Guaranteed Bang for the Buck: Modeling VDI Applications to Identify Storage Requirements

Hao Wen, David H.C. Du, Milan Shetti, Doug Voigt, and Shanshan Li

Abstract—In the cloud environment, most services are provided by virtual machines (VMs). Identifying storage requirements of VMs is challenging, but it is essential for good user experiences while optimizing use of storage resources. Determining the storage configuration necessary to support and satisfy VMs first requires an accurate description of the VM configurations, and the problem is further exacerbated by the diversity and special characteristics of the VMs. In this paper, we study Virtual Desktop Infrastructure (VDI), a prevalent and complicated VM application, to identify and characterize storage requirements of VMs and determine how to meet such requirements with minimal storage resources and cost. We first create a model to describe the behavior of VDI, and we collect real VDI traces to populate this model. The model allows us to identify the storage requirements of VDI and determine the potential bottlenecks of a given storage configuration. Based on this information, we can tell what capacity and minimum capability a storage configuration needs in order to support and satisfy given VDI configurations. We show that our model can describe more fine-grained VM behavior varying with time and virtual disk types compared with the rules of thumb currently used in industry.

Index Terms—VDI; modeling; storage;

1 Introduction

With the rapid development of virtualization technology, traditional data centers are increasingly replacing dedicated physical machines with virtual machines (VMs) to provide services. Apart from improving hardware utilization, virtualization enables seamless migration of applications to a different physical host for the purpose of load balancing, planned software/hardware upgrades, etc. To avoid migrating data along with the in-memory state of the virtual machines, virtual machine data is typically stored on shared storage. In the shared storage architecture, multiple VMs/applications compete with each other for input/output (I/O)resources and capacity of the storage system. To reduce costs, data center administrators need to determine how to meet the storage requirements of these VMs with the minimum amount of required resources.

In this paper, we investigate how to identify storage requirements to support one popular type of VM application, Virtual Desktop Infrastructure (VDI) [1], [7], [13], [20], [36]. VDI runs multiple VMs with different operating systems and applications on several physical servers in a data center. This use of VMs is also known as desktop virtualization with each instance called a virtual desktop. Current VDI sizing work [9], [10], [15], [29] is unable to give an accurate description of the storage requirements of virtual desktops. They either use rules of thumb to guide storage provisioning [15] or test the performance of their storage array under a given fixed number of VDI instances [10]. To ensure VDI users to not see degraded performance in practice, administrators typically over-provision storage resources which may cause some waste. In addition, how CPU, memory, and storage resources of virtual desktops are configured may have a considerable impact on the I/O behavior of VDI. For example, each virtual desktop may access multiple heterogeneous data disks at different times causing each data disk to see significantly different I/O workloads. Therefore, how physical storage is configured and where these data disks are placed will impact whether the storage requirements are met. Unfortunately, current VDI sizing work fails to give a clear description of VDI configuration and its required storage resources.

When considering VDI performance, CPU, memory and storage can all be potential bottlenecks. We assume enough CPU and memory are provided in a data center. Besides, VMs can be migrated to another host [24] if the current host utilization is high. In this paper, we focus only on the storage requirements of a given VDI configuration to guarantee good performance.

Our objective is to meet storage requirements of VMs with minimal storage resources. This depends on many practical factors. In this paper, we mainly focus on the required storage capacity, throughput, and IOPS (I/Os per second). We create a model to describe I/O behavior of a single virtual desktop as well as a group of virtual desktops. With the model, we are able to tell when and where the performance bottlenecks occur. Based on this foundation, we can identify what capability a storage appliance needs in order to satisfy a given VDI configuration and its storage requirements.

To create such a model, we need to know the detailed implementation of VDI, virtual desktop types, and access patterns of virtual desktops. The implementation includes the organization of underlying storage and the composition of each virtual desktop. Considering that there are multiple virtual desktop types, the model should adapt to both homogeneous and heterogeneous combinations of virtual desktops. Each desktop undergoes certain stages (boot, login, steady state, and logoff) during its life cycle and accesses multiple different data disks at different stages. The access pattern of a virtual desktop is affected by its current stage and the data disks it accesses at different stages. Those data disks have different functions and see distinct I/O access patterns. When large numbers of virtual desktops arrive at different times, the aggregation effect of I/Os will lead to more variance of storage access patterns.

Our contributions are summarized as follows:

- We describe several different representatives of VDI and discuss their unique storage access patterns.
- We propose a system model to describe the I/O behavior of both homogeneous and heterogeneous configurations of VDI. To the best of our knowledge, this is the first model to do so for real life VDI systems.
- We identify the storage requirements of VDI and determine the bottlenecks on specific target virtual disks at a specific time.
- With the detailed storage requirements, we show how to size a minimum storage configuration that satisfies these VDI requirements.

The organization of this paper is as follows: Section 2 provides a detailed background of VDI and presents its unique characteristics. In Section 3, we propose our system model. Section 4 analyzes the VDI traces and discusses the storage requirements and bottlenecks generated from the model. Section 5 shows the application of our model in real life. In Section 6, we discuss future research work on migrating virtual desktops from VMs to Docker containers including some preliminary experimental results showing the feasibility of using Docker containers in VDI. Section 7 presents related work and Section 8 concludes the paper.

2 Background

Virtual Desktop Infrastructure (VDI) is a virtualization technology to provide desktop environments to remote users. VDI runs desktop operating systems (e.g., Windows, MacOS, etc.) on virtual machines (VMs) in a data center and presents normal desktops to remote users. Thus, a virtual desktop is a desktop environment or desktop operating system running in a VM. A virtual desktop can be associated with multiple different virtual disks. These virtual disks can reside either on local or shared storage in a data center. Virtual desktops are managed centrally in VDI. A user can use client devices such as personal computers, laptops, tablets, and mobile phones to connect to and operate his/her virtual desktop. An example of VDI architecture is shown in Figure 1. In VDI, all virtual desktops can be thought of as clones. They are clones of a master image. A master image is a VM template from which other virtual desktops originate. Virtual desktops are running in VMs created through hypervisors on physical servers.

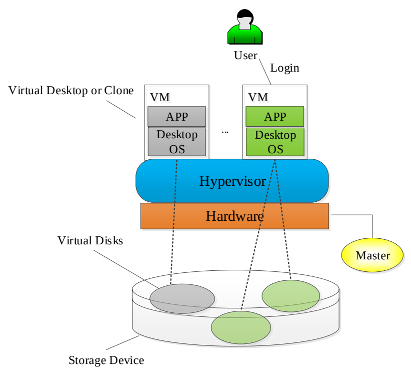


Fig. 1: An example of VDI architecture.

In the remainder of this section, we first introduce different clone types. Next, we describe how virtual desktops are assigned to users. We then introduce how we get different virtual desktop types by combining clone types and assignments. Finally, for each virtual desktop type, the associated data disk types are introduced.

2.1 Clone Type

There are two main types of virtual desktop clones. One is the full clone. A full clone copies the master image into its own virtual disk (not shared). The other type is the linked clone. Different from full clones, linked clones share the same operating system (OS) data as long as they are linked to the same replica (a clone of the master image). Each replica serves as a common base for a group of linked clones.

2.2 Virtual Desktop Assignment

Virtual desktop assignment binds a user with a virtual desktop. There are two main types of virtual desktop assignment: dedicated assignment and floating assignment. Dedicated assignment assigns a virtual desktop exclusively to a certain user. After the assignment, when a user tries to log into his/her virtual desktop, it is always the same VM serving the user. User profiles and user data are permanently saved in its local virtual disks. Both full clones and linked clones can be dedicated. Floating assignment arbitrarily assigns a virtual desktop to a user. Each time a user logs in, he/she may be assigned to a different VM. User profiles and user data are not saved in local virtual disks but are rather saved remotely. Only linked clones can be assigned as floating. Dedicated assignment has the advantages of good virtual desktop launching speed and user data access speed, but floating assignment allows for more flexible resource allocation across the whole system.

2.3 Virtual Desktop Types and Their Associated Disks

Combining clone type with assignment gives three kinds of virtual desktops: floating linked clone, dedicated linked clone, and dedicated full clone. Each type of virtual desktop is associated with a different set of virtual disks. According to the usage and types of data stored, there are six types of virtual disks: master image, replica, primary disk, persistent disk, remote repository, and full clone disk. The remainder of this section describes which of these virtual disk types are associated with each of the three virtual desktop types.

