XEN Virtualisation Technology

Roger Whittaker

Novell Technical Services Bracknell

rwhittaker@novell.com

November 14, 2006

AMD

Novell

Motivation and Background

Why Virtualisation

Efficient use of resources

- most hardware is drastically underused

Consolidate multiple legacy systems

Flexibility: move systems from one physical server to another

Capacity planning: hardware becomes a fluid resource

Simplifies the application stack: largely removing hardware and drivers from the equation

Virtualisation background

Hypervisor-based virtualisation: Mainframe

Hardware emulation:

- VMWare
- MS Virtual Server

Other ways of running “multiple systems”:

- User Mode Linux (UML)
- OpenVZ, Virtuozzo
- Linux Virtual Server
- BSD jails
- Solaris zones, containers
- Wine, win4lin, dosemu, others

Virtualisation background

- On x86-based architecture
  - VMWare
    - Workstation (running on host OS)
    - GSX server (running on host OS)
    - ESX server (host OS hidden from user)
  - Bochs (other architectures also)
  - Qemu (other architectures also)
  - MS Virtual PC
  - MS Virtual Server
    - Now permits non-MS guests

Virtualisation present and future

Hypervisor approach

- Most efficient compromise
- Allowing access to hardware through thin hypervisor layer

- With “legacy” hardware requires modified guest OS
  (“paravirtualisation”)

- With “new generation” hardware allows unmodified guest OS
  (“full virtualisation”, “hardware assisted virtualisation”)

The Xen hypervisor

Spinoff from research at University of Cambridge

- Project leader: Ian Pratt
- `http://www.xensource.com/`
- Now at version 3.0.x
- Now going mainstream in Linux vendor offerings
  - SUSE more enthusiastic than Red Hat at present
- Xensource and others offering commercial tools and add-ons

The Xen hypervisor

- xen is a minimal kernel

```
/boot/grub/menu.lst

title Xen
   kernel xen.gz
   module vmlinuz-xen
   module initrd-xen
```

The Xen hypervisor

Xen essentially defines a (slightly) different architecture:

- x86-xen or x86_64-xen

The kernel is written to that architecture

- modifications are not major
- with modifications kernel runs as both “host” (dom0) and “guest” (domU)

Terminology

- prefer dom0, domU as technically dom0 is also a “guest” of Xen.

The Xen hypervisor

- Running `xen.gz` alone gives us no access
  - Need to run a xen-enabled kernel on top
- Operating systems running on Xen are “domains”
  - Initial OS installation “host system” is dom0
  - Further installations “guests” are known as domUs (unprivileged domains)
  - a domU typically uses dom0 for access to hardware via Xen
  - xen-tools package in dom0 for managing domUs

Paravirtualisation

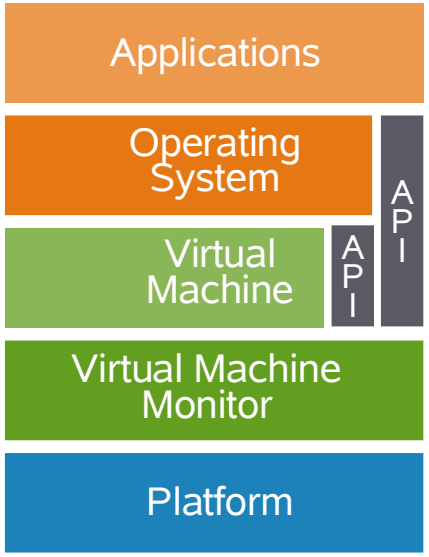
Paravirtualization

A technique in which the VMM is supplemented by an API that provides an assist for certain situations

Requires hardware dependent portions of the guest OS to be modified to become aware of the virtualisation layer

Allows you to avoid hard-to-virtualise processor instructions by replacing them with a procedure call that provides that functionality

