1. INTRODUCTION

As the rapid development of virtualization technology, traditional data centers are gradually superseding physical machines with virtual machines (VMs) to provide services. Virtual machines hosting different applications (e.g., database server, mail server) run on the same physical host. Apart from improving hardware utilization, virtualization enables migrating applications seamlessly 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 stored on shared storage. This enables a virtual machine, irrespective of the physical machine it is hosted on, to access its data over the network from the storage system. Since in the shared storage architecture, multiple VMs/applications will compete with each other for IO resources and capacity resources of the storage system, it is essential to provide minimum guarantees of quality of service (QoS) for VMs.

Any design trying to provide QoS guarantees to VMs all rely on an accurate demand description of these VMs. Such demands are often defined as VM policy. Current researches and products, however, either assume we already have this kind of policy or rely on the administrator to know and give everything. Most of those researches trying to provide QoS guarantee solutions have no idea about the characteristics of QoS demands of VMs. In order to give an accurate or at least practical VM demand description, we first need to understand the IO behaviors of VMs in data center. Some researches just use general or simulated workloads to describe VM IO behaviors based on which they try to give QoS guarantees. Considering the variety of applications running in VMs, VMs will show diverse behaviors. Therefore any work trying to provide a general QoS guarantee solution for all VMs will be inappropriate. Without knowledge of the exact application running in your VMs, trying to provide QoS is aimless. However, there are innumerable kinds of VMs, it is impossible to characterize IO behaviors of every VM. Fortunately, there is one type of VM that is prevalent as well as complicated—Virtual Desktop Infrastructure (VDI)[15, 32]. VDI runs desktop operating systems and applications inside VMs that reside on servers in the data center. Desktop operating systems inside VMs are also referred to as virtual desktops. When sizing VDI, the guidance provided by enterprises simply assume that administrator knows everything, including QoS requirements of VDI, which is actually not the fact. Because a single VDI application is already complicated enough, it is impossible for the administrator to describe the QoS requirements of all VMs on each underlying storage component at each time without analysis and characterization based on trace. Therefore, in this paper we focus on VDI to describe how to characterize QoS demands of VMs.

Current VDI sizing work [12, 7, 8] are unable to give accurate QoS demand description of virtual desktops. They either use rules of thumb to guide storage provisioning [11] or test the performance of their storage array given a fixed number of VDI instances [8]. Therefore, to ensure performance in practice, people always over provision storage resources, which definitely causes a huge amount of waste. Otherwise, however, VDI users will be in risk of degraded performance. Our objectives are guaranteeing QoS of VMs with minimal storage resources. It relies on a mathematical model based on real trace to describe storage QoS demands, e.g., capacity, throughput, latency etc. The model can describe QoS demands 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 bottlenecks are. And then based on this solid ground, we can tell whether a storage system can meet QoS requirements given a VDI setting and what minimal storage resources are required in order to meet such QoS requirements.

Accurately describing QoS demands of VMs relies on an appropriate model. Creating a mathematical model to describe real-life system is very challenging. The complexity of VDI even makes creating such model more challenging. First, VDI has multiple virtual desktop types. So the model should adapt to both homogeneous and heterogeneous combination of virtual desktops. Second, each desktop will undergo several stages during its life cycle. Different virtual desktops will undergo different stages. Even at the same stage, the behavior of each virtual desktop type is also different. So the model should consider different stages. Third, each virtual desktop will access multiple different data disks at different stages during life cycle. Those data disks have different functions and IO access patterns. The number, type and IO patterns of data disks are also different for different virtual desktop types. Fourth, when trying to integrate multiple VMs together in the model, it should realize that multiple VMs will arrive at a different time. At a specific time, VMs may stay at different stages. Therefore, the model should take into considerations IO accesses on different data disks of different virtual desktop types arriving at different time and now are in different stages.

In this paper, we analyze the IO behaviors of VMs in VDI by building model based on real traces that we collect. We install commodity VDI in our lab environment and collect IO traces of VDI used by students in four labs sharing our cluster. We analyze IO behaviors of different virtual desktop types and extract typical IO behaviors of each virtual desktop type from multiple traces. We analyze and derive IO parameters of each virtual desktop type at different stages during VM life cycle. The QoS demands are then generated by plugging IO parameters into the model and running simulators based on the model. The model can fit to any VDI environment. Researchers and enterprises can easily collect traces of their own environment and plug into our model to generate their own QoS requirements.

Our findings can be summarized as follows:

- We collect and analyze the trace describing the life time of different types of virtual desktops in VDI. We extract important IO parameters from the traces.

1 can blend this para with the previous second...
• We propose a system model to describe the storage QoS requirements of both homogeneous and heterogeneous VDI VMs. By plugging the IO parameters into the model, we generate the QoS requirements of VDI and derive bottlenecks on specific target data disks at a specific time. Best to our knowledge, this is the first model to describe IO behaviors of real life VDI system.

• We are able to give suggestions on VDI configuration and storage provisioning based on the QoS requirements we derive and the capability of storage system.

The outline of this work is as follows: §2 provides detailed background of VDI and presents the uniqueness of VDI. In §3, we propose our system model. §4 analyzes the data from trace and the QoS requirements and bottlenecks generated from the model. §5 applies our model to commodity storage system and analyze the generated results. §6 presents related work. §7 concludes the paper.

2. BACKGROUND
Virtual Desktop Infrastructure (VDI) is a virtualization solution to provide desktop environment to users. VDI deploys desktop operating systems on virtual machines and presents desktops as the normal ones to users. A user can use client devices such as personal desktops, tablets, mobile phones to connect to and operate their virtual desktops. Different from remote desktop, virtual desktop in VDI is presented as a service to users. All management are done at the data center. Users don’t have to manage provision, installation and update of the desktop.

