# Challenges in Building Scalable Virtualized Datacenter Management

Vijayaraghavan Soundararajan and Kinshuk Govil
VMware, Inc.
3401 Hillview Ave.
Palo Alto, CA 94306
{ravi,kinshuk}@vmware.com

## ABSTRACT

Virtualization drives higher resource utilization and makes provisioning new systems very easy and cheap. This combination has led to an ever-increasing number of virtual machines: the largest data centers will likely have more than 100K in few years, and many deployments will span multiple data centers. Virtual machines are also getting increasingly more capable, consisting of more vCPUs, more memory, and higher-bandwidth virtual I/O devices with a variety of capabilities like bandwidth throttling and traffic mirroring.

To reduce the work for IT administrators managing these environments, VMware and other companies provide several monitoring, automation, and policy-driven tools. These tools require a lot of information about various aspects of each VM and other objects in the system, such as physical hosts, storage infrastructure, and networking. To support these tools and the hundreds of simultaneous users who manage the environment, the management software needs to provide secure access to the data in real-time with some degree of consistency and backward-compatibility, and very high availability under a variety of failures and planned maintenance. Such software must satisfy a continuum of designs: it must perform well at large-scale to accommodate the largest datacenters, but it must also accommodate smaller deployments by limiting its resource consumption and overhead according to demand. The need for high-performance, robust management tools that scale from a few hosts to cloud-scale poses interesting challenges for the management software. This paper presents some of the techniques we have employed to address these challenges.

## Categories and Subject Descriptors

C.4 [*Computer Systems Organization*]: Performance of Systems

## General Terms

Performance, Management, Measurement, Design.

## Keywords

Virtual Machine management, cloud computing, datacenter management, management workload

## 1. INTRODUCTION

One of the killer applications of virtualization is server consolidation. Many datacenters have a policy of running a single application per server, even if the application is mostly idle and the resulting physical server utilization is only 5% on average. Virtualization allows these low-utilization applications to run in isolated containers on a single physical host. For example, rather than employing 10 physical servers, each with a single application running at an average utilization of 5%, a single physical server can house the 10 applications in separate virtual machines and utilize 50% of capacity of physical servers. In addition to better efficiency, virtualization allows administrators to deploy VMs on existing hardware rather than purchasing new hardware every time new applications are deployed. The subsequent reduction in hardware has a number of implications: 1) fewer servers to manage, 2) fewer servers consuming power, and 3) less power needed for cooling.

The benefits of virtualization do not stem from just reduced hardware cost. Virtualization drastically simplifies the day-to-day life of a datacenter administrator in a variety of other ways: deploying virtual servers is vastly simpler than deploying physical machines, involving essentially a file copy followed by a customization script as opposed to racking up a physical server and connecting the server to power grids, networks, and storage endpoints. Keeping applications running during maintenance windows is also simpler with virtualization: an administrator can move virtual machines (VMs) while they are still running, fix the hardware they had previously been running on, and then move the VMs back, all without incurring application downtime. This can be significantly more difficult in a physical environment. Datacenter management software is responsible for these deployment and migration tasks, and the combination of high-performance virtualization and a feature-rich management layer has enabled a number of end-users to employ a 'virtualization-first' policy when it comes to provisioning new servers.

The design of the management layer is driven by the need to create features that simplify day-to-day operations. As datacenters grow in terms of number of servers and applications being managed, the management layer must provide these services at scale to meet increased demand. Virtualization further increases the scale of computation. For example, recent hardware advances have allowed more CPUs to be run on an individual socket [2][10] and more VMs to run per core. Higher-density DIMMs allows VMs with larger memory to run on a single machine. High-speed IO and convergent IO fabrics have allowed larger numbers of IO-intensive VMs to run on the same host. As a result, the datacenter is comprised of many more computing elements than in the past. For example, in 2003, the standard virtualization building block was a 2-way SMP host. Each CPU would run between 4-8 VMs, for a total of 8-16 VMs per host. With today's technology, we see 32 cores per host, and with 8-10 VMs per core, this leads to 256-320 VMs per host. With a standard 40-host rack, this leads to over 12,000 VMs in a rack vs. 640 VMs in a rack *circa* 2003. Moreover, the memory allocated for a VM has increased dramatically as well, from an average of around 256MB for server workloads in 2003 to 4-8GB today. As for I/O, 10Gbps networking is becoming the standard deployment per host (with 100Gbps on the horizon) and 8Gbps is seeing rapid adoption for connection to Fibre Channel arrays.