Higher performance than full virtualisation



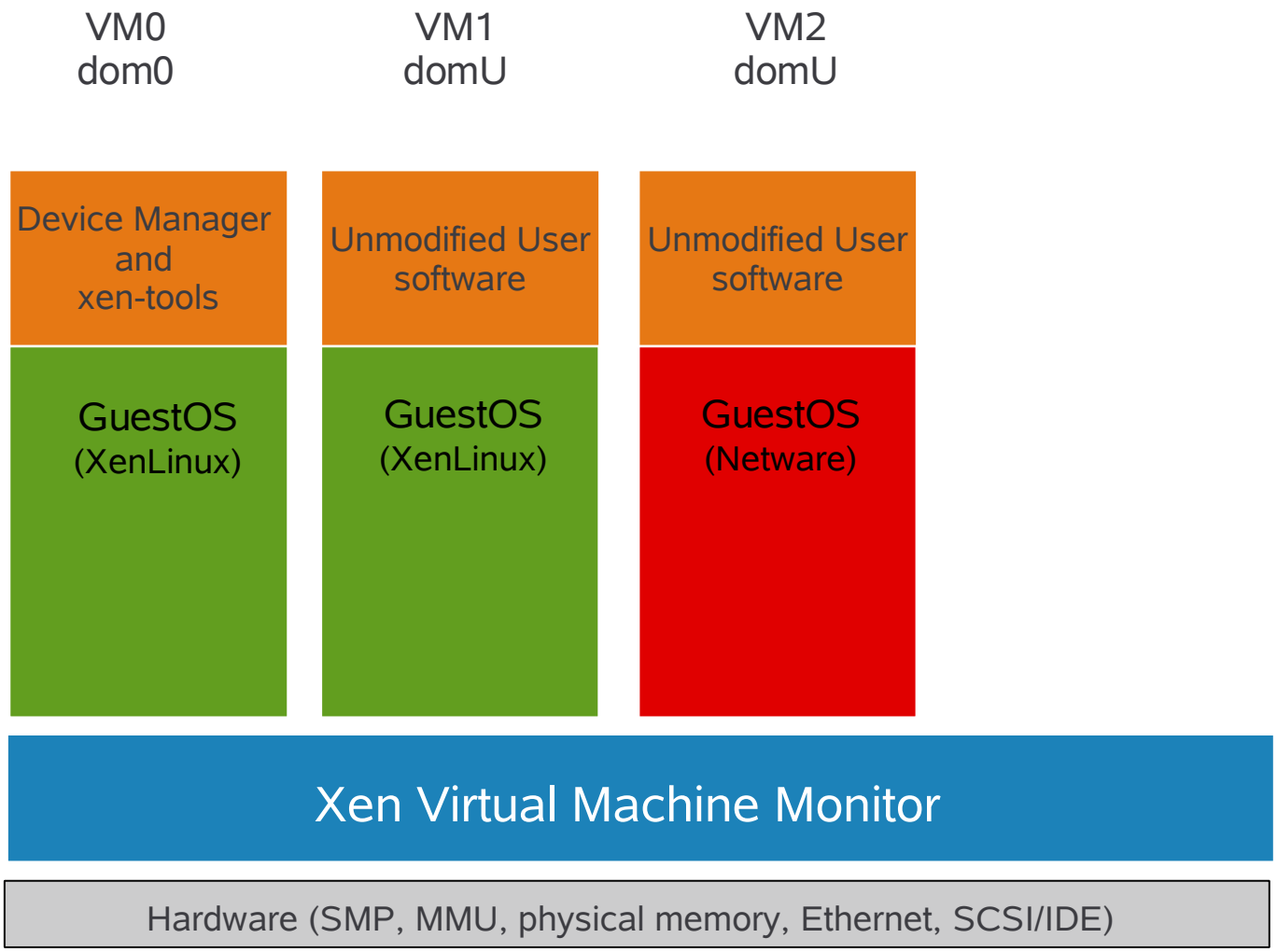
Paravirtualisation

With traditional hardware

Modified kernel required in domU (“guest”) OS

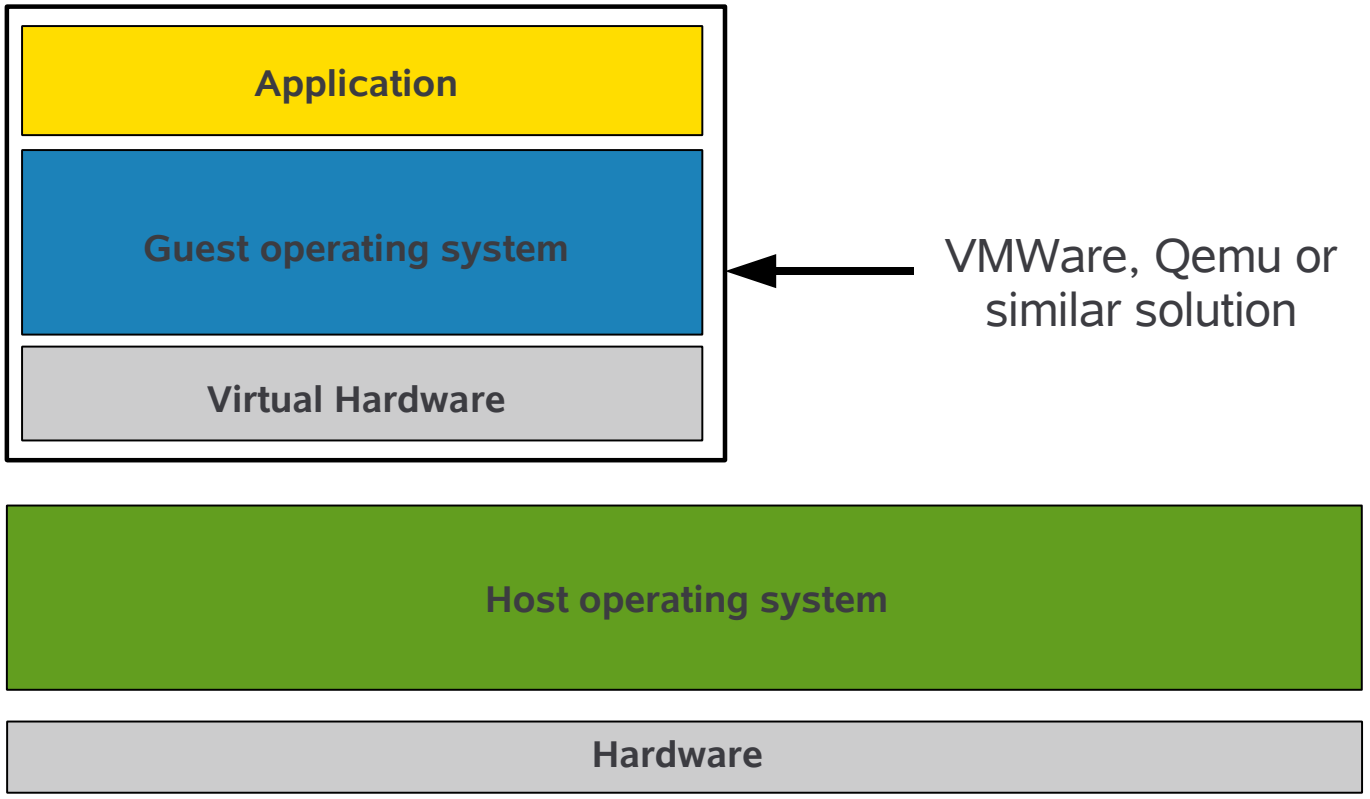
- Linux
- Netware 6.5 SP6 (authorised beta)
- NetBSD 3.0 / 3.1
- FreeBSD (7.0 CURRENT) [i.e. experimental future]
- OpenBSD (recently announced)
- Plan 9
- Open Solaris

Xen 3.0 Architecture (paravirtualisation)

