2008 – 2009 Xen.org Roadmap Document

November 2008

Stephen Spector
Ian Pratt
Xen Community

Table of Contents

Executive Overview

This document contains the most recent roadmap for the open source Xen hypervisor built by the Xen.org community. The latest available release of Xen 3.3 is currently available at http://www.xen.org/download/ and is not presented in this document. Instead, the focus will be on the next several releases of the hypervisor with some items being close to the next release and other items planned for a later release date. For more information on the community, please visit http://www.xen.org or follow the community blog at http://blog.xen.org.

Xen Project Methodology

The Xen hypervisor project is open source with all releases including the most recent available from a Mercurial Repository at http://xenbits.xensource.com/ or via a source code browser at http://lxr.xensource.com/lxr/source for the latest unstable release in active development. All code submissions must be made as patches which are sent to the xen-devel@lists.xensource.com mailing list for consideration.

Mailing list registration is at http://lists.xensource.com/ and support on the Mercurial Repository is in the Xen Wiki at http://www.xen.org/files/hg-cheatsheet.txt.

Releases

Rather than the `big bang' 12 month development cycle used in Xen 1.0, 2.0 and 3.0, we want to make the development toward the next major release far more incremental. We aim to continue the current practice of going through a stabilization phase and having point releases every 10-12 weeks. Just as with Linux, the point release will be maintained with bug fixes until the next point release.

One side effect of this approach is that at the time of a release not all features that are new in the code base may necessarily be stable in time for the release. We won't hold the release, but will simply disable such features or document the issues. The aim is obviously to avoid regressions in functionality that worked in earlier releases.

Performance and Scalability

As to be expected in any product release, the Xen.org community will ensure that newer releases are of higher quality in terms of performance and scalability. It is crucial that all new features added into the hypervisor deliver on these metrics to ensure continued industry leadership in the hypervisor market. Having these two metrics as a guideline also promotes innovation within the community as developers find new algorithms and solutions to new and existing problems.

There are several performance monitoring tools available in the community to support these efforts: xen-oprofile, xen software performance counters, and xentrace. Xen-oprofile is a sample based profile system useful for looking at system-wide CPU consumption. The xen software performance counters are useful for tracking event occurrences in Xen (and hence spotting anomalously high counts of supposedly rare events), and can also be used to collect various histograms of scheduler and memory management data. There is also xentrace which can be enabled to collect timestamped trace records into a buffer to enable detailed event time-lines to be collected.

Smart Scheduler for Hyperthreading

(open)

Testing and Debugging

Users require Xen to be a rock-solid stable system component, achieving greater stability than the OSes which run on it. Generally, we don't do too badly on this front. We're fortunate to have a relatively small and tight code base, coupled with significant investment in testing by a number of parties.

The tip of the -unstable tree is exercised daily by IBM, Intel and Virtual Iron as well as Citrix. All test results are sent regularly to the xen-devel mailing list. The continuous testing provides a good way of monitoring the progress toward stability of new features, and for flagging regressions.

In fact, most potential regressions never make it out into the -unstable tree, as they are picked up by automated testing in the staging tree, which gives change-sets a grilling on three different machines (32b, PAE, and x86 64) before pushing the change-sets out to -unstable. Hard-core developers wanting to see check-ins as soon as they happen can do so by subscribing to xen-changelog@lists.xensource.com.

Prior to the 3.0 release, Xen.org released a special testing CD and encouraged users to download and run it on their hardware then upload the results. The CD provided booted a native Linux kernel followed by 32b, PAE and x86 64 kernels (as appropriate for the hardware), running a whole series of tests and benchmarks under each and recording the results to a log file which was then uploaded. The test CD proved very useful identifying machines Xen struggled on, and helped get many of the issues fixed prior to release. Post release, the 3.0 demo CD has proved useful in debugging various platform issues reported by users, enabling them to collect and supply developers with data from both native and Xen kernel boots. The last demo CD released was for version 3.03 at http://www.xen.org/download/index_3.0.3.html.

