

Image Management in a Virtualized Data Center

Tingxi Tan,^{*} Rob Simmonds and Bradley Arlt
Grid Research Centre, University of Calgary

Martin Arlitt and Bruce Walker
HP Labs

ABSTRACT

Industrial research firms such as Gartner and IDC are predicting an explosion in the number of online services in the coming years. Virtualization technologies could play an important role in such a world, as they create an opportunity to provide services in a cost-effective manner. However, to achieve ideal savings, more dynamic environments must be created, with Virtual Machines (VMs) being provisioned and altered on-the-fly. Management issues arise when using these elastic resources at scale. In this study, we provide an initial investigation of performance and scalability issues for image management in a virtualized data center. Results provided show that the choice of storage solution and access protocol matters. For example, our tests show the time to start a VM from a local hard drive under I/O intensive workload increases by a factor of 15 and for certain shared storage options, this factor increases to 30 times.

Keywords

virtualization, image management, performance, scalability

1. INTRODUCTION

Major shifts in the way Information Technology (IT) functionality is produced and consumed are looming. Software providers are beginning to provide their products as services from the Internet “cloud”. The Web 2.0 and social networking paradigms have made the creation and use of services accessible to a greater audience. While users benefit from increased choice, IT providers are faced with greater challenges. For example, the popular social networking site Facebook allows developers to easily expose their applications to a large audience. This has attracted over 25,000 applications (as of May 22, 2008). The lifespan and usage level of these applications varies greatly, and may change over time. Many of these applications are ideally suited for Virtual Machines (VMs), that can be dynamically adjusted to meet the current workload.

With the flexibility of VMs come additional challenges, particularly if the scale of virtualized data centers we envision is realized. For this reason, we have begun to examine the performance and scalability of different VM management tasks. Our initial examination indicated issues related to the storage systems used to support the virtualized environment. In this paper, we provide some preliminary results that demonstrate how the choice of storage matters in designing a scalable image management solution. In particular, we find that for I/O intensive workloads, high performance shared storage solutions are highly desirable in comparison to local hard

disk storage. For example, our tests show the time to start a VM from a local hard drive under I/O intensive workload increases by a factor of 15. Furthermore, we demonstrate that for certain shared storage options, this factor increases to 30 times. Therefore the choice of shared storage also matters.

The rest of the paper is organized as follows. Section 2 discusses related projects. Section 3 provides background information on virtualization and storage solutions. Section 4 describes the methodology and tools we used to perform our experiments, while Section 5 presents our results. Section 6 concludes our paper with a summary of our work, and examines topics for future research.

2. RELATED WORK

The use of virtualization in data centers is not a new concept. Numerous open source tools and commercial products already exist. Similarly, tools are available for managing virtualized IT environments. For example, Platform Computing offers their Virtual Machine Orchestrator (VMO)¹, and VMware offers their VirtualCenter² management product. Both provide user-friendly interfaces for customers to administer their own virtualized environments. In academia, tools such as Shirako [1], Usher [3] and Maestro-VC [2] enable dynamic provisioning of computational resources, with scientific users as their initial audience. To the best of our knowledge, none of these have quantified performance and scalability issues related to the storage systems and image management.

The creators of Xen³ proposed a distributed filesystem called Parallax [5] in an attempt to address image management issues. The goal of that research project was to provide a unified view of physically distributed hard disks, global block-level snapshots and copy-on-write semantics. However, their prototype implemented a limited set of features and performance results are not publicly available.

3. BACKGROUND

3.1 Virtualization

Virtualization technologies enable physical resources to appear as multiple logical resources. In our study, Xen provides the virtualization capabilities. The Xen virtualization platform creates, manipulates and monitors full and paravirtualized VMs. A Xen hypervisor runs as native software on the hardware platform of a physical machine. It provides

