RED HAT :: SAN DIEGO :: 2007

SUMMIT

Best Practices using Xen Virtualization w/ Red Hat Enterprise Linux 5

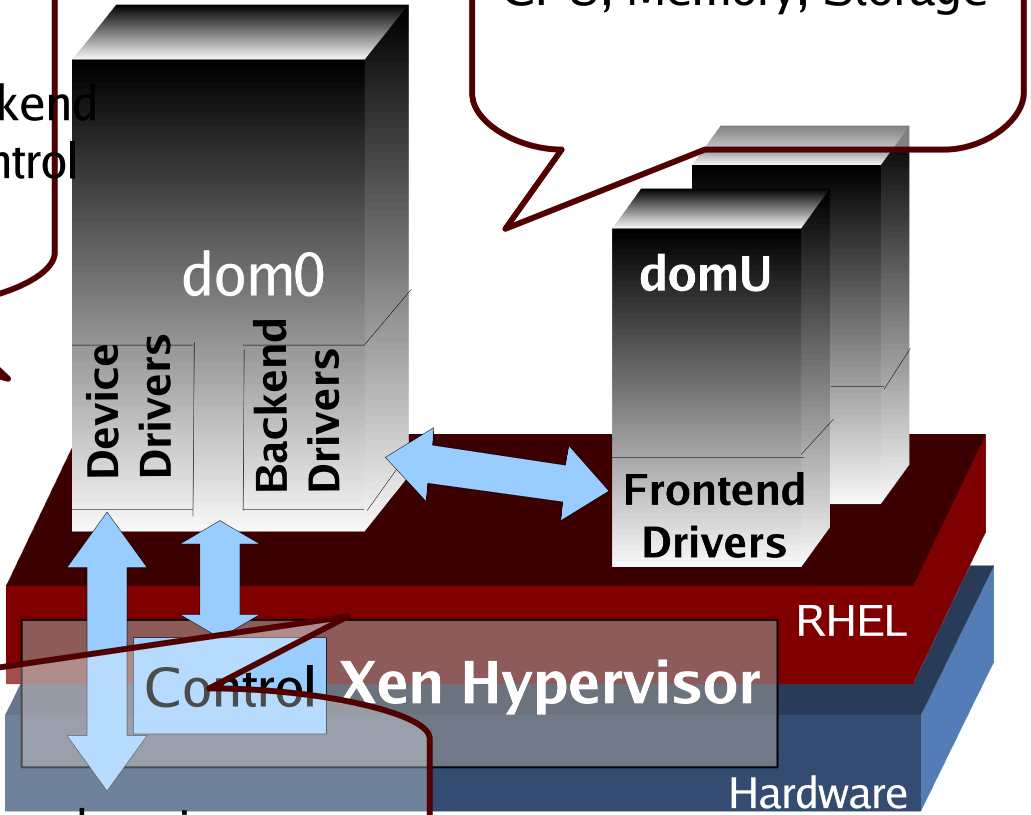
Name D. John Shakshober, Jan Mark Holzer

Date May 9, 2007

Xen Architecture

Domain0
Privileged Domain,the host.
Provides hardware support (backend drivers) interfaces for guests control and management tools

Unprivileged Domain:
The Guest or the Virtual Machine.
CPU, Memory, Storage



Xen Hypervisor
provides IRQ routing, Scheduling , and inter-domains communications. The Hypervisor with the Dom0 Device Drivers provide transparent sharing of resources. It also enforces strict resource limitations (example: RAM).

RHEL5/Virt Features

- Based on Xen 3.0.3
- SMP and UP guest support
- Para-virt and Fully-virt guest support
  - Full-Virt/HVM requires appropriate hardware support with Intel/VT or AMD/AMD-V enablers (to run unmodified operating systems)
- All major Xen 3.0.3 features supported in RHEL5
  - Credit Scheduler
  - Xen Virtual Framebuffer Support
  - Migration
  - Pause/Resume/Save/Restore support for guests
  - Dynamic control of resources (Memory/CPU)
  - Virtual Network support (bridged and routed)
- Additional features to enhance Xen base features
  - Installation support via virt-manager and virt-install
  - RHN integration
  - Support for Anaconda for guest installations
  - Broad testing and QA coverage

Hypervisor – guest domain compatibility

	32bit PAE paravirt Guest	*32bit HVM Guest*	*64bit paravirt Guest*	*64bit HVM Guest*
32bit (PAE) Hypervisor / dom0	🟩	🟩	🟥	🟥
64bit Hypervisor / dom0	🟥	🟩	🟩	🟩

RHEL-5 Virtualization Status

- Limitations (dom0/domU)
  - Memory
    - i386 – ~16GB (no 4G/4G support)
    - x86_64 - ~64GB GA kernel, ~256GB day0 errata kernel
    - ia64 – tech preview
  - CPUs
    - i386 - 32 processors on HV, 32 processors on dom0/domU
    - x86_64 - 32 processors on HV, 32 processors on dom0/domU
    - ia64 – tech preview
    - The dom0/domU limits are because of a hypercall limitation; changing this would change the ABI, so this will probably not happen for the life of RHEL-5.
    - Current HV limit is (probably) just implementation limited. There is no reason the HV couldn't see 64 processors and assign 32 to dom0 and 32 to a domU. Possible 5.1 material, depending on upstream Xen.
  - Network
    - Limited to 3 virtual NICs per domain.

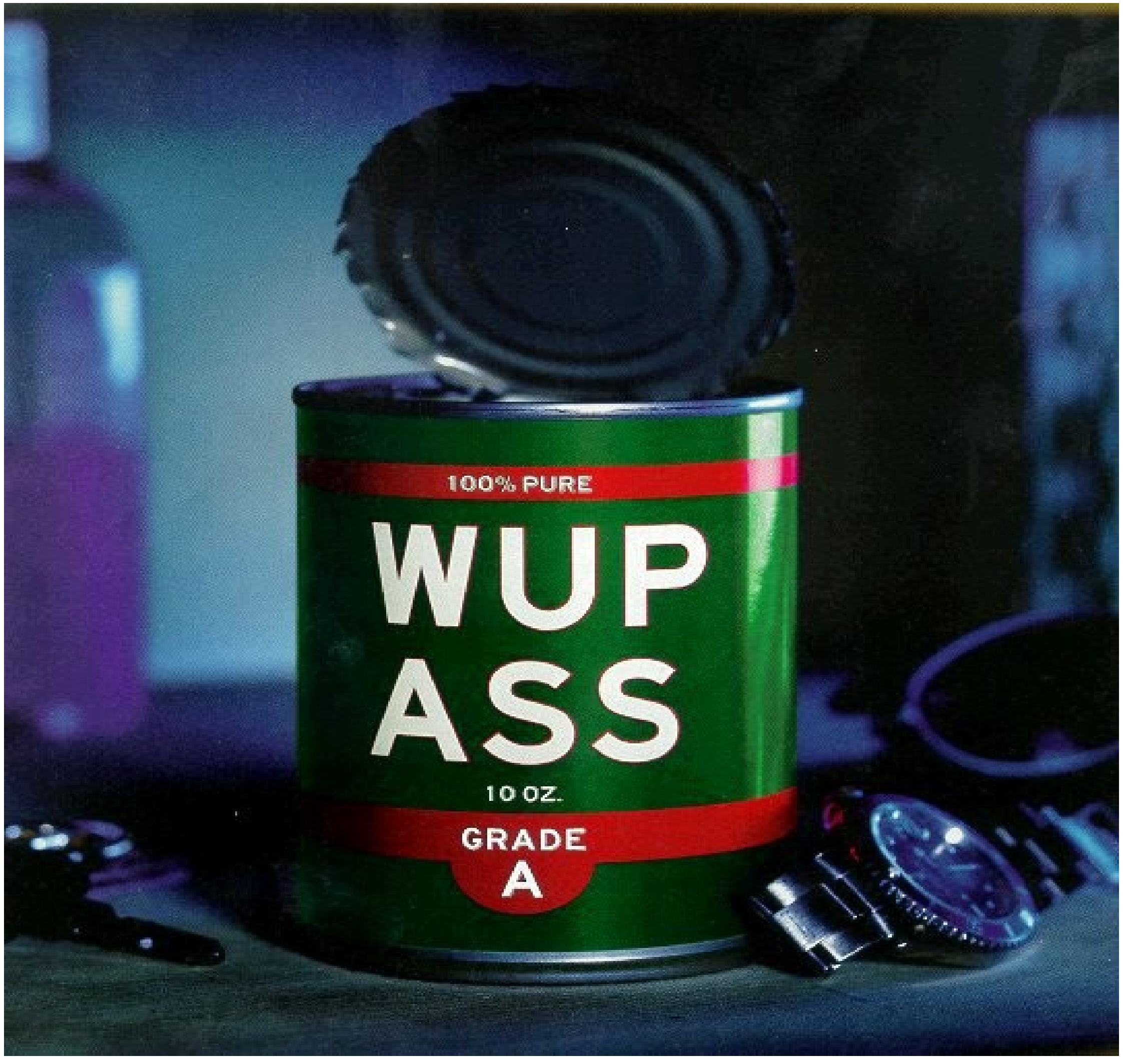
RHEL-5 Virtualization Status (continued)

- Limitations (fully virtualized)
  - Memory
    - i386 – ~1.5G
    - x86_64 – Unknown, > 4G
  - CPUs
    - i386 – single processor
    - x86_64 – single processor
    - ia64 – single processor

RHEL-4 Virtualization Status

- domU support only
- Supported arches – i386, x86_64
- Support for save/restore/migrate
- No PCI passthrough support
- Limitations
  - Memory
    - i386 - 16GB (no 4G/4G support)
    - x86_64 - Unknown, theoretically 256G and higher
  - CPUs
    - i386 - 32 CPUs
    - x86_64 - 32 CPUs
    - Limitations cannot be lifted because of HV ABI limitations
    - as well as kABI issues in the domU kernel.
  - Network
    - Limited to 3 virtual NICs

Forward Looking Planning



Product features subject to change prior to availability

Upstream Xen

- Make push to get Xen support in mainline kernel
  - Ported to use paravirt ops
  - Fairly likely for -mm in short term
  - Limited functionality, domU only, UP only, i386 only
- Decouple the dom0/domU kernel from hypervisor/tools
  - This is good, coalesce red hat and other linux trees
- Support for more hardware features
  - E/NPT – Nested Page Tables
  - VT-D – Intel IOMMU support
  - NUMA
- PPC support

RHEL Updates

- RHEL 5.1 Update
  - Focus: HVM features (improvements post 3.0.3)
  - Paravirtualized drivers
    - RHEL3 , <=RHEL4u4 , Windows 2003 (possibly Vista) incl WHQL
  - Xen 3.0.5 hypervisor (possible 3.0.5+ patches)
  - Hybrid user space to ensure RHEL 5 GA compatibility w/ enablement of 3.0.5+ features (eg, NUMA topology, loopback removal, kexec/kdump)
    - Not shipping xen-api in RHEL 5
- Post RHEL 5.1
  - Continue virtualization management infrastructure
  - Path for CIM support
  - Continue to improve add'l HVM support for maximum number of OSes

Product features subject to change prior to availability

Beyond RHEL 5.1

- Common Virtualization Infrastructure
    - CIM on libvirt
    - VLAN & storage management APIs
    - Support for KVM in Fedora Core 7
- Joint Partner Efforts
  - Ongoing Virt-Manager development positive
  - Increased PV/HVM driver coverage (testing)
  - Platform Support
    - Joint support for 3rd party OS, drivers, certification, ... ?
  - CIM support?
  - Large scale platform testing
  - NUMA topology support

Basic Configuration Recommendations

- Considerations
  - Secure RHEL5 platform layer before installing any virtual machines or applications
  - Run SElinux to run in 'enforcing' mode
  - Remove or disable any unwanted services
    - AutoFS, NFS, FTP, WWW, NIS, telnetd, sendmail etc...
  - Only add minimum number of user accounts needed for platform management
  - Avoid running applications on dom0/Hypervisor
    - Running applications in dom0 may impact virtual machine performance
  - Use central location for virtual machine installations
    - Will make it easier to move to shared storage later on

Basic Xen commands

- Once you have your first guest installed you can use the following commands for some basic management
- To startup a guest
  - # /usr/sbin/xm create -c GuestName
  - Where GuestName is the name you gave for your guest during the installation
  - The -c will attach a xen console to your vm
- A variety of other commands are available via xm including
  - # /usr/sbin/xm help
  - For a list of commands that can be run
  - Use '--long' in addition for extended help text
  - You can also use #/usr/sbin/xm help –help 'Command' for a specific command

Basic Xen commands contd.

- # /usr/sbin/xm list (--long)
  - List running domains/guest and their status/accumulated CPU time
- # /usr/sbin/xm top
  - for a display showing what your virtual machines are doing similar to that provided by top
- # /usr/sbin/xm shutdown GuestName
  - to nicely shut down a guest OS where foo is the name of your guest.
- # /usr/sbin/xm destroy GuestName
  - To power down a guest (hard reset)

Basic Xen commands (Suspend/Resume)

- `# /usr/sbin/xm save GuestName GuestName.restore`
  - to save the state of the guest 'GuestName' to the file GuestName.restore
- `# /usr/sbin/xm restore GuestName.restore`
  - to restore the above saved guest
- `# /usr/sbin/xm pause GuestName`
  - to suspend a running guest (release CPU cycles but retain memory footprint)
- `# /usr/sbin/xm unpause GuestName`
  - to resume a previously suspended guest

Basic Xen commands (Resource Management)

- # /usr/sbin/xm vcpu-set <dom> <value>
  - set the number of CPUs available to <dom> to <value> (only works for dom0/paravirtualized guests)
- # /usr/sbin/xm vcpu-list
  - List the physical-virtual CPU bindings
- # /usr/sbin/xm mem-set <dom> <value>
  - balloon <dom> up or down to <value> (only works for dom0/paravirtualized guests)
- # /usr/sbin/xm sched-credit -d <DomainID>
  - Display credit schedule information and set cap/weight for individual domain

Other Useful Commands and Tools

- Basic Xen commands
  - xm log
  - xm dmesg
  - xm info
  - xm top
- virsh
  - virsh (dumpxml GuestName)
- Tools
  - strace, lsof, iostat/vmstat
  - Systemtap
  - /var/log/messages
  - /var/log/xen
  - AVC messages (setroubleshoot)

Tools (continued)

- xm mem-set
- xm vcpu-list
  - list virtual CPU assignments/placements[root@grumble xen]# xm vcpu-list

```
–   Name                    ID VCPUs   CPU State   Time(s) CPU Affinity
–   Domain-0                 0     0      0   r--      708.9 any cpu
–   Domain-0                 0     1      1   -b-      572.1 any cpu
–   r5b2-mySQL01            13     0      1   -b-       16.1 any cpu
```

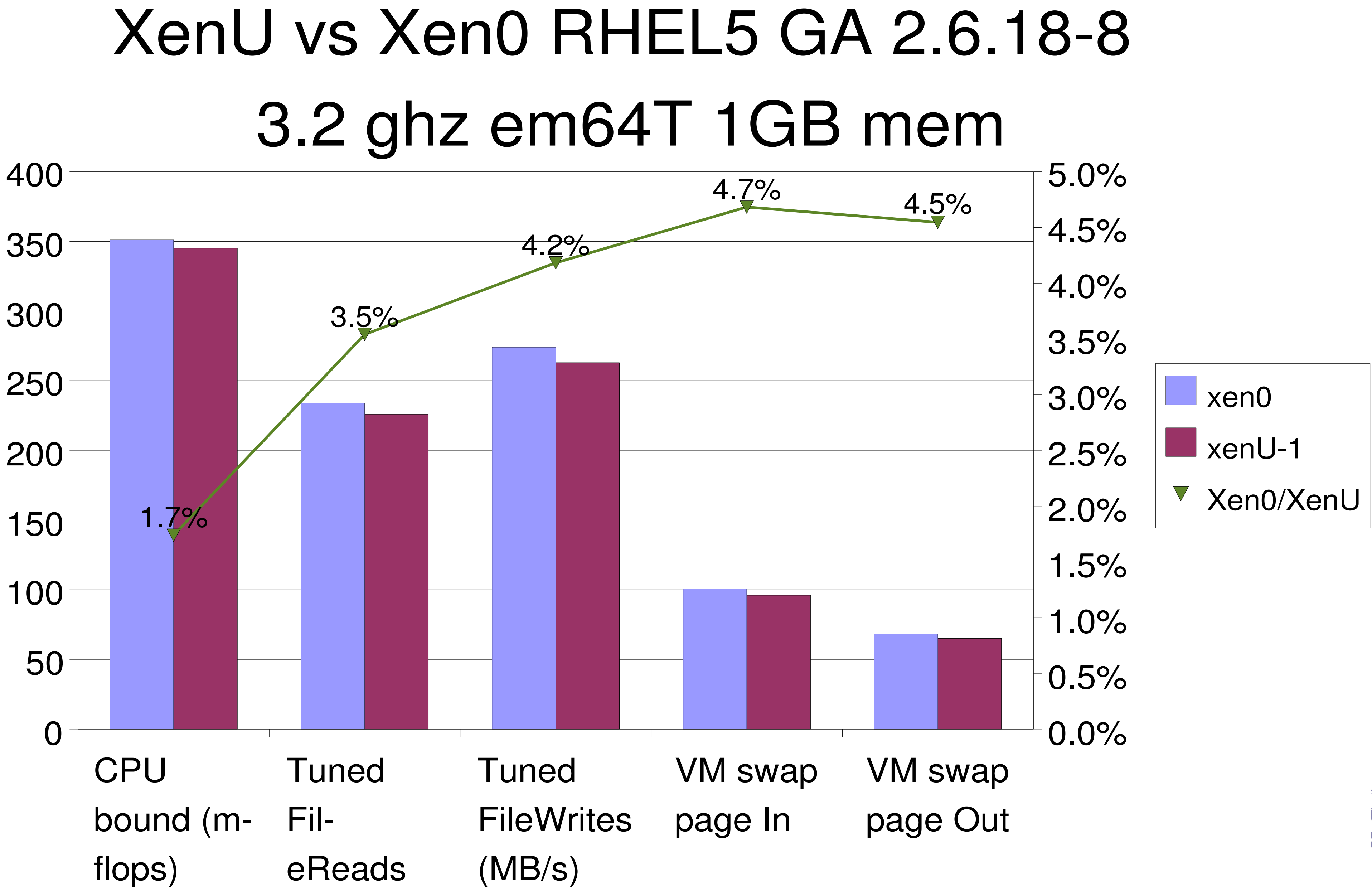
- xm vcpu-pin
- xm vcpu-set
- xm sched-credit
  - display scheduler parameters for a given domain

```
–   [root@grumble xen]# xm sched-credit -d 0
–   {'cap': 0, 'weight': 256}
–   [root@grumble xen]# xm sched-credit -d 13
–   {'cap': 25, 'weight': 256}
```

Red Hat Virtualization Performance Testing

- 3 Ghz, EM64T 2cpu w/ HT, 16GB mem, Fiber Channel
- 2.4 Ghz AMD64 2cpu w/ AMDv, 4GB mem 1gb-iSCSI
- Testing RHEL5 RC1 2.6.18-8
  - CPU (linpack), VM(aim), FS/IO (Iozone), Net (Netperf)
  - Xen0 (hypervisor kernel) native perf baseline
  - XenU guest, measure overhead = xenU/xen0
  - Run multiple guest share resources
    - 1GB memory per guest
    - FC Qlogic/Emulex, iSCSI e1000 nic
      - File domains, phy-disk, lvm-volumes
    - Single 1 gigabit network cards
- Efficiency = Sum [xenX]/xen0

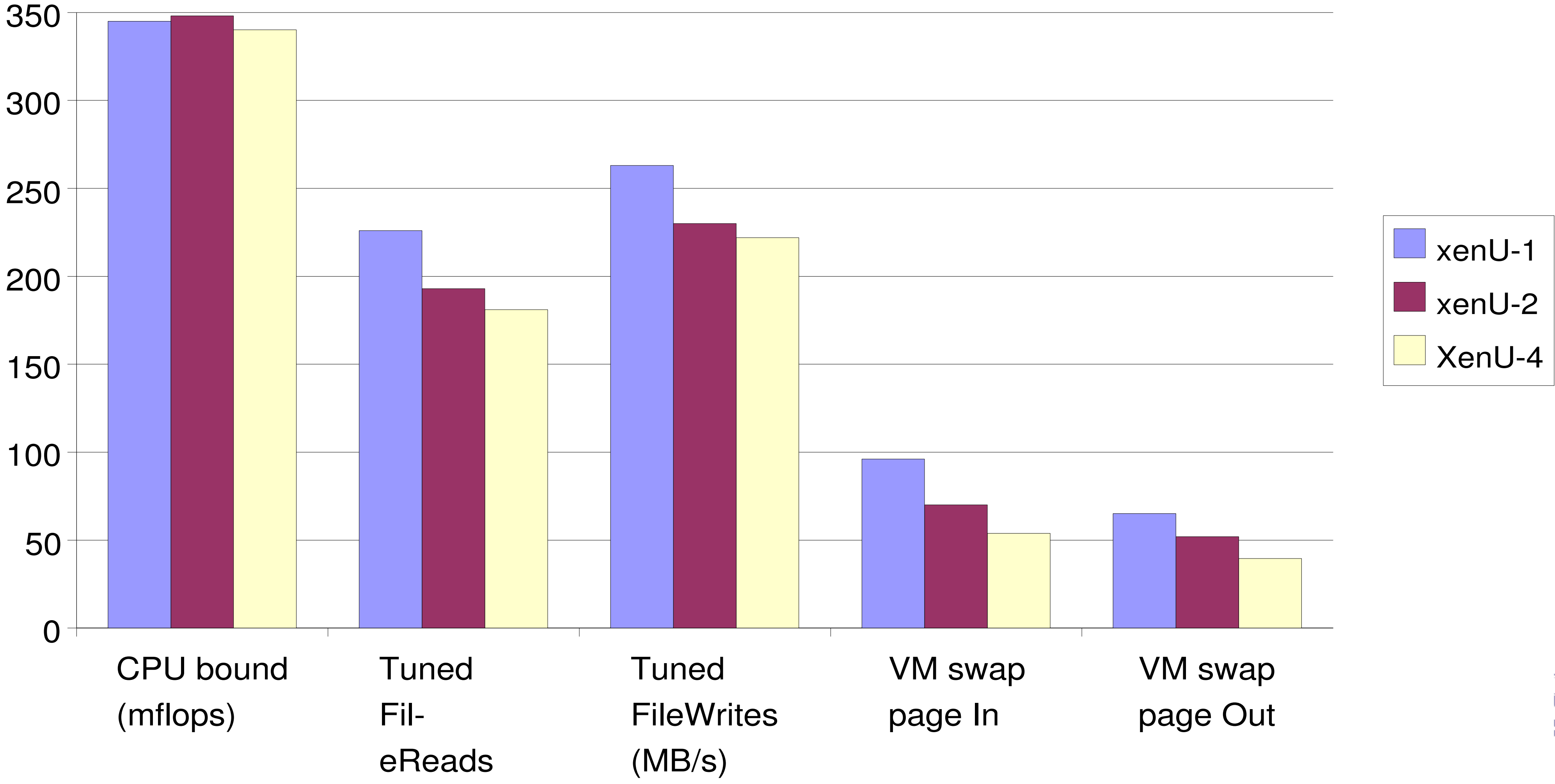
Red Hat Virt Single-guest Performance



Red Hat Virt Multi-guest Performance

Multiiple XenU Scalability 1,2,4 on RHEL5 GA 2.6.18-8

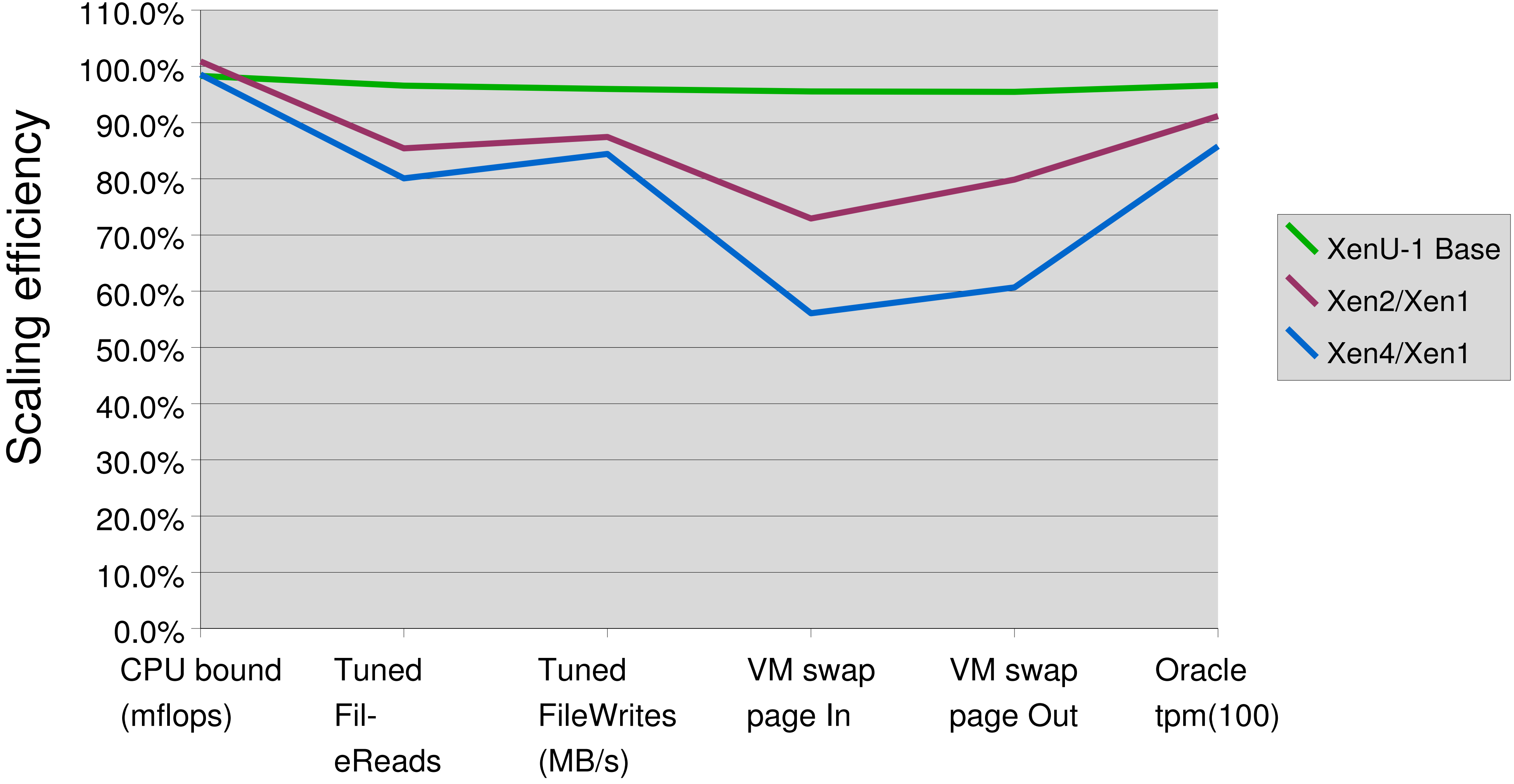
3.2 Ghz 2-cpu/2ht em64T 4GB memory, 1GB/xen guest