As the above numbers suggest, datacenters will potentially have hundreds of thousands of VMs in a compute farm, and the ability to quickly display relevant information to an administrator is vital. At these scales, the MTBF of components practically ensures that there will be constant failures, and the ability to hone in on these failures and signal them to the administrator is necessary. Finally, as more and more mission-critical applications run in VMs, the availability of this infrastructure becomes more important, so availability of both the applications and the management infrastructure is essential. As a concrete example, when customers employ desktops using Virtual Desktop Infrastructure (VDI), if the desktops are temporarily unavailable due to issues with the management infrastructure (especially at 9am when employees come in to work and access their desktops in a single burst), the result is a significant loss of productivity across an entire corporation.

In this paper, we discuss some of the challenges we encountered in creating a scalable virtualized management infrastructure. We discuss our mechanisms for supporting for large numbers of hosts, VMs, and administrators while still providing fairness, security, availability, robustness, and backward compatibility. Each of these issues requires careful coordination between the hypervisor layer and the management infrastructure.

The structure of this paper is as follows. In section 2, we will give a brief overview of management infrastructure in virtualized environments. In section 3, we discuss some of the challenges faced in designing an infrastructure for use at both small and large scales. In section 4, we describe our mechanisms for dealing with these challenges. In section 5, we describe our techniques for testing at scale. We give concluding remarks and areas for future work in section 6.

## 2. VIRTUALIZATION MANAGEMENT

To understand the needs of virtualization management, it is helpful to consider the administrative requirements of a non-virtualized datacenter. On a day-to-day basis, a datacenter administrator must perform a variety of tasks:

1. **Deploy new systems.** This involves attaching machines, networks, and SANs to the infrastructure and making sure new systems fit within the power budget for a rack or datacenter.
2. **Deploy new applications.** An administrator may be asked to create instances of applications (like databases or mail servers) for end-users. This may involve deploying a new system just for this application, or trying to find an existing server to host the application.
3. **Perform ongoing maintenance.** In large datacenters, machines are periodically powered-down for software or firmware upgrades, or machines that have malfunctioned are taken off-line. In addition, machines may need to be reconfigured (with additional network cards or storage adapters added to the host) or networks and storage may need to be re-partitioned as more users come online.
4. **Perform performance debugging.** If an end-user complains that her application is not performing well, a datacenter administrator might need to examine the hardware and software configuration of the host to determine why.

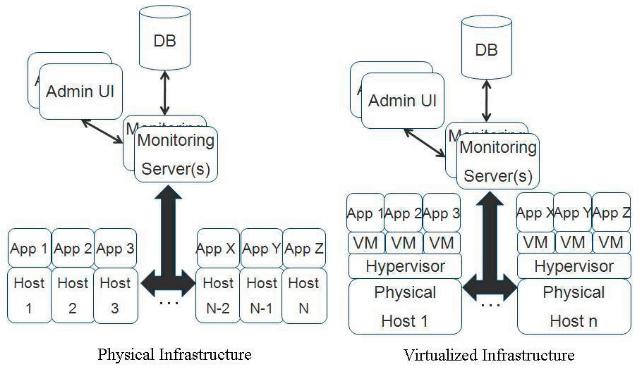

**Figure 1: Contrasting Physical Datacenter Management and Virtualized Datacenter Management.** In a physical environment, each host may run a single application, and the management layer must monitor hosts, applications, storage, and networking. In a virtualized environment, applications run inside VMs, and the management layer must also monitor VMs and the hypervisor. In VMware vSphere, the monitoring servers are called *vCenter* servers, and the physical hosts run ESX.

5. **Perform proactive system monitoring.** Many datacenters have a *network operations center* (NOC) for giving a high-level view of the entire datacenter. The NOC can show at a glance which hosts are up, which hosts have failed, and which networks and storage devices are available or are experiencing contention (and might cause outages). When there is contention, an administrator may need to do manual load balancing by moving applications among servers.
6. **Perform backups and recovery.** One of the most important tasks in a datacenter is making sure that mission-critical data is backed up, and that data can be recovered in case of an outage.

The administrator of a virtualized datacenter has all of the above tasks, but many of them are made easier with virtualization. For example, deploying a virtual server in a virtualized environment is nearly analogous to deploying a new application in a physical infrastructure, and can be as simple as selecting an existing host, cloning the VM files to that new host, and running a customization script to modify IP addresses and host names, all without the need for new cabling, additional hardware, or consideration of whether there is sufficient power budget within a rack. Figure 1 illustrates the similarities between the management infrastructures for a physical datacenter versus a virtualized datacenter. In general, there is a central server or group of servers to monitor the status of each host and propagate information to each of the system administrators. In addition, configuration and statistics information is persisted in a database. Finally, management systems provide an administrator user interface ("Admin UI") in order to allow system administrators to perform tasks on the infrastructure, for example, reconfiguring or rebooting a host. There are several commercially available tools for monitoring and management of both physical and virtual infrastructures[4][6][8][9][11][12][18].