Even with the best possible test and QA program, some number of in-field crashes will be sadly inevitable. In these circumstances, it's important that developers can gather as much information as possible from the incident. For user-space incidents, `xen-bugtool' is useful for collecting log files and system details. The xen-bugtool command line application will collate the Xen dmesg output, details of the hardware configuration of your machine, information about the build of Xen that you are using, plus, if you allow it, various logs. For more serious failures of domain 0 or xen itself we can't rely on things getting logged in standard files, and need support for taking a system core dump. An effort led by Simon Horman and Magnus Damm from VA Linux ported the kexec and kdump utilties to the Xen environment for system core dumps; more information at http://www.xen.org/files/summit_3/kexec_kdump.pdf.

For xen developers there are a number of debugging aids that can called on. There is a serial gdb stub that can be connected to remotely and used to examine both xen and domain 0. Since this halts the system while the debugger is connected, it typically can't be used on production systems. One frequently useful tool is the debug console, which by default is accessed by hitting ctrl-A three times on the serial console. Hitting `h' gives a help menu of features the console can perform. It's main use is for diagnosing the state of the system in event of a hang. Register dumps can be used to see what all the CPUs are doing, whether guests are servicing interrupts etc. The console also provides access to the software performance counters and other statistics.

Debugging guest VMs can be achieved using Xen's gdbserver support. When gdbserver is started in dom0 against a particular VM, it effectively implements gdb serial stub functionality on behalf of the guest, enabling gdb to connect to the gdbserver via a TCP port. gdbserver has recently been extended to support both PV and HVM guests. If getting register and stack state for all the VCPUs from a guest is all that's required, the `xenctx' command is quicker than firing up gdb.

Mukesh Rathor from Oracle has recently introduced updated debugging tools for the Xen hypervisor and guests at Xen Summit June 2008, http://www.xen.org/files/xensummitboston08/Mukesh%20xendbg-present.pdf. The gdbsx tool allows for source code debugging of guests and the kdb tool can provide low level debugging for the Xen hypervisor and Domain 0.

All bugs found in Xen by users or developers should be submitted to the Xen.org Bugzilla system at http://bugzilla.xensource.com/bugzilla/index.cgi. A community mailing list for tracking all Bugzilla updates, xen-bugs, can be subscribed to at http://lists.xensource.com/mailman/listinfo/xen-bugs. All reported bugs are also posted weekly in the Xen.org Wiki at http://wiki.xensource.com/xenwiki/XenBugs for developers looking to work on open issues.

Xen in Linux Mainstream

The effort currently underway to bring Xen Domain 0 support directly into the Linux kernel is referred to as paravirt ops (pv-ops for short). It has been developed as an API allowing the Xen hypervisor to override operations normally run on the Linux OS at the API level. It allows a single kernel binary to run on all supported execution environments including the native machine. The operations in paravirt ops are divided into three categories:

- Simple Indirect Call – high level functionality where overhead of indirect call is not important
- Indirect Call – low level functionality that are called often and are performance critical
- Macros – A set of assembly code macros needing paravirtualization because they include sensitive instructions or performance critical code paths

<text on standard Xen port from Jeremy Fitz.>

Recent information on the IA 64 port of Xen and paravirt ops was presented at Xen Summit June 2008 by Isaku Yamahata from VA Linux; http://www.xen.org/files/xensummitboston08/paravirt_ops_ia64-for-pdf.pdf for presentation and abstract at http://www.valinux.co.jp/documents/tech/presentlib/2008/xensummit_boston/Paravirt_Ops_on_Linux_IA64_Abstract.pdf.

OS Support

The following OS kernels have been ported to the latest version of Xen 3.x:

- Linux – 2.6.18
- Solaris 10
- Open Solaris - http://opensolaris.org/os/community/xen/
- FreeBSD 6.3 - http://wiki.freebsd.org/FreeBSD/Xen (There is currently support for 6.3, RELENG_6 (6.x), RELENG_7(7.x), and HEAD (development branch) SMP + PAE. 64-bit

support will be coming in the near future. There is no near-term plans for dom0 support.

