks*, 53:2905–2922, December 2009.
22. Citrix. Xendesktop planning guide - hosted vm-based resource allocation, 2010.
23. M. Rosenblatt. A central limit theorem and a strong mixing condition. *Proceedings of the National Academy of Sciences of the United States of America*, 42(1):43–47, 1956.
24. Ramya Raghavendra, Parthasarathy Ranganathan, Vanish Talwar, Zhikui Wang, and Xiaoyun Zhu. No power struggles: Coordinated multi-level power management for the data center. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, March 2008.
25. Uri Lublin and Dror G. Feitelson. The workload on parallel supercomputers: modeling the characteristics of rigid jobs. *Journal of Parallel and Distributed Computing*, 63(11):1105 – 1122, 2003.
26. 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*, 41(1):23–50, 2011.
27. Silvano Martello and Paolo Toth. *Knapsack problems: algorithms and computer implementations.* John Wiley & Sons, Inc., New York, NY, USA, 1990.
28. William Voorsluys, James Broberg, Srikumar Venugopal, and Rajkumar Buyya. Cost of virtual machine live migration in clouds: A performance evaluation. In *Proceedings of the 1st International Conference on Cloud Computing*, pages 254–265, 2009.

29. Franco Travostino, Paul Daspit, Leon Gommans, Chetan Jog, Cees de Laat, Joe Mambretti, Inder Monga, Bas van Oudenaarde, Satish Raghunath, and Phil Yonghui Wang. Seamless live migration of virtual machines over the MAN/WAN. *Future Generation Computer Systems*, 22(8):901–907, 2006.

A Appendix: Impact of overbooking on user satisfaction: analytical use case

For the customer, it is important that the negative impact of overbooking on the amount of SLA violations experienced is acceptable. Applying a certain overbooking degree and the advanced resource scheduler essentially means that a virtual desktop *never* encounters an SLA violation when the requested resources do not exceed the reserved resources. When the virtual desktop requests more resources than reserved, it *can* encounter an SLA violation depending on the resource requests of other virtual desktops executed on the same host. Only when the total amount of requested resources exceeds the available resources (F) of the host at least one virtual desktop experiences an SLA violation. The probability that this occurs is calculated as

$$\begin{aligned} &\text{Prob}[\#\text{SLA violations on host } i \geq 1] \\ &= 1 - \int_0^F N(\sum_{j=0}^{n_i} \mu_j, \sum_{j=0}^{n_i} \sigma_j^2) \end{aligned}$$

Deducing the amount of virtual desktops experiencing an SLA violation - when the advanced resource scheduler is applied - is complicated. To make an analytical deduction of the probability that a virtual desktop encounters an SLA violation treatable, the following assumptions are made. We assume that the distributions of the resource consumption of the virtual desktops are identical and independent normal distributions: $N(\mu, \sigma^2)$. The deduction below treats a fully reserved host with three VDs and an overbooking degree of 50% (i.e., μ resources are reserved for each VD).

It is obvious that when all three virtual desktops request less resources than reserved, no SLA violations occur on the host. The domain in which all three virtual desktops request less resources than reserved is called $D_0 = \{\forall req_1,\ req_2,\ req_3\ :\ req_1, req_2, req_3 \leq \mu\}$. On the other hand, when all three virtual desktops request more resources than reserved, all three virtual desktops experience an SLA violation. This domain is called $D_3 = \{\forall req_1,\ req_2,\ req_3\ :\ req_1, req_2, req_3 > \mu\}$. When some virtual desktops request more resources than reserved and others request less resources than reserved, elaborations are required to determine the probability that a virtual desktop encounters an SLA violation. In this deduction, two cases can be distinguished: *(i)* the combination of one VD requesting less resources than reserved and two VDs requesting more resources than reserved (domain $D_1 = \{\forall req_1, req_2, req_3 : req_1 \leq \mu \,\&\, req_2, req_3 > \mu\}$) and *(ii)* the combination of two VDs requesting less resources than reserved and one VD requesting more resources than reserved (domain $D_2 = \{\forall req_1,\ req_2,\ req_3\ :\ req_1, req_2 \leq \mu\ \&\ req_3 > \mu\}$).

In the first case, i.e. in domain D_1, either 0, 1 or 2 virtual desktops experience an SLA violation. The probability that no SLA violations occur is calculated as the probability that the amount of additional resources requested by req_2 and req_3 are smaller than the amount of resources put in the resource pool by req_1. The probability that two SLA violations occur is calculated as the probability that both req_2 and req_3 request more than half of the resources put in the resource pool by req_1. Finally, the probability that exactly one virtual desktop encounters an SLA violation can be deduced from the previous probabilities.