### 2.1 Virtualization Operations

In addition to the standard operations in the datacenter, administrators in virtualized environments perform a number of operations that are unique to virtualization.

1. **Live migration.** Live migration [13] involves moving a VM from one physical host to another while the VM is still running. This is used when a given host requires maintenance but its VMs must be kept running, and is also used for load balancing, in which VMs are moved from heavily-loaded physical servers to lightly-loaded servers to balance the overall CPU/memory resource usage in the datacenter..
2. **Snapshot operations.** A *Create Snapshot* operation checkpoints the state of a VM. This allows a user to perform operations on the VM and then rollback to a known state in case of failure. A common use for snapshots is when installing the latest version of an application. The user snaphots the VM and then installs the software. If the installation succeeds, the snapshot can be removed and the user simply continues from the current state. If the software is buggy or crashes, however, the user can *revert* the snapshot, restoring the VM to the checkpointed state without the software installed. *Committing a snapshot* means writing to disk all of changes that occurred since the VM snapshot was taken, and removing the snapshot file.
3. **Clone VM.** A *VM clone* creates a replica of a powered-off VM. One common use case occurs when a new employee joins a company and the standard desktop VM image is cloned and customized for the new employee.

These benefits are some of the reasons that virtualization has become the backbone of various cloud-computing platforms [1][17]. Because of the scale of cloud computing, there is the additional complexity of a large number of administrators for the physical/virtual servers and end-users for the virtualized applications. Moreover, datacenters are becoming multi-tenant: that is, multiple customers are sharing the same physical infrastructure, so security and isolation between users is crucial.

To illustrate the importance of management operations in the virtualized datacenter, Table 1 shows the frequency of the various operations observed across a number of production virtualized datacenters [16] as a function of the number of VMs. In addition to the operations described earlier, the table includes one more operation, *Patch Install*, which involves installing a patch on a physical host (e.g., updating the hypervisor) or installing a patch in a VM (e.g., updating the guest OS with the latest security fixes). Virtualization is particularly valuable with patching, since the state of a VM can be captured with a snapshot prior to installing the patch, and if the patch causes the VM to crash, the snapshot can be reverted.

As the table indicates, the valued-added services (provisioning, live migration) are performed quite frequently, even in production datacenters. Consider powering on VMs. It may be rare to power on thousands of physical hosts in a short time window, but this may be routine in a datacenter with a thousand VMs. In addition, in many datacenters, these operations may be very bursty, with the number of operations varying dramatically depending on the day. For example, there may be more snapshots taken on a day when a new software package is being tested, and far fewer on other days. In addition, these bursts typically occur during short maintenance windows within the datacenter, causing temporary large spikes in management operations [16].

**Table 1: Management Operations at Various Customer Sites.** Virtualization enables different types of operations from a physical datacenter. These operations often scale with the size of the inventory.

| Operation | Observed Frequency in Various Datacenters |
|---|---|
| VM reconfigure | 2x per day per VM |
| Automated Live Migration | 6x per day per VM |
| VM powerOn | 1x per day per VM |
| VM powerOff | 1x per day per VM |
| VM reset | 2x per day per VM |
| Patch Install | 1x per day per VM |
| Create Snapshot | 3% of VMs per day |
| Snapshot Revert | 12% of VMs per day |
| Snapshot Commit | 1% of VMs per day |
| VM Clone | 2% of VMs per day |

## 2.2 VMware vSphere Architecture

The management layer designed by VMware is known as vSphere [18]. The base vSphere architecture is similar to Figure 1, and includes a single monitoring (*vCenter*) server, a database for archiving configuration and performance data, and agents running on each physical (ESX) host. The vCenter server also supports an API for third-party tools and UI clients to perform and monitor operations on vSphere [21]. The vCenter server itself is a collection of processes for monitoring the individual hosts and maintaining a cache of state about those hosts. Any commands from UI or third-party clients must first communicate with the vCenter server. For example, if a user wishes to power on a VM, the user selects the VM to be powered on, and the client sends the command to the vCenter server. The vCenter server redirects the command to the appropriate physical host where the VM is located and tracks the progress of the task on the host. The agents on the host perform the command and then send a task completion along with updated configuration information to vCenter, which archives the information in the database. In addition to performing tasks, the vCenter server also functions as a monitoring server, collecting statistics information from each host on a periodic basis. If a client wishes to observe resource usage information about hosts and VMs (for example, the CPU usage of a group of VMs over the last hour), these requests go to the vCenter server, which queries hosts and the database to retrieve the latest version of the data and then sends this data back to the requesting client.