Floating linked clone. By definition, each linked clone is linked to a shared replica (a clone of the master image). To provision linked clones, a replica must first be created from the master image. In the linked clone pool, we should provision spare space for multiple replicas with different operating systems. Besides the shared replica, a primary disk contains the essential system data that is needed for each linked clone to remain linked to the shared replica and to function as an individual desktop. Floating linked clones are usually configured not to save user profiles and user data in their local virtual disks. User profiles are preserved in a remote repository independent of the virtual desktop. Each user has his own repository. Typically, they are stored in a NAS (Network Attached Storage) device. In the remaining of this paper, we use remote repository and NAS interchangeably.

Dedicated linked clone. Dedicated linked clones include those data disks essential for linked clones: replica and primary disk. However, what is unique about dedicated linked clones is that a separate persistent disk can be configured to store user profiles and user data. This disk is dedicated to a user. Attaching a persistent disk to a linked clone, virtual desktop makes that virtual desktop dedicated to the user. A remote repository is also needed to permanently store user profiles and user data.

Full clone. A full clone is always dedicated. Each full clone is an independent virtual desktop. Therefore, a full clone uses its own full clone disk (its regular virtual disk) to store the operating system, user profiles, and user data.

3 System Model

In order to understand the storage requirements of a VDI system, we propose a model to describe a VDI system in a data center. Based on this model, we can infer when and where the storage bottlenecks are. Since virtual desktops run in VMs, we describe the I/O behavior of VMs that are hosting virtual desktops in the model. We first model a single VM and then include different types of VMs to model a large number of VMs in VDI.

3.1 VM Life Cycle

A VM in VDI has multiple stages during its life cycle. Each stage shows distinct I/O behavior. Overall, a VM life cycle has four stages: **boot, login, steady state, and logoff**. In the boot state, VMs are booting. If many of those virtual desktops are powered on at the same time, or concentrated within a small time period, it becomes a boot storm. For large systems serving virtual desktops across multiple time zones, there can be several boot storms per day. After desktops are powered on, users will log into the desktops. Since the boot stage can be a storm, the login stage can also be a storm just after boot. After users log in, they start their everyday work, and virtual desktops transit to steady state. Compared with the boot stage and login stage, the storage throughput and IOPS are more predictable as people go about their daily routine at steady state. Depending on the application, different people may generate different I/O loads (e.g., watching videos, editing documents, etc.). Nevertheless, the I/O behavior of a virtual desktop in this stage is much steadier than in the login or boot stage. Logoff is the final stage during a VM life cycle.

3.2 Data Access Sequence

As we discussed in the previous section, different virtual disks are accessed by various types of virtual desktops. Even for the same virtual desktop type, virtual disks may see distinct I/O access patterns at different stages. We will now discuss how each virtual desktop type accesses virtual disks at each stage.

Data accesses of floating linked clones. The storage architecture of floating linked clones is shown in Figure 2(a). Multiple floating linked clones are running on hypervisors across different physical servers. On each server, there may be one or multiple master images to generate linked clones. Each floating linked clone is linked to a shared replica. Additionally, a primary disk is bound to a floating linked clone. These virtual disks are grouped into different data stores. Typically, each data store only has one type of virtual disk. In theory, there are no rules regarding which types of storage (SSD, HDD, etc.) should host those virtual disks. Administrators can opt to place those virtual disks on any type of storage. However, without detailed analysis and characterization of VDI demand, it is hard for the administrator to give a good placement that satisfies VDI storage requirements. This is the focus of our paper. The green lines and red lines in the figure show the data accesses in the boot and login stage, respectively. When a virtual desktop is booting, shared OS data have to be read from the replica first. These data are loaded into VM memory to initiate a system boot. Those essential binaries, libraries, etc. are written to the linked clone primary disk as well for future accesses. When a user tries to log in, the virtual desktop must first load user profiles from the remote repository to memory to authenticate the user and then configure the desktop settings. The user profiles are also written to the linked clone's primary disk for future accesses. After the desktop environment is loaded, the virtual desktop goes to the steady state. The user operations during steady state may need to access user data like the user's personal documents, videos, photos, music, etc. stored in the remote repository. These data are downloaded to the primary disk when first accessed. All subsequent accesses are directed to the copies on primary disk. Any changes to the user data are synchronized to the remote repository at regular intervals. Once the user logs off, that virtual desktop is cleaned, so no user profiles or user data is saved on that primary disk. When that user logs into his/her desktop again, a different VM may be assigned.

Data accesses of dedicated linked clones. The data accesses of dedicated linked clones are shown in Figure 2(b). During the boot process, there is no need to load OS data from the replica anymore as long as it is not the first boot of a fresh desktop. Those OS data are already stored in primary disk. During the login process and steady state, user profiles and user data are read from the persistent disk rather than from the remote repository. The persistent disk acts as a cache of the remote repository. During steady state, synchronization of user profiles and user data between persistent disk and the remote repository occurs.

Data accesses of full clones. A full clone is like a regular desktop. All information including OS data, user profiles, and user data is stored in full clone disk. Thus, all I/O accesses are on this type of virtual disk during all stages.

Table 1 shows when each type of virtual desktop will access each kind of virtual disk. In the table, we omit master image because it is not accessed by virtual desktops at runtime. We also omit logoff stage because people do not care about the performance of logoff.

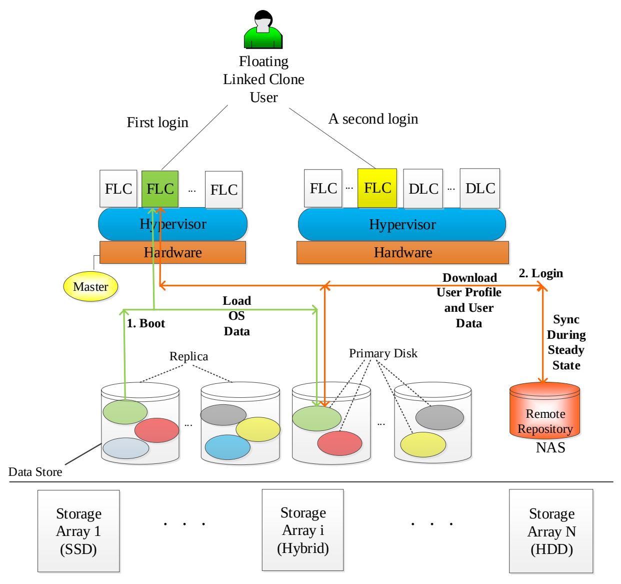
3.3 VM Model

We define a model to answer at time t, how much data will be read from and written to each virtual disk at a given time t. The basic idea is to sum all read I/Os and write I/Os happening on the same virtual disk at time t. In the following subsections, we discuss the model for a single virtual desktop and multiple virtual desktops, respectively.

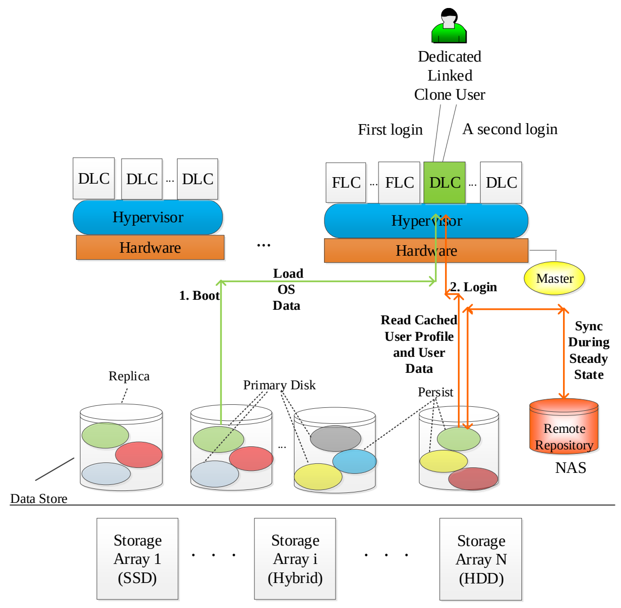
3.3.1 Single VM

Formula 1 calculates the overall amount of data accessed on a *target* by a VM in life cycle *stage* for a single VM. The *target* is the virtual disk that I/Os reach as listed in Table 1. The *stage* is the VM life cycle. $RWper_{\text{stage,target}}$ is the read ratio or write ratio during different stages on different targets when we calculate the size of read data and write data, respectively. The I/O sizes $S^i_{\text{stage,target}}$ are several discrete values. Since there are many different I/O sizes, here we only choose several significant I/O sizes at each life cycle stage on each target. An I/O size is significant when it accounts for most of the I/Os. We decide significance using two factors: 1) the access frequency of the I/O size is high, and 2) the total amount of data transferred under this I/O size is large. For a stage and target, the percentage of total I/Os that are of a certain significant I/O size i is denoted as $Psize^i_{\text{stage,target}}$. $E_{\text{stage,target}}(t)$ describes the expected number of I/Os at time *t*, which tells how many I/Os arrive at *target* during *stage* at time *t*. In practice, when calculating how many I/Os are expected to come at time *t*, we can multiply by a small time interval dt (e.g., 1 second). We do the summation for all significant I/O sizes i to obtain the overall amount of data accessed on a target at VM life cycle stage.

$$\sum_i E_{\text{stage,target}}(t) \times dt \times RWper_{\text{stage,target}} \times S^i_{\text{stage,target}} \times Psize^i_{\text{stage,target}} \tag{1}$$



(a) Floating Linked Clone Storage Configuration



(b) Dedicated Linked Clone Storage Configuration

Fig. 2: VDI Storage Configuration.

