Volume 10 Issue 03 Published, August 10, 2006 ISSN 1535-864X DOI: 10.1535/itj.1003

Intel® Technology Journal

Intel® Virtualization Technology

Virtualization in the Enterprise

More information, including current and past issues of Intel Technology Journal, can be found at:
http://developer.intel.com/technology/itj/index.htm

Virtualization in the Enterprise

Patrick Fabian, Technology and Manufacturing Group, Intel Corporation
Julia Palmer, Information Technology, Intel Corporation
Justin Richardson, Information Technology, Intel Corporation
Mic Bowman, Corporate Technology Group, Intel Corporation
Paul Brett, Corporate Technology Group, Intel Corporation
Rob Knauerhase, Corporate Technology Group, Intel Corporation
Jeff Sedayao, Information Technology, Intel Corporation
John Vicente, Information Technology, Intel Corporation
Cheng-Chee Koh, Information Technology, Intel Corporation
Sanjay Rungta, Information Technology, Intel Corporation

Index words: virtualization, use cases, case studies, enterprise IT

ABSTRACT

We present how an enterprise IT organization sees virtualization in the enterprise and how it can be applied. We look at key enterprise services and applications used within Intel's IT department and examine the issues associated with virtualizing servers within the context of those services. We demonstrate that virtual machine (VM) isolation does not extend to performance isolation as we show how applications running in separate VMs can significantly interfere with each other. Enterprise services depend on host characteristics like available cycles, platform configurations, and on proximity to other services. We define a taxonomy of these dependencies derived from our study. Next, we describe uses of Intel® Virtualization Technology$^{\Delta}$ (Intel® VT) that we are investigating. The ability to run multiple operating systems (OSs) is of great interest in our design environment where highly specialized tools are tied closely to OS versions. The ability to checkpoint, suspend, resume, and migrate VMs is very useful when we run long simulations. The ability to allocate VMs at the location of choice opens up other possible use cases, such as network monitoring, security monitoring, and content distribution. We see this capability also enabling safe yet realistic experimentation, as a way to extend virtualization into clients. Finally, we present a real case study applying virtualization to enterprise IT problems. This virtualization program achieved higher server utilization, made it easier to manage datacenter assets, and reduced the consumption of datacenter resources (floor space, power, etc.), as well as simplified server releases through standardization.

INTRODUCTION

Virtualization is touted as a new and upcoming trend in computing. Simply stated, virtualization is a technology to run multiple independent virtual operating systems (OSs) on a single physical computer. It is not a particularly new idea in the enterprise, having been implemented in the 1960s on IBM mainframes [1].

A number of characteristics of virtualization make it a much discussed topic of conversation today. One is the potential to better use compute resources, allowing an enterprise to maximize its investment in hardware. In an average datacenter, the majority of the infrastructure resources are used about 25% of the time. Virtualizing a large deployment of older systems on fewer highly scalable, highly reliable, modern, enterprise-class servers significantly reduces hardware costs for infrastructure services. Multiple hardware and software solutions are available on the market and ready to provide a secure, easily managed platform to deploy, manage, and remotely control VMs.

Virtualization offers so much more than just server consolidation. Intel's IT organization has been studying other uses of virtualization that can add tremendous value to an enterprise. Virtualization features such as the ability to suspend, resume, checkpoint, and migrate running VMs is extremely useful in dealing with long running jobs. If a VM with a long running job checkpoints its state and then the physical machine it is on fails for some reason, the job can be restarted from where it left off, along with the VM, rather than being restarted from the beginning.

A key difference of virtualization today and the mainframe age is the ability to allocate a VM at the location of a service's choice. This notion of Distributed Virtual Machines (DVMs) opens a whole host of possible uses, such as network monitoring, security policy validation, and content distribution. It enables enterprises to create such things as virtual secure enclaves and do safe yet realistic testing of large scale, even planetary scale, services. This idea is useful and compelling enough to power the PlanetLab testbed [2], which is slated to become a core part of a next-generation Internet project called GENI [3].

Virtualization, while a viable technology today, is not without issues. Allocating VMs for enterprise services is not as simple as finding the first available host. Services have dependencies on network topology and other services. Also, VMs, while offering many types of isolation, do not offer complete performance isolation. VMs can interfere with each other.

This paper examines the virtualization of physical host machines, enterprise services, and multi-site instantiation of virtual environments. First, we introduce the difficulty of virtualizing enterprise service host machines. Second, we discuss use cases that can give IT organizations many new options in supporting their company's business units. Third, we review a case study of server virtualization for a business group at Intel and the process they followed from project inception through implementation. We conclude the paper with a discussion of our results and a description of future work.

CHALLENGES OF VIRTUALIZATION IN THE ENTERPRISE

For batch-oriented tasks, provisioning VMs and getting predictable performance appears to be relatively straightforward. This seemly simple task can be difficult if VMs interfere significantly with each other. When we introduce virtualization with enterprise services like the Domain Name System (DNS) [4], VM provisioning becomes more complex, especially as the location of the physical machine hosting the VM becomes important. In this section, we describe the challenges of server virtualization in an enterprise context.

Studying the Issues of Virtualization in the Enterprise

Our approach to studying the issues of virtualization in the enterprise had two parts. First, we looked at how VMs running on the same physical host could affect each other, particularly when different applications were running on the VMs. Second, we looked at key enterprise services and investigated how these services would fare in a virtual environment.

This is how we studied virtual machine interference:

- We obtained baseline performance (a control) of an application running on one VM.
- We attempted to optimally degrade the performance of one of two VMs running on the host, typically by attempting to use some shared resource.
- We documented and analyzed the results.

Once we had studied how individual VMs interact on one physical host, we looked at how VMs would interact in an enterprise. We first looked at the services that are the most critical and generate the most volume on the Intel Wide Area Network (WAN). Our goal was to examine those applications for performance bottlenecks and platform dependencies that would be problematic when servers for those applications and services would be virtualized. In addition, we also looked for five additional applications commonly used within Intel. For all of these services and applications, we searched for information on the Web and talked to IT personnel who are expert at running them in Intel's IT environment, looking for the issues mentioned previously.

VM Interference

VMs, as a technology, offer many advantages to users and administrators. Security isolation prevents a malicious application from accessing data or altering running code. Fault isolation prevents one misbehaving application from bringing down the whole system—rather than rebooting the box, one can simply reboot the VM. Environment isolation allows multiple OSs to run on the same machine, accommodating legacy applications and cutting-edge software alike.

While VMs offer these forms of isolation, we have observed that modern VM environments [4, 5] do not really provide performance isolation. While in theory, the virtual machine manager (VMM, also known as the hypervisor) "slices" resources and allocates shares to different VMs, there are still ways in which the behavior of one VM can adversely affect the performance of another. Furthermore, the isolation that VMs provide limits visibility of an application in a VM into the cause(s) of performance anomalies that occur in a virtualized environment. Contemporary platforms with Intel VT, however, provide mechanisms that we can use to detect and classify performance interference, which can then be used for a number of purposes:

- As input to the local scheduler, which can alter its behavior (e.g., change quanta or ordering) to ameliorate the effects of the interference.

- As input to a global scheduler, or orchestration engine, which uses the information to rearrange the placement of VMs to minimize interference and improve performance.
- As "metering" data, so that systems that charge for usage (e.g., free-market allocation systems such as HP Labs' *Tycoon*[*] [6], grid computing pay-per-cycle or "cycle-rental" schemes, etc.) can more accurately charge/credit users for the resources they consume/provide.