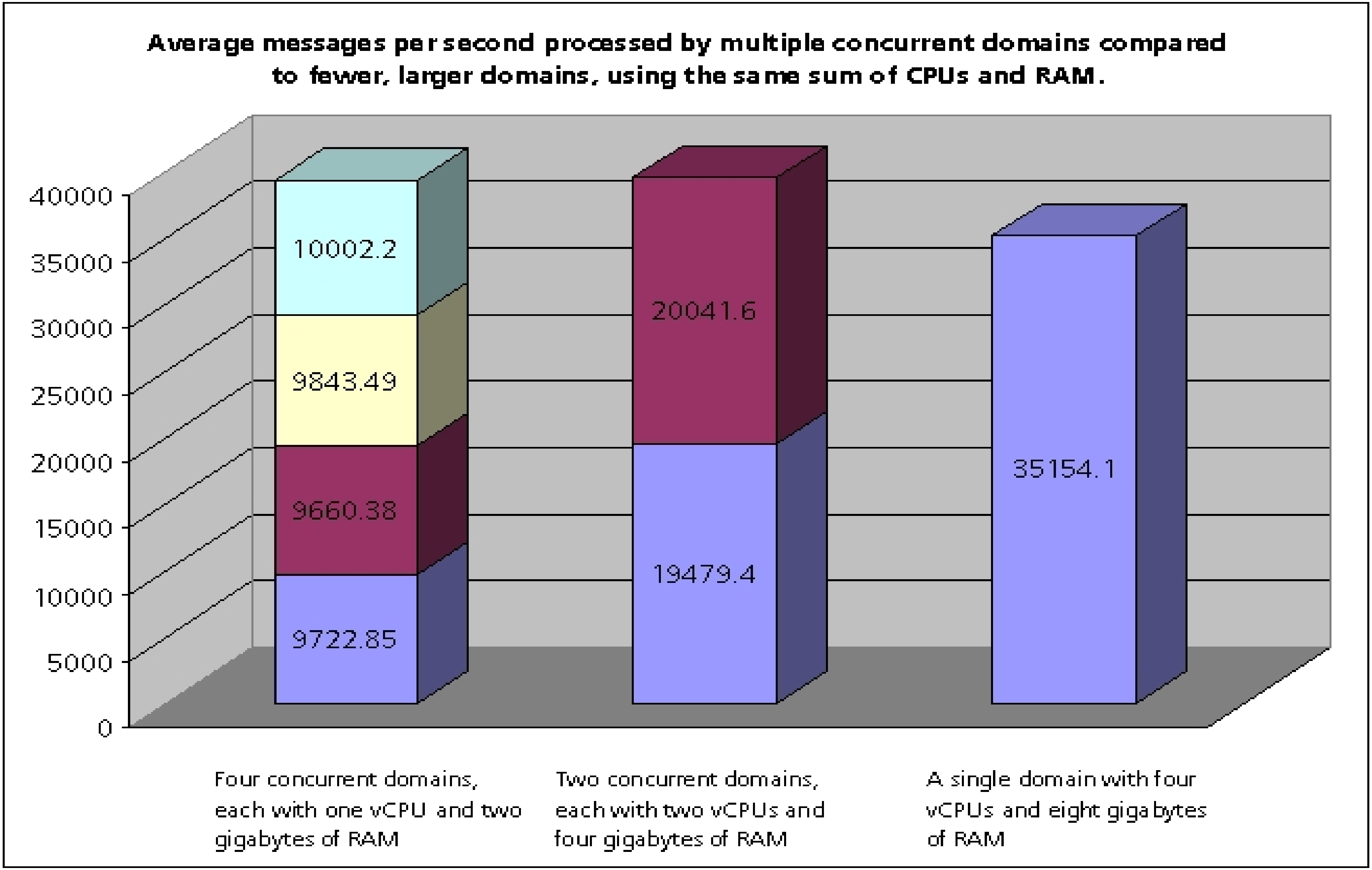
Red Hat Virt Multi-guest Efficiency

Efficiency of Multiple XenU 1,2,4 on RHEL5 GA 2.6.18-8

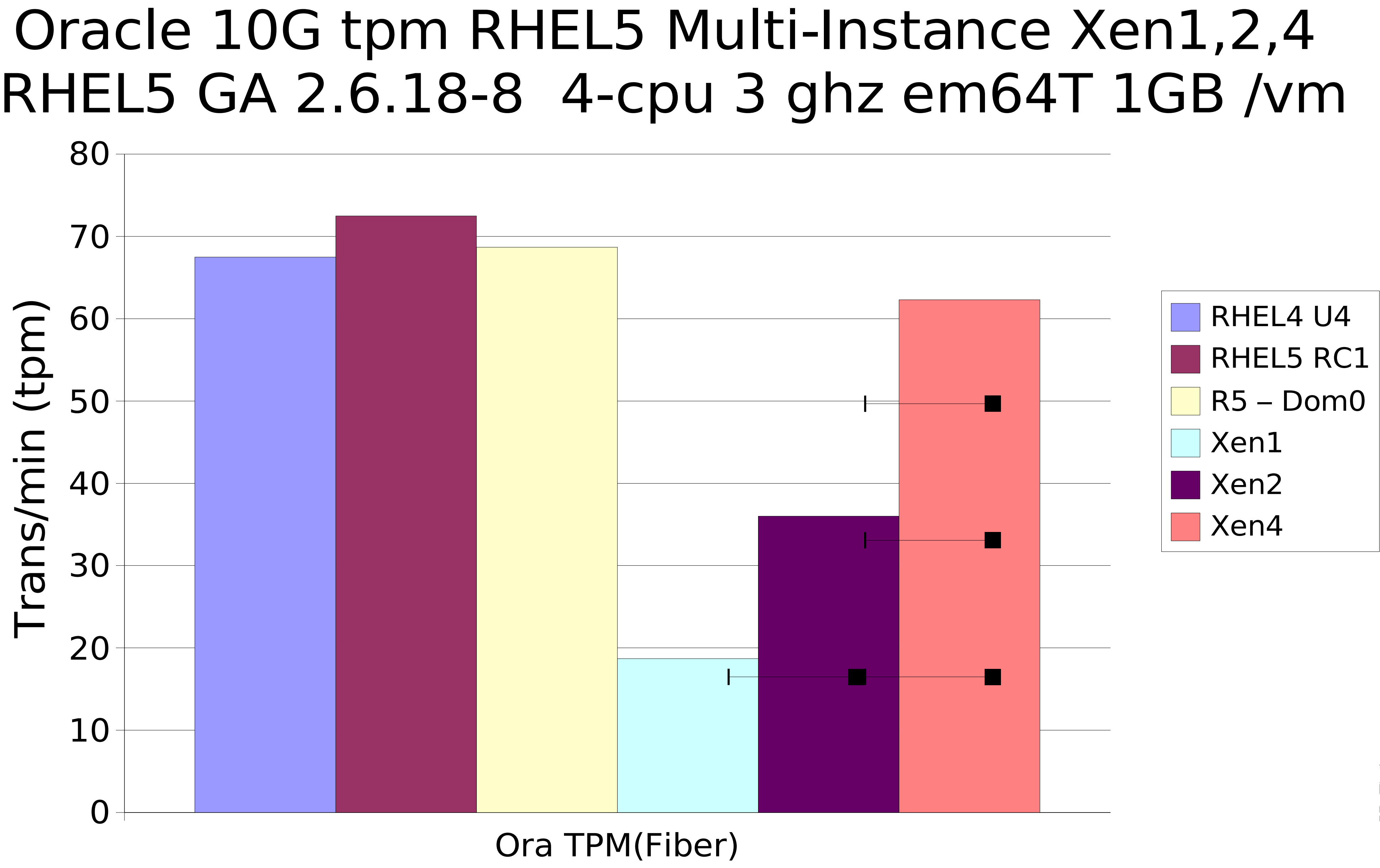
3.2 Ghz em64T 4GB memory, 1GB/xen guest



Red Hat Virt Java Multiguest Performance



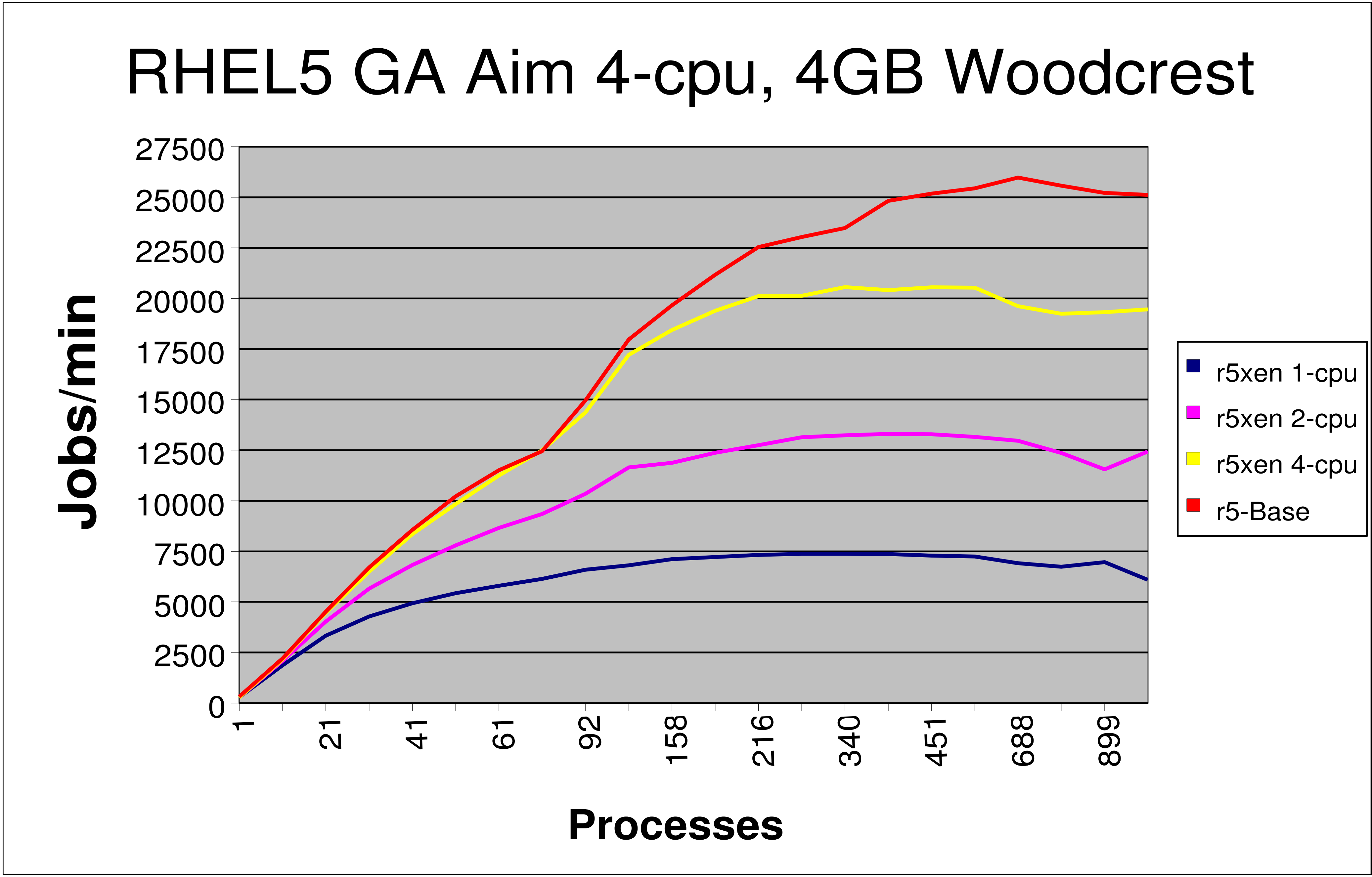
Red Hat Virt Multiple Guest Performance



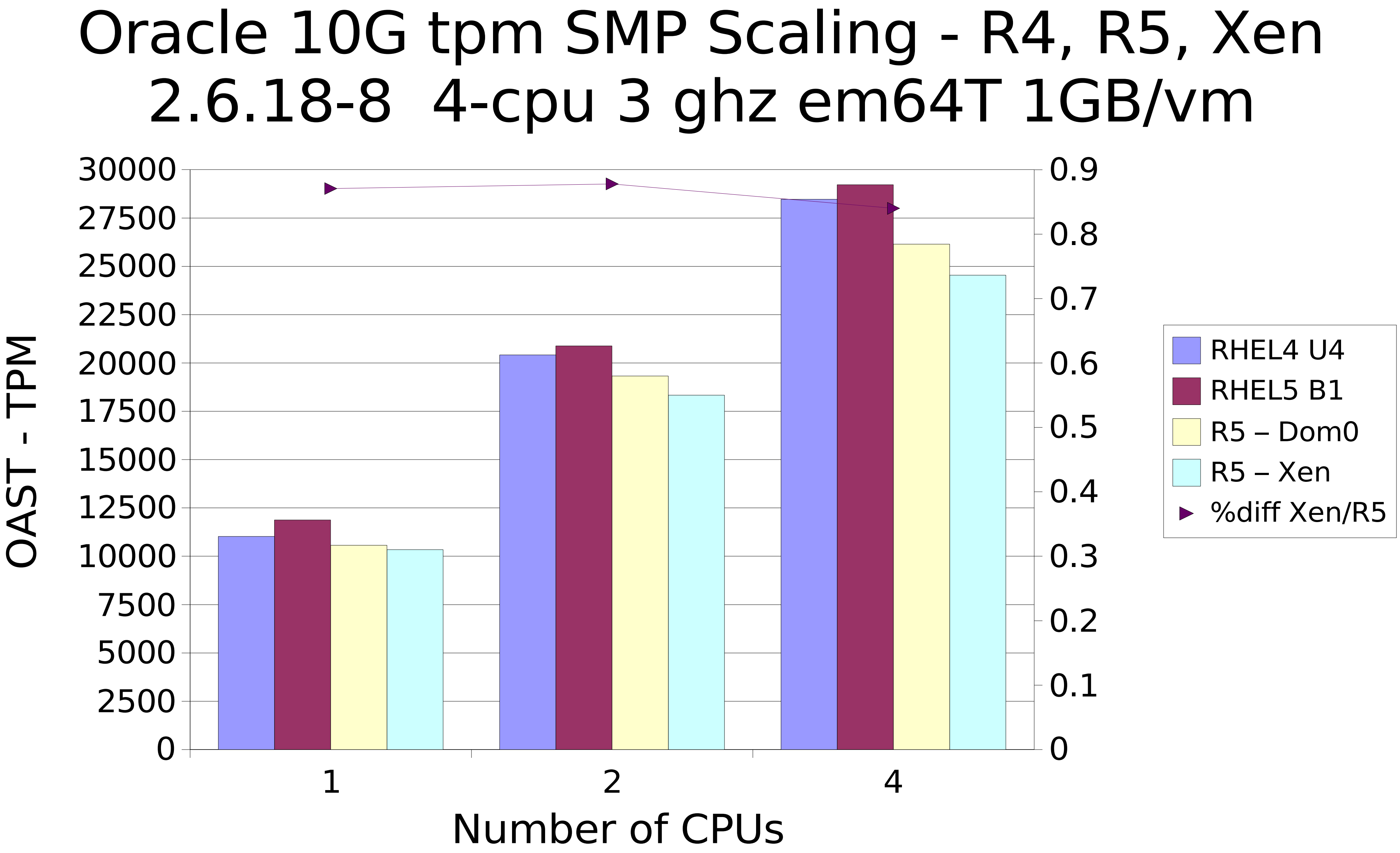
SMP in Guest OSes

- Takes great care to get good performance while remaining secure
- Paravirtualized approach yields many important benefits
  - Avoids many virtual IPIs
  - Enables ‘b ad preemption’ avoidance
  - Auto hot plug/unplug of CPUs
- SMP scheduling (at hypervisor)
  - Strict gang scheduling not optimal
  - Credit Scheduler in Xen 3.0.2 (RHEL5)