TABLE 1: Virtual Disks Accessed at Each Stage by Different Virtual Desktop Types
B=Boot L=Login S=Steady State

	Replica	Primary Disk	Persistent Disk	NAS	Full Clone Disk
Floating Linked Clone	B	B,L,S	-	L,S	-
Dedicated Linked Clone	-	B,L,S	L,S	S	-
Full Clone	-	-	-	-	B,L,S

3.3.2 Multiple VMs

In contrast to a single VM, more factors need to be considered when including a number of virtual desktops: 1) VMs start to boot (arrive) at different time, 2) I/O behavior of different virtual desktop types are different. 3) IO behavior of VMs running different operating systems or user applications are different. The VM arrival rate is the key to including multiple VMs in the model. In a data center, VMs arrive at different time, so the multiple VM model should also include an arrival distribution. We use a function $N(x)$ to describe the number of VMs arriving at time x. Different virtual desktop types determine which virtual disks are targets. Virtual desktop types along with the operating system types and user applications affect the arriving I/O sizes as well as their percentages at time *t*. When we determine the amount of data accessed on each target, we base it on the virtual desktop type and the stage they are in according to Table 1.

3.3.2.1 Assumptions

In order to make the model of multiple VMs simple and practical, we make several assumptions. First, the VM arrival rate is not related to virtual desktop types, operating systems, or user applications, and this arrival rate of different virtual desktops follows a distribution. Second, login immediately follows boot and there are no idle intervals. In the following subsections, we show how these assumptions are applied to the model.

3.3.2.2 Multiple VMs of the Same Type

If multiple VMs are of the same virtual desktop type and have the same operating system and user applications, their parameters are all the same. In this case, we only need to consider how to integrate the I/O requests of VMs at different stages into the model. The overall amount of data accessed on virtual disk *target* by VMs at life cycle *stage* for multiple VMs of the same type can be calculated using Formula 2. $N(x)$ is the VM arrival rate and indicates the number of VMs arriving at time $x(where x \leq t)$. For each group of $N(x)$ VMs that arrive at time x, $E_{\text{stage,target}}(t)$ describes the expected number of I/Os for these VMs at time *t*. In other words, it tells how many I/Os arrive at *target* from VMs in *stage* at time *t*. For all VMs that are now in *stage* at time *t*, their arrival time must fall into a prior time interval $[t_1, t_2]$ determined by the amount of time needed for each stage and where $t_2 \leq t$. We calculate how much data is read or written by VMs currently in *stage* that arrived at any time point in $[t_1, t_2]$ and add them together to obtain the overall amount of data accessed on virtual disk *target* by VMs in life cycle *stage* at a given time *t*. The other parameters are the same as the single VM model.

$$\sum_{x=t_1}^{t_2}[N(x) \times \sum_i (E_{\text{stage,target}}(t) \times dt \times RWper_{\text{stage,target}} \times S^i_{\text{stage,target}} \times Psize^i_{\text{stage,target}})] \quad (2)$$

Here we take one virtual disk as an example. Suppose the target is a primary disk, and we want to calculate the amount of data read from this target at time *t*. According to Table 1, linked clones access a primary disk in the boot, login, and steady state stages. Therefore, the data accesses should be the combination of three parts: I/Os from VMs in the boot process, I/Os from VMs in the login process, and I/Os from VMs in steady state. I/Os from VMs in the boot process at time *t* can be calculated by

$$\sum_{x=t-t_0}^{t} [N(x) \times \sum_i (E_{\text{boot,primary}}(t) \times dt \times RWper_{\text{boot,primary}} \times S^i_{\text{boot,primary}} \times Psize^i_{\text{boot,primary}})] \tag{3}$$

Here we assume each of these VMs of the same type takes t_0 time units (e.g., seconds) to finish issuing I/Os during its entire boot process. VMs that started booting (arrived) during time interval $[t - t_0, t]$ are still in their boot process. For every group of VMs that arrived at each time point in $[t - t_0, t]$, we calculate how much boot data they read from the virtual disk at time *t* and add them together. Similarly, I/Os from VMs in the login process at time *t* can be calculated by

$$\sum_{x=t-t_0-t_1}^{t-t_0} [N(x) \times \sum_i (E_{\text{login,primary}}(t) \times dt \times RWper_{\text{login,primary}} \times S^i_{\text{login,primary}} \times Psize^i_{\text{login,primary}})] \tag{4}$$

Here we assume VMs of this same type take t_1 time units to finish issuing I/Os during their entire login process. According to Assumption 2, login follows boot immediately, and there are no idle intervals. Therefore, VMs that arrived during time interval $[t-t_0-t_1, t-t_0]$ are in their login process at time *t*. Finally, I/Os from VMs in their steady state at time *t* can be calculated using

$$\sum_{x=start}^{t-t_0-t_1} [N(x) \times \sum_i (E_{\text{steady,primary}}(t) \times dt \times RWper_{\text{steady,primary}} \times S^i_{\text{steady,primary}} \times Psize^i_{\text{steady,primary}})] \tag{5}$$

Here we assume an initial point in time *start* when VMs start to arrive. Any VMs that arrived before $t - t_0 - t_1$ are now (at time *t*) in their steady state.

Since a virtual disk is not accessed during all stages by all virtual desktops, when calculating the total amount of data read/written on target at time *t*, we include only I/Os from appropriate virtual desktop types and the stages they are in according to Table 1. By selecting different targets, we can obtain the amount of data accessed on all virtual disks. By traversing time *t*, we can see how data accessed on a target varies with time. Therefore, we can estimate when a bottleneck happens on each target.

3.3.2.3 Multiple VMs of Different Types

VMs with different virtual desktop types, operating systems, and user applications show different I/O behavior. We define a VM type as VMs running the same type of virtual desktop, the same type of operating system, and the same user applications. For each VM type, we apply Formula (2) to calculate how much data are read from and written to each corresponding target at time *t*. The corresponding targets are chosen from Table 1 according to the virtual desktop type. In a data center, each VM type accounts for a different proportion of I/Os. When combining them together, we need to use the weighted proportion of each VM type. According to Assumption 1, the VM arrival rate is not related to its virtual desktop type, operating system, and user application. The overall arrival rate *N(x)* is the same when calculating each VM type. Therefore, we can use the weighted average of all VM types to get the total amount of data accessed.

4 Data Analysis and Evaluation

In order to get correct values of I/O parameters in our model, we collect boot, login, and steady state traces of different types of virtual desktops in VDI. We then analyze I/O behavior of virtual desktops and derive those parameters in our model from the traces. The storage demands are then generated.

In this section, we first analyze the burstiness of I/Os in order to describe the expected number of I/Os at time *t* in our model. Then we analyze I/O behavior of each virtual desktop type from traces. Finally, we show a simulation using our model to generate I/O demands on each target.