Our research to date has shown that shared state in resources under contention can indeed dramatically affect the performance of a VM, beyond the expected performance degradation that is due to simple time-slicing of the resource.

The first type of interference we studied was the interference within the processor's L2 cache and to server state (disk head position, cache state, etc.). We designed experiments to quantify these types of interference by running "benchmark" workloads against other VMs with code designed to be explicitly pessimistic in terms of interference to that particular benchmark. Our results show that in each case, there can be a non-negligible effect on the benchmark's performance.

For the cache experiment, we wrote code to continuously write to a large (bigger than the L2 cache) array in one VM to show how this would interfere with a memory-intensive application in another VM. We looked at the Freebench test suite [7] because it was freely available and had been used in other VMM performance testing [8]. We selected Freebench's analyzer benchmark as an application. The analyzer's performance is limited by the memory subsystem, making it a good candidate for cache interference. It runs a deterministic computation, so we used time-to-completion as our measurement of performance.

Our experiments compared the runtime of the program versus another VM executing a simple spinloop. Because of this, the slowdown seen can be attributed directly to cache interference (i.e., its degradation is due to more than simply sharing half the CPU with another VM). We ran our experiment on several types of Intel® platforms, with varying configurations. A typical run is shown in Figure 1. As the amount of cache used by our interference-generating application increases, the slowdown in application performance increases. That a dirty cache slows down an application's execution is not surprising; however, the application and its OS are completely unaware that the cache is being dirtied, and because they are running on a VM, typical techniques (e.g., OS task scheduling) are unavailable. With Intel VT features, we are able to determine the interference and provide that information to higher-level constructs (e.g., the hypervisor scheduler, or a global orchestration system) as mentioned above.

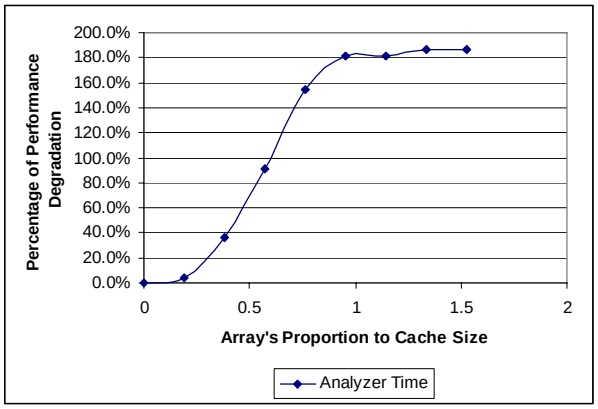

Figure 1: Performance degradation as cache is increasingly dirtied

We also ran tests for storage interference. The simplest example entailed two VMs accessing the same disk device, to most easily demonstrate head-position and disk-cache state (outside the VM). Our results (as in the CPU cache case that we compared against a simple spinloop VM) showed similar amounts of additional degradation—between 50% and 90% depending on the nature of the disk access (sequential/random and character/block).

Virtualization and Service Dependencies

To get the list of critical network services, we consulted with Intel IT's WAN engineers. They reported the following are the five most critical network services:

- Exchange*
- DNS
- Active Directory*
- Chip design associated file transfers
- Web proxy traffic

In addition, we studied the following internal applications:

- Internet Information Server (IIS[*])
- Apache Web server[*]
- SQL server[*]
- Oracle
- Sendmail[*]

We looked at a number of service orchestrators also. We looked at how Oracle orchestrates its operations, as well as the IBM/Auremia director[*], the HP Workload Manager[*], the 3-DNS[*] and Big IP[*] load balancers from F5, and Microsoft's Visual Studio 2005[*]. To do this, we

looked at documentation as well as talked to operational experts within Intel's IT organization. In addition, we found that some services and applications like DNS, Sendmail, and Active Directory, have some mechanisms that perform orchestration functions.

We found that service platform dependencies fall into the following categories:

- Network
- Host/System
- Storage
- Services

We next discuss these dependencies and emphasize the aspects that are typically not covered by service orchestrators. At the end of each section, we describe constraints that need to be specified by orchestrators to deal with these dependencies, since these would be additional concerns with provisioning VMs for those services.

Network

We found that services had a number of network dependencies that are not typically dealt with by service orchestrators. Often these dependencies rely on network topology specifics. One example is the Web proxy service offered by Intel IT. This service proxies Web traffic between systems on the internal Intel network and the Internet and reduces network traffic by caching Web pages already fetched. The proxy service maintainers require that a directly connected proxy (with direct access to the Internet) has a high bandwidth network path to the Internet. For fairly large Intel sites with low bandwidth links to the Intel® WAN, Intel® IT deploys a proxy server locally and chains this proxy server to another. While some orchestration specification languages like JSDL [9] allow conditions to be set on network bandwidth, they do not address network topology.

Intel's DNS service relies on multiple network connections from a site for deployment. Intel sites with only one connection to the Intel® internal WAN have DNS servers deployed to them. Multiple and reliable network connectivity is a dependency for the DNS service.

DNS also monitors query latencies and uses them to generate basic orchestration functionality. It records the time it takes to process queries for a domain and uses the name servers for that domain that respond the fastest. In this case, network latency is a significant contributor to how the DNS service behaves and partitions work.

Some services want to see a specific number of network interfaces on the platform. Some deployments of the Oracle database system require three network interfaces. One of the network interfaces is used for heartbeat information between servers and requires low latency. Other services assume that they have a network interface (or at a minimum, an IP address on an interface) that can be directly reached at a particular port. This applies to services that use well known ports, such as Web servers like Apache or Microsoft IIS. While Web servers can do virtual hosting, they assume that a standard HTTP port is directly reachable, and in a virtualized environment, this implies that there is an IP address per each VM that runs a Web server. For each IP address, it is assumed that there is a MAC address associated with that interface, and that there is a way to route packets to each VM.

We define network constraints that we need to manage as follows:

- Minimum bandwidth between server hosting service and particular points.
- Topology and availability requirements, in particular minimum availability and/or a minimum number of paths from a server's location to other locations.
- Minimum or a specific number of network interfaces.
- Maximum latency between server and other servers.

Host

We found two notable host dependencies that are not covered by orchestration service specifications. The first is a dependency on a fixed amount of CPU resources. A commonly used mail forwarding program called Sendmail depends on the notion of load average to decide whether to queue up mail or whether to reject mail connections. In a virtual environment where resources are shared equally among VMs, an application cannot be certain whether it will receive the same amount of CPU resources since other VMs may be assigned to the same physical host it runs on. Thus, using load average is not an accurate indicator of the available resources.

The second host dependency is non-pageable memory. The Exchange mail service relies on having a significant amount of memory that cannot be paged out for good performance. While orchestrators allow you to specify how much memory a job or service may require, there do not seem to be options for non-pageable memory.

Service Affinity/Proximity Requirements

One key service dependency that is not always captured in orchestrators is affinity or proximity to other services. A good example is Exchange and Active Directory. Exchange requires fast responses from Active Directory. Operationally, an Active Directory server should be on the same segment as an Exchange server. Deviations from this configuration have proven disastrous operationally.

An additional aspect of service dependency is the need for a maximum time to complete a service's basic transactions. DNS operations personnel recommend that DNS queries must be resolved within one second in order to prevent applications that rely on DNS from hanging.

Storage

Some applications rely on specific platform features. For example, some versions of Oracle require that Oracle write directly to disk blocks. Other applications, such as Active Directory, require large disk-write caches.