¹<http://www.platform.com/Products/platform-vm-orchestrator/>

²<http://www.vmware.com/products/vi/vc/>

³<http://www.xensource.com/>

^{*}contact author at txtan@cpsc.ucalgary.ca

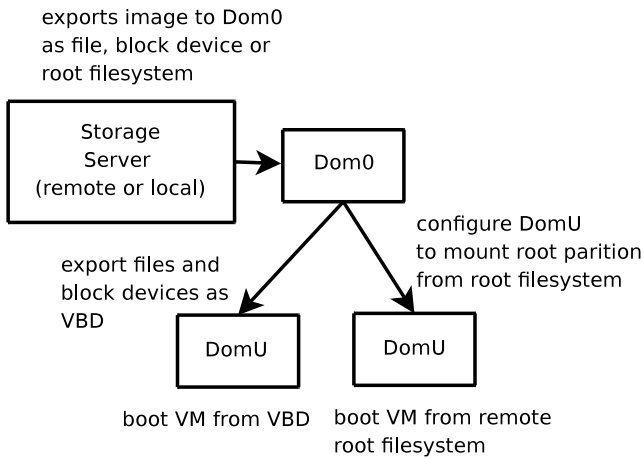


Figure 1: storage access for Xen VMs

special control privileges to a management domain (Dom0) which in turn is allowed to instantiate and control the life-cycle of multiple guest domains (DomU). A DomU is a VM created by exporting an image to it. An image is an operating environment that consists of a complete filesystem with an operating system and a custom or generic software stack. It may or may not include the kernel, ramdisk or additional application storage space. There are a variety of storage options for providing an image; we discuss these options in Section 3.2. In addition, a *prestine* or *non-prestine* VM may be required. A prestine VM is instantiated from a new or untainted image. Such a VM is desirable for some situations, such as running applications that utilize confidential data. Providing a prestine environment is more costly, as each VM requires its own copy of an image. Non-prestine VMs can be provisioned more quickly, by re-using existing images. Some users may prefer the lower cost of these, if they also have less strict privacy requirements.

3.2 Storage Solution

A VM image can be stored locally or remotely as a file, a block device, a logical volume, a root partition, or a combination of these. Xen supports multiple methods in accessing VM images (Figure 1). Files and block-based images are translated to Virtual Block Devices (VBD) before being exported to a VM. Alternatively, VBDs can be bypassed by using network-based storage directly from within the VM. Two popular IP-based network storage solutions are NFS (Network File System) and iSCSI. NFS is a filesystem protocol that allows a file server to export directories (NFS shares) to remote client machines. The communication between the server and client uses Remote Procedure Calls (RPC). Recent versions (NFS v3 and v4) communicate over TCP by default if both the client and server support it. iSCSI allows SCSI commands to be exchanged over TCP/IP. This avoids the need for expensive Fibre Channel connectivity to a SCSI storage device. Readily available high speed interconnects like 1 and 10 Gb/s Ethernet makes iSCSI an attractive low-cost alternative for a network-based storage solution. A detailed performance comparison of NFS and iSCSI is available in [4]. Both protocols provide similar functionalities, and are examined in our study.

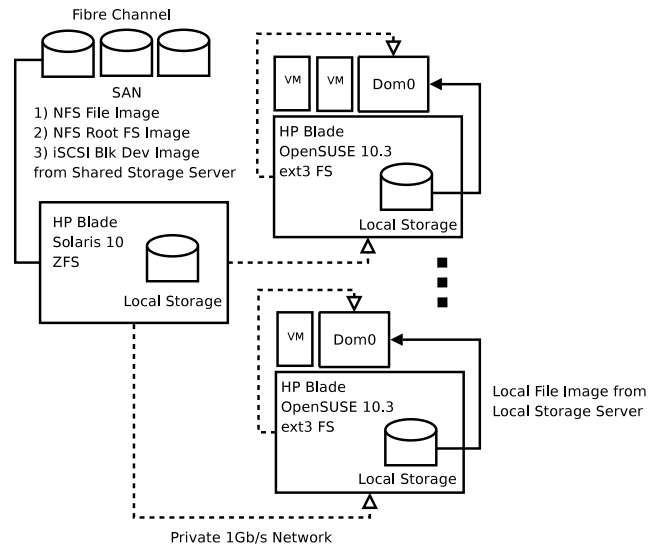


Figure 2: our experimental environment

4. METHODOLOGY

4.1 Experimental Environment

Our experimental testbed is shown in Figure 2. It includes six HP ProLiant BL465c G1 blade servers. Each blade has two AMD 2216 HE (dual-core, 2.4 GHz) CPUs, 8 GB RAM, a 73 GB SCSI local disk, and two 1 Gb/s NICs. Four of the blades are used to run VMs that generate additional load on the shared components of the test environment. Each of these has Xen 3.1 installed, with the OpenSUSE 10.3 Linux operating system. The fifth blade also has Xen 3.1 installed. We utilize this blade to measure the boot time of the LAMP VM (described in more detail in Section 4.3.1). Images as raw files are cached on the local disk of this blade, thus the fifth blade itself is considered a storage server when we evaluate the **Raw File from Local Disk** storage option. When evaluating this option, both the load generating VMs and the LAMP VM will be started on the same blade. We implemented a shared storage server on the sixth blade server. This server runs Solaris 10 update 4 (no virtualization on this server). This server is connected to a high performance SAN (HP EVA 8100) which it uses as a backend store for all VM images. ZFS⁴ on Solaris makes it simple to make snapshots of images and export them as an iSCSI target or NFS share. Figure 2 also shows the differences between the local and shared storage server architecture.