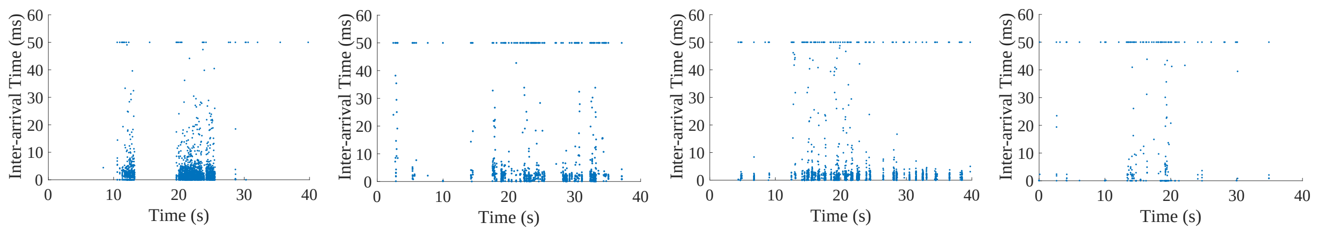
4.1 Trace Collection

We collect VDI traces on four 1U Dell r420 servers, each with two Intel Xeon E5-2407 v1 2.2GHz quad-core processors and 12 GB of DRAM. Servers are connected to Dell/Compellent SC8000 storage which has eight 400GB SSDs and 8 600GB HDDs. In our trace capturing VDI environment, we have VMware vSphere Hypervisor (ESXi) 5.5.0 installed on four servers to form a cluster. We use vCenter Server 5.5 Update 1 as the cluster manager. The VDI product installed is VMware Horizon View 6.0. Windows 7 x64 is used as the desktop OS. To capture traces of different clone types, we set up floating assigned linked clone pools, dedicated assigned linked clone pools and full clone pools. In this environment, any I/Os generated by a virtual desktop go through the hypervisor (ESXi) first and then go to the physical storage. To capture block traces, a common method is running blktrace at the host machine. Unfortunately, ESXi is a commodity hypervisor so we cannot install or run blktrace on it. Instead, we use a different method. We setup an NFS server to provide storage to the virtual desktops. We mount multiple volumes on this NFS server and configure them as data stores of ESXi servers. Through this configuration, we can choose to put virtual disks in the data stores presented by this NFS server when creating virtual desktop pools. At the NFS side, we run blktrace on multiple volumes to collect block traces. By analyzing the block traces collected from different volumes, we are able to analyze the I/O behavior of different virtual desktops.

Since NFS just transmits the original file I/Os to VMDK (Virtual Machine Disk) from ESXi hosts to the NFS server, the I/O patterns in traces collected from ESXi (if we could) should be the same as those in traces collected from our NFS server. Using this setup, we collect traces of virtual desktops at their boot and login stages. During the steady state stage, the workload depends on the real user application behavior. Here we run VMware View Planner 3.5 [16] as a workload generator to generate steady state workload. The applications include Adobe Reader, Excel, Internet Explorer, PowerPoint, and Word. VMware View Planner executes open, read, write, save, and close operations using these applications to simulate a real real-life workload. We set the workload to pause randomly for between zero and 10 seconds between operations. In this way, we generate light or medium load during a steady state.

4.2 Burstiness of Requests

Figure 3 depicts the inter-arrival time of I/Os of a floating linked clone from the traces. For those sparse requests whose inter-arrival time is greater than 50ms, we simply truncate them to 50ms. The coefficients of variation of the inter-arrival time from (a) to (d) in Fig. 3 are 13.61, 8.15, 8.72, and 3.57 respectively. A higher coefficient of variation indicates more bursty arrival patterns. In addition, it is also obvious that data points are aggregating at some time points rather than evenly distributing



(a) Boot on Replica (b) Boot on Primary Disk (c) Login on Primary Disk (d) Login on NAS

Fig. 3: Inter-arrival Time of I/Os of a Floating Linked Clone

across time. This also indicates that I/Os on all targets are bursty. Other clone types show similar bursty features. Thus, it is not appropriate to describe the expected number of I/Os at time t by statistical models, like the famous Poisson Process. Since the actual arrival pattern happens during a very short time interval (μs) and we are only interested in measuring I/Os in terms of seconds, we can simply assume that requests arrive uniformly within bursts and include this burstiness in our model. The expected number of I/Os at time t, if t falls into a burst, is the average number of I/Os per time unit during the interval of that burst. Otherwise, when t is outside of a burst, the expected I/O count is zero.

4.3 Single Virtual Desktop Analysis

In this subsection, we analyze the I/O behavior of a single virtual desktop from traces. The traces include the I/O requests sent to different targets by different types of virtual desktops in boot, login ,and steady state stages.

4.3.1 Floating linked clone

During the boot process of a floating linked clone, reads are dominant on the replica, accounting for 99.8% of total I/Os on the replica. The total amount of data read is 188MB. To the contrary, writes become dominant on the primary disk, accounting for 99.9% of total I/Os on the primary disk. The total amount of data written is 20MB. This is because when a floating linked clone is booting, it needs to load OS data from the shared replica first. Some of this data is written into the primary disk for future use. During the login process, I/Os happen on the primary disk and NAS. On the primary disk, reads account for 22.7% and writes account for 77.3% of disk activity. On the NAS, reads account for 69.5% and writes account for 30.5 of I/OsIn this process, the user profile needs to be loaded from NAS to enable identity authentication and desktop configuration. Some of these data are written into primary disk for future use. Other applications involved during the login process may also write to the primary disk.

4.3.2 Dedicated linked clone

The boot process of a dedicated linked clone is quite different from a floating linked clone. Dedicated linked clones preserve data in primary disks after users log off rather than reload data for every instance as is done with floating linked clones. In most cases, dedicated linked clones do not need to load OS data again from a replica during the boot process. Loading OS data into the primary disk only happens the first time a fresh desktop is booted. Here we only consider the most general case where primary disks already preserve OS data. In the traces, we find boot I/Os only go to primary disks. Dedicated linked clones utilize persistent disks to preserve user profiles and user data. Therefore, I/O behavior of dedicated linked clones during login are different from floating linked clones. Some I/Os are now shifted to the persistent disk, and the number of I/Os accessing NAS is reduced. This is because we do not need to load as much data from NAS to the primary disk when users log in, since the data are already there in persistent disk. We can directly read user profiles from the persistent disk to proceed with the login process. On the primary disk, reads account for 48.9% and writes account for 51.1%. On the persistent disk, reads account for 24.6% and writes account for 75.4%. On the NAS, reads account for 32.2% and writes account for 67.8%.

4.3.3 Full clone

In a full clone, I/Os are aggregated in one type of virtual disk. During the boot process, reads account for 42.2% and writes account for 57.8%. During the login process, reads account for 69.0% and writes account for 31.0% of I/Os.

4.4 Multiple Virtual Desktops

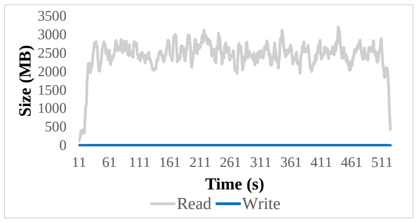
With traces collected for each type of virtual desktop, we can now aggregate multiple virtual desktops together. We do experiments to simulate multiple virtual desktops arriving at different times and see how the I/Os on each target vary with time.

Experiment Setup. We assume a company uses VDI for its employees and has 5,000 virtual desktop instances. Without loss of generality, we assume the arrivals of employees follow a Poisson distribution and the arrival rate is 10 per second. In order to aggregate I/Os of VMs at different stages in Formula 2, we use the parameters derived from the traces in the model. Other users of our model can add their own customized virtual desktop types and VM arrival rate to get their own results. In the following subsections, we show how much data is accessed on each virtual disk at any time since the first user arrives under four scenarios: 1) they uses all floating linked clones, 2) they use all dedicated linked clones, 3) they use all full clones, 4) they use a mixture of different clones.

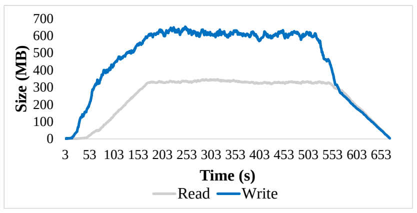
4.4.1 Floating linked clones

If we assume this company prefers that employees share virtual desktops as much as possible in order to reduce license costs (Windows, VMware, etc.), it may use all floating linked clones. Figure 4 shows the amount of data accessed on each of the targets from when the first floating linked clone arrives to when all floating linked clones have transitioned to steady state.

