Xen Summit North America at Oracle
Abstracts

Ignacio M. Llorente & Constantino Vazquez
The OpenNebula VM Manager

OpenNebula is an open source virtual infrastructure engine that enables the dynamic placement of VMs on a pool of physical resources. OpenNebula extends the benefits of Xen virtualization platform from a single physical resource to a pool of resources, decoupling the server not only from the physical infrastructure but also from the physical location. It provides a powerful and agile CLI and API for monitoring and controlling large scale VM deployments, including networking and image management, and a flexible and generic framework to define new policies for capacity provision. Additionally, OpenNebula provides plugins to access Amazon EC2 to supplement local resources with cloud resources to satisfy peak or fluctuating demands in the service workload.

The aim of the presentation is to describe the main features and benefits of OpenNebula and to show a demonstration of the dynamic scaling of virtual infrastructure using Amazon EC2 resources. MORE INFO: http://www.OpenNebula.org

Kevin Tian
The On-going Evolutions of Power Management in Xen

Full Document: http://www.xen.org/files/xensummit_oracle09/xensummit_intel.pdf

While virtualization and power management technologies stay as hottest topics in computer research, increasing attractions have emerged when two are combined. Researches in this area [1] [2] [3] have focused mainly on global coordination framework and power budget policy in large scale data center. On the other hand, insufficient exploitations are seen on role of two important factors: green computing capabilities of VMM, and distinct designs of guest OS-es.

This paper looks into new evolutions of on-line power management technologies in Xen, relating to both hardware power-saving features and software optimizations for power. Several comparisons are made to help understand associated effects regarding to power efficiency: first, compare to old Xen revision for achieved improvements (Figure 1); second, compare to native OS for remaining gap; last, comparison between Xen and KVM is also presented which is the 1st data

as known in this area. Latter two are included in Figure 2.

Then, another interesting work we present is to measure impact factors from VMs. We compare possible factors at two levels: OS difference (HVM vs. PV, Linux vs. Windows, etc.), and also power friendly implementations within OS-es (HZ, tick-less, etc.). Other factors such as timer mode could also draw some difference. The final result highlights the key role of VM itself in overall power efficiency, which could assist user choosing appropriate guest OS-es in real usage.

Ze'ev Maor
CVF – Client Virtualization Framework

Full Document: http://www.xen.org/files/xensummit_oracle09/xensummit_neocleus.pdf

Endpoint virtualization is emerging as the next big thing in the PC industry. While server virtualization has become an industry standard, we're just beginning to tap into the huge potential of virtualizing the endpoint.

Realizing this potential requires two components: an endpoint oriented hypervisor, and a comprehensive framework for virtualization based software development.
We present the "Client Virtualization Framework", a modular software development and run-time platform, allowing developers to leverage state of the art virtualization technologies while focusing their development efforts on their core business.

Steven Maresca
Project Zentific

Zentific is a web-based management interface for the effective control of virtual machines running upon the Xen(R) hypervisor. Designed to maintain a high level of reliability, security, scalability, and ease of use, the interface serves both administrators and end-users of all proficiency levels. In the spirit of the open-source projects which have made Zentific possible, the application is itself open to the community.

Foremost in discussion is an overview of Zentific, its features, and possible use cases. Zentific can provide management of virtual machines from anywhere in the world. With user management and role based privileges, the same application provides and controls administrator and end-user resources. Unlike some management systems, Zentific can be placed into an already established infrastructure.

Virtual machines can be controlled in the usual basic manners (start, stop, pause, etc), but also in terms of memory allocation, revisioned configuration management, disk snapshots and resizing, and graphical/text console. A wealth of statistics (real-time and historical) are presented, including disk/network i/o, CPU, and memory utilization. Zentific is particularly well suited to cases where high value is placed upon user access and management permissions for a collection of virtual and physical machines; specific examples include hosting companies and educational institutions.