We utilize *sar* for collecting CPU, memory, and disk utilization information on Linux and Solaris. For network utilization, we use *nicstat*, a Solaris freeware that polls *kstat* to present a time series of NIC utilization information.

4.2 Metrics

We utilize two metrics to evaluate the performance of the system under study. Our first metric is *boot time*; i.e., the time to boot a VM from an image. This is one method of quantifying the *cost* of bringing a VM online. As with booting a physical machine, the time required to boot a

⁴<http://opensolaris.org/os/community/zfs/>

VM depends on internal factors (e.g., the number of services started) and external factors (e.g., the CPU speed, amount of RAM). We note that the cost to start a VM could also include the initial distribution of the image to the storage servers. However this factor is decoupled from the cost as it is independent of the actual boot time. It could be added in future work if deemed necessary.

Our second metric is the *load* on the storage server. Load is defined as a combination of CPU and network utilization from the point of view of the storage server. We utilize this metric to understand the scalability of each shared storage solution.

4.3 Factors and Levels

We examine two factors in our study. The first is the type of VM image (Section 4.3.1), the second is the type of storage used (Section 4.3.2). We fix a third factor (the virtualization technology) for all tests. As described in Section 3, all of our experimental results are based on Xen version 3.1.

4.3.1 VM Image Options

We utilize four different VM images to create a variety of workloads on the system under study. All of the images are 1 GB in size, and are allocated 1 virtual CPU and 256 MB RAM. The first image has no active services, and thus serves as the baseline (idle) case. The next two images emulate a CPU intensive and I/O intensive VM respectively. The CPU intensive VM simply generates random numbers continuously. Once booted, the I/O intensive VMs start the *IOzone*⁵ benchmark and repeatedly runs the write test, with a file size of 256MB and a 4 KB record length. The fourth image contains a LAMP (Linux-Apache-MySQL-PHP) software stack. This image instantiates a web server VM from which we measure the boot time metric.

4.3.2 Storage Options

Our study evaluates four storage options for VM images. The levels we considered are: **1) Raw Files from Local Disk, 2) Raw Files through NFS, 3) Root Filesystem through NFS** and **4) Block Devices through iSCSI**. The two raw file options use the Xen DomU configuration parameter “*file*.” to mount the image as a loop device on Dom0. Xen creates a VBD from the loop device and exports it to DomU to boot the VM. The NFS root filesystem option uses the Xen DomU configuration parameters “*nfs_server*.” and “*nfs_root*.” to instruct DomU to mount its root partition from the specified NFS share. This method allows storage to be attached directly from within DomU. For block devices, an iSCSI target for the device is exported to Dom0, which in turn initiates the target as a local SCSI device. Using the configuration parameter “*phy*.”, Dom0 exports a VBD from the local SCSI device to DomU to boot the VM.

4.4 Methodology

All of our tests were conducted in a similar manner. The i^{th} load generating VM running in the environment will be booted sequentially on the $\lfloor \frac{i}{4} \rfloor^{th}$ blade. This means a maximum of four VMs were started on each blade, and up to a maximum of 16 load generating VMs on four blades. All

⁵<http://www.iozone.org/>

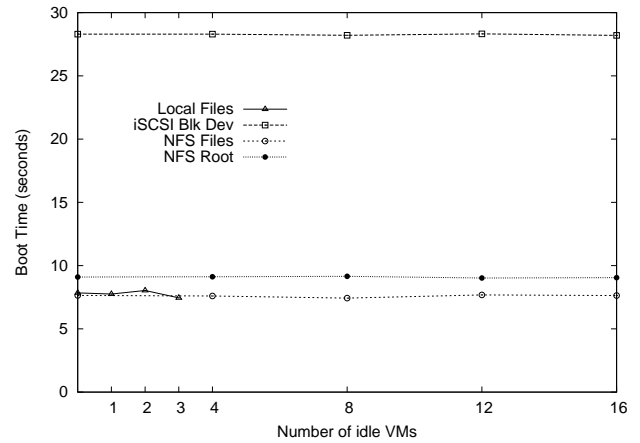


Figure 3: Boot times for LAMP VM with Idle Load

the load generating VMs running in the environment (for a particular test) would either be all idle, CPU intensive or I/O intensive VMs.