In addition to updating vCenter after task completions and providing up-to-date utilization statistics, hosts also periodically synchronize with the vCenter server to keep it up-to-date. For example, if the amount of disk space available to a host changes because additional VMs have been deployed on that host, then the vCenter server is apprised of this change. These changes are then persisted to the database. The vCenter server contains an in-memory version of much of the data so that client requests can be satisfied quickly.

In large environments, it is helpful to split the physical hosts among several vCenter servers, as indicated in Figure 1. In vSphere, the vCenter servers are connected via *Linked Mode*. Linked mode serves two main purposes. First, it allows UI aggregation and provides a single view for the entire infrastructure: the user logs in to a single vCenter server, but information for all vCenter servers is presented in the same view, and queries for any of the servers are seamlessly redirected to the appropriate server. Second, in Linked Mode, user roles are synchronized across the vCenter monitoring servers using LDAP, allowing an administrator to assign privileges once and have those privileges applied across the entire environment.

There are interesting tradeoffs in the overall architecture of whether management code should reside on several dedicated server VMs or should be spread onto every single ESX host. We believe we should leverage ESX hosts for stateless work that reduces load on the network and the management servers, but that the bulk of the state should reside on the management servers. In a number of environments, there will be several management servers, and they should coordinate with each other to provide a seamless experience.

## 3. REQUIREMENTS AND CHALLENGES AT SCALE

To build a management infrastructure like vSphere, we have to consider a variety of use cases. Some deployments consist of just a few hosts and hundreds of VMs, while others have thousands of hosts and tens of thousands of VMs and geographically-distributed administration. At cloud-scale, these numbers can easily grow by another factor of ten. In addition, customers will have different numbers of end-users, administrators, and third-party clients interacting with the infrastructure. We tried to design vSphere with the goal of being able to scale to all of these use cases.

What are the requirements that result from such scale? From a user's perspective, there are several key requirements:

1. *Performance/Fairness*. The virtualization layer must provide performance guarantees based on what the end-user has purchased, while the management layer must provide fairness guarantees between administrators. For example, the management layer must make sure that if there are hundreds of administrators, each is still able to submit tasks and perform monitoring operations at the same time without interfering with each other. Moreover, this fairness must be enforced when resources are oversubscribed, regardless of the number of end-users or administrators.
2. *Security.* There are many aspects to security in a virtualized environment. The most straightforward consideration is preventing users from interacting with each other's VMs. Another aspect is ensuring that a given VM does not consume all of the resources on a host and therefore prevent another VM from running. Moreover, the management layer must close any covert channels: it must guarantee that a given customer is unable to snoop on the network packets of another customer, for example, or read the memory locations of another customer's VMs. Finally, the management layer must provide roles and permissions in order to control what operations can be done by end-users vs. what operations can be done by administrators. With increasing numbers of users and VMs, the number of combinations of users, objects, and permissions grows drastically, and it is the responsibility of the management infrastructure to keep the overheads small while still providing sufficiently granular permissions.
3. *Robustness.* Different environments have radically different deployment concerns. For example, environments that span multiple geographies need to worry about low-bandwidth, high-latency communication between a central monitoring server and the hypervisors. The enterprise customer may have large numbers of geographically-distributed hosts, each with a lot of memory, may have hundreds of VMs per host, and may require large-scale automation, while a small-to-medium business (SMB) customer may only have a few hosts, and may perform all tasks manually using a UI. The overhead of virtualization management in each case should scale with the number of managed entities, rather than be a fixed cost per host, and the management tools should cater to both types of customers.
4. *Availability.* For the enterprise datacenter running mission-critical applications, the management layer itself may qualify as a mission-critical application, and therefore must be highly-available. As environment sizes grow, failures are guaranteed to happen with some regularity, and must be properly handled. For deployments with remote sites, the central administrators must rely on an always-available management layer, since the physical hosts are not easily accessible.
5. *Backward Compatibility.* As a datacenter or group of datacenters becomes larger, environments will be upgraded on a rolling basis, and the management layer must be capable of dealing with such heterogeneity.

For software maintainability purposes, it is important that the VMware vSphere software conform somewhat to a 'one-size-fits-all' design schema. Specifically, a single platform must be capable of supporting customers of all shapes and sizes, rather than having custom software for each use case.

## 4. APPROACHES TO ISSUES OF SCALE

In this section, we discuss how vSphere tackles some of the issues of scale presented in the previous section.

### 4.1 Performance/Fairness

With a large number of users and workloads, the system needs to ensure that each workload is getting its proper share of resources. The ESX kernel ensures that different VMs running on the same host get the share of resources that they are entitled to by time slicing CPU, chaining memory allocations, and throttling I/O bandwidth. Distributed Resource Scheduling (DRS) [19], in coordination with the ESX-level schedulers, ensures that VMs across a group of ESX hosts are getting the appropriate share of resources by adjust host-level entitlements and live-migrating VMs.