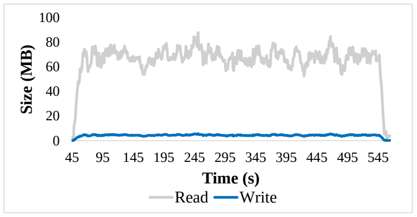
On the replica, as seen in Figure 4(a) the I/Os are read dominant and quite heavy. The rate of data read rises sharply to around 2.8 GB/s within the first 30 seconds. In the next 500 seconds, the workload is relatively stable and stays around 3 GB/s with a peak of 3.3 GB/s. Once all virtual desktops have arrived and their boot processes are completed, the I/Os start to drop dramatically within the final 20 seconds. Overall, replica disk activity is read intensive for floating linked clones. Unlike the



(a) Replica

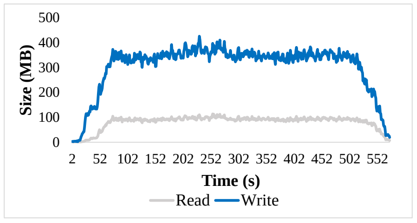


(b) Primary Disk

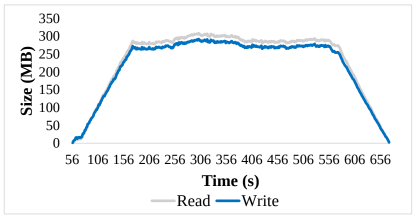


(c) NAS

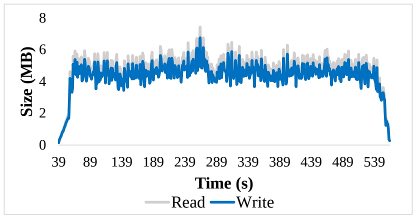
Fig. 4: Amount of Data Accessed on Targets of 5,000 Floating Linked Clones



(a) Primary Disk



(b) Persistent Disk



(c) NAS

Fig. 5: Amount of Data Accessed on Targets of 5,000 Dedicated Linked Clones

replica, the I/Os on the primary disk in Figure 4(b) are more balanced as it is accessed during all stages. However, it still sees a large volume of I/Os with data rates in the hundreds of megabytes per second for reads and writes. As shown in Figure 4(c), the amount of data accessed on NAS is quite small. If more applications are installed and more user data is generated, NAS activity will increase.

From this simulation, we obtain the detailed storage requirements of a floating linked clone as listed in Table 2.

4.4.2 *Dedicated linked clones*

If we assume this company wants to reduce license costs and at the same time employees are inclined to have dedicated virtual desktops, it may use all dedicated linked clones. Figure 5 shows the amount of data accessed on each of the targets from when the first dedicated linked clone arrives until all dedicated linked clones have transitioned to steady state.

The I/Os on the primary disk of the dedicated linked clones, as seen in Figure 5(a), are much lighter than those of the floating linked clone. Once all VMs finish their boot and login stages, the I/Os on the primary disk are minimal. On the persistent disk, as we see in Figure 5(b), reads and writes mainly rise during the login stage and drop to a minimum amount in the steady state. Figure 5(c) shows I/Os on NAS.

Detailed storage requirements of a dedicated linked clone based on this simulation are listed in Table 3.

4.4.3 *Full clones*

If we assume this company wants to avoid the complexity of server and storage configurations caused by linked clones, it may use all full clones. Figure 6 shows the amount of data read and written on the full clone disk. We can see the total amount of data read is much greater than the total amount of data written. There is an obvious stage of high I/Os where VMs are in their boot and login stage. The I/Os drop suddenly when all VMs finish booting and then gradually decline to the minimum as the VMs transition to steady state.

Detailed storage requirements of a full clone obtained from this experiment are listed in Table 4.

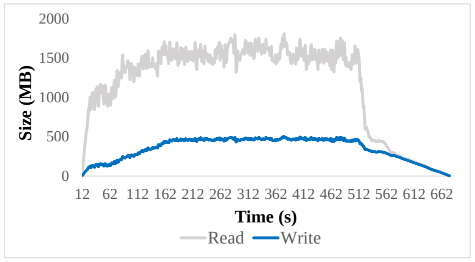


Fig. 6: Amount of Data Accessed on Target of 5,000 Full Clones

TABLE 2: Requirements of a Floating Linked Clone

	Replica		Primary		NAS	
	R	W	R	W	R	W
Average KB/s	497.50	0.044	50.69	93.37	13.45	0.89
Capacity per VDI user if thin provisioned: 4GB. 3 shared replicas: 75GB Shared NAS: 1TB						

TABLE 3: Requirements of a Dedicated Linked Clone

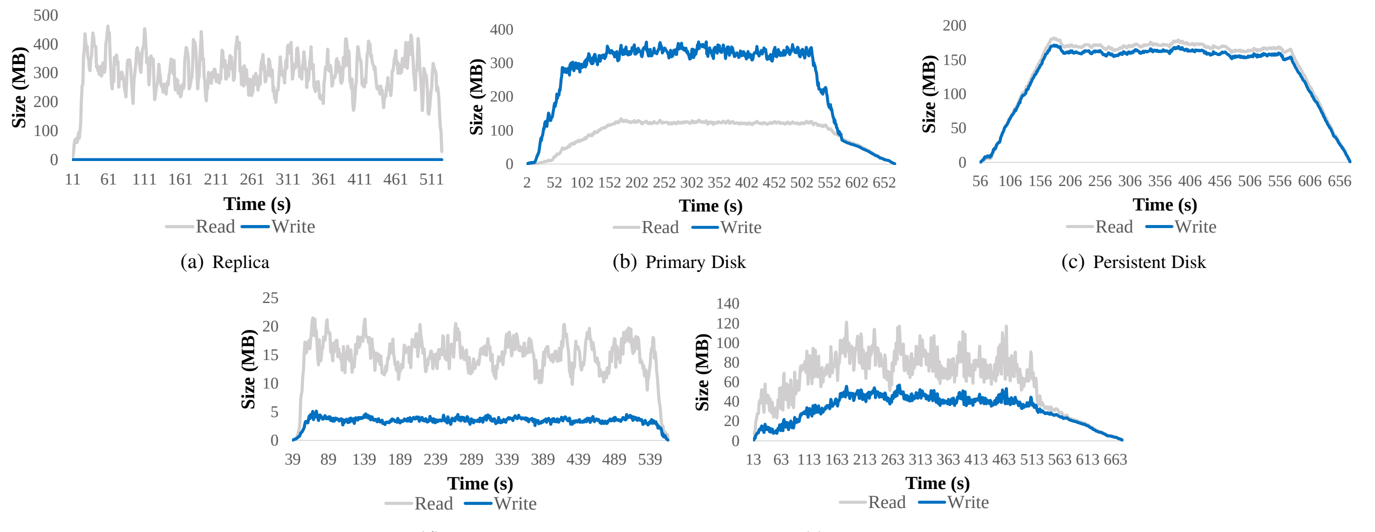
	Primary		Persistent		NAS	
	R	W	R	W	R	W
Average KB/s	16.69	62.59	48.02	45.43	0.98	0.90
Capacity per VDI user if thin provisioned: 12GB. 3 shared replicas: 75GB Shared NAS: 1TB						

TABLE 4: Requirements of a Full Clone

	Full Clone Disk	
	R	W
Average KB/s	238.07	71.22
Capacity per VDI user if thin provisioned: 12GB.		

4.4.4 *A mixture of different clones*

Now assume this company has various needs regarding virtual desktops. First of all, it does not want to pay high license costs, hence most virtual desktops are linked clones. Also, most



(a) Replica
(b) Primary Disk
(c) Persistent Disk
(d) NAS
(e) Full Clone Virtual Disk

Fig. 7: Amount of Data Accessed on Targets of a Mixture of Clones

employees require exclusive use of virtual desktops, so most of the linked clones are dedicated linked clones. Finally, there is a small department developing software tools that require high performance. To avoid the configuration complexity and degraded performance of linked clones, a small number of full clones are provided. According to this scenario, we evaluate a combined ratio of 3:6:1 for floating linked clones to dedicated linked clones to full clones. Figure 7 shows the amount of data read and written on the replica, primary disk, persistent disk, NAS, and full clone virtual disk in this mixed configuration.

The I/Os on the replica arise from the boot of floating linked clones. The amount of data accessed shows a similar pattern as Figure 4(a). Since the number of floating linked clones is only 30% of the 5,000 virtual desktops, there are fewer I/Os than there were in the 5,000 floating linked clone case. Compared with dedicated linked clones (Figure 5(a)), , floating linked clones (Figure 4(a)) have more intensive I/O accesses on the primary disk. The pattern of data accessed on primary disk in this mixed case (Figure 7(b)) is similar to the floating linked clone case. However, the total amount of data accessed is smaller. The persistent disk (Figure 7(c))is exclusively accessed by the dedicated linked clones, so its I/O pattern is similar to Figure 5(b). Although dedicated linked clones access NAS (Figure 7(d)), the I/Os are reduced a lot due to the existence of the persistent disk. Therefore, the I/O pattern on NAS is similar to that of the floating linked clones (Figure 4(c)) but with a smaller total data amount. Finally, the full clone virtual disk (Figure 7(e)) is only influenced by full clones, so it shows the same pattern as in Figure 6 with a proportional reduction in data trasferred.