Currently, there are mainly two VDI products: VMware Horizon View and Citrix XenDesktop. Without loss of generality, we will use VMware Horizon View as a paradigm to describe basics of VDI. In VDI, all virtual desktops can be referred as clones. Virtual desktops can be classified into different clone types according to how data are shared among them. After virtual desktops are provisioned, they should be assigned to users when users try to access them. The assignment can either be dedicated or floating according to the policy of assignment. In the following of this section, we will first describe different clone types and their features. Next, we will introduce how virtual desktops are assigned to users. Finally, we will describe what are the storage architecture of virtual desktops under different clone types and assignment types.

2.1 Clone Type
Different clone types have different IO access patterns and hence have different QoS requirements. There are mainly two types of virtual desktop clones. One is full clone virtual machine. Full clones are clones of a master image or parent VM. It creates its own virtual disk from all data of parent VM. The other type is linked clone virtual machine. There is also a master image or parent VM in linked clone architecture. But what is different is linked clones will share same OS data as long as they are linking to the same replica. Replicas are clones of master image. Each replica will serve as a common base for a group of linked clones. Linked clones each have unique identities and can be operated exactly like a physical desktop.

Linked clones have several advantages. The most outstanding one should be space savings. Since different linked clones link to the same replica, they don’t need to include these common data into their linked clone disks. Therefore, with only customization data contained, linked clone can have as much as 70 percent less capacity requirement than full clones. The second is highly efficient provisioning, in forms of patch management and desktop restore. The desktop restore is achieved by refresh operation. It will restore a VM to the snapshot created after initialization. Refresh operation is able to clean redundant and garbage data in virtual disk to spare more storage capacity. If linked clones are configured to refresh after each session, users will always be assigned a brand new virtual desktop after login. In full clones, since each virtual desktop is individual, updates and patches should be applied to each desktop one by one (A third party software is needed if you want to manage update centrally). However, in linked clone, one can apply patches and updates to a group of linked clones automatically and easily through recompose operation. To recompose linked clones, you first install patches and updates to the master image or you can make a new VM which contains the patches and updates be the master image. Then you take a snapshot of the updated master image. Then you can initialize the recomposition and select the snapshot as the new replica for a group of linked clones. Linked clone OS disk or primary disk (a virtual disk of linked clone) will be recreated and some important system data to keep linked clone functionality will be copied to linked clone primary disk. Finally these linked clones are anchored to the new replica. After each recompose operation, patches and update are applied to a group of linked clones automatically. The third benefit is management flexibility. In linked clone, management people are free to refresh, recompose or even delete virtual desktops since user profile and user data are stored in separate datastores. If you configure to delete user data after user logs off, linked clones will only contain system data, which will be safer to do maintenance.

2.2 Virtual Desktop Assignment
Different virtual desktop assignment will also influence IOs on storage and hence influence QoS requirements. There are mainly two types of virtual desktop assignment: dedicated assignment and floating assignment. Dedicated assignment (The corresponding virtual desktop is known as persistent or stateful desktop) will assign virtual desktops exclusively to certain users. It means each time when users request virtual desktops, they will always be assigned the same VMs. Both full clones and linked clones can be assigned dedicatedly. Floating assignment (The corresponding virtual desktop is also known as non-persistent or stateless desktop) will assign virtual desktops arbitrarily to users. Each time when users login, they are actually logging into different virtual desktops. Only linked clone can be assigned floatingly. Obviously, floating assignment has higher flexibility and is easier to manage, but it also brings additional cost on persona and user data management.

2.3 Storage Architecture of VDI
Combining virtual desktop type with assignment, we have floating linked clone, dedicated linked clone and dedicated full clone. These different combinations lead to different storage access patterns and provision requirements. Hereby we will detail the storage architecture under these three virtual desktop types.

Floating Linked Clone Storage Configuration. The storage architecture is shown in figure 1. Each linked clone is linking to a replica. So in the linked clone pool, we should spare space for replicas. Linked clone itself should also have a primary disk containing essential system data to remain linked to the base image and to function as a unique desktop. These disks will see different access patterns during boot, login and steady state stages. When a virtual desktop is boot, shared OS data has to be read from the replica first. These data will be loaded into VM memory to initiate system boot. Some of them will be written to the linked clone primary disk as well for future system access. After the VM is boot, it waits for users to login. Since this kind of linked clone is floatingly assigned, each time when a user logs in, he may be assigned a different virtual desktop on a different host. Since floating linked clones are usually configured to delete user data after logoff, user profiles and user data management will affect login process significantly. Each time when user logs in, the fresh desktop must be configured with user’s profile, e.g. authentication information, desktop theme, environment settings and application local settings. So the user can believe it is his own desktop, although it may not be the same virtual desktop. User’s profile are preserved in a remote repository independent of virtual desktop. Each user has his own repository. Typically, it’s a NAS file system. So when user logs in, the virtual desktop must load user profile from remote repository to memory first to authenticate the user and configure the desktop settings. Some of these data will also be written to linked clone primary disk for future access. Once the user successfully logs in, the virtual desktop goes to the steady state, which means the user starts...
his routine operation, e.g. editing documents, watching videos, web browsing, reading etc. Some of these operation needs to access user data, which mainly contains user’s own documents, videos, photos, music etc. stored in the remote repository as well. These data are not downloaded to local desktop at a time. They are copied to primary disk only when first accessed. All subsequent accesses are actually to the copies on primary disk. Any changes to the user data will be synchronized to the remote repository at regular intervals.

**Dedicated Linked Clone Storage Configuration.** The storage architecture is show in figure 2. Dedicated linked clone also has replicas, primary disks and remote repository as floating linked clone. But what is different is in dedicated linked clone pool, you can configure a separate persistent disk to store user profile and user data. This disk is dedicated to users. So by attaching a persistent disk to a linked clone virtual desktop, it makes that virtual desktop dedicated to the user. Each time when a user logs in, he will always be assigned the same virtual desktop which has his persistent disk attached. During the boot process, there is no need to load OS data from replica anymore as long as it’s not the first boot of a fresh desktop. During the login process and steady state, user profiles and user data are read from the persistent disk rather than the remote repository. The synchronization happens between the persistent disk and remote repository. Actually, the persistent disk performs as a cache of remote repository.