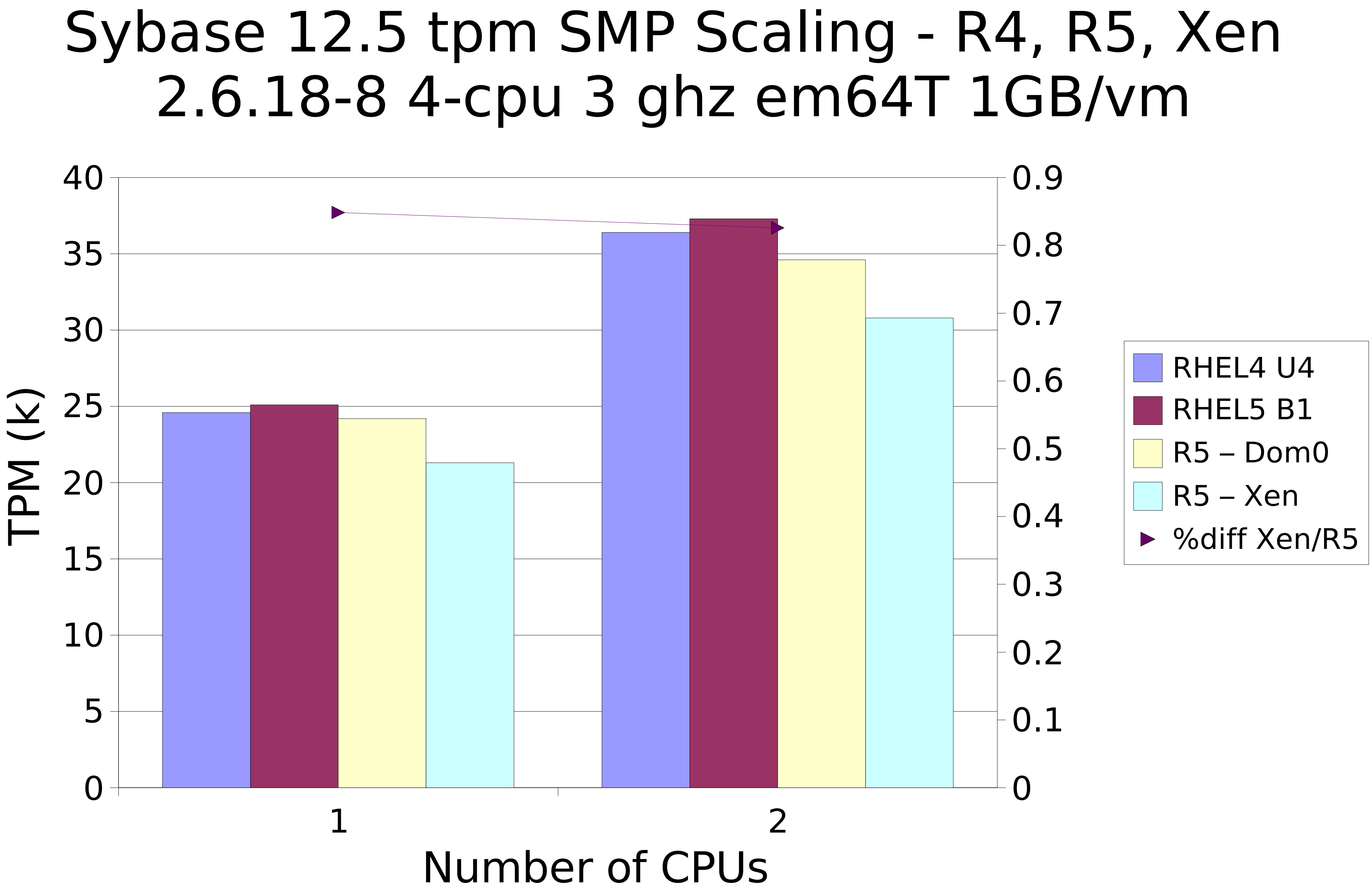
Red Hat Virt SMP guest Performance



Red Hat Virt SMP Performance

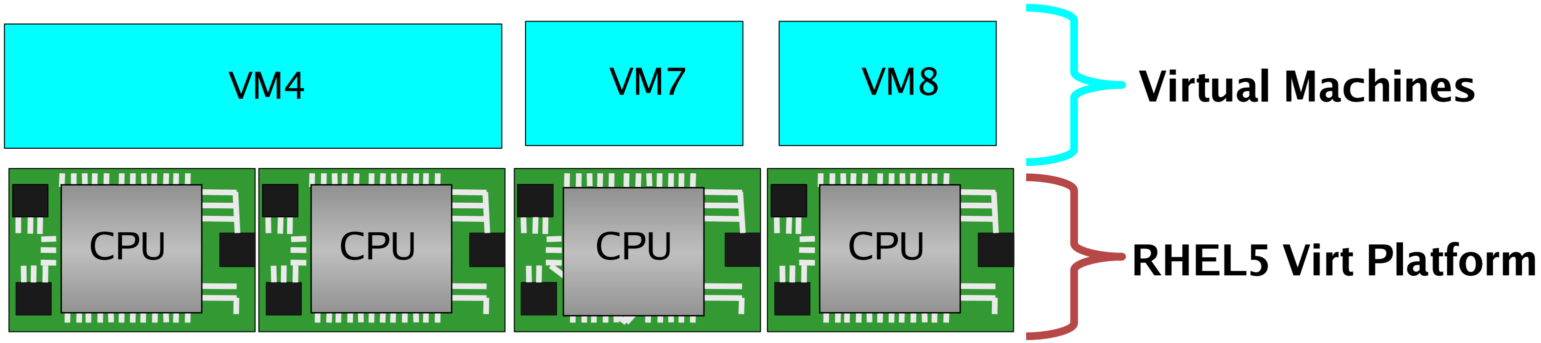


Red Hat Virt SMP guest Performance



Virtual SMP combined with sub-CPU granularity

All available in one offering on RHEL5

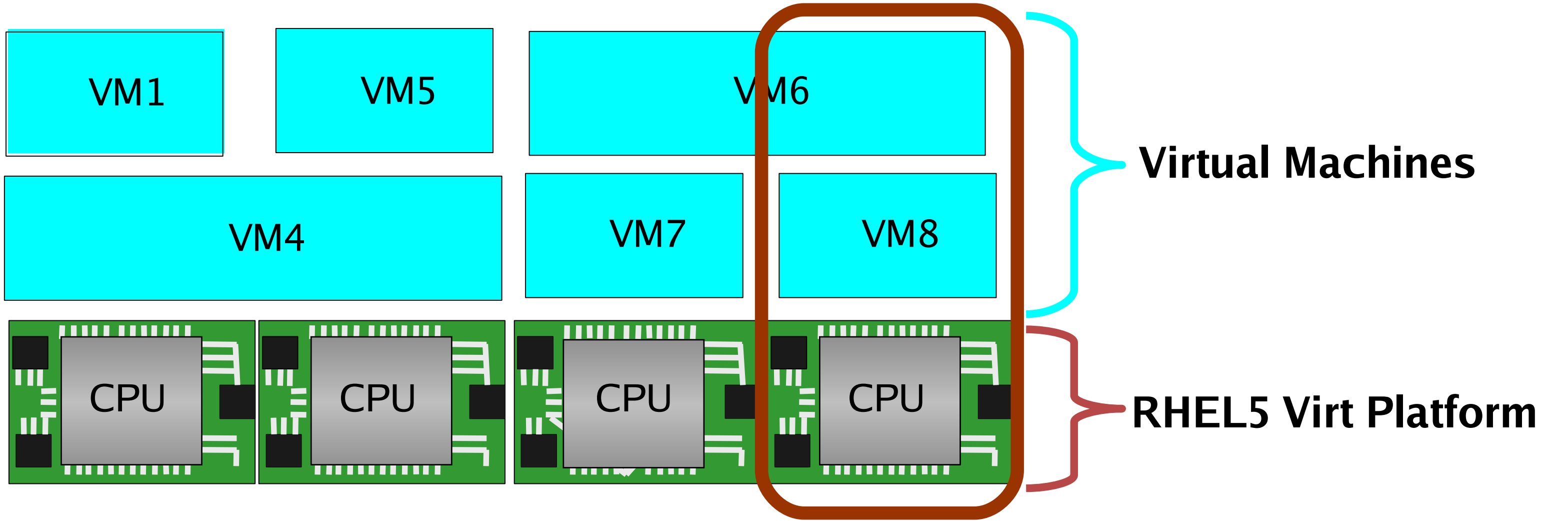


VMn == domUn

Virtual SMP combined with sub-CPU granularity

All available in one offering on RHEL5

Higher resource utilization



VM1

VM5

VM6

VM4

VM7

VM8

CPU

CPU

CPU

CPU

Virtual Machines

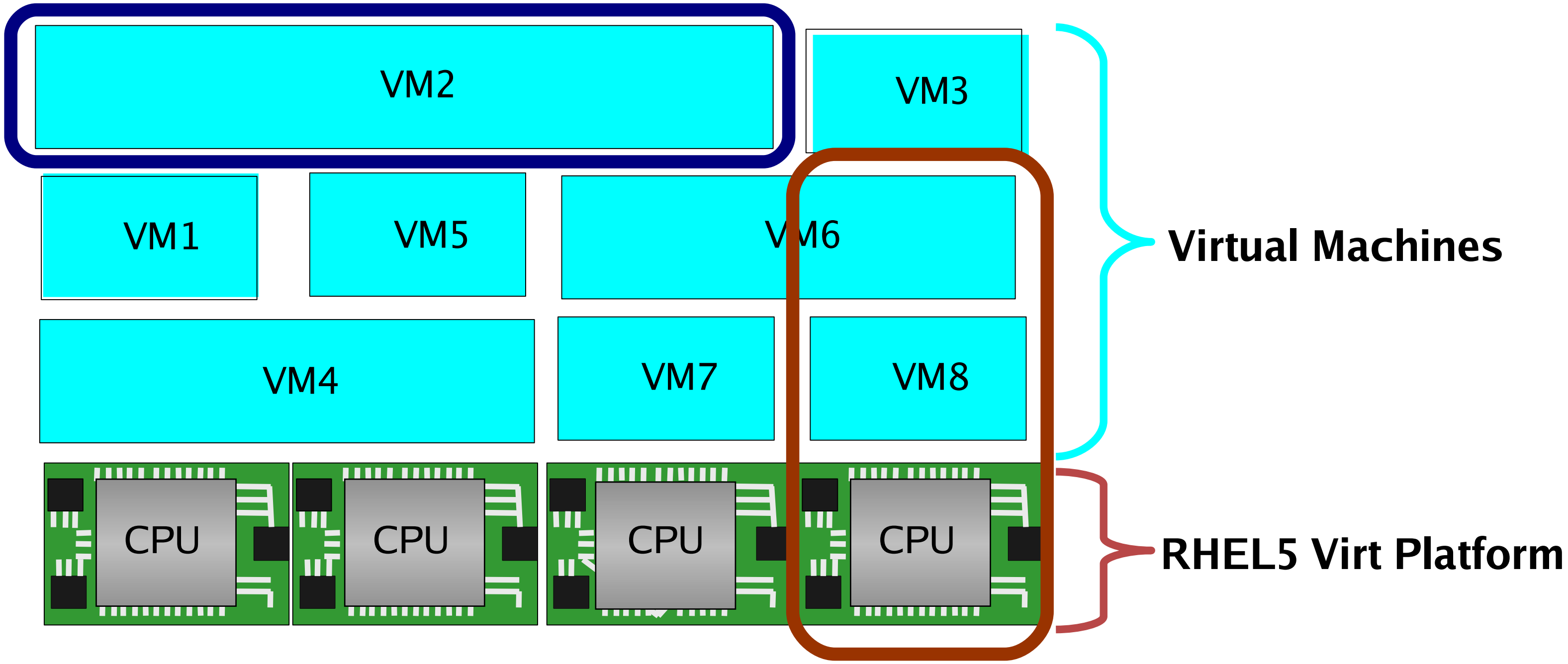
RHEL5 Virt Platform

VMn == domUn

Virtual SMP combined with sub-CPU granularity

All available in one offering on RHEL5

Virtual machine scalability and Higher resource utilization



VMn == domUn

Xen CPU Schedulers : New Default

sched-credit Set or get credit scheduler parameters

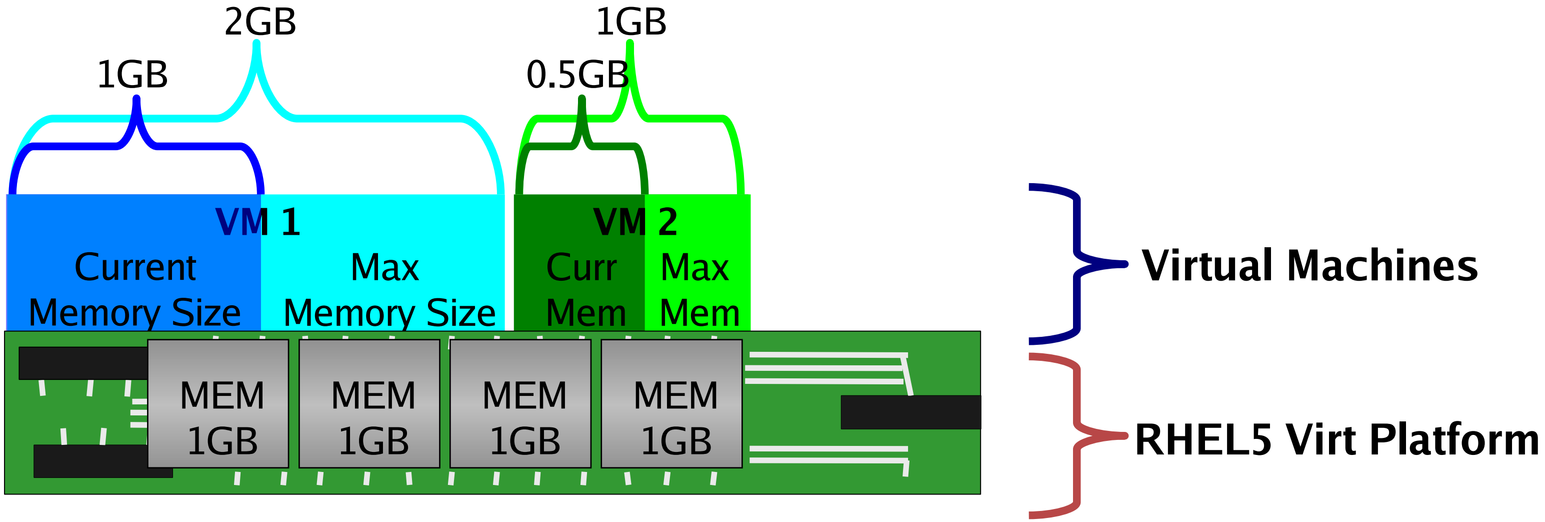
xm csched -d <domain> lists weight and cap
xm csched -d <domain> -w <weight> [256] sets the weight 1..65536
xm csched -d <domain> -c <cap> [0] sets the cap%

IO Elevators

Standard FC5/FC6 elevators
CFQ completely fair queing – Default – known -0-25% regression
Deadline – sensitive to I/O latency – best I/O intensive – 0-3% of R4.
NOOP – no attempt to balance I/O at kernel level
AS – anticipatory elevator – best interactive perf, insert delays in I/O

Memory ballooning

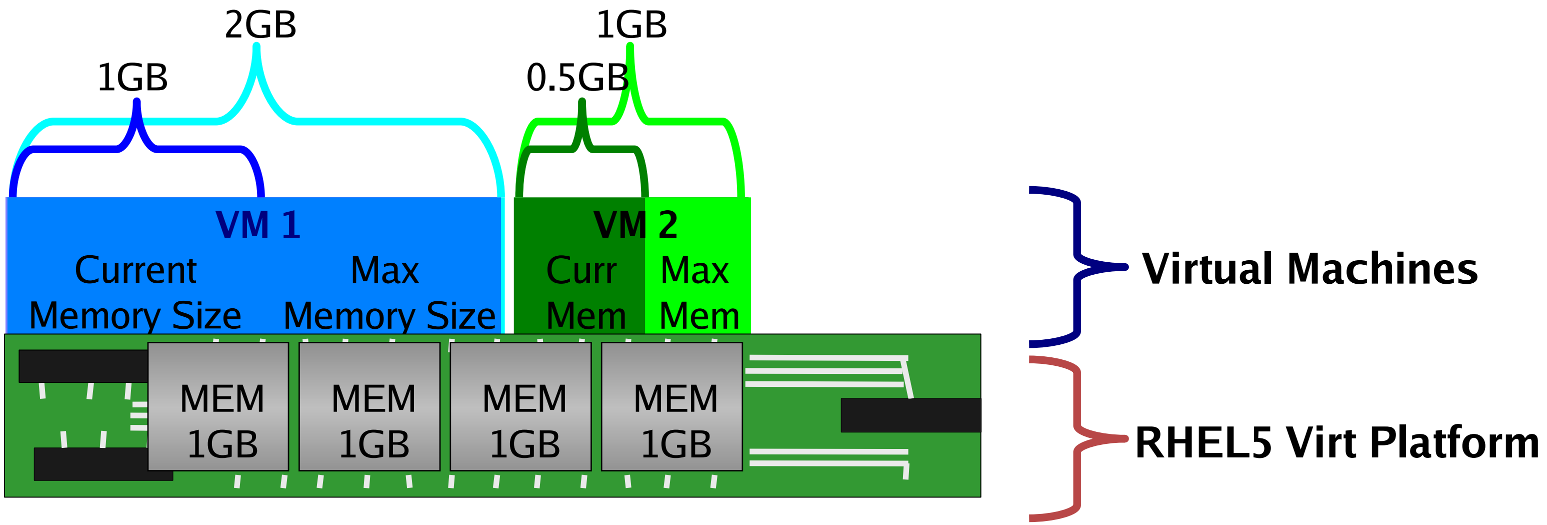
- Guest can be configured to balloon/growtheir current memory footprint
- Allows for online expansion and growth
  - Can use virt-manager or CLI interface for management



VMn == domUn

Memory ballooning

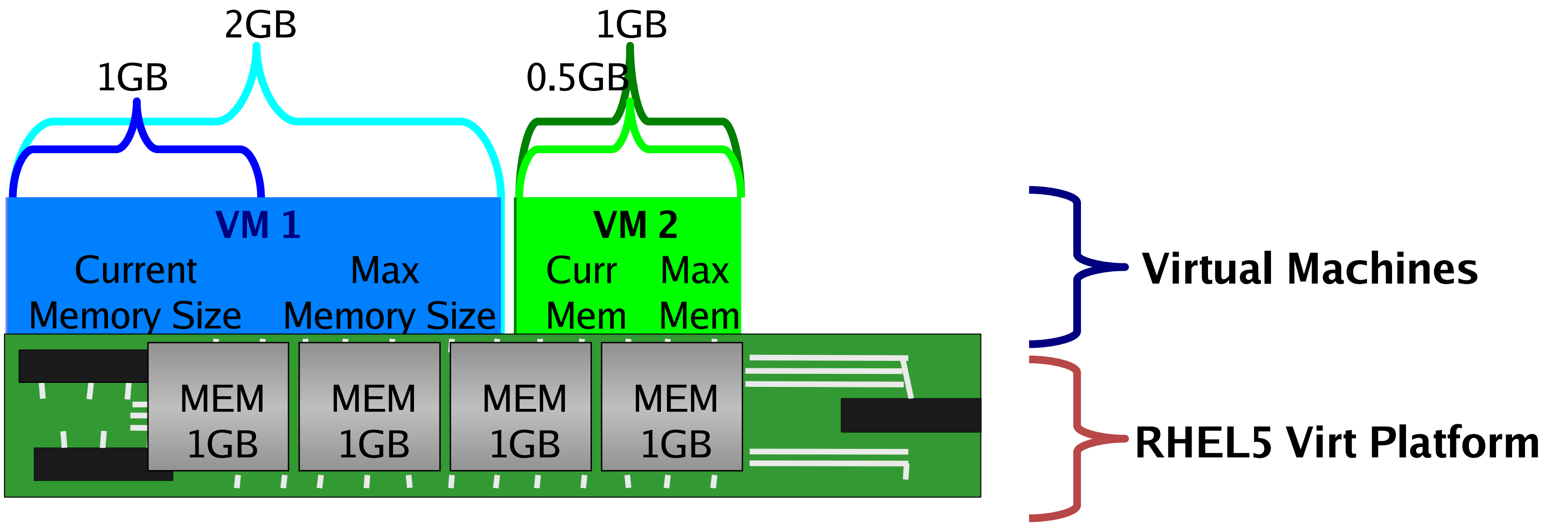
- Growing guest VM1 to 2GB using memory ballooning



VMn == domUn

Memory ballooning

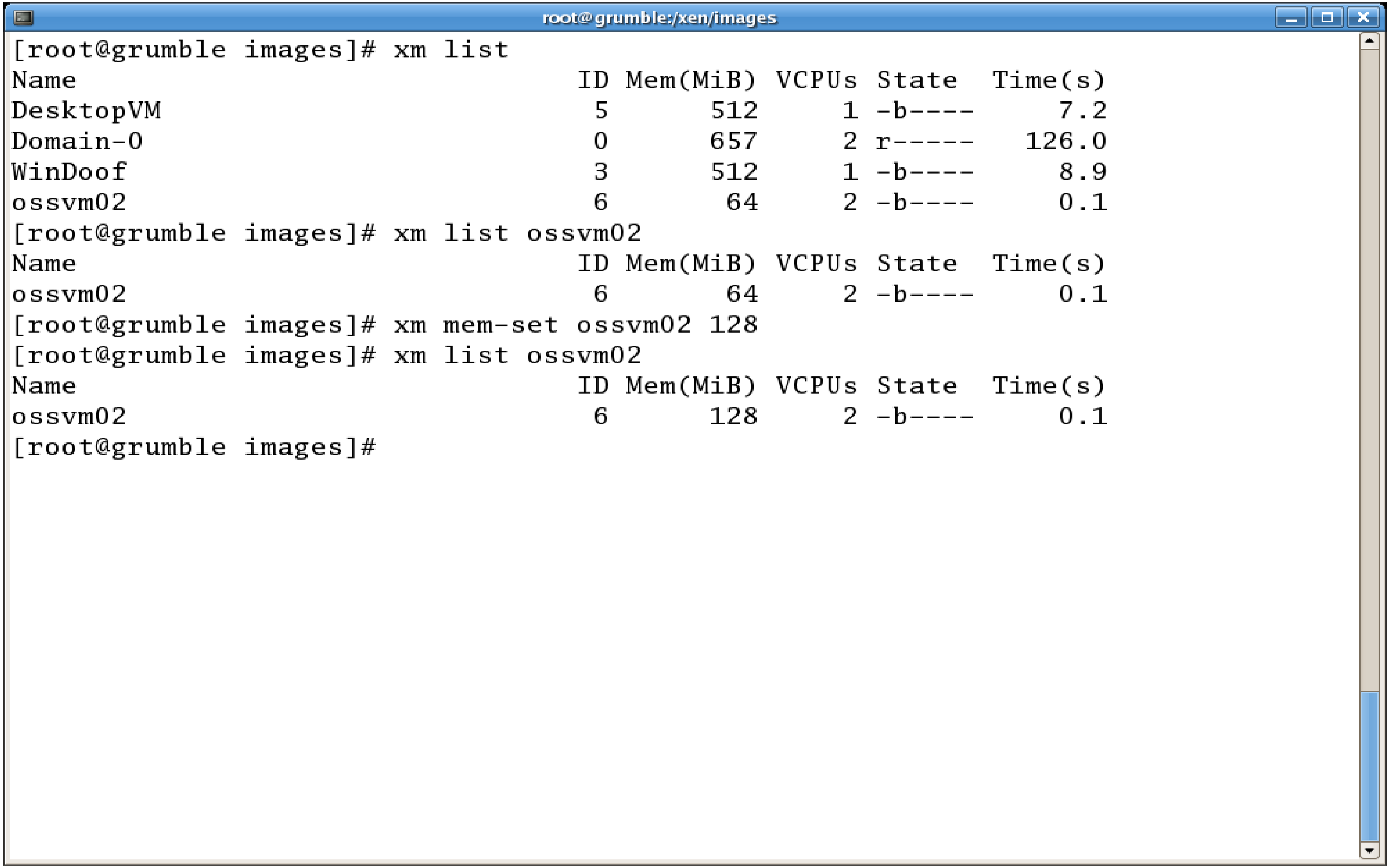
- Growing guest VM2 to 1GB using memory ballooning
- Now both guests have increased their available memory online
  - Resize database SGA
  - Increase available VM for applications etc...



VMn == domUn

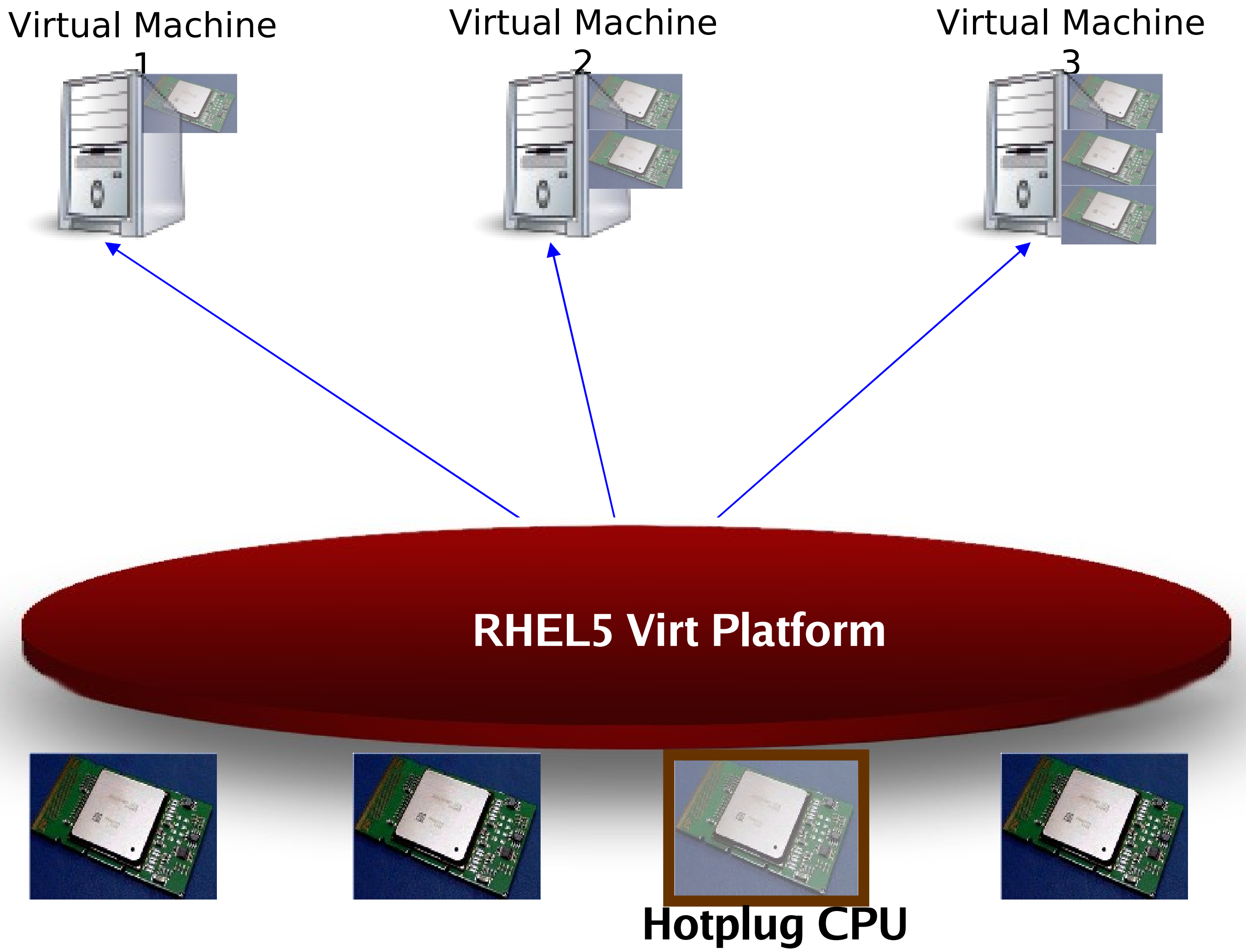
xm mem-set

- Ballooning of guest memory footprint



```
[root@grumble images]# xm list
Name                                      ID Mem(MiB) VCPUs State   Time(s)
DesktopVM                                  5      512     1 -b----      7.2
Domain-0                                   0      657     2 r-----    126.0
WinDoof                                    3      512     1 -b----      8.9
ossvm02                                    6       64     2 -b----      0.1
[root@grumble images]# xm list ossvm02
Name                                      ID Mem(MiB) VCPUs State   Time(s)
ossvm02                                    6       64     2 -b----      0.1
[root@grumble images]# xm mem-set ossvm02 128
[root@grumble images]# xm list ossvm02
Name                                      ID Mem(MiB) VCPUs State   Time(s)
ossvm02                                    6      128     2 -b----      0.1
[root@grumble images]#
```

Dynamic CPU allocations.. ub-CPU granularity



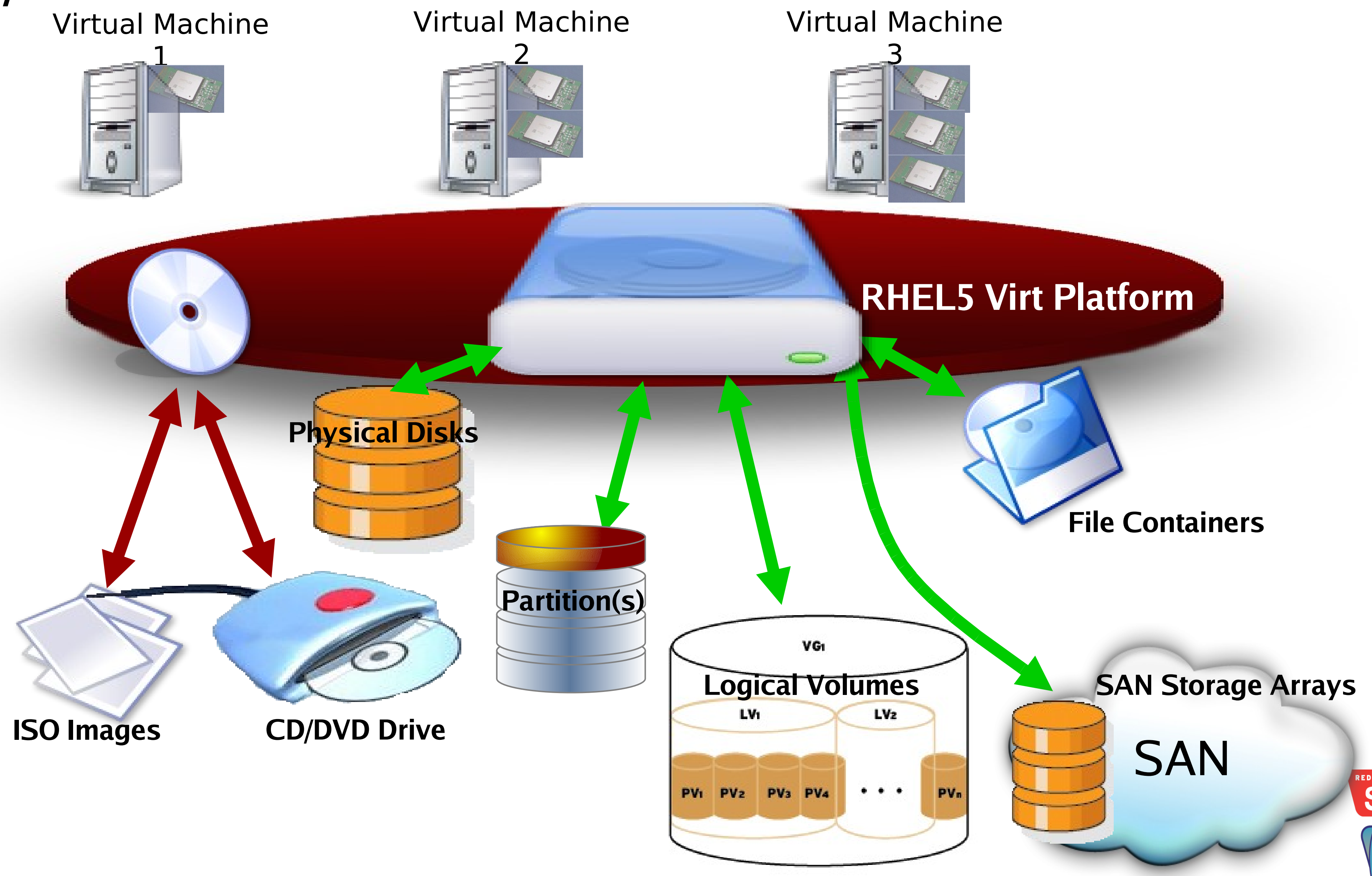
- CPU cycles dynamically allocated to guests as needed
- When oversubscribed (more demand than resources), fair share allocation to active VMs

  pport for steal time

- Guests can leverage hotplug CPUs

- Fine grain resource allocation guarantees with credit scheduler

I/O virtualization



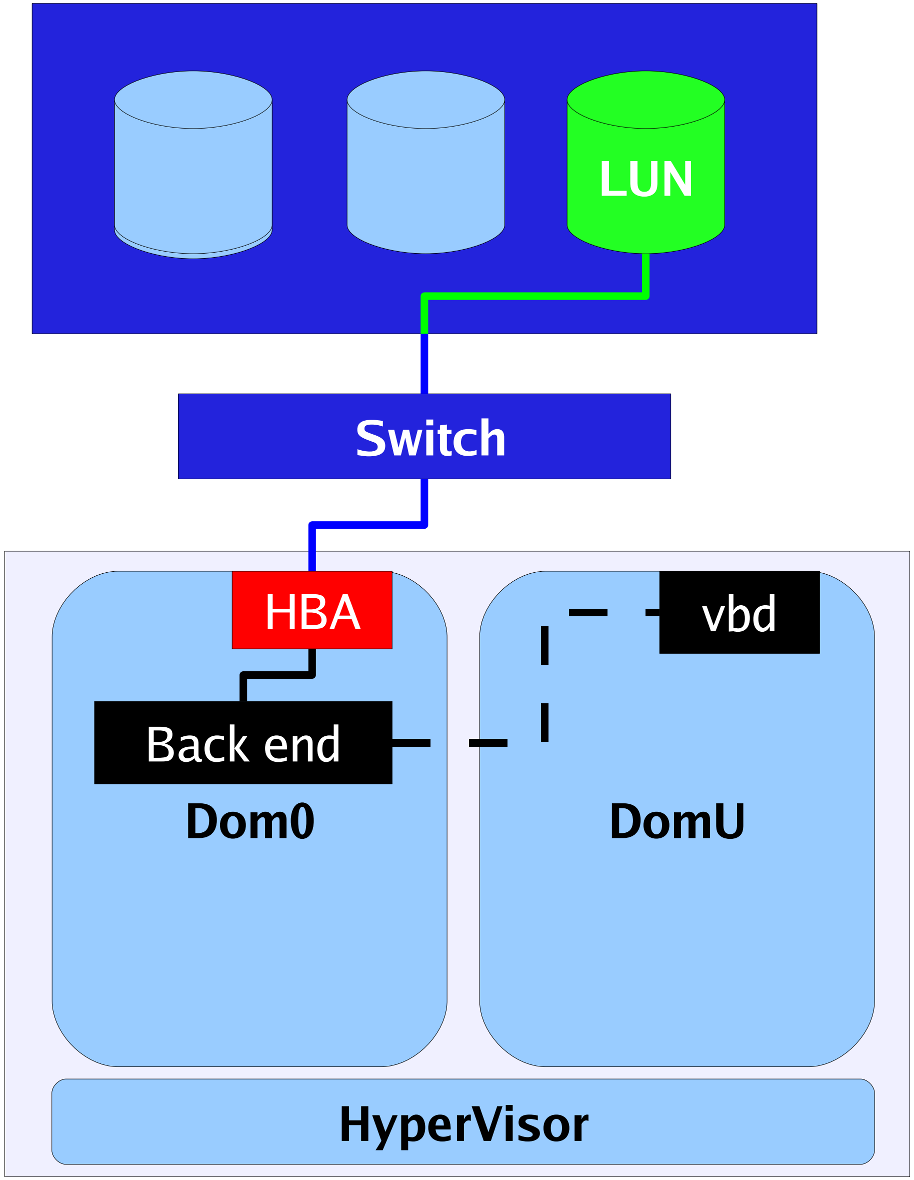
Storage Architecture

- Xen Storage
  - Physical block device
    - Local partitions
    - Logical volumes
    - LUNs on a SAN
  - Virtual block device
    - Files
      - Disk images
      - Filesystem image
      - ISOs
  - Network storage protocols
    - NFS, CIFS, GFS, iSCSI, GNBD, etc

