

# Academic Poster Abstracts VMworld 2008, Las Vegas

Poster Title	Page
A Clean Environment for Web Applications Using Lightweight Virtualization.....	2
Internet Cleanroom: A System Using On-Demand Virtualization to Enhance Client-Side Security while Keeping Usability.....	3
Virtualized Gaming Environments for Teaching Students About Complex Networked Systems	4
Coordinated Resource Management on Virtualized Multicore Platforms.....	5
Predicting VM Behavior for DRS & DPM Cost/Benefit Analysis.....	7
Decentralized Data De-Duplication in VMware's SAN File System.....	8
Improving Storage VMotion Performance with Disk I/O Heuristics.....	9
Bonsai: Shrinking Virtual Disks Via De-Duplication and a Global Common Disk Block Store.....	10
Virtualizing Shader Model 3.0 for More Realistic Graphical Effects in a VM.....	11
Improving the Performance of VMware Workstation Resume Using Working Set Estimation...	12

*Abstracts cited on pages 7, 8, 9, 10, 11 and 12 are copyright protected by VMware, Inc. All other abstracts have been reproduced with the permission of the respective authors.*

## **A Clean Environment for Web Applications Using Lightweight Virtualization**

Jiang Wang, Yih Huang, Angelos Stavrou and Anup Ghosh, *George Mason University*

Vulnerable applications bring significant risks for the operating system, which may host sensitive data and access to other sensitive systems. Currently, most network applications and non-networked applications run side by side on a single operating system. Once a network application is compromised by an outside attacker, the whole operating system is “owned” by the attacker, and often unbeknownst to the system owner. As a result, people may lose sensitive data such as passwords, credit card numbers, and on-line banking credentials. Anti-virus software detects the virus, worm and other malware largely by signatures, so they are largely ineffective in detecting zero-day attacks. Intrusion detection systems face similar problems in being reactive in detecting threats.

Alternative approaches that use privilege-based access control, such as IE7, can limit the damage to the operating system; however, it cannot prevent malware that can escalate its privileges. Another method to protect network applications is using hardware virtualization to separate applications from other parts of the operating system. Running network applications in a virtual machine provides root security for the host operating system as long as the virtual machine monitor's integrity remains intact. But hardware virtualization introduces heavy overhead, because each virtual machine has its own operating system.

In order to balance the isolation and performance, we use operating system level virtualization to isolate every instance of user applications, and use a stackable file system to separate the system and application binaries from user's data. Operating system level virtualization has a better performance because it does not virtualize the hardware, but shares the operating system kernel with many virtual environments. By using operating system virtualization, we can isolate every instance of network applications while maintaining good performance. In addition, we use a stackable file system, unionfs, to separate system binaries from user data.

We create the virtual environment on-demand for each instance of an application when it is opened. After termination, our system removes all the changes to the virtual environment. Our goal is to run every instance of applications, such as web browser and office productivity software and media players in an isolated container and protect the basic binaries of virtual environment from being changed by malware.

We implemented a prototype to demonstrate our approach; the results show that it has good scalable performance. The prototype is based on OpenVZ — an operating system level virtualization engine for Linux — and unionfs file system. We first created a base operating system which contains minimal operating system binaries and a web application – the Konqueror browser in our implementation. However, our architecture can be used for any application, such as work processing software and email client. We mount the basic operating system as a read-only layer of unionfs plus another empty writable layer together to construct a new virtual environment. Since the basic operating system is mounted as read-only, it will never be changed by the software in the virtual environment. All the writes to the file system go to the writable layer. By this means, we can easily remove the whole writable layer after the application is closed, while not causing faults for the applications. Moreover, the read-only layer of basic operating system can be mounted to multiple virtual environments at the same time, saving a lot of disk space. We compared the performance of OpenVZ virtual environment with VMware Workstation and native Linux.

## **Internet Cleanroom: A System Using On-Demand Virtualization to Enhance Client-Side Security while Keeping Usability**

Jiang Wang, Anup K. Ghosh and Yih Huang, *George Mason University*

Vulnerable network applications such as web browsers, email clients, media players, and office productivity software bring significant risks to the operating system and users. To protect the user's computer, currently anti-virus software is the standard defense used to detect malware largely by signature matching. As a result, anti-virus software usually cannot detect new attacks and is largely ineffective at addressing metamorphic code and other current malware threats. Intrusion detection systems face similar problems in being reactive in detecting threats.

