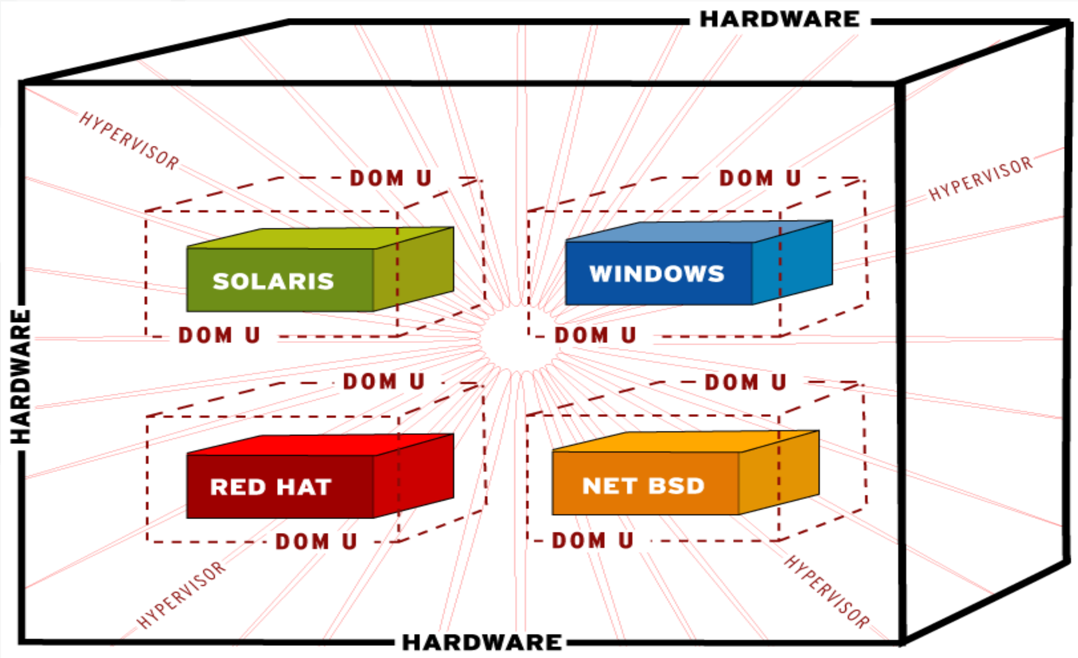
Xen and the State of Open Source Virtualisation

Sung Min David Joo
Senior Technical Account Manager,
Red Hat Korea

What's Virtualisation?

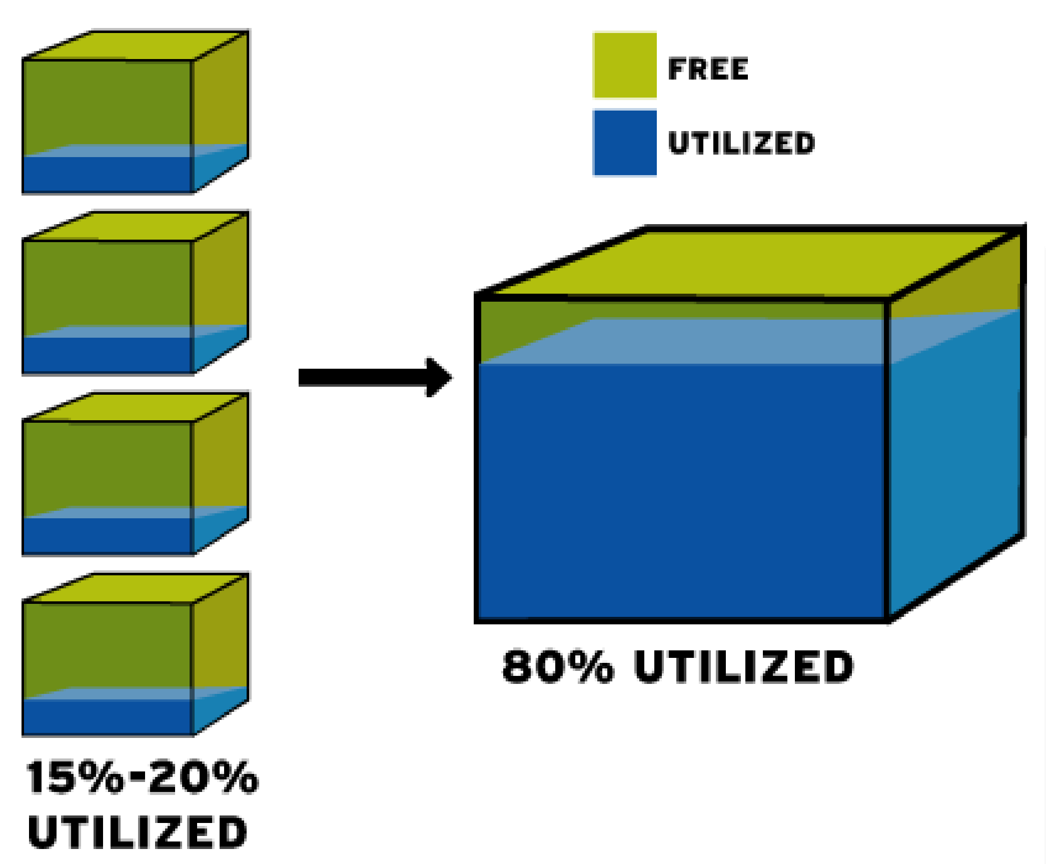
- Running different *Virtual Machines* (VMs) on a single machine.
  - Different isolated *guest operating systems* with different applications on same physical hardware.

- A supervising master program called a *Hypervisor* manages these Virtual Machines.



Benefits of Virtualisation

- **Reduced cost**
  - Dramatic lowering of TCO
- **Security**
  - Continuous availability
- **Agility**
  - Operational scalability



Power of Open Source Architecture
OPEN SOURCE SYMPOSIUM 2006

Virtualisation Models

- Single Kernel Image (SKI)
- Full Virtualisation (FV)
  - Processor Emulation
  - “Native” Virtualisation
  - Hardware Assisted
- Para-virtualisation (PV)

Virtualisation Models - I

- Single Kernel Image (SKI)
  - Light weight virtualisation where a shared host operating system spawns multiple user spaces.
  - Each virtual operating systemmust be identical.
  - Examples:
    - Solaris Zones
    - SWsoft Virtuozzo
    - Linux-VServer

Virtualisation Models - II

- Full Virtualisation (FV)
  - Two categories
    - Processor Emulation
    - “Native” Emulation
  - Two classes of hardware to be emulated
    - Processor & supporting chipset
    - Hardware
      - IO Controllers – Storage, network, etc.
      - Video card
      - USB
      - etc.

Virtualisation Models - II

- Full Virtualisation (FV): Processor Emulation
  - Uses software to emulate CPU
  - All “guest” calls to CPU are handled by software
  - Allows emulation to cross hardware platforms.
    - eg. Run Windows on Mac hardware.
    - Emulate x86 hardware using software running on PowerPC
  - Disadvantage
    - Very slow!
  - Examples
    - Bochs, Qemu, VirtualPC (PowerPC version)

Virtualisation Models - II

- Full Virtualisation (FV): "Native" Virtualisation
  - Requires same chip architecture
  - Some CPU instructions executed directly
  - Kernel / Real-mode CPU instructions are dynamically re-written
  - Binary on-the-fly patching/rewrite of those calls
  - No modifications required for guest operating systems
  - Disadvantage
    - Slow performance
  - Examples
    - VMware, VirtualPC, VirtualServer

Power of Open Source Architecture
OPEN SOURCE SYMPOSIUM 2006

Virtualisation Models - II

- Full Virtualisation (FV): Hardware Assist
  - CPU emulation difficult in x86 architecture
    - The x86 architecture not designed with virtualisation in mind. Kernel expected to run in “ring 0”
  - Existing approaches incur performance penalties
  - CPU vendors developing Hardware extensions to support virtualisation
    - Intel – VMX extensions (vanderpool)
      - CoreDuo, Pentium D 900 series, Pentium 4 662 & 672
    - AMD – SVM (pacifica)
  - Provides on-chip support for virtualisation
    - *Still requires Hypervisor*

