

# VIRTUALIZATION

**Carlos Piedrafita**

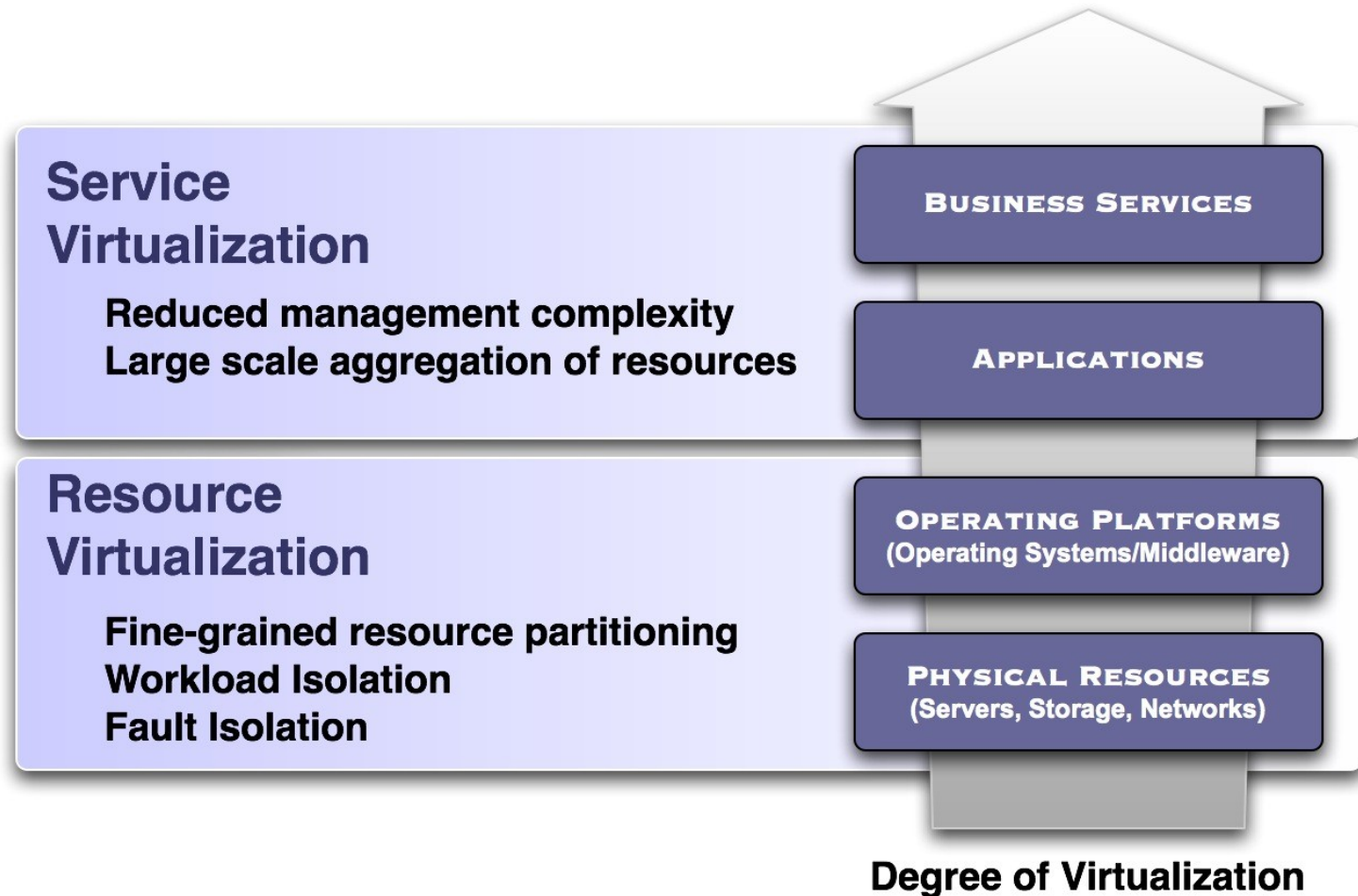
Solution Architect

Sun Microsystems Iberica

# Virtualization

- “A software abstraction layer added onto some entity so that the new entity exhibits the interface properties of the original or a normalized interface broadly similar to that of the original...”
  - *Enterprise Grid Alliance*
- Virtualization layer...
  - > Hides the true implementation of the virtualized entity
  - > Enables the original to be changed or replaced without impacting the entities that have dependencies on it
  - > Enables a single entity to be partitioned and presented as a set of virtual entities
  - > Enables a set of discrete entities to be treated as a single aggregate entity

# Evolution Of Virtualization In The Data Center



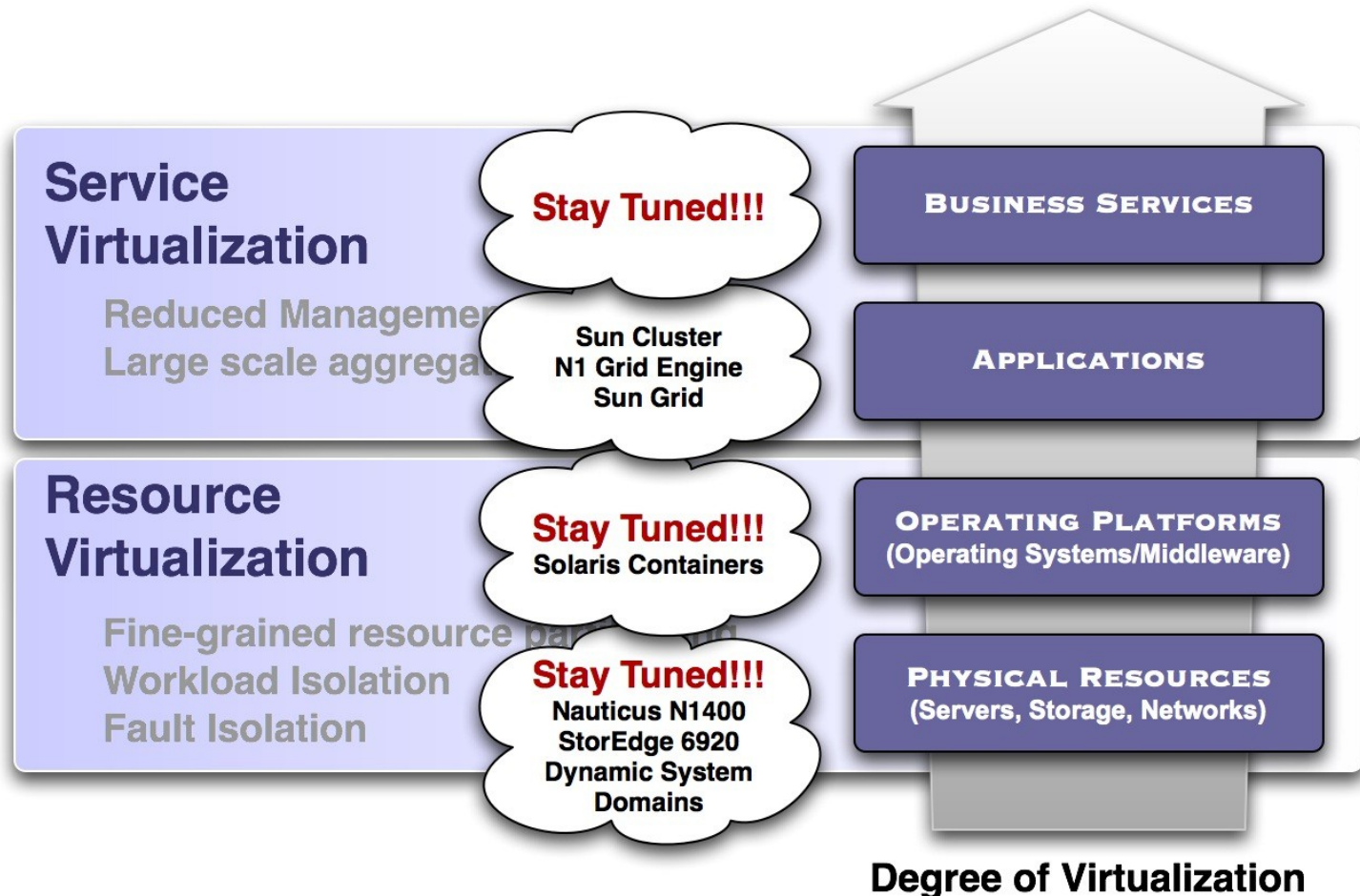
# Evolution Of Virtualization In The Data Center

- Today...
  - > Most solutions focused on **resource virtualization** – servers, storage, networks, and operating systems
  - > Some focus on application virtualization
  - > Lack of integration
  - > Heavy reliance on consulting services to get everything rights
- Tomorrow...
  - > Resource virtualization is being standardized and commoditized
  - > Seeing demand/focus shift from resource virtualization to **business service virtualization**
- Key Requirements:
  - > Management of services on top of large scale, highly distributed and complex deployment environments
  - > Seamless integration with existing virtualization and system/service management solutions is highly desired
  - > End-to-end consistency is **key**

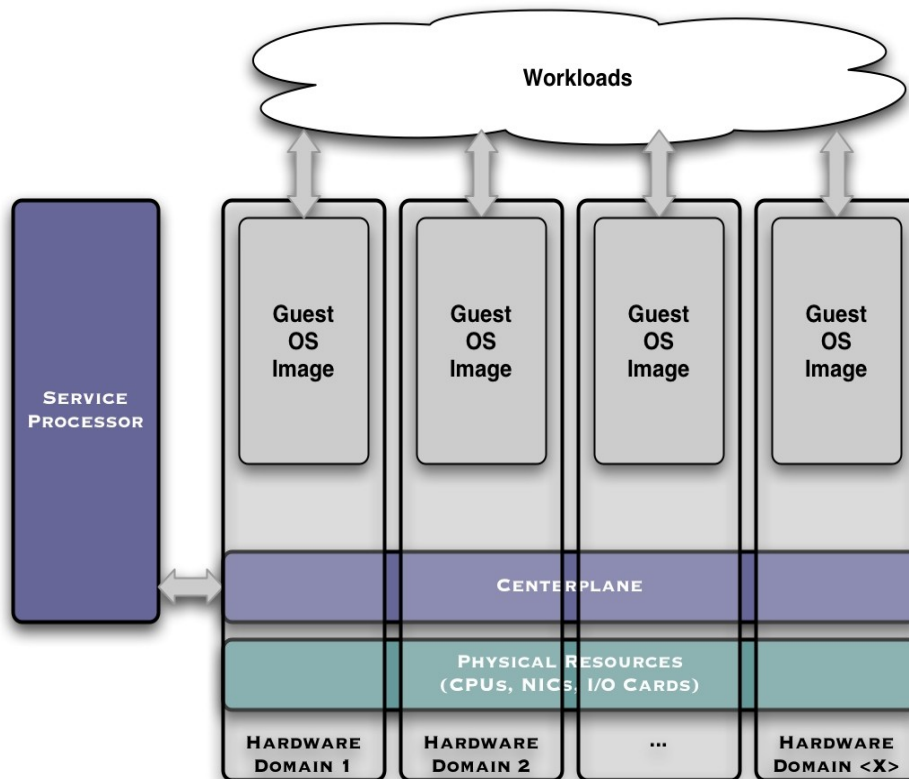
# SMI Virtualization Story

- Virtualization is an integral part of Sun's system management vision and strategy
- Sun is already shipping a number of offerings and solutions that enable bare metal, server, OS and application deployment virtualization:
  - > **Hardware Virtualization:** Dynamic System Domains
  - > **Storage Virtualization:** StorEdge 6920,
  - > **Network Virtualization:** N1000 Secure Application Switches
  - > **OS Virtualization:** Solaris Containers, Solaris support for Xen & VmWare
  - > **Application Virtualization:** Sun Cluster, N1 Grid Engine
- Sun's goal is to enable end-to-end virtualization of the entire data center management stack, from servers to services
- Stay tuned for more announcements in this space...

# Current SMI Virtualization Offerings

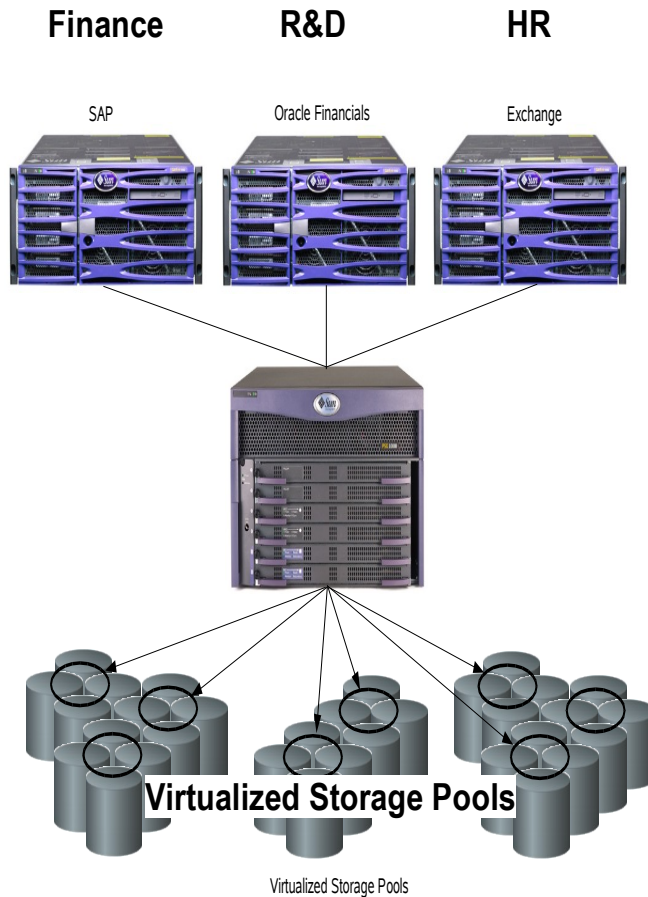


# Server Virtualization – Dynamic System Domains



- A physical server hardware virtualization solution
- Partitions a single physical system into one or more “domains”
- Service processor assigns physical hardware resources to domains
- Resource assignments are done on system board basis