OTHER USE CASES FOR VIRTUAL MACHINES IN THE ENTERPRISE

Virtualization is typically discussed in the context of datacenters, where multiple VMs are loaded onto a single host to increase server utilization or reduce the cost of buying new hardware. We cover this use case extensively in an upcoming section. Virtualization enables other capabilities that can be extremely useful to enterprises. We now discuss enterprise use cases for virtualization that go beyond the usual examples of increased utilization, which are being investigated by Intel's IT organization. We start with ways to enhance the operation and efficiency of large-scale computation. We then talk about distributing virtualization, and how the ability to allocate VMS in the location of choice opens up new applications and paradigms for service deployment.

Enhancing Standardization

While a completely homogeneous computing environment would yield obvious efficiencies, it is generally not realistic. Intel's design environment supports a huge variety of software tools for a diverse roster of design teams, some of whom joined Intel as a result of acquisitions. The design environment employs laptop PCs, dedicated compute servers, and everything in between.

In this complex environment, virtualization could achieve many of the efficiencies of homogeneity. A software tool could be bundled with its own specialized virtual computing environment. When a new version of this bundle is standardized, it could be quickly pushed out to all sites on top of VMMs without requiring expensive and time-consuming OS upgrades. This is especially helpful to small sites which often lack sufficient staff, and sometimes lack even all the required computing platforms, to implement a never-ending stream of company-wide directives.

Making it easy and inexpensive to push out new standard images to all sites not only reduces costs, but accelerates innovation, because it frees a small team to develop specialized expertise in a product and to support it worldwide instead of relying on generalists dispersed across many teams. It enables the leveraging of good ideas and fixes from any of these specialized groups, because these fixes and ideas can be quickly and easily applied everywhere.

Legacy Operation

Closely related to the standardization issue is an inevitable heterogeneity across time. OSs and microprocessor architectures evolve, and they sometimes even die. Yet important legacy software tools that depend on particular OS versions are often useful far into the future. It is expensive and sometimes insecure to maintain special-purpose "classic" configurations. In addition, tools that run on them can't ride Moore's Law to ever better performance.

If snapshots of such classic configurations were encapsulated as VMs, then they could be inexpensively "revived" whenever and wherever needed and on enhanced hardware, resulting in the associated tool being executed faster.

This strategy would also be useful for any application that is run only occasionally, especially if it needs to or is expected to run on a dedicated server. For example, during a downtime, whether planned or unplanned, a temporary mail forwarding server could be activated at some unaffected site. This approach reduces not only costs, but also the risk that an infrequently used application has fallen behind and is now incompatible with changes in the computing environment. Such incompatibilities are typically discovered at the worst possible time, that is, at the exact moment when the application is needed.

Checkpointing

A grand reliability goal is to guarantee to users that no job will ever fail to complete for external reasons, such as a machine reboot, a power outage, a disk crash, or even a catastrophic failure such as an earthquake. An important enabling technology that would be immediately useful is the automatic checkpointing of a VM.

It should be possible to schedule a periodic saving of the VM state that could be used to go back in time and replay history from that archived moment, but without the externally induced failure. Less frequently, redundant copies of the state could be stored away remotely, at a rate correlated with the probability of various risks which grow with the duration of the task. For example, a simulation of the earth's climate that required a year of calendar time to complete would almost certainly be disrupted during that year by some external event. It could be argued that the more sophisticated the application, the more likely its developers are to have already built in checkpointing mechanisms. But even if we ignore the fact that many real

applications disregard checkpointing altogether, application-level checkpointing mechanisms can't reasonably be expected to cope with catastrophic failures. Where should the application save its own state to protect against fire and flood?

Checkpointing could obviate the need for engineers to submit redundant jobs as insurance. Today, the longer or more critical a task, the more likely it is to run multiple instances of the same job, causing the actual resource utilization for a computing task to be much larger than the resource requirements for an individual job might indicate. With VMs and checkpointing, a single job would only need to be run once because it could be resumed elsewhere, even if there were an external failure. Likewise, if a machine must be rebooted for OS patches, a planned site-wide downtime, or simply as an attempt to put it back into a good state, the tasks running on it could be terminated easily enough and resumed on some other machine. This eliminates the need to prevent long jobs from being submitted days in advance of such maintenance and the temptation to postpone prudent maintenance because of the speed bump it throws into user schedules.

In the long-term, a VM could support some analog of apoptosis (programmed cell death), killing itself when it detects errors. A daemon could automatically roll back (terminate and then reincarnate elsewhere) any VM that hasn't recently enough provided proof that it is healthy.

An issue that needs to be investigated is how to deal with external (non-virtual) state elements, such as the actual current calendar time and ongoing network communications, that can't be checkpointed. Another key issue concerns licensing. Some software applications require a license for physical CPU, while others require a license per actively running copy. The issues of program state and licensing need to be answered when deploying VM checkpointing in the enterprise.

Performance Isolation

When choosing a shared computing resource, such as a server on which to run a VNC [10] session, it's difficult to predict the impact of contention. The longer the task, the greater the chance that some other user may consume an unfair share of the resource and degrade one's own effectiveness. Although using VM checkpointing could enable the victims to pick up and move to "greener pastures," it would be better to prevent a "tragedy of the commons." If we can ration real-world computing resources by configuring the parameters (memory size, processor speed, disk access speed, etc.) of each VM assigned to a task, then limits on consumption would be inherent to the resource capacity of the VM that task is running in. A number of VMMs are capable of allocating computing resources and of performing a measure of performance isolation. While we have shown in a previous section that VMs can significantly interfere with each other, particularly through the interactions of shared resources like the cache, there are other resources like CPU cycles and memory that can be allocated in a way that significantly isolates the performance effects of tasks from each other.

Distributed Virtual Machines

The ability to allocate VMs at the location of choice is a capability we call Distributed Virtual Machines (DVM). DVMs enable a whole host of possible applications and servers. Throughout this section, we use the terms distributed virtualization, overlays, and distributed virtual machines, interchangeably. We view DVM as the methodology of choice for realizing robust, computationally rich, networked virtualization and for implementing overlays.

The Origins and Impact of DVM

One of the earliest and most successful implementations of DVMs is PlanetLab [2]. PlanetLab is a world-wide overlay network with over 689 nodes at 334 locations around the world. Designed to be a testbed and deployment platform for researchers to study planetary-scale distributed systems and services, PlanetLab has distributed virtualization at its core. Researchers allocate a "slice," a set of VMs, at the locations of their choice. Using VMs allows the researchers to develop and deploy innovative new services that do not interfere with each other on the same physical hosts. Using this model of computing, several innovative services with content distribution [11] and network measurement [12] were developed and deployed. These types of applications, and the way that PlanetLab was designed for the safe development and deployment of services, have implications for the way that DVM can be used by enterprises.

Network Monitoring

A global organization has many Internet users scattered across the planet. Some are Intel customers, some are suppliers, and some are employees. Employees can be within Intel's firewalls or working remotely from home or from customer sites located anywhere on the globe. Services that are utilized include Web sites such as Intel's corporate presence at www.intel.com, various e-commerce applications, and VPN connectivity back into Intel. This requirement for global access can result in Intel's Network Operation Center (NOC) receiving complaints about performance from any spot on the planet to any one of Intel's many DMZ zones. For example, the NOC might get a call from a user in China saying that the response for

an e-commerce application is very poor. Is the problem local to China? Is the problem local to the Internet connection in question? Is the problem Internet-wide? The NOC needs tools to be able to answer those questions.