Virtualisation Models - III

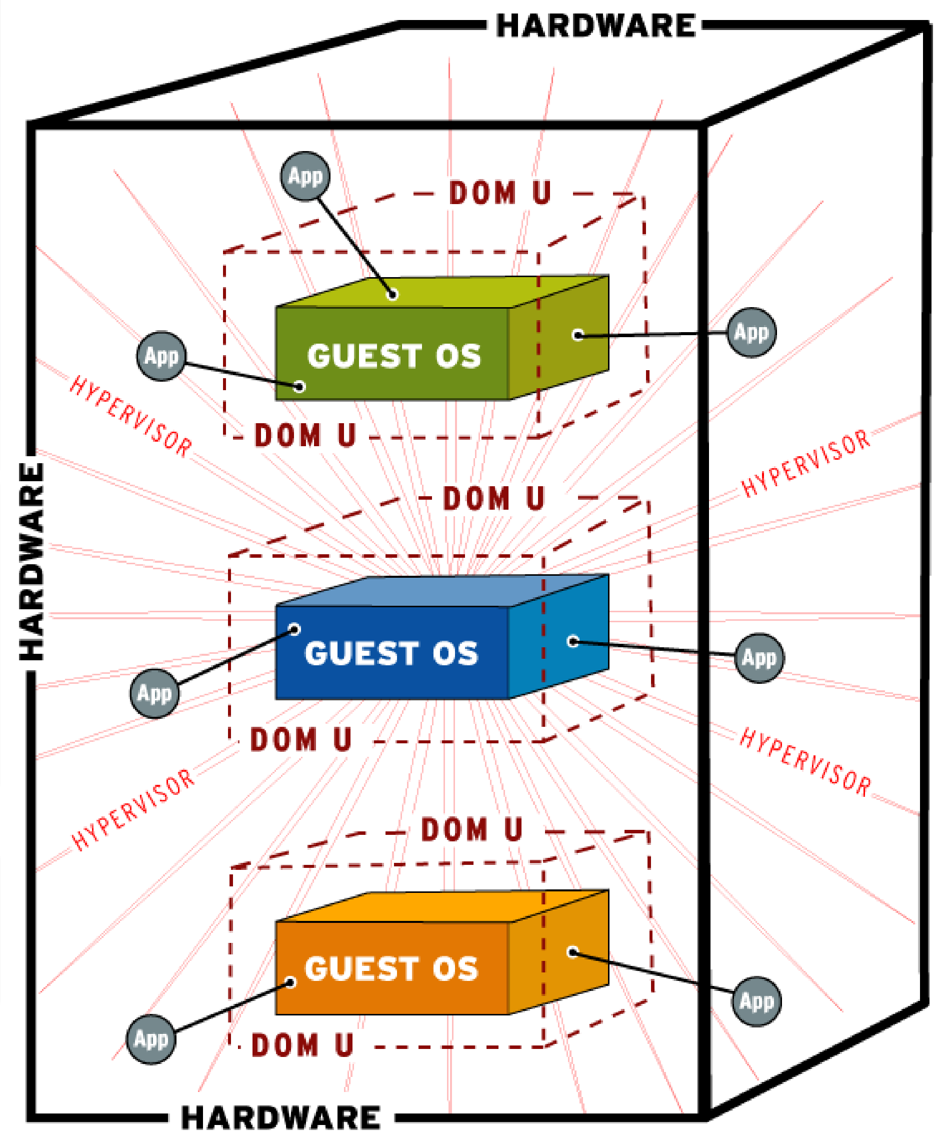
- Para-Virtualisation (PV)
  - Idea born from research project at University of Cambridge, England
  - Requires minor changes to guest operating system
  - Make Operating system "virtualisation aware"
    - Guest operating system "cooperates" with Hypervisor
    - No need to emulate hardware and CPU instructions
    - Operating system talks to Hypervisor instead of emulation layer
  - Advantage
    - Near Native speeds 0.5% -> 3% overhead

Xen Virtualisation Technology

- Open Source project founded by Cambridge University
- Developed by the open source community
- Supported by leading software and hardware vendors
  - XenSource, Red Hat, IBM, Intel, AMD, Novell
- Widely accepted by open source community
- Not just Linux
  - *BSD, Open Solaris, Plan 9 .....

Xen Virtualisation Technology

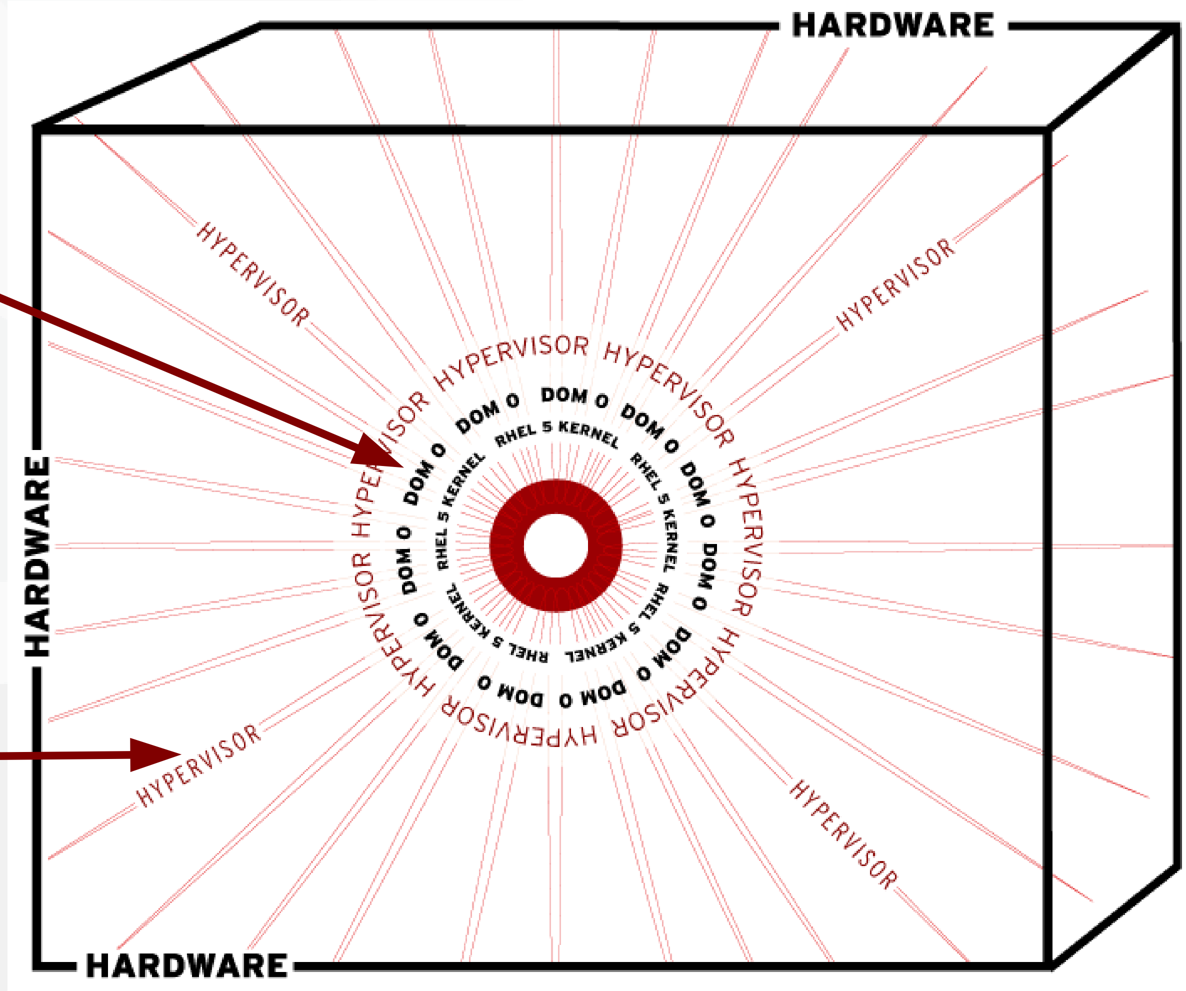
- **Almost native performance**
- Creates an "apparent" independent server for each guest operating system
  - **Completely and securely isolated**
  - Allows **multiple workloads** to **co-exist** safely
- **Migrate guests** quickly as required.
- **Clone guests** without adding cost or complexity.



Xen Architecture - Host

Domain 0
The master domain, which provides hardware support as well as interfacing to guests and management tools.

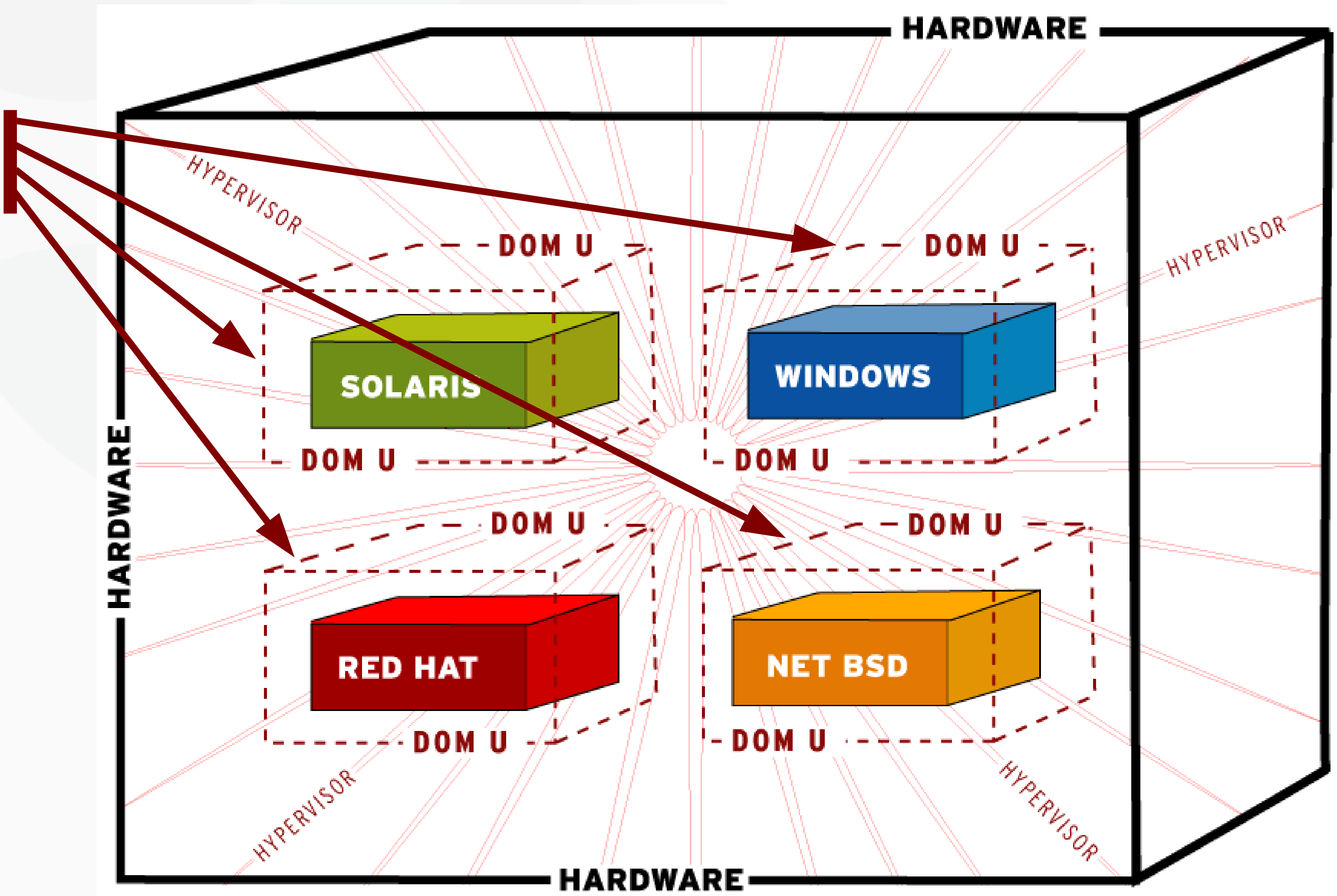
Xen Hypervisor
Provides low-level hardware control, scheduling, and communications. This allows transparent sharing of resources and enforcing resource limits.