- NetBSD 4.0 - http://www.netbsd.org/ports/xen/howto.html
- Open BSD – Late 2009 release being planned based on update to FreeBSD
- Plan9 - http://netlib.bell-labs.com/wiki/plan9/Installing_in_Xen_3.0/index.html

(OS) Vendor Support

The following OS vendors support a Xen hypervisor as part of their operating system distribution:

- Oracle - http://www.oracle.com/technology/products/vm/index.html
- openSuSE - http://en.opensuse.org/Xen
- SUSE Linux Enterprise Server 10 - http://www.novell.com/products/server/virtualization.html
- Sun xVM
- Red Hat Enterprise Linux 5
- Gentoo
- Fedora 8 - http://fedoraproject.org/wiki/Tools/Xen
- Centos
- Debian - http://wiki.debian.org/Xen
- Ubuntu - https://help.ubuntu.com/community/Xen

Xen Client Initiative

As server virtualization continues to provide enterprise customers with increased security, performance improvements, reduced costs, and other benefits, many companies are also looking at virtualizing the desktop CPU to see similar benefits. This type of virtualization requires a more sophisticated solution than just hosting desktops on servers with remote presentation and centralized management. By isolating desktop CPU time with virtual machines, companies can add more workloads on a single physical machine and take advantage of the processing power generally not used on a desktop CPU.

The Xen Client Initiative is a newly launched project enabling the Xen hypervisor to run on any type of desktop or mobile device. The project is manged through a series of working groups with information on all activities on the Xen.org Wiki at http://wiki.xensource.com/xenwiki/General_Project_Information.

Device Pass-Through

The concept of “pass-through” involves the translation of physical devices (e.g. memory mapping, IO ports, etc) to virtual devices. This methodology is critical for virtualization solutions and is especially important as moving Xen to client devices requires a new set of pass-through algorithms to be written for devices such as USB ports, wireless cards, and graphics cards.

Wireless Pass-Through

Wireless LAN devices are currently passed through to a single domain. This domain has full ownership

of the WLAN device and can use native OS drivers and software. This includes management and configuration the WLAN hardware with native applications. This provides a user experience that is identical to WLAN use in a non-virtual environment. Additionally, the wireless connection can be shared with other domain on the Xen Client host using native OS applications. One example of this is passing a WLAN card to a Windows Vista primary domain. This primary domain can be connected to other running domains through other emulated or para-virtual network interfaces. In this example, the primary domain can use Windows Internet Connection Sharing to allow the other connected domains to share the wireless connection.

The current Xen Client solution for wireless WLAN pass-through presents some security issues in cases where the domain that owns the WLAN device device is untrusted or insecure (i.e. this breaks the domain isolation model that many of Xen's strengths are built on). This is due to the fact that traffic from secure domains may be bridged through to the insecure WLAN owner domain. Future hardware level device virtualization support may be leveraged by Xen Client to allow sharing of WLAN devices between domains directly. This would preserve domain isolation and address the security issues mentioned above. This would also allow all domains to interact with wireless hardware with native OS drivers and software.

Graphics Card Pass-Through

One of the biggest bottlenecks of virtualization is the display rendering.

Because you use a emulated graphic card you loose all the 3D capability of your graphic card. The idea of graphic pass-through is to affect a native graphic card to a virtual machine, it can be an integrated or a legacy pci-express graphic card. In the scenario you will be able to install the native driver and experience a full graphic support inside your virtual machine.

USB Pass-Through

Noboru Iwamatsu from Fujitsu is leading an effort to pass-through a USB device for a Xen hypervisor running on a client device. The latest presentation on this work is at http://wiki.xensource.com/xenwiki/Working_Group_Core_Hypervisor?action=AttachFile&do=get&target=USBDesign.pdf.

Power Management

Xen 3.3 incorporates full support to processor power management features: C-states and P-states. When immediate power saving is brought to users, current implementation is still not well-tuned for diverse demands, such as performance-per-watt. Works on-going from Intel OTC team are now focusing on improving Xen to a more power efficient model, on both client and server platform. Micro-accounting information is being analyzed to reduce break events for longer C-states residency at idle, and workloads are being measured for both avoiding unexpected performance drop and improving performance-per-watt. Different guests will be checked to understand their power characteristics. Some ideas are already in devleopment. Range timer, which aligns timer expirations to reduce timer interrupts, has been accepted in xen unstable. Power aware scheduler, taking power factor into consideration (like package C-state dependency), provides flexibility to balance user performance/power requirement. We believe more will show up along with the tuning process.