Sandboxing is another method to protect the end user's computer. Different levels of sandboxing are feasible, ranging from language software fault isolation, process level system call mediation, to hardware virtualization level. Unfortunately, language and process level sandboxing are susceptible to bypass, and current hardware virtualization sandbox, such as the Tahoma system, separates the applications as well as the data that they usually share, therefore sacrificing usability.

To provide strong isolation between applications while keeping usability, we use hardware virtualization to create virtual machines on-demand and supply a persistent storage to save useful data. We separate network applications or applications that run untrusted code in their own virtual machine and seamlessly integrate it with user's host desktop by using a redirection module. Unlike the Tahoma system, which modifies the source code of the browser applications, our method is compatible with current proprietary commercial applications.

We implemented a prototype called Internet Cleanroom on Microsoft Windows. It uses VMware Workstation as the virtualization layer and includes redirection modules, a control console and dispatcher modules. To enable seamless integration with the standard desktop experience, we intercept Windows API calls from the host operating system that create new processes, and then re-direct the creating process events to a host dispatcher. The dispatcher then redirects network applications to a pre-built virtual machine. If the virtual machine is not started yet, the control console starts it first. The guest dispatcher running inside the virtual machine launches the network application for the user. The control console also brings the virtual machine to the front and show its window if it is minimized. We use VMware Workstation as the virtualization engine.

In each virtual machine, we install a basic Windows XP distribution plus the applications that should be redirected to each virtual machine (VM). We use snapshot function of VMware to start the virtual machine in a pristine state as well as to restore the VM to its pristine state periodically or by user control. On the GUI of control console, an indicator shows the health state of each virtual machine. When the time reaches the configured restore time or if malware detection software in the virtual machine detects some malware, the indicator turns from green to red to warn the user. The guest dispatcher probes the log file of malware detection software to find the alerts.

For persistence of data, we share a folder on the host to each guest as the persistent storage for the virtual machine. The user needs to manually save his data to this folder for persistence beyond the current session. A program running on the host (which is trusted) moves the files in the persistent storage to another host folder after the virtual machine terminates. This prevents the malware in the guest to access all the persistent data while letting the user keep data between sessions. We evaluated the performance of Internet Cleanroom for five popular applications.

## **Virtualized Gaming Environments for Teaching Students About Complex Networked Systems**

Kirill Kourtchikov and Joel Wein, *Polytechnic Institute of NYU*

Complex distributed systems (CDSs) and networks are ubiquitous in the modern computing experience; therefore, it is critical that computer science curricula train students who are able to think about the design, implementation and management of such systems in a serious fashion, and who are able to understand such systems and contribute to their evolution.

Giving students such training is not, however, an easy task, as they require a wide range of skills, both purely technical and meta-technical. Technically, they need to understand in a deep way multiple layers of the network protocol stack, they need to be able to think about designing systems of significant scale and with significant heterogeneity, and they need to be able to think about how to secure these systems and insure their resiliency. Meta-technically, they need the ability to interact in or lead cross-functional teams, to make hypotheses about problems based on scattered and incomplete evidence, to investigate issues fearlessly despite only partial knowledge of the system, and to plan for management and evolution of such a system.

We believe that virtualization is a key enabling technology to bridge the gap between current curricula and current needs. By leveraging recent advances in virtualization technology, we can develop significantly richer and more engaging teaching environments that can easily be shared amongst multiple institutions.

Second, we hypothesize that many of the skills we wish to teach our students can be well taught in a gaming environment; and are working to develop collaborative multi-player games that teach numerous skills necessary in CDS design, implementation, management and troubleshooting. These games are fundamentally enabled by virtualization, which lets us flexibly construct multiple complex environment instances to support the game.

In this poster we will report on progress using VMware technology to support a multi-player competitive CDS-troubleshooting game. Students work in teams to avoid degradation of ad revenue in a distributed ecommerce application. We will present the principles we believe can be well-learned in such an environment and preliminary experience with the use of virtualization to achieve these goals.