Power of Open Source Architecture
OPEN SOURCE SYMPOSIUM 2006

Xen Architecture - Guests

Dom U
The Virtual Machine that runs the guest operating system.



Xen Architecture - Memory

- Memory Management
  - Works in cooperation with Hypervisor
- Memory “Balloon” Driver
  - Set minimum and maximum memory allocation for a domain
  - Domain can request more memory (up to it's maximum)
  - Returns unused memory back to the pool

Xen Architecture

- Typically hardware accessed by Domain 0
  - DomU's use "front end" drivers
- Devices can be "hidden" from Domain 0
  - Allow a device to be directly connected to DomU
  - eg. Network card, specialized I/O card
- A dedicated "Resource Domain" can be created
  - Moves some or all devices from Domain 0
  - Creates a more stable Domain 0
    - Moves "suspect" device drivers from Dom0
    - If Resource domain(s) crash they can be quickly restarted

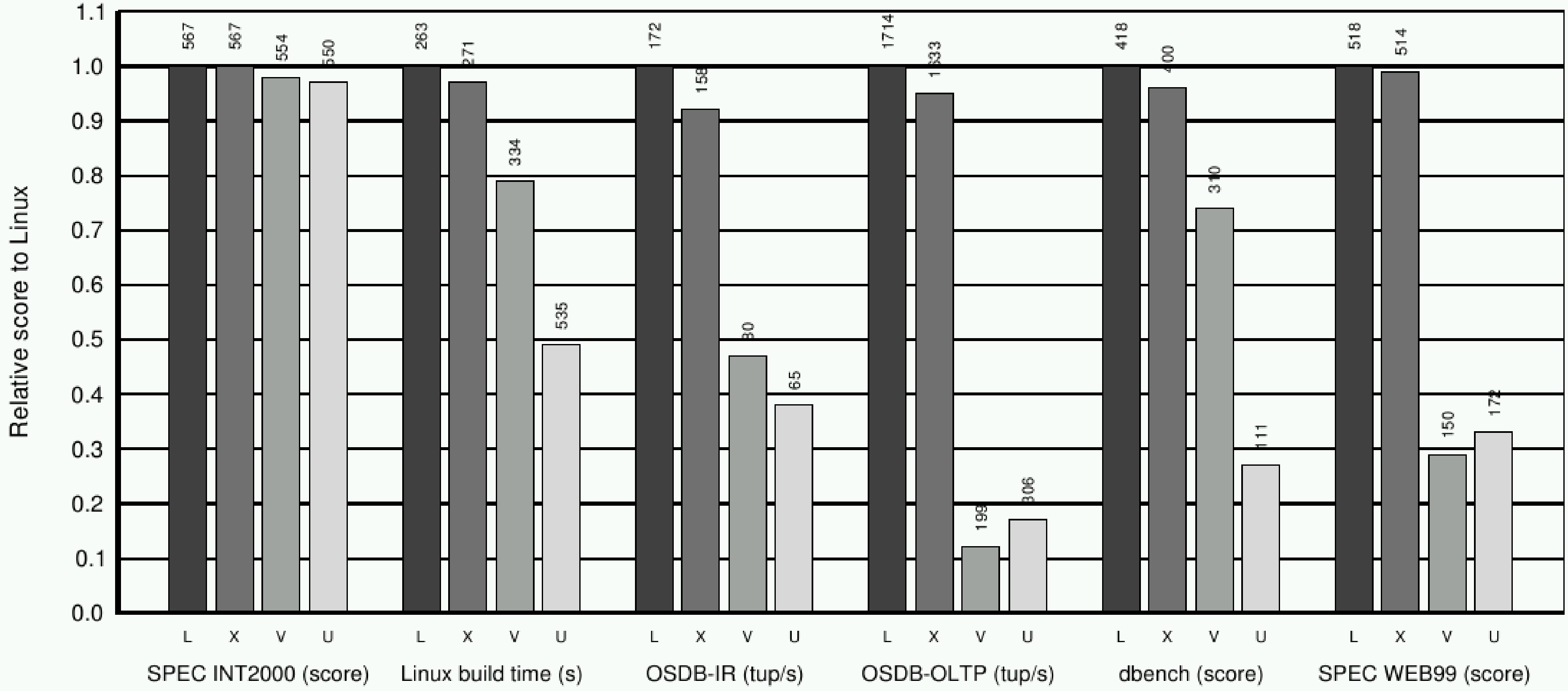
Xen Architecture - Block Device

- Block Devices (disks) are connected to domains
  - File in Domain 0
    - eg. /opt/vm/disk.img
    - Disk image can be a single file systemor complete disk image including partitions
    - Simple to implement but bottleneck for high I/O deployments
  - Physical device
    - eg. /dev/sda6
  - Logical volume
    - Using LVM
  - Devices appear as simple virtual disks in Dom U

Xen Architecture - Network Device

- Virtual Interfaces are created.
  - Virtual interface in Dom0 maps to interface in Dom U
    - Multiple virtual devices can be created in Dom U
- Virtual interfaces can be connected in two ways
  - Bridging
    - Uses bridge-utils to bridge the Dom0, DomU and 'real' interface
  - Routing
    - Uses network routing & iptables
  - Direct access to NIC card
    - eg. For firewall appliance.

Xen Performance



Relative performance of native Linux (L), XenoLinux (X), VMware workstation 3.2 (V) and User-Mode Linux (U).

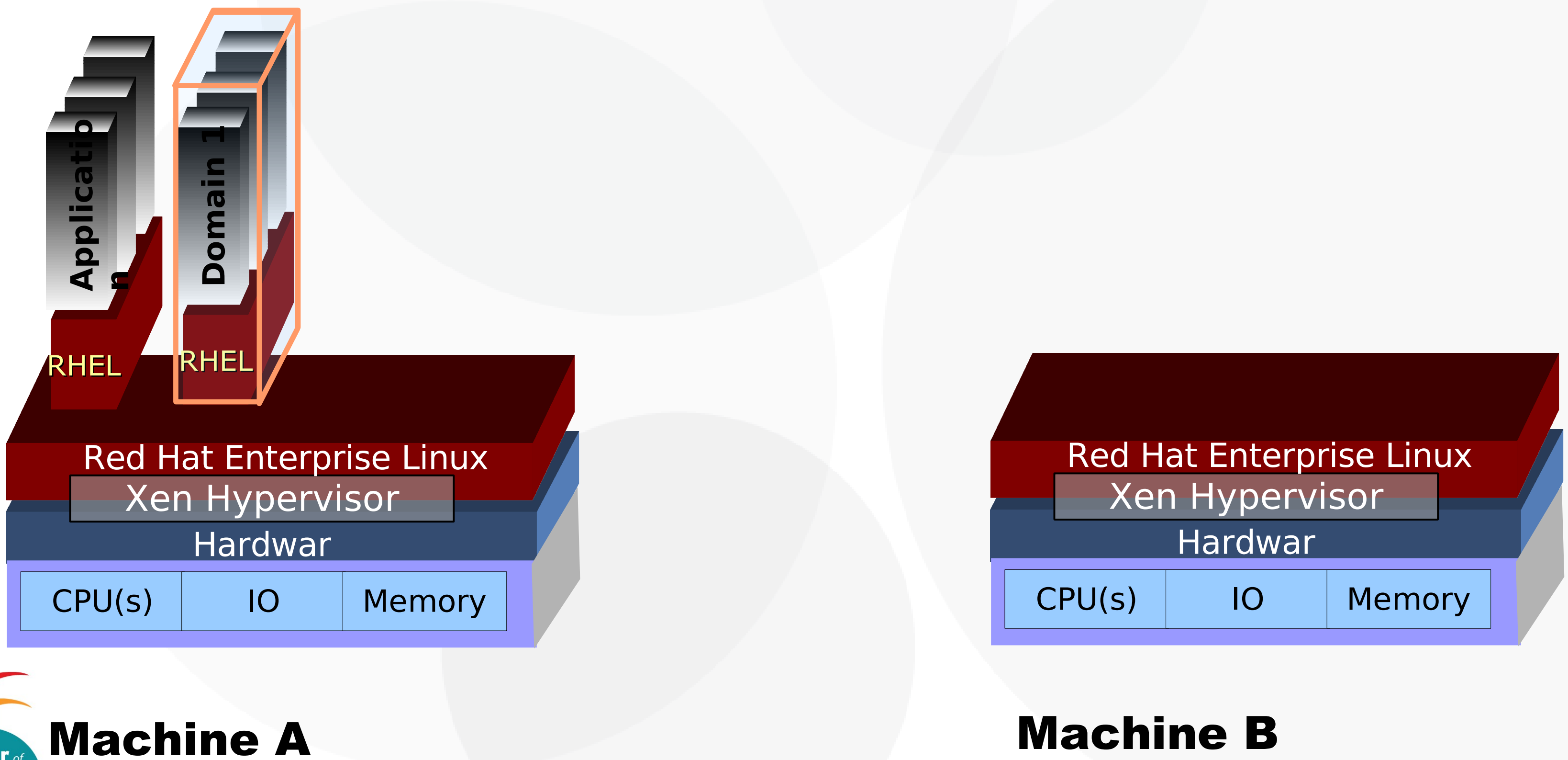
Source : XenSource

Xen Architecture

- Virtual machines (domU's) don't access hardware directly
- They see only the front end drivers
- Not tied to a particular physical machine
  - Unless hardware is directly connected (uncommon)
- Comprised of
  - Configuration file, disk image, memory image
- Domains can be "Migrated"
  - Moved between physical machines
  - Can be performed "Live" without suspending guest
  - **"down time" between 60ms and 300ms!!!**