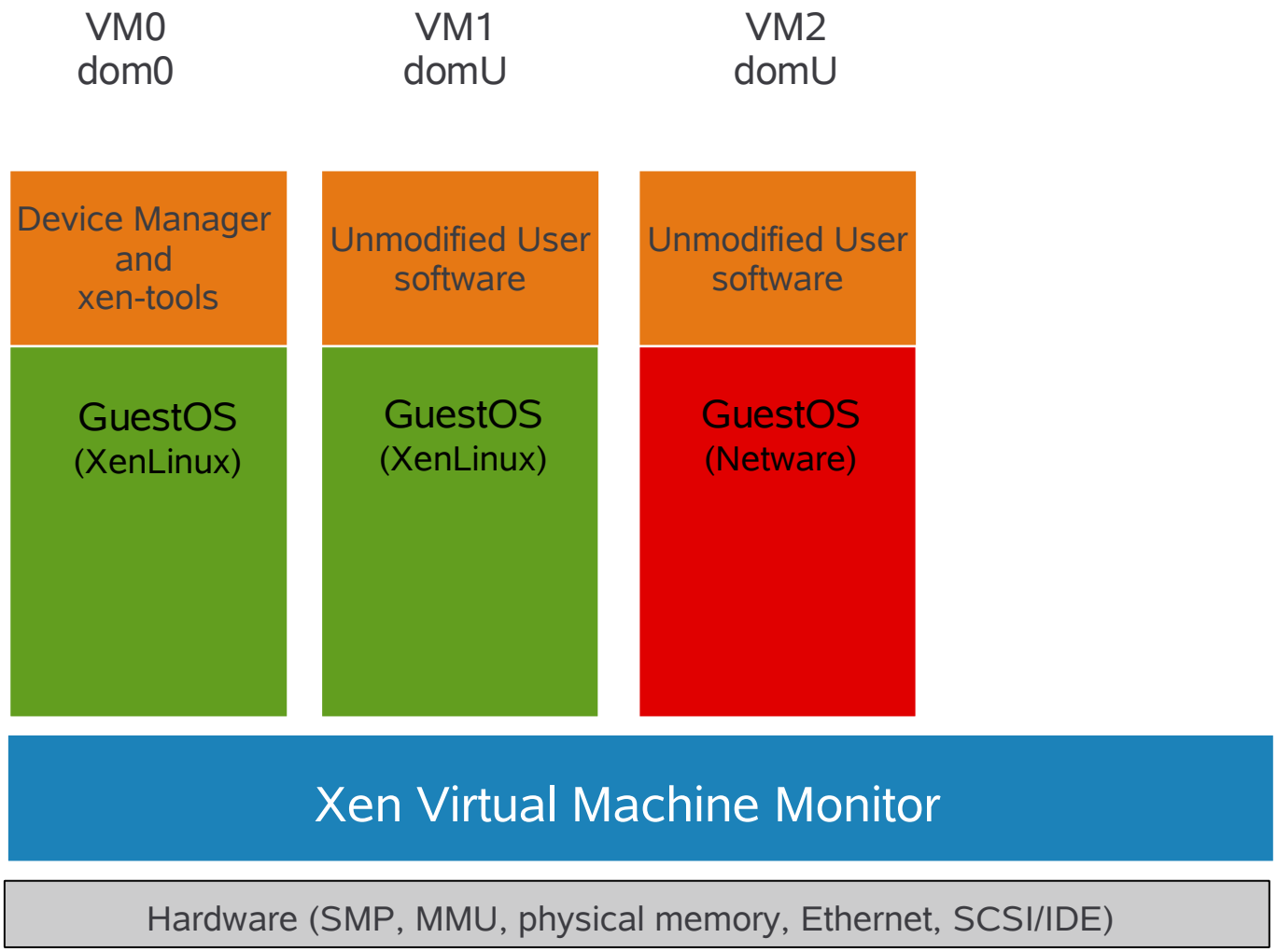


How Xen does not work

Ceci n'est pas le Xen...



Xen 3.0 Architecture (paravirtualisation)



Managing “guest” domains

The xen-tools package provides the xm command

```
xm create [-c] configfile
xm list
xm destroy domain-id
xm top
xm console domain-id
xm mem-set domain-id
xm migrate [-l] domain-id host
xm save domain-id
```

Managing “guest” domains

Configuration files

- on SLES by default in `/etc/xen/vm/`

Virtual disks

- on SLES by default in `/var/lib/xen/images/`
  - (note – can use “real disk device” - simultaneous dual-boot system or entire LUN from SAN)
- But can be anywhere: for testing it is convenient to keep configuration file and disk image in same directory

Xen domU configuration file example

```
disk = [ 'file:/var/lib/xen/images/fedora/hda,hda,w']
memory = 256
vcpus = 1
builder = 'linux'
name = 'fedora'
vif = [ 'mac=00:16:3e:ff:f6:7f' ]
dhcp = "dhcp"
extra = ' TERM=xterm'
bootloader = '/usr/lib/xen/boot/domUloader.py'
bootentry = 'hda1:vmlinuz-2.6.15-1.2054_FC5xenU,\
 initrd-2.6.15-1.2054_FC5xenU.img'
```

Xen domU configuration file example

```
disk = [ 'file:/var/lib/xen/images/netbsd.dsk,hda,w']
memory = 256
vcpus = 1
builder = 'linux'
kernel = '/home/roger/netbsd/netbsd-XEN3_U'
name = 'netbsd'
vif = [ 'mac=00:16:3e:ff:f6:80' ]
dhcp = "dhcp"
```

Global configuration file

`/etc/xen/xend-config.sxp`

Allows the configuration of:

- migration (will we accept incoming migrations?)
- minimum memory for dom0
- CPU allocation for dom0
- location of custom networking scripts
- location of log file (by default `/var/log/xend.log`)

Xen control daemon

xend (Xen control daemon)

must be running if anything is to work

```
rcxend start|stop|restart|status
```

xendomains (Controls domUs if required)

start configured domains when dom0 comes up or manually:

```
rcxendomains start
```

Managing domains

Memory

- `xm mem-set domain-id`
  - dynamically changes memory allocated to domain
  - change will immediately show up within the domain
- `xm mem-set Domain-0`
  - applies equally to Dom0
  - in some cases setting this to a low value is useful
    - (particularly for hardware assisted virtualisation)
  - may need to reset to “correct value” after “destroying” domains

Managing domains

- Migration
  - assumes shared storage
  - suitable file system
  - shutdown and migrate
  - or ...
  - live migration (virtually imperceptible to clients)
    - but consumes considerable resources on the host prior to the migration

Managing domains

- Using YaST
  - Package `yast2-vm`
  - Command `yast2 xen`
  - YaST allows installation onto new domains
  - From CD or iso image
  - From network source
  - YaST starts an xterm connected to the domU console
  - Writes a suitable configuration file and creates disk image
- Easy with SUSE installation source
  - (because YaST knows location and name of kernel and initrd)
  - Slightly harder with non-SUSE installation source

Managing domains (paravirtualisation)

Disk image is a file (or real disk partition)

Can contain all partitions created by installation

`lomount` allows us to mount selected partitions e.g.:

```
lomount -diskimage domu.img -partition 1 /mnt
```

Useful for finding kernel and initrd filenames

Booting is done “externally” (by specifying kernel or by `domUloader.py`)

There is no “virtual BIOS”

MBR and boot loader within domU are irrelevant

Creating domains

Easy alternative method: Qemu

- Create disk image in Qemu
- Run in xen

```
qemu-img create guest.img 4G
qemu -hda guest.img -cdrom install-cd.iso -boot d -m 512M
```

Full (hardware assisted) virtualisation

Full (hardware assisted) virtualisation

Requires latest processors

- AMD 'Pacifica' (“I/O Virtualisation Technology” or “AMD-V”)
- Intel 'Vanderpool' (“VT-x”)

```
grep flags /proc/cpuinfo
```

look for:

- `svm` (AMD)
- `vmx` (Intel)

Full (hardware assisted) virtualisation

- Allows unmodified guest OS
  - In particular legacy operating systems from Microsoft
  - Still widely used
  - Consolidate Windows systems as guests in Xen environment

Full (hardware assisted) virtualisation

- Uses virtual BIOS
- Uses SDL or VNC to provide display
- Unmodified guest can be
  - Windows
  - legacy Linux (unmodified)
  - other x86 OS including Solaris x86