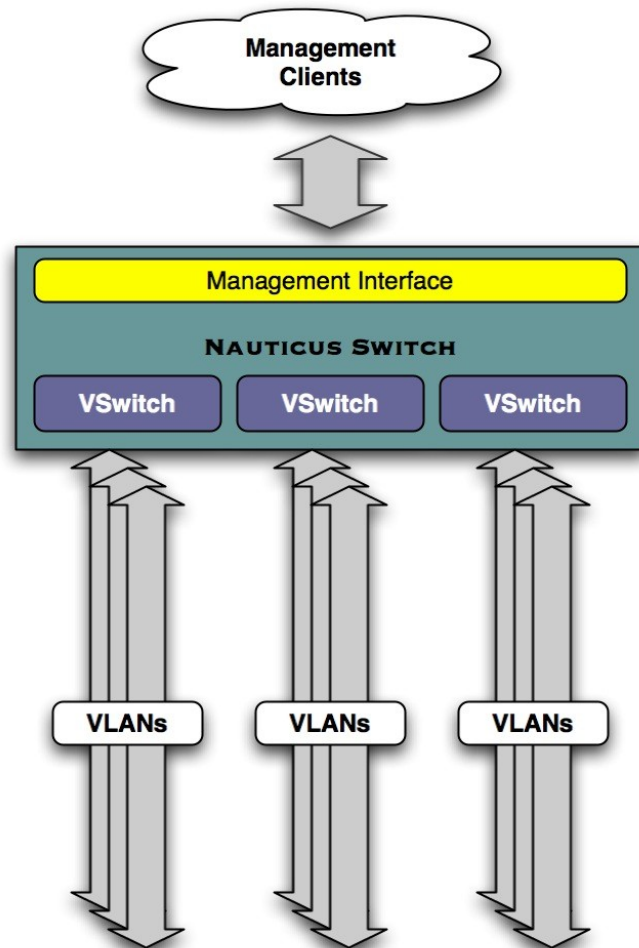
# Storage Virtualization – Sun StorEdge™ 6920



- A storage virtualization solution
- Partitions a pools of physical storage resources into one or more virtualized storage pools independent from each other
- Storage control software is used to administer storage pools

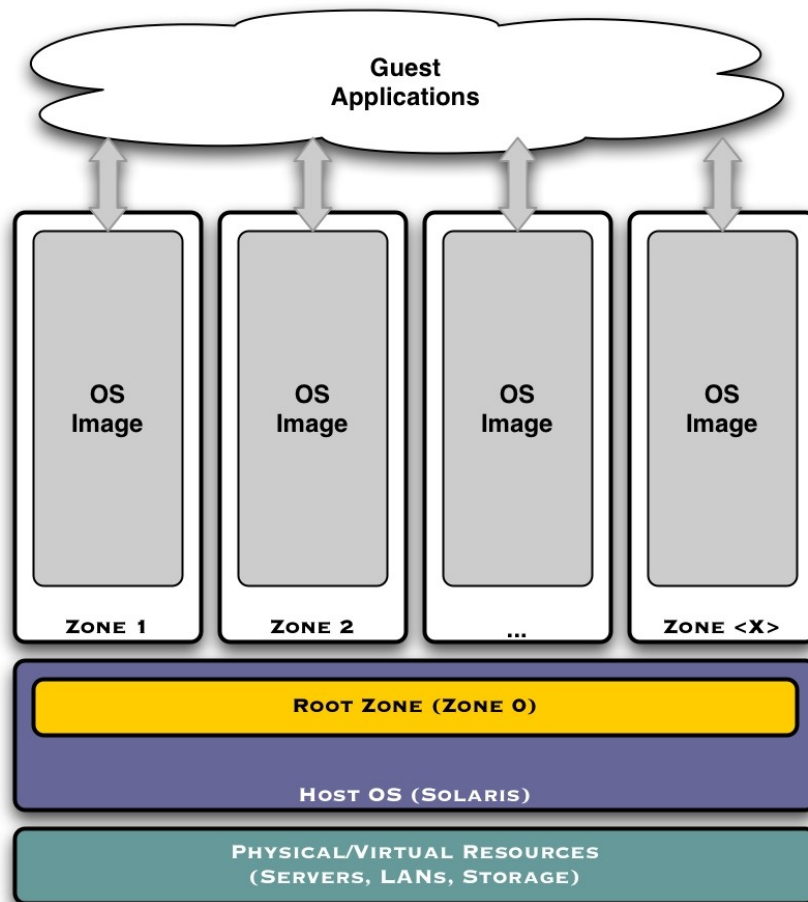


# Network Virtualization – Sun Secure Application Switches



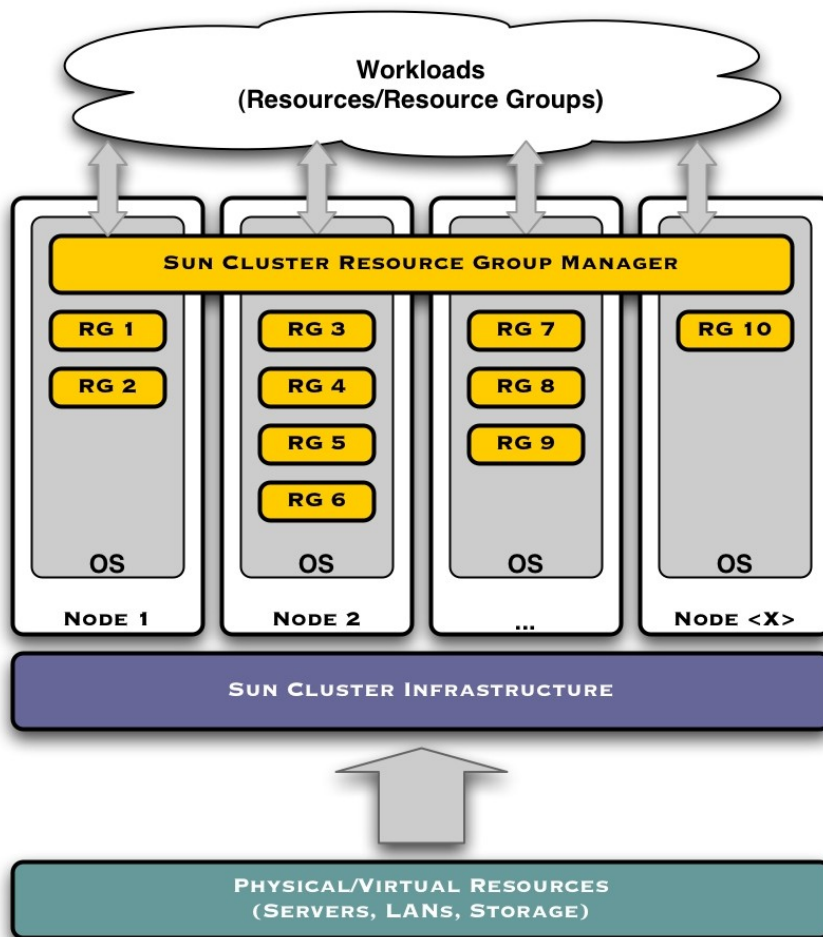
- A network virtualization solution
- Partitions a single physical network switch into one or more virtual switches
- Each virtual switch is fully isolated from the others and has full functionality of a physical network switch
- Provides application load balancing and high performance security

# Server Virtualization – Solaris Containers



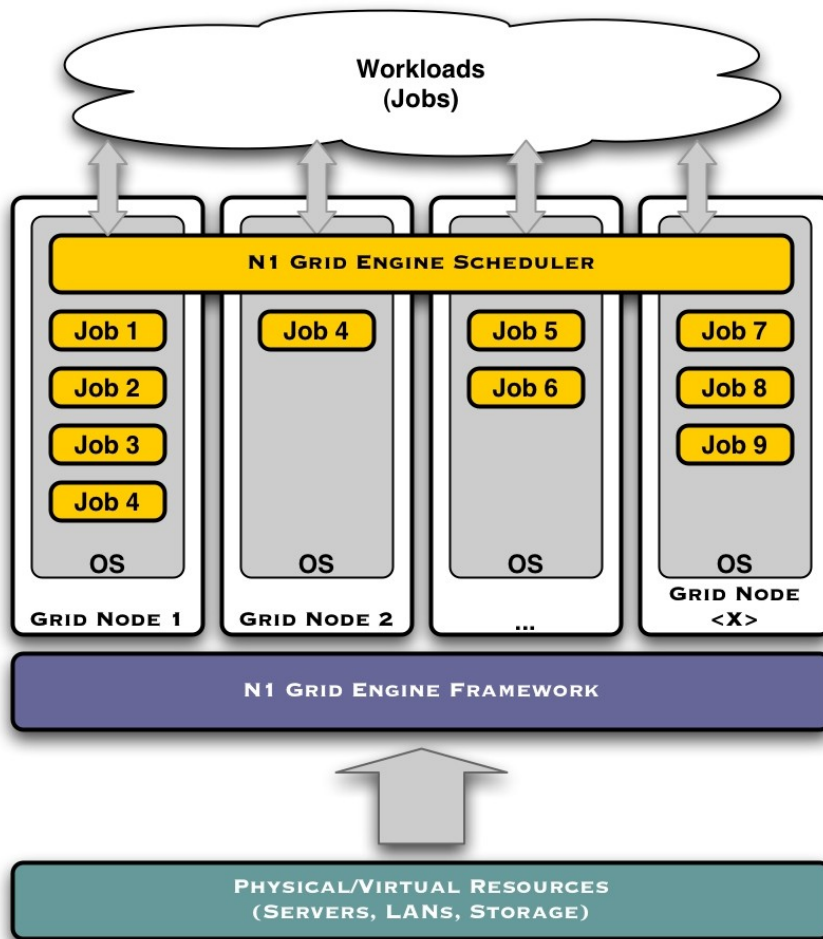
- A Solaris virtualization solution
- Partitions a single Solaris instance into two or more instances, each running in its own “container” (a.k.a. zone)
- Root zone provides means of control and administration

# Application Virtualization – Sun Cluster



- An application virtualization solution with focus on high availability
- Aggregates a set of operating platforms into a HA execution environment for applications
- Underlying cluster resources can be moved/reassigned without impact on deployed applications

# Application Virtualization – N1 Grid Engine



- An application virtualization solution with focus on batch processing
- Aggregates a set of operating platforms into a batch “job” execution environment
- Underlying grid resources can be moved/reassigned without impact on running jobs

# Server Virtualization

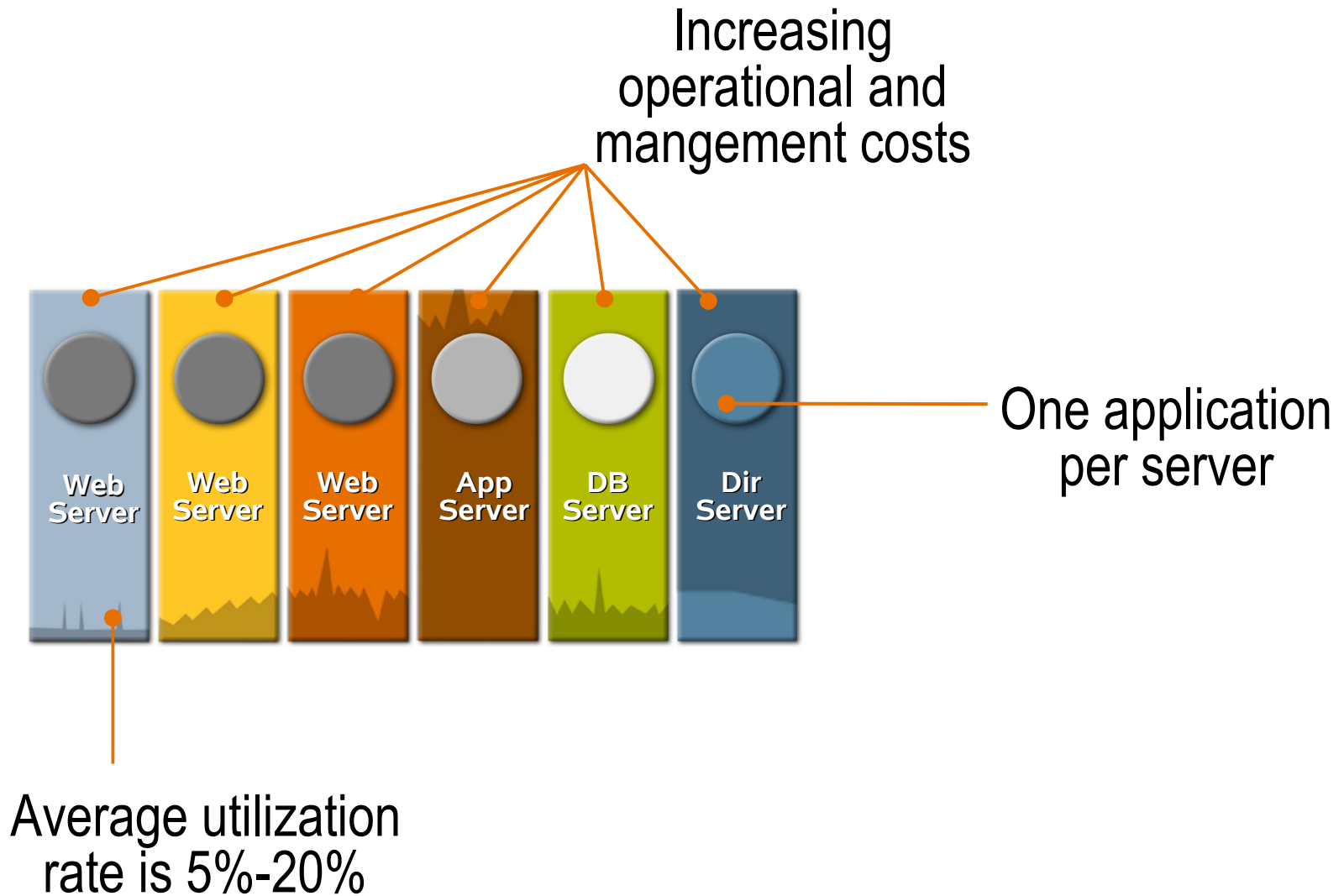
# Consolidation Motivating Factors

- Marketing-speak “Customers are interested in improving the utilization of their computing resources”
  - > Translation: they want better return on the money they've spent
- One way to do this is server consolidation
- Consolidation requires (at least) the following
  - > **1. Non-interference of independent workloads: the need driving server virtualization.** Consolidation also requires...
  - > 2. Resource management (to ensure service levels are met)
  - > 3. Resource accounting (to pay for shared resources)
  - > 4. RAS services (since eggs now in fewer baskets...)