We measured the boot time of the LAMP VM on the fifth blade under a variety of load conditions. This corresponds to measuring the boot time when 0, 1, 2, 3, 4, 8, 12 or 16 load generating VMs were running in the environment. For the local disk test, only 0 to 3 load generating VMs were utilized and they all resided on the same blade that the LAMP VM was started on.

5. RESULTS

This section provides our experimental results. We examine the experiments according to the type of VM used to generate additional load on the system under study.

5.1 Idle VM Workload

Our initial experiment considers the time to boot a LAMP VM, when some number of idle VMs are present in the environment. Figure 3 shows no noticeable difference in boot times for all storage options as we increase the number of VMs. Result are also similar for the raw file from local disk, NFS files and NFS root options. In these configurations, the LAMP VM boots in under 10 seconds even as the number of idle VMs was increased from 0 onwards. Booting the LAMP VM over iSCSI took considerably longer, at around 28 seconds (we will speculate on the cause of the additional delay after we examine all of the results in Section 5.3).

5.2 CPU intensive VM workload

Next, we repeated the methodology utilizing the CPU intensive VMs to place additional load on the system under study. Figure 4 indicates that the additional load on each of the virtualized blade servers had little effect on the boot times of the LAMP server. When utilizing a file on local disk, the boot times for the LAMP VM remain around 7.5 seconds, the same as in the previous experiment. However, there is a small, but noticeable increase in these boot times, suggesting the CPU-intensive VMs had a minor effect on the results. Overall though, Xen appears to do a good job at providing performance isolation between the VMs running on each blade.

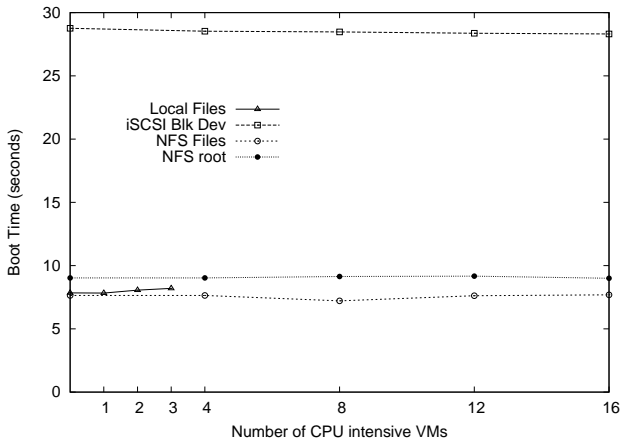


Figure 4: Boot times for LAMP VM with CPU intensive Load

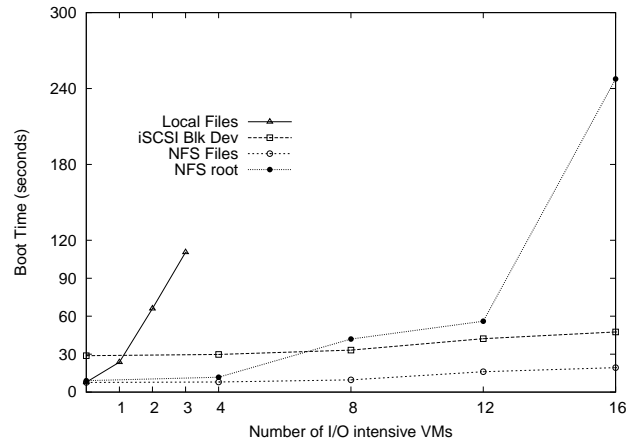


Figure 6: Boot times for LAMP VM with I/O intensive Load

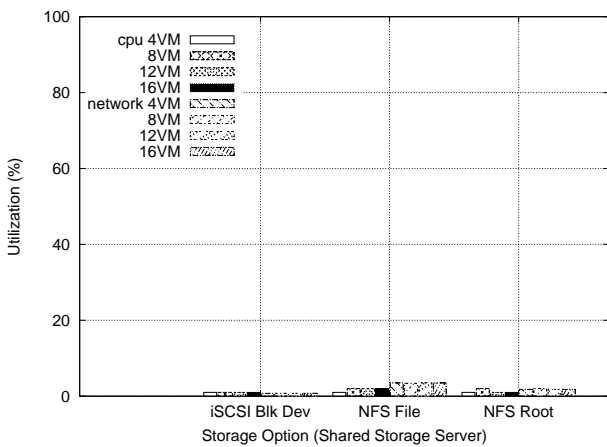


Figure 5: Resource Utilization for Shared Storage Servers under CPU intensive Load