Another important work in this area is to provide more user flexibility. Xen itself can't understand everything on-going and no algorithm can be optimal for all requirements. Flexibility has to be provided to management tools and adminstrators. Tools are in development to assist users/developers. Xenpm, a simple tool to acquire Xen power management information, has been in xen-3.3-testing and xen-unstable tree. Xentrace is also extended to log Xen power management activities. We will update user/developer manuals for available power features. More controll knobs will be exposed. For example, more cpufreq governors (userspace, powersave, performance, etc.) can be added. Also 'Soft' cpu offline may be requested to stop one physical cpu by offloading all its workloads and then putting it into deepest C-state. With more options/knobs, it's then necessary to support power profiles to simplify configuration. Finally, community is expected to have more tryouts on Xen power management which would be very important to quality and evolvement in this area.

Schedulers

(Don Dugger at Intel)

Battery Management

Mobile processors, one of the main target platforms for the Xen Client Initiative, require the guest operating system to have access to power management information such as the battery status. In an attempt to provide bare metal like power management experience, we has already laid the ground work by providing more aggressive Cx, Px support in Xen/Dom0, exposing battery information to guest etc.

The virtual ACPI layer now exposes battery objects using control method battery interface as outlined in the ACPI specification. To display battery information through its standard user interface (e.g., Control Panel->Power Options->Power Meter tab on XP), the guest OSPM software queries this vACPI CMBattery interface.

Upon receiving battery information request from the guest OSPM software, our vACPI layer either delegates the request to QEMU or directly reads/writes to the appropriate battery port to provide battery information to the guest. The direct port read/write approach, which is more commonly know as pass-through battery management approach is efficient but limited in scope as it is highly platform dependent. Where as, delegation to QEMU approach, also known as non pass-through battery management approach is more versatile but has a bit more dependency on dom0 than the pass-through approach. To use either of the approaches and provide battery management support within a guest OS, all that need to be done is to set a configuration file option!

Security

A project underway within the Naval Research Lab's Center for High Assurance Computer Systems (http://chacs.nrl.navy.mil/), Project Xenon, is looking directly at the source code to measure the assurance level and create a unique, highly secure version of the Xen hypervisor. The results of this effort are fed directly back into the Xen development community with important data on how the source code can be improved to create a more secure solution. Information on this project can be found in the initial presentation at Xen Summit April 2007 (http://www.xen.org/files/xensummit_4/XenSummitSpring07_McDermott.pdf) and the follow-up

presentation at Xen Summit June 2008 (http://www.xen.org/files/xensummitboston08/XenSummitSpring08.pdf) by John McDermott.

Memory Management

Valuable resources in physical servers are often underutilized and virtualization can be employed to consolidate multiple physical servers so that resources are more fully utilized. If a VMM creates the illusion that there are more instances (or bandwidth) of a resource than actually physically available, that resource is considered to be overcommitted (or oversubscribed). For example, if three uniprocessor servers are virtualized (as guests) and consolidated onto one dual-processor server, the CPU resource is considered overcommitted. If all three guests each need to run workloads that always require an entire CPU, these guests would be considered poor candidates for virtualization. But, virtualization works most of the time because most resources on many physical servers are badly underutilized.

One key resource however is a notable exception: physical memory. Unlike CPUs and I/O cards, an operating system on a physical system generally utilizes its physical memory fully. Pages of memory that are not used for kernel/application code or data are used to cache disk data. Since disks are extremely slow and it is common for the same disk data to be used again and again, a well-managed cache of disk pages can dramatically improve performance of many workloads. Consequently, when physical servers are consolidated, it is considered unwise to overcommit physical memory. Indeed, Xen does not support memory overcommitment at all. For example, if one wishes to consolidate multiple 1GB guests on a 4GB Xen server, only three guests can be accommodated.