4.5 Validation

In order to evaluate the correctness of our model, we compare the I/O requirements we generate from our model with experimental results from real Hewlett Packard Enterprise (HPE) systems. Two experiments are done by running virtual desktop workloads in two different environments in HPE 3PAR. In the first environment, three HP Proliant BL460c Gen8 servers run the VDI workload. Each server has two Intel Xeon 2.9GHz, octa-core CPUs and 128GB of memory. One HPE 3PAR StoreServ 7450 is used as VDI storage. It has 24 920GB MLC SSDs and 8Gbps Fibre Channel SAN connectivity. 40 volumes are provisioned for testing, each of which has 125GB of capacity. In the second environment, two HP ProLiant DL560 Gen8 servers run the VDI workload. Each server has four Intel Xeon 2.7GHz octa-core CPUs and 256GB of memory. One HPE StoreVirtual 4335 is used as VDI storage. It has nine 400GB MLC SSDs, 21 900GB 10K RPM SAS HDDs, and 10Gbps iSCSI SAN connectivity. 12 volumes are provisioned for testing, each of which has 125GB of capacity.

In both environments, VMware vSphere 5.1 and vCenter Server 5.1 are the deployed hypervisor and cluster manager. IOmark-VDI [4] is used as a benchmark tool. This tool re-creates a VDI environment automatically. The created virtual desktops run Windows 7 x64 as the guest OS. The tool generates a workload of boot stage and steady state stage linked clones. Since IOmark-VDI does not provide a login stage workload, we only evaluate our simulated results in the boot stage and steady state stage.

The tested results in the first and second environment are shown in Tables 5 and 6, respectively. In the experiments, different numbers of floating linked clones are booted at the same time. During steady state, the "Standard" workload is running. From these tables, we can see the peak IOPS per virtual desktop, which occurs during the boot stage, is 139 regardless of the testing environment as long as there are enough system resources. The peak

TABLE 5: I/Os on HP 3PAR StoreServ 7450

User Count	Max IOPS	Max IOPS per user	Max Reads per second	Max Writes per second
512	71168	139	64051	7117
1056	146784	139	132195	14589
1504	209056	139	188150	20906
2016	280224	139	252202	28022

TABLE 6: I/Os on HP StoreVirtual 4335

User Count	Max IOPS	Max IOPS per user	Max Reads per second	Max Writes per second
148	20513	139	18462	2051
211	29329	139	26396	2932
318	44202	139	39781	4420
537	74643	139	67178	7465

read IOPS is around nine times of the peak write IOPS. During steady state, the average IOPS is 6.26. In our model comparison experiment, we simulate multiple floating linked clones arriving at the same time by applying our model. We set the arrival rate of virtual desktops to match the total number of virtual desktops from the HPE system tests. We find the peak IOPS per virtual desktop is 141 and the peak read IOPS is 8.8 times the peak write IOPS. During steady state, the average IOPS is 7.01. Given these results, we conclude that the simulation based on our model is able to reflect the real workload characteristics from the HPE testing results.

5 Application of Proposed Model

In this section, we show how to apply our model in real life. We identify more fine-grained storage requirements of a VDI system and compare them with the storage performance requirements provided by VMware. Then we show what storage system is needed to deploy such a VDI system.

5.1 Fine-grained Storage Requirements

We first show we can find more accurate and fine-grained storage requirements of a VDI system. We calculate the IOPS demand on each type of virtual disk of different virtual desktops by applying the traces as inputs into the model.

To assist administrators when they are determining the resources necessary to support VDI, VMware has given IOPS requirements as a rule of thumb [19] as shown in Table 7. It classifies users based on their IOPS requirements. However, we know different types of virtual desktop have different storage requirements, and for the same type of virtual desktop, the storage requirement of each virtual disk is also different. The VMware guidance based on rules of thumb does not describe these differences in detail. With our model, we can easily calculate the average IOPS of each target of each virtual desktop type, as shown in Table 8, based on the required read and write throughput of each virtual desktop and the significant I/Os in the traces. We also give the corresponding classification of storage requirements at the virtual disk granularity for each virtual desktop type. In our VDI traces, all virtual desktops run light or medium load jobs like reading PDF files or editing Word, Excel, and PowerPoint documents, so they should be characterized as $Light$ or $Medium$. We can see in Table 8 that the storage requirements of all virtual disks of our virtual desktops are in the $Light$ and $Medium$ classes. Thus, our model is useful and able to describe more fine-grained storage requirements of virtual desktops in a VDI environment.

TABLE 7: VDI IOPS Requirements from VMware

User Classification	IOPS Requirements Per User
Light	3-7
Medium	8-16
Standard	17-25
Heavy	25+

TABLE 8: Average IOPS on Each Target of Each Virtual Desktop

Floating Linked Clone			Dedicated Linked Clone			Full Clone
Replica	Primary	NAS	Primary	Persistent	NAS	Full Clone Disk
5.45	12.25	0.61	5.27	8.26	0.22	9.52
Light	*Medium*	*Light*	*Light*	*Medium*	*Light*	*Medium*

IOPS is widely used in industry to describe storage requirements and capabilities. VMware uses IOPS to guide the VDI storage sizing. For example, they use the IOPS in Table 7 to calculate the performance requirement for each LUN (logical unit number, used to identify a device or a logical disk) when determining the storage hardware necessary for VDI. However, only considering IOPS is less than adequate. As seen in Section 4, the amount of data read per second from the replica can be very large during boot time. However, the average IOPS on the replica listed in Table 8 does not show this information. If storage is allocated to the replica based on that *Light* IOPS classification, users could experience long latency during boot time. Thus, throughput and read/write ratio should also be considered when allocating storage for VDI. Fortunately, our model can provide this type of information and can guide administrators to an even more fine-grained storage configuration. Our model can show the storage requirements like storage capacity and throughput on each target directly and how they vary with time. In addition, some VDI users have response time requirements. We can find the expected response time of various storage systems by combining the expected IOPS from our VDI model with the Response Time/Throughput relationship shown in benchmark results provided by the SPC (Storage Performance Council, they have extensively tested many storage systems and freely provide plots of response time given IOPS for each system) [11], [12].

5.2 Sizing Storage Hardware for Specific VDI Requirements

When we decide how much storage hardware we need, we can first look at the VMware guidance [15] which considers IOPS and storage capacity. Then we add more dimensions by considering the distinct I/O access patterns in terms of read/write ratio and throughput on different types of virtual disks. For example, assume we are sizing storage for a company that uses VDI for its employees across three time zones. In each time zone, it deploys 5,000 floating linked clones and the I/O access patterns of these virtual desktops are the same as the example in Section 4.4. We are going to buy one of the storage systems from Table 9. The prices of the storage systems in the table increase as the performance improves. We are going to choose the cheapest storage system that can meet the storage requirements of the VDI system.

We can first consider the I/Os during steady state, as it is only determined by user behavior. If employees in this company generate a standard IOPS, e.g. 8 IOPS per user, then we need a storage system that can support $8 \times 5000 \times 3 = 120,000$ IOPS. In this case, HP 3PAR T400 may be a good choice. Then we should consider the I/Os during the boot and login stage, which are less user related and more determined by the virtual desktop type itself. Since these 5,000 virtual desktops finishes booting and login within a small time period according to Section 4.4, I/Os during boot and login stages from different time zones do not overlap. But they occur periodically. According to our model, as described in Section 4, we know the rate of data read on replica will keep around 3 GB/s and rise to 3.3 GB/s at maximum. The primary disk will see a stable read rate of 350 MB/s and stable writes at 600 MB/s. The remote repository will stay at 70 MB/s of reads. We consider the most intensive I/O access period here in order to give the best guarantee. In total, this company will see approximately 4 GB/s of sustained data access periodically. The total capacity requirement can also be calculated to be 66 TB. Therefore, we need a storage system that has at least 4 GB/s of throughput and 66 TB of capacity. In this case, HP 3PAR T400 cannot meet this throughput requirement, and the HP 3PAR T800 may be a good choice.