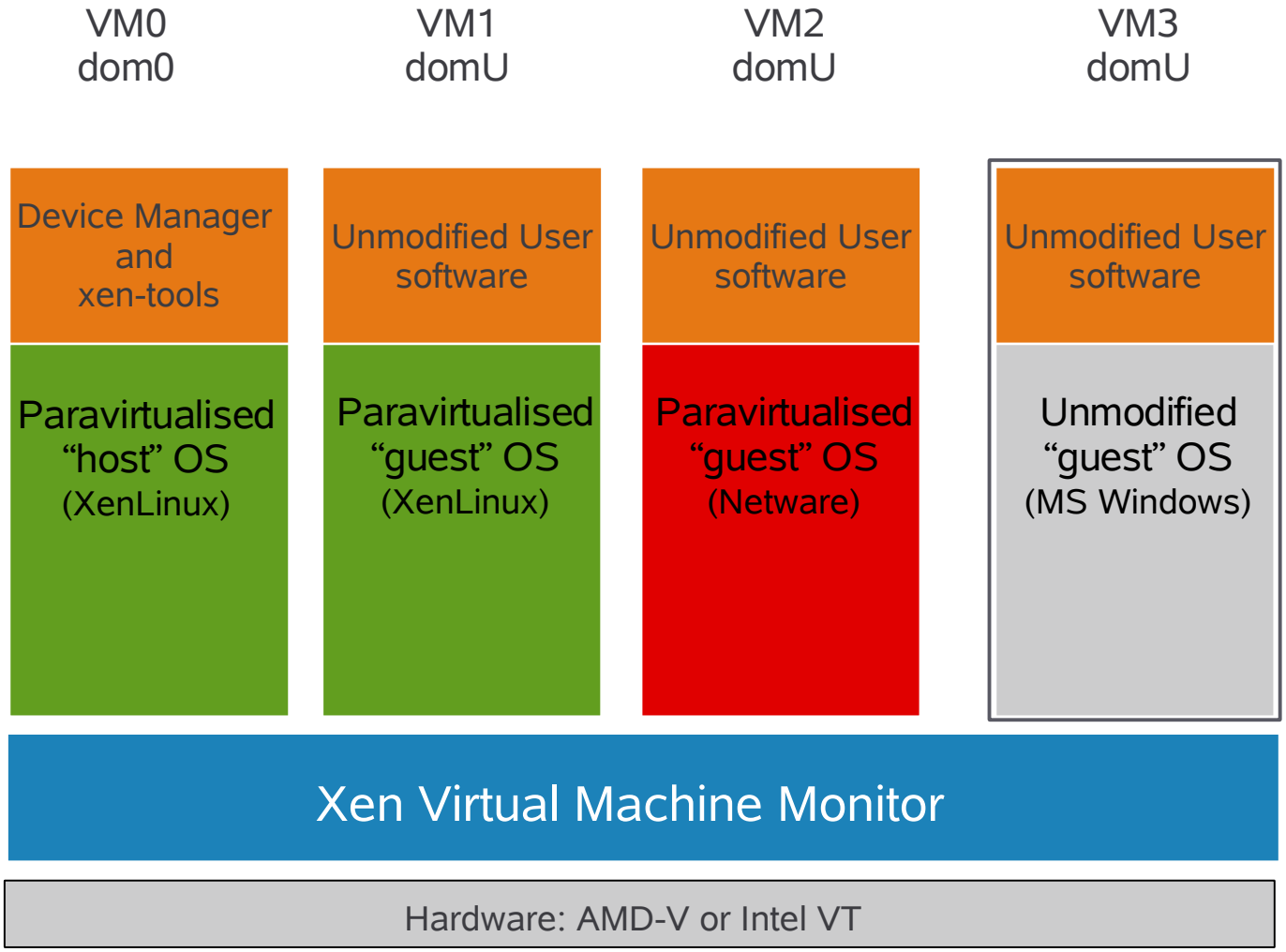
Full (hardware assisted) virtualisation

Install through YaST from iso image or CD

(some manual editing of configuration file may be needed)

Or create in Qemu, run in Xen

Xen 3.0 Architecture (with full virtualisation)



Driver domains

Normally driver access will be handled through dom0

- But unprivileged domains can also be given access to hardware through native drivers
- Permits hosting native drivers in domains other than domain 0
- Supports pass-through as well as emulation mode

Virtual Device Drivers

XEN supports Driver Domains:

- Driver domains have physical access to the I/O devices
- A driver domain virtualises the I/O device for access by other guest OS instances
- Multiple Guests share the same physical device

XEN also allows a guest OS to own a physical device exclusively (not shared)

- This deployment scenario is particularly relevant for performance critical services

SMP support

True SMP support in domUs

Dedicate or allocate processor power to guests

Allocate to a particular CPU or group of CPUs

Or allow Xen to allocate dynamically

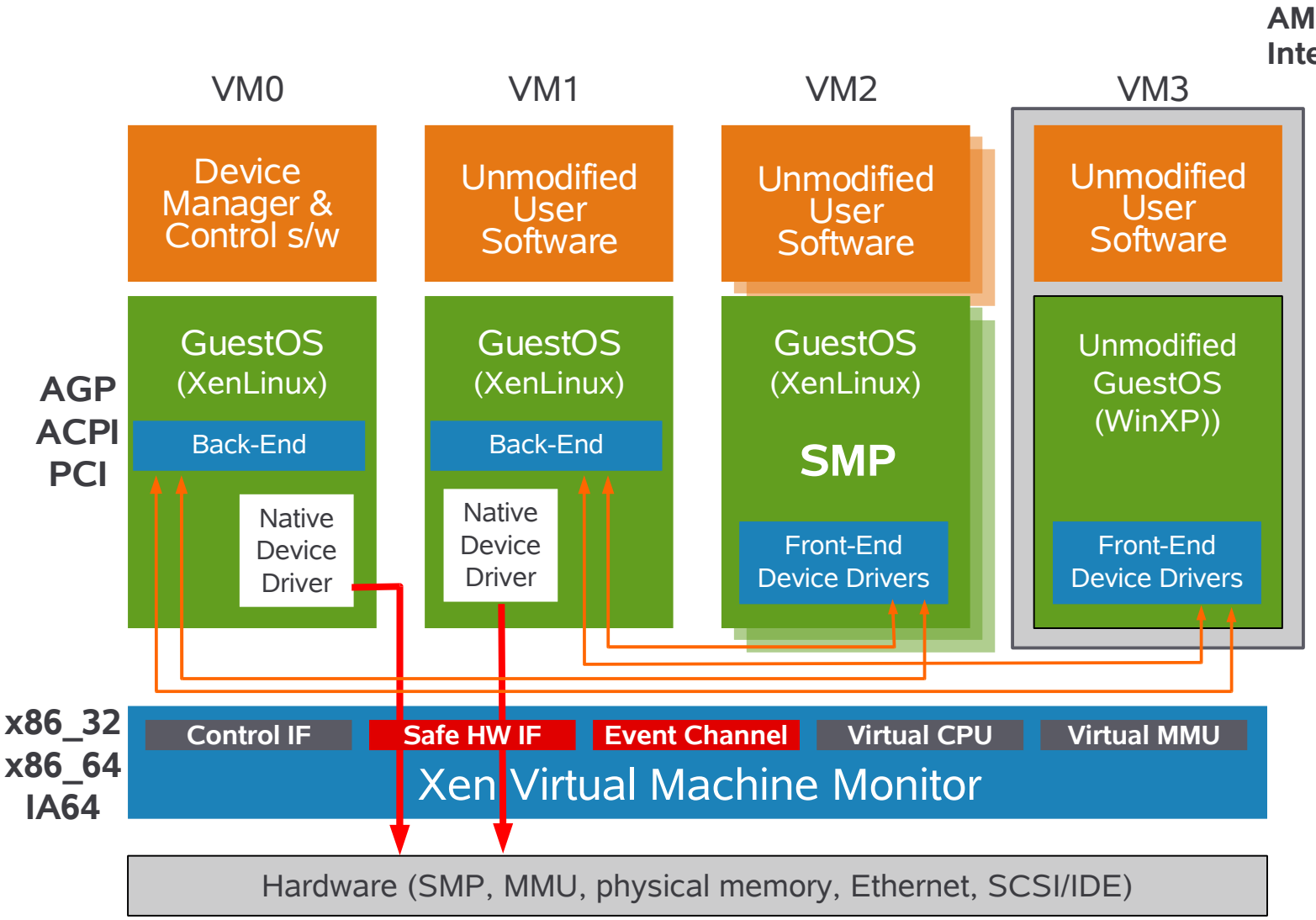
VMs can be configured to see more CPUs than are physically present