Dan Megenheimer from Oracle is leading an important Xen project to look at various algorithms to take advantage of overcommiting memory without causing significant impact on system performance. Recent work done with Ballooning drivers and Xen is leading to new possibilities for introducing memory overcommitment to Xen. The most recent presentation on this topic is available at http://www.xen.org/files/xensummitboston08/MemoryOvercommit-XenSummit2008.pdf and a detailed abstract can be found at http://wiki.xensource.com/xenwiki/Open_Topics_For_Discussion?action=AttachFile&do=get&target=Memory+Overcommit.pdf.

Disk Management

Live Resizing of DomU Volumes

Currently in Xen it is not possible to resize domU disks on the fly. If you resize domU disk in dom0 on the fly (for example LVM volume), domU still won't see the new (bigger) disk size before domU is shut down and restarted.

Resizing domU (XVD/VBD) disks cannot be done without downtime. This is how things are at the moment."

This feature request is about having a way to notify domU (on the fly) about disk size change, and the domU kernel to pick up the new size on the fly.

Current upstream vanilla/kernel.org Linux kernels have online SCSI disk resize support implemented.. so the Linux kernel side of this feature is already implemented.

Also upcoming RHEL 5.3 will have SCSI disk online resize support. If NOT talking about virtualization, usually this online resize feature is handy with SAN storage; grow LUN on the SAN and get the Linux server to see the new LUN size on the fly.

When talking about server virtualization; guest online disk resize is a very welcome feature, to be able to grow and scale virtual servers without downtime.

So the conclusion is: Xen support for 'on the fly disk (XVD/VBD) resizing' is missing.

Components that need to have support for this implemented:

- Xen hypervisor (some way of forwarding the 'disk-resize' event to guest vm)
- Xen dom0 tools (some way to send the 'disk-resize' event to guest vm)
- Xen domU kernel (ability to receive and handle the 'disk-resize' event)
- Xen PV drivers (in HVM guests; ability to receive and handle the 'disk-resize' event)

Links of interest:

http://thread.gmane.org/gmane.linux.scsi/44335 & http://lists.xensource.com/archives/html/xen-devel/2008-09/msg00158.html & https://bugzilla.redhat.com/show_bug.cgi?id=455692

Networking

Smart IO Devices

A proposed Xen project, Xen Lite, is presented later in this document. Leveraging smart IO devices, the Xen hypervisor can be reduced in size and scope by eliminating the software required for virtualizing previously “non-smart” IO devices.

Fault Tolerance

Andrew Warfield and his team at the University of British Columbia are currently working on Project Remus, a high availability software solution for fault tolerance via asynchronous virtual machine replication. Capitalizing on the ability of virtualization to migrate running virtual machines between physical hosts, Remus replicates snapshots of an entire running OS instance at very high frequencies between a pair of physical machines. Using this technique, Remus converts the execution of a VM into a series of replicated snapshots. External output, specifically transmitted network packets, are not released until the system state that created the external output has been replicated on the secondary machine.

An introductory paper on this topic is available at http://www.cs.ubc.ca/~andy/papers/remus-nsdi-final.pdf.

Another project within the community to solve fault tolerance issues is Project Kemari (http://www.osrg.net/kemari/) being led by Yoshi Tamura from NTT Cyber Space Labs. The project's focus is to deliver high availability with no special hardware or application modifications. Project Kemari takes two common fault tolerance solution approaches and combines them to take advantage of the benefits of both solutions:

- Lock-stepping - all external events are logged on the primary VM and replayed on the secondary VM
- Continuous checkpointing – the state of the primary VM is frequently transferred to the secondary VM

Kemari transfers the state of the primary VM to the secondary VM when an event occurs. This approach enables the primary VM and the secondary VM to be kept in sync with less complexity compared to lock-stepping. Furthermore, Kemari does not require external buffering mechanisms which impose output latencies, which is necessary in continuous checkpointing. Kemari synchronizes the VMs when the primary VM is about to send an event to devices such as storage and networks.