TABLE 9: Specifications of 4 HP 3PAR Storage Systems

HP 3PAR Storage	F200	F400	T400	T800
Max Throughput	1300 MB/s	2600 MB/s	2800 MB/s	5600 MB/s
Max IOPS	46,800	93,600	128,000	256,000
Max Capacity	128 TB	384 TB	400 TB	800 TB
Drive Types	50GB SSD 300 & 600 GB FC 2TB NL	50GB SSD 300 & 600 GB FC 2TB NL	50GB SSD 300 & 600 GB FC 2TB NL	50GB SSD 300 & 600 GB FC 2TB NL

If we inspect the I/O throughput on each virtual disk, we can find most of the load is on the replica and the throughput requirement on the primary disk and remote repository is not that high. Therefore, we can deploy different targets on different storage, and we suggest using tiered storage to satisfy the storage requirements with minimum cost. We know there is a high throughput requirement on replicas, and I/Os are read dominant, so it is a perfect match to deploy replicas on SSDs. We call this group of SSDs Tier-1 storage. Compared with the replica, I/Os on the primary disk are more balanced. Considering that the workload is more write intensive on the primary disk, SSDs do not help as much (assuming SSD writes are slower than SSD reads). To save money, we can place primary disks and the remote repository on HDDs. For better performance, we can use high performance HDDs, e.g., 15K RPM HDDs. We call these HDDs Tier-2 storage. In this case, we can configure an HP 3PAR T800 into tiered storage, which is able to migrate data between tiers automatically to reduce the number of disks needed. Therefore, in order to satisfy the storage requirements of this company's 5,000 floating linked clones, we suggest buying an HP 3PAR T800 and configuring it as tiered storage to reduce cost.

6 Discussion and Future Work

Current VDI design is built on virtual machines. Recently, a lightweight virtualization technology call containers [5], [22] has become widely accepted in industry. Applications can run in containers just like in traditional VMs. Containers on the same host will share the same operating system kernel. Each container can have its own libraries, binaries, and namespace. They are segrated on the same host. From the host's point of view, each container runs as a process. On the other hand, VMs run on the hypervisor, a specialized OS, upon which each VM will run a full copy of an operating system. Consider a situation where existing resources cannot meet the storage requirements of VDI (e.g., if the company in the previous section where using an HPE 3PAR F400 to run 5,000 floating linked clones). Instead of suggesting an immediate hardware upgrade, we are exploring a possible remedial solution to migrate virtual desktops from VMs to containers, e.g., Docker containers [2], [31]. We will now present some preliminary results and analysis of this idea.

We first collect traces of a virtual desktop built on a Docker container. Booting such a virtual desktop is the process of creating and running a new Docker container. During the boot stage, a total of 18.09 MB of reads and 7.85 MB of writes are generated to boot the container. The reads are mainly due to loading OS data, the Docker daemon and runC (underlying Docker runtime technology) [8]. The writes during boot mainly come from adding a writable ”container layer” on top of underlying image layers. In this virtual desktop implementation, the application is configured to run immediately after the container is up. The application inside this container generates 80.08 MB of reads during boot. It is obvious that reads and writes during the boot stage of a virtual desktop inside a container are far fewer than those of a floating linked clone. This is because booting a container is exactly the process of forking a process on the host. There is no need to load all OS data from replicas as floating linked clones. Writes are only for writing the thin writable container layer. In addition, all read I/Os are eliminated during subsequent boots from that virtual desktop because the OS caches the data, and containers share the system cache with the host.

The login stage of virtual desktops inside containers has similar I/O patterns as in VMs. It reads user profiles to authenticate users and configure desktop settings. User profiles can be stored in a local Docker union file system or in volumes provided by underlying storage. According to the current virtual desktop implementation in containers, all user profiles are permanently stored. There is no need to first load them from a remote repository.

Therefore, it is worthwhile to consider replacing some of the floating linked clones in VMs with virtual desktops using Docker containers in the situation we are looking at. First, virtual desktops in containers can maintain the flexibility of floating linked clones. In our experiments, it only takes 1.45 seconds for the Docker daemon and runC to finish booting a virtual desktop in a container, while it takes 39.80 seconds to boot a floating linked clone in a VM. This nearly instant boot time makes deleting a virtual desktop after a user logs off and booting another when a user logs in cost far less than the same process using floating linked clones in VMs. Second, during the boot process, there are far fewer reads and writes than when using floating linked clones. If it is not the first time running a virtual desktop, data are already cached, and reads can be eliminated. While containers have advantages over VMs, virtual desktops using containers are not yet as mature as using VMs, and only some independent open source projects can be found. Containers themselves also have security limitations as containers share the same OS. Containers also do not support data persistence as well as VMs do, but storage companies are working on this issue. Docker is open source and under rapid development, so the acceptance of virtual desktops using containers will continue to increase.

7 Related Work

7.1 VDI and Its Enhancement

Currently, there are multiple VDI solutions such as VMware Horizon View [13], Microsoft Virtual Desktop Infrastructure [1], RedHat Enterprise Virtualization (RHEV) VDI [7] and Citrix Xen [20]. No matter which solution is chosen, storage performance is a big hurdle. VMware stated that over 70% of performance issues are related to storage. There are multiple storage solutions aiming to improve storage performance for VDI. VMware uses content-based read cache(CBRC) [14] to improve performance by caching common disks in the ESX host server. Unlike VMware CBRC, which restricts cache access to the same host, Infinio builds a distributed version of host side cache [3]. Another solution from PernixData utilizes server flash to accelerate VDI performance [6].

7.2 VM Characterization and Storage Requirements

I/O workload characterization [23], [25], [28], [33], [34] has been an important topic for storage researchers. Traditionally, the I/O workloads are collected from physical servers. Recently, more and more researchers have focused on the uniqueness of VM workloads. Tarasov et al. studies the extent to which virtualization is changing existing NAS workloads [35]. Gulati et al. presents a workload characterization study of three top-tier enterprise applications using the VMware ESX server hypervisor [26]. There are also characterizations based on other workloads including cloud backends. Mishra et al. try to characterize the workload of Google compute clusters [32]. Their goal is to classify workloads in order to determine how to form groups of tasks (workloads) with similar resource demands. Although these studies have some characterization of VM I/O behavior, they do not quantify I/O demands from the perspective of satisfying storage requirements.

Most of the studies trying to provide methods of meeting VM requirements overlook the characteristics of the VM storage requirements. Gulati et al. [27] do study how to improve I/O performance of VMs, but they only use Iometer to generate some workloads, thus cannot fully represent the storage requirements of VMs. Le Thanh Man et al. [30] study how to minimize the number of physical servers needed while ensuring Service Level Agreement (SLA) requirements. They place the virtual desktops whose access patterns have low correlation coefficient on the same servers, so the opportunity for CPU contention in the same servers is low. However, they mainly focus on CPU. The storage in VDI has its own characteristics and configurations, thus requiring a special discussion.

The manuals of commercial VDI products and the underlying storage are based on either rules of thumb to guide storage provisioning [19] or test the performance of their storage array given a fixed number of VDI instances [10]. Some products specifically try to meet VM storage requirements. VMware's vSAN [18] is such a product, and VMware has already integrated it with vSphere [17]. vSAN shows the available storage capabilities and claims they can be used to handle the storage requirements of VMs. When deploying VMs, administrators choose among existing storage and place VMs into the selected storage. However, such a description of VM storage requirements using storage capabilities can be inaccurate.

Another product that tries to meet VM storage requirements is Common Provisioning Group (CPG) [21] from HPE. It pools underlying physical devices into a unified storage pool called a CPG. VMs can draw resources from CPGs, and volumes are exported as logical unit numbers (LUNs) to hosts. CPG gives a detailed organization of the underlying storage. It tries to meet storage requirements of VMs by organizing underlying storage resources, but not from the VM perspective.

8 Conclusions

In this paper, we create a model to identify the storage requirements of one prevalent virtual machine type, VDI. We populate the parameters of our proposed model with real traces. Using our model, we demonstrate an example of how data accesses vary with time on different virtual disks for different types of virtual desktops. We further validate the usefulness of our model and show we can identify more accurate and fine-grained storage requirements of VDI than current industry methods. Based on the storage requirements identified and the bottlenecks determined, we are able to better guide administrators in their configuration of a storage system to meet the storage requirements with minimum resources.

Acknowledgment: We thank Jim Diehl for reviewing and revising the paper. This work is partially supported by the following NSF awards: 1305237, 1421913, 1439622 and 1525617.

References

[1] Desktop virtualisation. https://www.microsoft.com/en-in/cloud-platform/desktop-virtualization. Accessed: 2017-10-01.

[2] Docker documentation. https://docs.docker.com/. Accessed: 2017-8-14.

[3] Infinio. http://www.infinio.com/. Accessed: 2017-9-29.

[4] Iomark workloads. http://www.iomark.org/content/workloads. Accessed: 2018-3-31.

[5] Lxc. https://help.ubuntu.com/lts/serverguide/lxc.html. Accessed: 2017-9-29.

[6] Optimize vdi with server-side storage acceleration. http://pernixdata.com/sites/default/files/resources/PernixData_Optimize_VDI_WP_0.pdf. Accessed: 2017-9-29.

[7] Red hat enterprise virtualization for desktops. https://www.redhat.com/f/pdf/rhev/RH_WP_RHEVDesktops_web.pdf. Accessed: 2017-9-29.