# Other Motivators / Use Cases

- There are other important uses for virtualization
  - > Upgrade OS version or patch level with concurrent operation
  - > Migrate from one OS to another
    - > Classic mainframe example: migrate from VSE to MVS
  - > Coexist multiple OSES for different types of work
    - > Eg: MVS and Linux on zSeries, or Linux and Win2K on x86
  - > Provide separate fault, security, admin domains of same OS level
  - > Relieve scalability constraints of a given OS via multiple instances
  - > Use legacy OS on newer kit (ex: run NT4 on current x86)
  - > OS development, sandbox, play-pen in congenial environment
- These use cases are frequently seen in customer datacenters – don't forget to consider them as well

# Situation Today

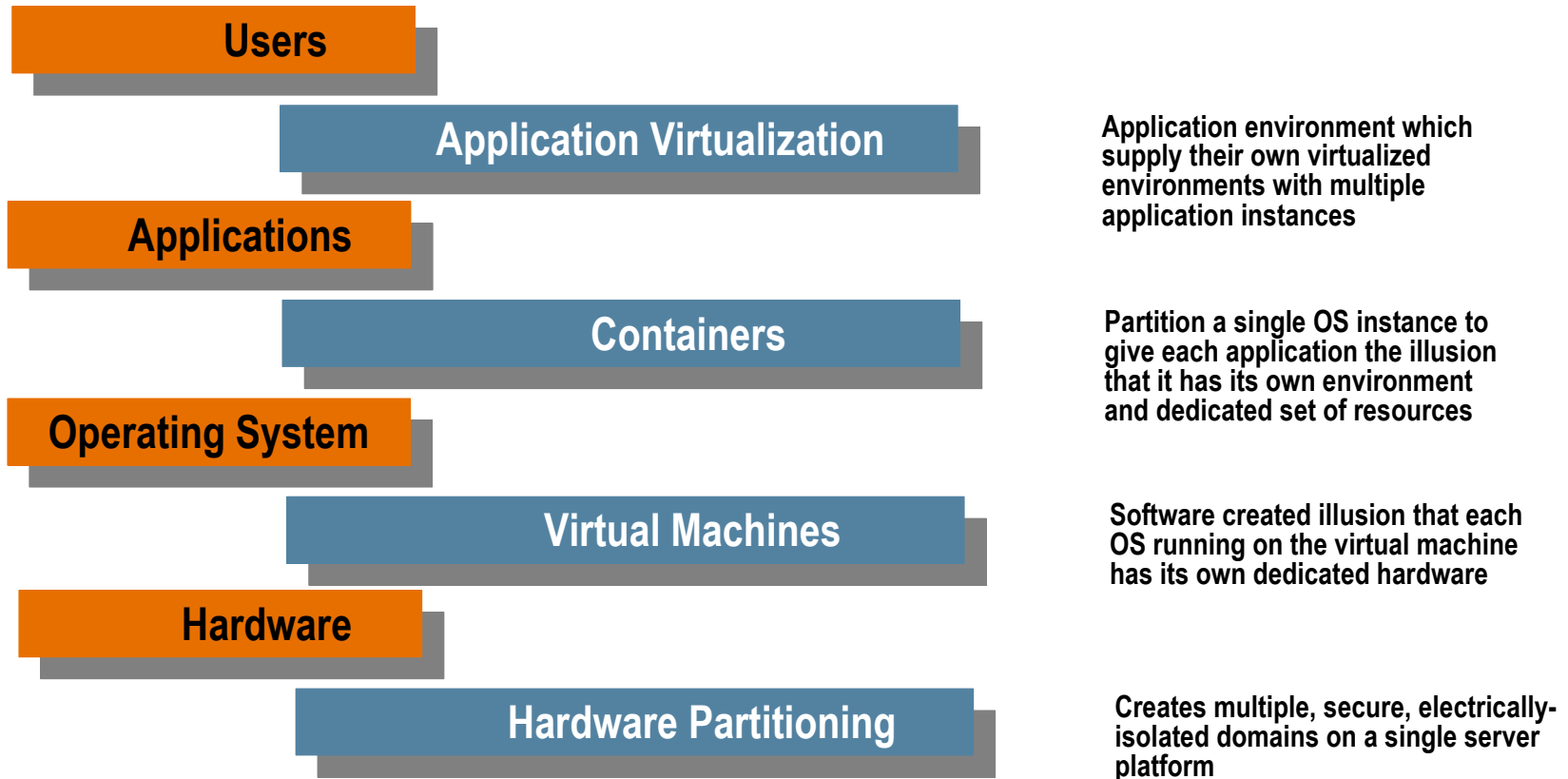




# Situation Today

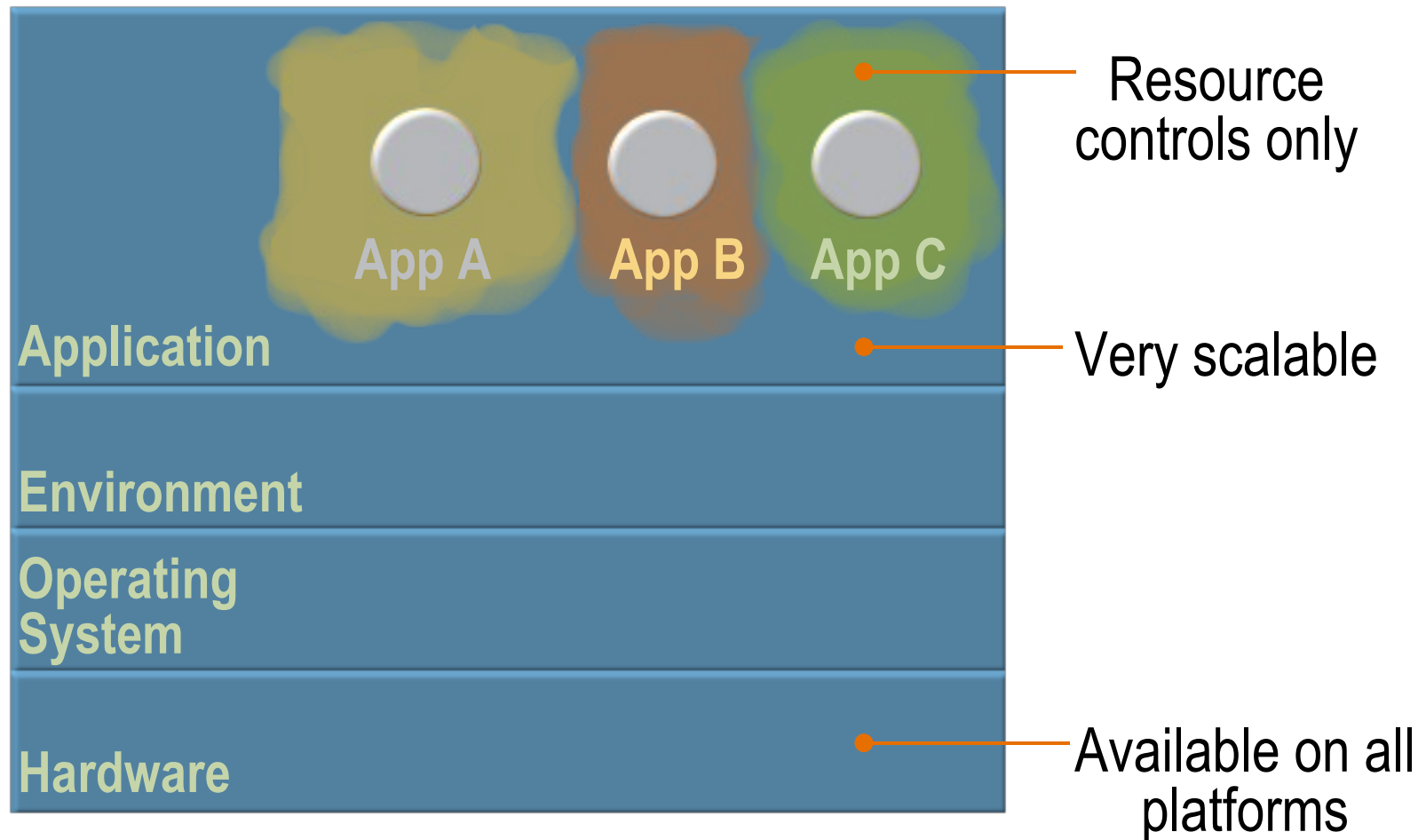
- Individual applications or workloads deployed on dedicated, separate systems
  - > For security, isolation, performance reasons etc.
- Customers want to increase asset utilization and reduce costs
  - > Average utilization in datacenter below 15%
- Key trends
  - > Operational and management costs increasing
  - > HW costs gradually becoming less significant
- Customers are looking at consolidation as a solution

# Virtualization and Partitioning

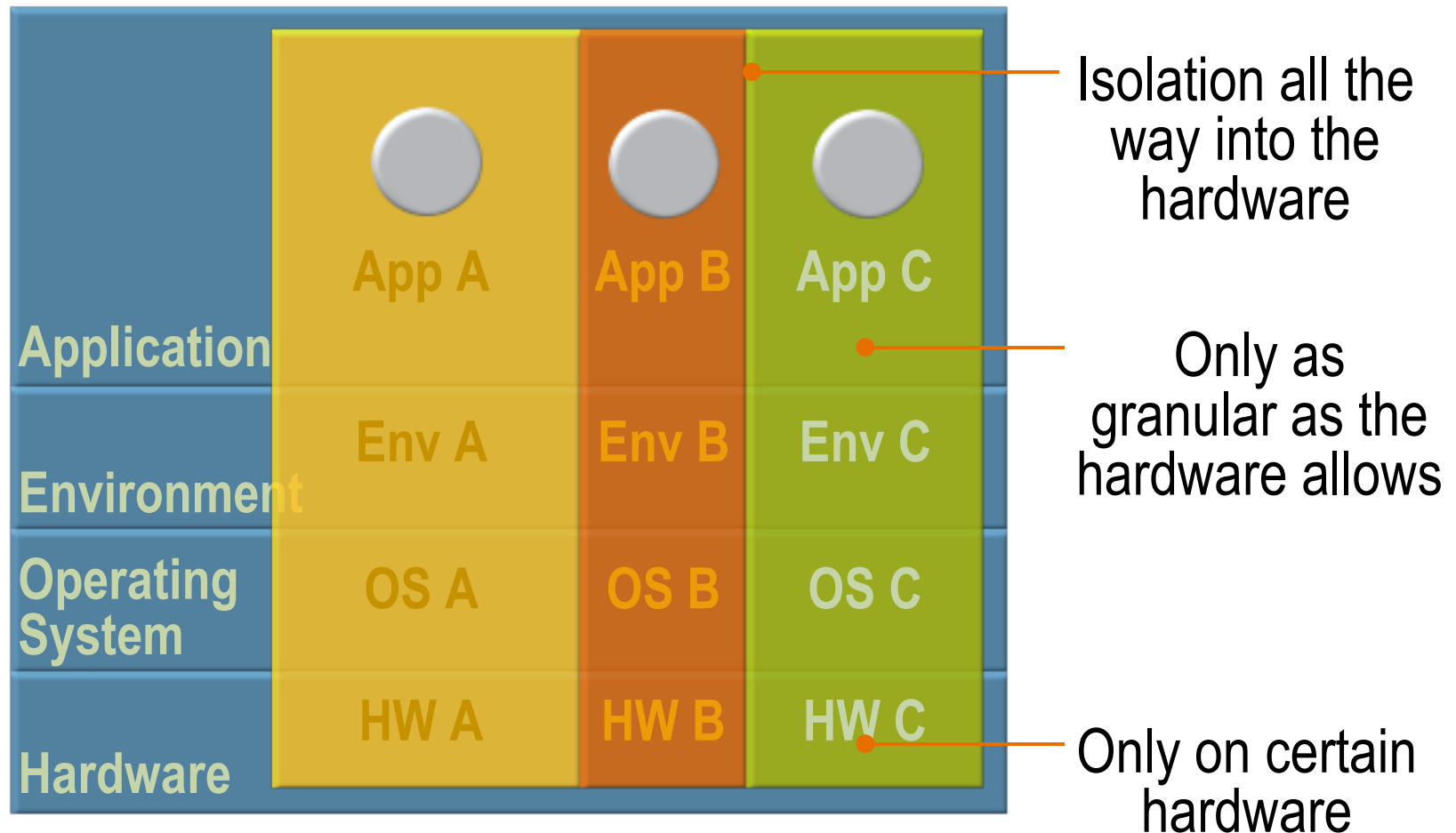


**Virtualization technologies between layers of the hardware and software stack give the illusion of a dedicated environment to the layer above.**

# Resource Management



# Hard Partitioning



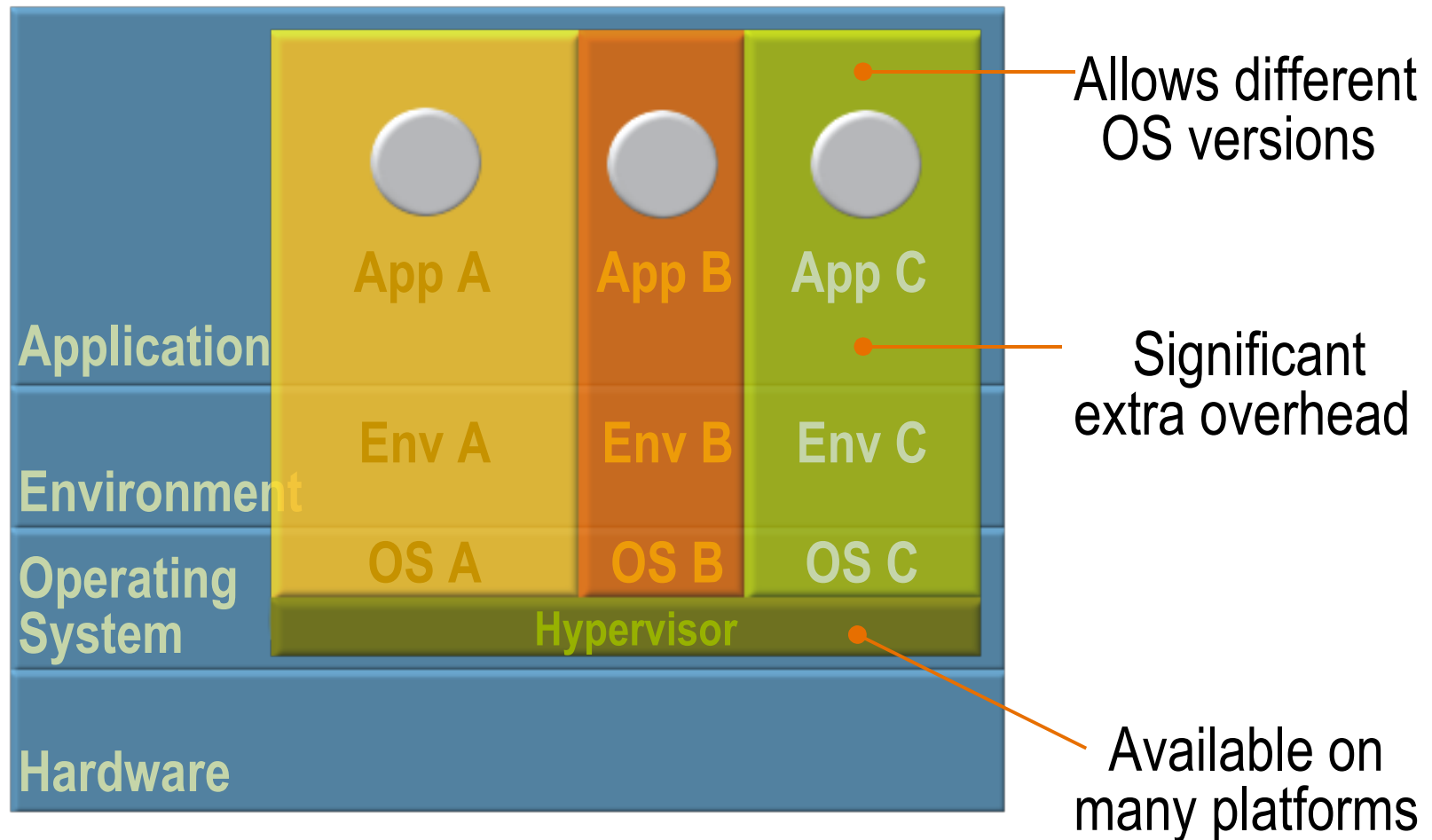
# Partitioning key points

- Typically is highly reliable (including in customer estimation)
  - > Relatively small (compared to a full-function OS) firmware or hardware solution. Not usually viewed as single point of failure
- Doesn't permit large number of OSES
  - > In all cases since each partition/domain requires dedicated RAM
    - > Eg: a 32GB machine can host no more than four 8GB instances
  - > Other constraints based on vendor and architecture
- Some implementations (eg: IBM LPARs) let multiple OS instances share CPU capacity
  - > Example: 3 copies of z/OS share a pool of 4 CPUs
  - > Overhead varies: some overhead (3-10%) with zSeries LPARS
  - > High overhead with IBM pSeries micropartitions – as much as 40%