## **Coordinated Resource Management on Virtualized Multicore Platforms**

Mukil Kesavan, Sanjay Kumar, Ripal Nathuji, Adit Ranadive, Ada Gavrilovska and Karsten Schwan,  
*Georgia Institute of Technology*

Virtualization technologies like VMWare's ESX server, the Xen hypervisor, and IBM's longstanding mainframe-based solutions have gone beyond becoming prevalent solutions for resource consolidation, but are also enabling entirely new functionality in system management. Examples include dealing with bursty application behaviors or providing new reliability or availability solutions, e.g., for coping with emergencies. Management is particularly important in Utility Data Center or Cloud Computing solutions, which use virtualization to offer resources to applications run by different customers and must therefore, provide different levels and types of services to diverse codes running on the same underlying hardware (e.g., time-sensitive trading systems jointly with high performance software used for financial analysis and forecasting). Here, guest applications are packaged as sets of virtual machines (VMs) and such VMs are provided just the resources they need and only when they need them, rather than over-provisioning underlying platforms for worst case loads. This creates opportunities for reductions in the operational costs of maintaining the physical machine and datacenter infrastructure, as well as the costs of power consumption associated with both. The latter is becoming important as improved densities enabled by small form factor blade platforms and multicore processor architectures are making it difficult to sustain datacenter environments.

Significant challenges remain before it will be possible to effectively and at low cost manage virtualized systems or more generally, manage entire compute clouds and their potentially rich collections of VM-based applications. Technical elements of these management challenges range from scalable monitoring and control mechanisms, to effective algorithms and methods that make and enact management decisions, to high level decision making processes and policy considerations. A specific problem in this context is that management actions must typically touch upon multiple resources in order to be effective, e.g., power management that must address CPUs, memory, and devices; end-to-end performance management that must consider CPU cycles and network bandwidths; and reliability management that must consider resources spanning multiple machines or even different datacenters (i.e., clouds).

Our group adopts a coordinated resource management approach for dealing with multiple resources. This 'active' approach can react to changes in any one resource concerning its effects on the multiple VMs using it. For instance, a change in allocation of network bandwidth to a VM may cause changes to its CPU allocation, to avoid inappropriate (too short or too long) message queues at the network interface. Toward this end, the coordinated approach includes methods for tuning the use of each of the resources required by a VM or set of cooperating VMs, using mechanisms that change CPU allocations for a compute-intensive workload, while additional methods are used to maintain its IO allocation to low levels, or vice versa for workloads reaching an IO-intensive phase. Additional technical solutions must address the fact that such mechanisms and the associated allocation algorithms must exploit the multicore nature of future cloud nodes. This includes dealing with multiple applications VMs sharing a single platform, which can lead to violation of VMs quality of service requirements due to hardware effects like cache thrashing, IO interference, memory bandwidth contention, and similar issues. Finally, we must offer methods for making current and future cloud infrastructures energy-aware.

The coordinated approach is based on information about platform resources and the applications using them. To characterize applications, we formulate the needs of each VM as a multi-dimensional vector, termed Class of Service (CoS). The vector expresses the relative importance, priority, or resource expectations of a (set of) guest VM. Since CoS specifications will typically not be complete and sometimes not even known, the CoS vector of a running application is continually refined using online monitoring. Using these resource specifications, active

management is performed in multiple ways; (i) using a black box approach, based on historic (i.e., observed) information regarding VM behavior to 'guess' its future trends, coupled with an algorithm which makes continuous adjustments in the platform resources allocated to the VM; or (ii) incorporating mechanisms that permit external actuating events to trigger VM or application-specific resource allocation decisions. Such external events may be provided by the cloud administrator, statically at VM creation time (e.g., a financial application may provide information requesting the doubling of its resources on the 15th of each month due to an anticipated increase in loads), or it may be dynamically provided by the guest VM. The latter option requires the VM to include a management agent that knows how to export management-related information regarding its resource needs or QoS requirements (e.g., using emerging standards such as WSDM).