**Full Clone Storage Configuration.** Full clone is like our regular desktop. All accesses only happen on one virtual disk type during all stages. OS data, user profile, user data are stored in this virtual desktop type.

We summarize in table 1 when each data disk will be accessed for each type of virtual desktop.

**Tiered Storage in VDI.** In VDI storage architecture, there are multiple data disks. The performance requirements on these data disks are different. For replica disk, since shared OS data need to be read from this disk by all the linked clones linking to this replica, it may see an intensive data access during boot. Compared with replica disks, primary disks and persistent disks may see fewer data access. In floating linked clone, most IO happen on primary disk after boot. In dedicated linked clone, since user profile and user data are cached in persistent disk, persistent disk may see a higher IO demand than primary disk. So organizing these data disks into different tiered storages may fit best to their IO demands. Just as an example described in figure 1 and figure 2, we build 2 tiered storages above different storage array.

The model is to answer at time $t$, how much data will be read from each data disk and how much data will be written to each data disk. We will try to generate a mathematical model for a single VM in VDI first.

**3.1 Single VM Life Circle**

In order to model a large number of VMs in VDI, we need to figure out the IO behaviors of a single VM during the life cycle. Overall, the VM life cycle can be divided into several stages: boot, login, steady state and logoff.

**Boot:** When users are starting work, they have to boot their virtual desktops first. If these desktops are powered on at the same time, or concentrating within a small time period, it becomes a storm. They will contend for the system resources simultaneously, e.g. replicas of linked clone will be read a lot.

**Login:** 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. In this stage, user profiles will be loaded to enable identity authentication and desktop configuration. If these user profiles are required to be loaded within a short time period, either persistent disk (dedicated linked clone) or network bandwidth (floating linked clone) need to have enough capacity.

**Steady State:** After users log in, they start their everyday work. It’s should be very random, except periodical synchronization with NAS in linked clones. And for floating linked clones, there will also be additional IO caused by downloading user data from remote repository to clones when they are accessed.

**Logoff:** This is the final stage during the VM life cycle. A final synchronization with remote repository should happen for linked clone. Although the logoff process will not affect user experience of the user, these additional IO will be directed to the storage system after all and hence affect other users.

**3.2 Single VM Model**

The model is to answer at time $t$, how much data will be read from each data disk and how much data will be written to each data disk. We will try to generate a mathematical model for a single VM in VDI first.

In order to know how much data are read and written, we need these information: the target of each IO request, read/write ratio, IO size, percent of each IO size and the number of IOs at time $t$. Since we know the exact data disks that each type of clones will access during each different stages of life cycle, the target can be replica, primary (primary disk), persist (persistent disk), NAS or full (full clone disk). We use $R_{stage,target}$ and $W_{stage,target}$ to represent the read ratio and write ratio during different stages on different targets. The subscript stage can be boot, login, steady and logoff. The IO size $S_{stage,target}$ are several discrete values. Since there are many different IO sizes, here we will only choose several significant IO sizes at each VM life cycle stage on each target. The percentage of each significant IO size can be denoted by $P_{size,stage,target}$. The expected
number of IOs at time $t$ could be described by a statistical function $E_{\text{stage,target}}(t)$ which tells how many IOs are expected to arrive at target on stage at time $t$. With these parameters, we can calculate that at time $t$ the size of data read from target at VM life cycle stage is:

$$
\sum_i E_{\text{stage,target}}(t) \times dt \times R_{\text{per,stage,target}}
\times S_{\text{stage,target}} \times P_{\text{size,stage,target}}.
$$

(1)

And the size of data written to target at VM life cycle stage is:

$$
\sum_i E_{\text{stage,target}}(t) \times dt \times W_{\text{per,stage,target}}
\times S_{\text{stage,target}} \times P_{\text{size,stage,target}}.
$$

(2)

By multiplying a small time interval $dt$, we can get how many IOs are expected to come at time $t$. When plugging in IO sizes and the percentage of selected IO sizes, we select those significant IO sizes. An IO size is significant when it accounts for most of the IOs. It has two factors: 1) the frequency of the IO size is high. 2) The total size of data transferred under this IO size is large. Some IO size may have high frequencies but since its IO size is very small, the total size of data transferred under it may be neglectable. On the other hand, some IO size have transfer large size of data, but since its frequency is too small, representing other IOs with it will lead to mistakenly large IOs. Therefore, both of these two factors need to be considered. We first sort IO sizes in descending order under criteria of percentage and size of transferred data respectively. We then give these two ranks weights $\alpha$, $\beta$ and the significance of an IO can be calculated by:

$$
S_{\text{size}} = \alpha \times \text{rand under percentage}
\times \beta \times \text{rand under size of data transferred}
$$

(3)

### 3.3 Multiple VMs Model

From the single VM model, we can get at time $t$ how much data are read from and how much data are written to each data disk. When integrating a number of VMs in VDI, more factors need to be considered. 1) VMs will start to boot or arrive at different time. 2) IO behaviors of different virtual desktop types are different. 3) IO behaviors of VMs running different operating systems, user applications are different. The VM arrival rate is key to integrate multiple VMs. Since in data center, VMs will arrive at different time rather than arrive simultaneously, the integration should also follow such arrival distribution. We use a function $g(n, t)$ to describe the probability that there are $n$ VMs arriving at time $t$. Different virtual desktop types will determine which data disks are target in equation (1) and (2). Along with the operating system types and user applications, it will also affect the arriving IOs at time $t$ as well as significant IOs. We will discuss how to deal with these factors under VDI environment.

#### A. Assumptions

In order to make the multiple VMs model simple and practical, we make several assumptions here. First, VMs of different types all follow the same arrival rate, regardless of virtual desktop types, operating systems and user applications. Second, VMs of the same type have the same arrival IO distribution, significant IOs during boot, login and steady state and will spend the same amount of time to issue IO requests during boot and login. Here VMs of the same type means the virtual desktop type, operating systems and user applications are the same. Third, login follows boot immediately and there are no intervals. From the following sections, we will see how these assumptions are applied into the model.