Yoshi's presentation from Xen Summit June 2008 is available at http://www.xen.org/files/xensummitboston08/tamura_xen_summit_presentation_final.pdf with a detailed abstract at http://wiki.xensource.com/xenwiki/Open_Topics_For_Discussion?action=AttachFile&do=get&target=Kemari_08.pdf.

Xen Introspection Project

The purpose of the Xen Introspection Project is to develop a comprehensive set of APIs allowing developers and solution vendors to have direct access to a variety of monitored features for a specific running virtual machine on a Xen hypervisor. The API set will expose the following types of information for a specific virtual machine:

- Memory and CPU
- Storage
- Networking
- Application Execution Debugging - step control of individual processes or threads

These APIs will enable the following types of solutions:

- Security
- Forensics
- Debugging
- System Management

Work is already underway at several universities to enable some of these features.

Georgia Tech has the XenAccess Library Project; a development project that allows for the introspection of memory and disk access for a specific virtual machine. More information on this project is available at http://groups.google.com/group/xenaccess.

The University of Alaska Fairbanks and the University of California at Davis are working on the VIX (Virtual Introspection for Xen) tool suite. An introductory paper on this research is available at http://www2.computer.org/portal/web/csdl/doi/10.1109/MSP.2008.134 or http://portal.acm.org/citation.cfm?id=1368517.

The Xen Introspection Project is a community effort within Xen.org to leverage the existing research presented above with other work not yet public to create a standard API specification and methodology for virtual machine introspection. As there are a variety of topics within this project, the Xen.org community is looking for support from graduate students at various universities looking for interesting

and useful thesis projects.

Xen API

A set of programming interfaces to control, manage, and monitor a Xen hypervisor or collection of hypervisors is referred to as the Xen API. The latest Xen API description is available at http://wiki.xensource.com/xenwiki/XenApi?action=AttachFile&do=get&target=xenapi-1.0.6.pdf. C and Python bindings for the existing Xen API are available in the main Xen tree at *tools/libxen* for C bindings and *tools/python/xen/xm/XenAPI.py* for Python bindings.

An active project is currently in process within the Xen.org community; Xen API Project (http://wiki.xensource.com/xenwiki/XenApi). The goal of this project is to align the Xen API functionality with the existing Citrix XenServer APIs found at http://community.citrix.com/cdn/xs/sdks. A mailing list xen-api@lists.xensource.com is available for anyone interested in working on the current Xen API.

Several projects within the community are in process for managing a Xen hypervisor environment:

- Xengine Project - http://www.negativeblue.com/xengine/
- Zentific Project - http://www.zentific.com/screenshots.php
- Project Ganeti - http://code.google.com/p/ganeti/

Xen Lite

Xen Lite is a new proposal from Ian Pratt for community consideration and implementation. As newer servers contain more CPUs and smart IO devices become more common, Xen can become even "thinner" as it will no longer require virtualization of many devices that are necessary today. These smart IO devices are presented in more detail in the paper, "Scalable I/O Virtualization via Self-Virtualizing Devices", available at http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.60.1057.

Abstract

The virtualization of I/O devices is an integral part of system virtualization. This includes both virtualizing the physical devices and managing them across multiple guest virtual machines (VMs) or domains running on top of a virtual machine monitor (VMM) or hypervisor (HV). This paper presents the notion of self-virtualizing devices, where for higher end, 'smart ' I/O devices, selected virtualization functionality is offloaded onto the devices themselves. In particular, a self-virtualizing device is aware of being used in a virtualized environment and implements the virtual device abstraction on the device itself. It also presents an interface to the HV to manage these virtual devices. Using this virtual device abstraction, a guest domain can interact with the physical device with minimal HV involvement. The outcomes are reduced I/O virtualization costs caused by the HV (and additional software components required to virtualize the device) and improved scalability for device interactions with guest domains. The prototype self-virtualizing device described in this paper leverages the multi-core nature of modern computer architectures. It uses the IXP2400 network processor with multiple internal specialized communication cores to implement a self-virtualizing network interface attached to a x86 based host machine. The device is used in conjunction with the Xen hypervisor and provides virtual network interfaces.