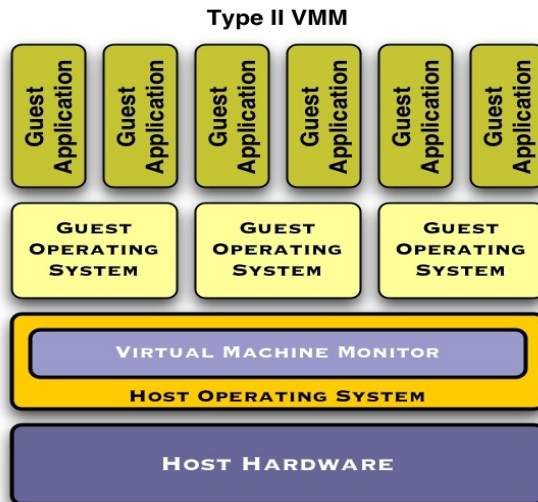
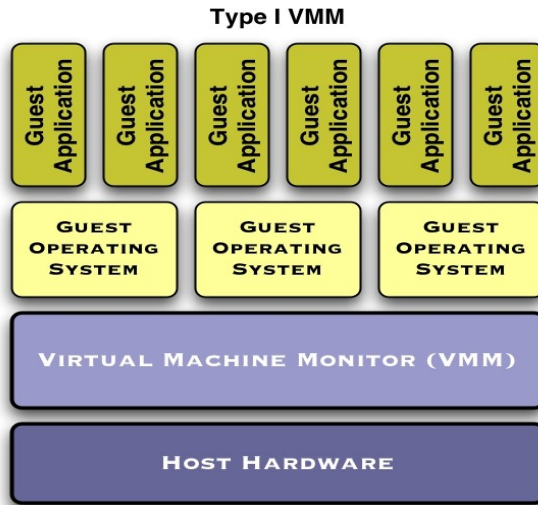
# Partitioning key points

- Cost of CPU, RAM footprint of multiple OS copies
  - > Per-instance binaries, daemons, and kernel processes add CPU, RAM load
  - > eg: Linux 100Hz jiffies, or buffer flush kernel processes
- Overall system debugging, tuning, and observability
  - > An OS in a partition is just like an OS in its own box; no added capability for debugging, performance tuning.
  - > Partitioning itself adds to the equation: TLB/cache pollution, or deferred response to device interrupts when sharing CPUs
- Manageability of multiple OSes no better than on separate boxes: reduces TCO benefits
  - > Multiple OS instances to maintain, patch, administer, monitor

# Virtual Machines



# Virtual Machines



- VMM Host (“hypervisor”) creates virtual machines for “guest” operating systems
  - > Guest OS runs in “user mode”. VMM traps and emulates privileged instructions
- Can run multiple, different, guest systems
  - > Different OSES, or different OS levels
- CPUs shared, guest RAM may be pageable
  - > Number of guests based on capacity rather than architectural limit or dedicated RAM
- Separate security, admin and failure contexts
- VMM can be a hypervisor OS (no userland) or an application running in a conventional OS
- Synthetic or abstract resources (possibly better primitives for I/O than “real machine”)
  - > Can replace “trap+emulate”, at lower cost



# Virtual Machine CPU and RAM

- VMM time-slices between guests just as regular OS time-slices between process
  - > Some VMMs permit dedicated CPUs as an option
- VMM allocates RAM to guest just as regular OS gives RAM to process
  - > Permits arbitrary (subject to load) number of guests
  - > Must control thrashing/overcommitment just as in regular OS
  - > Level 1 memory: Real storage (RAM) == “host real”
  - > Level 2: “host virtual” == “guest real” (guest thinks it's RAM)
  - > Level 3: “guest virtual” (address spaces provided by guest)
  - > Map level 1 to level 3 via software-created *shadow page tables*
    - > Hardware needs valid maps, is unaware of 3 levels
    - > Translations flushed/remade on both host and guest context switch
  - > Some VMMs permit dedicated, contiguous RAM as well

# Virtual Machine challenges

- Hypervisor is a single point of failure
- Variable degree of overhead
  - > Depends on hardware, hypervisor, guest OS, and workload
  - > Some architectures have complex semantics for privileged instructions; can be expensive to virtualize, depending on I/O, memory load, context switch
- CPU, RAM footprint of multiple OS copies
  - > Per-guest copies of binaries, daemons, and kernel processes (eg: housekeeping like buffer flush or Linux jiffies). Can cause thrashing if not very careful
- Overall system management and observability
  - > Guest OS in virtual machine can be challenge for performance questions
  - > Multiple OS instances (and hypervisor itself) to maintain, patch, administer
- Some of these issues have existed for 30+ years

# Tricky Virtual Machine issues

- Variable interrupt latency if other guests are running
- Timer management is complex
  - > Guest “CPU consumed” clock shouldn't advance when guest not dispatched, or guest charges to end-user time spent in other guest
  - > Guest notion of “how busy the machine is” is *always* wrong. Just what do `vmstat`, `top`, or `prstat` measure in a guest?
- Nested resource management can be a problem:
  - > VMM doesn't know when high priority guest is doing low priority work (starves other guests, or spends its high priority CPU access and then needs it later)
  - > Heisenberg effect when agents within otherwise-idle guests are asked what they are doing (dispatch idle OS, page its contents in from swap)
- Anti-social behavior: Guests should not spin on locks or when idle

# Tricky Virtual Machine issues

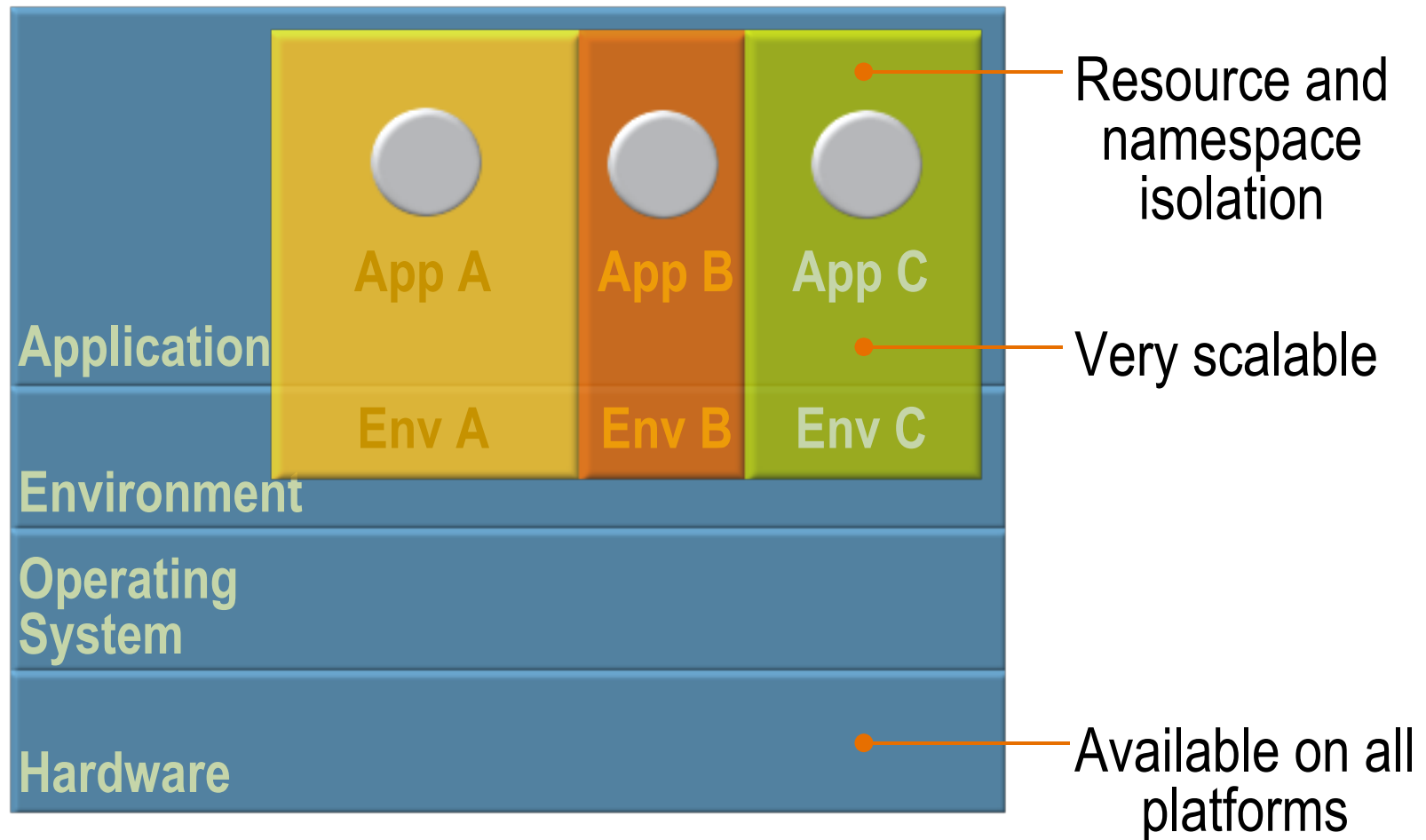
- Most OSes use all available RAM
  - > Makes sense on a real machine,
  - > In a virtual machine it increases pressure on a shared resource
- OSes typically don't have API to tell hypervisor “I don't need these pages any more – you can take them”
- Result: poor locality of reference: WSS approaches VM's “memory size” (its virtual RAM size)
- Per-OS file cache, per-OS duplication of every binary
  - > Some VMMs (eg: mainframe VM) have explicit shared memory segments for shared binaries or read-only filesystem
  - > Some VMMs (eg: VMware) detects identical pages by scanning and hash marking

# Tricky Virtual Machine issues

## Nested Resource Managers

- “An app designed to absorb all of a machine is a poor candidate for multiprogramming” (Lorin). An OS is such an app
  - > Lack of visibility from guest to host's load (“Should I reduce my consumption?”)
  - > Lack of visibility of host to guest's work and relative importance to other work
- Nested LRU with virtual storage guest can cause MRU (Wheeler)
  - > Guest selects a page for replacement, but the hypervisor is also under memory pressure and has already paged the guest-real page out
  - > Causes double paging: guest OS page out requires host page-in
  - > Can ameliorate via handshaking/balloon methods
- Typical answer: add RAM and CPU (which reduces cost savings)

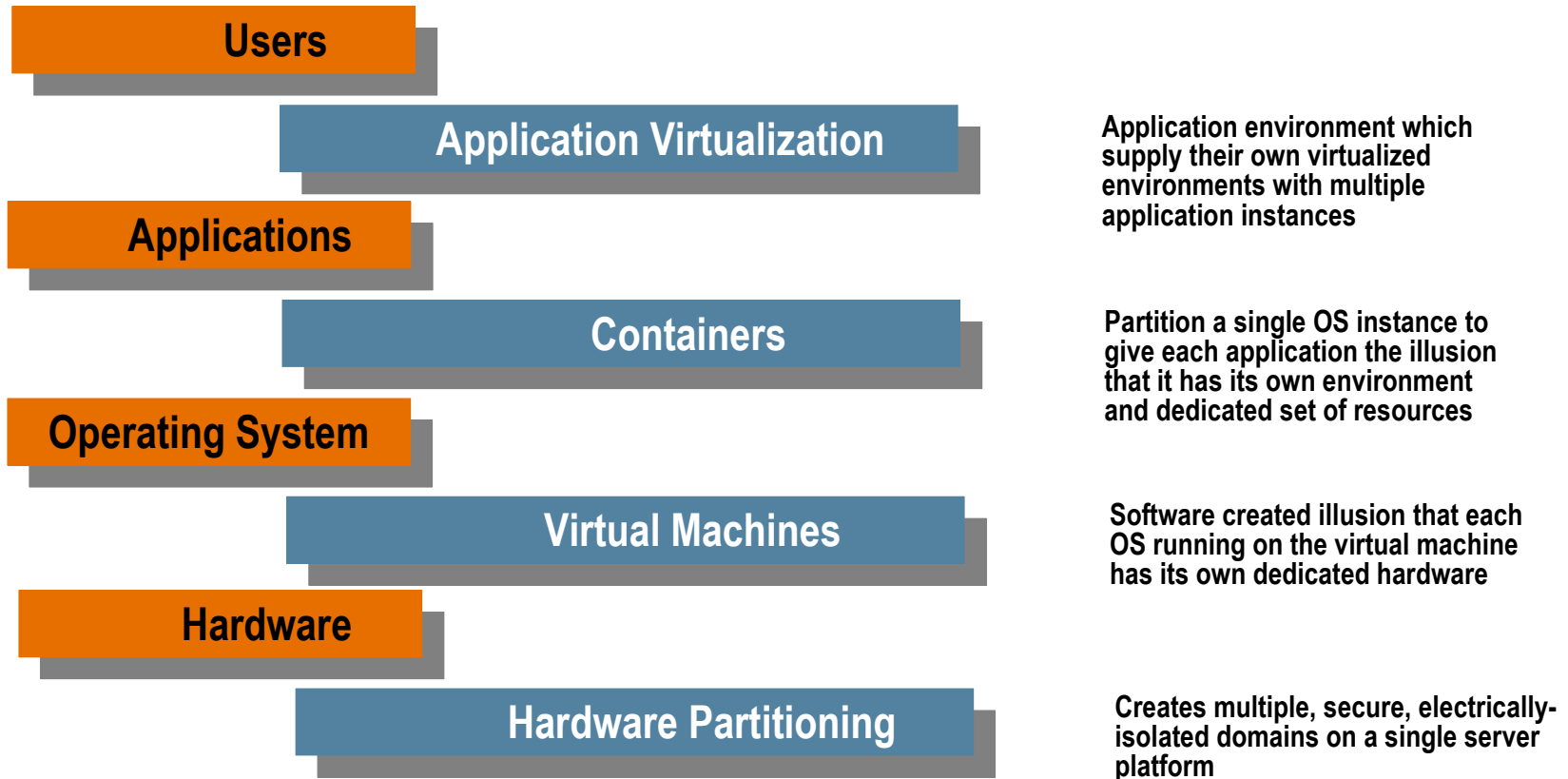
# OS Virtualization



# OS-level Virtualization

- A different approach: provide virtualized OS environments instead of virtualized machines
  - > Process and OS artifact abstractions rather than machine
- Solaris Containers (Zones) is the premiere example of this.
- Benefits: eliminate the technical and management complexities of the other virtualization methods.
  - > Reduced system overhead
  - > Reduced system/software management
- Downside:
  - > a single OS kernel; not for heterogeneous computing
  - > virtualization not quite transparent at application level