Of equal importance are the major components of Zentific. Providing a vast amount of data is the statistics polling daemon; utilizing Xen's low level C libraries, this is the only component installed

upon physical hosts and places special emphasis upon efficient, non-invasive data collection. Some information has not been extracted by other libraries, and as a stand-alone component, the daemon may be useful in expanding many projects. This data is aggregated and stored for later analysis upon a central Zentific server. This server also provides the backend with which the browser frontend communicates. This frontend is perhaps the most essential portion of the project, as it organizes and presents physical host and virtual machine data to the user. Finally, an optional tool installed within virtual machines (both Windows and Linux) provides further augmented data and control via integration with xenstore.

Finally, while the primary focus of Zentific has been Xen support, its platform-agnostic architecture permits easy integration with other virtualization platforms. Even now, swift progress is being made for support VMWare, KVM, and others. Work is ongoing and community contributions are welcomed.

Noboru Iwamatsu
Paravirtualized USB Support for Xen

USB is the most standard connection method for PC peripheral devices.

Supporting USB devices within guest domains is quite essential especially for the client-side virtualization.

In Xen Summit Asia 2008, we introduced the development status of our paravirtualized USB driver. Xen already has two options for using USB devices within guest domains (PCI pass-through or Qemu's UHCI emulation), but paravirtualized USB driver provides more flexible and effective alternative to Xen.

We have improved the implementation since last summit, and achieved a good performance. In this summit, we would like to show the design, implementation and some performance results.

Yunhong Jiang
Towards Mission-critical Xen

One of the major usage models in virtualization is server consolidation. This means multiple services running in guests are hosted in one physical system.

An single hardware error in the host platform may cause the whole system crash and all guest services terminated. This imposes unique challenges to service availability in virtualization environment.

In this presentation, we will discuss how to extend Xen towards mission critical, by enhancing its error handling capability. The basic idea is to do proactive self-healing to prevent error from happening, to recover from non-critical error and to do error containment for critical error to reduce negative impact. More specifically, here we will cover CPU, memory and PCI-E hardware error handling. Furthermore, we will discuss the guidelines on how to use the Xen error handling API, to further improve Xen robustness.

Akio Takebe
PCI pass-through techniques improve performance, security, and so on

If we use guest environment without emulation disk, it's great. To use such the guests, we need to boot them from SAN/SAS of their PCI pass-through.

We made BCV[*1] support for HVM guest BIOS. We succeeded in booting HVM guests from SAN/SAS of their pass-through card. But the guest BIOS of xen still needs some features for BCV support. They are BCV table, BCV priority, runtime functions, pmm services[*2]... We talk about their feature and discuss about improvements of the guest BIOS.

[*1] BCV: Boot Connection Vector
[*2] pmm services: POST Memory Manager service
REF: BIOS Boot Specification v.1.0.1

And also, we will briefly report current status of the pvSCSI driver improvements:

a) adding new assignment mode in order to assign whole HBA to specified guest
b) shorten processing time at xend for scanning all disks existing.

Joel Becker & Tao Ma
REFLINK operation in ocfs2

Computers require storage to host their operating system software. This is often a physical disk in a standard computer. Virtualized environments use many techniques to provide virtual machines with the appearance of a physical disk. It is very common for the virtual disk to be backed by a regular disk file in the hypervisor. These disk images are often large and change very little. Backing them up requires significant space, leading to various snapshot technologies. Many virtual machines have near-identical images; it is always preferable to share the identical portions.

We are currently adding an operation named REFLINK to the ocfs2 filesystem. The REFLINK, or "reference counted link", operation creates a new file in the filesystem that shares all of the data extents of a source file. It's calling semantics are identical to the link(2) system call. The source must not be a directory or special file, and the target must not exist. Where the link(2) call creates a new directory entry for the same inode, the REFLINK operation creates a new inode. The new inode shares all of the data extents of the source file. If either file is modified, copy-on-write semantics are followed.