Looking at the shared storage solutions indicates similar results to the idle case. All three options seem to scale well with the iSCSI case having the longest boot times.

Figure 5 shows the CPU and network utilization on the ZFS server during the iSCSI and NFS tests. Not surprisingly (given the workload is CPU intensive on the blades where the VMs are running), the utilization of both of these resources is relatively low, regardless of the number of VMs running in the environment.

5.3 I/O-intensive VM workload

Our final experiment examines the performance and scalability of the environment when the dominant workload in the environment is I/O intensive. Figure 6 shows the boot times for this experiment. The results are substantially different from the previous experiments. Looking first at the local file case, we see that the boot times of the LAMP VM quickly and rapidly increase. With one I/O intensive VM running, the LAMP VM boot time triples and with two I/O intensive VMs the boot time increases by a factor of 8. When three I/O intensive VMs are running, the LAMP VM boot time

increases by almost a factor of 15. The most likely cause of this delay is the local disk is overloaded. Figure 7 verifies this is the case. Figure 7 shows that with even one I/O intensive VM running, the local disk has an average utilization of around 90%. Although the CPU utilization is only 47% on average, about three quarters of this is spent waiting for I/O to complete (CPU iowait state). When the number of I/O-intensive VMs is increased to two or more, the local disk is fully utilized, and the time the CPU spends waiting for I/O increases further. Clearly, in environments with I/O intensive workloads, relying on a single local disks will limit scalability.

Figure 6 also provides the boot times for the tests where shared storage is utilized. From this figure we can see that the NFS file and iSCSI solutions work the best; although the boot time increases for both, it does so at a slow rate. Both are able to sustain the load placed on them by up to 16 I/O intensive VMs. Figure 6 also shows that the NFS root solution scales differently than the NFS file solution even though both options use the same underlying data sharing protocol. The boot times for both options remain relatively similar when up to four I/O intensive VMs are running in the environment. However, the NFS root case increases substantially when more I/O intensive VMs are added. With 16 I/O intensive VMs running, the boot time for the LAMP VM increases from 9 seconds to just over 240 seconds, a factor of 30 times slower.

To understand the different behaviours we observed for the I/O intensive experiment, we examine the resource utilization on the shared storage server. These results are shown in Figure 8. This figure reveals several key differences between the NFS file and NFS root options. First, the NFS root tests access network storage directly from within the VM. This fully utilizing the network with only four I/O intensive VMs running. Beyond this, the ZFS server is network-bound, and thus the boot time of the LAMP VM is delayed waiting for the image to be delivered.

The NFS file tests show that neither the ZFS server's CPU or network resources are fully utilized. This is because the Xen

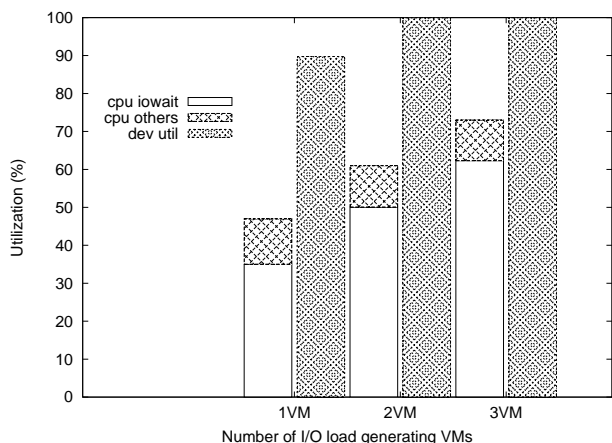


Figure 7: Resource Utilization for Local Storage Server

image storage access mechanism (see Section 4.3.2) mounts the NFS file as a local loop device. This allowed the image to be cached in the local RAM on Dom0, thus resulting in higher performance. However, the Linux loop device presents its own set of problems under memory pressure and should be used with caution in a production environment⁶. Xen’s Block Tap mechanism is an alternative to loop devices though our tests with this resulted in kernel crashes, which we believe is due to the immaturity of the toolset. The NFS file tests also shows the network is nearing full capacity in the 16 I/O intensive case. This indicates that the NFS file approach is also network-limited beyond 16 I/O intensive VMs. However, since the CPU utilization is still only 33%, the ZFS server could serve additional VMs if a larger capacity network link was used.