A key question is where to monitor. The typical DMZ firewall model lends itself to monitoring the DMZ systems from within the DMZ. This ends up creating a monitoring model with limited scope that does not address problems with transit from anywhere in the world to the DMZ. An alternative would be an approach that examined Web logs for performance problems [13] or looked at traffic flow data using Cisco NetFlow* [14]. Because of our traffic volume and the fact that we didn't have Web servers at all of our Internet DMZs, we ruled out this option. It would be extremely useful to be able to proactively monitor for performance problems all around the world using active measurements. Active measurements from regions in the world could be taken from commercial services like Keynote* [15] or from hosts in datacenters strategically placed around the world. Using commercial services would limit the kind of applications we could run to monitor the DMZs and it could be fairly expensive. Deploying our own hosts in the locations around the world from where we want to monitor would permit much more flexibility, but would be even more expensive.

DVM presents a relatively inexpensive and flexible platform for global-scale monitoring, but poses challenges with software distribution and application management.

Security Monitoring

The traditional, closed network control model has disadvantages in protecting the enterprise networks from distributed network attacks because of data inaccuracy, inability to perform overall impact analysis, and lack of data correlation from distributed sources in large networks. As more and more enterprises move towards relatively "open" perimeters (sometimes without realizing it as through unauthorized wireless access points and VPN connectivity) and distributed network environments in order to meet business demands, the associated provisioning and management cost will consequently increase, as will the complexity. The IT infrastructure needs to be able to provision security requests quickly and be pre-positioned and ready for such requests. The notion of trusted and un-trusted network zones is fast changing in today's enterprise network. Enterprise networks are no longer a simple 2-trust level like they were a few years ago with "internal trusted" and "external un-trusted" zones. The requirement for protecting the resources at the service level is becoming more a reality, and the infrastructure to support this is at best expensive and difficult to justify from an IT security standpoint. Also, simply implementing network and service-level security such as firewalls, IPS, anti-virus, and a whole slew of defenses is not sufficient. In order to ensure these complex network and service-level security enforcements are functioning as desired, an automated and proactive security monitoring system is becoming more essential for enterprises. Proactive network security monitoring is required to validate the security implementations, patching, and provisioning of software to ensure it is not vulnerable to the most recent threats and to avoid costly network downtimes, security incidents, denial of service attacks, and worm and malware attacks, all of which impact productivity and service availability. In addition, regulatory and legal compliance requirements, such as the U.S. HIPAA, Sarbanes Oxley regulations and European privacy laws, are getting more strict for all types of enterprises to ensure they are following the rules to protect their assets, resources, and information.

Vulnerability scanning for the enterprise network to ensure compliance to minimum security specifications and auditing of network security policy to ensure it is implemented per the documented enterprise security policy are examples of add-on security monitoring that the enterprise IT would like to deploy extensively but which is limited due to the static nature of deploying these applications. Using the DVM approach, the ability to create instances/clones of systems that would be able to generate the required security monitoring functions would be extremely simplified. In addition, it would help create multiple views for network security assessments and monitoring. For example, in order to assess the effectiveness of network security implementations, such as firewalls, IPS, authentication/authorization, and other security enforcements, enterprises would have to perform the network security assessments/audits/scans from various parts of the network, such as from within the DMZ/internal network and from the external connected network. This would not only help validate the overall picture of the security posture for the network but also ascertain whether the implemented controls are sufficient. With DVM and the ability to "suspend, copy and resume" a VM, network security becomes relatively simple and cost effective. Another advantage of being able to inexpensively create multiple instances of the network security monitoring system would be to increase the speeds and parallelism of the results. Network security monitoring is then transformed from an infrequent and expensive annual or quarterly audit to a proactive one that can identify and fix security vulnerabilities as soon as they appear on the network.

The DVM approach to network security monitoring as discussed above would help reduce the cost of provisioning these relatively complex auditing/monitoring/scanning applications as compared to the traditional method of static provisioning of standalone security monitoring systems. Using the DVM approach

would reduce the capital costs of the hardware and the cost of the provisioning tasks required to maintain physical systems for these functions. Operational costs for network security operational staff would also be reduced as network staff would be able to leverage the network for simplicity of "on-demand" VMs for the network security monitoring functions. Using the classical server/operating system/application model, and not the DVM model, it is almost impossible to monitor to the level required to be proactive enough to identify security gaps before they are widely exploited.

Content Delivery

A content delivery overlay provides a common service to various applications such as distributed file storage and sharing. Each overlay node maintains a small overlay routing table for finding the destination with the shortest path length of complexity O(logn), where n is the network size. But these overlay search algorithms make the underlying network transparent to the overlay and only find the shortest search path in terms of the number of virtual hops in the overlay.

Safe Yet Realistic Experimentation

A challenging aspect of enterprise environments is the difficulty of testing and introducing new or innovative services into an established infrastructure. Changes are strictly controlled because changes in the computing environment can negatively affect critical enterprise services. This is particularly true when introducing new services to an already running physical host. The new service or application may require system libraries and other software that could potentially break existing services if introduced. Moreover, usage loads introduced by new services on existing infrastructure (both network and CPU) can potentially starve existing services. Thus, the traditional enterprise approach is to bundle new hardware with each new service. Deploying new hardware for each additional service severely slows the introduction of new services, adds to the Total Cost of Ownership (TCO), and further complicates change control. Testing of new services is often done in isolated lab environments, where realistic conditions are difficult, if not impossible, to replicate.

Alternatively, the ability to create VMs that are effectively isolated from each other and share resources fairly resolves these problems. The fundamental idea here is to decouple the introduction of new services from the deployment of new hardware. New services can be deployed on existing hardware by allocating VMs in the preferred service locations. The VM isolation shields existing services from library conflicts with new services, which are sequestered in their own individual VMs. Deploying new services on existing servers also speeds the development and testing of new services, in a realistic, closer-to-production environment without impacting existing services and without requiring installation of new hardware.

Virtual Enclaves

Within large and complex enterprises, there is a need to separate mission-critical environments from the rest of the organization. Critical areas like manufacturing should be immune to worms and malware that might proliferate in the rest of the organization, and access to these critical areas needs to be restricted to those individuals who need it. Fundamentally, these critical areas require their own separate enclave. The traditional approach to building these enclaves is to use dedicated hardware, as shown in Figure 2. This approach has several drawbacks. Deploying the entire infrastructure needed to make the enclaves self-sustaining (such as DNS servers) is time-consuming and expensive. If the infrastructure in one of the enclaves goes down, there is no easy way of getting more resources, short of either repairing the down nodes or installing new equipment.

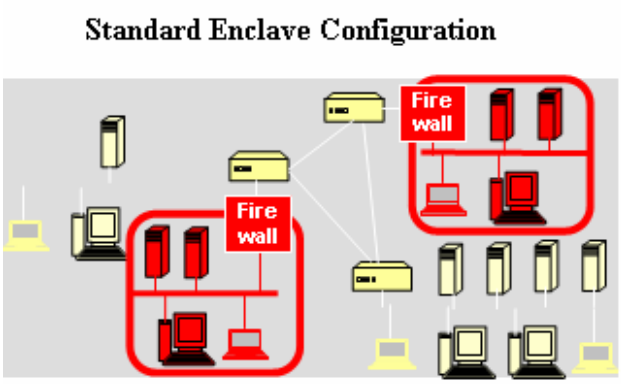

Figure 2: Enclaves currently need to be implemented with physical partitioning and hardware firewalls

The use of DVMs combined with overlay routing technology provides an innovative new way of deploying these enclaves. The VMs required by each service can be joined together with a secure overlay. The overlay isolates and controls access to the VMs as shown in Figures 2 and 3.