### B. Multiple VMs of the Same Type

If multiple VMs are of the same virtual desktop and have the same operating system type and user applications, their parameters are all the same. Therefore, we only need to consider how to add the IO requests of VMs at different stages. Here we take one data disk as an example. Let’s say targets-primary disk. And we only calculate how much data are read from the target at time $t$. This should be the integration of 4 parts: IOs from VMs arriving at time $t$, IOs from VMs still in boot process but not finish yet, IOs from VMs in login process and IOs from VMs in steady state. IOs from VMs arriving at time $t$ can be calculated by

$$
N(t) \times \sum_i (\lambda_{\text{boot,primary}} \times dt \times R_{\text{per,boot,primary}}
\times S_{\text{boot,primary}} \times P_{\text{size,boot,primary}}).
$$

(4)

$N(t)$ indicates the number of VMs arriving at time $t$. $\lambda_{\text{boot,primary}}$ is the expected number of IOs per time unit of one VM. By multiplying $dt$ (1s, 1 minute etc.) we can expect how many IOs will come at time $t$. $R_{\text{per,boot,primary}}$ is the read percentage of IOs during boot stage on primary disk. $S_{\text{boot,primary}}$ are significant IO sizes during boot stage on primary disk and $P_{\text{size,boot,primary}}$ are the corresponding percentage of significant IO sizes.

IOs from VMs still in boot process but not finish yet at time $t$ can be calculated by

$$
\sum_{x=t-t_0}^{t} [N(x) \times \sum_i (\lambda_{\text{boot,primary}} \times dt \times R_{\text{per,boot,primary}}
\times S_{\text{boot,primary}} \times P_{\text{size,boot,primary}})]
$$

(5)

According to assumption 2, these same VMs will spend the same amount of time to issue IO requests during boot. So here we assume it takes $t_0$ to finish issuing IOs for all these same VMs during boot process. Therefore, during time interval $[t-t_0,t]$, we can find those VMs still in boot process. So what we need is for every time point in $[t-t_0,t]$, calculating formula (4) and adding them together.

IOs from VMs in login process at time $t$ can be calculated by

$$
\sum_{x=t-t_0}^{t} [N(x) \times \sum_i (\lambda_{\text{login,primary}} \times dt \times R_{\text{per,login,primary}}
\times S_{\text{login,primary}} \times P_{\text{size,login,primary}})]
$$

(6)

According to assumption 2, these same VMs will spend the same amount of time to issue IO requests during login. And according to assumption 3, login follows boot immediately and there are no intervals. So here we assume it takes $t_1$ to finish issuing IOs for all these same VMs during login process. Therefore, during time interval $[t-t_0-t_1,t-t_0]$, we can find those VMs in login process. So what we need is for every time point in $[t-t_0-t_1,t-t_0]$, calculating the size of data read during login and adding them together.

IOs from VMs in steady state at time $t$ can be calculated by

$$
\sum_{x=t}^{t-t_0} [N(x) \times \sum_i (IOPS_{\text{steady}} \times dt \times R_{\text{per,steady,primary}}
\times S_{\text{steady,primary}} \times P_{\text{size,steady,primary}})].
$$

(7)
So here we assume all VMs have an initial time start point start. Any VMs arrived before $t - t_0 - t_1$ are now in steady state. So what we need is for every time point in $[\text{start}, t - t_0 - t_1]$, calculating the size of data read during steady state and adding them together. Since in steady state arrival of IO requests doesn’t follow Poisson Process any more, we use an average value $IOP_{steady}$ to represent the number of IOs per time unit.

The size of data written to target at time $t$ can be calculated in the same form as formula (4)-(7). We only need to modify them according to formula (2) by replacing $R_{per}$ with $W_{per}$ and replace read significant IOs with write significant IOs.

Since a data disk will not be accessed during all stages, so when calculating the total size of data read/written on target at time $t$, we only need to choose those related formula according to 1. For example, if our virtual desktop type is floating linked clone and we want to know how much data are read from replica, we only need to consider the boot stage by applying only formula (4) and (5). If a data disk is accessed during all stages, then we have to apply formula (4)-(7). By selecting different targets, we can get size of data read at all data disks. By traversing all time $t$, we can get how the IO requests on a target varies with time. Therefore, we can have a clue about when the bottleneck will happen on each target.

### C. Multiple VMs of Different Types

Before we analyze how to integrate IOs from VMs of different types, let’s discuss how different types will affect the IO behaviors of VMs during the life circle.

1. **Different virtual desktop types.** As we have already discussed, the data disks that VMs will access are different for different types of virtual desktops. It is summarized in table 1. We can see when we consider a particular target, we only need to consider IOs from part of these 3 virtual desktop types. On the other hand, even some data disks may be accessed by multiple virtual desktop types at a particular stage, it doesn’t mean the IOs parameters of them are the same on these data disks. For example, both floating linked clone and dedicated linked clone will access primary disk during steady state, but compared with floating linked clone, dedicated linked clone will access primary disk less than floating linked clone. In reality, it accesses persistent disk mostly during steady stage. So we do need to consider the influence of different virtual desktop types on IOs on the same data disk at the same stage.

2. **Different operating systems.** During the boot stage, virtual desktops need to read operating system data. Different operating system types mean the IO requests composition can be different and the arrival of IO requests can also be different. Not only in boot stage, since the login process also needs to load some system related configuration data, so operating system types could also make a difference at login stage. (3) **Different user applications.** User applications related to user data. Since user data are accessed during login and steady state stage, so different user applications may cause different IOs during login and steady state stage.

Since operating system types, virtual desktop types and user application types will affect IO behaviors of VMs during the life time, we define three sets to describe these factors:

$$OS = \{OS_1, OS_2, ..., OS_n\}$$

$$VD = \{FLC, DLC, FC\}$$

$$APP = \{app_1, app_2, app_p\}$$

We also use a set VM to describe the types of VMs that will affect IO behaviors during the life time. And we define the types of VMs be the Cartesian product of set OS, VD and APP. For each data center, the OS set, VD set and APP set may be different according to specific configurations. So the elements in VM set are also different.