Besides VM resource-consumption fairness, we also need to provide fairness for management operations, because management operations can also be resource-intensive and consume shared resources. For example, unchecked numbers of concurrent live migrations could overwhelm the network while transferring memory between hosts. In addition, unchecked storage live migration (Storage VMotion) [23] and clone operations could overwhelm the storage subsystem. Finally, an administrator may continuously keep issuing VM power on/off operations faster than the time it takes to power on/off a VM.

We have built multiple mechanisms to limit the impact of such scenarios. For live migration and storage VMotion, we have configurable limits on how many simultaneous instances of these operations to allow per host, per network, and per storage endpoint. The defaults are chosen based on experimentation, but this is not ideal because the best values really depend on the hardware capabilities and dynamic load. In the future, we plan to monitor hardware capability and load to decide how many such operations to allow simultaneously.

To prevent one user from continuously issuing more operations and adding significant latency for other users, we have implemented a simple mechanism where the queue of waiting operations is grouped by user/session, and we have policies to make sure each user/session gets a turn. To allow load balancing of requests and to avoid starvation of client calls to the server, the vCenter server contains multiple thread pools for different types of requests. Client data-retrieval requests, requests for synchronizing data between hosts and the vCenter server, and client task requests are served from different pools. Each incoming request to the vCenter server contains a client identifier so that the server can load balance requests among clients. These policies currently do not treat all users and extensions the same, but we plan to have a better prioritization in the future.

## 4.2 Security

As scale increases, the size of the attack surface increases, which makes it harder to administer and reason about the security policy. We use several techniques to reduce this complexity. The ESX hypervisor is responsible for isolating VMs from one another and making sure they don't access each other's state. For communication among VMs within and across hosts, products like vShield Zones [24] provide firewall, NAT, and intrusion detection/prevention capabilities. For dealing with administrator and end-user access throughout the infrastructure, vSphere provides a comprehensive permissions model. Permissions are defined as a 3-tuple: user, action, object (user A is allowed to perform action B on object C).

To simplify the administration of the entire environment, we reduce the number of tuples by allowing grouping in all three dimensions (users, actions, and objects). For example, permissions can be assigned according to groups defined by the underlying user-management/authentication mechanisms (e.g., Active Directory or LDAP), so that all users in a particular /etc/group or Windows group can be given the same permissions. Actions are grouped into *roles*. The system comes with predefined roles that contain actions typically performed by users with that role/job. For example, the default VM user role includes power-on, power-off, reset, and suspend commands, but does not include ESX host configuration commands. Users can customize these roles, and these settings are replicated to all vCenter servers grouped together in linked mode to reduce complexity even further in distributed or multiple site environments. Objects are grouped into hierarchical folders, similar to a filesystem directory structure. Permissions assigned at a folder level are propagated to all descendants of the folder unless overridden at a lower level. This model works well in most cases, but as we scale bigger and encounter more diverse organizations, we are investigating providing grouping label/tag-based and query/expression-based grouping of objects in the future.

## 4.3 Robustness

### *4.3.1 Network Topology*

The issues of scale do apply solely within a datacenter. For example, in remote-office/branch office (ROBO) scenarios, a store may have a large number of branches geographically distributed across a continent or around the world. Each store has some physical hosts running VMs, and management is often centralized at the main headquarters. The stores are connected to the centralized manager over WAN links. As a result, the latencies for management operations can be longer than for a standard datacenter deployment, and depending on the loss characteristics of the connection, the centralized manager may be disconnected from the remote hosts at various times. Moreover, the bandwidth will be much less than in datacenter deployments. Finally, administrators themselves typically communicate with the centralized manager over WAN links, and slow access provides a poor user experience.

The limited bandwidth available between the hosts and central manager suggests that data transmission must be minimized. The primary method of dealing with the high-latency, low-bandwidth links between the central management server and the remote offices is to limit the amount of data to be transferred and also allow disconnected operation. In vSphere, we use compression to limit the data communicated between the ESX hosts and the vCenter server. This data compression applies to all data: configuration changes, task updates, and statistics traffic. Moreover, for communication between clients and the vCenter server, we have APIs for clients to subscribe to specific fine-grained changes on a per-object basis (like VMs or hosts), allowing clients to receive changes rather than constantly retrieving the entire configuration data for an object on any action.