[8] runc. https://github.com/opencontainers/runc. Accessed: 2017-8-14.

[9] Server and storage sizing guide for windows 7 desktops in a virtual desktop infrastructure. https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/technicalnote/view/server-storage-sizing-guide-windows-7-technical-note.pdf. Accessed: 2017-9-29.

[10] Sizing and best practices for deploying vmware view 5.1 on vmware vsphere 5.0 u1 with dell equallogic storage. http://en.community.dell.com/dell-groups/dtcmedia/m/mediagallery/20219029/download. Accessed: 2017-9-29.

[11] Spc-1 and spc-1e benchmark results. http://spcresults.org/benchmarks/results/spc1-spc1e. Accessed: 2018-3-18.

[12] Spc benchmark 1/energy extension official specification. http://www.storageperformance.org/specs/SPC-1_SPC-1E_v1.14.pdf. Accessed: 2017-10-01.

[13] Vdi virtual desktop infrastructure with horizon. https://www.vmware.com/products/horizon-view. Accessed: 2017-10-01.

[14] View storage accelerator in vmware view 5.1. https://www.vmware.com/files/pdf/techpaper/vmware-view-storage-accelerator-host-caching-content-based-read-cache.pdf. Accessed: 2017-10-01.

[15] Vmware horizon 6 storage considerations. https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper/vmware-horizon-view-mirage-workspace-portal-app-volumes-storage.pdf. Accessed: 2017-10-01.

[16] Vmware view planner installation and user's guide. https://my.vmware.com/web/vmware/details?downloadGroup=VIEW-PLAN-300&productId=320. Accessed: 2017-9-29.

[17] Vmware vsphere. https://docs.vmware.com/en/VMware-vSphere/index.html#com.vmware.vsphere.doc/. Accessed: 2017-10-01.

[18] vsan. https://www.vmware.com/products/vsan.html. Accessed: 2017-10-01.

[19] Vmware virtual san design and sizing guide for horizon view virtual desktop infrastructures. https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/whitepaper/products/vsan/vmw-tmd-virt-san-dsn-szing-guid-horizon-view-white-paper.pdf, 2014. Accessed: 2017-10-01.

[20] Citrix virtual desktop handbook 7.x. https://support.citrix.com/article/CTX221865, 2017. Accessed: 2017-9-29.

[21] Hp 3par storeserv storage concepts guide. http://h20564.www2.hpe.com/hpsc/doc/public/display?docId=c04204225, 2017. Accessed: 2017-9-29.

[22] D. Bernstein. Containers and cloud: From lxc to docker to kubernetes. *IEEE Cloud Computing*, 1(3):81–84, 2014.

[23] P. Carns, K. Harms, W. Allcock, C. Bacon, S. Lang, R. Latham, and R. Ross. Understanding and improving computational science storage access through continuous characterization. *ACM Transactions on Storage (TOS)*, 7(3):8, 2011.

[24] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, pages 273–286. USENIX Association, 2005.

[25] C. Delimitrou, S. Sankar, K. Vaid, and C. Kozyrakis. Accurate modeling and generation of storage i/o for datacenter workloads. *Proc. of EXERT, CA*, 2011.

[26] A. Gulati, C. Kumar, and I. Ahmad. Storage workload characterization and consolidation in virtualized environments. In *Workshop on Virtualization Performance: Analysis, Characterization, and Tools (VPACT)*, 2009.

[27] A. Gulati, C. Kumar, and I. Ahmad. Modeling workloads and devices for io load balancing in virtualized environments. *ACM SIGMETRICS Performance Evaluation Review*, 37(3):61–66, 2010.

[28] W. He, D. H. Du, and S. B. Narasimhamurthy. Pioneer: A solution to parallel i/o workload characterization and generation. In *Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on*, pages 111–120. IEEE, 2015.

[29] V. Infrastructure. Vdi server sizing and scaling. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.134.7129&rep=rep1&type=pdf. Accessed: 2017-10-01.

[30] C. L. T. Man and M. Kayashima. Virtual machine placement algorithm for virtualized desktop infrastructure. In *2011 IEEE International Conference on Cloud Computing and Intelligence Systems*, pages 333–337, 2011.

[31] D. Merkel. Docker: lightweight linux containers for consistent development and deployment. *Linux Journal*, 2014(239):2, 2014.

[32] A. K. Mishra, J. L. Hellerstein, W. Cirne, and C. R. Das. Towards characterizing cloud backend workloads: insights from google compute clusters. *ACM SIGMETRICS Performance Evaluation Review*, 37(4):34–41, 2010.

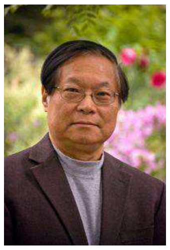
[33] C. Muelder, C. Sigovan, K.-L. Ma, J. Cope, S. Lang, K. Iskra, P. Beckman, and R. Ross. Visual analysis of i/o system behavior for high-end computing. In *Proceedings of the third international workshop on Large-scale system and application performance*, pages 19–26. ACM, 2011.

[34] S. Sankar and K. Vaid. Storage characterization for unstructured data in online services applications. In *Workload Characterization, 2009. IISWC 2009. IEEE International Symposium on*, pages 148–157. IEEE, 2009.

[35] V. Tarasov, D. Hildebrand, G. Kuenning, and E. Zadok. Virtual machine workloads: the case for new benchmarks for nas. In *FAST*, pages 307–320, 2013.

[36] A. Velte and T. Velte. *Microsoft virtualization with Hyper-V*. McGraw-Hill, Inc., 2009.

Hao Wen received the BS degree in computer science and technology from Huazhong University of Science and Technology, Wuhan, China, in 2013. He is working towards the PhD degree in computer science at the Department of Computer Science and Engineering, University of Minnesota, Twin Cities. His research focuses on storage QoS, software defined storage, Docker, Kubernete and data deduplication.



David H.C. Du is currently the Qwest Chair Professor in Computer Science and Engineering at the University of Minnesota-Twin Cities and the Director of the NSF I/UCRC Center for Research in Intelligent Storage (CRIS). He received his B.S. from National Tsing-Hua University (Taiwan) in 1974, M.S. and Ph.D. from University of Washington, Seattle in 1980 and 1981 respectively. His current research focuses on intelligent and large storage systems, cyber-physical systems, and vehicular/sensor networks. He is an IEEE Fellow (since 1998) and serves on editorial boards of several international journals. He was a Program Director (IPA) at National Science Foundation (NSF) CISE/CNS Division from 2006 to 2008. He has served as Conference Chair, Program Committee Chair, and General Chair for several major conferences in database, security and parallel processing.

Milan Shetti is currently the General Manager at Storage Division of Hewlett Packard Enterprise since December 2017. He was the Chief Technology Officer of Data Center Infrastructure Group at HPE He has an extensive background in the storage industry and has contributed to its industry standardization efforts. He joined HP as part of HP's acquisition of IBRIX in late 2009. Prior to this, he served as the President and CEO of IBRIX, setting strategic direction and presiding on all day-to-day operations of the company. Previously at Sun Microsystems, he held a number of business and technical leadership positions, including Technical Director for file system and Data Management group in the Network Storage division.

Doug Voigt is currently the Distinguished Technologist of Hewlett Packard Enterprise. With over 30 years experience in HP's storage business he is heavily involved with storage strategy, architecture and intellectual property. He has provided technical leadership, organizational guidance and planning for numerous protocol, implementation, architecture and advanced development projects in disk and disk array product lines. His career includes 7 years experience in disk controller protocol development, 17 years of experience in disk storage management automation, disk array development and distributed array architecture. Most recently he has 7 years experience in storage technology and IP evaluation. Highlights of his contributions include work on Service Oriented Storage, Quality of Service, IPv6, Storage Utility, Storage Consolidation, Federated Arrays, HP AutoRAID, Attribute Managed Storage, Disk Controller Fault Tolerance, SCSI and other disk protocol standardization and implementation efforts. He serves as Vice Chairman of Storage Networking Industry Association. He currently has 25 US patents, primarily in the field of virtual arrays with 11 patents pending. He holds MS and BS degrees computer science and electrical engineering respectively from Cornell University.

Shanshan Li received the MS and PhD degrees from the School of Computer Science, National University of Defense Technology, Changsha, China, in 2003 and 2007, respectively. She was a visiting scholar at Hong Kong University of Science and Technology in 2007. She is currently an assistant professor in the School of Computer, National University of Defense Technology. Her main research interests include distributed computing, social network, and data center network. She is a member of the IEEE and the ACM.