So, $VM = OS \times VD \times APP$

Each element in VM set indicates one VM type. For each VM type, we will choose from formula (4)-(7) and also the formulas of the write version to calculate how much data are read from and written to each corresponding target at time $t$. Here we will discuss with the read

### 4. DATA ANALYSIS

In order to get IO parameters in our model, we collected traces of different VM types in VDI during boot, login and steady state respectively. We install commodity VDI-VMWare Horizon View in our lab environment. We create 100 virtual desktop instances and let students in our four labs which share clusters access them in their daily routine. We collect IO traces of VDI when they are in boot, login and steady state stages. We analyze IO behaviors of different virtual desktop types and extract typical IO behaviors of each virtual desktop type from multiple traces. In this chapter, we will first analyze the burst of traces in order to describe the expected number of IOs happened in our model. Then we will detail the description of parameters in the model. Finally a simulation result will show the burdens of IO on each target.

#### 4.1 Burst of Requests

We discover that the IO requests of VMs in VDI show a burst pattern. So it is not appropriate to describe the expected IOs by statistical models, like famous Poisson Process. Figure 3 depicts the inter arrival time of IOs of floating linked clone. Other clone types show similar burst features. For those sparse requests whose inter arrival time are greater than 50ms, we simply truncate them to 50ms. We can see no matter whether it is dense or sparse, IOs on all targets are burst. Further, we assume that requests arrive uniformly within bursts. Since the actual arrival pattern happens during very short time interval ($\mu$s) and we are only interested in IOs in measure of seconds. So we can simply plug in those bursts in our model. The expected number IOs at time $t$, if $t$ falls into a burst, is the average number of IOs of that burst. Otherwise, it is 0.

#### 4.2 Traces of a Single Virtual Desktop

In this section we will derive the parameters that we will use in the model by analyzing traces. The traces include the IO requests sent to different targets by different types of virtual desktops at boot, login and steady state stages.

**A. Floating linked clone**

During the boot process of floating linked clone, reads are dominant on replica, accounting for 99.8% of total IOs on replica. And the total size of data read are 188MB. On contrary, writes become dominant on primary disk, accounting for 99.9% of total IOs on primary disk. And the total size of data written are 20MB. This is because when a floating linked clone is boot, it needs to load OS data from shared replica first. Some of them will be written into primary disk for future use. The significant IO sizes and corresponding percentages are listed in figure 2.

During the login process of floating linked clone, IOs happen on
writes account for 30.5%, with totally 7.3MB size of read and 0.5MB size of write. And on NAS, reads account for 69.5% and writes account for 92.2%. Reads account for 7.8%. And totally 1.6MB size of write. We find IOs happen only on primary disk. Writes are dominant, but we can see the write ratio here is a little high. This is because the reads on NAS are mostly reading user profile. Since in our experiment, user does not have so many data, the size of read on NAS are relatively small. And since there are also some inevitable writes by VDI system (e.g., log) on NAS, so the percentage of read in our trace is not so dominant. But we believe as user has more and more profile data, reads will become more dominant on NAS during login. The significant IO sizes and corresponding percentages are listed in figure 3.

B. Dedicated linked clone

The boot process of dedicated linked clone is quite different from floating linked clone. Although dedicated linked clones are still linking to replicas and do need to load OS data into primary disk, it only happens at the first time of boot. Since dedicated linked clones will preserve data in primary disks rather than refresh them as floating linked clones after users log off, so in most cases dedicated linked clones do not need to load OS data again from replica during boot process. So here we only consider the most general cases where primary disks already preserve OS data that are needed during boot process. We find IOs happen only on primary disk. Writes are dominant, accounting for 92.2%. Reads account for 7.8%. And totally 1.6MB size data are read and 14.6MB size data are written. The significant IO sizes and corresponding percentages are listed in figure 4.

Dedicated linked clones utilize persistent disks to preserve user profile and user data. So IO behaviors of dedicated linked clone are a little bit different from floating linked clone in login process. On primary disk, reads account for 48.9% and writes account for 51.1% with totally 22.1MB size of read and 20.4MB size of write. On persistent disk, reads account for 24.6% and writes account for 75.4% with totally 1.5MB of read and 2.4MB of write. On NAS, reads account for 32.2% and writes account for 67.8% with totally 1.5MB of read and 0.7MB of write. The difference here from floating linked clone is some IOs are now shifted to persistent disk and IOs on NAS are reduced a lot. This is because we do not need to load so much data from NAS to primary disk when users log in, and they are already there in persistent disk. So we can directly read user profile from persistent disk to proceed with login process. Communication cost is reduced. The significant IO sizes and corresponding percentages are listed in figure 5.

C. Full clone

Full clone only has one virtual disk. All IOs happen on that virtual disk. So different from linked clones where IOs are separated to different disks, IOs are aggregated in full clone disk. During the boot process, reads account for 42.2% and writes account for 57.8% with totally 80.4MB of read and 16.6MB of write. During login process, reads account for 69.0% and writes account for 31.0% with totally 44.2MB of read and 8.6MB of write. The significant IO sizes and corresponding percentages are listed in table 5.

Table 2: Significant IOs of Floating Linked Clone during Boot

<table>
<thead>
<tr>
<th>Replica</th>
<th>Primary</th>
</tr>
</thead>
<tbody>
<tr>
<td>IO Size/KB</td>
<td>Percentage</td>
</tr>
<tr>
<td>128</td>
<td>20.5%</td>
</tr>
<tr>
<td>4</td>
<td>8.9%</td>
</tr>
<tr>
<td>16</td>
<td>8.5%</td>
</tr>
<tr>
<td>256</td>
<td>4.0%</td>
</tr>
</tbody>
</table>

Table 4: Significant IOs of Dedicated Linked Clone during Boot

<table>
<thead>
<tr>
<th>Primary</th>
</tr>
</thead>
<tbody>
<tr>
<td>IO Size/KB</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>20</td>
</tr>
<tr>
<td>132</td>
</tr>
</tbody>
</table>

4.3 Simulation Results

With traces collected for each type of VM, we now can aggregate multiple VMs together. Here we did experiments on multiple virtual desktops arriving at different time and see how the IOs on each target vary with time.