Lastly, the iSCSI results indicate similar scaling behaviour as NFS files. This suggests that the iSCSI protocol utilizes caching techniques similar to loop device caching. However, the iSCSI boot times were consistently 20 seconds longer when compared to NFS files and NFS root (up to a load of 4 VMs). One hypothesis is that the iSCSI driver in use requires tuning. Since the shared storage solutions all used the same underlying physical infrastructure, we have a good reason to suspect the driver, rather than an inherent performance problem with the iSCSI protocol. However, this hypothesis remains to be validated.

6. CONCLUSION

In the near future, data centers are likely to utilize virtualization technologies on much larger scales and in more dynamic ways than are done today. As a preliminary investigation into the performance and scalability issues that await, we examined the behavior of several different storage solutions with respect to a number of workloads. We also indicated how the types of workloads affect provisioning new resources.

Our results indicated that for I/O intensive workloads, the single local hard disk in our test node significantly restricted

⁶<http://lxr.xensource.com/lxr/source/tools/blktap/README>

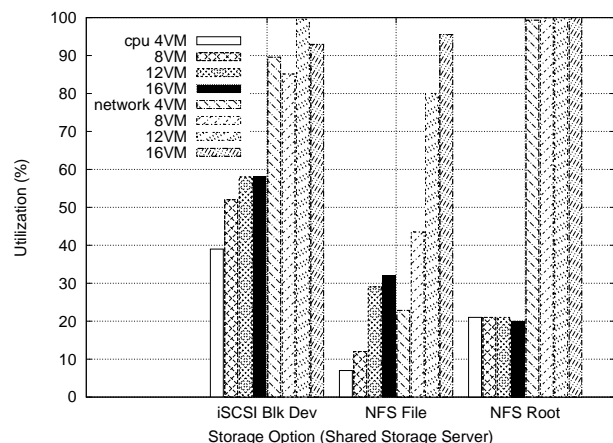


Figure 8: Resource Utilization for Shared Storage Server under I/O intensive Load

performance. Attaching a fast disk array overcomes the local disk bottleneck and could potentially reveal other bottlenecks that eluded us in this study. Shared storage solutions are more scalable in our measurements, although care is still needed in selecting an appropriate one.

From our findings, we believe that some non-obvious bottlenecks also exist. We intend to further analyze the performance of iSCSI in order to quantify its lower than expected performance (relative to NFS) within our experimental environment. We also plan to perform the measurements using a different virtualization technology. This will help us understand which performance issues are particular to Xen, and at the same time, demonstrate the portability of our measurement methodology to other virtualization tools

This is only the beginning of performance evaluation in this space. As noted earlier in our paper, we expect very large scale (e.g., 10,000s of VMs) environments; our work obviously looked at much smaller environments than this. We will continue to explore this and related topics, but obviously contributions from a wider audience are needed.

7. REFERENCES

- [1] D. Irwin, J. Chase, L. Grit, A. Yumerefendi, D. Becker, and K. Yocum. Sharing networked resources with brokered leases. *USENIX Technical Conference*, 2006.
- [2] N. Kiyancilar, G. Koenig, and W. Yurcik. Maestro-vc: A paravirtualized execution environment for secure on-demand cluster computing. *CCGrid*, 2006.
- [3] M. McNett, D. Gupta, A. Vahdat, and G. M. Voelker. Usher: an extensible framework for managing clusters of virtual machines. *Proceedings of the USENIX Large Installation System Administration Conference*, 2007.
- [4] P. Radkov, L. Yin, P. Goyal, P. Sarkar, and P. Shenoy. A performance comparison of NFS and iSCSI for IP-Networked storage. *Proceedings of USENIX Conference on File and Storage Technologies*, 2004.
- [5] A. Warfield, R. Ross, K. Fraser, C. Limpach, and S. Hand. Parallax: Managing storage for a million machines. *Proceedings of the 10th Workshop on Hot Topics in Operating Systems*, 2005.