# Virtualization and Partitioning



**Virtualization technologies between layers of the hardware and software stack give the illusion of a dedicated environment to the layer above.**



# Virtualization Technologies

# Mainframe VM

## Longest duration example of virtual machines

- S/370 and successor architectures are virtualizable
  - > Supervisor state instructions issued in problem state generate an exception, (“privop exception”) which the hypervisor intercepts and emulates
- VM (CP/67 -> VM/370 -> VM/ESA -> z/VM)  
hypervisor/VMM
  - > Thin-style hypervisor, CP (Control Program), with no userland
  - > User abstraction is VM with virtual system config: boot their OS into it
  - > Adds synthetic “diagnose” instructions for efficient I/O and other services, mostly used by CMS (Conversational Monitor System) timesharing OS
  - > Other OSES run with little modification: z/OS (current incarnation of MVS), TPF, VSE use native privileged instructions. zSeries Linux uses some CP calls
  - > Scripted administration from authorized virtual machines running CMS
  - > Many services (eg: TCP/IP, SNA, NFS) provided by daemon virtual machines
    - > Helps maintenance, redundancy (eg: can have multiple IP stacks)

# Mainframe VM

## From VM/370 to current date in a single slide

- VM resource manager for CPU, RAM, I/O
  - > CPU shares allocated to virtual machine
  - > Can reserve RAM for favorite guests to protect working sets
  - > Can set maximum WSS to force guests to self-page if host storage is stressed
- Performance considerations
  - > Privops and guest app system calls cost much more virtually than real
  - > Some things hard to do (example, the “interval timer”). Some workloads designed for “real” machines difficult to do efficiently (eg: MVS, Linux)
  - > Useful metrics, concepts, like TVRATIO (“total/virtual”) for cost of virtualization
  - > Many exceptions now handled at lower cost by firmware
  - > Working set inflation, nested resource management remain problems
- IBM long ambivalent, reinvesting here for zSeries Linux
  - > Plan to add cooperative memory management, as in VMware

# IBM zSeries (mainframe) LPAR partitioning

- Elements of VM hypervisor moved into firmware
  - > Up to 60 partitions based on model. RAM is effective constraint
  - > Physical CPUs can be dedicated to a partition or shared
  - > CPU weights assigned to partitions; resource capping available
- Supports IBM OSES (z/OS, TPF, VM, VSE) and Linux
- Little virtualization provided
  - > VLAN, and “Hiper Sockets” (sic) for inter-partition networking
  - > OSES can use “diagnose” calls to indicate they are going idle
- Slight overhead (say, 3%. YMMV)
  - > Less if dedicated CPUs used. More if CPUs must cross “books” to access RAM

# IBM POWER5 partitioning

- POWER5 micro-partitioning with a hypervisor
  - > Up to 254 partitions based on model. RAM is effective constraint
  - > CPU granularity down to 1/10<sup>th</sup> of a CPU; can cap CPU resources
  - > Substantial overhead – as high as 40%
- *Paravirtualization* technique in which OS makes hypervisor calls (“hcalls”) instead of native supervisor instructions
  - > hcalls to yield the CPU if idle, for spin-loops, virtual I/O, ...
- Virtual I/O Server (VIOS) provides VLAN, VSCSI
  - > Executes in a partition, not part of the hypervisor
  - > Virtual SCSI, FC disks
  - > Virtual Ethernet for inter-partition VLAN or external networking

# HP Virtual Server Environment (VSE)

- For HP Integrity (Itanium based) servers, due 12/2005
  - > Not yet shipping, so details are skimpy
- Allows multiple instances of HP-UX (initially) with
  - > Linux and Windows 2003 to be added in 2006
  - > OpenVMS to be added later
- Four guests per physical CPU initially. Architectural limit of twenty per CPU
- Uniprocessor guests at first, with virtual SMP to follow
- Overhead said to be 2 to 15% dependent on workload
- Managed by a "Service PArtition" (SPAR) running a lightweight version of HP-UX

# Microsoft's x86 Virtualization

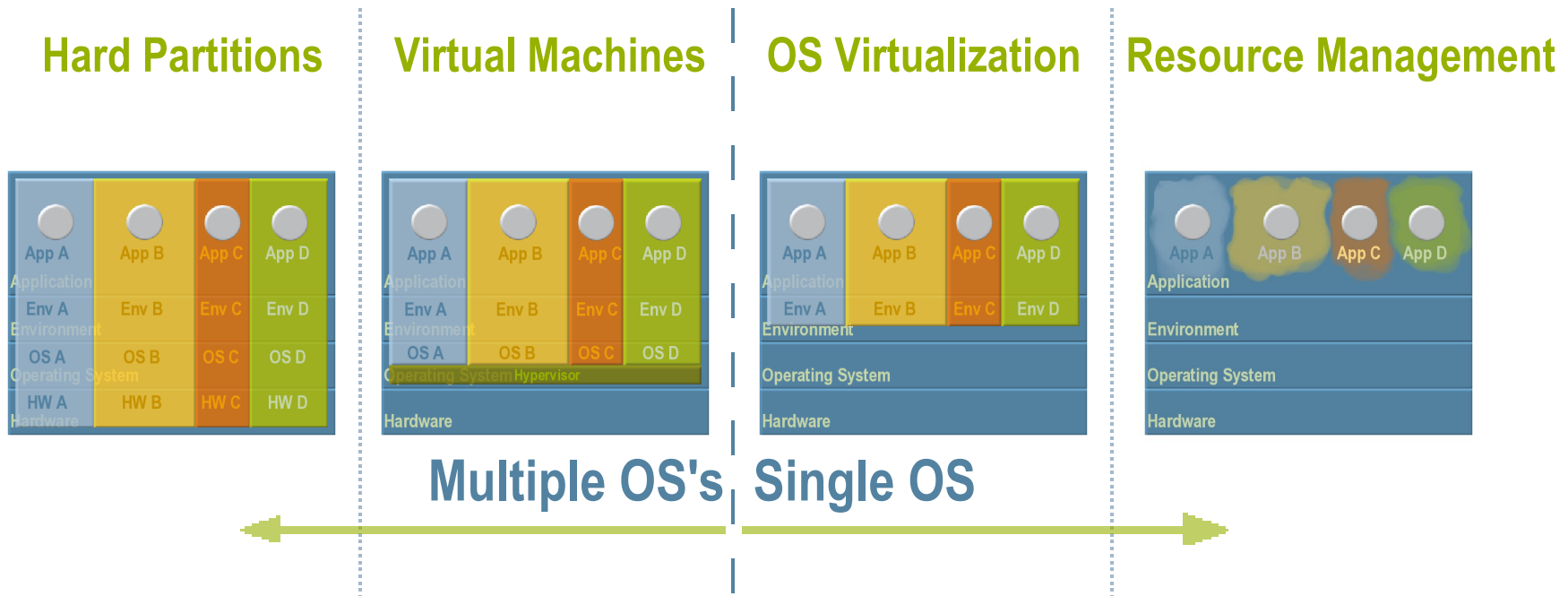
- *Virtual PC* (desktop) and *Virtual Server* products
- Host, guests are Windows 32-bit uniprocessors only
  - > Virtual Server 2005 R2 will support 64-bit
  - > Virtual Machine Additions (VMA) guest video driver and clock sync
  - > Linux has been tested and seen to work, but is not supported
- For consolidation and for running NT4 on current computers
- Restrictions on supported applications
  - > Cannot run Exchange, Sharepoint, Speech and Certificate Servers.  
See <http://support.microsoft.com/kb/897614/en-us>
- Windows System Resource Manager (WSRM) can manage CPU and RAM (Windows Server 2003 Enterprise Edition and Datacenter Editions)

# SUN Virtualization Technologies



# Possible Solution Approaches

- Technologies enabling consolidation
  - > Resource Management
  - > Virtualization and Partitioning Technologies
- Industry analysts distinguish four categories:



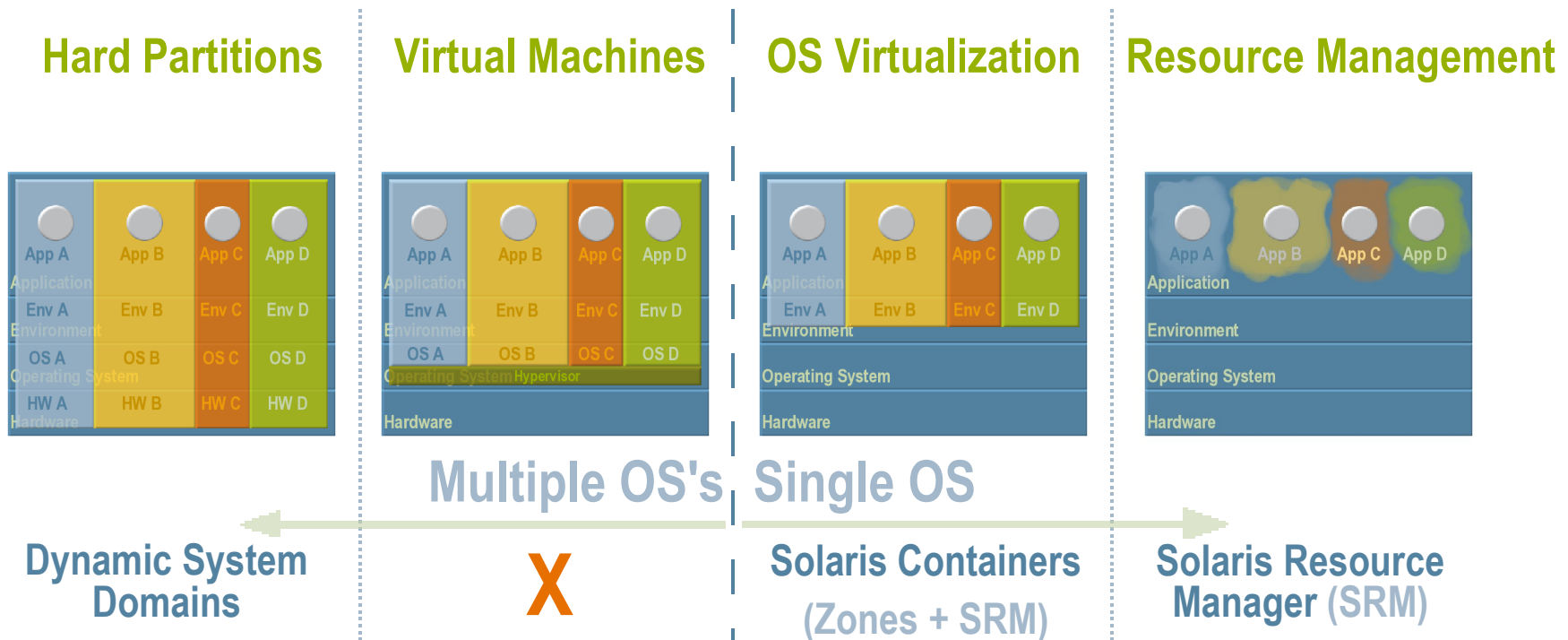
# Evolving Solutions

- Partitioning in the hardware is moving up the stack
- OS Virtualization builds on Resource Management features



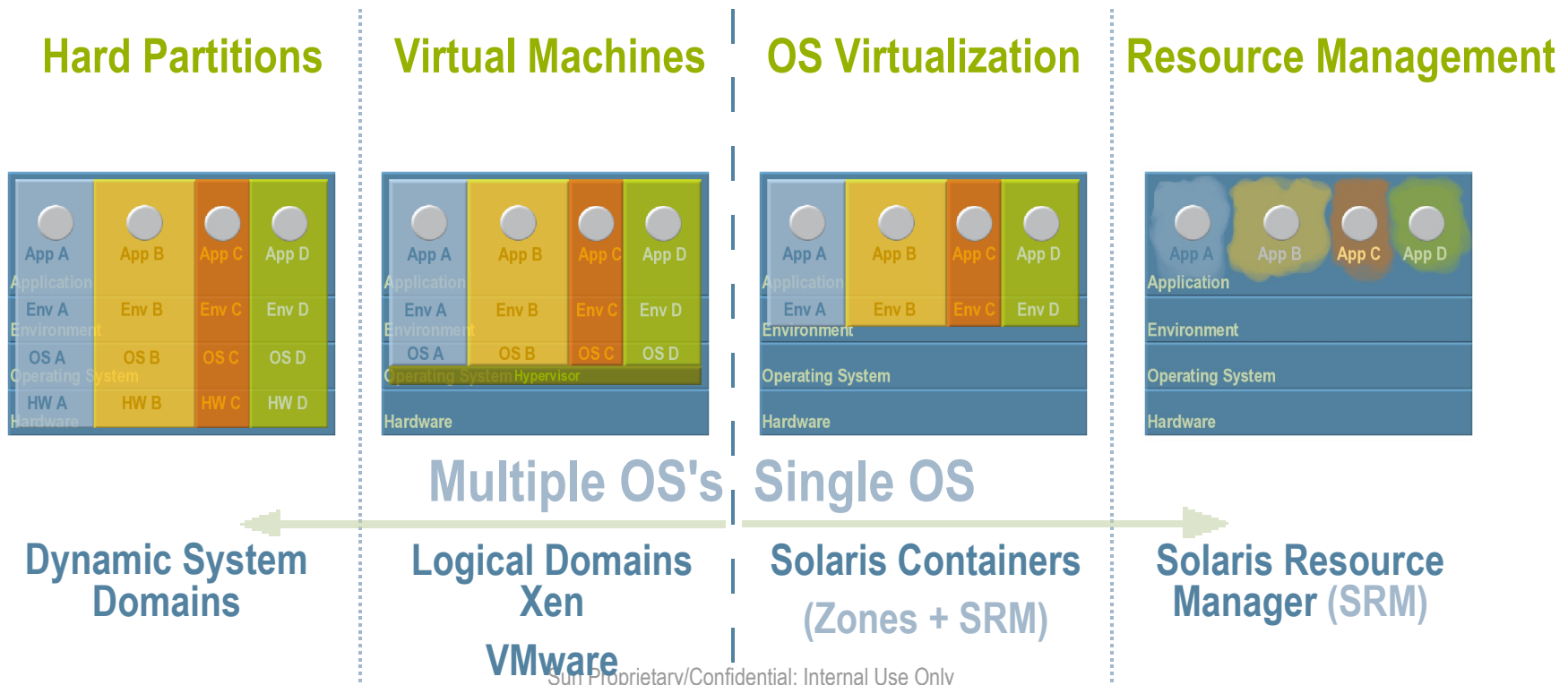
# Solutions from Sun

- Sun offers 3 options today
- Next (logical) step is to offer Virtual Machine technologies
- It's all about **Customer Choice**