Storage Architecture

- Xen Storage
  - In domU the “frontend”is blkfron t
  - domU does not know how storage is provided
    - File, physical partition, lvm etc
  - Standard naming is xvdX for devices
    - eg. /dev/xvda, /dev/xvdb
  - Blkfront is a generic block device
    - domU doesn't see it as IDE or SCSI
  - Backend devices can be migrated
    - eg. physical to file based
    - Guest will be unaware of the change

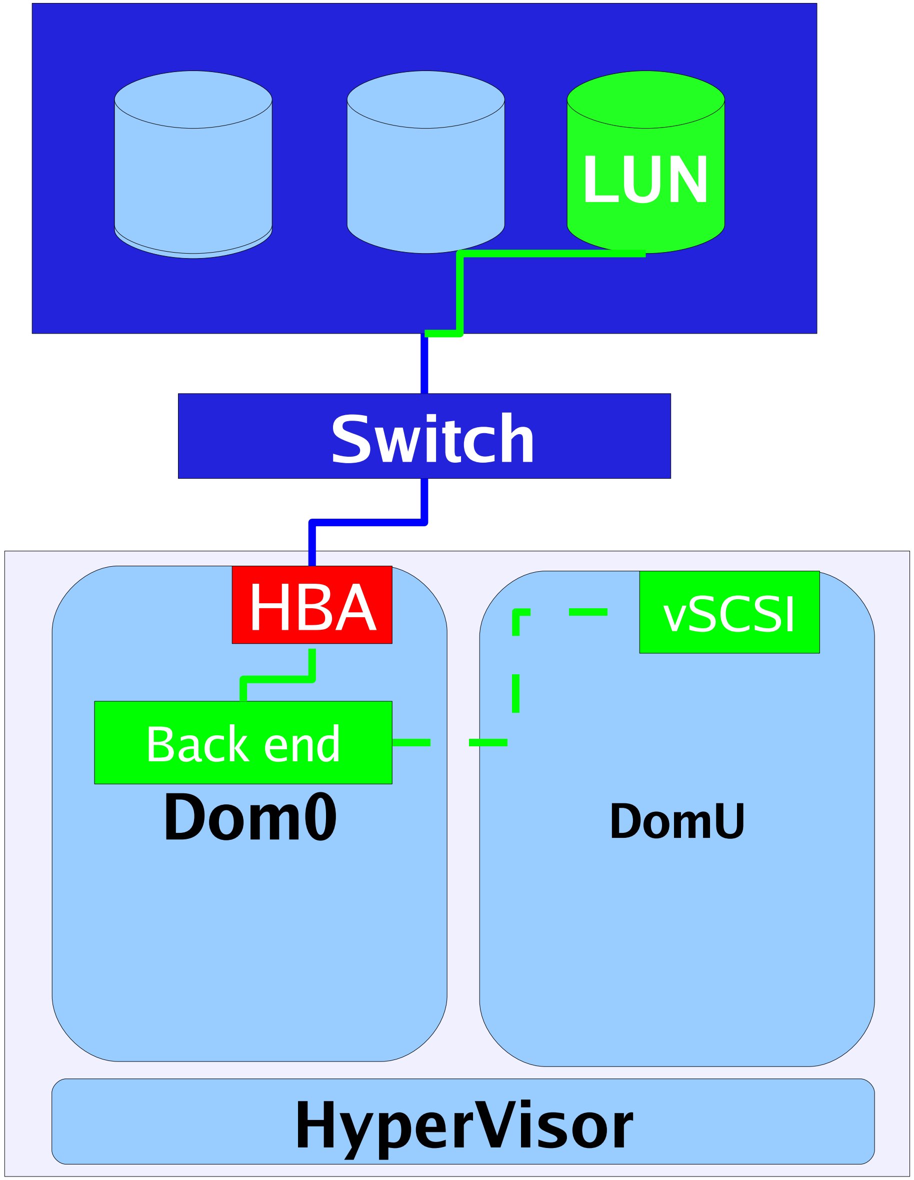
Storage: Underlying SAN architecture



Basic configuration

- Dom0 contains all hardware drivers as usual: supports all normal dom0 devices
  - FC, iSCSI, Infiniband etc.
- Unstructured virtual block device exported to domU guest via hypervisor
- All policy for binding

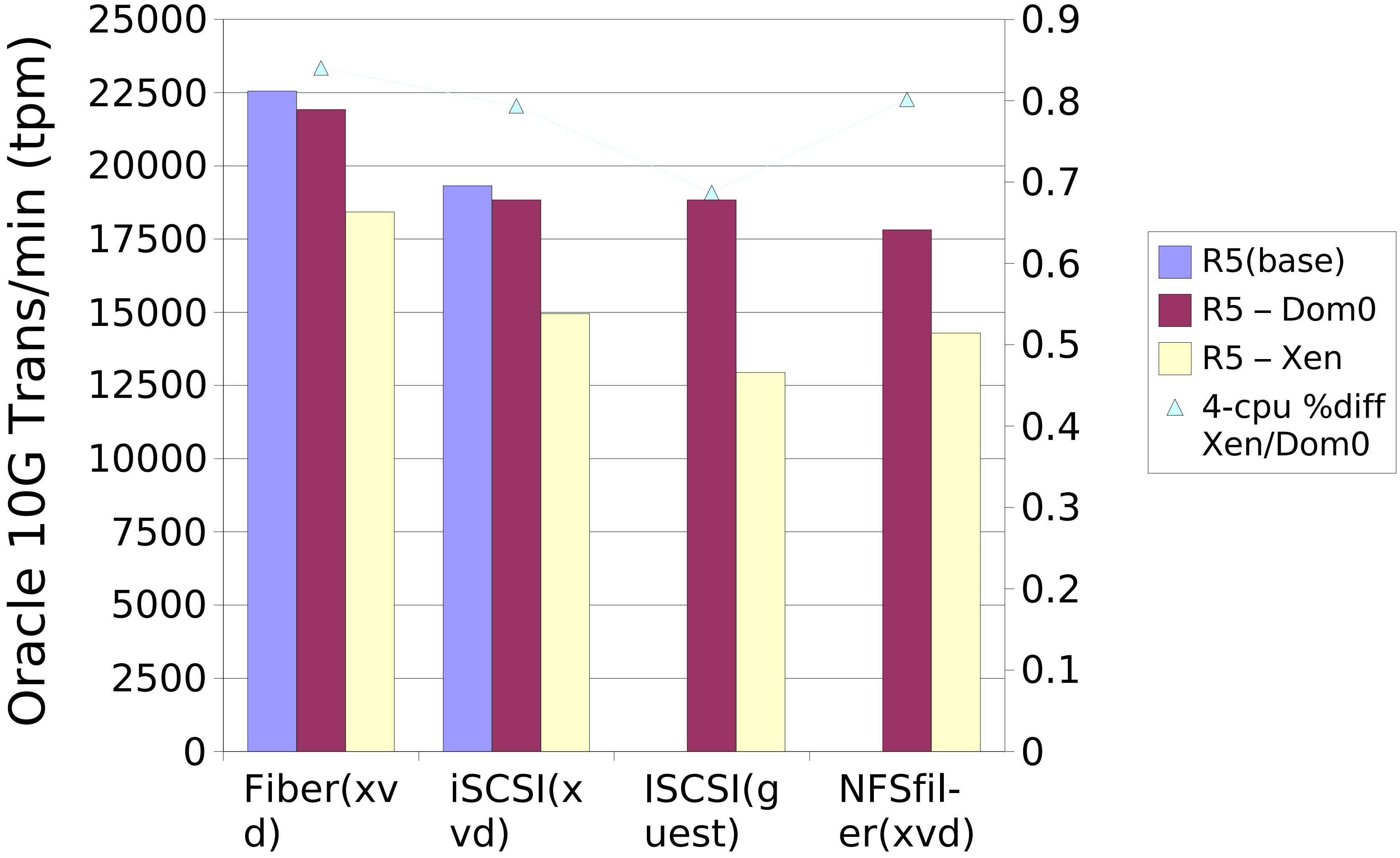
Storage: Virtual extension



LUN proxying

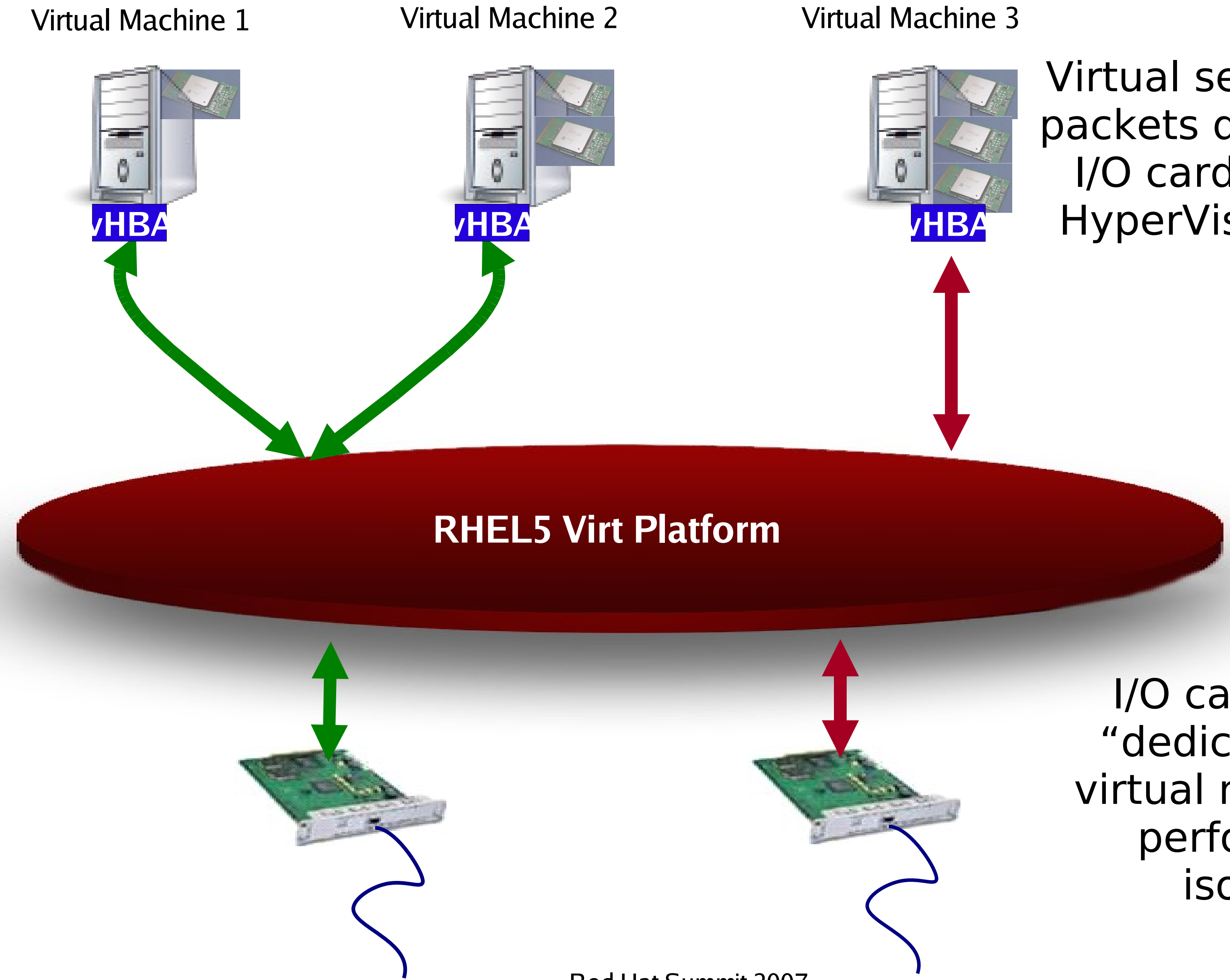
- Dom0 still contains hardware drivers
- Structured SCSI device exported to domU
- Device ID/enumeration works from domU
- No SAN management from domU
- Still no SAN filtering per guest

Red Hat Virt Storage Alternatives



Red Hat Company Confidential

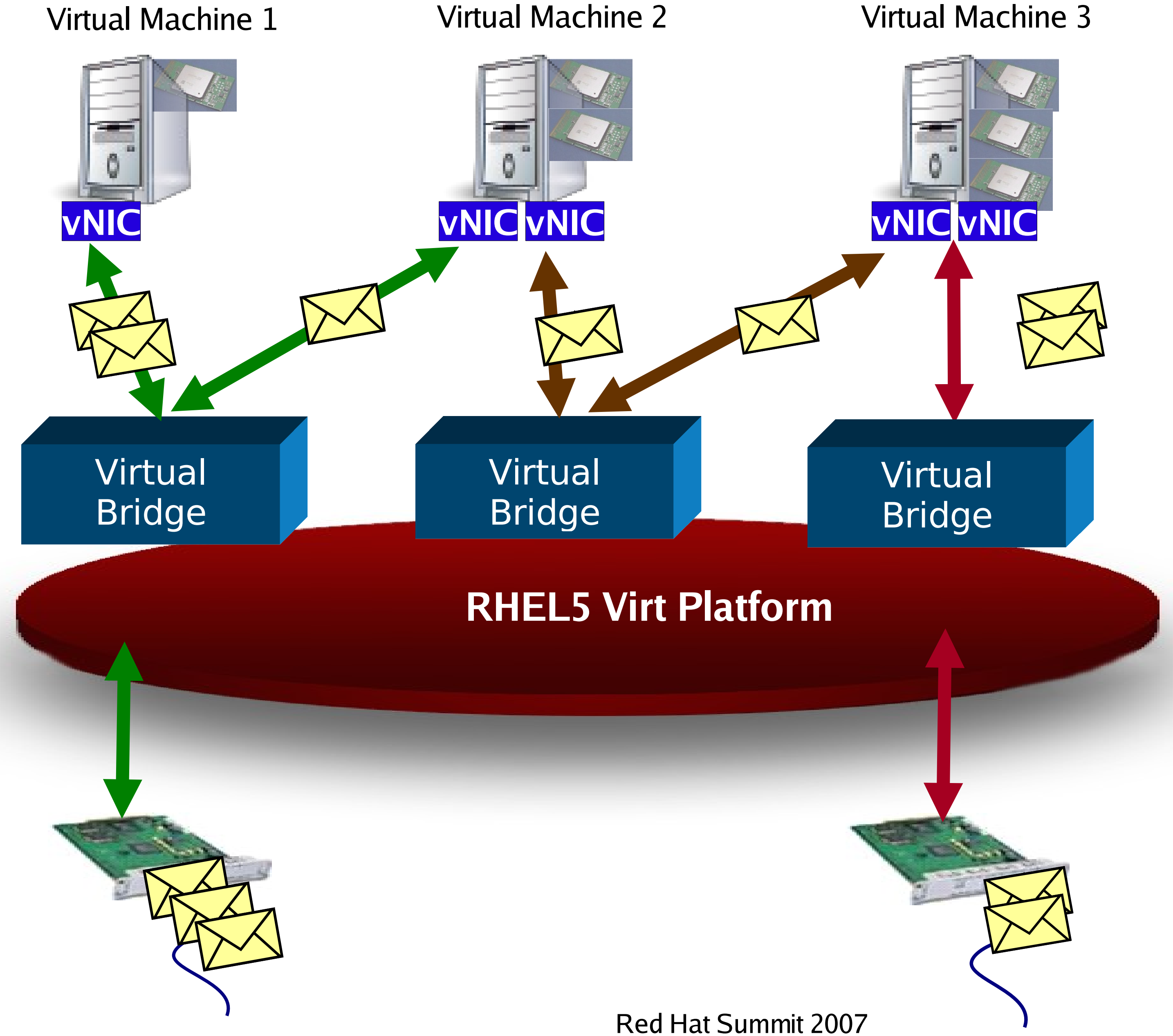
Dynamic I/O Sharing



Virtual server’s I/O packets directed to I/O cards by the HyperVisor/dom0

I/O card can be “dedicated” to a virtual machine for performance isolation

Dynamic Network I/O Sharing



Virtual machineś network packets directed to physical NIC by the HyperVisor/dom0

Virtual NIC may be defined without a physical NIC for guest-to-guest communication

NIC can be “d edicated” to a virtual machine for performance isolation

Networking

- Xen supports a modular network architecture
- Different “network m odels”can be configured
  - Bridging
    - Create a virtual bridge device
    - Bridges the real physical and virtual interfaces
    - Allows guest to 'share' physical network card
  - NAT
    - Uses Network address translation to provide masqueraded interface
  - Routed
    - Creates virtual nic. Uses routing to map to external device

Networking

- Currently NAT and Routed are not working
  - Broken up stream
  - Issues with iptables rules and configuration
  - Work-arounds possible –not covered today

- Default network type is bridging
  - Just Works ™ for wired network connections
  - Inconsistent results with wireless networks
    - Some wireless nics prevent multiple mac address
    - Work around with ebtables (in FC extras)
      - Not covered today

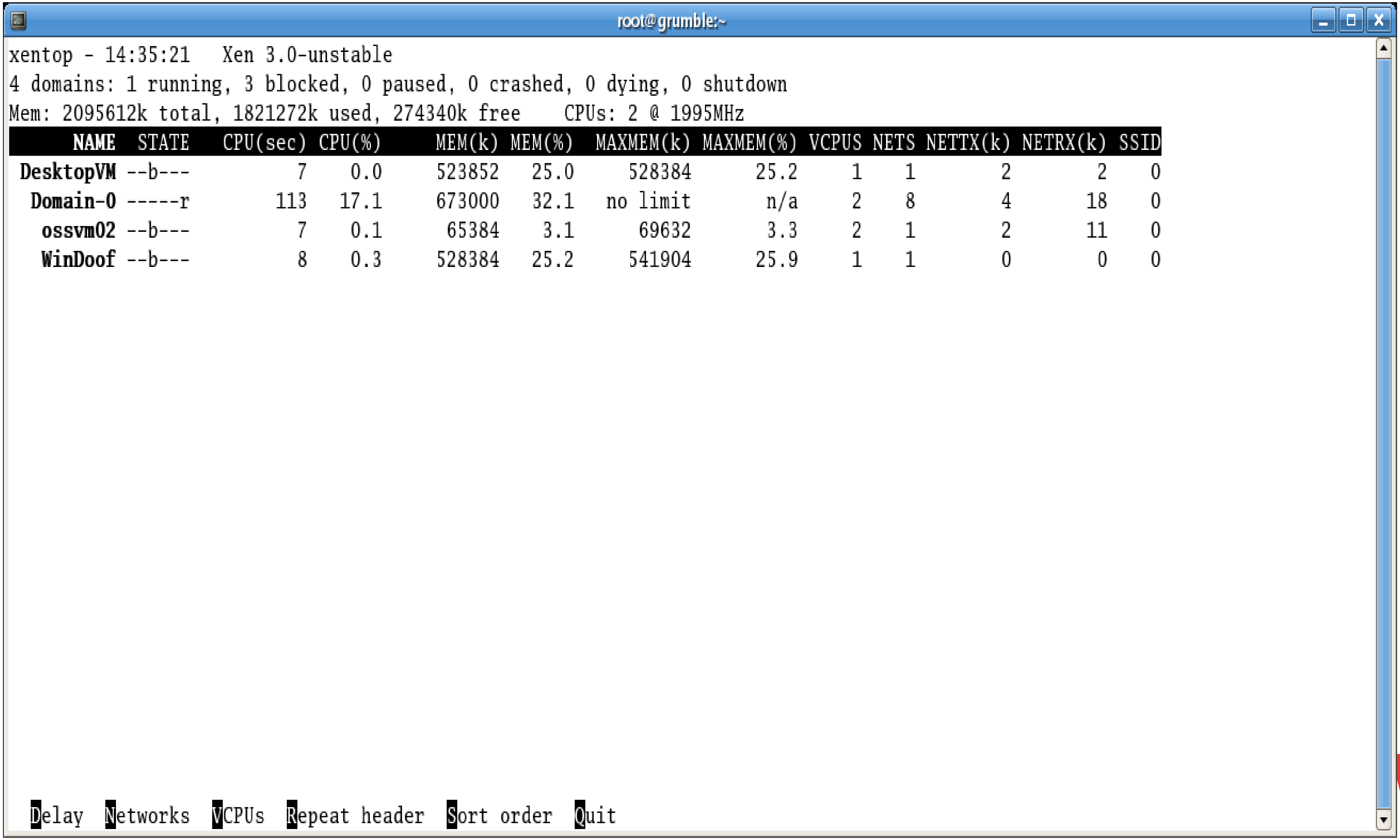
Networking

- Challenge :
  - Laptop Configuration
    - Bridging does not work (effectively) with laptops
    - Need consistent demo environment
      - For wired networks
      - For wireless networors
      - When working offline -on a plane
      - Work on vpn

Networking

- Solution :
  - Use dummy network driver
    - Requirement :
      - Must use static IP's in domU
      - DHCP server isn't available offline
      - Cannot run dhcp server on dummy interface

xm top



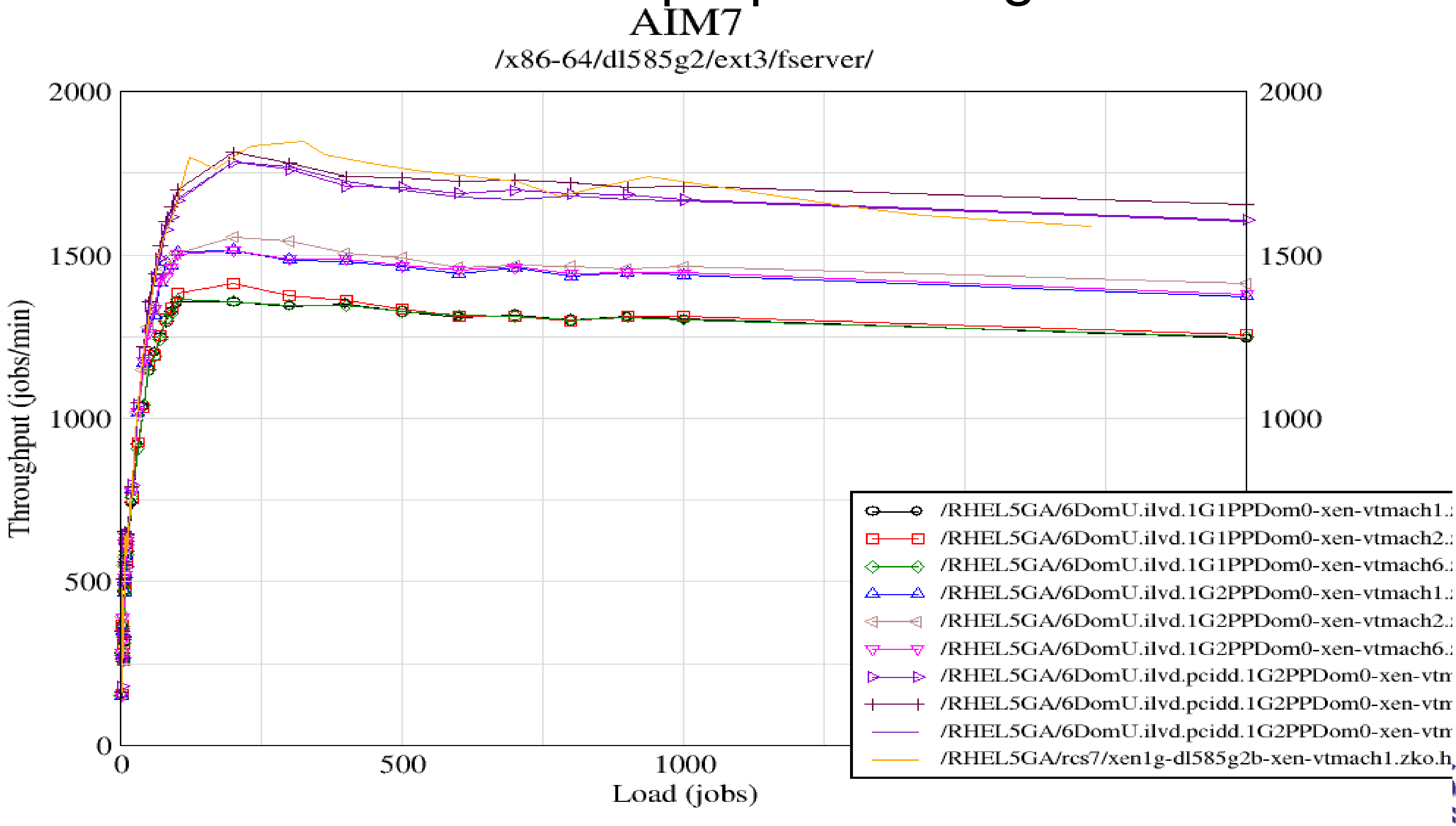
```
xentop - 14:35:21   Xen 3.0-unstable
4 domains: 1 running, 3 blocked, 0 paused, 0 crashed, 0 dying, 0 shutdown
Mem: 2095612k total, 1821272k used, 274340k free    CPUs: 2 @ 1995MHz
      NAME  STATE   CPU(sec) CPU(%)     MEM(k) MEM(%)  MAXMEM(k) MAXMEM(%) VCPUS NETS NETTX(k) NETRX(k) SSID
 DesktopVM --b---          7    0.0     523852   25.0     528384      25.2     1    1        2        2    0
  Domain-0 -----r        113   17.1     673000   32.1   no limit       n/a     2    8        4       18    0
   ossvm02 --b---          7    0.1      65384    3.1      69632       3.3     2    1        2       11    0
   WinDoof --b---          8    0.3     528384   25.2     541904      25.9     1    1        0        0    0



  Delay  Networks  VCPUs  Repeat header  Sort order  Quit
```