Another important consideration with poor connectivity is that the remote hosts should operate whether or not they are connected to the centralized manager. This means that VMs should continue to run whether or not the ESX host is connected to the centralized manager, and it also means that an administrator be able to make changes to a host even if that host is not connected to the centralized manager. In vSphere, VMs will continue to run whether or not the ESX host is connected to a vCenter server. In addition, we have designed our APIs such that the administrator UI can connect to a vCenter server or connect directly to an ESX host. If an ESX host gets disconnected from its vCenter server, the administrator can still manage the host in the short term by connecting to it directly. Any changes that occur during the disconnection phase are automatically merged to the vCenter server when connectivity is restored.

### *4.3.2 APIs*

Another issue of robustness concerns our APIs. The management API needs to be simple enough that administrators can write scripts to monitor or perform tasks, while rich enough to allow third-party developers to create customized large-scale monitoring tools. Moreover, the underlying primitives must provide enough expressiveness that higher-level software can perform well at scale. For example, consider a query to request configuration data for all VMs. The result set may include metadata describing the object and specific attribute requested, and may repeat this data for each object. While this script may work well for small environments 10 VMs, it may perform very poorly or cause undue load on the vCenter server when applied to 1000 VMs because of the overhead of processing the metadata. As mentioned earlier, to address this challenge, our API allows users to specify which objects and which data to retrieve at a fine granularity. A standard method to achieve scalability is to use this API to create multiple clients that monitor disjoint sections of the environment and send requests to the vCenter server separately. For ease of use, we also provide scripting toolkits for high-level languages like Perl and PowerShell [21]. These toolkits are targeted for simple scripting by administrators. These toolkits are layered upon a web services API that allows low-level access to every object in the environment: the user has the option to use this web services API to generate client-side stubs in whatever language they wish (e.g., Java, Ruby, or Python). The impedance match between the toolkit and the API can be challenging. For example, a toolkit may retrieve metadata for objects on each invocation because each invocation is assumed to be stateless, while a program written using APIs may optimize network bandwidth by explicitly caching such metadata. The proper choice of toolkit vs. direct API programming depends on administrator needs and which resources are most constrained: for a WAN environment, perhaps network bandwidth is more expensive and direct API programming is necessary, while a datacenter may have a lot of network bandwidth and may find a toolkit suitable.

### 4.4 Availability

Even with the highest-quality components with large MTBFs, when operating in a large environment with many instances of the components, failures happen frequently [3][5][14]. We need to design the management software assuming that failures will happen and limit the impact of failures. The typical ways of doing this are either to have redundant components when the cost is acceptable, or restart failed components as fast as possible.

For virtual machines, VMware Fault Tolerance [15] keeps a shadow VM for every protected VM, running in lock step with the primary. It requires high-quality network connection between the two. If that cost is too high, customers can choose to run VMware High Availability [20] to restart VMs quickly.

Besides building these availability features for VMs, we also need to limit the impact of faults that affect the management software, because customers rely on it during critical times, such as in the morning when people arrive at work and need to power on virtual machines for their desktop. For a large VDI customer with, say, 10K employees, every minute the management software is unavailable during office arrival time means (1 minute * 10K employees = ) 166 hours of lost productivity.

Currently, we address this problem in two ways: 1) partition the environment into multiple vCenter monitoring servers that are linked together but fail independently (*Linked Mode*), and 2) rely on a hot standby to quickly restart the affected vCenter monitoring application. We keep a passive system booted up into the OS ready and synchronized with updates from the primary active server. As soon as a failure is detected, the standby server will start up the vCenter monitoring application and connect to the vCenter monitoring database. We have optimized the vCenter monitoring server application startup process so that the newly-started application can be ready in a few minutes even when managing up to 10K VMs. In the future, we plan to further optimize this down to less than a minute, and also explore active/active clustering.

### 4.5 Backward Compatibility

Designing for scalability is challenging by itself, but is even more difficult when taking backward compatibility into account. The vSphere management layer deals with two very different types of backward compatibility issues, both of which are affected by large scale: managing multiple versions of the ESX hypervisor and serving earlier versions of the vSphere management API. In typical datacenters, hardware is upgraded during periodic refresh cycles, rather than all at once. With larger and larger environments, the number of disparate versions of the hypervisor can increase, and the management layer must be capable managing these hosts properly. Moreover, interoperability between hosts is important because some VMs may be mission-critical and therefore must be live-migrated from a host-to-be-upgraded to another host.

The vSphere management layer provides support for backward compatibility in two ways. First, the API for communication between the vCenter server and each host is versioned—this allows the vCenter server to determine the capabilities of each host. Second, when vCenter first connects to a given ESX host, vCenter uploads a versioned management agent onto that host. All communication between vCenter and a host occurs via this agent. Each combination of vCenter server and ESX host has a version of this agent, and when a host is upgraded, vCenter automatically uploads the proper version of this agent to the host.