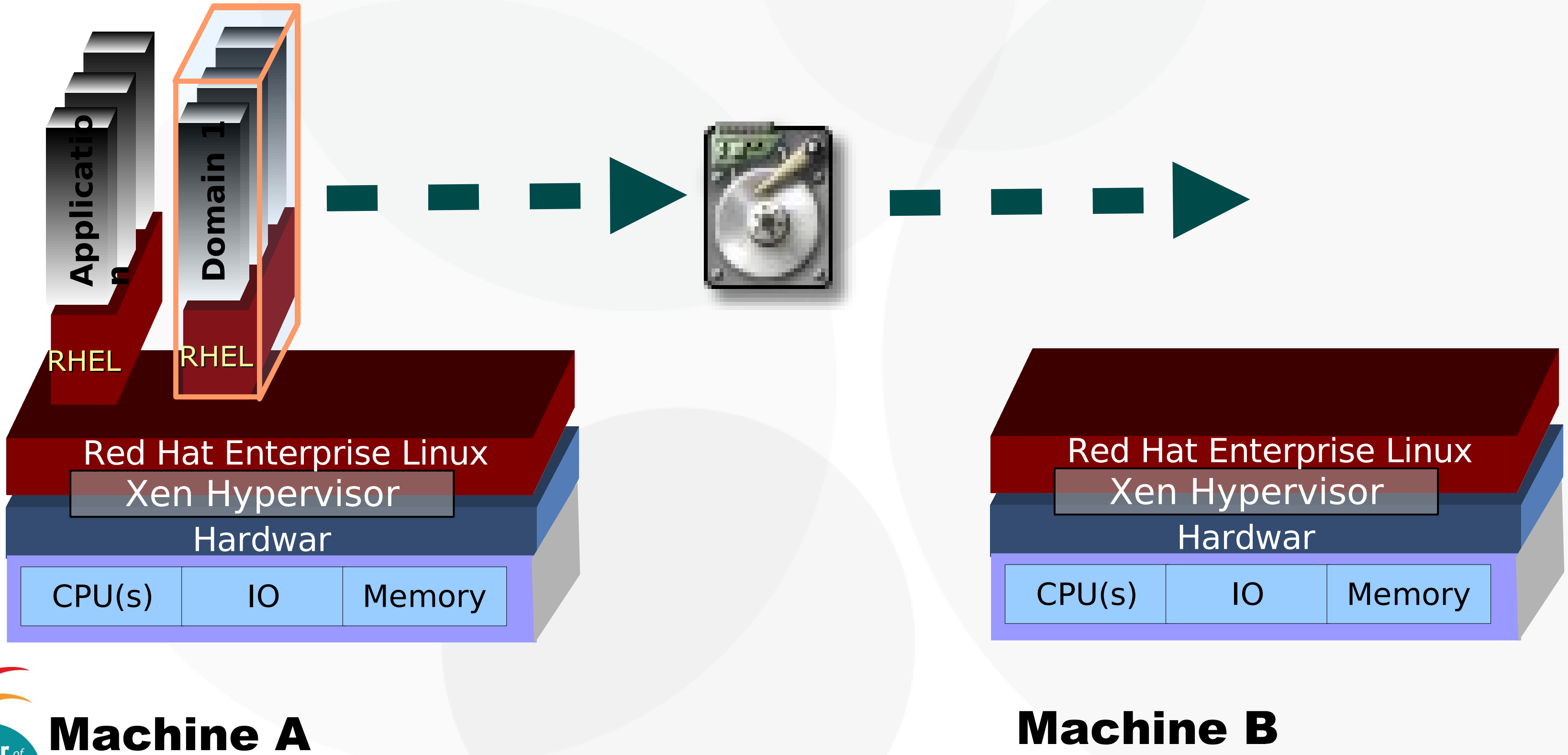
Live Migration

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



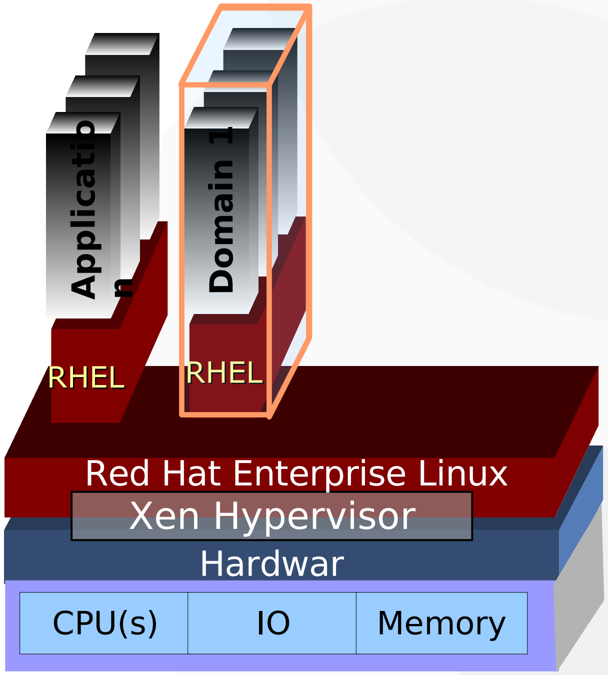
Live Migration

Step 1 :
Mirror block devices (disk) on Machine B

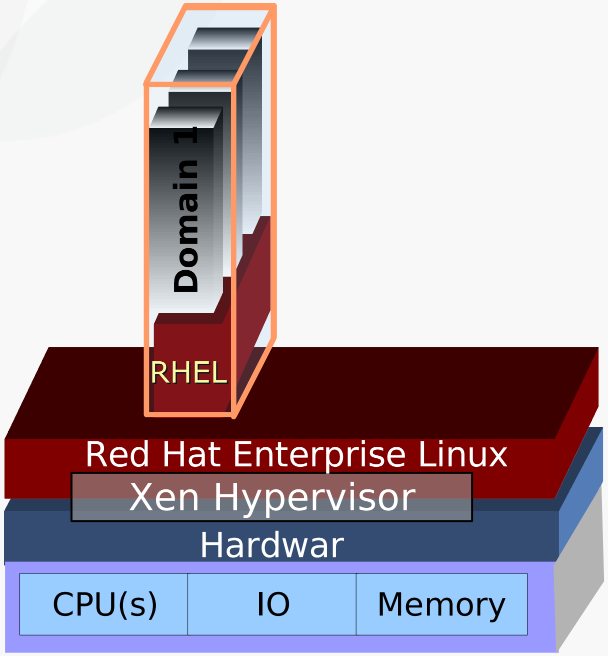


Live Migration

Step 2 :
Initialize container on Machine B



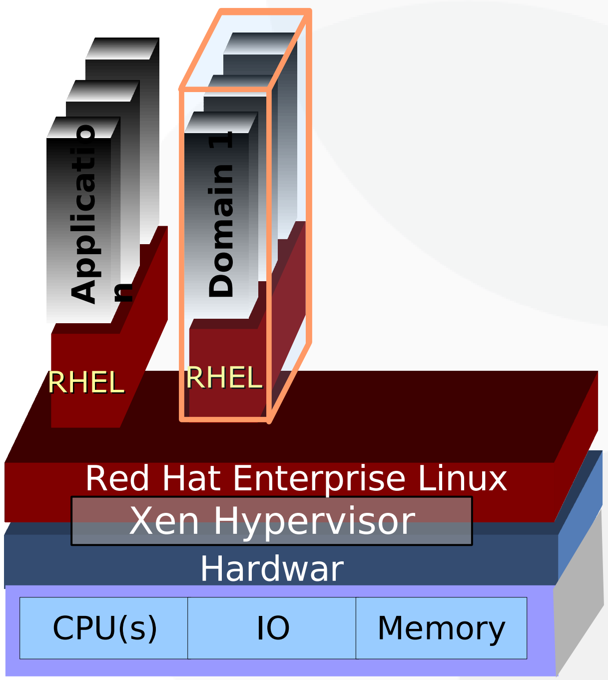
Machine A



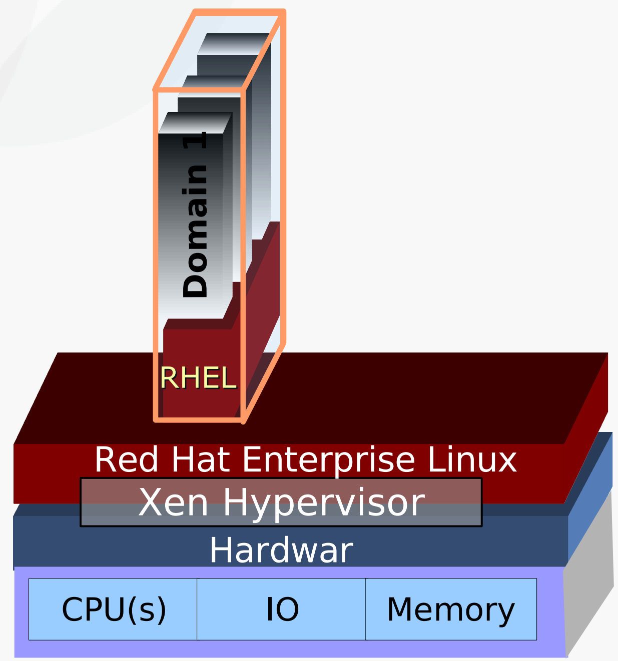
Machine B

Live Migration

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



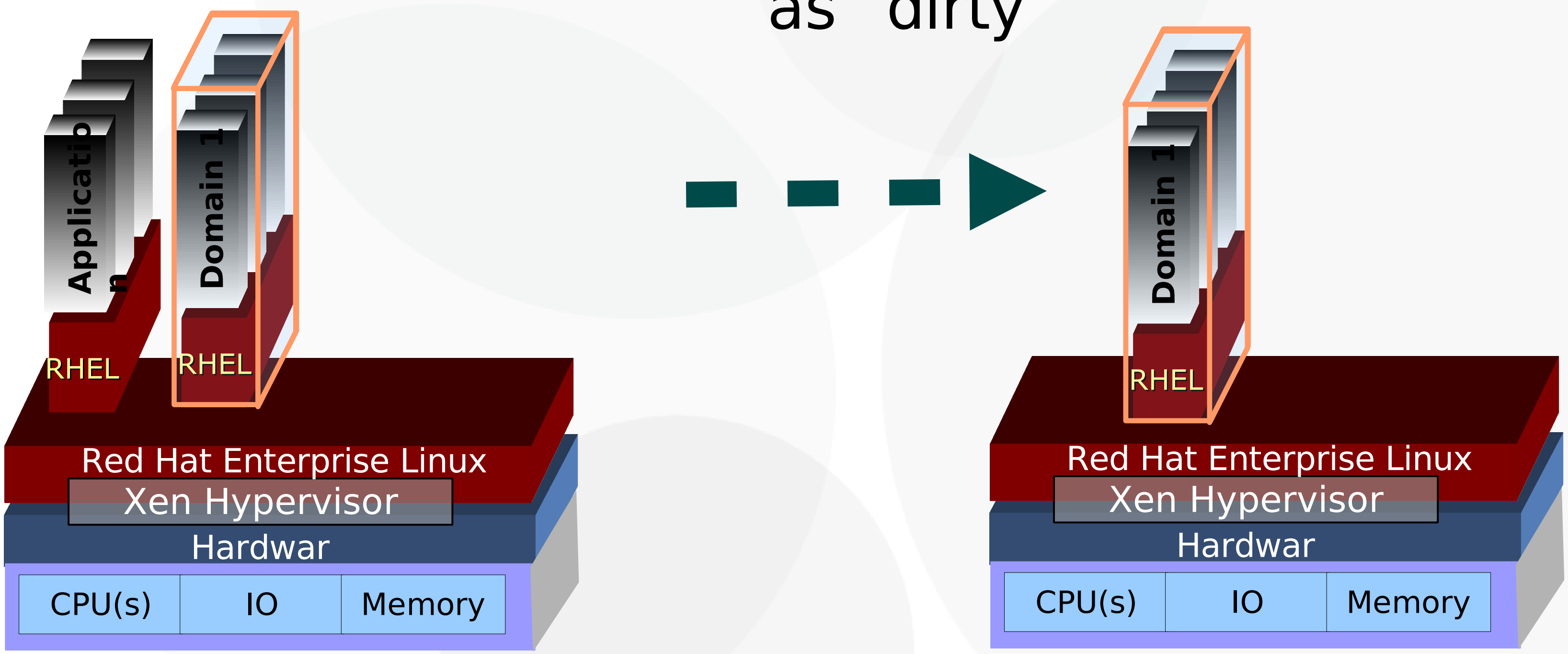
Machine A



Machine B

Live Migration

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