We have performed coordinated management for multiple types of resources, including CPU, I/O and power resources on modern multicore platforms. Results indicate that we can meet application-level SLAs with significant resource reductions, using up to 50% less CPU resources, or offer an up to 34% reduction in power consumption. These experimental evaluations use industry-standard benchmarks, including RUBiS, Hadoop, Nutch, and SPEC, as well as benchmarks derived from the HPC community. The current implementation of the mechanisms is targeted on Xen-based platforms, but we are currently porting and, hopefully, verifying the importance of the coordinated active management approach for ESX-based platforms. We expect to be able to include the ESX based data in our poster presentation.

## **Predicting VM Behavior for DRS & DPM Cost/Benefit Analysis**

Chengwei Wang, VMware Intern, *Georgia Institute of Technology*

To better analyze and handle the impact of long-latency operations such as host power-off/-on and VMotion of large memory VMs, DRS operation could move toward a more proactive model, for which accurate prediction of VM behavior is important. In this project we first define the underlying concepts for VM behavior prediction including prediction goodness measures and the representation of stable characteristics. Three predictors, Oracle, simple reactive predictor and GlitchFree have been implemented and evaluated. We identify two points of improvement to achieve better prediction accuracy: (i) Mitigating glitch behavior of VM workloads; and (ii) using a forward-walk approach to get consistent history information. We have implemented a new predictor, GlitchFree, which incorporates these points and provides 1.7X average improvements to simple reactive approach on prediction accuracy. These predictors are evaluated on real customer workload traces extracted from drm dump files. Our ongoing and future work includes the evaluation of new predictors--namely, a probability-based GlitchFree predictor and a linearly-decreasing stable time predictor--and refining of our current goodness measure.

## **Decentralized Data De-Duplication in VMware's SAN File System**

Austin Clements, VMware Intern, *Massachusetts Institute of Technology CSAIL*

Recent inroads towards online duplicate data elimination have demonstrated the feasibility of de-duplication beyond archival and backup applications. However, existing approaches do not complement the scalability and fault tolerance of decentralized shared storage systems such as those that form the cornerstone of large virtual infrastructures. Our system builds atop VMware's shared storage file system to provide completely decentralized, scalable, online de-duplication with minimal performance overhead.

Our working prototype has demonstrated a two-to-one reduction in the storage requirements of actively used virtual machines. De-duplication is highly effective as part of a virtual infrastructure because one datacenter may store hundreds of identical copies of system files, application files, and files shared between users of the infrastructure. Our system detects and eliminates such duplication on a cluster-wide scale, leading to lower storage and management costs, shorter backup and recovery times, and better cache utilization.

## **Improving Storage VMotion Performance with Disk I/O Heuristics**

Yan Tang, VMware Intern, *Cornell University*

Storage VMotion (SVMotion) is essential for increasing the mobility of virtual machine in a datacenter. It enables the live migration of virtual machine disks across storage arrays.

SVMotion HEAT project is targeting at reducing the migration time of Storage VMotion. We introduce two optimizations, hot block tracking with HEAT map and progressive updating change map. A HEAT map is used to track the hotness of disk blocks and identify frequently-changed hot blocks. It skips the copy of hot blocks in the pre-copy phase and leaves them to the final copy phase. Progressive updating change map eliminates the redundant copy of non-dirty blocks.

Our experiment results show that, SVMotion HEAT reduces the migration time by 10% to 30% in various workloads and configurations. Besides SVMotion HEAT, we also have disk write latency data to show that disk block coalescing potentially will save more time by issuing one single large IO instead of multiple small IOs when transferring dirty blocks.

## **Bonsai: Shrinking Virtual Disks Via De-Duplication and a Global Common Disk Block Store**

Matt Aasted, VMware Intern, *Franklin W. Olin College of Engineering* and  
Meera Shah, VMware Intern, *Carnegie Mellon University*

VMware has gone a long ways towards freeing machines from their physical restraints. The machine-as-a-file paradigm simplifies the copying, sharing and transporting of virtualized machines. However, this shift hasn't transformed usage to the extent of, say, the move from CDs to MP3's which today are copied carelessly, sent in e-mail, stored on flash cards etc. Transportation of virtual machines is still difficult, requiring external drives, long compression times and multiple media disks, specialized networking and lots of time. Unlike digital music where size has largely plateaued, the basic size of virtual machines is likely to grow over time with new larger operating systems and applications.