Experiment Setup

We assume a company uses VDI for work use and it has 5000 virtual
desktop instances. Assume the arrival of employees follows Poisson distribution and the arrival rate is 10 per second. These 5000 virtual desktops can be all floating linked clone, all dedicated linked clone, all full clone or a mixture of all types. Next we will show under each of these occasions, how much data are accessed on replica, primary disk, persistent disk, NAS and full clone disk respectively since the first user arrives.

### A. Floating Linked Clone

Floating linked clones mainly access replica and primary disk during boot; primary disk and NAS during login; primary disk during steady state. So overall, replica, primary disk and NAS are accessed. Figure 4 shows the size of data accessed on each of these targets from the first virtual desktop arrival (or start to boot) till all virtual desktops transiting to steady state.

On replica, as in figure 4(a) the IOs are read dominant and quite heavy. The size of data read could rise sharply to around 1000MB/s within the first 30 seconds. In the next around 500 seconds, the workload is relatively stable and keeps high around 1000MB/s. The largest rate could be 1400MB/s. And once there is virtual desktop finishing boot, the IOs start to drop dramatically within the next 20 seconds. So overall, replica is read intensive for floating linked clone. Data read per second could be gigabytes for the above 5000 virtual desktops company.

Different from replica, the IOs on primary disk in figure 4(b) are more balanced, since it is accessed during all stages. Because data read from replicas will be written to primary disk during boot and some writes also happen on primary during login, primary disk could still see a large volume of IOs. The size of data written will keep dominant and stable around 1000MB/s. Once all VMs finish booting and transits to login, the writes will drop sharply and get close to the size of data read. That’s the feature of IOs during login shown in trace section 4.2. The size of data read keeps stable mostly. And finally, all VMs go to steady state and IOs are relatively small.

As shown in figure 4(c), size of data accessed on NAS is quite small. Since during login process, user profile needs to be downloaded from remote NAS, so more reads happen on NAS. Although the IO size here is pretty small, it will increase as more applications are installed.

### B. Dedicated Linked Clone

Dedicated Linked Clone mainly access primary disk during boot; primary disk, persistent disk and NAS during login; primary disk and persistent disk during steady state. So overall, primary disk, persistent disk and NAS are accessed. Figure 5 shows the size of data accessed on each of these targets from the first virtual desktop arrival (or start to boot) till all virtual desktops transitioning to steady state.

The IOs on primary disk of dedicated linked clone are much lighter compared with floating linked clone, as seen in figure 5(a). The size of data written will rise to around 300MB/s and keeps for 500 seconds. Once all VMs finish boot and login, the IOs on primary disk are minimal. That’s unique for dedicated linked clone is persistent disk. Since all user profiles are cached in persistent disk and will be accessed mostly during login, we can see in figure 5(b) reads and writes mainly rise during the login stage. On NAS, IOs still mainly happen during login stage as shown in figure 5(c). A little bit difference here is reads are much fewer that that of floating linked clone, because most user profile are cached in persistent disk. So IOs on NAS are reduced.

### C. Full Clone

Full clone only has one type of virtual disk. So all IOs happen on that disk regardless of the stages. Figure 6 shows the size of data read and written on full clone disk for the above 5000 virtual desktops company. We can see the total size of read is greater than the total size of write from VMs arrival till VMs transiting to steady state. And there is an obvious stage of high IOs where VMs are in boot and login stage. The IOs will drop down suddenly when all VMs finish booting and then drop to minimum slowly when all VMs transit to steady state.

### D. A mixture of different clones

Since we already know how much data are read and written on each target for each type of virtual desktop, we are interested to see what is the IO pattern on each target if those 5000 virtual desktops are a mixture of all different clone types. Here we simulate a combination of 3:6:1 for floating linked clone, dedicated linked clone and full clone. This ratio is reasonable. Because linked clones have advantages of space saving and ease of management, they should be deployed much more than full clones. And since dedicated linked clones will save a lot of IOs that happens on floating linked when loading OS data from replica to primary disk and loading user profile and user data from NAS to primary disk, it should be a common deployed virtual desk-

---

**Table 3: Significant IOs of Floating Linked Clone during Login**

<table>
<thead>
<tr>
<th>Size/KB</th>
<th>read Percentage</th>
<th>read Size/KB</th>
<th>read Percentage</th>
<th>read Size/KB</th>
<th>read Percentage</th>
<th>read Size/KB</th>
<th>read Percentage</th>
<th>read Size/KB</th>
<th>read Percentage</th>
<th>read Size/KB</th>
<th>read Percentage</th>
<th>read Size/KB</th>
<th>read Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>8</td>
<td>4</td>
<td>12</td>
<td>8</td>
<td>126</td>
<td>8</td>
<td>23.0%</td>
<td>8</td>
<td>132</td>
<td>8</td>
<td>38.6%</td>
<td>8</td>
<td>50</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>4</td>
<td>12</td>
<td>8</td>
<td>256</td>
<td>8</td>
<td>46.2%</td>
<td>8</td>
<td>132</td>
<td>8</td>
<td>43.6%</td>
<td>8</td>
<td>25</td>
</tr>
<tr>
<td>12</td>
<td>8</td>
<td>8</td>
<td>12</td>
<td>8</td>
<td>256</td>
<td>8</td>
<td>3.8%</td>
<td>8</td>
<td>132</td>
<td>8</td>
<td>43.6%</td>
<td>8</td>
<td>25</td>
</tr>
</tbody>
</table>