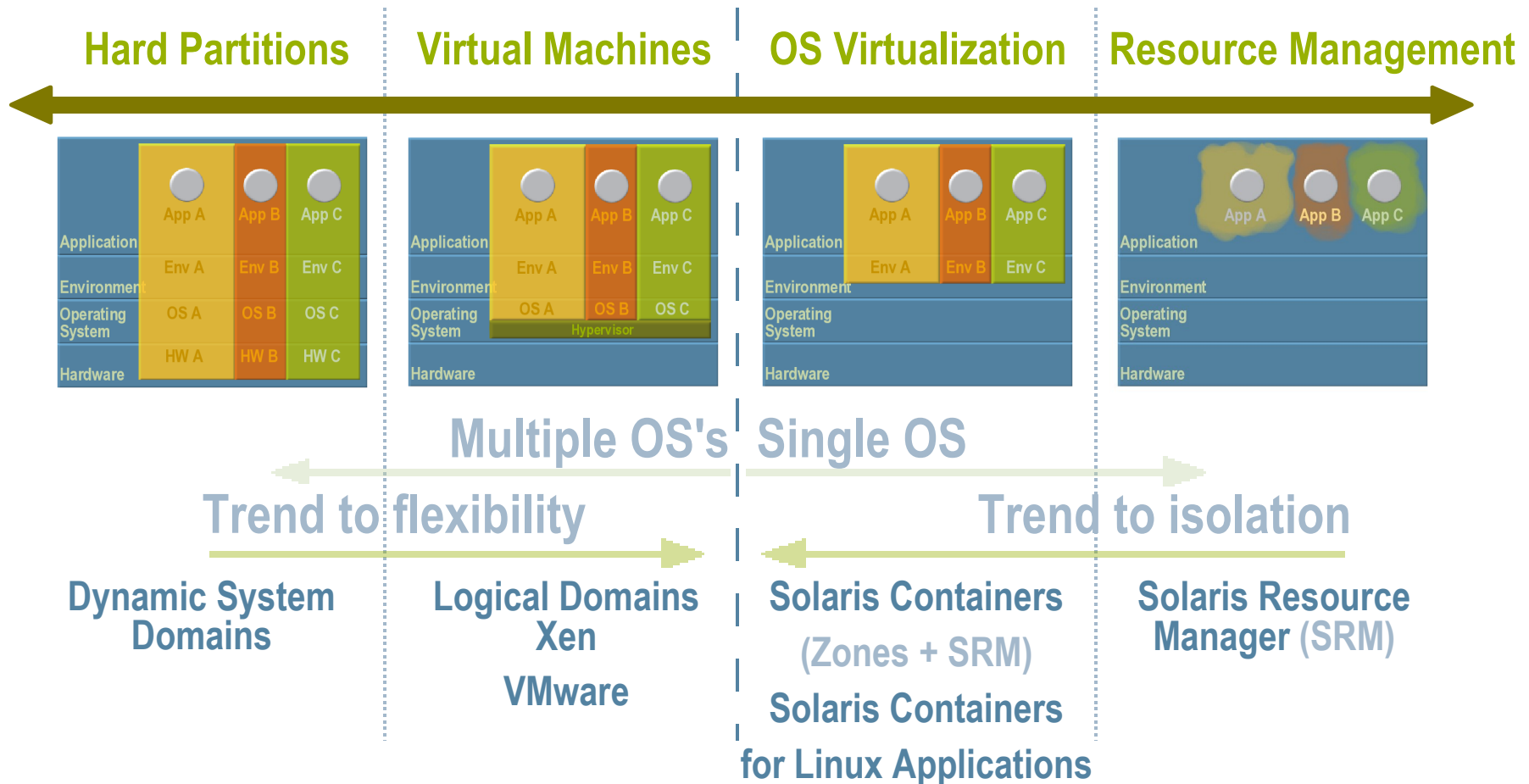
# Solutions from Sun

- Sun offers 3 options today
- Next (logical) step is to offer Virtual Machine technologies
- It's all about **Customer Choice**



# Solutions from Sun

- It's all about **Customer Choice**



# Approaches to Virtualisation

- True virtualisation
  - > Emulate machine architecture down to register level
  - > Can run any OS / software
  - > Performance penalty
- Para-virtualisation
  - > Define a SW interface to hypervisor
    - > Requires O.S. Modification
  - > More efficient implementation - no emulation required

*Sun's approach is para-virtualisation*

# Increasing Customer Choice

## Virtual Machine Technologies

- **Logical Domains**

- > Technology based on multicore SPARC chips
- > Developed for future Sun SPARC systems

- **Xen**

- > Open Source
- > Porting Solaris on x86

- **VMware**

- > Reselling / partner agreement
- > Solaris on x86 as client OS

# Enhancing Flexibility

- **Solaris Resource Manager**
  - > Physical Memory Control
  - > Swap sets
  - > Enhancements automatically part of Solaris Containers
- **Solaris Containers**
  - > Extending back-up, recreation, migration capabilities
  - > Solaris Containers for Linux
- **Network virtualization**
  - > Additional separation between Solaris Containers



# Solaris Containers, Solaris Containers for Linux, Zones, BrandZ

- **Solaris Containers** initially combined Zones & Resource Management
- **BrandZ** is an extension to Zones technology
  - > Enables Solaris Containers to assume different OS personalities a.k.a. “Brands”
- **Solaris Containers for Linux Applications** build on BrandZ, allowing Linux applications to run unmodified in a Container
- Ideal for Linux consolidation and development as well as migration to Solaris

# Technology Sweet Spots

- **Solaris Containers**

- > Lowers kernel instances, reduces OS management costs, fine-grained security
- > Single-kernel, heterogeneous application environments

- **Dynamic Systems Domains**

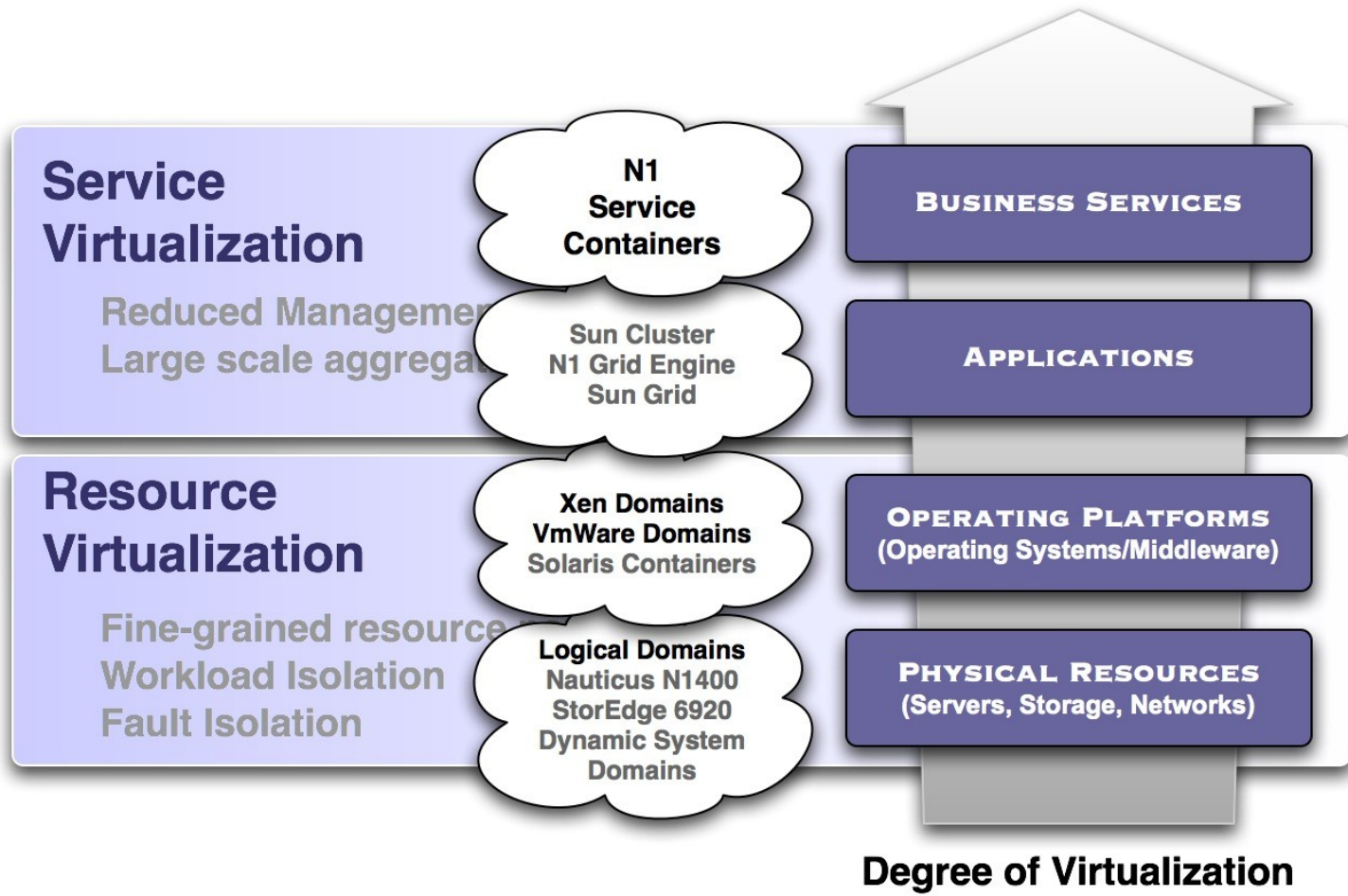
- > Maximizes hardware isolation

- **Virtual Machines**

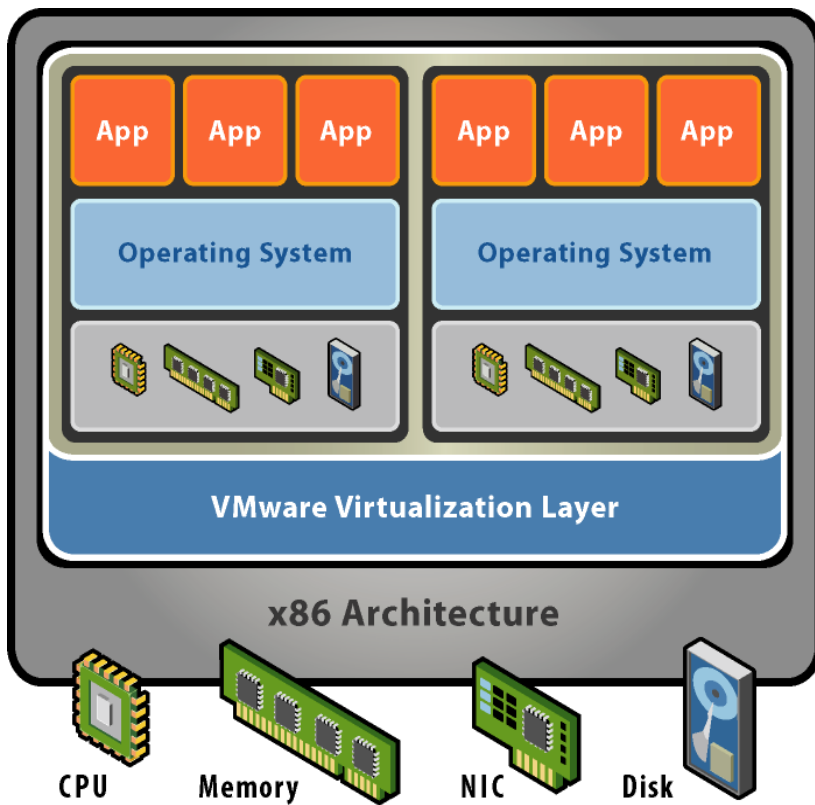
- > Multiple-kernel heterogeneous full OS environments

- Technologies are complementary

# Future Virtualization Offerings

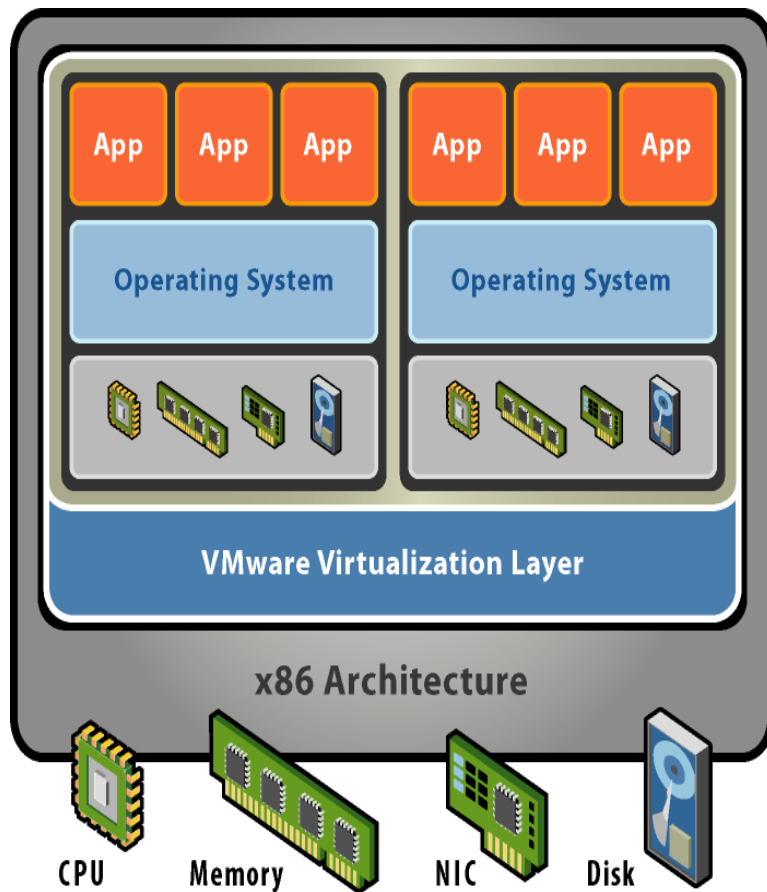


# VMware ESX Server Strategy



- Separates OS and HW to break HW dependencies
- Encapsulate OS and apps into a VM
- Fault and security isolation
- OS choice flexibility