Red Hat Virt Performance

Enhance CPU Perf w/ Vcpu-pin Tuning



Configuration files

- Typical paravirtualized configuration file w/ external physical lun (sdb1 -> xvdb)

```
# Automatically generated xen config file
name = "rhel5pvtest"
memory = "500"
disk = ['tap:aio:/var/lib/xen/images/rhel5pvtest.dsk,xvda,w',
'phy:/dev/sdb1,xvdb,w',   ]
#
vif = [ 'mac=00:16:3e:48:fe:a6, bridge=xenbr0', ]
vfb = ["type=vnc,vncunused=1"]
uuid = "494e5923-9911-7144-5604-a0e9ad798021"
bootloader="/usr/bin/pygrub"
vcpus=1
on_reboot    = 'restart'
on_crash     = 'restart'
```

Configuration files (continued)

- Typical fully virtualized configuration file w/ external lun sdb1-> hdb

```
# Automatically generated xen config file
name = "rhel5fvtest"
builder = "hvm"
memory = "500"
```

- `disk = [ 'file:/var/lib/xen/images/rhel5fvtest.dsk,hda,w', 'phy:/dev/sdb1,hdb,w', ]`

```
vif = [ 'type=ioemu, mac=00:16:3e:78:aa:da, bridge=xenbr0', ]
uuid = "6329dc89-6d8a-a350-4163-0c806baaffd0"
device_model = "/usr/lib/xen/bin/qemu-dm"
kernel = "/usr/lib/xen/boot/hvmloader"
vnc=1
vncunused=1
apic=1
acpi=1
```

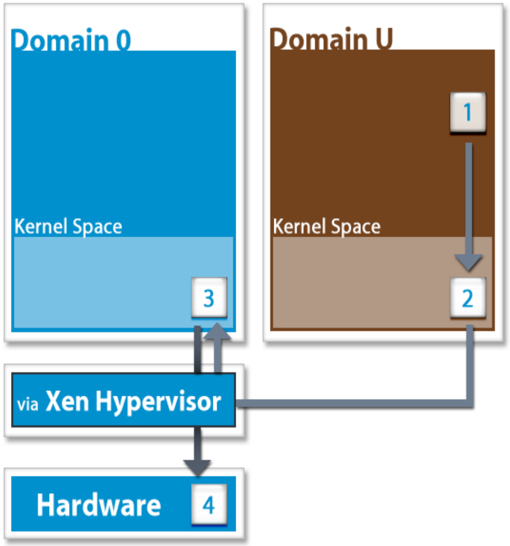
Adding Storage Dynamically to guest

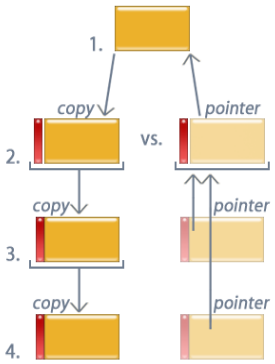
- Identify the block device or image file you want to make available to the virtual machine (for our example we use /dev/sdb1)
- After you've selected the storage you want to present to the guest you can use the “xm block-attach” command to assign it to your virtual machine
- The syntax for “xm block-attach” is
  - xm block-attach <domain> <backdev> <frontdev> <mode>

Device Driver Enhancements

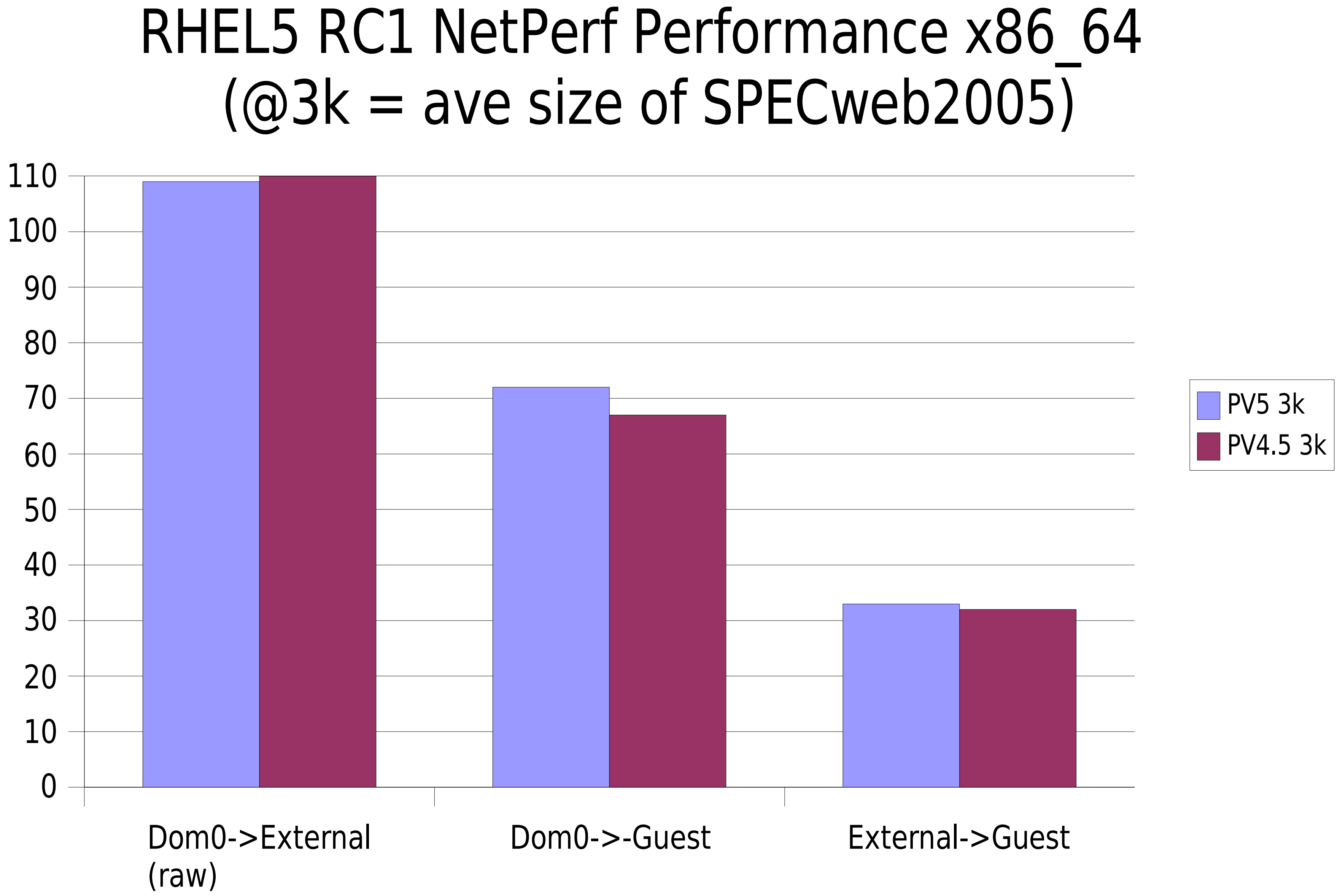
Network I/O

- Reduce number of memory copies required for network I/O
- Emulate h/w acceleration in virtual device drivers or h/w emulation
  - TSO (TCP Segment Offload)
  - TOE (TCP Offload Engine)
- Use real h/w acceleration from dom0
- Applies to both PV and FV paradigms
- Use hardware queuing to get network packets queued up in the ring buffer for each domU directly (Intel I/OAT)

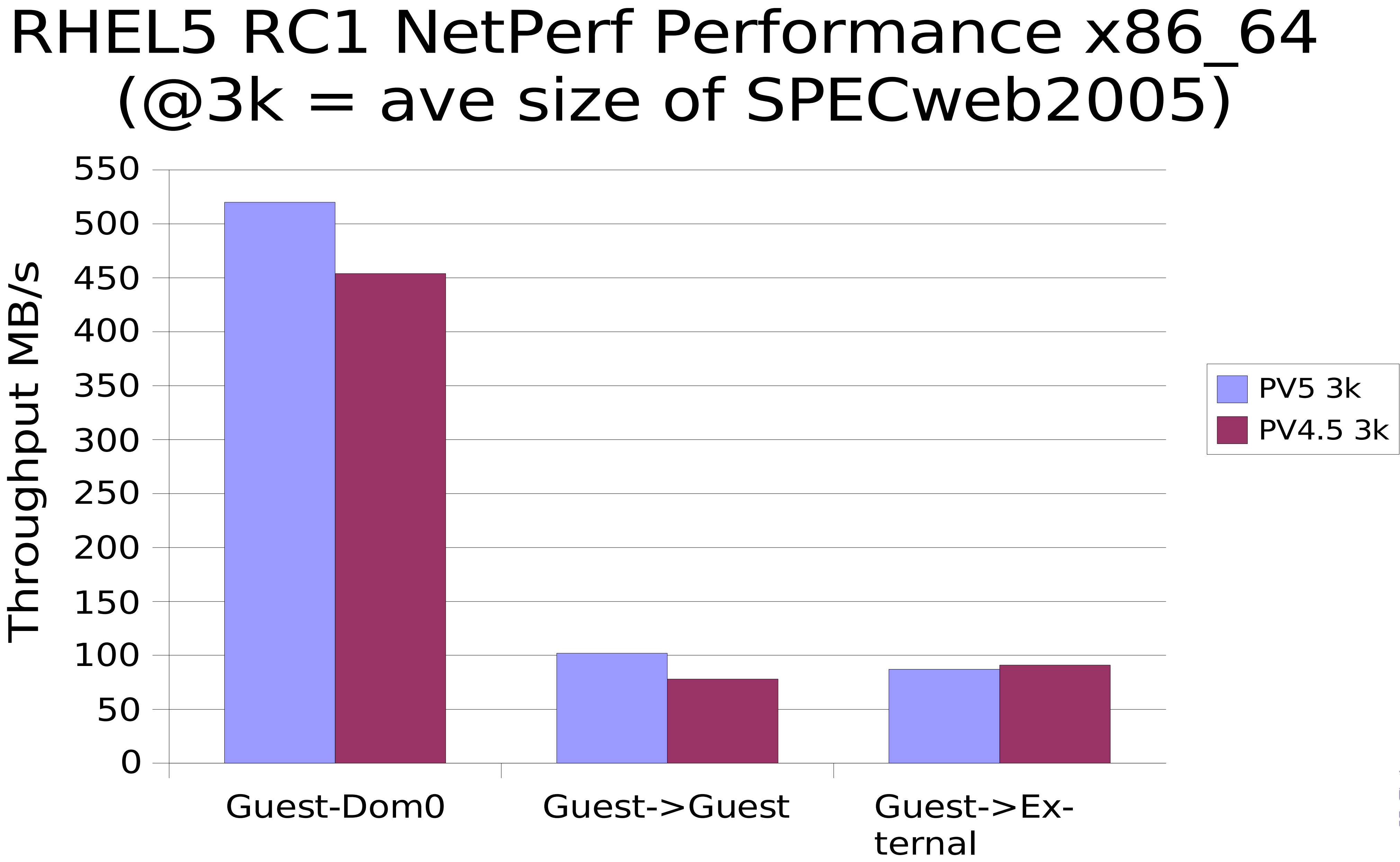




Red Hat Virt Network Performance



Red Hat Virt Network Performance



Red Hat Virt Network Performance

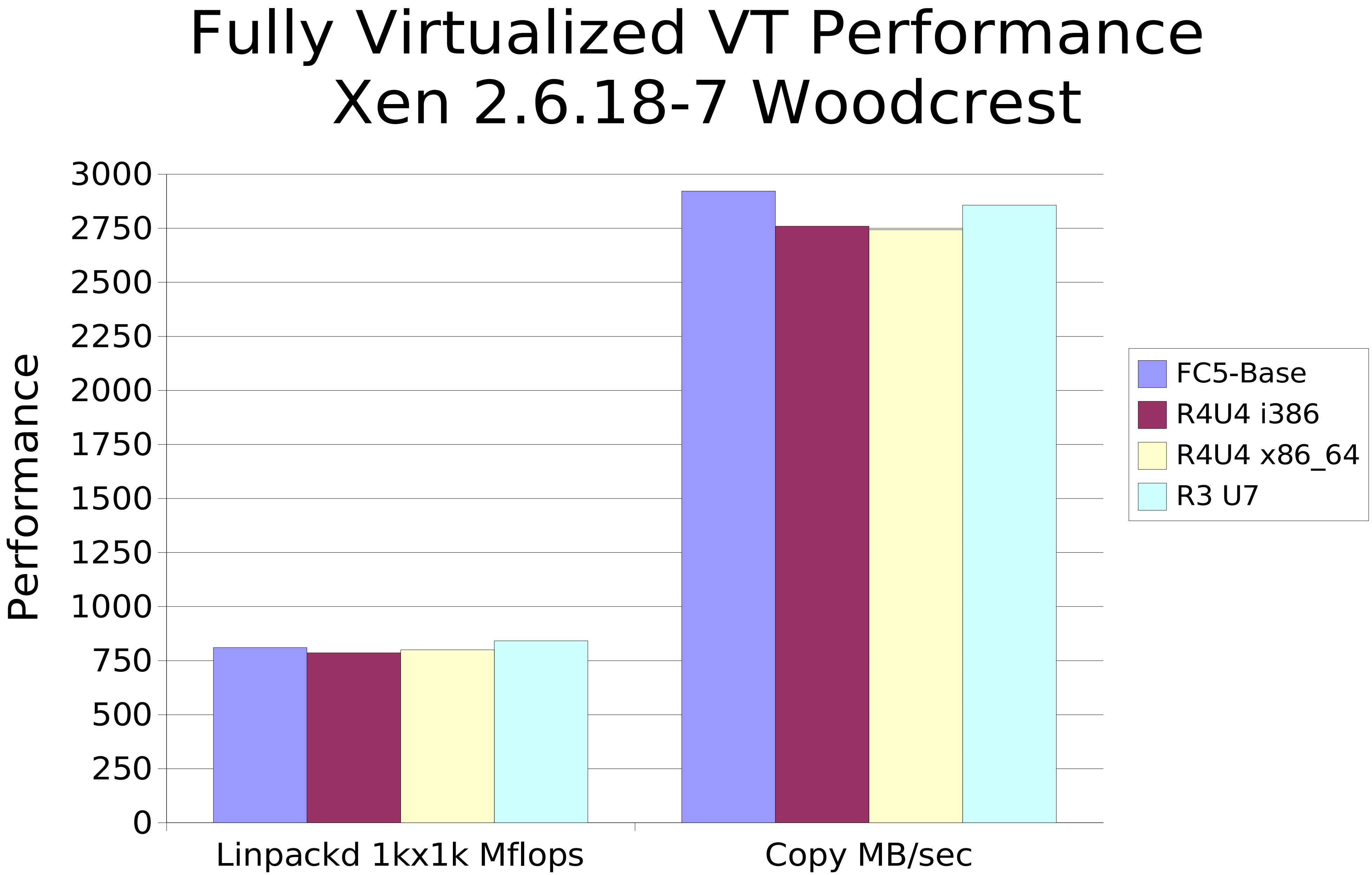
Limit guest using “rate=xxx”

- The r“ ate=” option can be added to the V“ IF=” entry in a virtual machine config file to limit a virtual machine's network bandwidth or to specify a specific granularity of credits during a specified time window
- The time window is optional to the 'rate=' option
  - The default time window is 50ms
  - A smaller replenishment /time window will make for slightly less bursty transmission, but there is an overhead as the replenishment rate is increased.
  - Actually the default 50ms is a good tradeoff and you probably don't need to change it
- Example use of the rate parameter are :
  - 'rate=10Mb/s'
    - Limit the outgoing network traffic from the guest to 10MB/s
- In the virtual machine configuration a sample VIF entry;
  - vif = ['rate=10MB/s , mac=00:16:3e:7a:55:1c, bridge=xenbr1']

VT-x (Intel) / Pacifica (AMD)

- Enables Guest OSes to be run without paravirtualization modifications
  - E.g. Windows XP/2003, non-pv guests (CentOS, RHEL3/4, etc..)
- CPU provides traps for certain privileged instrs
- Shadow page tables used to provide MMU virtualization
- Xen provides simple platform emulation
  - BIOS, Ethernet (e100), IDE and SCSI emulation
- Install paravirtualized drivers after booting
- for high-performance IO

Red Hat Xen FV Performance



Xenoprof

- http://xenoprof.sourceforge.net/xenoprof_2.0.txt
- Can be used to do profiling of HV, dom0, domU
- Can do “a ctive” or “p assive” profiling of domains
  - Active is when the domain itself is involved in the profiling (requires paravirtualized domains)
  - Passive is when the domain is not directly involved in the profiling, but some data can still be ascertained about it
- Active profiling setup

1) Start up any domains that you want to profile
2) Install oprofile and kernel-debuginfo package
3) Make sure oprofiled is stopped on all domains

```
# opcontrol -shutdown
```

Xenoprof (continued)

- Active profiling setup (continued)

4) On domain-0, start up the oprofile daemon with:

```
# opcontrol --reset ; opcontrol --start-daemon --active-
  domains=0,3 --vmlinux=/usr/lib/debug/lib/modules/2.6.18-
  8.el5xen/vmlinux --xen=/usr/lib/debug/boot/xen-syms-2.6.18-
  8.el5.debug -separate=all
```

5) On each domain you want to profile:

```
# opcontrol --reset ; opcontrol --start -
  vmlinux=/usr/lib/debug/lib/modules/2.6.18-8.el5xen/vmlinux
```

6) Now run your application/workload

7) On each domain you are profiling:

```
# opcontrol --stop ; opcontrol -shutdown
```

8) On domain-0

```
# opcontrol --stop ; opcontrol -shutdown
```

9) You have to run opreport on each domain to get individual profiling data

10) Running opreport on domain-0 will show you hypervisor time as well

High Availability

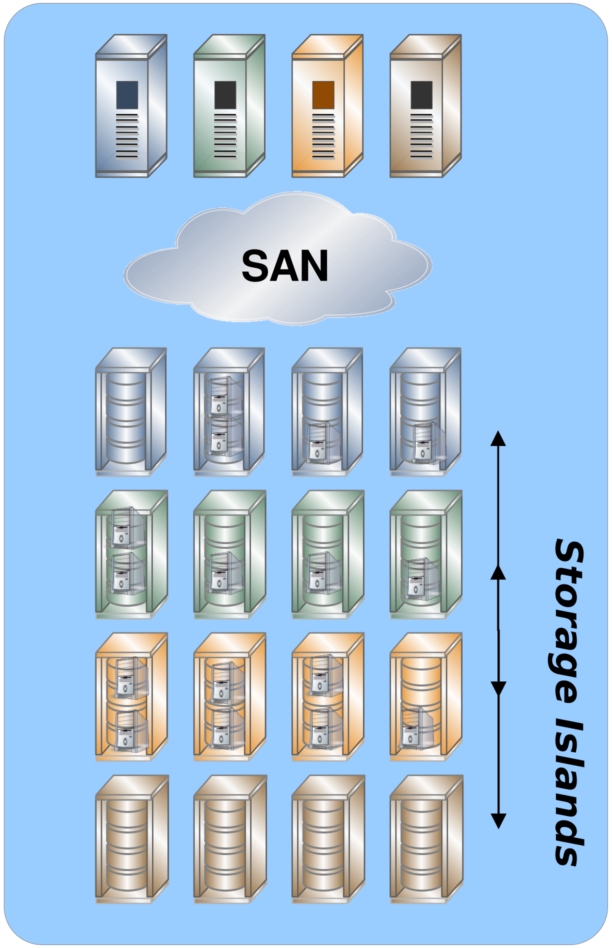
High Availability

- Streamline testing of HA implementation
  - Build Redhat Cluster Suite environment without the need of (addtl.) physical hardware
  - Enables development of HA scripts and integration without the need for a permanent RHCS Test or QA environment
  - Can enable ISVs to look at HA and integrate their applications into RHCS (dont need multiple physical server, storage etc..)
- Highly available Virtual Machines using RHCS
  - Local cluster
  - GFS2 can be used as central storage pool
  - IPVS for scale out load balancing