The REFLINK operation provides systems that are limited to inexpensive storage the ability to snapshot and clone virtual machine images. Because the sharing is at the level of data extents, only those data extents that are shared need to be tracked. The sharing of a data extent is not limitless - 2^32-1 inodes can share a particular data extent - but it is high enough that it should suffice all by the most extreme circumstances.

Xiaowei Yang
Evaluation and enhancement of Xen scalability

One of the most important strength of virtualization is consolidation, that is to pack multiple VMs onto one physical platform. As modern computer architecture evolves, the consolidation-ratio gets bigger and bigger. Given that, the scalability of a virtualization implementation becomes a critical requirement, besides a single VM's performance. Previously few studies cover this topic on Xen, especially in an analog enterprise usage model. Our work focuses on that area - evaluate and enhance Xen scalability.

To understand Xen scalability in subsystem level like CPU, scheduler, memory and IO, simple benchmarks are selected. To measure it in a whole, vCon is chosen, which is a comprehensive benchmark designed to reflect the real enterprise virtualization usage. To identify the potential issues, Xen-specific performance monitoring tools (e.g. Xenoprof, xentrace and xentop) and generic ones in Linux are used. Generally Xen scalability is in a good shape in the aspects of CPU and memory virtualization. At IO side, it improves a lot with hardware support (VT-d & VT-c).

Moreover, several issues have been addressed. For instance, vCPU migration is identified too frequent in some multiple VMs test. Per our measurement, migration cost could be big and sustain for micro-seconds in some cases. Another example is stubdom may make scalability worse, as more vCPUs are introduced and the close relationship of stubdom and pairing HVM is not considered by the scheduler. Our experimental analysis will show how those issues impact Xen scalability in specific areas, and corresponding enhancements will be proposed.

Chris Smowton
Flexible and Secure Hardware 3D Rendering on Xen

Full Document: http://www.xen.org/files/xensummit_oracle09/xensummit_chris.pdf

I describe the design and implementation of a highly general system for hardware accelerated 3D graphics rendering by unprivileged guest VMs. Tungsten Graphics' Gallium3D driver architecture is used to provide a hardware-independent interface for remoting.

Derek Murray
Satori: Enlightened page sharing

Full Document: http://www.xen.org/files/xensummit_oracle09/xensummit_satori.pdf

We introduce Satori, an efficient and effective system for sharing memory in virtualised systems. Satori uses enlightenments in guest operating systems to detect sharing opportunities and manage the surplus memory that results from sharing.

Our approach has three key benefits over existing systems: it is more able to detect short-lived sharing opportunities, it is efficient and incurs negligible overhead, and it maintains better performance isolation between virtual machines.

We have implemented and evaluated a prototype of Satori for the Xen virtual machine monitor. In our evaluation, we show that Satori quickly exploits up to 94% of the maximum possible sharing with insignificant performance overhead. Furthermore, we demonstrate workloads where the additional memory improves macrobenchmark performance by a factor of two.

Chuck Yoo
Real-time and VMM

As VMM is considered for embedded devices, new requirements arise.

One of them is real-time support. Traditionally, real-time support involves a scheduling policy in which all the real-time tasks can finish their jobs within given deadlines.

However, real-time task scheduling is not guaranteed in a virtual machine-based system because the physical resource scheduling and management is done at the virtualization layer.

Furthermore, real-time system support in VMM requires huge engineering efforts because real-time schedulers need specific system parameters such as execution time of tasks, execution period, system utilization, etc. Those system parameters are affected by the underlying hardware and its configuration, which requires huge efforts to find out the proper operation configuration.

We are investigating to support real-time on Xen.

Our approach is to develop a new virtual machine model that achieves the performance isolation in terms of real-time semantics. The model should be able to construct a virtual machine that can use the real-time schedulability analysis model. Moreover, there should be minimum effort to re-engineer real-time guest OS to preserve the real-time property. In order to preserve the behavior of the real-time guest OS over the virtualization layer, we need to timely schedule the virtual machine and control the unintended interruption so that the preemption cannot change the behavioral property of the system.