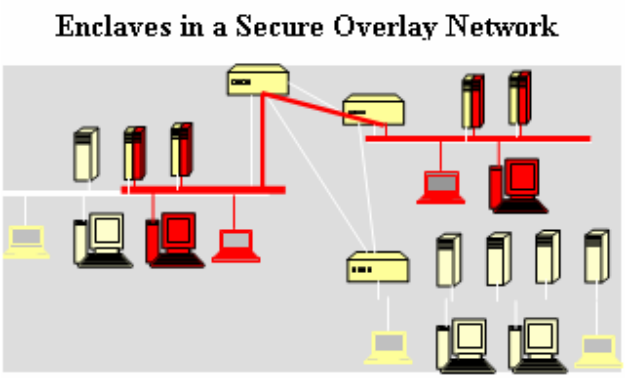

Figure 3: A secure overlay network connects distributed virtual machines

This approach has several benefits. If there is sufficient capacity, no new machines need be deployed. These new virtual enclaves can be deployed in a dynamic manner at a greatly reduced cost. If network segments go down, overlays can route around the problems. If hosts go down, VMs can be moved or allocated on other physical hosts.

Extending Virtualization into Clients

The computational, network, and storage resources of mobile devices (laptops and handheld devices) in an enterprise typically have low utilization and are not available for use by enterprise applications or services that could best utilize them. We envision an environment where the OS with which a mobile user interacts, is one of many OSs that run over VMMs. While the mobile user is interacting with the device, a VM dispatch service can request that the device's VMM create VMs for a variety of tasks, as displayed in Figure 4. These tasks can range from doing computations to running services like file systems, content distribution, and other services like Voice over IP (VoIP). This work can be transparent to the end user and done in the background.

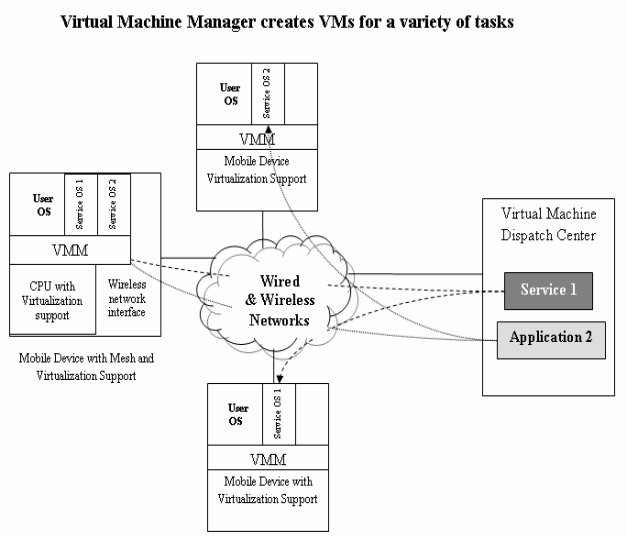

Figure 4: Dispatching DVMs to mobile devices

This architecture extending virtualization into clients and dispatching work via VMs to mobile devices has significant advantages over the current situation in most enterprises. Enterprises typically have low utilization of their mobile resources. Our proposed architecture enables better utilization and can potentially add enormous amounts of shared resources to an enterprise. It also has advantages when it comes to management of systems and services. Having a VMM underneath the OS visible to a user makes it easier to restart or rebuild the users' OS. Services can take advantage of the location of mobile devices and dispatch service instances in VMs that are close to their designated clients. This frees a service from having to manage network parameters such as delay and throughput to a central site. The service is also easier to maintain in the face of node outages because work can be moved between mobile devices.

The similarities between overlay networks and ad hoc networks, along with the technical merits that each introduce through their integration, motivated our interest to investigate and implement an alternative architecture of overlays on wireless mesh networks, called OverMesh [16]. Integrating overlays and wireless mesh enables OverMesh to be flexible enough to serve many networking purposes.

While OverMesh is similar to current ad hoc, sensor networking, and peer to peer computing systems, it is also architecturally distinct from these systems. These are the differentiating properties of OverMesh:

- *Infrastructure-free*: a peer-to-peer edge/access system is suggested over current hierarchical physical formations.

- *Network virtualization*: based principally on a distributed virtual machine overlay strategy.
- *Emergent control and manageability*: utilize learning and statistical inference techniques to off-load human-dependency on operational management and provisioning.
- *Cooperative and adaptive end-to-end control*: tighter layer integration and automation of application-to-network control and management through cross-layer facilities.

OverMesh can be applied to a variety of wireless mesh networks. At its current stage, we chose to realize it on one of the mesh networks that is being actively standardized—the IEEE 802.11s WLAN mesh network [17]. The PlanetLab service architecture [18] was customized and integrated with the WLAN mesh network to manage the DMV-based overlays. We believe that the implemented OverMesh platform will provide a unique testbed for developing a wide variety of services and applications on wireless mesh networks.

An IT Overlay

To experiment with, test, and deploy services using DVMs, Intel's IT department has created the IT Overlay. We envision it as an overlay network that will include hosts within Intel and eventually extend to hosts residing outside of Intel's firewalled perimeter, as shown in Figure 5. Systems hosting VMs have been deployed at five sites within Intel, with more to be added as use of the IT Overlay increases. Intel is also part of the PlanetLab consortium, and Intel IT hosts two PlanetLab sites. Intel has deployed a monitoring service that takes advantage of the distributed nature of PlanetLab.