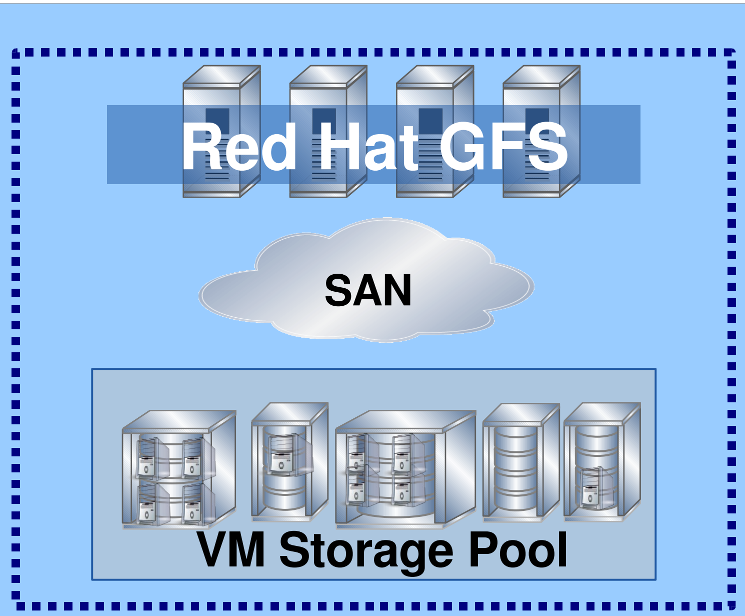
GFS as a central VM pool

- Reduce complexity and management efforts
- Foundation for live migration

Without GFS

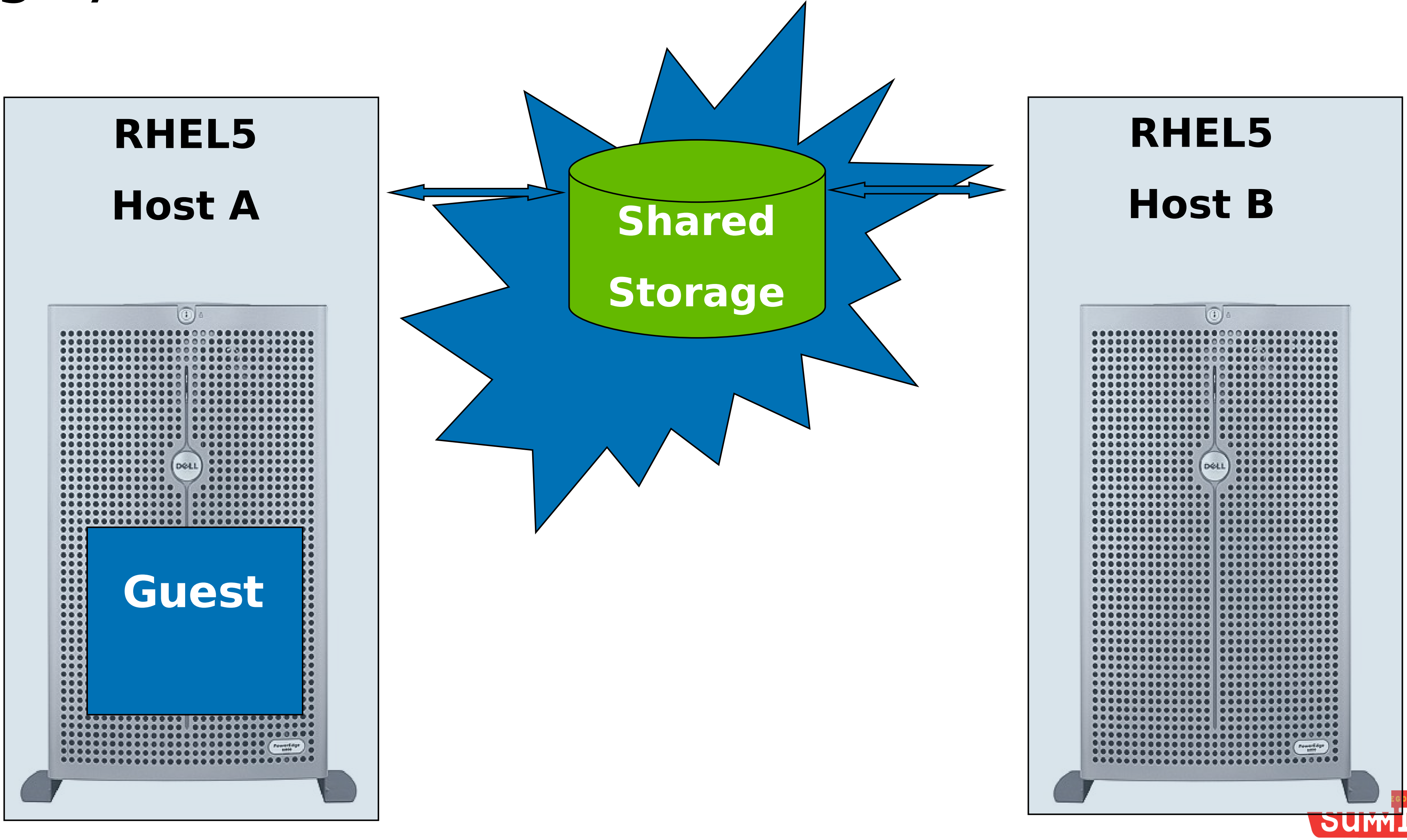


With GFS



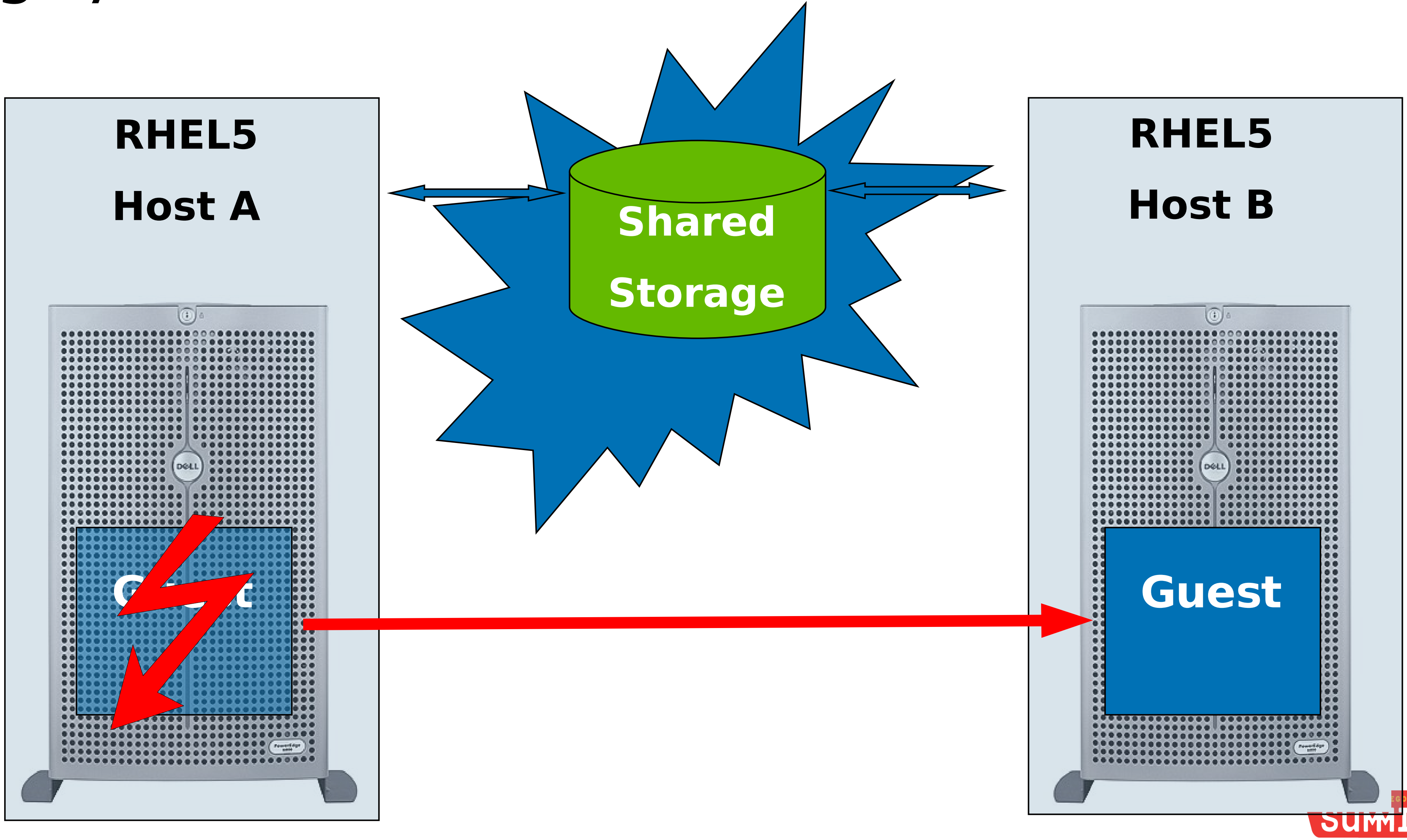
- Central pool for virtual machines
- Can run on any server part of the cluster
- Ensures continuous VM and data availability
- Simplifies data management & lower cost
- Ease of replication

Highly Available RHEL5 Host



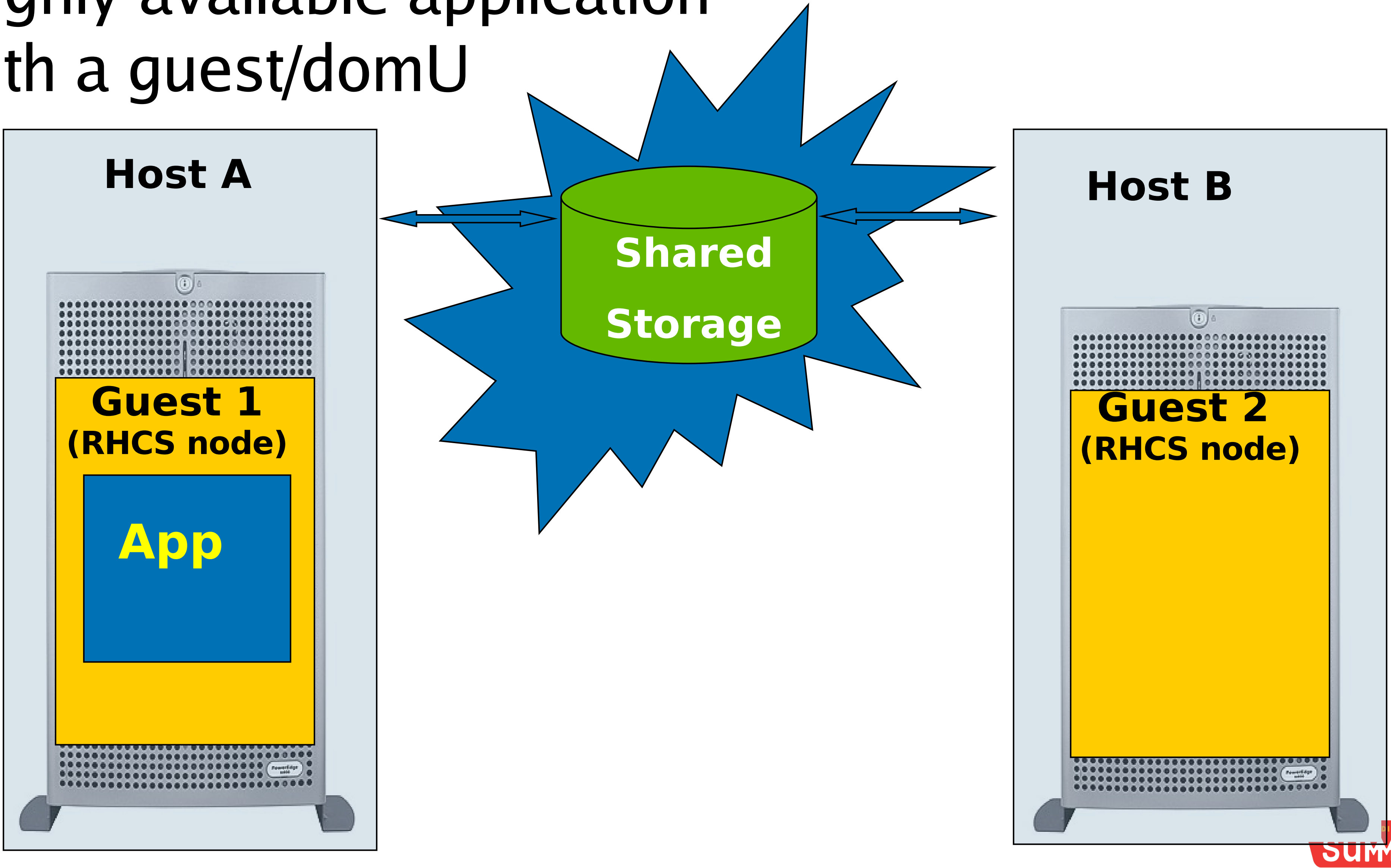
Guest running as a RHCS service

Highly Available RHEL5 Host



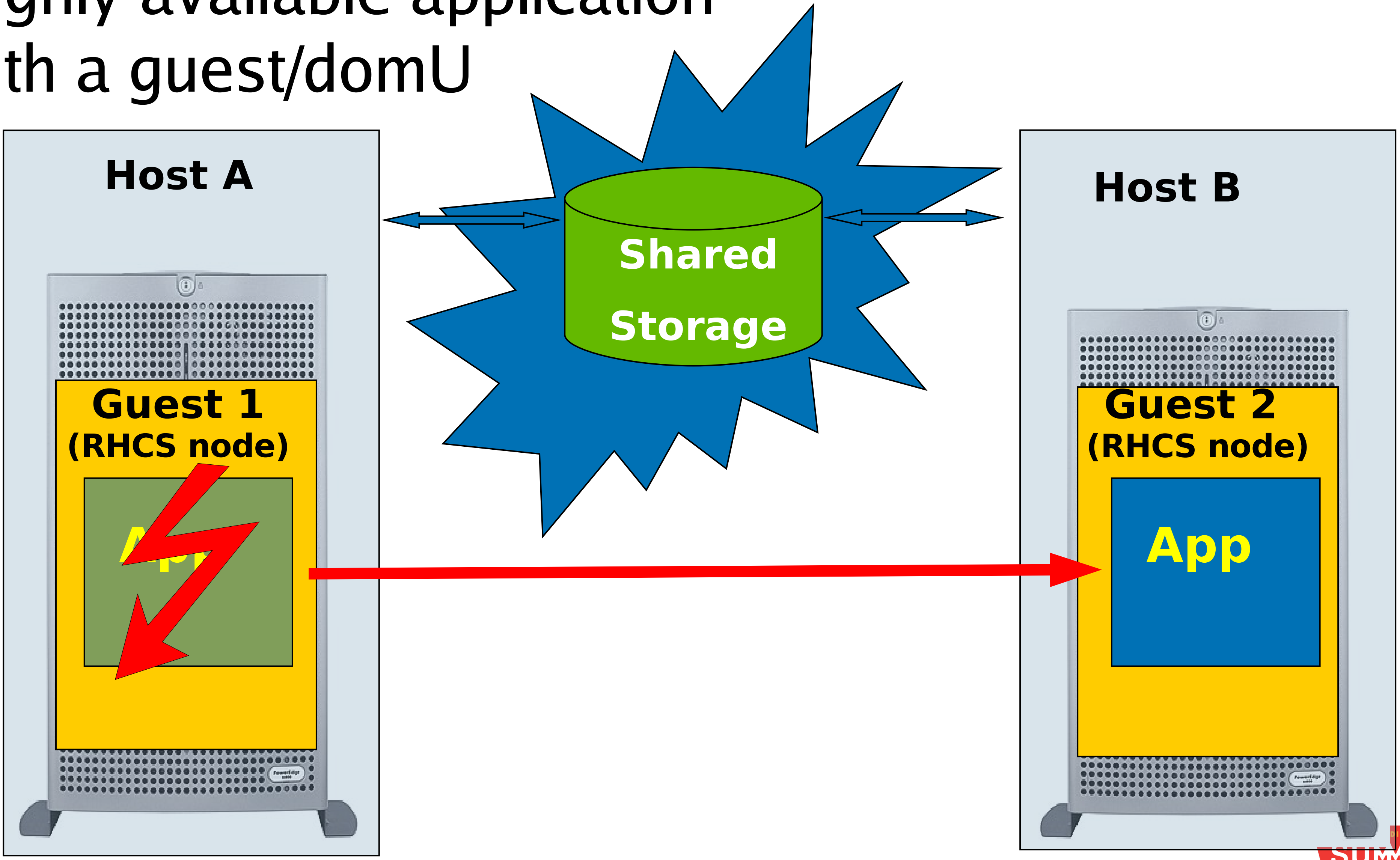
Guest running as a RHCS service failing over to standby Hypervisor