How can VMware make virtual machines more portable (including time effort)? Size is key, and the virtualized disk dominates the size of a VM. The enabling factor is that virtualized disks tend of have a lot of redundancy both between disks that share a common operating system or application set and within a single disk. The goal is to collapse this redundancy in a way that's transparent to the guest and end user with no issues or complications around licensing, installing or upgrading software and minimal security and performance impacts (potentially performance gains). In this way we can dramatically reduce the size of VMs and make virtual machines more space efficient and portable on local/cluster, company, and global domains.

Most system software today is a monoculture and therefore there is a high level of replication between VMs in the operating system and supporting libraries. Additionally, there tend to be repeated blocks within a given disk image. This duplication comes from several sources, including multiple installations of a library, dll caches and system restore partitions. There is often further redundancy within a block, and therefore size may be saved by compressing at the block level. These sources of duplication provide the opportunities that a de-duplication system can exploit. Therefore, we created Bonsai to reduce the size of disk images.

Bonsai has three components. First, it de-duplicates within a single virtual disk image (VMDK). This is done using a strong cryptographic hash and the assumption of no collisions. Second, it uses a central block store on a set of servers to de-duplicate between VMDK images. The server assigns common blocks a unique identifier; it provides identifiers in response to hashes and blocks in response to identifiers. If the server has the block, the client stores holds only the identifier of the block when the VM is not being used. Finally, the unique blocks not found on the server are compressed and stored locally on the client.

A prototype of this system was created by VMware interns Matt Aasted and Meera Shah during the past summer under the direction of Saman Amarasinghe and Tim Garnett. This system could shrink and expand a VMDK file, de-duplicating both locally and against the server. We found that we could expect compression ratios of greater than 50% on almost all non-trivially sized VMDK files. Furthermore, if the statically-built server had VMDK files similar to the VMDK, the results were substantially better, with 50% to 85% of blocks in an installed and patched system found on the server. This results in compression ratios of 50% with a low hit rate to as high as 95%.

Future directions for the project include integration into workstation and extension of the VMDK file format to fetch blocks at runtime rather than requiring the entire file to be expanded in advance. A global de-duplication service is also being considered, as well as using de-duplication in the storage for cloud computing.

*© 2008 VMware, Inc. All rights reserved.*

## **Virtualizing Shader Model 3.0 for More Realistic Graphical Effects in a VM**

John Bauman, VMware Intern, *Carnegie Mellon University*

Over the years, the capabilities of graphics cards have advanced. They have transitioned from the dumb framebuffers that could only move pixels from memory to screen, to devices designed to quickly put colored polygons on a monitor, and finally to programmable stream processors, capable of creating amazing graphics effects in real time.

Along with the change in hardware, the way that we program these cards must change as well. Developers must create programs to run on these graphics cards, and Shader Model 3.0 is a newer programming model for these cards that lets developers better exploit their potential for realistic graphics effects, such as high dynamic range imaging and displacement mapping.

With this poster, I give an overview of how 3D graphics work, what shaders are, what the features are that Shader Model 3.0 enables, and how we implement Shader Model 3.0 to let virtual machines use these advanced features.

## **Improving the Performance of VMware Workstation Resume Using Working Set Estimation**

Irene Zhang, VMware Intern, *Massachusetts Institute of Technology*

Unlike other virtualization overheads which are measured in CPU clock cycles, the time required to restore a Virtual Machine (VM) from a checkpoint on disk is measured in tens of seconds. Attempts to hide this latency with "lazy" restore techniques (in which the guest is started before the restoration of memory is complete) often degrade the user experience. Unless the memory checkpoint is cached on the host, the guest will be unresponsive during almost all of the lazy restore due to disk-thrashing when the guest accesses physical memory that has not yet been restored.

The performance of lazy restore can be greatly improved by prefetching the working set of guest memory pages before starting the guest to reduce disk-thrashing. Using simple working set estimation methods, the guest can be used immediately after the VM starts. In our tests, the total time to restore a snapshot to a usable state was reduced from 76 seconds to 36 seconds, an improvement in performance of more than 50%.