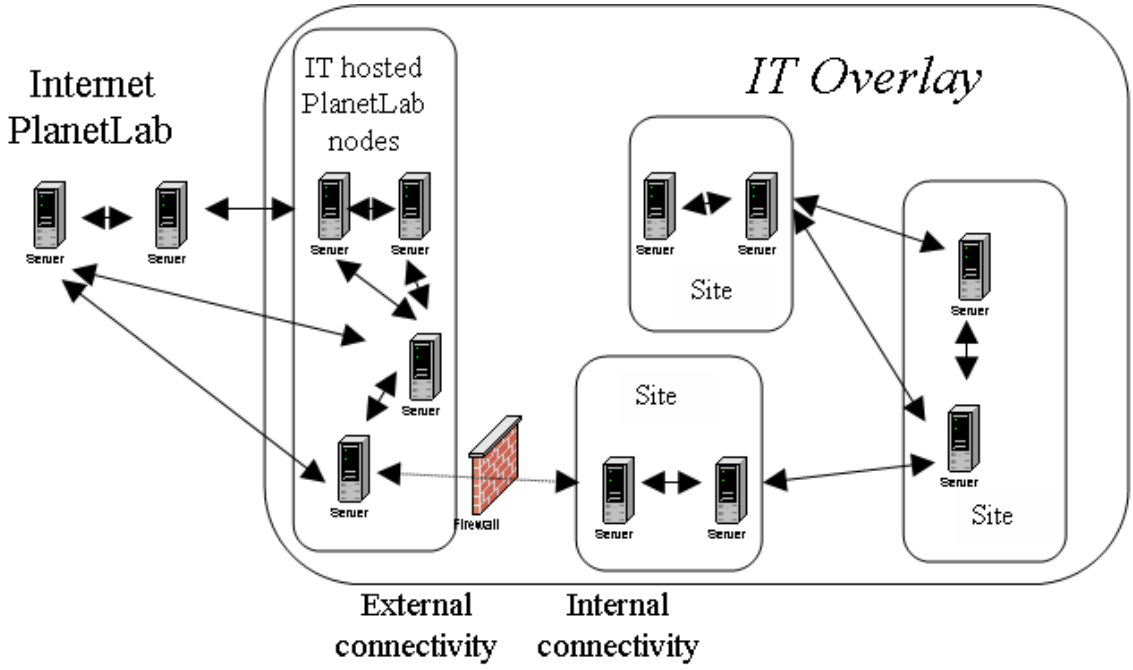

Figure 5: The IT Overlay inside and outside of Intel's firewalls

The internal portion of the IT Overlay will be modeled after PlanetLab. Services will be able to make requests to a central authority that will dispatch VMs to run applications and experiments. We intend to use the interfaces and APIs created by PlanetLab to dispatch VMs, although the Overlay uses Xen* [5] domains for VMs rather than the VServers [19] implementation. We envision running security, network monitoring, and content distribution applications on the IT Overlay and opening it up as a testbed and deployment vehicle for DVM-enabled services.

A CASE STUDY OF SERVER VIRTUALIZATION USING VMWARE

Here is a brief history leading up to the discussion and decision to implement server virtualization for a manufacturing support group at Intel. This organization's server population grew 65% over the last three years with

2006 projections meeting or exceeding this trend. As this organization grew and acquired servers, many of these acquisitions were waterfalled servers being released by other Intel business units. The initial costs made this type of acquisition financially attractive but as we move forward four years, most of these servers have reached their end of life. Another factor is that the primary datacenter for this group is projected to reach complete build out in 12-18 months with no plans to expand. The challenge for the organization was to continue supporting the server growth and replace aging hardware with limited datacenter space while maintaining the same high level of customer support.

This group partnered with a forward-thinking IT group to evaluate, plan, and implement a virtual server environment. In this case study, we walk you through the steps, lessons, hurdles, and successes of this effort. The covered topics include software evaluation, candidate evaluation, hardware design, host hardware setup, virtual server setup, server testing procedures, and initial results.

There are multiple factors to consider when evaluating and selecting server virtualization software. Our team carefully reviewed leading technology products and evaluated different system design options. The two most popular virtualization architectures were host-based virtualization (Microsoft Virtual Server 2005*; VMware GSX 3.1*; Microsoft Virtual PC 2004*; VMware Workstation 5.0*) and full virtualization (VMware ESX 2.5*).

Host-based virtualization requires the installation of a base OS first and then a VMM to be responsible for the execution of all VMs. In addition to the VMM application, the OS can execute other applications (e.g., Anti-Virus, Backup). The downsides to this type of architecture are a heavy performance penalty, high system resources utilization by host system management, additional work to support host maintenance and management, and the upkeep of host security.

The full virtualization design starts off with the installation of a mini kernel (a hypervisor optimized for virtualization) on the physical server. This kernel uses minimal system resources since it focuses only on tasks required for virtualization, and it does not run unnecessary processes or applications. The hypervisor provides full hardware virtualization and distributes the necessary system resources to all VMs. Each VM contains its own OS and cannot distinguish it is running on virtualized hardware. This architecture is ideal for consolidating high-end datacenter solutions.

The decision process to determine the proper virtualization architecture is a critical and time-consuming task. Our team researched benchmarking results of multiple virtualization products and analyzed the cost and supportability options. We prioritized our list of requirements and rated the various software options. We evaluated four products against our requirements and scored their performance. Our requirements included performance, manageability, supportability, stability, security, and a wide range of capabilities. Table 1 is an example of how we did our comparison: (utilizing fictitious data).

Table 1: Product evaluation scorecard example

Product requirements	Product A	Product B	Product C	Product D
1. Performance	10	8	10	7
2. Manageability	6	10	3	8
3. Supportability	8	10	4	10
4. Stability (VM uptime)	7	4	8	3
5. Capabilities	9	8	7	4
6. Security	10	5	7	4
Total:	**50**	**45**	**39**	**36**

After evaluating the scores, we selected a full virtualization software solution for our virtual server environment.

When virtualizing a datacenter, the project's success is directly dependent on choosing the appropriate candidates. We approached this step by defining our virtualization strategy for this business unit. First, we divided their server environment into four categories:

- Ideal candidates
- Candidates
- Potential candidates
- Not a candidate

To categorize each server, we started by collecting data on performance, system utilization, end-of-service timelines, business area, and application specifics. Once the selection criteria data were collected, we mapped our servers against the selection criteria to determine in which virtualization category a server belonged. Once categorized, our team focused on 75 candidates and worked with the business unit to evaluate application specifics and machine load analysis. With our performance evaluations and customer input, we assembled the server requirements:

- CPU consumption
- Required memory
- Disk I/O intensity

- Network requirements
- OS configuration

We used these data when evaluating different hardware platforms for our virtualization environment.

To maximize Return on Investment (ROI), number of virtual systems, and performance, this team's final choice for the virtual host servers was the 4-way Dual-Core Intel® Xeon® processor 7040$^{\Phi}$ 3.0 GHz-2 MB L2 cache system with 16 G of RAM, 2 x 2 Gb, 64-bit/133 MHz PCI-X-to-Fiber Channel Host Bus Adapters and three Network Dual-Port PCI-X 1000T Gigabit Server Adapters.

The hardware selected for this virtual environment is based on an Intel IT standardized platform. The team focused on designing a robust virtual infrastructure without introducing single points of failure. This design would address our customer's primary concern with consolidation of multiple applications to a single physical machine.

The team agreed on an environment that would be immune to hardware failure and power interruptions while possessing the ability to load-balance. The consolidated applications would reside on host servers containing dual power supplies, mirrored hard drives, and teamed network interface cards. The centralized storage solution selected is a multi-terabyte storage area network (SAN) with full fault-tolerant capabilities. Connections to the host servers were made possible through two 2 Gb fiber channel switches configured with redundant paths. This design enables load-balancing, as all VM files reside in a central location and access is possible by each host. Figure 6 shows the details of this design.

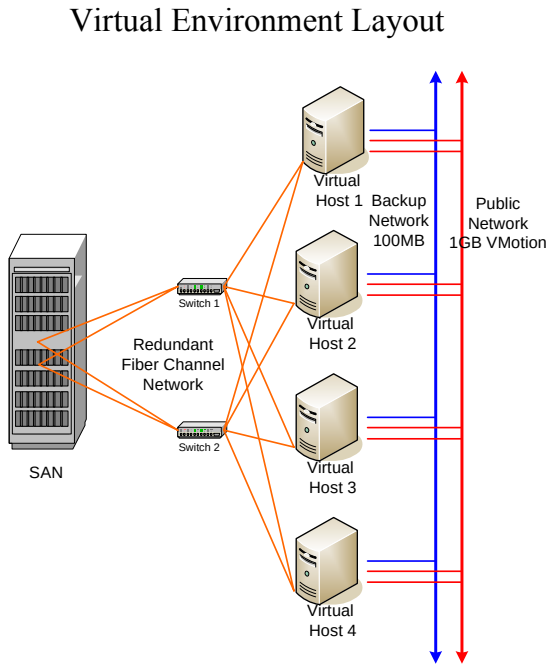

Figure 6: Layout of virtual environment detailing the built-in redundancy

Utilizing an available software feature, VMs can be migrated to another physical host. This migration is done in an active state and causes no server downtime while applications continue to operate uninterrupted. End users are unaware of such migrations. We use this tool to aid in managing downtimes, load-balancing, and other resource alignment needs.