In the second case, i.e. in domain D_2, either 0 or 1 virtual desktop experiences an SLA violation. Similar to the first case, the probability that no SLA violations occur is calculated as the probability that the amount of additional resources requested by req_3 is smaller than the amount of resources put in the resource pool by req_1 and req_2. The probability that one SLA violation occurs is easily deduced from the previous probability.

Discussing the calculation of all above probabilities in detail might be superfluous. Basic statistical methods can be applied to calculate the probabilities. As an example, the main steps of the first case (i.e., domain D_1) are presented below.

The probability that no SLA violations occur is calculated as

$$\begin{aligned} & Prob[\#\text{SLA violations} = 0 \mid \text{domain } D_1] \\ = \; & Prob[req_1 \leq \mu - (E[req_2 + req_3] - 2\mu) \mid \text{domain } D_1] \\ = \; & \int_0^{3\mu - E[req_2 + req_3]} f_{req_1}(x_1)\, \mathrm{d}x_1 \end{aligned} \tag{2}$$

with $f_{req_1}(x_1)$ the density function of the distribution of req_1 in domain D_1.

To calculate the expected value of the sum of req_2 and req_3, the density function of the sum of those resource requests has to be composed first. In the case of identically and independently distributions, the density function for $y = req_2 + req_3$ is

$$f_y(y) = -\frac{2}{\sigma\sqrt{\pi}} \times erf\left(\frac{2\mu - y}{2\sigma}\right) \times exp\left(-\left(\frac{2\mu - y}{2\sigma}\right)^2\right).$$

Substituting the expected value $E[y] = E[req_2 + req_3]$ of the density function $f_y(y)$ in equation (2) allows to calculate the probability that no SLA violations occur in domain D_1.

Next, the probability that two SLA violations occur is elaborated - under the assumption that the distributions of the resource requests are identical and independent - as

$$\begin{aligned} & Prob[\#\text{SLA violations} = 2 \mid \text{domain } D_1] \\ = \; & Prob[req_2 > \mu + \frac{\mu - E[req_1]}{2} \mid \text{domain } D_1] \\ & \times Prob[req_3 > \mu + \frac{\mu - E[req_1]}{2} \mid \text{domain } D_1] \end{aligned} \tag{3}$$

$$= \left(Prob[req_2 > \mu + \frac{\mu - E[req_1]}{2} \mid \text{domain } D_1]\right)^2. \tag{4}$$

The average number of SLA violations in domain D_1 is then calculated as

$$\sum_{i=0}^{2} i \times \text{Prob}[\ i \text{ SLA violations} \mid \text{domain } D_1].$$

In general, the average number of SLA violations on a host with n_i virtual desktops is calculated as:

$$\sum_{x=0}^{n_i} \text{Prob}[\text{requests} \in \text{domain } D_x] \times \sum_{i=0}^{x} i \times \text{Prob}[\ i \text{ SLA violations} \mid \text{domain } D_x]$$

$$\text{with} \quad \text{Prob}[\text{requests} \in \text{domain } D_x] = \frac{1}{2^{n_i}} \binom{n_i}{x} \text{ for } 0 \leq x \leq n_i.$$

The general approach of the analytical deduction presented above is applicable when the number of VDs on a host increases, however it becomes hard to analytically deduce the density distribution of the sum of a large amount of resource requests.

Therefore, simulations are used to determine the average amount of SLA violations on a host when more VDs are executed on the host. The results of the simulations can be found in Figure 12. In each simulation, the resource requests of a VD are distributed according to a normal distribution $N(10000, 1500)$ and the total amount of FLOPS of the host is equal to the total amount of reserved resources. Each simulation has been conducted until there was no significantly difference between the running average of two consecutive iterations.

The results of Figure 12 show that applying an overbooking degree of 50% does not necessarily lead to a probability on SLA violations of 50% when the advanced resource scheduler is used.

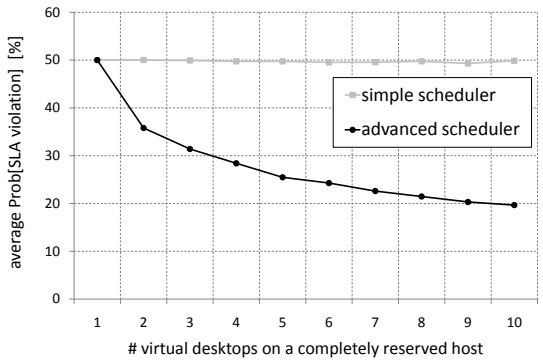


Fig. 12 Probability that a virtual desktop experiences an SLA violation for different amounts of simultaneous virtual desktops on a host with an overbooking degree of 50%. In each case, the total amount of available FLOPS of the host is completely reserved by the virtual desktops. The results are compared for the simple scheduler (i.e. standard scheduler in CloudSim) and the advanced scheduler discussed in section 3.2.