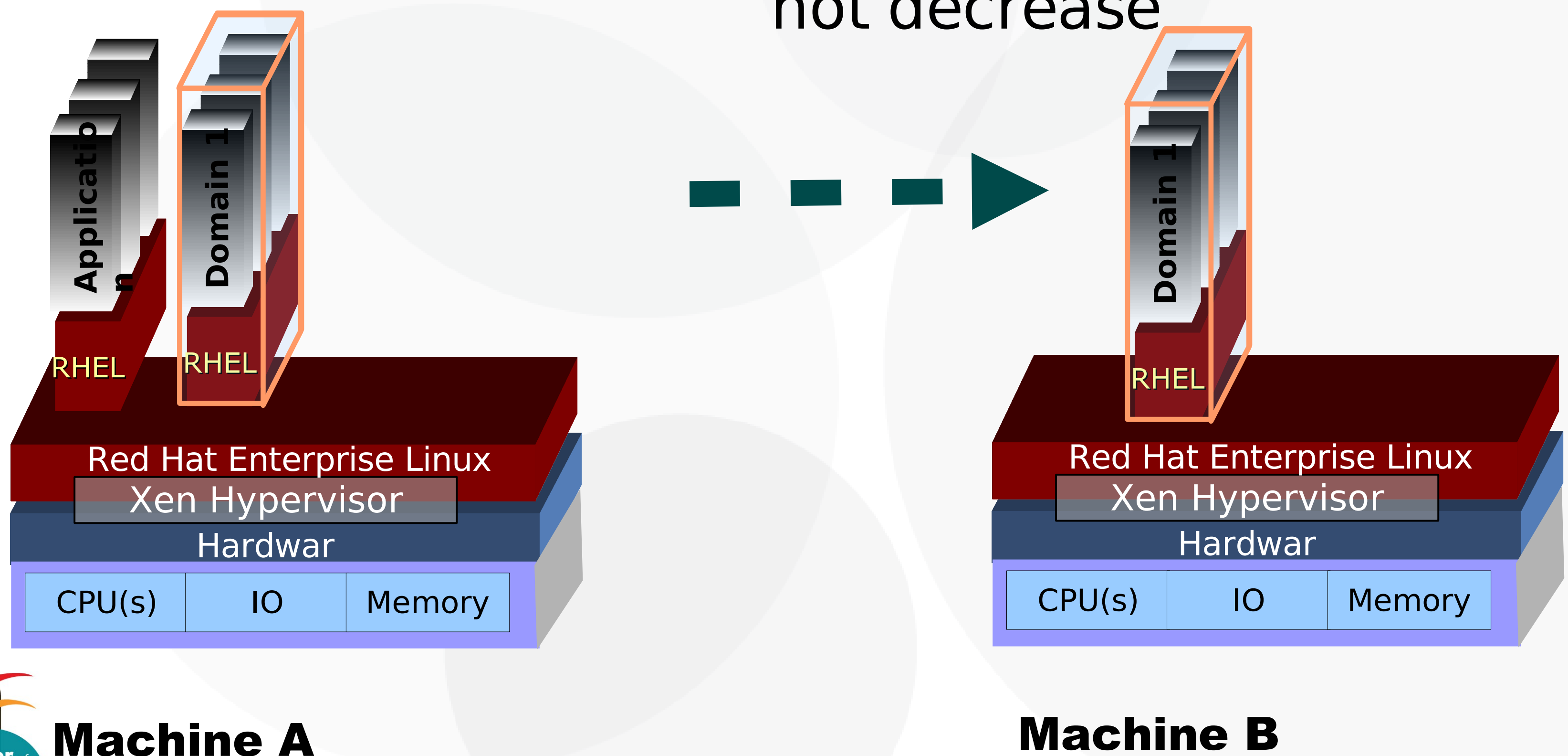


Machine A

Machine B

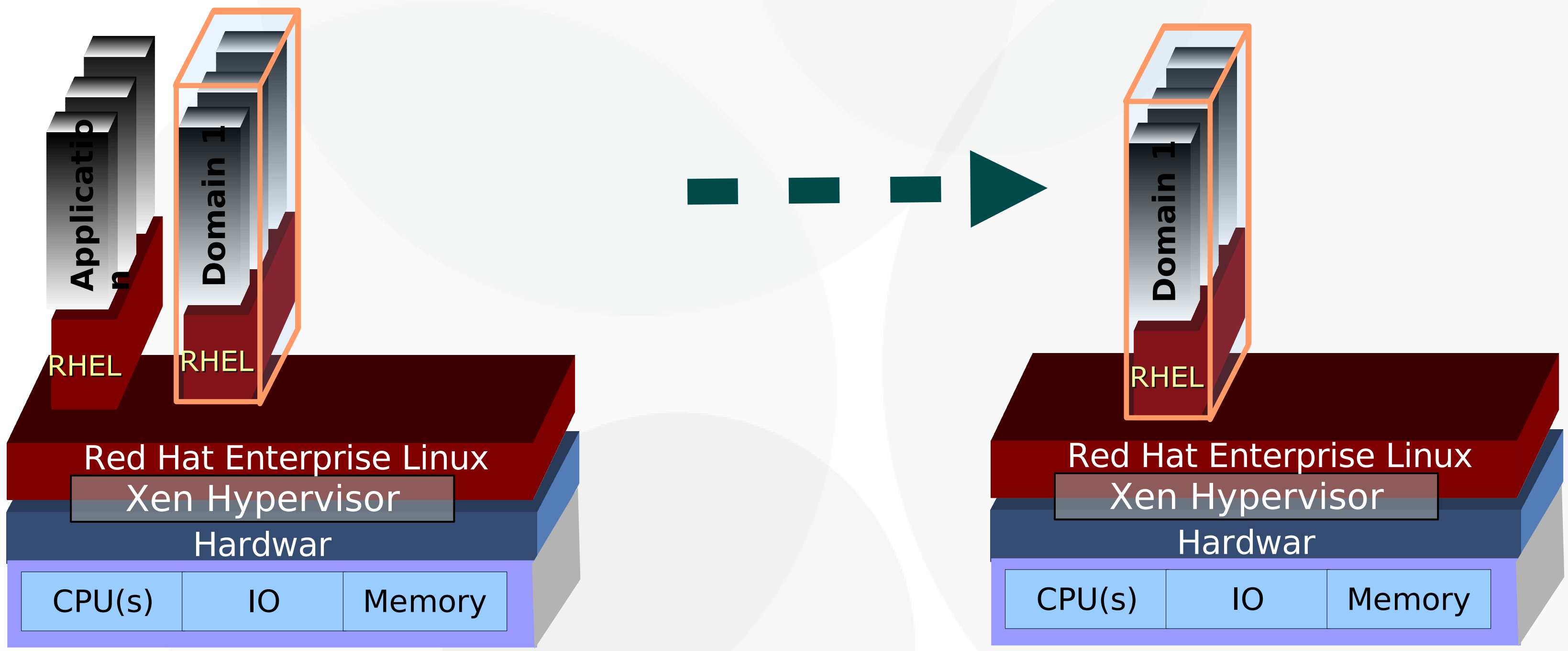
Live Migration

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



Live Migration

Step 6 :
Domain 1 is suspended on Machine A. Remaining “dirty” pages copied

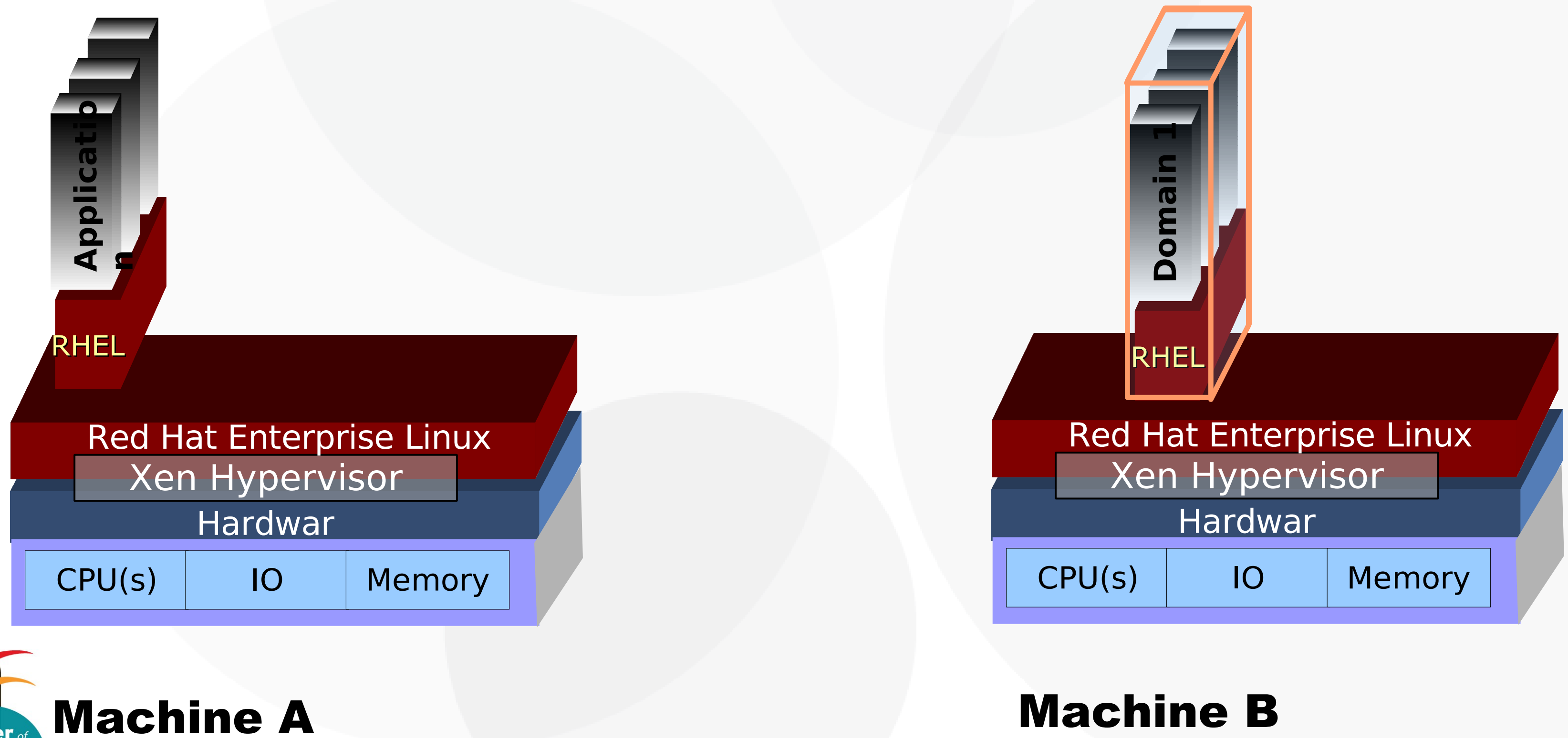


Machine A

Machine B

Live Migration

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

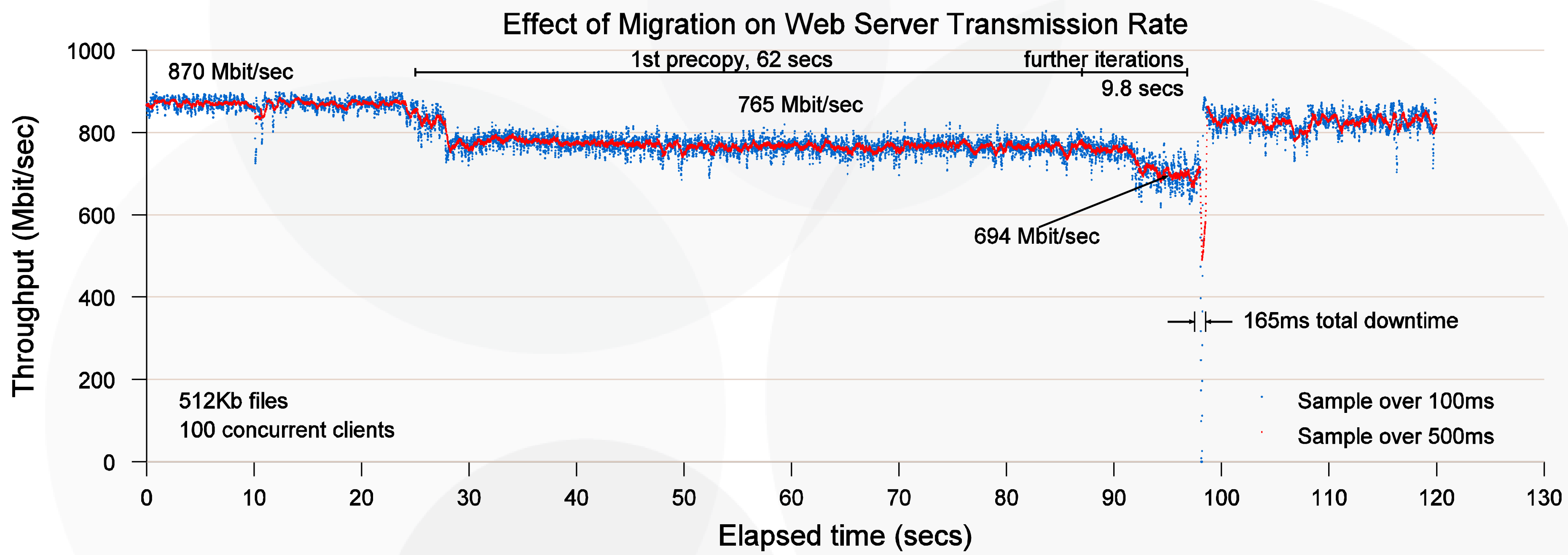


Power of Open Source Architecture
OPEN SOURCE SYMPOSIUM 2006

Live Migration

- Migration requires high speed network connectivity
  - Same Layer 2 network preferred
- Application profile effects migration performance
  - High number of dirty pages -> Longer to transfer
    - Potential for longer downtime
- On average 60 -> 300ms downtime
- Changes to disk image during transfer need to be handled
  - Recommend using Shared file system
    - GFS, SAN / NAS

Migration Performance



Source: XenSource

Management API

- libvirt
  - Stable API for tool/app development
    - CIM providers
    - Python, C bindings, scriptable
  - Hypervisor agnostic (Xen, QEMU, ...)
  - Local VM functionality
    - Start, stop, pause, ...
    - Support for hot and cold migration
  - **http://www.libvirt.org**