We are also porting a real-time OS on top of Xen and should be able to report the status and findings in Xen summit.

Dan Magenheimer
Transcendent Memory on Xen

Full Document: http://www.xen.org/files/xensummit_oracle09/xensummit_transmemory.pdf

Our approach, Transcendent Memory, is fundamentally different in that it assumes that working set estimation and prediction will never be sufficiently accurate and, instead, provides a mechanism for mitigating the impact of that inaccuracy. The result is a dramatic reduction in paging and thrashing in many workloads and memory configurations, and thus an increase in VM and system wide performance. At the same time, Transcendent Memory contributes a novel way of multiplexing physical memory and so provides greater flexibility for highly dynamic virtual environments.

Ben Serebrin
Cross-vendor migration: What do you mean my ISA isn't compatible?

Full Document:
http://www.xen.org/files/xensummit_oracle09/xensummit_CrossVendorMigration.pdf

Virtualization enables *Live Migration*, the ability to move a running guest from one physical

machine to another with little or no disruption to the guest or its users. Live migration allows various load-balancing and high-availability techniques to be implemented by the hypervisor and datacenter management software. Unfortunately for live migration, CPU features are added over time and existing guest Oses are not well-equipped to handle CPU changes while the OSes are running. A sysadmin would prefer to view a homogenous data center where every machine can run every guest, but most real datacenters have several generations of machines that have been added over time. The first efforts of live migration can handle CPU changes within a CPU vendor's product line, but further efforts are required to allow guests to migrate between Intel and AMD CPUs.

This presentation focuses on the Instruction Set Architecture (ISA) issues involved in cross-vendor live migration. Other interesting topics, such as handling the migration of active network connections, the migration of emulated devices, and the migration of directly-assigned devices, are addressed elsewhere and are out of this presentation's scope.

Ben Serebrin
Nested Page Tables

Existing CPU hardware allows software to perform all virtualization functions with emulation, binary translation, paravirtualization, or basic hardware virtualization features. However, these software-intensive techniques can be prohibitively non-performant and complicated. Shadow paging, the process of handling the guest's access to x86 page tables, imposes significant software work to properly handle all types of guest access to the paging system. Substantial software complexity is needed to improve a naïve shadow paging implementation's performance to reach an acceptable fraction of native paging performance.

AMD's nested paging hardware eliminates many of the software overheads required in emulation of x86 memory management units. The nested paging architecture adds a second level of page translation tables to the existing x86 architecture's first level. The cost of this second level of translation is an increase the TLB miss latency, and AMD's CPUs implement caching to significantly reduce this latency. Additional hypervisor effort can improve nested paging performance. We present the design of AMD's nested paging caching and some performance measurements, intending to guide hypervisor MMU code toward optimal nested paging performance. We also present an overview of AMD's processor roadmap.

Jose Renato Santos
Achieving 10 Gb/s using Xen Para-virtualized Network Drivers

Full Document: http://www.xen.org/files/xensummit_oracle09/xensummit_networking.pdf

The high processing overheads incurred in Xen Para-virtualized Network Drivers limit overall I/O performance. For example, the number of processing cycles consumed for receiving network packets is 4.7 times higher in Xen than in native Linux. This high cost limits network throughput for Xen guests to only 2.9 Gb/s using a 10 gigabit NIC on a modern server which is able to achieve 9.3 Gb/s when running native Linux.

In a previous Xen summit we described several techniques to reduce the overhead of Xen para-

virtualized (PV) network drivers. In particular we argued that the use of multi-queue NICs, (e.g. Intel VMDq) and the reuse of granted I/O buffer pages over multiple I/O operations could significantly reduce the overheads of Xen PV network drivers. Since then we have added support to multi-queue NICs in Xen netback drivers and evaluated its performance improvements when using a 10 Gigabit Intel NIC with VMDq support. In addition we have designed and implemented a novel grant reuse mechanism based on a software I/O translation table which allows grants to be reused over multiple I/O operations and does not require any communication with the driver domain when the guest needs to revoke an active grant.