After reviewing multiple virtualization case studies, the team agreed on a 20:1 consolidation ratio limit of VMs to a single physical system. Our initial design consists of 4 physical machines with 15 virtual guests configured on each. This will incorporate 60 ideal candidates targeted for consolidation while reserving resources for potential migrations. In case of physical server failure, the VMs on the failed host would migrate to the 3 remaining hosts as seen in Figure 7. This will permit 5 additional VMs to migrate to each host, respectively, maintaining the 20:1 consolidation and 100% availability.

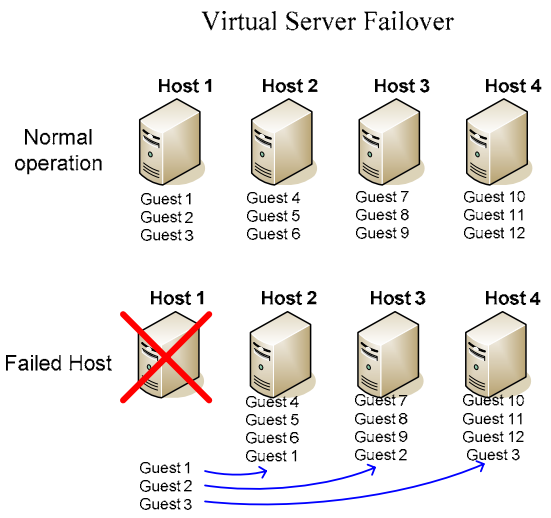

Figure 7: Demonstration of failover when a host system fails

It was easy to justify this project because we were up against several looming obstacles. First, the hardware in use was aged and being purged from our current supportability model. Replacement of this hardware on a one-for-one basis was very costly. Second, the datacenter is constrained by space and power. We needed a solution that would free up physical space in the computing environment. By replacing out-dated servers with virtual servers, we not only saved ~40% on hardware upgrade costs but more importantly extended the capacity of our datacenter. This basic ROI did not investigate costs associated with power, network, AC, etc. Figure 8 shows our first-year ROI.

Return on Investment Worksheet

Component	Qty	Hardware Costs	Software Costs	Rack & Storage	Total Cost
80 servers	80	$9,000.00	$0.00	$6,000.00	$726,000.00
Total to replace aging hardware					**$726,000.00**
Virtualization Software (approximate cost)	1		$50,000.00		$50,000.00
Hardware - (servers, SAN, switches)	1	$243,000.00		$500.00	$244,000.00
Total (Virtualization hardware)					**$294,000.00**
First Year Return on Investment					**$432,000.00**

Figure 8: Our first-year ROI calculation (software costs are approximations)

As the approval, purchasing, and installation of the actual virtualization island was in process, the team utilized a validation environment to begin building server configurations and testing potential candidate servers. To do this, we established an overall test, validation, and implementation plan for our "Ideal Candidate" servers. We notified the owners of these machines of the timeline for testing and identified our criteria for a successful test.

The technical team defined and created a "gold build" server definition (based on the data collected during server classification).

As the testing timeline progressed, server owners were notified three weeks prior to their servers being created. This notification included a detailed timeline for the next five weeks and the requirements for completing a virtualization test. Two weeks before testing began, the server owners met with the virtualization team to discuss special requests, variations from the gold build configuration, and to approve VM resource allocation. After this meeting, the technical team provisioned the new servers and kept them in a "power off" status. The server owners then had to prepare their test plan, success criteria, and migration strategy during these two weeks. The test plan had to include a regression test for any application installed on the server to ensure it executed properly, along with the server functions. Two days prior to the start of virtual server testing, test plans, success criteria, and migration plans were reviewed and approved. Once all requirements were met, the servers were released to the testing team to build their applications, copy data, and configure the server with all required software and application information. The test team did all OS and application testing in a two-week period and met with the virtualization team at the end of the two weeks. When all success criteria had been met, the server was shut down. Once the final hardware landed in the datacenter, the server configurations were moved to the production hardware, restarted, and each test group did a quick validation to ensure the server was in the state it was shut down in.

That was when the virtualization team turned over the "keys" to the server owner and the owners executed their migration plan and moved the physical machine contents to the VM. When the final migration was complete, the physical machines were powered off and removed from the datacenter within 30 days.

RESULTS

When this Intel® business group set out to upgrade its computing environment, the expected results were to have an equal environment to the existing one. After the server virtualization was completed, the expected ROI was realized along with additional benefits such as datacenter floor space, power, cooling, and network relief as well as easier manageability for the IT support team. This installation also built a solid production platform to begin

deploying enterprise services and monitoring capabilities through provisioning of virtual servers for these purposes.

CONCLUSION

In this paper, we explored the issues of implementing virtualization in the enterprise. We analyzed IT services and looked at how those services would be impacted in a virtualized environment.

We looked at several use cases being currently investigated by Intel's IT department. The DVM concept provides a new way of looking at deploying services, and it enables a further use case that we are exploring with our OverMesh implementation and the IT Overlay.

We presented a case study of virtualization of a datacenter in which the VMware ESX Server was used that allowed us to consolidate 20 or even more servers onto a single physical server reducing hardware, electrical, cooling, and administrative costs. Our solution provides robust resource controls for different types of applications, and we can control the levels and limits of CPU, networking, memory, and disk I/O allocated to and used by each virtual system.

Utilizing the virtual environment, IT can quickly create new servers; and virtual servers can be deployed in 30 minutes vs. 60 days to purchase and deploy a physical server. We achieved our goals to minimize our physical footprint in the datacenter, lower our administrative costs, improve our network uptime, and deploy new servers and applications faster. According to Thomas Bittman, research vice president and distinguished analyst at Gartner Inc., "integration of virtualization technology with the operating system is a natural evolutionary step for the x86 platform."

ACKNOWLEDGMENTS

We acknowledge our reviewers Dev Pillai, Mani Janakiram, Greg Priem, Joe Whittle, Nicolas Robins, Vivekananthan Sanjeepan, Robert Adams, George Clement, Raju Nallapa. We also acknowledge the work of Rita Wouhaybi, Gang Ding, Winson Chan, Hong Li, Manish Dave, Claris Castillo, and Stacy Purcell in investigating enterprise uses of Distributed Virtual Machine Technology.

REFERENCES

[1] Goldberg, R., "Survey of virtual machine research," *IEEE Computer Magazine*, 7:34–45, June 1974.

[2] L. Peterson, T. Anderson, D. Culler, and T. Roscoe, "A Blueprint for Introducing Disruptive Technology into the Internet," in *Proceedings of HotNets I,* Princeton, NJ, October 2002.

[3] http://www.geni.org*

[4] Waldspurger, C. "Memory Resource management in VMware ESX Server," in *Proceedings of the Symposium on Operating Systems Design and Implementation* (OSDI '02), December 2002.

[5] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Hegebar, I. Pratt, A. Warfield, "Xen and the Art of Virtualization," in *Proceedings of the ACM Symposium on Operating Systems Principles* (SOSP). October 2003.

[6] Lai, K., Huberman, B., and Fine, L., "Tycoon: A Distributed Market-based Resource Allocation System," in *CoRR: Distributed, Parallel, and Cluster Computing*, 2004.