Red Hat's Added Value

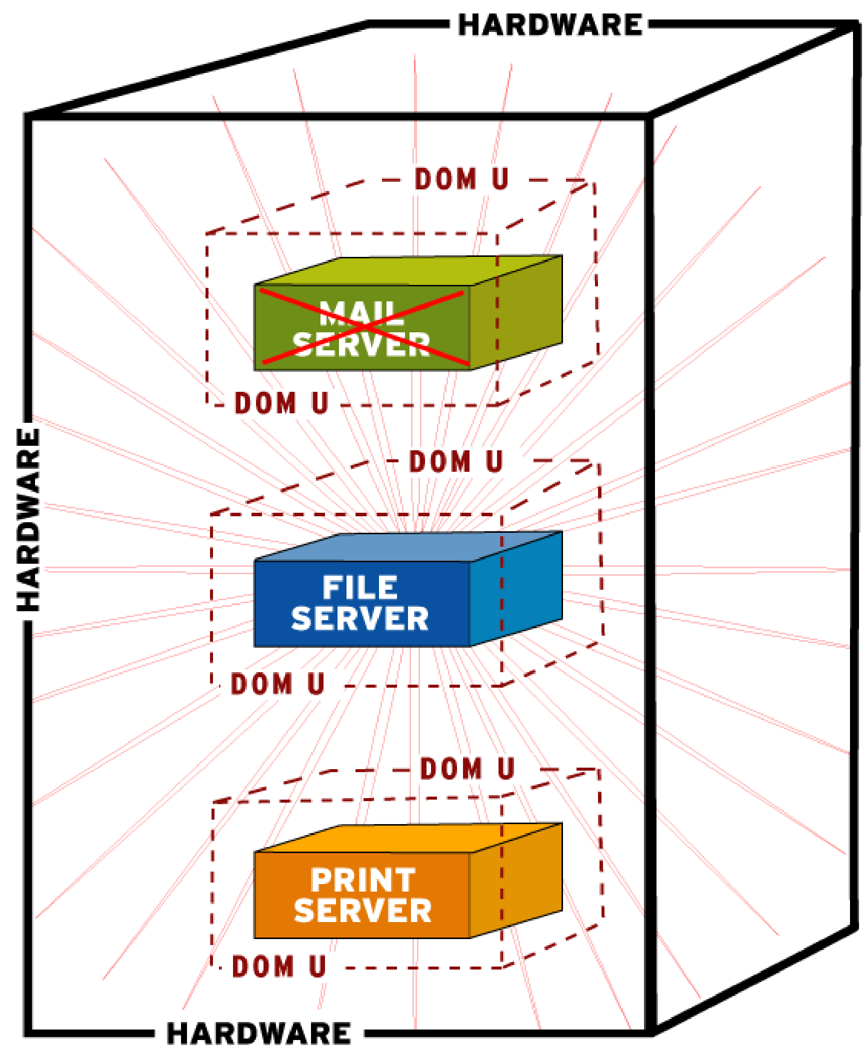
- Server/operating system virtualisation
  - Xen (integrated into kernel and OS platform)
- Storage virtualisation
  - Red Hat Global File System/CLVM
- System management, provisioning, resource management
  - Red Hat Network, libvirt
- Application environment consistency with non-virtualised environments

Red Hat's Added Value

- Installation tools
  - Anaconda
    - The “Red Hat Installer” is virtualisation-aware.
    - Eases virtualisation setup and installation
- ISV and IHV Certification
  - World's leading open source Linux provider has the largest network of certified software applications and hardware systems

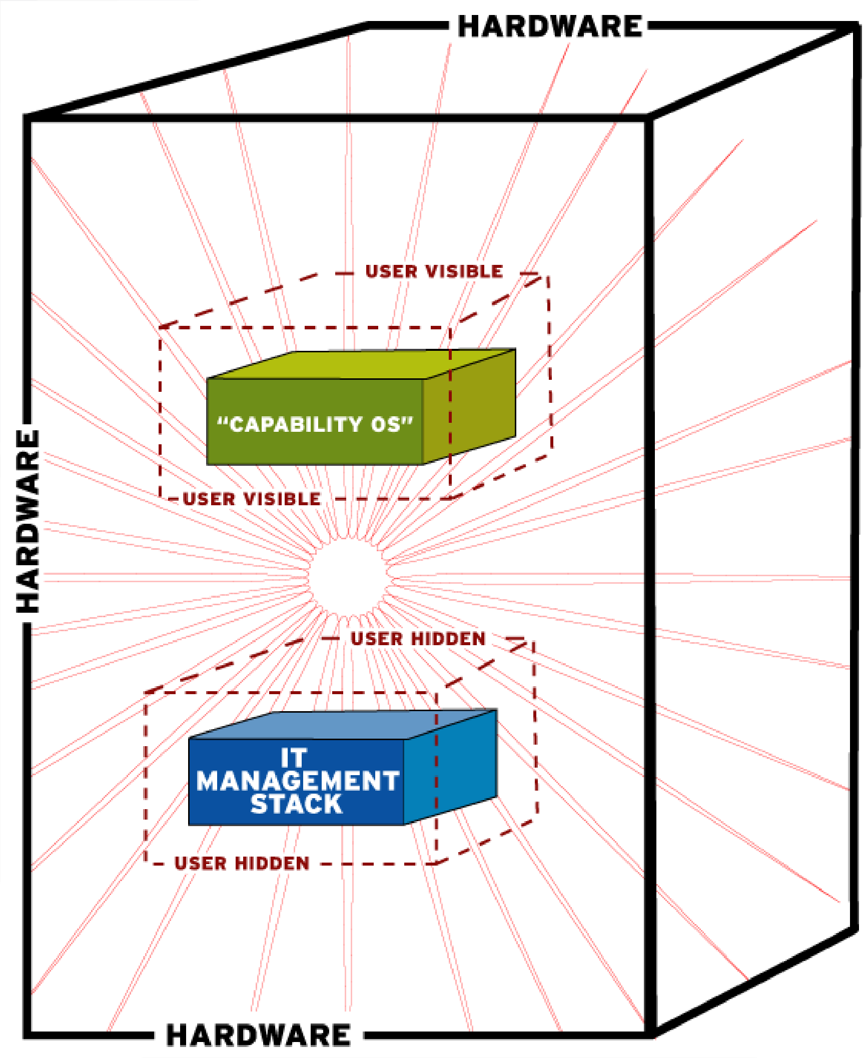
Solving Real Problems

- Failure Isolation
  - **Failing mail server does not impact the other servers.**
    - Prevent major crashes.
    - In the event of a security failure, contain leaks or theft.



Solving Real Problems

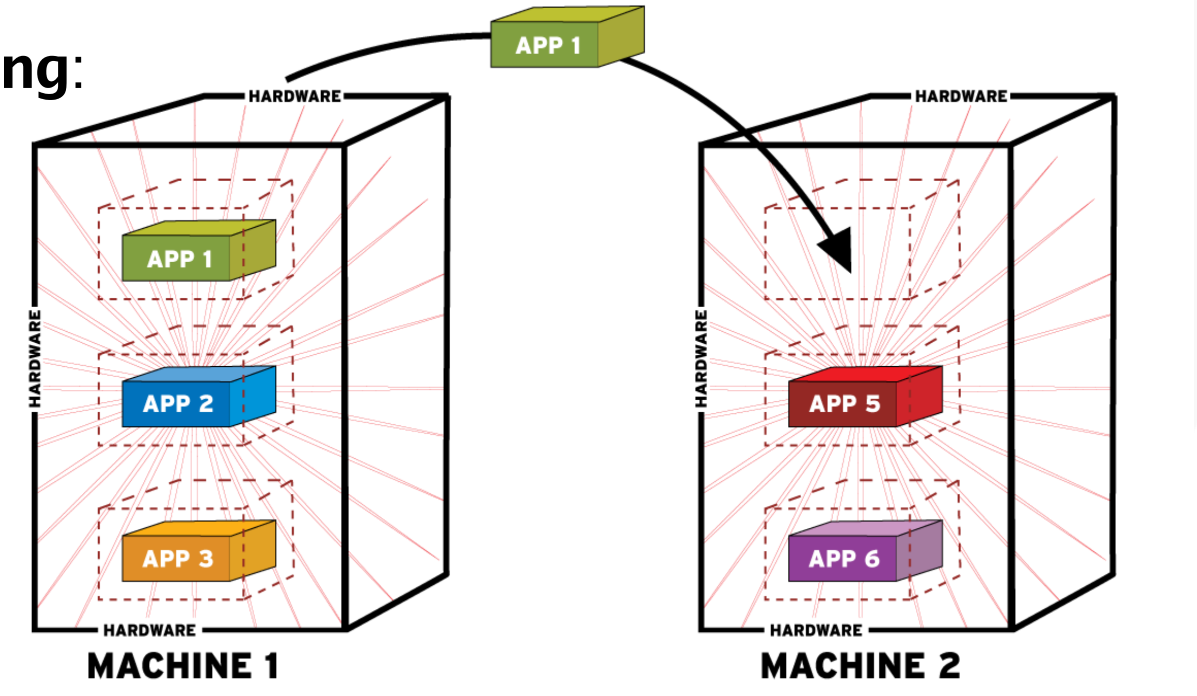
- Control without constraints
  - **IT locks down one guest, user is empowered to manage the other.**
    - The value of user-based innovation.