The use of multi-queue NICs and grant reuse significantly reduce the overhead of driver domains. However, significant I/O virtualization overheads remain in the guest domains. To address these overheads, we also describe and evaluate additional I/O virtualization optimizations for guest domains.

Yoshio Turner
JustRunIt: Experiment-Based Management with Xen

Full Document: http://www.xen.org/files/xensummit_oracle09/xensummit_justrunit.pdf

Managing data centers is a challenging and often manual endeavor. In particular, many management tasks involve selecting a proper resource allocation or system configuration from many possible alternatives. Evaluating each alternative often requires understanding its performance, availability, and energy consumption implications.

This paper presents JustRunIt,an infrastructure for experiment-based management of data centers. JustRunIt uses Xen virtual machine (VM) cloning and workload duplication to run experiments on a few machines in a large data center to predict the impact of configuration changes. We argue that this approach for systems management is often cheaper and more accurate than previous approaches that rely on analytical modeling. The full paper will present case studies using JustRunIt for two common tasks: server consolidation/expansion and evaluating hardware upgrades. The evaluation shows that JustRunIt can produce results realistically and transparently, and nicely complements automated management systems.

Eddie Dong
Status of SR-IOV

SR-IOV capable network devices offer the benefits of direct I/O throughput and reduced CPU utilization while greatly increasing the scalability and sharing capabilities of the device. Since disclosure of Xen/SR-IOV architecture in last summer Xen summit, We have enabled SR-IOV support in Xen for network, and keep working with community to push the solution to both Linux and Xen upstream. In the mean time, we have tuned and improved the performance, scalability as well as quality of service for network.

In this talk, we will report the latest Xen SR-IOV status, along with challenges of performance and scalability. We will also present our ABI solution to push SR-IOV ahead, as well as some preliminary performance & scalability data if possible.

Diwaker Gupta
Difference Engine

By multiplexing hardware resources among virtual machines (VMs) running commodity operating systems, VMMs decrease both the capital outlay and management overhead of hosting centers. Appropriate placement and migration policies can take advantage of statistical multiplexing to effectively utilize available processors. However, main memory is not amenable to such multiplexing and is often the primary bottleneck in achieving higher degrees of consolidation.

Previous efforts have shown that content-based page sharing provides modest decreases in the memory footprint of VMs running similar operating systems and applications. Our studies show that significant additional gains can be had by lever- aging both sub-page level sharing (through page patching) and in-core memory compression.

In this talk I'll present Difference Engine, an extension to the Xen virtual machine monitor, to support each of these -- in addition to standard copy-on-write full page sharing -- and demonstrate substantial savings not only between VMs running similar applications and operating systems (up to 90%), but even across VMs running disparate workloads (up to 65%). In head-to-head memory-savings comparisons, Difference Engine outperforms VMware ESX server by a factor of 1.5 for homogeneous workloads and by a factor of 1.6-2.5 for heterogeneous workloads. In all cases, the performance overhead of Difference Engine is less than 7%.

Allen Kay
VT-D Development in Xen

The purpose of this talk is to provide a status update on the VT-d development in Xen. It will give an overview of existing capabilities in upstream Xen tree. Clearing up common confusions over VT-d features such as interrupt remapping in many people's minds. Provide the audience an overview on how to assign a device and an overview on what devices can be passed through to the guest, what devices cannot be passed through to the guest, and what devices must be passed through as a group. I will also introduce new features such as ATS for supporting device translation cache and Remapping Hardware Status Affinity for efficiently supporting DMA remapping in NUMA platforms.

Patrick Colp
VM Snapshotting

Full Document: http://www.xen.org/files/xensummit_oracle09/VMSnapshot.pdf

There has been an emergence of dependability- and security-related applications which take advantage of the isolation provided by VMMs. These applications can verify properties about a running VM from an external context, segregated from malicious code. Some potential uses include dynamic introspection of a system, execution rollback, and VM fork.