CPU Virtualization

Virtual CPUs (VCPUs)

- Each guest can be assigned an arbitrary number of VCPUs
- VCPUs mapped to physical CPUs in the hypervisor
- VCPU binding and placement supported

Xen 3.0 Architecture



I/O Virtualization

Networking:

- Normal bridging, routing, iptables etc.

Block:

- Supports full decoupling between backend and frontend devices – a SCSI backend can be exported as an IDE disk

Support For Hardware Assisted Virtualisation (AMD-V or Intel VT-x)

Permits hosting unmodified binary OSs:

- Legacy Linux
- Windows platforms

Hardware support:

- CPU traps privileged instructions
- Supports expanded protection rings

Platform emulation:

- BIOS emulation
- Device models emulated (for booting)

Migration

VM Migration

- Why is VM relocation useful?
  - Managing a pool of VMs running on a cluster
  - Taking nodes down for maintenance
  - Load balancing VMs across the cluster
- Why is it a challenge?
  - VMs have lots of state
  - Some VMs will have soft real-time requirements
    - E.g. web servers, databases, game servers
  - Can only commit limited resources to migration

VM Migration

Currently well supported for

- Paravirtualisation with Linux domU

Not yet (Xen 3.0.3) working for

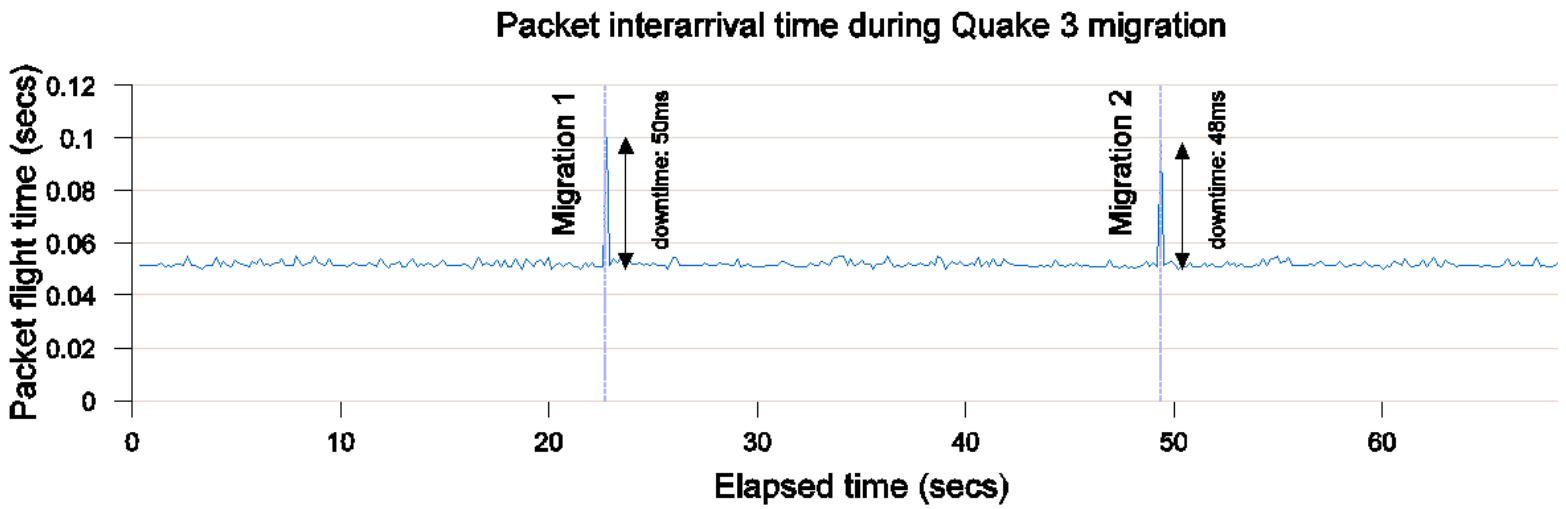
- Windows in hardware assisted virtualisation
  - (but this is working with bleeding edge code)
- Some non-Linux paravirtualised guests