Highly available application with a guest/domU



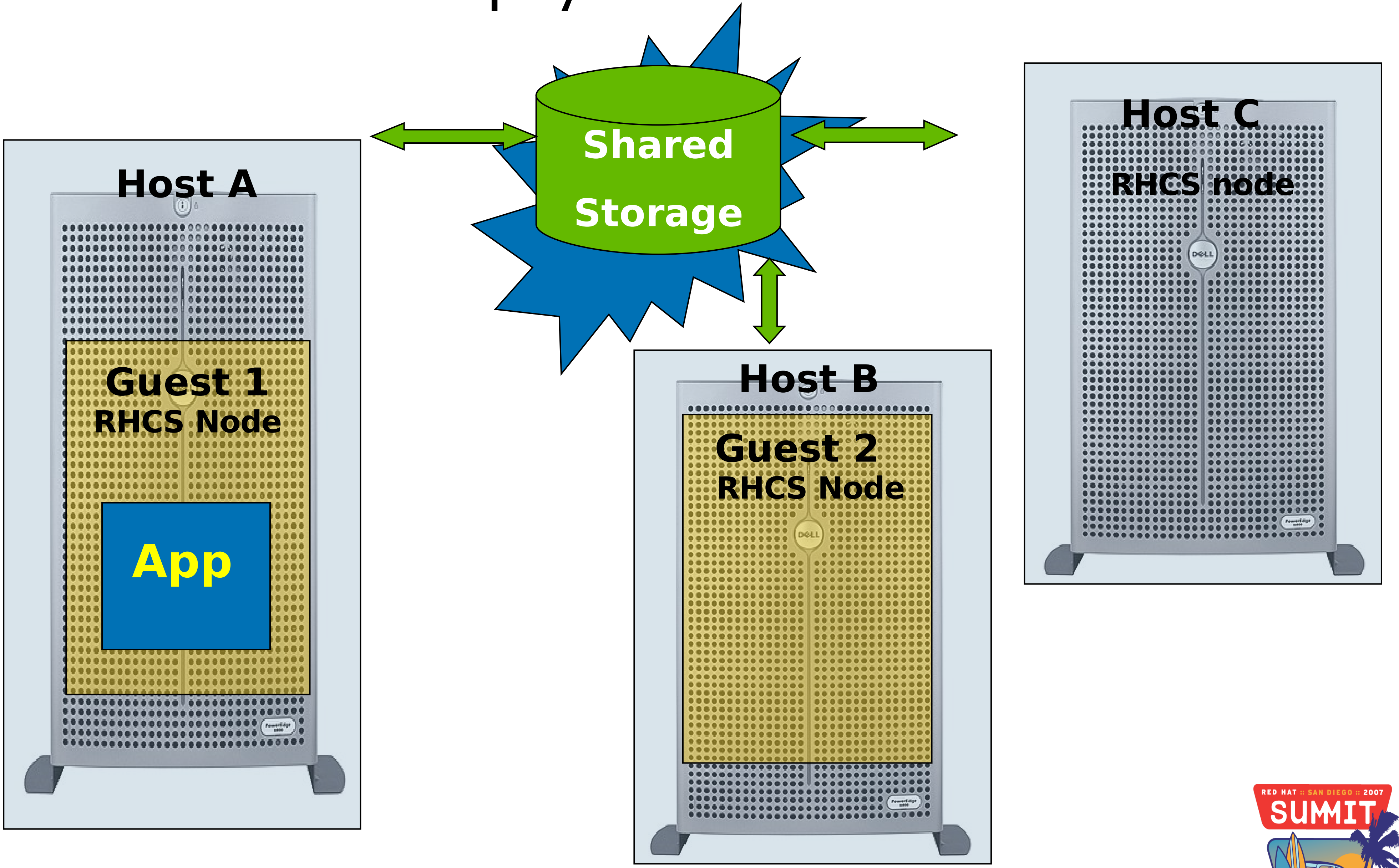
RHCS cluster between domUs/guests

Highly available application with a guest/domU

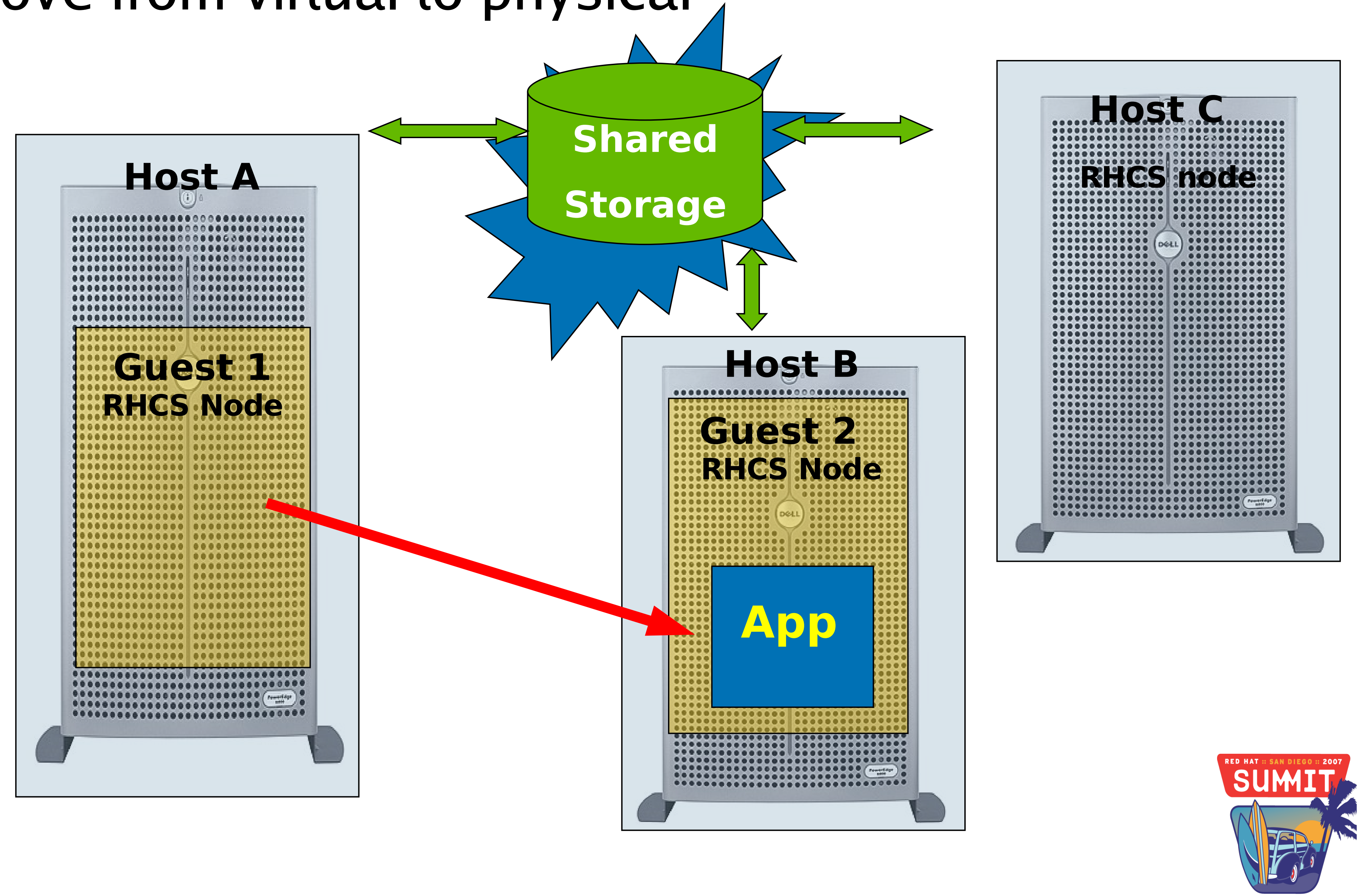


RHCS cluster between domUs/guests
Failing over applications inside a domU/guest

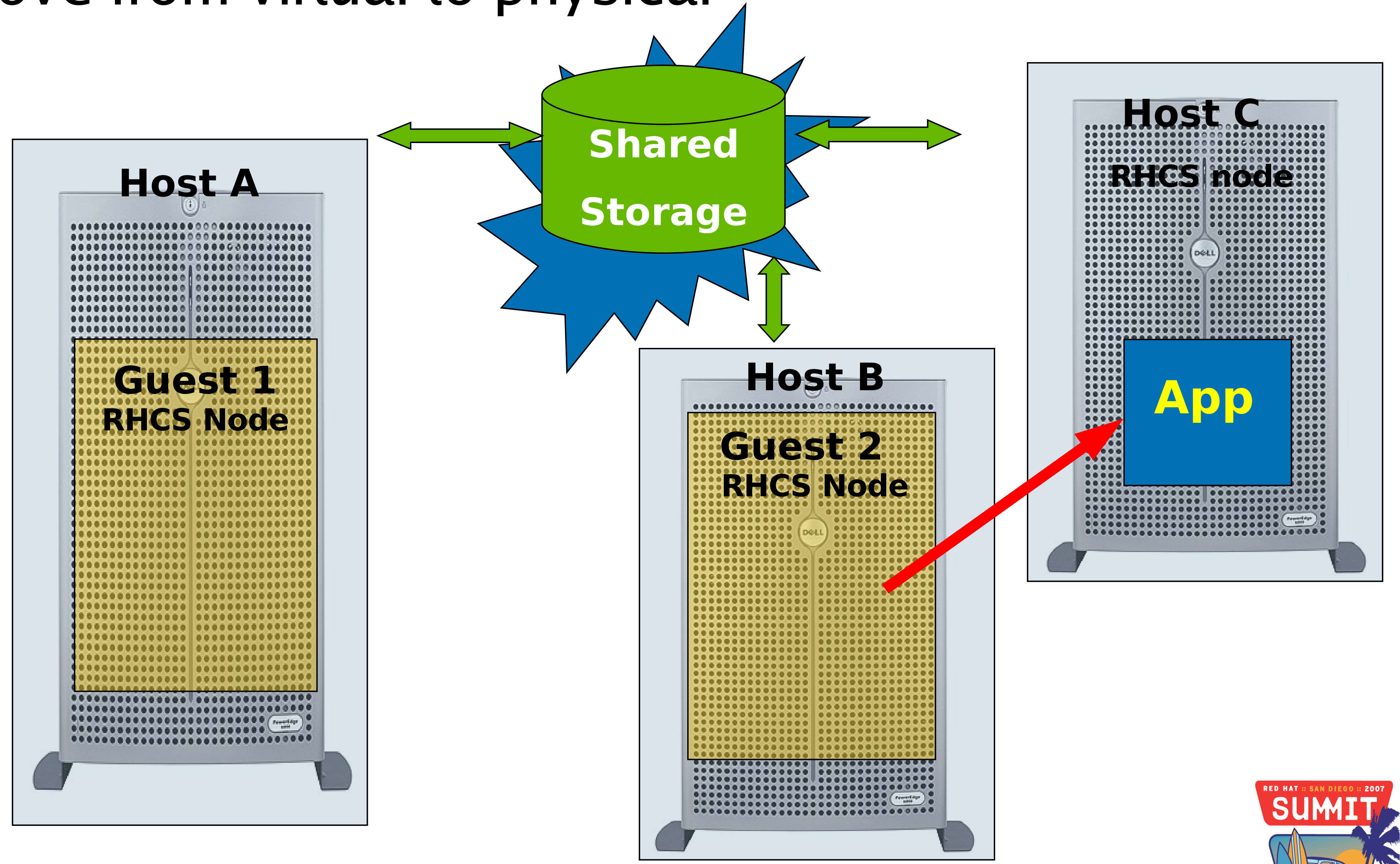
Move from virtual to physical



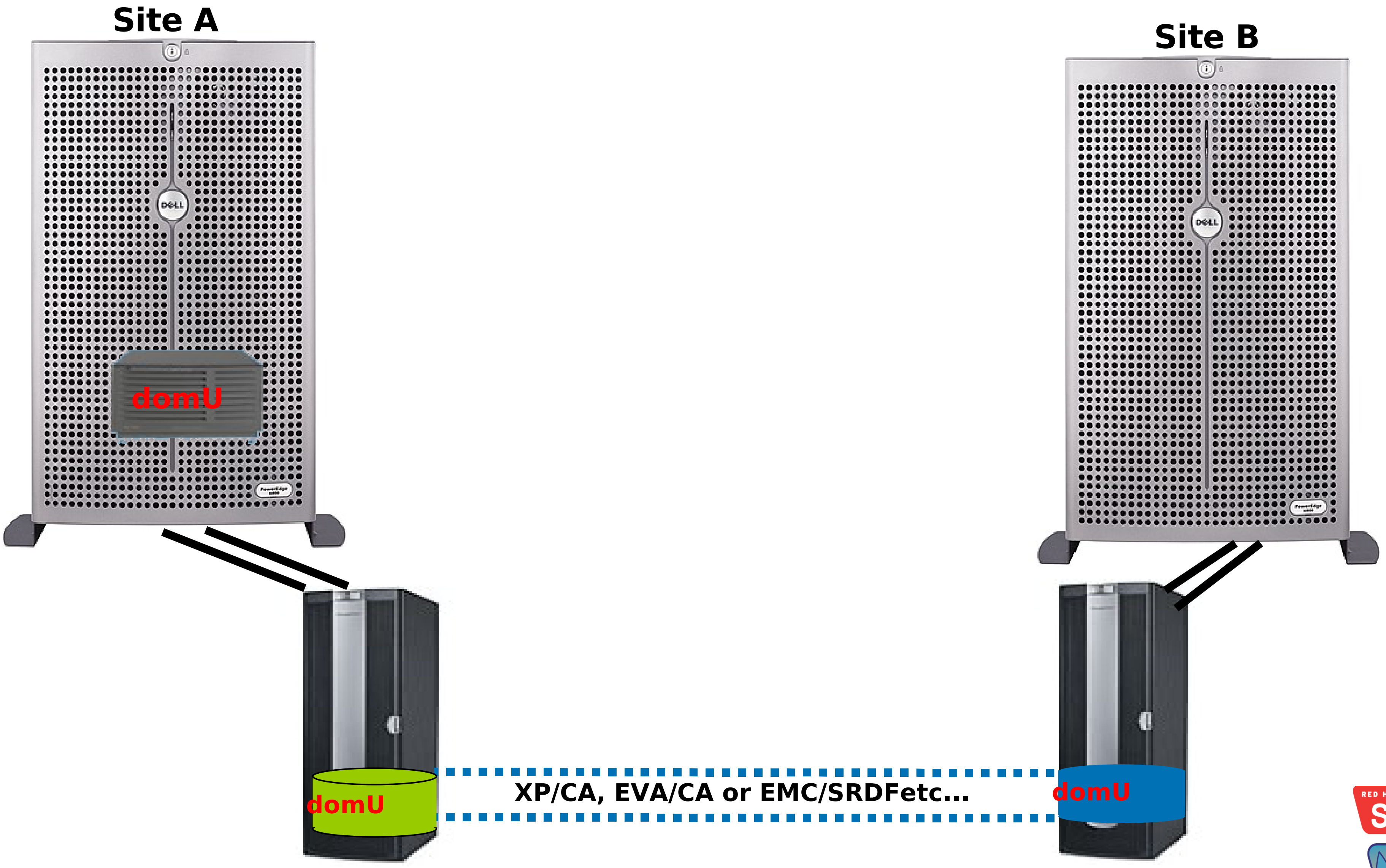
Move from virtual to physical



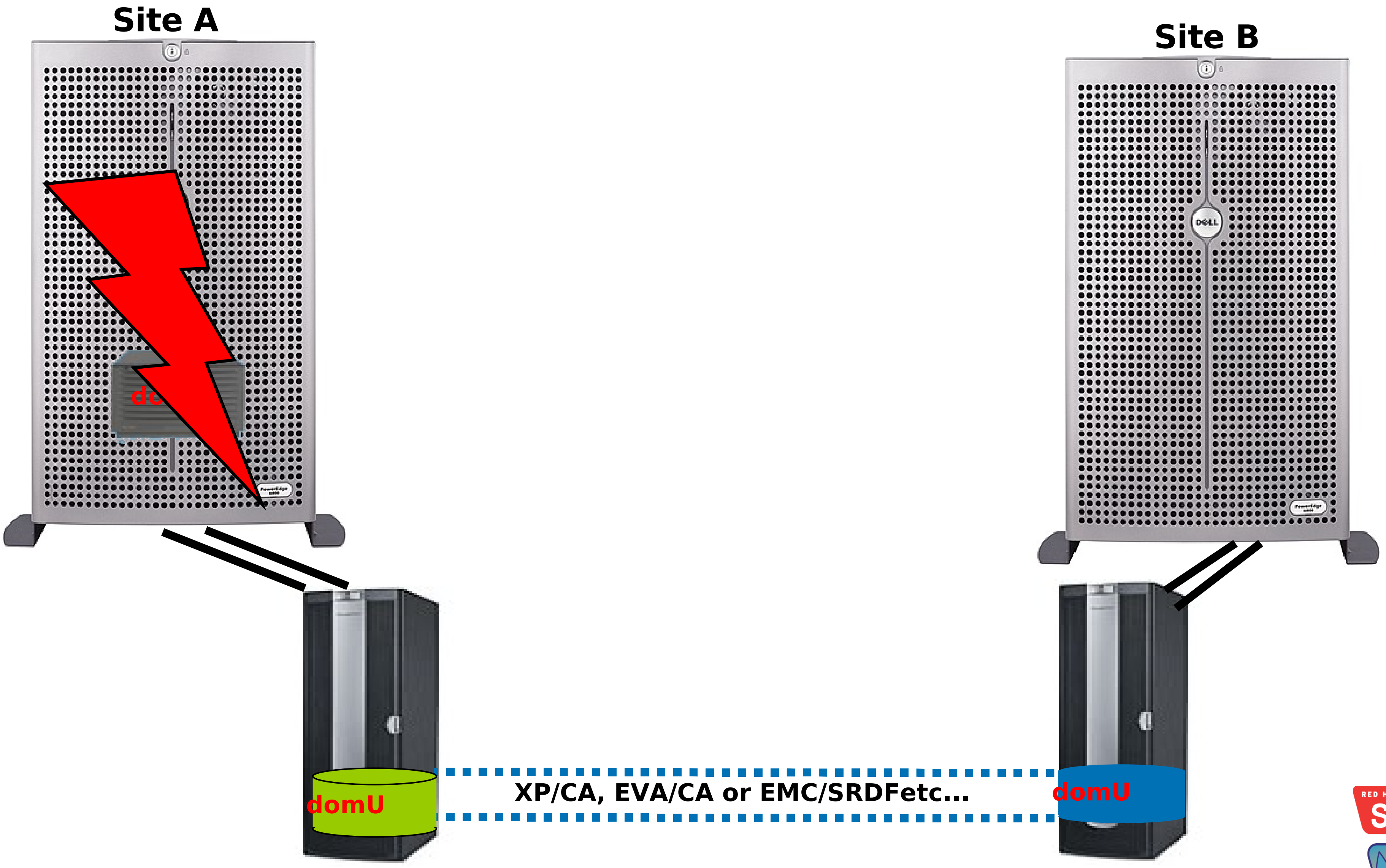
Move from virtual to physical



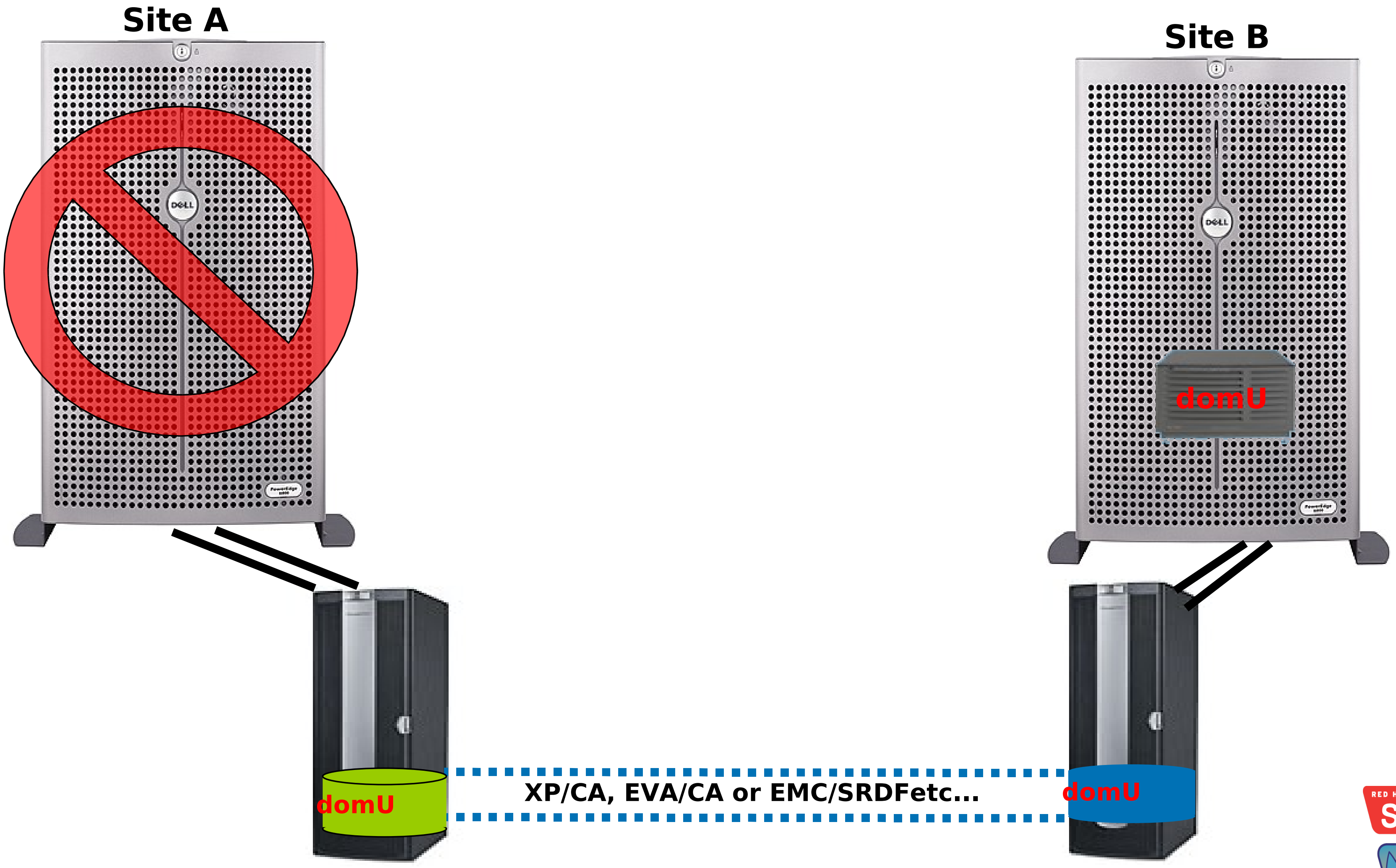
Virtual Machines and DR/DT



Virtual Machines and DR/DT

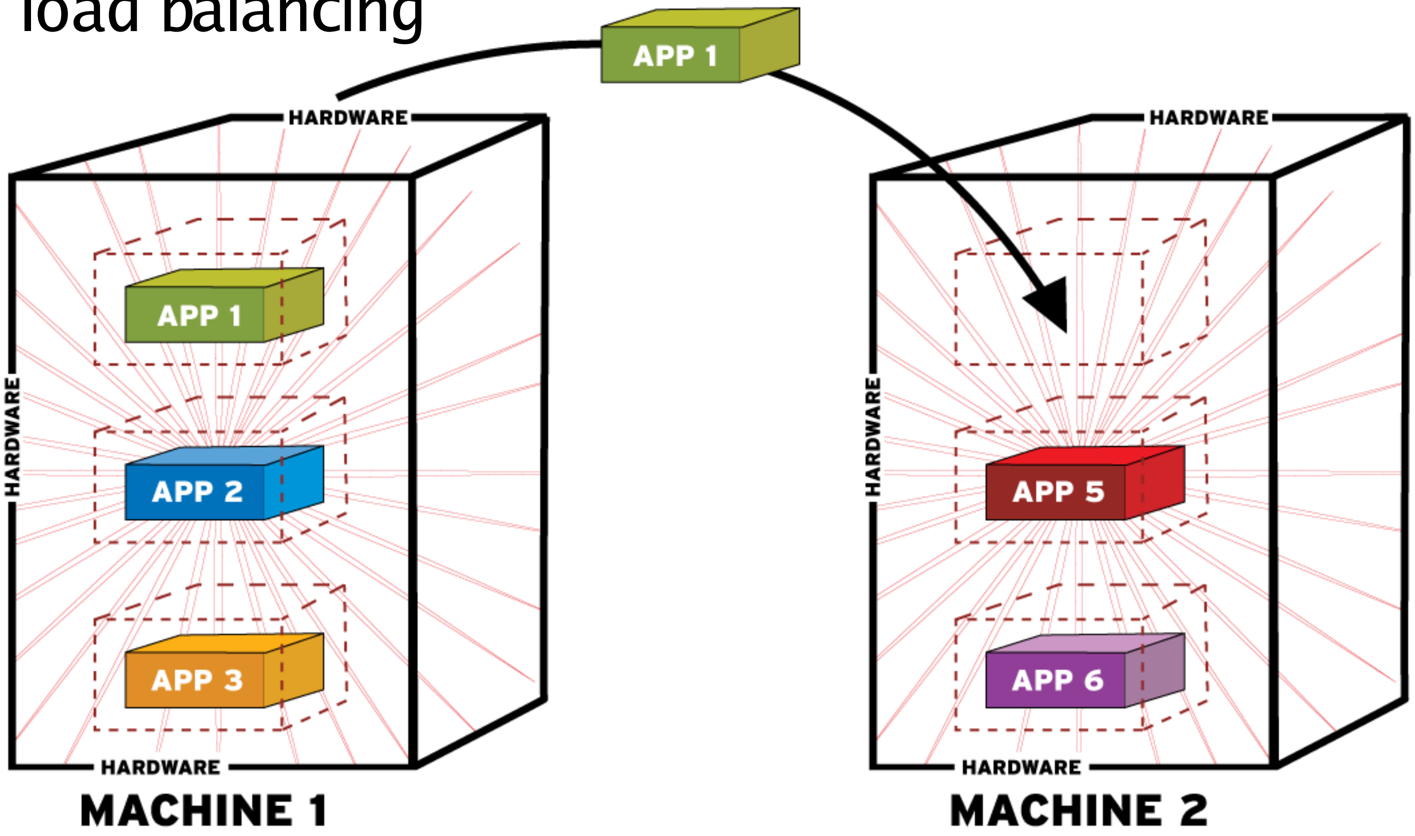


Virtual Machines and DR/DT

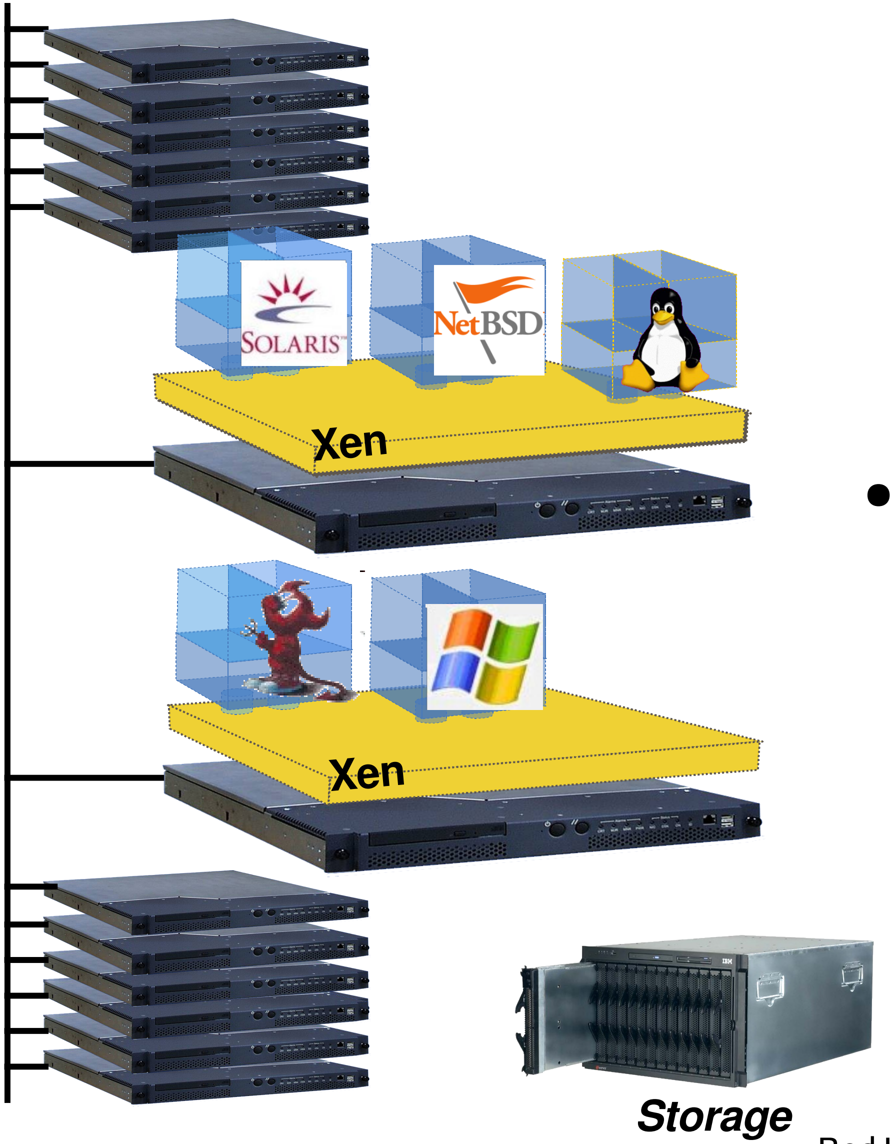


Live Migration

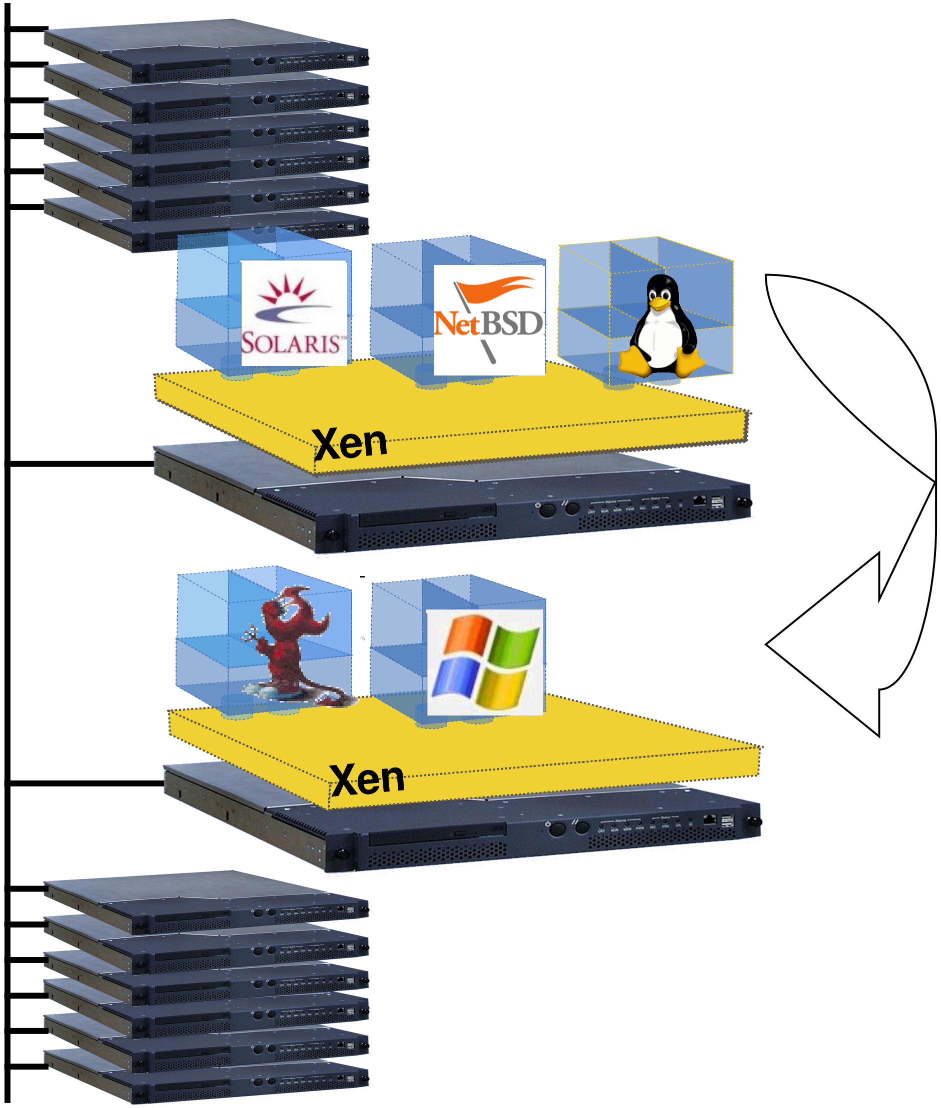
- Implement zero downtime environment for applications, hardware and operating system maintenance
- Deploy a pro-active management approach
- On-demand capacity management
- Dynamic load balancing



- Storage
  - NAS: NFS, CIFS
  - SAN: Fibre Channel
  - iSCSI, network block dev
  - drdb network RAID
- Good connectivity
  - common L2 network
  - L3 re-routeing



VM Relocation : Motivation

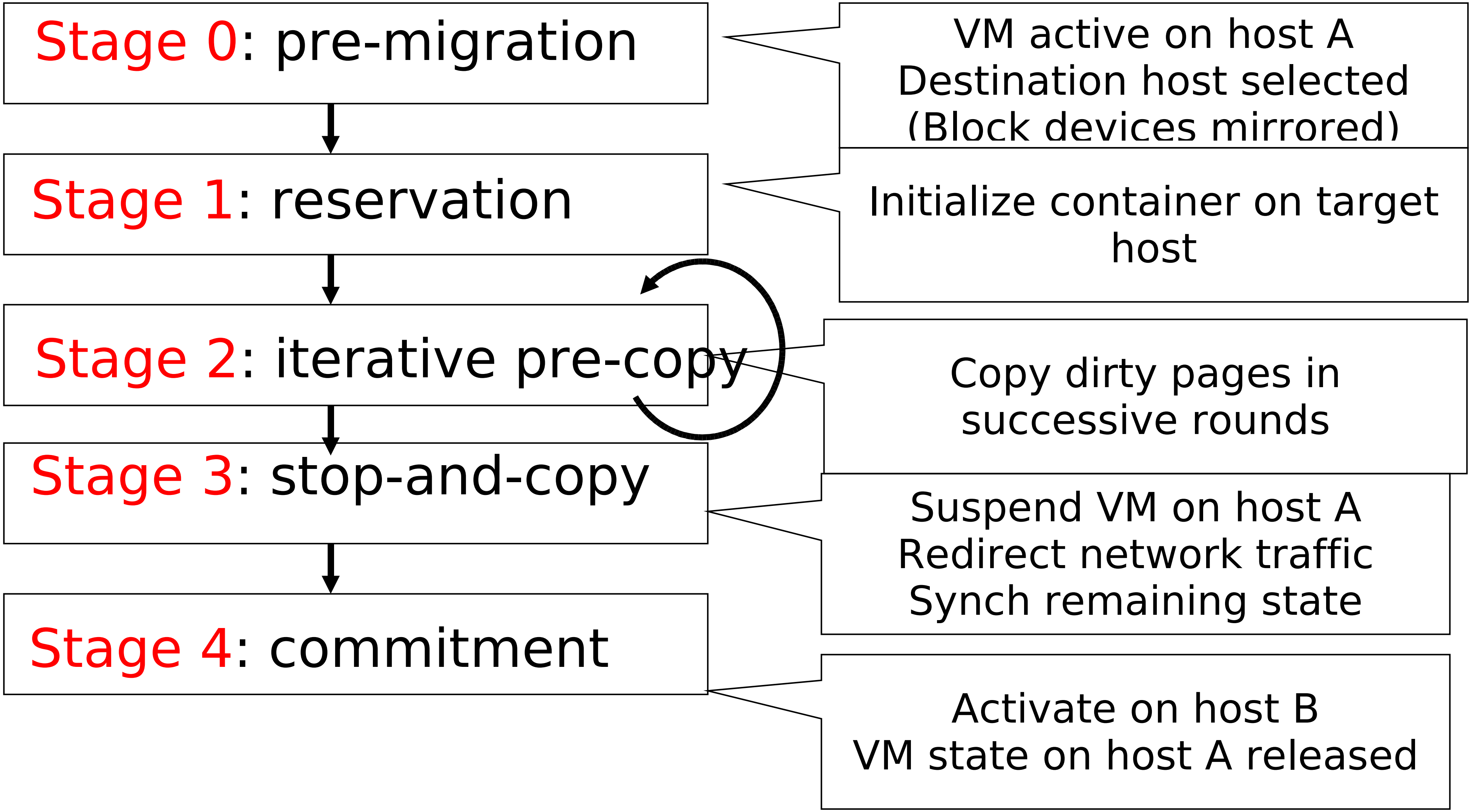


- VM relocation enables:
  - High-availability
    - Machine maintenance
    - Replacing old hardware
  - Load balancing
    - Statistical multiplexing gain

Migration

- Requires shared storage
  - Both domain 0's must access same disk image or physical device
- Using Physical device
  - LUN on SAN
  - Exported block device
    - ISCSI, GNBD
  - Both domain 0's should use same name for device
    - eg. /dev/sdg
    - Use UDEV rules for mapping if required

Migration Life Cycle



Stage 0: pre-migration

VM active on host A
Destination host selected
(Block devices mirrored)