Open Virtual Format

The Xen.org community fully supports the Open Virtual Format (OVF) project being led by IBM with the latest information at http://xml.coverpages.org/ni2007-09-11-a.html. Mike Day from IBM presented the latest update on this open source project at Xen Summit Boston; http://www.xen.org/files/xensummitboston08/open-ovf-proposal.pdf.

The OVF specification describes an open, secure, portable, efficient and extensible format for the packaging and distribution of (collections of) virtual machines. Its goal is to facilitate the automated, secure management not only of virtual machines, but the appliance as a functional unit.

Most importantly, according to the DMTF announcement: "OVF specifies procedures and technologies to permit integrity checking of the virtual machines (VM) to ensure that they have not been modified since the package was produced. This enhances the security of the format and will alleviate security concerns of users who adopt virtual appliances produced by third parties. OVF also provides mechanisms that support license checking for the enclosed VMs, addressing a key concern of both independent software vendors (ISVs) and customers. Finally, OVF allows an installed VM to acquire information about its host virtualization platform and run-time environment, which allows the VM to localize the applications it contains and optimize its performance for the particular virtualization environment."

The proposed OVF uses existing packaging tools to combine one or more virtual machines together with a standards-based XML wrapper, giving the virtualization platform a portable package containing all required installation and configuration parameters for the virtual machines. This allows any virtualization platform that implements the standard to correctly install and run the virtual machines.

As described in the specification Introduction, OVF supports content verification and integrity checking based on industry standard public key infrastructure, and provides a basic scheme for management of software licensing. It supports validation of the entire package and each virtual machine or meta-data component of the OVF during the installation phases of the VM lifecycle management process. It also packages with the appliance relevant user-readable descriptive information that can be use by a virtualization platform to streamline the installation experience. OVF supports both standard single VM packages, and packages containing complex, multi-tier services consisting of multiple interdependent VMs. While OVF is virtualization platform neutral, it also enables platform-specific enhancements to be captured: it supports the full range of virtual hard disk formats used for VMs today, and is extensible to deal with future formats that may arise. OVF also permits the encoding of vendor specific meta-data to support specific vertical markets; it s supports user visible descriptions in multiple locales, and supports localization of the interactive processes during installation of an appliance, allowing a single packaged appliance to serve multiple market opportunities.

Xen Ports

The main Xen hypervisor is built for the Intel / AMD 32-bit x86 platform and is the primary focus of the community; however, there are two additional ports supporting the ARM processor and 64-bit Itanium processor.

IA 64

The IA64 port now has functional parity with x86 covering both PV and HVM domain, most of PV drivers, save/restore/live relocation for both PV and HVM domain, xenoprof and kexec/kdump. And its performance is excellent and usable for enterprise use.

There is also efforts to merge modification into the Linux upstream by using paravirt_ops/ia64.

Only rather new features aren't supported yet. Like stub domain, MSI support.

xen/ia64 has its own functionality. The most noticeable one is SIO (Self IO emulation) which Trinstan Gingold presented at Xen Summit June 2008 http://www.xen.org/files/xensummitboston08/xen-sioemu.pdf

In terms of hardware compatibility, it has been extensively tested to operate on the following configurations:

- Intel Tiger4
- HP Integrity series systems
    - Some of the cellular (NUMA) systems (sx1000/sx2000 based system) might not support the full range of memory or cell local memory configurations.
- Bull NovaScale 4000 and 6000 systems, with 5000 to follow
- Fujitsu Primequest series

In terms of future work, the main areas are:

- Xen domain 0 support into the Linux kernel upstream using paravirt_ops/ia64.
    - implementing rather new functionalities like stub domain, power management, VT-d, MSI, MSI-X
- SIO(Self IO emulation)
- more stability/performance/scalability testing and improvements

An email mailing list on this project is available at Xen-ia64-devel@lists.xensource.com.

ARM

<Info from Sang-bum>

The latest information on the port is available in the Xen.org Wiki at http://wiki.xensource.com/xenwiki/XenARM. An email mailing list on this project is also available at xen-arm@lists.xensource.com.