**Table 5: Significant IOs of Dedicated Linked Clone during Login**

<table>
<thead>
<tr>
<th>Size/KB</th>
<th>read Percentage</th>
<th>read Size/KB</th>
<th>read Percentage</th>
<th>read Size/KB</th>
<th>read Percentage</th>
<th>read Size/KB</th>
<th>read Percentage</th>
<th>read Size/KB</th>
<th>read Percentage</th>
<th>read Size/KB</th>
<th>read Percentage</th>
<th>read Size/KB</th>
<th>read Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>8</td>
<td>4</td>
<td>12</td>
<td>8</td>
<td>126</td>
<td>8</td>
<td>23.0%</td>
<td>8</td>
<td>132</td>
<td>8</td>
<td>38.6%</td>
<td>8</td>
<td>50</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>8</td>
<td>12</td>
<td>8</td>
<td>256</td>
<td>8</td>
<td>46.2%</td>
<td>8</td>
<td>132</td>
<td>8</td>
<td>43.6%</td>
<td>8</td>
<td>25</td>
</tr>
<tr>
<td>12</td>
<td>8</td>
<td>128</td>
<td>8</td>
<td>23.0%</td>
<td>16</td>
<td>7.3%</td>
<td>8</td>
<td>23.0%</td>
<td>16</td>
<td>7.3%</td>
<td>8</td>
<td>23.0%</td>
<td>16</td>
</tr>
<tr>
<td>256</td>
<td>1.5%</td>
<td>132</td>
<td>2.3%</td>
<td>1.5%</td>
<td>132</td>
<td>2.3%</td>
<td>1.5%</td>
<td>132</td>
<td>2.3%</td>
<td>1.5%</td>
<td>132</td>
<td>2.3%</td>
<td>1.5%</td>
</tr>
</tbody>
</table>

---

**Figure 6: Size of Data Accessed on Target of Full Clone**
top type. Figure 7 shows the size of data read and written on replica, primary disk, dedicated disk, NAS and full clone virtual disk.

The IOs on replica arise from boot of floating linked clones. So the size of data accessed shows a similar pattern in figure 1(a). Since the number of floating linked clones is only 3/10 of the 5000 virtual desktops, the size of IO is much fewer than that 5000 floating linked clone case. Compared with dedicated linked clone, floating linked clone has more intensive IO access on primary disk. So the pattern of size of data accessed on primary disk is more similar to that floating linked clone case. However, the same, the size of data accessed is much smaller. Persistent disk is accessed exclusively by dedicated linked clone. So its IO pattern is similar to figure 2(b). Although dedicated linked clone will access NAS, the IOs are reduced a lot due to existence of persistent disk. So the IO pattern on NAS is similar to floating linked clone. Finally, full clone disk is only influenced by full clone. So it should show the same pattern as in figure 4.

According to the above analysis, we can draw the conclusion that floating linked clone has more influence on different disk targets. Although they only account for half of the total dedicated linked clones, the IO pattern is more influenced by floating linked clone rather than dedicated linked clone or anything else. This is because the size of data read and written on floating linked clone is much more intensive than dedicated linked clone. The high volume of IOs arises from loading data from replica to primary disk during boot and loading data from NAS to primary disk during login. Dedicated linked clone is more IO efficient because it is stateful and reserve those data across boot and login.

5. EVALUATION

We have already created the model to answer how much data are read and written on each storage target in VDI at different virtual desktop stages. And the model is fed with data derived from collected traces. It can explain for both a single virtual desktop and multiple virtual desktops. This model is able to guide companies to configure VDI and provision storage for VDI. In this chapter, we will apply the model in two ways. One is showing how our model answers how many virtual desktop instances a commodity storage system can support. The other is answering whether a storage system can support a typical VDI. If not, what kind of storage configuration can meet VDI QoS requirements? And we will give suggestions on VDI configurations and storage provisioning based on the results. The storage system that we choose to evaluate is HP 3PAR StoreServ F400, whose maximal capacity is 384TB, maximal throughput is 2600MB/s.

5.1 Tune VDI

We will first show how to configure VDI in order to get QoS requirements met by a commodity storage system. We will answer how many virtual desktops StoreServ F400 can support in terms of storage capacity, throughput and IO latency respectively. In table 8, 9, 10, we show the average throughput of read and write of those 5000 virtual desktops configurations in section 4.3, based on which we will calculate how many virtual desktops can be supported by F400. In order to not influence performance a lot, we have to consider the most intensive IO access period, that is from VMs arrival till VMs transiting to steady state. If the requirements during this period can be met, the requirements during steady state are not problems. So the average throughput we calculate is the average throughput during this intensive IO access period. We also list the storage capacity requirement for each type VDI user. The number of virtual desktop instances can be supported $N_s$ is calculated by:

$$N_s = \frac{(Storage\_Capability - Shared\_NAS)}{Requirement} \quad (8)$$

The Storage_Capability is the maximal capacity, throughput capability of a storage system. The Requirement is the capacity, average throughput for VDI. For latency requirement, we are based on the SPC-1 [9] performance test on StoreServe F400 [10]. We expect the IO latency requirement to be 5ms or less.

<table>
<thead>
<tr>
<th>Table 8: Requirements of Floating Linked Clone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replica</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>R</td>
</tr>
<tr>
<td>Average KB/s</td>
</tr>
<tr>
<td>Capacity per VDI user if thin provisioned: 4GB.</td>
</tr>
<tr>
<td>Shared NAS: 1TB</td>
</tr>
</tbody>
</table>

Based on the storage capability of StoreServ F400 and requirements of each type of virtual desktops, we show in table 11 the num-
6.1 VDI and its enhancement

The fundamental technology of virtual desktop infrastructure is virtualization. There are multiple popular virtualization solutions, like VMware ESX Server [17, 18], Hyper-V [3], KVM [27] and Xen [22], upon which VDI solutions of VMware Horizon View [13], Microsoft Virtual Desktop Infrastructure [16], RHEV VDI [6] and Xen Citrix [1] are built respectively.

No matter which solution is, storage is a big hurdle on performance. 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 disk in ESX host server. So VMs can retrieve data from host memory which is loaded by other VMs earlier on the same host. IOs issued to storage can be saved a lot, and therefore the potential bottleneck on replica can be alleviated. We call this group of SSDs Tier-1 storage. For dedicated linked clone, IOs are minimal on replicas as long as it’s not the first boot of a fresh desktop. Primary disks will see a little bit high volumes of IOs during boot process. But it’s only 30% of floating linked clone. So we can still place primary disks on HDDs, Tier-2 storage. For full clones, since all IOs happen on one type of virtual disks and they are balanced, we can either place them on Tier-1 storage or Tier-2 storage according to the overall performance of other virtual desktops.