Xen Migration Strategy

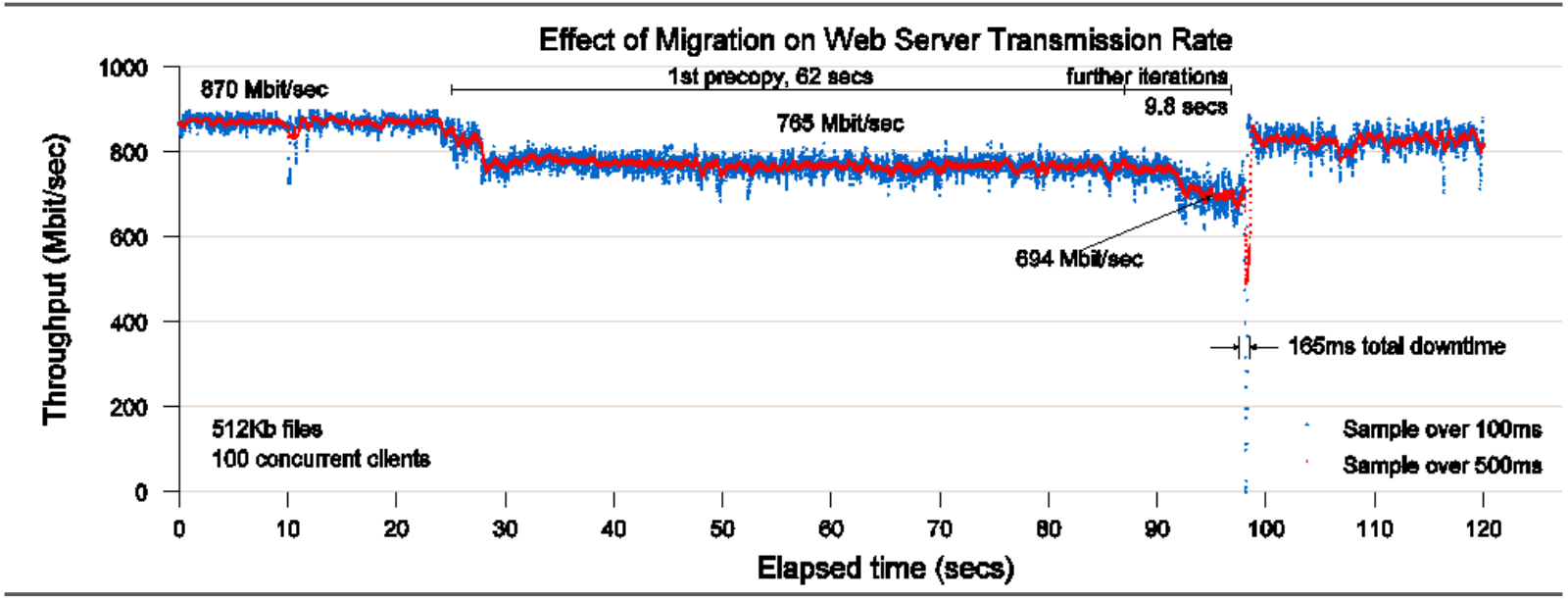
Supports services that may have soft-real-time constraints as well as resource-constrained migration:

- Control the I/O resources used for migration
- Migrate pages outside the “working set” without suspending the service
- Suspend the service during the migration of the “working set”