Stage 1: reservation

Initialize container on target host

Stage 2: iterative pre-copy

Copy dirty pages in successive rounds

Stage 3: stop-and-copy

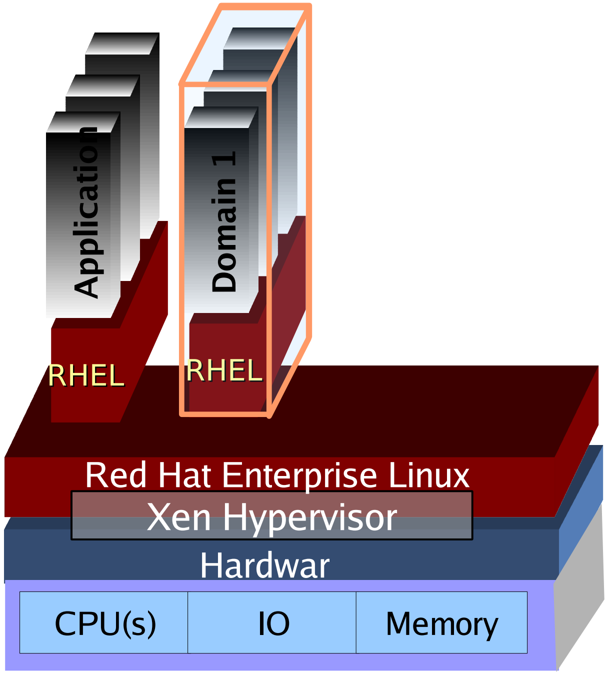
Suspend VM on host A
Redirect network traffic
Synch remaining state

Stage 4: commitment

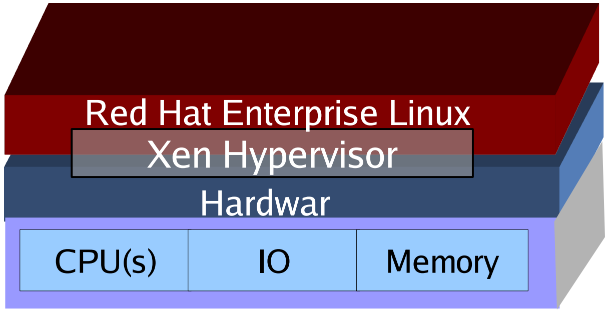
Activate on host B
VM state on host A released

Migration

Domain 1 running on physical machine A is to be moved to Machine B Currently users are accessing Machine A



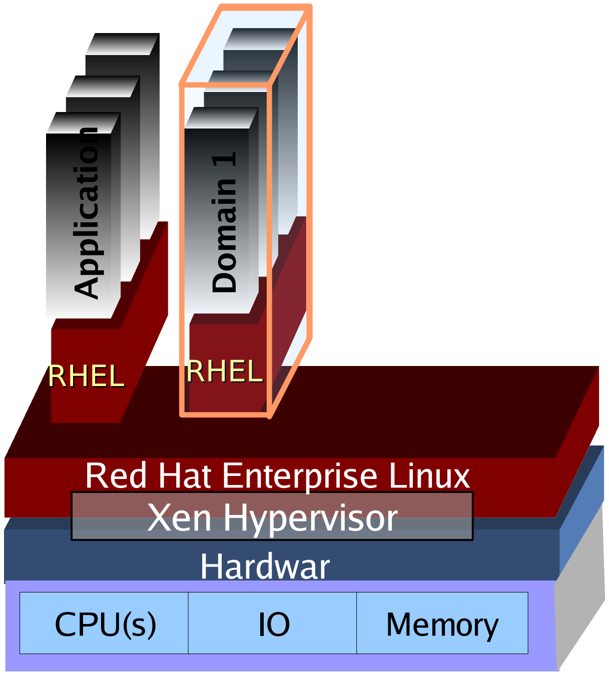
Machine A



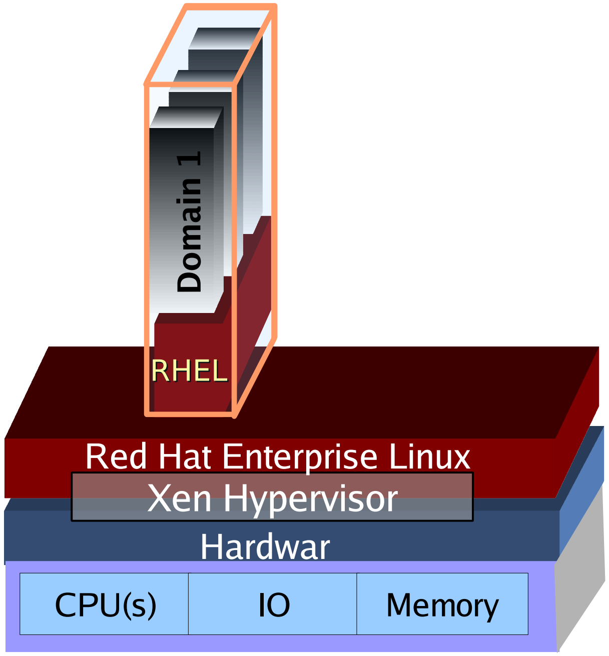
Machine B

Migration

Step 1 :
Initialize domain on Machine B
VM config transfered from machine A



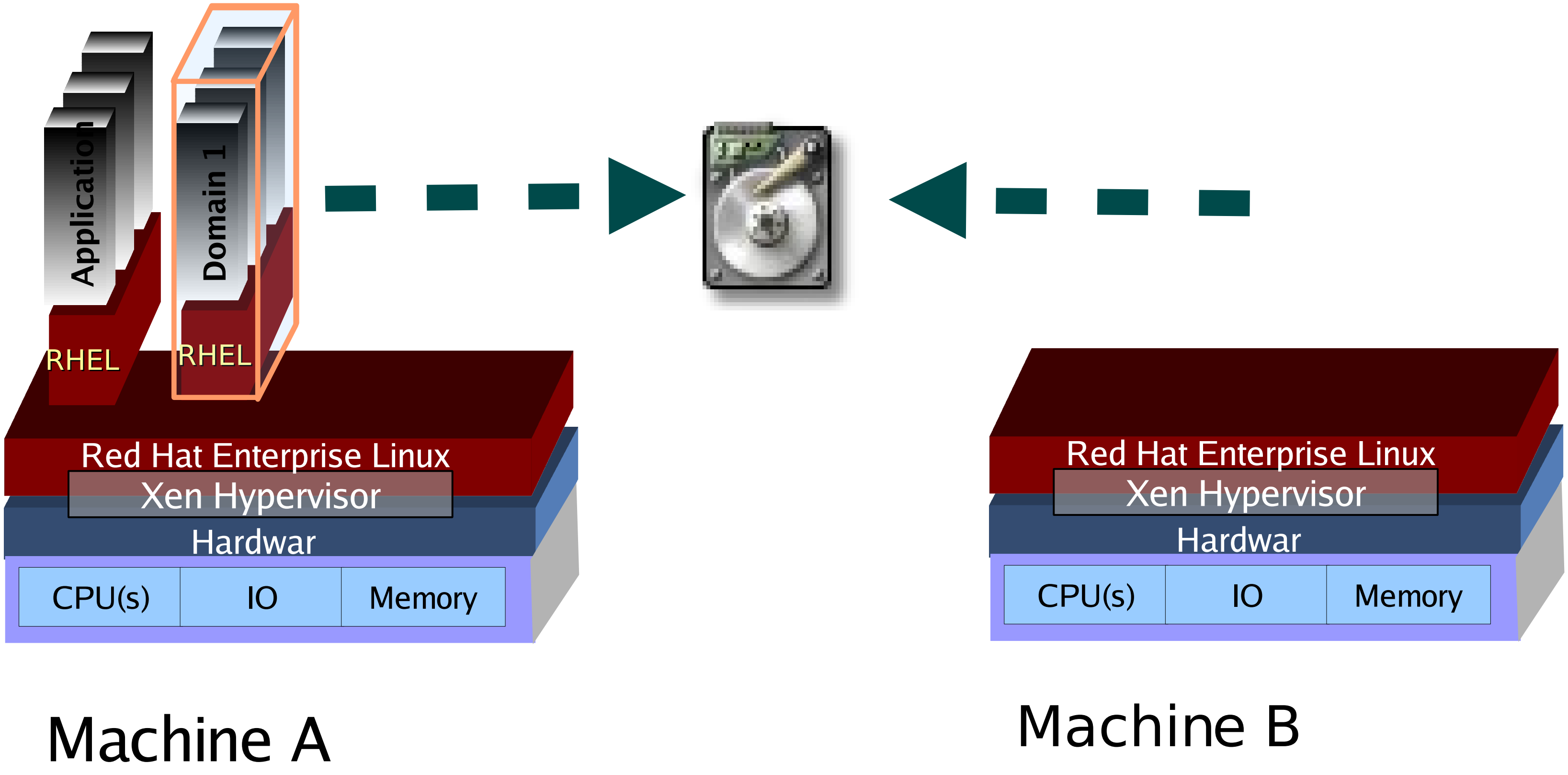
Machine A



Machine B

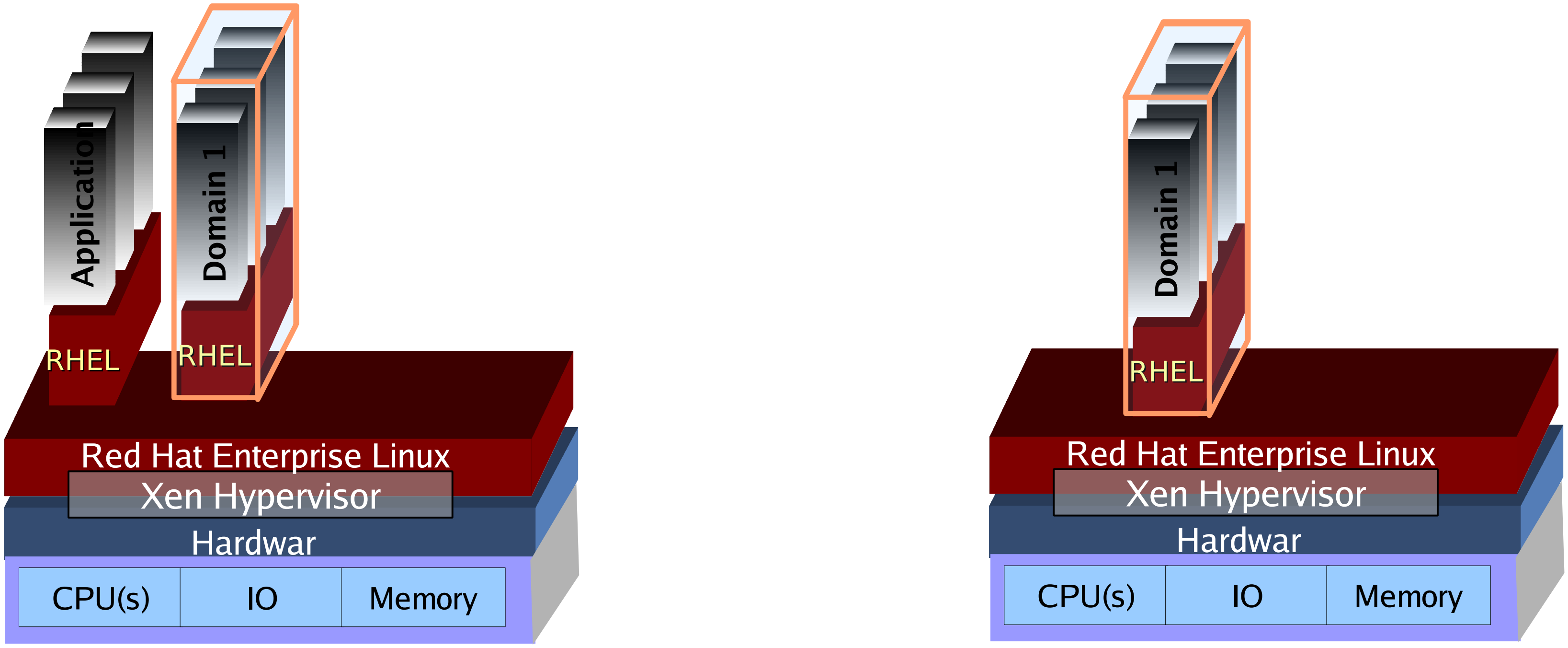
Migration

Step 2 :
Access block devices (disk) on Machine B



Migration

Step 3 :
Machine A commits ~10% of resources to migration
Start shadow paging

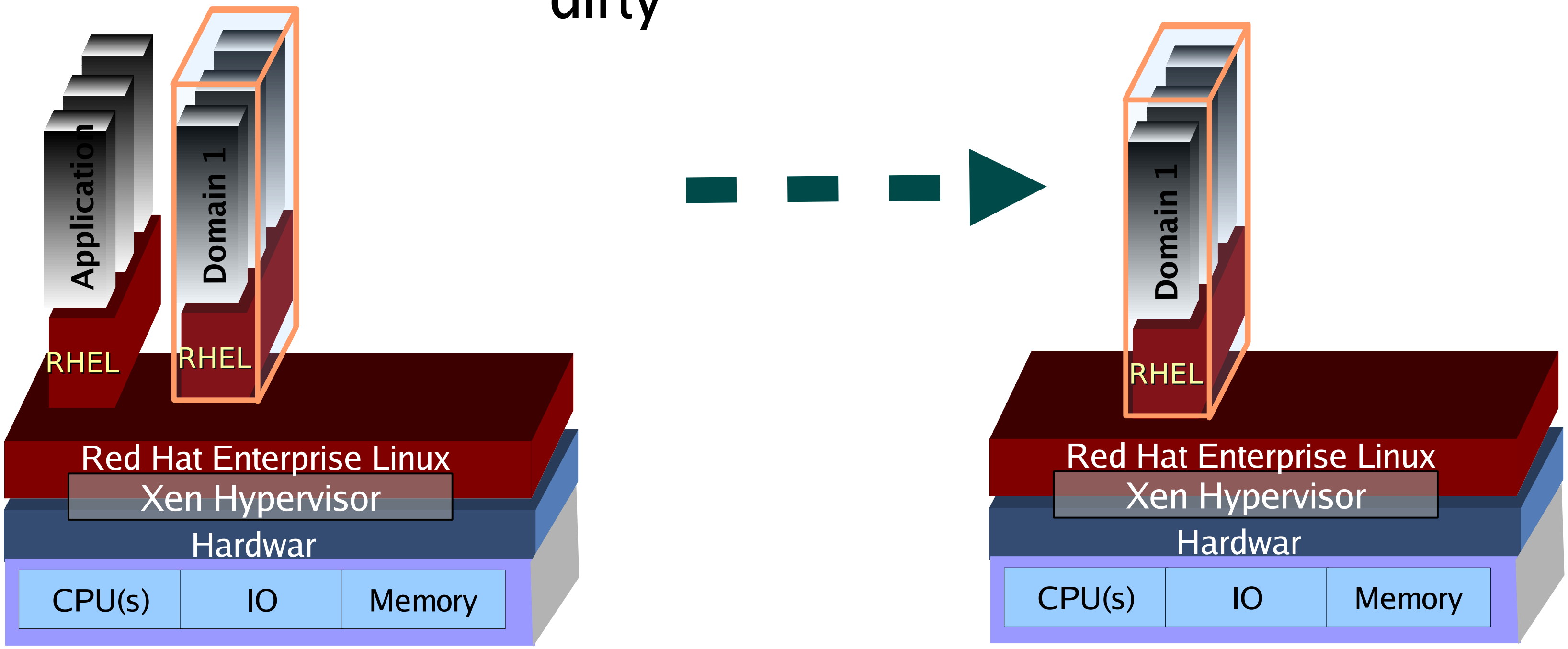


Machine A

Machine B

Migration

Step 4 :
Start copying memory image from Machine A to Machine B
Changed memory pages marked as "dirty"

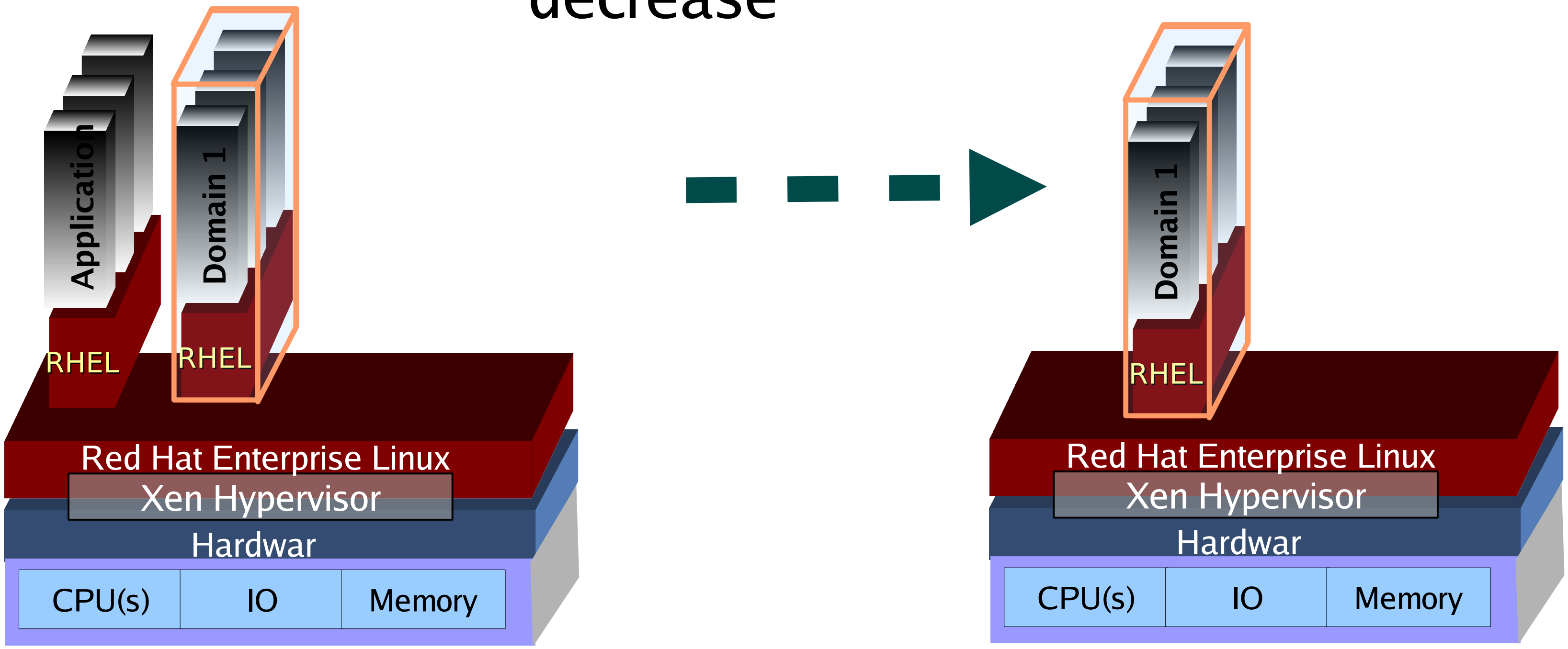


Machine A

Machine B

Migration

Step 5 :
Copy dirty pages.
Step completed multiple times until number of dirty pages does not decrease

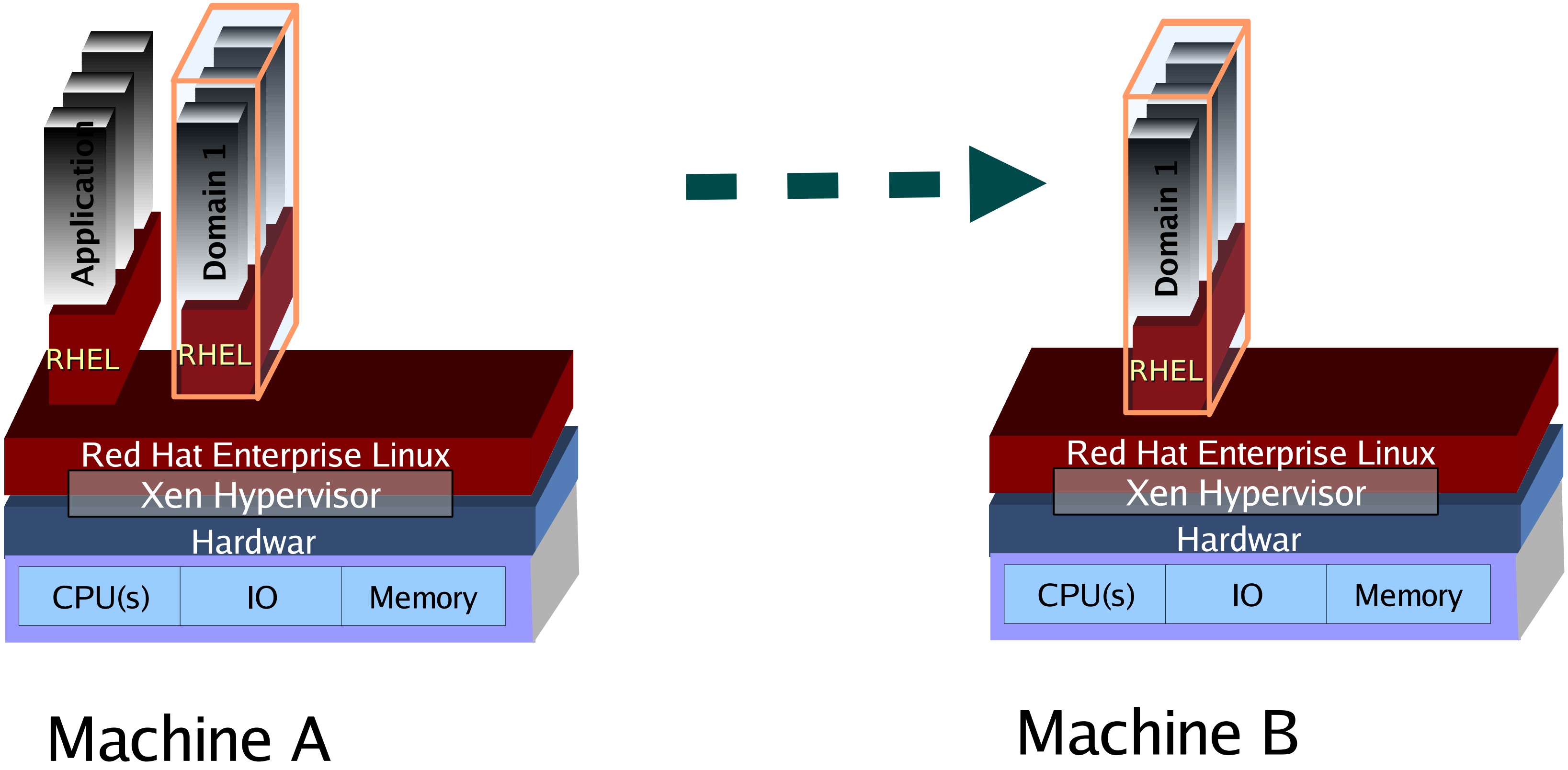


Machine A

Machine B

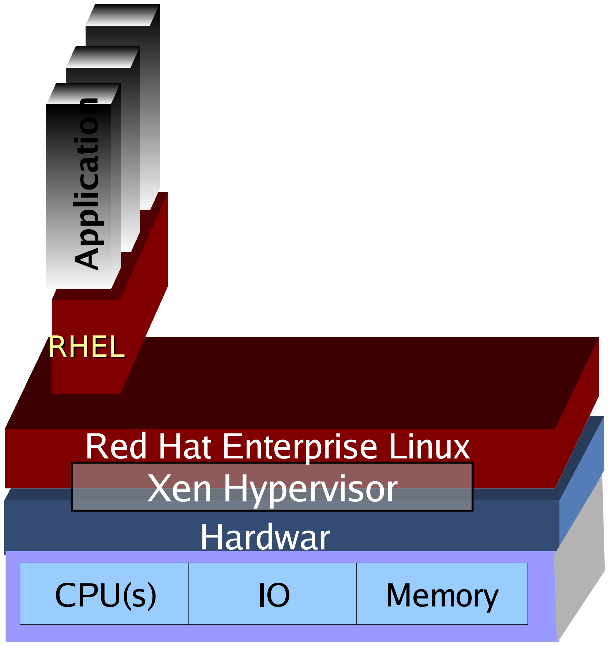
Migration

Step 6 :
Domain 1 is suspended on Machine A
Remaining “dirty”page s copied

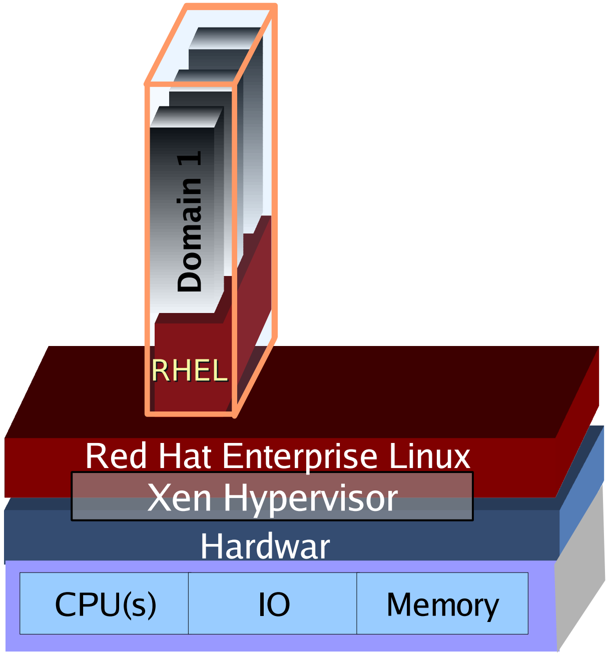


Migration

Step 7 :
ARP redirect used to point network traffic to machine B
Domain 1 restarted on Machine B



Machine A



Machine B

Red Hat Xen Performance Summary

RHEL5 Virtualization Performance Summary

- Single guest virtualization overhead (<5%)
- Scalability sharing single results w/ multi guest (<15%)
- Avoid paging/swaping using dynamic virt tools
  - Vcpu-set
  - mem-set
- SMP guest 80% scalable
    - Use credit schedule tuning multiple guest
- Storage alternative – FC, NFS and iSCSI
- Network Performance
  - 70% of peak in PV,
  - 3-4x 1-Gbit performance guest-dom0

Resources

- Red Hat
  - http://www.redhat.com/
- Virtualization Infocenter
  - http://www.openvirtualization.com/
- Libvirt
  - http://www.libvirt.org/
- Virt-Manager
  - http://virt-manager.et.redhat.com/
- Red Hat Cluster Suite
  - http://www.redhat.com/solutions/gfs/