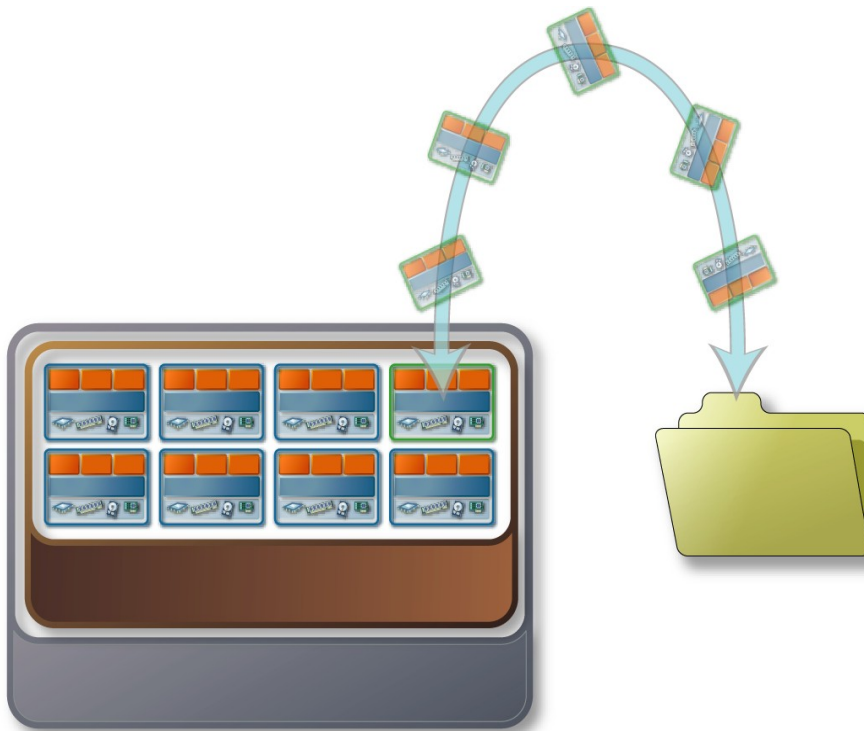
# VMware – x64/86 virtual machines



- Can run under host OS, Windows or Linux (VMware Workstation or GSX)
- Or run on bare metal (VMware ESX)
- Guest can be most unmodified x86 OSes (Linux, Windows, BSD)
- Solaris works as a guest. Sun partnering with VMware
- Virtual uniprocessor except with VMware ESX, which allows 2-CPU guest
- Guest OS RAM is pageable; total RAM can be over-committed

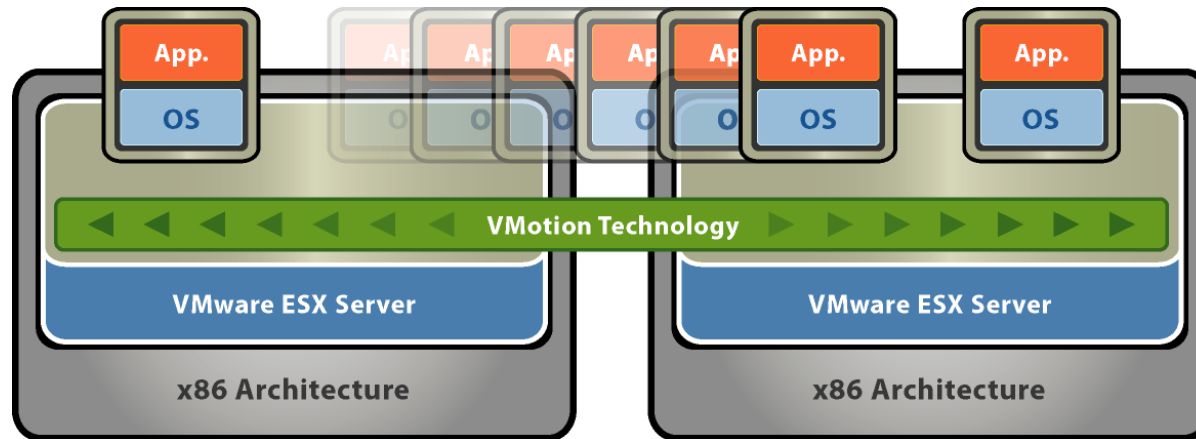
# OS and APP a Data File

- Server provisioning similar to copying a file
- Server migration becomes similar to data migration
- Data management techniques can be used for server management
  - > Cloning
  - > Versioning
  - > Server archival
  - > Remote mirroring



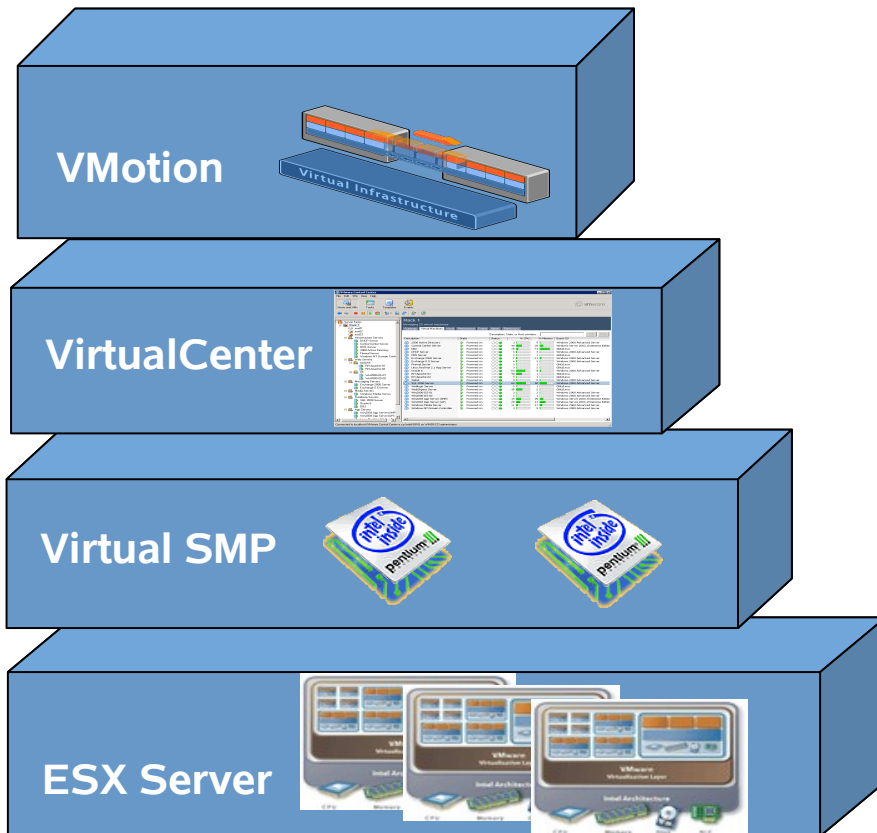
- OS, apps, data, devices, and state are now a file

# VMotion



- Migrate running VM's to different physical servers without service interruption
- Zero downtime maintenance to service underlying hardware and storage
- Balance workloads across the data center to fully utilize resources in response to changing business demands

# Core Building Blocks - Virtual Infrastructure Nodes



- **ESX Server**
  - > Hosts multiple virtual machines (VMs)
- **Virtual SMP**
  - > Enables dual virtual CPU VMs
- **VirtualCenter**
  - > Enables centralized management
- **VMotion**
  - > Enables migration of VMs between physical hosts
- **Virtual Infrastructure Node (VIN)**
  - > ESX Server + VC Agent + VSMP + Vmotion



# VMware ESX Server and Dual-core AMD Opteron

- NUMA Optimizations
  - > ESX Server balances virtual machines and their related data between the available NUMA nodes
  - > ESX Server attempts to maximize use of "local memory," that lies on the same NUMA node that the virtual machine that is running on.



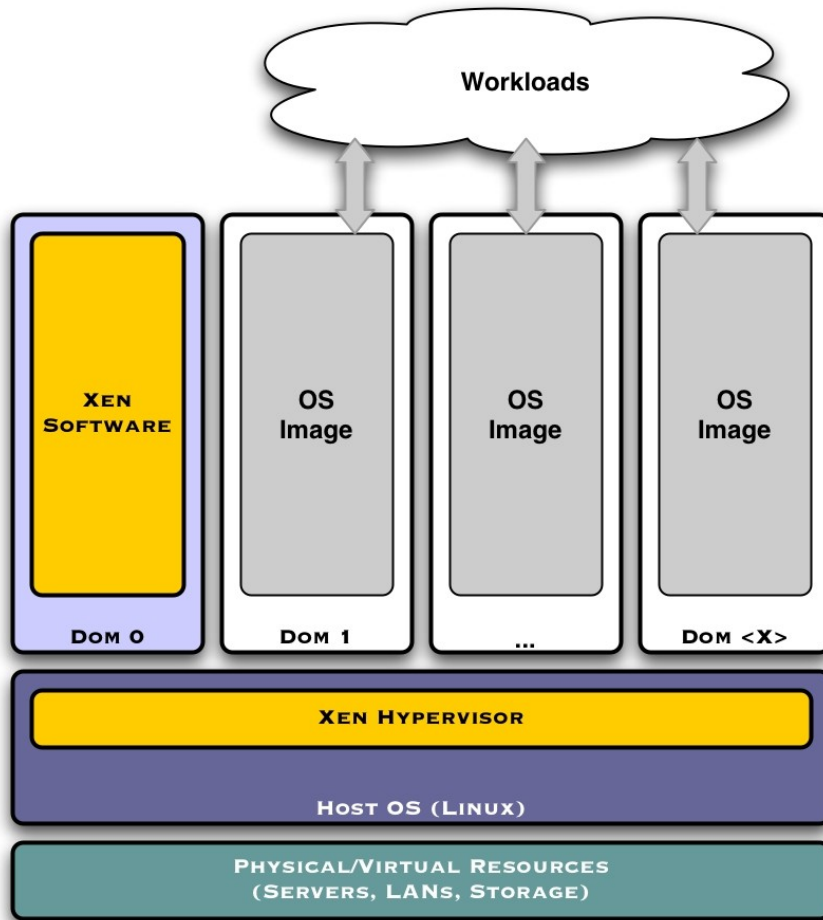
# VMware challenges on x64/x86

- Current x86 architecture hard to virtualize
- Some privops are ignored in guest mode, rather than causing a trap.
  - > Example: POPF instruction doesn't set/clear interrupt flags
  - > Silent “no operation” not acceptable to proper guest operation
- VMware dynamically scans and rewrites portions of the guest's code, with caching of patched locations
- Consequence is CPU overhead (estimated: 15-25% - YMMV)
  - > Far less for compute-intensive workload. VMware says 3-6%
- Inspiring AMD and Intel to add architectural enhancements (AMD's Pacifica, Intel's VT/Vanderpool)

# VMware – clever RAM features

- “Balloon” technique to reduce host-level storage contention
  - > Guest daemon locks/pins guest-real pages when host load high
    - > Pinned pages aren't used, so guest working set decreases
  - > Resort to paging as needed. Better to have guest page than VMware
    - > If guest is in page-wait, entire guest OS is blocked during page fault
    - > Guest knows better which of its pages/processes are important
    - > Downside: guest paging generates more guest I/O to intercept and emulate
- ESX transparent page sharing
  - > Detect identical pages via scan/hash fingerprint of page contents
    - > Guest TLBs changed to point to shared pages
    - > Use “copy on write” (COW) to detect alteration
  - > Best case scenarios >60% shared pages. Real world 7% to 30%

# Xen – Open Source x64/86 hypervisor



- Xen hypervisor is part of a host OS kernel
- Xen software resides in a special domain (Dom 0)
- Additional “domU” domains are instantiated over the Xen hypervisor, and run guest OSes
  - > Guests in x86 ring 1 (normally OS runs in ring 0)
- Guest OS's have to be modified in order to be hosted
  - > Uses “hypercalls” to reduce overhead and avoid scanning needed in VMware

# Xen

- *Paravirtualized* virtual machine monitor (VMM), or 'hypervisor' technology developed at the University of Cambridge U.K.
  - > Guest OS's are ported to the Xen platform.
  - > Machine dependent code of the OS – APIs and ABIs are unchanged.
- Secure partitioning between virtual machines, known as *domains* in Xen.
  - > Xen is responsible for managing all privileged state on the machine. Guests use hypercalls to invoke privileged operations via Xen.

# Xen Paravirtualization

- **Guest OS modified** to use hypervisor calls (“Hypercalls”) instead of native privileged instructions
- **Synthetic instructions** for I/O and memory management
  - > eg: guest sees TLB read/only, and requests changes via hypercall
  - > Synthetic I/O architecture and I/O virtualization
  - > Avoids scan and binary rewrites of guest OS code done by VMware
- **Reduces context switches** to hypervisor.
  - > Can batch requests for efficiency
- **Requires a port**
  - > Runs Linux, FreeBSD, NetBSD, Plan 9. Ports underway for Windows XP and Solaris (OpenSolaris has been booted under Xen - see [opensolaris.org](http://opensolaris.org))

# Xen's Design Principals and Goals

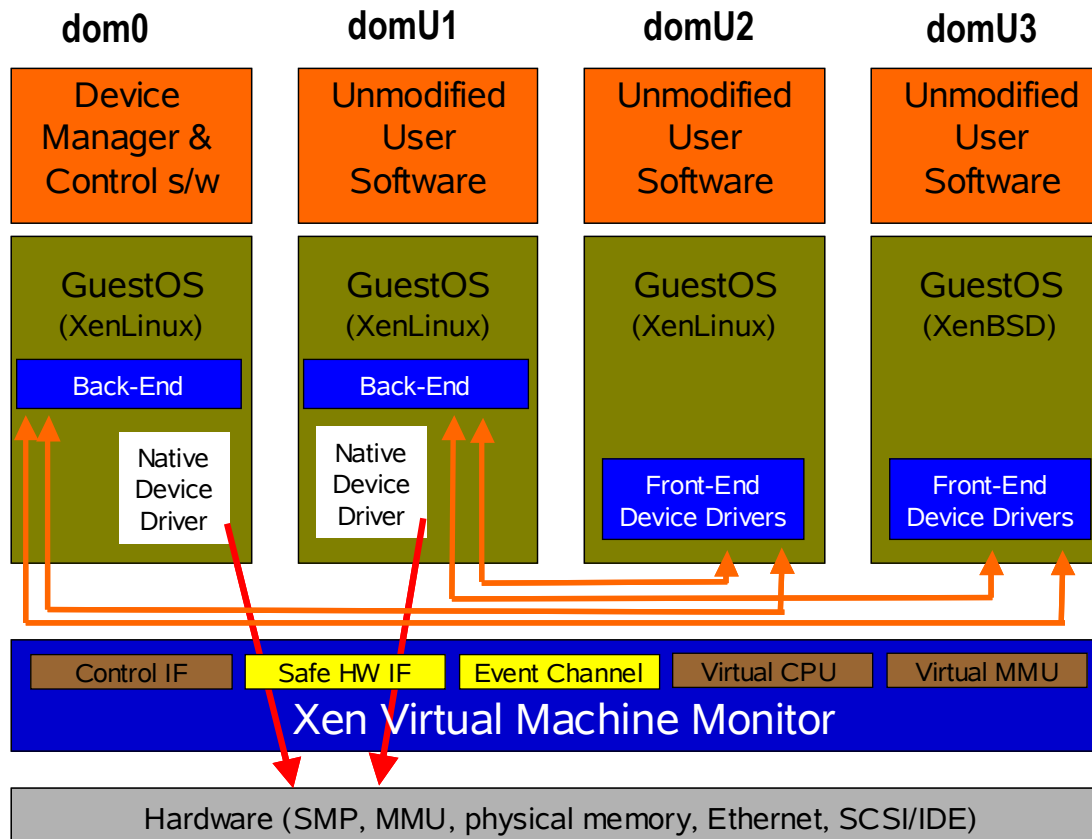
- Existing applications and binaries must run unmodified.
- Support full multi-process/application operating systems to permit complex server configurations to be virtualized within a single guest OS instance.
- Paravirtualization is required for high performance and strong isolation between domains on uncooperative architectures (x86)
- Support up to 100 domains on modern servers.
- Live migration of VM instances between Xen nodes