[7] P. Rundberg and Fredrik Warg, "Freebench v1.03," at http://www.freebench.org/*

[8] Clark, B., T. Deshane, E. Dow, S. Evanchik, M. Finlayson, J. Herne, and J. Matthews, "Xen and the Art of Repeated Research," in *Proceedings of the USENIX Annual Technical Conference*, Boston, July 2004.

[9] JSDL Version 1.0 recommendation, at http://www.ggf.org/documents/GFD.56.pdf*

[10] J. Dilley, et al., "Globally Distributed Content Delivery," in *IEEE Internet Computing*, September/October 2002, pp. 50–58.

[11] Tristan Richardson, Quentin Stafford-Fraser, Kenneth R. Wood, and Andy Hopper, "Virtual Network Computing," *IEEE Internet Computing*, Vol.2 No.1, Jan/Feb 1998, pp. 33–38.

[12] N. Spring, D. Wetherall, T. Anderson, "Scriptroute: A Public Internet Measurement Facility," *USENIX Symposium on Internet Technologies and Systems*, 2003.

[13] Cindy Bickerstaff, Ken True, Charles Smothers, Tod Oace, Jeff Sedayao, and Clinton Wong, "Don't Just Talk About the Weather–Manage It! A System for Measuring, Monitoring, and Managing Internet Performance and Connectivity," in *First Conference on Network Administration* (NETA '99), Santa Clara, 1999.

[14] Cisco Corporation. Netflow, at http://www.cisco.com/warp/public/732/Tech/nmp/netflow/index.shtml*

[15] Keynote. http://www.keynote.com/*

[16] J. Vicente, S. Rungta, G. Ding, D. Krishnaswamy, W. Chan, and K. Miao, "OverMesh: Network Centric Computing," under submission to *IEEE*

Communications Magazine, Emerging Technologies Series, 2006.

[17] IEEE 802.11s ESS Mesh Network working group at http://grouper.ieee.org/groups/802/11*

[18] Andy Bavier, Mic Bowman, Brent Chun, Scott Karlin, Steve Muir, Larry Petersen, Timothy Roscoe, Tammo Spalink, Make Wawrzoniak, "Operation System Support for Planetary-Scale Network Service," in *Proceedings of NSDI '04: First Symposium on Networked Systems Design and Implementation*, San Francisco, March 2004.

[19] Linux VServers at http://www.linux-VServer.org*

AUTHORS' BIOGRAPHIES

Patrick W. Fabian is the business operations manager for Intel's Enabling Technologies and Solutions group in the Technology Manufacturing Engineering organization. He works closely with IT to support his organization's infrastructure and server environment. Patrick holds a B.S. degree in Industrial Management and Computer Science from California University of Pennsylvania. He has over 25 years of IT experience, joining Intel in 1996 as an SAP developer. Along with developing key enterprise applications, his career at Intel includes managing SAP, Web, Teradata ETL, Microstrategy development teams and the infrastructure team responsible for Intel's enterprise data warehouse. His e-mail is patrick.fabian at Intel.com.

Julia Palmer is a senior systems engineer with Information Technology. She joined Intel in 1997 as an Automation Engineer for Fab 18 in Israel. Currently, Julia is leading multiple infrastructure projects for Storage, UNIX*, and Virtualization in the Manufacturing Computing organization. She holds an M.S. degree in Computer Science from Belarusian State University of Informatics and Radioelectronics. Her e-mail is julia.palmer at intel.com.

Justin B. Richardson is a Microsoft Certified Systems Engineer currently working for IT Manufacturing Computing Global Solutions. He joined Intel in 1996 and has used his expertise of server infrastructure and mass-storage solutions to support several manufacturing environments. His current virtualization projects are enabling server consolidation in a large-scale datacenter. His e-mail is justin.b.richardson at intel.com.

Mic Bowman is a principal engineer within Intel's System Technology Laboratory and principal investigator for the Distributed Virtual Machines Strategic Research Project. Bowman received his B.S. degree from the University of Montana, and his M.S. and Ph.D degrees in Computer Science from the University of Arizona. He joined Intel's Personal Information Management group in 1999. While at Intel, he developed personal information retrieval applications, context-based communication systems, and middleware services for mobile applications. Prior to joining Intel he worked at Transarc Corp. where he led research teams that developed distributed search services for the Web, distributed file systems, and naming systems. His e-mail is mic.bowman at intel.com.

Paul Brett joined Intel UK in 2000 as part of Intel´s Online Services group. He is currently working in Hillsboro, Oregon, focusing on Distributed Systems Management tools for developing, deploying and accessing planetary-scale services. From 1988 to 2000, Brett worked on the design and implementation of dependable systems for air traffic control. He is a graduate of the UK's Open University, where he earned a First Class Honours degree in Systems Engineering of software-based systems. His e-mail is paul.brett at intel.com.

Rob Knauerhase is a staff research engineer with Intel Labs. His professional interests include machine virtualization, Internet technologies, distributed systems, system software, and information privacy in the digital world. Knauerhase received an M.S. degree in Computer Science from the University of Illinois at Urbana-Champaign, and a B.S. degree in Engineering from Case Western Reserve University. He holds 14 issued patents, with more than 60 patents pending. He is a senior member of the IEEE and the IEEE Computer Society. His e-mail is knauer at jf.intel.com.

Jeff Sedayao is a staff research engineer in Intel's IT Research Group. He focuses on IT uses of virtualization, including applying PlanetLab and PlanetLab developed technologies to enterprise IT problems. Jeff has participated in IETF working groups, published papers on policy, network measurement, network and system administration, and authored the O'Reilly and Associates book, *Cisco IOS Access Lists*. His e-mail is jeff.sedayao at intel.com.

John Vicente, a senior principal engineer, is the director of Information Technology Research and chair of the IT Research Committee. John joined Intel in 1993 and has 22 years of experience spanning R&D, architecture, and engineering in the field of networking and distributed systems. John has co-authored numerous publications in the field of networking and has patent applications filed in internetworking and software systems. He is currently a Ph.D. candidate at Columbia University's COMET Group in New York City. John received his M.S.E.E. degree from the University of Southern California, Los Angeles, CA in 1991 and his B.S.E.E. degree from Northeastern University, Boston, MA in 1986. His e-mail is john.vicente at intel.com.

Cheng-Chee Koh is a senior systems engineer in Intel's Engineering Computing Group at Santa Clara, California. She received her B.A. degree in Mathematics and her M.S. degree in Computer Science from Indiana University, Bloomington. Her current interests include messaging, interactive computing, information security, and virtualization. Cheng-Chee has been with Intel for 13 years. Her e-mail address is cheng-chee.koh at intel.com.

Sanjay Rungta is a principal engineer with Intel's Information Services and Technology group. He received his B.S.E.E. degree from Western New England College and his M.S. degree from Purdue University in 1991 and 1993, respectively. He is the lead architect and designer for the Local Area Network for Intel. He has over 13 years of network engineering experience with three years of experience in Internet Web hosting. He holds one United States patent and three pending in the area of Network Engineering. His e-mail is sanjay.rungta at intel.com.

[Δ] Intel® Virtualization Technology requires a computer system with an enabled Intel® processor, BIOS, virtual machine monitor (VMM) and, for some uses, certain platform software enabled for it. Functionality, performance or other benefits will vary depending on hardware and software configurations and may require a BIOS update. Software applications may not be compatible with all operating systems. Please check with your application vendor.

[Φ] Intel® processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. See www.intel.com/products/processor_number for details.

For further information visit:

developer.intel.com/technology/itj/index.htm