We can see no matter which virtual desktop type is, the number of VMs supported is limited by the IO latency rather than storage capacity or throughput. This is reasonable since it is common in modern storage systems to use deduplication and thin provisioning to reduce storage used. So for VDI application, storage capacity is typically redundant and not restrictive compared with throughput and latency. And latency is the most direct metric that influences user experience, which now is more and more restrictive. So latency becomes the one that determines the number of virtual desktops supported finally. And it is also interesting to see that dedicated linked clone is the least costly among three types in terms of resource utilization.

5.2 Tune Storage

Now we will show how to configure storage system in order to meet QoS requirements of a typical VDI system. Apparently, if a company has the above 5000 floating linked clones configuration, a single StoreServ F200 with full usage is unable to support VDI QoS requirements. Users will experience severe performance downgrade.

Then what is an appropriate storage configuration? For floating linked clone, we know there is a heavy load on replicas and it is read dominant. So it is perfect to deploy replica virtual disks on SSDs. The high IOPS of SSD can improve the speed of loading operating system data from replica a lot, and therefore the potential bottleneck on replica can be alleviated. We call this group of SSDs Tier-1 storage. Compared with replica, IOs happened on primary disk are more balanced. And considering it is more write intensive on primary disk, SSDs do not help much. So we can place primary disks in HDDs. For better performance purpose, we can use high performance HDDs, e.g. 15K HDD. We call it Tier-2 storage. For dedicated linked clone, IOs are minimal on replicas as long as it’s not the first boot of a fresh desktop. Primary disks will see a little bit high volumes of IOs during boot process. But it’s only 30% of floating linked clone. So we can still place primary disks on HDDs, Tier-2 storage. For full clones, since all IOs happen on one type of virtual disks and they are balanced, we can either place them on Tier-1 storage or Tier-2 storage according to the overall performance of other virtual desktops.

6. RELATED WORK
izes server flash to accelerate VDI performance [5]. It clusters high-speed server resources, like flash, into a logical pool of resources to accelerate reads and writes to primary storage.

6.2 VM characterization and QoS

IO workload characterization [34, 30, 23, 33, 21, 20, 29, 26] has been an important topic for storage. Traditionally, the IO workloads are collected from physical servers. Recently, more and more researches have focused on the uniqueness of VM workload. Tarasov et al. studied the extent to which virtualization is changing existing NAS workloads [31]. They observed a significant difference in workloads between physical machines and virtualized clients. Tarasov’s research gives a general characterization of the workload changes in VM environment. Moreover, VMs will show special workload features based on applications running in VMs. Gulati et al. presents workload characterization study of three top-tier enterprise applications using VMware ESX server hypervisor [24]. It shows VMs will have its own unique IO behaviors based on the workloads of applications running on it. There are also characterizations based on other workloads including cloud backend [28]. Mishra et al. tries to characterize the workload of Google compute clusters with the goal of classifying workloads in order to determine how to form groups of tasks (workloads) with similar resource demands. Although these researches have some characterization of VM IO behaviors, they do not quantify IO demands from perspective of guaranteeing storage QoS.

On the other hand, most of those researches trying to provide QoS guarantee solutions have no idea about the characteristics of QoS demands of VMs. Gulati et al. [25] did suggest how to improve performance of VMs. They model both VM workloads and characterize underlying devices. However, it just uses Iometer to generate some workloads, thus can’t represent the QoS of VM. Considering the variety of applications running in VM, VMs will show diverse behaviors. Therefore any work trying to provide a general QoS guarantee solution for all VMs will be inappropriate. Without knowledge of the exact application running in your VMs, trying to provide QoS are aimless. That’s why we only choose one VM type but prevalent and typical enough to model storage QoS demands and guide on QoS guarantee. Since VDI has enterprise products, it should have storage QoS demands characterization and QoS guarantee solutions in manuals of either VDI products or underlying storage. Unfortunately, they either base on rules of thumb to guide storage provisioning [11] or test the performance of their storage array given a fixed number of VDI instances [8].

There are some products trying to provide some extent of QoS to VM. VSAN [19] is such a product of VMware and has already been integrated with vSphere [18]. Virtual SAN virtualizes local physical storage resources of ESXi hosts and turns them into storage pools that can be carved up and assigned to virtual machines and applications according to their QoS requirements. Use storage policies to define virtual machine storage requirements such as performance and availability in the form of a profile. Virtual SAN lays out the virtual disk across the Virtual SAN datastore to meet the specified requirements based on VM storage policy and storage capabilities. VSAN monitors and reports on the policy compliance and takes remedial action if necessary. With VSAN a limited QoS guarantee is provided. The method that VSAN tries to provide QoS is very rough. Actually they fail to profile VM requirements. What they do is just make a reference to the storage capabilities and claim that as the storage requirements of VM. Such profile doesn’t consider the unique characteristics of VM. They just say they know the storage requirements and can directly give numbers of each parameter that the VM requires.

Another product that can provide some extent of QoS to VM is CPG (Common Provisioning Group) [2] from HP. It pools underlying physical devices into a unified storage pool called CPG. VMs can draw resources from CPGs, and volumes are exported as logical unit numbers (LUNs) to hosts. CPG gives a detailed organization of underlying storage, but fails to consider the VM side. They don’t consider any characteristics of VMs. Only one side of QoS guarantee can’t make a good match between VM and underlying storage.

7. CONCLUSIONS

In this paper, we create a model to describe the QoS requirements of one prevalent virtual machine type - VDI. We populate the parameters of the model with real traces that we collect. We derive QoS demands on each target data disk at different time. And we apply our model to commodity storage system to guide VDI configurations and storage provisioning.

8. REFERENCES