OPEN SOURCE SYMPOSIUM 2006

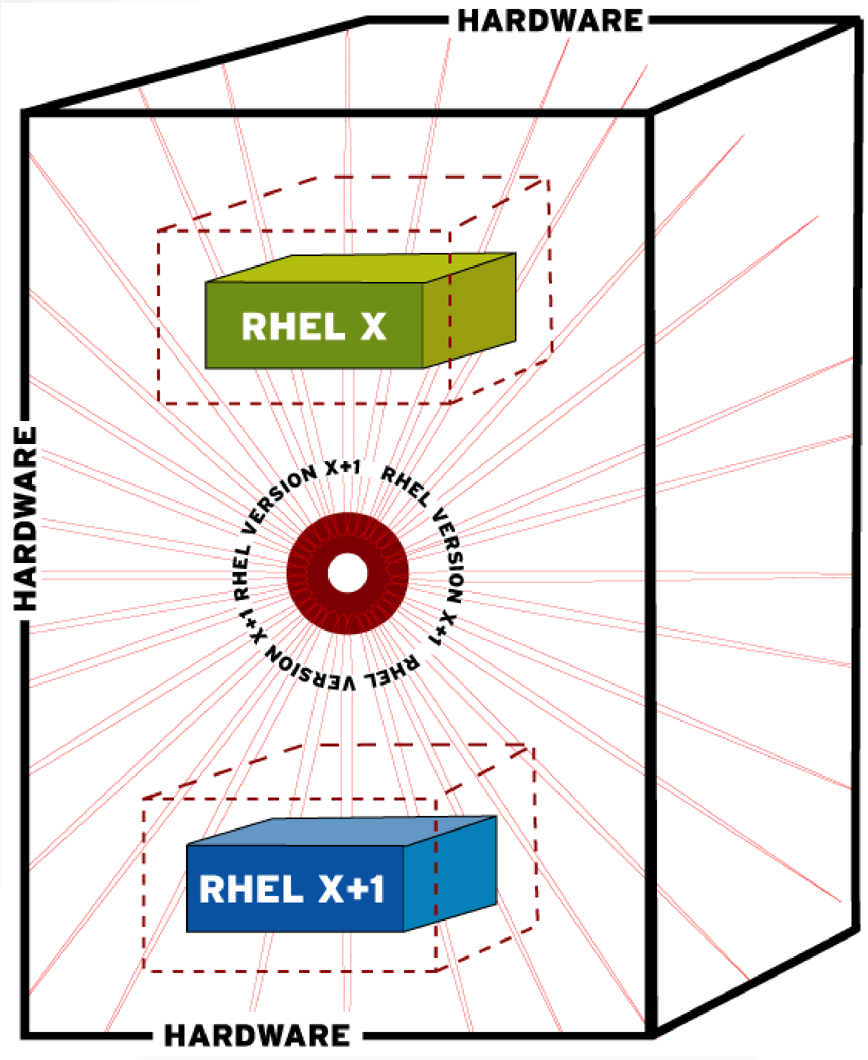
Solving Real Problems

- Live migration
  - **Virtual Machine relocation enables**
    - **High Availability**: machine maintenance
    - **Load Balancing**: statistical multiplexing gain



Solving Real Problems

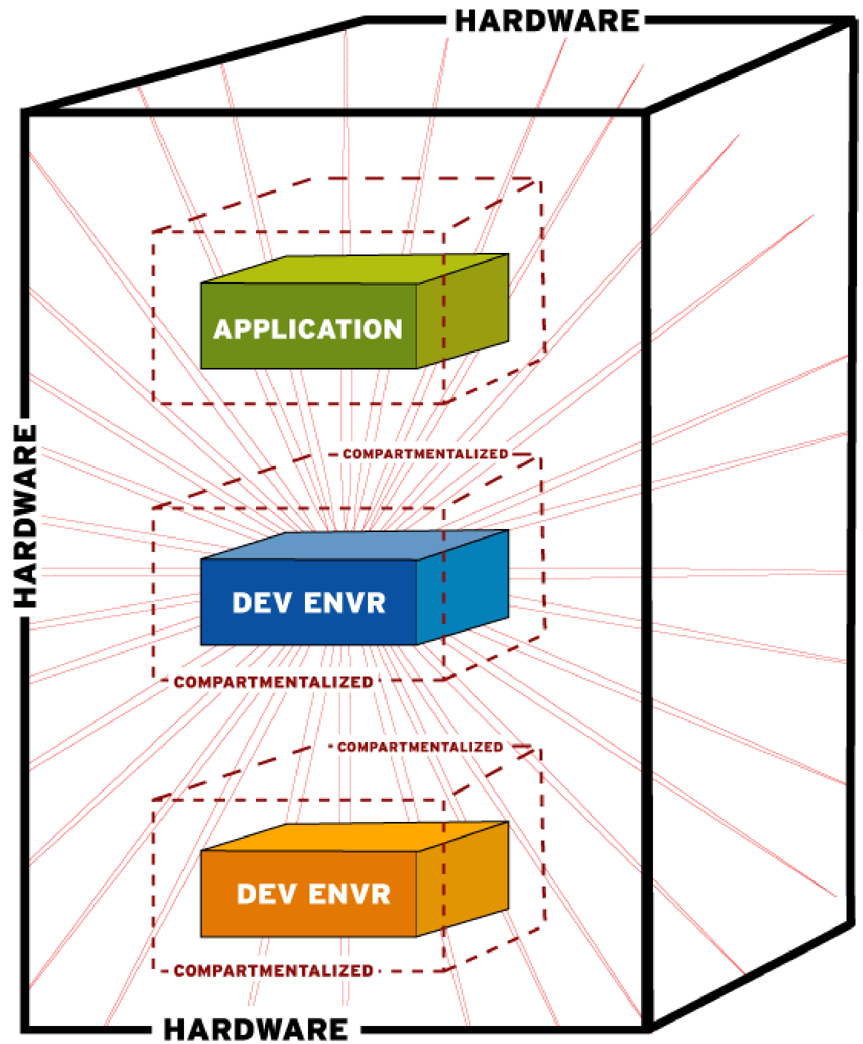
- Freedom from upgrades
  - Preserve the version X environment and its applications, deploy on version X+1 when it makes sense.
  - The hypervisor runs on version X+1 to gain maximum benefit from the new hardware and software.



OPEN SOURCE SYMPOSIUM 2006

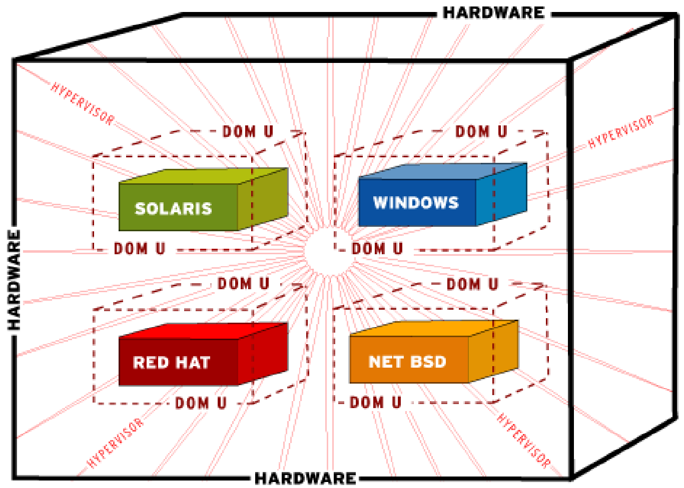
Solving Real Problems

- Development and QA environments
  - Secure and compartmentalized instances; think "chroot" jail.
  - Simplify test scripting and execution for qualifications.
  - Simplify test simulation.
  - Carve out resources and return when finished.

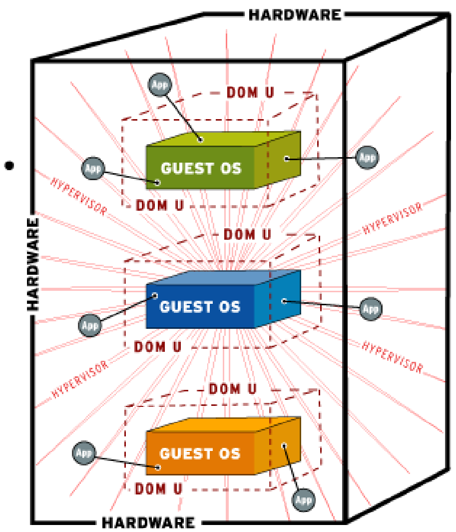


Consider the Possibilities

1.

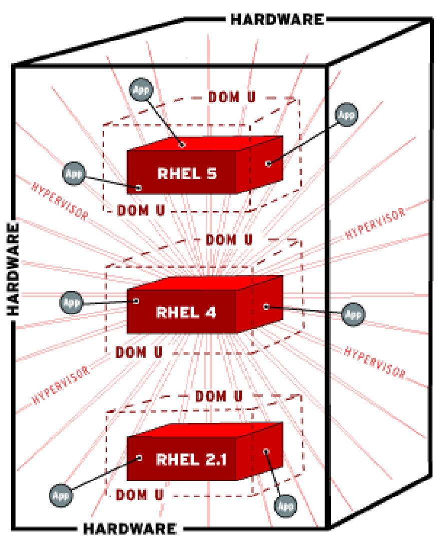


2.

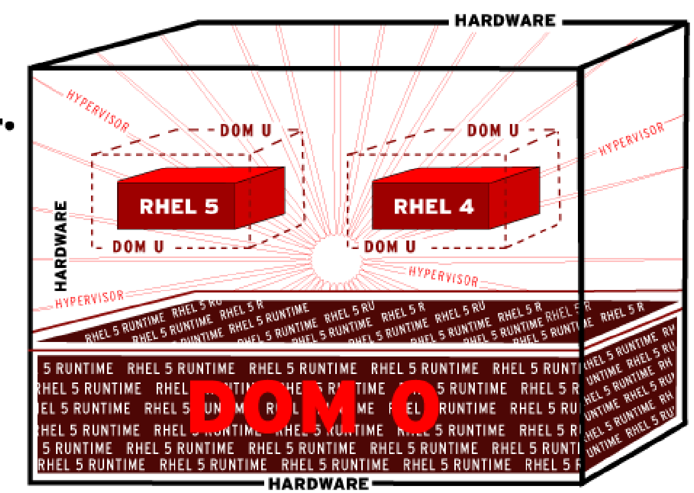


[CHOICE]

3.



4.



Sung Min David Joo
djoo@redhat.com