Quake 3 Server migration

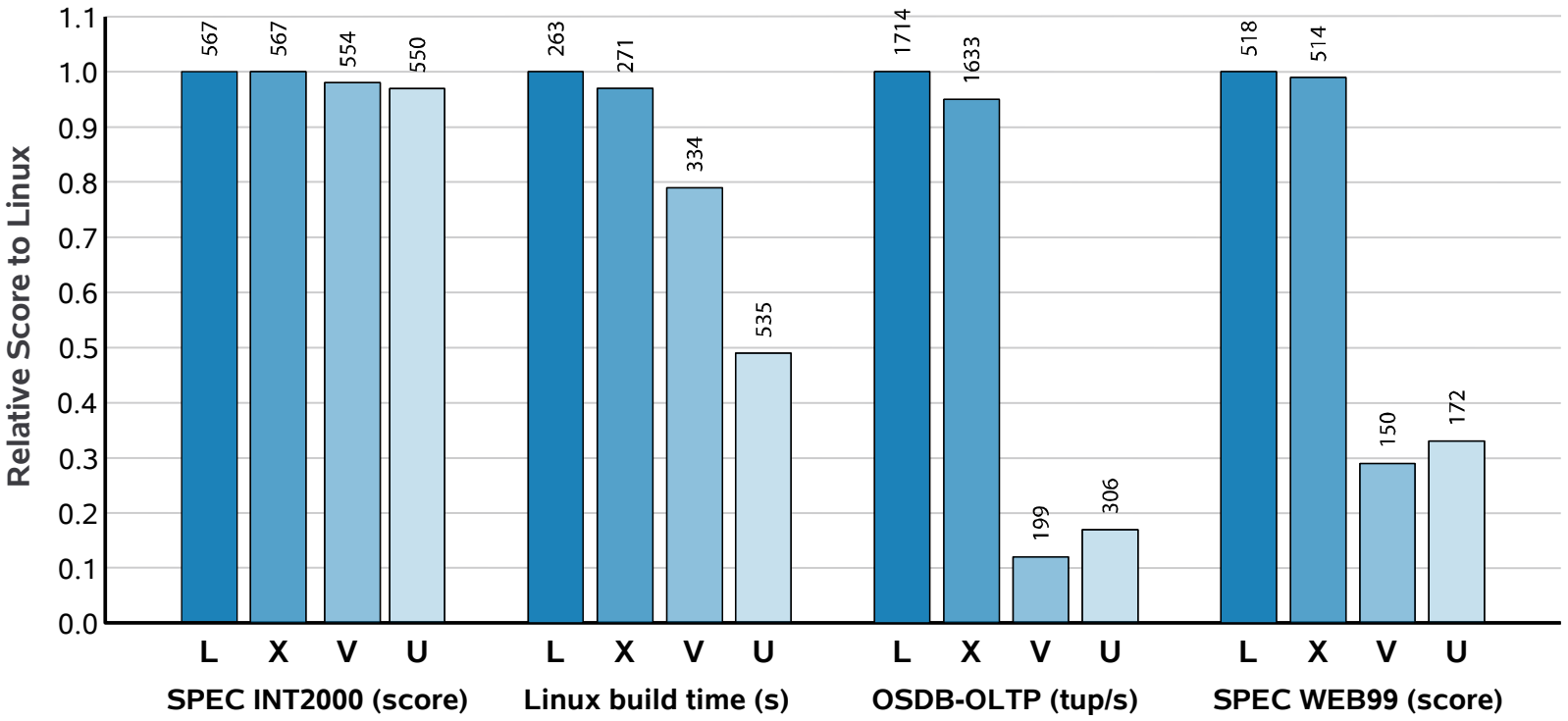


Rate Limited Migration



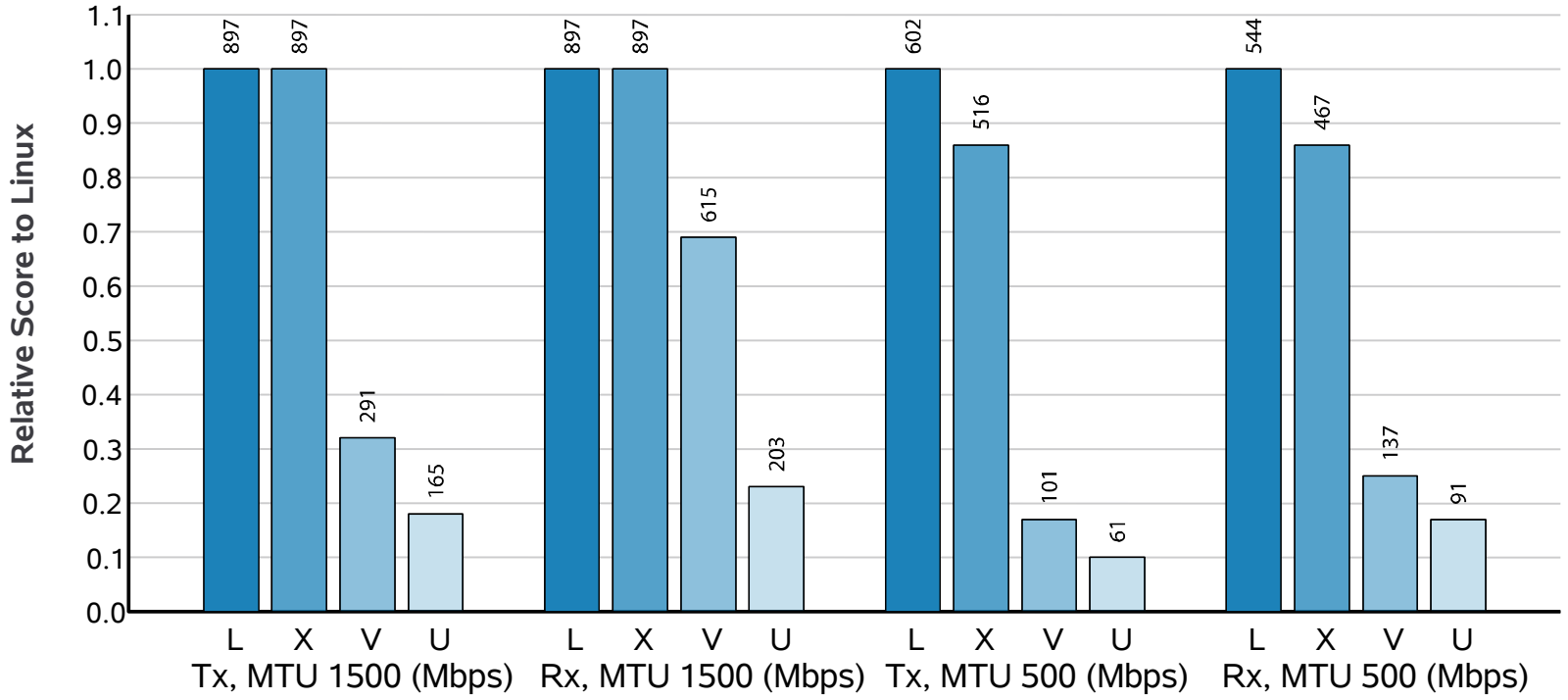
Performance measures

System Performance



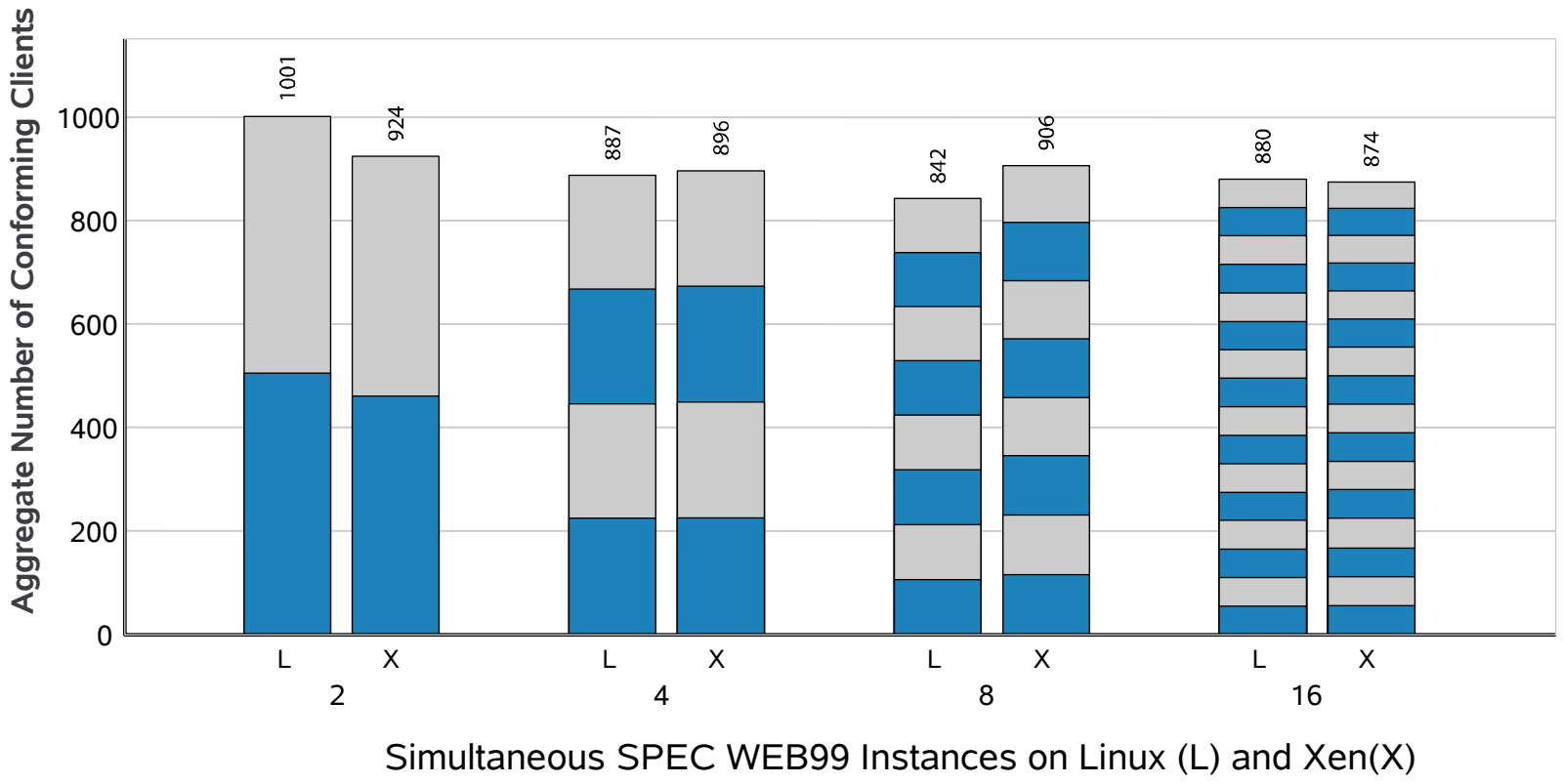
Benchmark suite running on Linux (L), Xen (X), VMware Workstation (V), and UML (U)