Sometimes, design for scalability requires coordination between the vCenter server and ESX hosts. This is easy enough to do with new ESX versions, but factoring in older hosts makes this quite challenging. For example, one scalability issue is network traffic for propagating statistics and configuration updates between ESX hosts and the vCenter server. At small scales, this traffic is negligible compared to the available bandwidth, but at very large scales, this traffic can ultimately overwhelm the network, especially when links are slow or have extremely limited bandwidth. To combat this issue, we introduced data compression in all messages sent from the ESX host to the vCenter server. In order to retrofit this to older hosts, we needed to download a new agent onto the ESX host and choose the proper layer in the stack at which the compression could be performed (in this case, the SSL layer), since older versions of the host are not aware of the compression capabilities.

The vSphere management layer provides a rich API for developing monitoring, control, and automation software. This API has been around for several years and we guarantee backward compatibility of this API. Many companies have built products using this API. Some aspects of the API make it hard to remove scalability bottlenecks in the system, because the API provides a level of consistency across large set of objects. In some cases, even though the API documentation does not provide such guarantees, the existing implementation provides them, and several third-party products rely on them. Changing even these would mean that those higher-level products would break and result in a poor experience for the end user. In many cases we have been able to address the API compatibility challenges by separating the API servicing code from the core of the vCenter code. In some cases we have introduced alternative (more efficient) interfaces and asked higher-level software developers to switch to it; however, that is a much slower process, since third party developers have their own installed base and backward compatibility issues.

## 5. SCALABILITY TESTING

One of the biggest challenges in the design of a management system is testing at scale. Testing requires generating a representative management load on a representative environment and determining how various attributes scale. As the number of VMs to be managed grows larger and larger, it becomes infeasible to create complete setups of such sizes. Instead, fast and accurate simulation and modeling is crucial.

A large virtualized environment consists of several vCenter servers in linked mode, more than hundred administrators, thousands of hosts and tens of thousands of VMs. To get the best accuracy, we simulate all the interactions between hosts/VMs and the management servers, but we do not modify the management server code itself. We have developed a simulated ESX host which can simulate VMs as well, but is much more lightweight than an actual host. In order to keep the simulation highly accurate and to reduce the maintenance cost, the simulated host runs mostly unmodified management agent software that would normally run on real ESX hosts, but we stub out the bottom layer that interacts with the ESX kernel and VMs. Currently, a typical physical host can run more than 30 simulated hosts and 1000 simulated VMs. These simulated hosts and VMs support the most common virtualization management operations described in section 2, and we model typical latencies of these operations by delaying the responses.

In order to simulate administrator operations, we have a benchmark suite which uses our APIs to send management commands (e.g., power on VM, VMotion, snapshot VM) to the vCenter server. The breakdown and frequency of commands is determined by profiling customer data. We typically run workloads that are at least 2x or 3x the worst customer loads we have seen in practice, in order to guarantee the reliability of vSphere. We measure the throughput and latency of commands and assess how the system behaves as we vary the number of VMs, hosts, administrators, and monitoring tools that access the APIs. For each experiment, we also measure CPU, memory, and I/O consumption of the vCenter servers and the databases to give sizing guidance to users. That data combined with profiling tools also helps us find the next code path we need to optimize.

In order to simulate various deployment strategies, we also utilize network simulation to inject latencies between the vCenter server and the simulated hosts. To simulate the remote branch office with poor WAN connectivity, we insert a router VM between the vCenter server and the ESX hosts and vary the latencies and error rates of the packets between vCenter and the hosts. We then perform our load tests to determine the impact on throughput and latency of the vCenter server.

In the future we will have to simulate bigger environments, so we are investigating ways of reducing the resource consumption of host simulator even further while still maintaining the accuracy. One approach we are considering is to leverage the homogeneity of VMs in the simulation and share the VM specific state across several VMs.

## 6. CONCLUSIONS AND FUTURE WORK

Virtualization increases resource utilization and takes advantage of emerging hardware trends in multi-core CPUs, high-speed IO devices, and enhanced memory systems. To best take advantage of these features, the virtualization management infrastructure must be scalable at all levels, secure, low-overhead, and extensible. We implement a highly-concurrent management layer with fine-grained synchronization to enable scale-up with more CPU and memory resources on a given management node. We allow federation among management instances to provide scale-out support. Typically, scaling work involves fixing locking bottlenecks and improving inefficient code. While there are a number of interesting challenges there and we have come up with novel solutions to them, this paper focuses on the other work we have done in order to scale the system.