A variety of useful introspectiong applications exist. Using simple introspection, it is possible for

a protected VM to execute things like virus scanning on an unprotected VM. This allows the virus scanner to exist outside of the realm of malicious code (i.e. the virus scanner will no longer be the subject of virus attacks). Dynamic attestation can be carried out by continuously performing sanity checks on OS data structures, such as task lists, scheduler queues, and page tables.

Without the ability to snapshot a VM, however, these tools must be able to cope with constantly changing memory contents, aggravating the process. For example, a clever virus may move itself around in memory so as to avoid detection by an external virus scanner.

A more complex use of snapshotting is that of execution rollback. This can be used to undo any changes that have been made to an executing VM. This provides the ability to perform whole-machine undos, allowing uses and system administrators to experiment with different system configurations, updates, etc. without jeopardizing the integrity of the system.

Execution rollback can also be implemented to improve the crash-resilience of a system. When a machine crashes, it can be rolled back to its last good checkpoint and the event which caused the crash, for example accepting a malicious packet from the network, can be avoided (i.e. the packet dropped). Thus, the availability and dependability of the machine is improved.

Perhaps one of the most complex applications of this snapshotting ability is that of VM fork, allowing a snapshotted VM to execute independently. While this facility has a variety of uses, the most predominantly described is that of "Gold Mastering." That is, a VM template is loaded into memory as a static snapshot, which is then used to quickly instantiate lightweight child VMs that share a common base memory image.

The mechanism we propose to enable VM snapshotting is implemented by performing Copy-on-Write of a VM's memory. This avoids the need to do a complete memory copy, saving both time and resources.

Unfortunately, while these applications are useful, they must over-come the challenge of examining a system which is constantly changing. To address this problem, we introduce a VM snapshotting ability for Xen. This snapshotting mechanism captures a consistent view of a VM's state at a point in time, while the VM continues to run.

Andrew Warfield
Tralfamadore: Online and Iterative Execution Analysis

Software developers spend considerable time attempting to understand how the systems that they maintain actually behave in the field. This talk will describe how the capture of high-fidelity instruction traces of production VM executions can be analysed in an iterative and online manner and allow dynamic analysis techniques to be more powerfully brought to bear on program understanding, analysis, and debugging.

John Byrne

Detecting and Correcting Transient Hardware Errors via Xen Extensions

There has been recent interest in providing Fault Tolerance in virtualized environments [Reus, Kemari, Vmware] using incremental checkpointing or lock-stepping. However, there are two significant limitations to all efforts to date --- detecting and correcting transient hardware errors. The International Technology Roadmap for Semiconductors has predicted significant reliability problems for future systems, increasing at a pace that has not been seen in the past. Smaller feature sizes, increased component density, and the reduced critical charge required to flip a bit are all expected to lead to an increased occurrence of *both* permanent hard errors (e.g., wear-out errors from dielectric breakdown, negative bias instability, electro-migration, etc) and transient soft errors (e.g., from alpha particles or electromagnetic noise). Indeed, a study from Intel estimates a 100-fold increase in transient faults when scaling from 180nm to 16nm. Such errors can affect system uptime and data integrity -- both key requirements for enterprise workloads.

The COVERT Xen-based implementation lock-steps VMs and compares output (network and disk) to detect transient hardware errors. Initial performance results using a variety of benchmarks indicate the prototype performance is quite good (< 10% degradation) and there are a variety of ways we can improve it further. To correct for transient errors we are adapting some of the checkpointing work done on Xen [Remus, Kemari]. Combining the checkpoints with very detailed event logging will allow us to reliably replay the execution of both VMs (perhaps on a different core) to correct for transient hardware errors. The extended abstract will detail the design of the existing prototype and the checkpoint extensions as well as present performance data on the current lock-step implementation.