TCP Performance



TCP bandwidth on Linux (L), Xen (X), VMWare Workstation (V), and UML (U)

Scalability



Novell® Virtualization Strategy

Novell® Virtualisation Strategy

- Virtual Machine Monitor (VMM) will be based on Xen technology (Xen is the open source virtualisation project)
  - Xen has the backing of major industry players:
    - Intel, IBM, HP and AMD
  - Other major Linux distributions are backing Xen
- Virtualize both of Novell Operating Systems
  - Host NetWare® as a guest OS
  - Host SUSE® Linux Enterprise Server as a guest OS
- Implement a comprehensive management suite
  - Directory enabled
  - Identity based

Xen Enabled OS Offerings

SUSE® Linux Enterprise Server 9 SP3:

- Supported only as a domU guest instance

SUSE Linux Enterprise Server 10:

- Supported as both dom0 and domU

Virtualized NetWare®:

- Supported only as a domU

Managing the Virtual Data Center

Management of VMM and Guest OS will be CIM based:

- Local or remote management consoles
- CIM based management will allow for a variety of management consoles from a number of vendors to be used
  - Allow for differentiation
  - Novell will not lock management vendors out
- CIM based tools will enable virtual machine management
  - Startup and shutdown
  - Live migration

Management tools

Novell is working with IBM, XenSource and others on an Open Source project on CIM providers

Using OpenWBEM interface standard for management tools

Policy Driven Data Center

Strategy: Directory enabled, policy based, utility computing

- Directory Enabled
  - Store virtual machine objects in the directory
    - Cluster Objects, VMs, Physical Nodes, Resource templates, Preconfigured OS configurations
- Policy engines to monitor and manage workload
- Virtual storage to enable fluid computing within the virtual data center

Novell® Reference Architecture

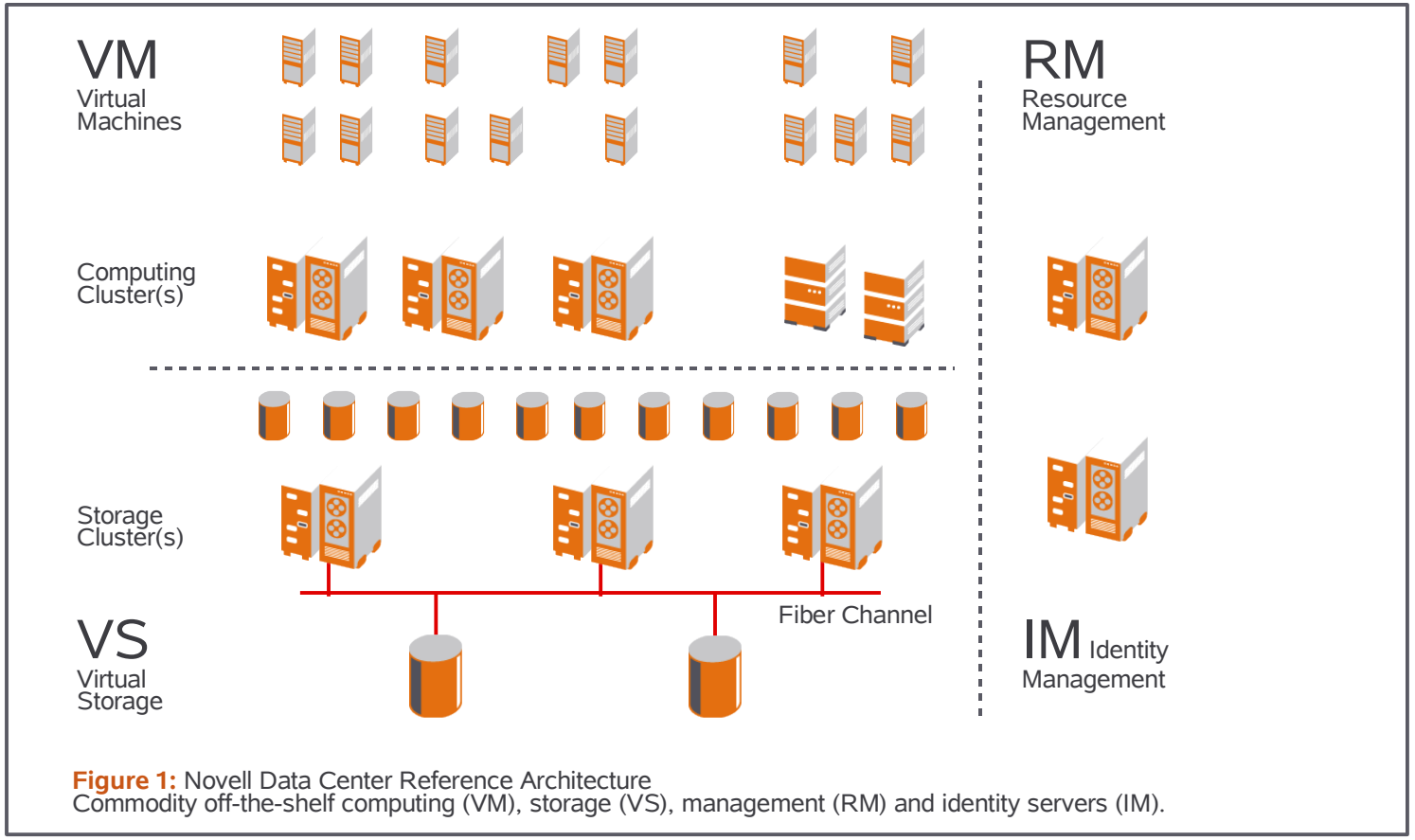


Figure 1: Novell Data Center Reference Architecture
Commodity off-the-shelf computing (VM), storage (VS), management (RM) and identity servers (IM).

Recent events

- Novell / SUSE is first to market (again)
  - (Linux for IA64, Opteron, zSeries, pSeries – unchallenged technological leader in bringing enterprise Linux to new platforms)
- Xen is ready for enterprise use in terms of stability
  - Enterprise-wide management tools coming very soon
  - SLES 10 or “foreign” OS under Xen is supported:
    - Windows NT, 2000, 2003, Vista (with suitable hardware)
    - SLES 8, 9, 10 (SLES9 SP3 and 10 will run paravirtualised)
    - Solaris x86 (with suitable hardware)
    - RHEL 3, 4 (with suitable hardware)
  - Coming soon
    - NetWare 6.5 SP6 and OES 2 (paravirtualised)

Many thanks to

Loan of hardware and generous help from:

Demonstration

Questions


Novell®