In addition to increases in managed hosts and VMs, increased scale also means there are more users interacting with the system, so the system has to provide fairness for all limited and shared resources in a way that does not impact overall throughput and latency of operations. The system also needs to provide secure access and ease the manageability burden of implementing the desired security policies. APIs also need to be properly designed for ease of use and efficient data management at scale. Management software needs to be designed for a bigger variety of network topologies and has to gracefully adapt to connectivity failures. Faults are not exceptions but rather normal behavior for large systems, so we have to contain the impact of faults and recover quickly. Improving the system in all these facets becomes even more challenging due to backward compatibility constraints inherent in large systems with a big ecosystem utilizing the rich APIs.

Scale will keep getting bigger for the foreseeable future because users are getting more comfortable with virtualization and are moving all server applications/workloads to virtual environments. Also, users are shifting desktop computing to virtual machines in backend servers and relying on thin terminals to access them. Finally, there is a shift towards cloud computing, and this means the cloud service providers will be supporting very large environments that support the computing needs of several companies. We need the management software to scale to match these needs, and as we progress into the future and these trends become reality, the increased scale will continue to present new and interesting challenges for management software. For example, one of the major research questions is how to monitor large environments. Various solutions exist for Grid-style computing [11] or for warehouse-sized computers [3][4][5], but part of the challenge is adapting such solutions for virtualized infrastructure, specifically, finding ways to correlate application performance within a VM to resource usage on the underlying host. In addition, visualizing such large amounts of data so that administrators can quickly diagnose and fix issues is quite challenging, as is automated health monitoring so that human intervention is not required. We will have to invent more novel techniques to address these challenges.

## 7. ACKNOWLEDGMENTS

We'd like to thank E. Lewis, Priti Mishra, Balaji Parimi, Shankar Unni, Ben Verghese, and Xiaoyun Zhu for invaluable feedback on earlier versions of this paper.

## 8. REFERENCES

[1] Amazon EC2. http://aws.amazon.com/ec2.

[2] AMD Magny-Cours. http://www.amd.com/us/products/server/processors/6000-series-platforms/Pages/6000-series-platform.aspx

[3] Barroso, L., et al. The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines. Morgan and Claypool Publishers. 2009.

[4] Boulon, J., et al. Chukwa, a large-scale monitoring system. In Proceedings of CCA '08. Oct 2008

[5] Dean, J. Designs, Lessons, and Advice from Building Large Distributed Systems. Keynote from LADIS 2009 (Big Sky, Montana, October 10-11, 2009).

[6] Dell. Dell OpenManage. http://dell.com/openmanage

[7] EMC. EMC Symmetrix DMX-3 950 specifications. http://www.emc.com/coll ateral/hardware/specification-sheet/c1153-dmx3-950-ss.pdf

[8] HP. HP OpenView. http://hp.com/openview

[9] IBM. IBM Tivoli. http://www-01.ibm.com/software/tivoli

[10] Intel Nehalem. http://www.intel.com/p/en_US/products/server/processor/xeon7000

[11] Massie, M.,et al. The Ganglia Distributed Monitoring System: Design, Implementation, and Experience. In Parallel Computing Volume 30, Issue 7, pp 817- 840, 2004.

[12] Microsoft. Microsoft System Center Virtual Machine Manager. *http://*.microsoft.com/systemcenter/virtualmachinemanager/

[13] Nelson, M., et al. Fast Transparent Migration for Virtual Machines. In Proceedings of USENIX ’05 (Anaheim, CA, April 10-15, 2005). 391-394.

[14] Pinheiro, E., et al. Failure Trends in a Large Disk Drive Population. In Proceedings of FAST ’07 (San Jose, CA, February 13-16, 2007). 17-28.

[15] Scales, D., et al. The Design of a Practical System for Fault-Tolerant Virtual Machines. In SIGOPS Operating Systems Review. Volume 44, Issue 4. 2010.

[16] Soundararajan, V., and Anderson, J. M. The Impact of Management Operations on the Virtualized Datacenter. In ISCA ’10 (Saint-Malo, France, June 19-23 2010). 326-337.

[17] Terremark. http://vcloudexpress.terremark.com/

[18] VMware vSphere. http://www.vmware.com/products/vsphere/overview.html

[19] VMware. VMware DRS. http://www.vmware.com/products/drs/

[20] VMware. VMware High-Availability. http://www.vmware.com/products/high-availability

[21] VMware. VMware SDKs and APIs. http://www.vmware.com/support/pubs/sdk_pubs.html

[22] VMware. VMware Site Recovery Manager. http://www.vmware.com/products/site-recovery-manager

[23] VMware. VMware Storage VMotion. http://www.vmware.com/products/storage-vmotion/

[24] VMware. VMware vShield Zones. http://www.vmware.com/products/vshield-zones/