# Xen project

- Higher performance than UML or VMware
  - > Small % degradation compared to Linux running native for a number of applications (SPECint, Linux build, OLTP, dbench, SPECweb99)
- Generalizable to different operating systems
  - > Future hardware, VT-x / Pacifica, to avoid need for future guest OS ports
- Paravirtualization can yield nice effects for host and guest designed to work together
  - > Generalizes VM/370 idea, influencing multiple VMM designs now
- Can relocate or migrate running virtual machines for load balancing or server maintenance (like VMotion)
  - > Incremental copy of changing pages while guest runs



# Xen 2.x Architecture



# Structure of Xen 2.x

- Xen hypervisor
  - > Virtualize CPU and memory, schedule VM's, validate PTE updates, maintain system time, IRQ routing.
  - > Hypercall interface, event channels, and inter-domain device channels.
- Domain 0 – System Controller
  - > Xen control interface and domain management
  - > Runs native and backend network & block drivers
  - > Guest domain builder

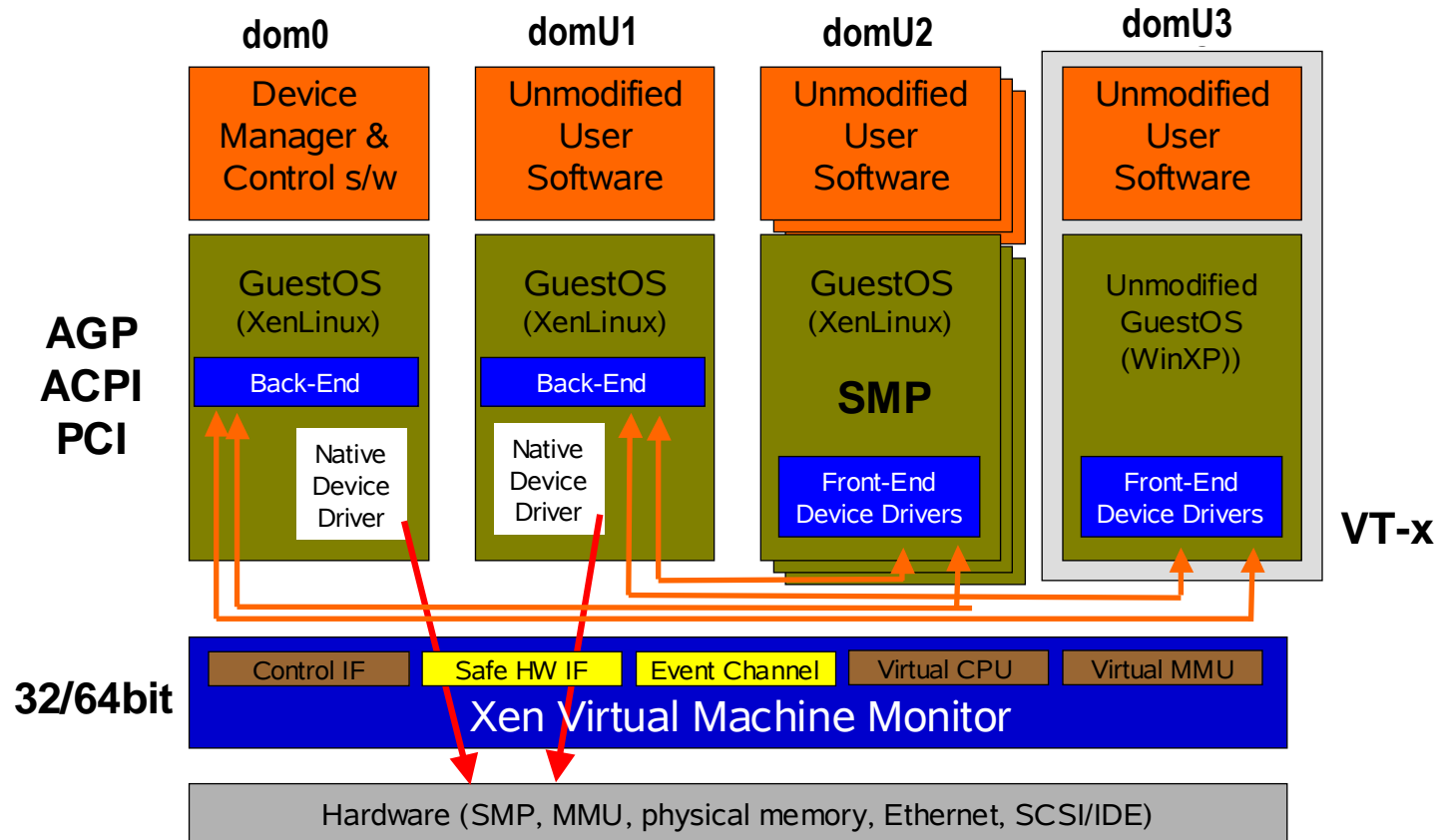
# Structure of Xen 2.x

- Guest Domain – Paravirtualized OS
  - > Privileged access replaced by hypercalls
  - > Virtual console, network and block devices
  - > Uniprocessor, 32-bit address space
  - > Loaded and managed by Domain 0
  - > Guest domains may be granted limited physical access to PCI devices by Dom0/Xen
  - > Event channels replace IRQs and provide the mechanism for inter-domain event notification

# Xen 2.x Key Capabilities

- Save/Restore and live migration of VM's
- Multiple OS's running simultaneously
  - > Linux 2.4 & 2.6, xBSD, (WinXP)
- Leverage device drivers available in non-Solaris kernels using split-mode virtual device drivers.
- Dedicated purpose domains
  - > Device driver, network firewall, etc.
- CLI and web-based domain management tools

# Xen 3.x Architecture



# Xen 3.x Key Features

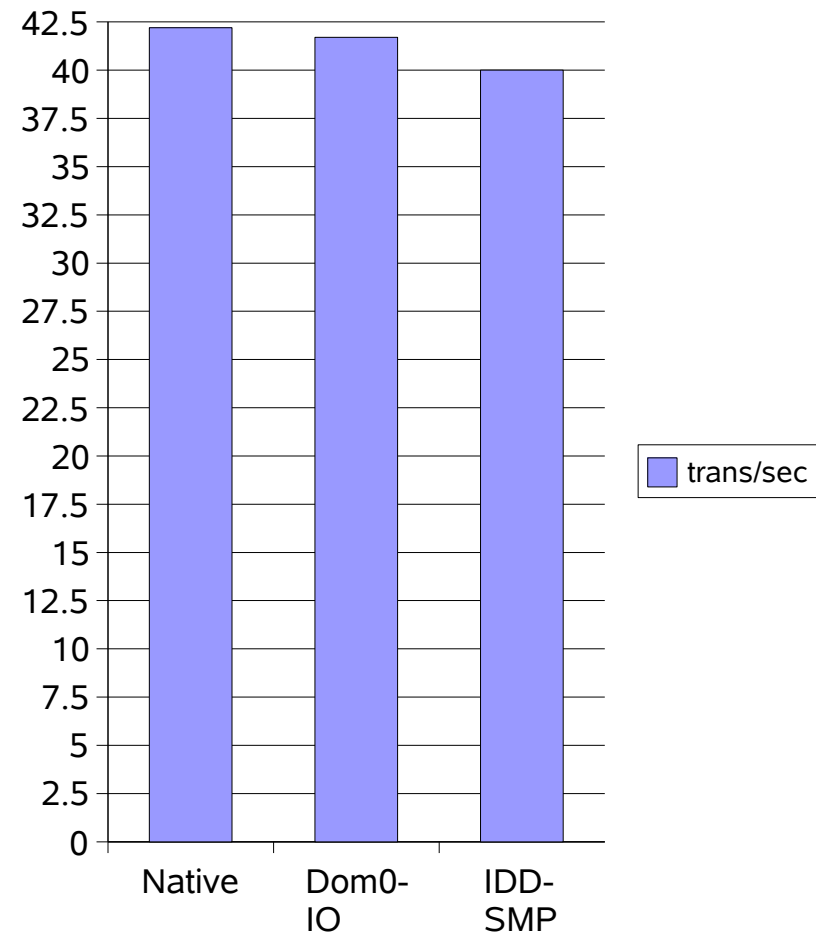
- Supports SMP guest domains
- Supports 32 & 64-bit address space domains
- Stable interfaces (control and hypercall) for guest OS's
- Capable of running native guest OS domains on Intel VT-x and AMD Pacifica based platforms
- Most platform initialization code (ACPI/APIC/PCI) moved from Xen to Domain 0

# Xen 3.x Key Features

- Virtual USB devices
- Improved CPU load balancing & QoS
  - > Hyper-threading aware
  - > Soft real-time scheduler
- Virtual Trusted Platform Module (TPM) support
- Improved Driver Domains
  - > Driver Restart/Replay
  - > SMP IDD

# Performance - Postmark

- Emulates workload of mail server
- Number of transactions/second.
- 1% - Dom0 I/O
- 5% - IDD



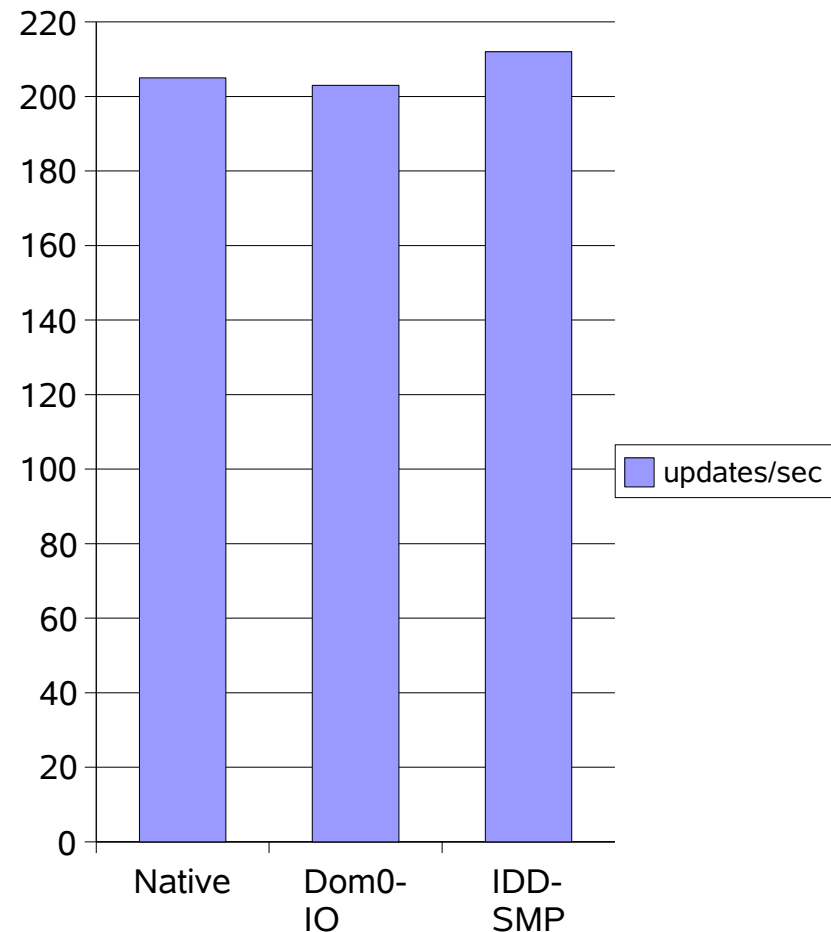


# Performance - OLTP

## OSDB/OLTP

- PostgreSQL 7.3.2
- Both Data Mining & OLTP access patterns
- No measurable impact on DM access (read-only)

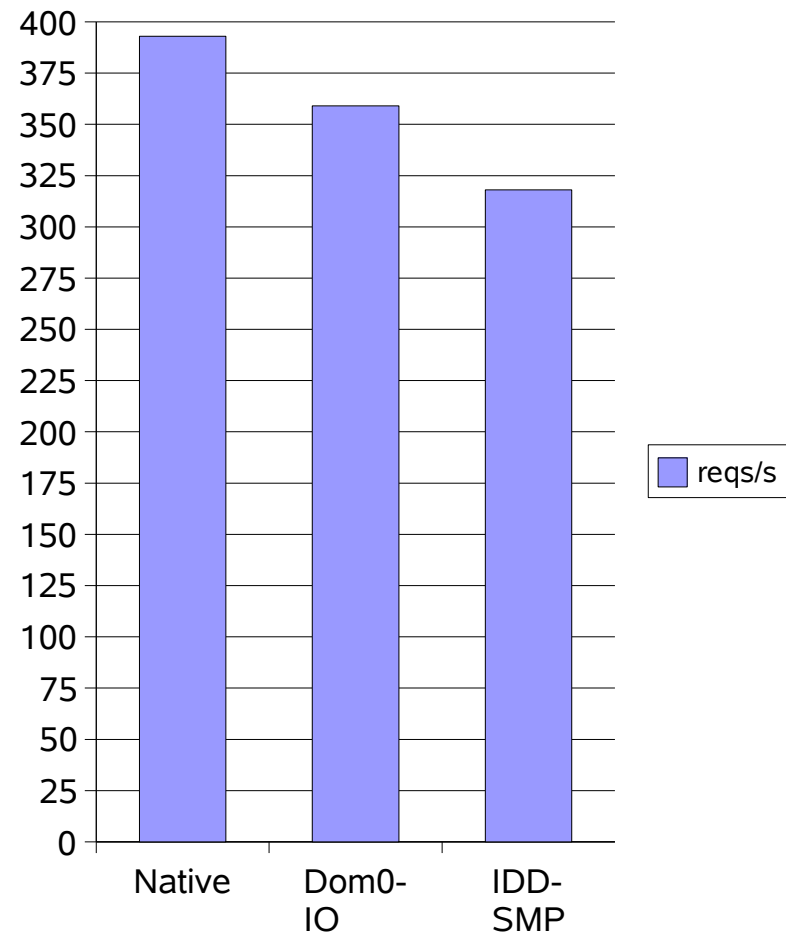
*IDD outperforms native (non-Xen) OS*



# Performance - httpperf

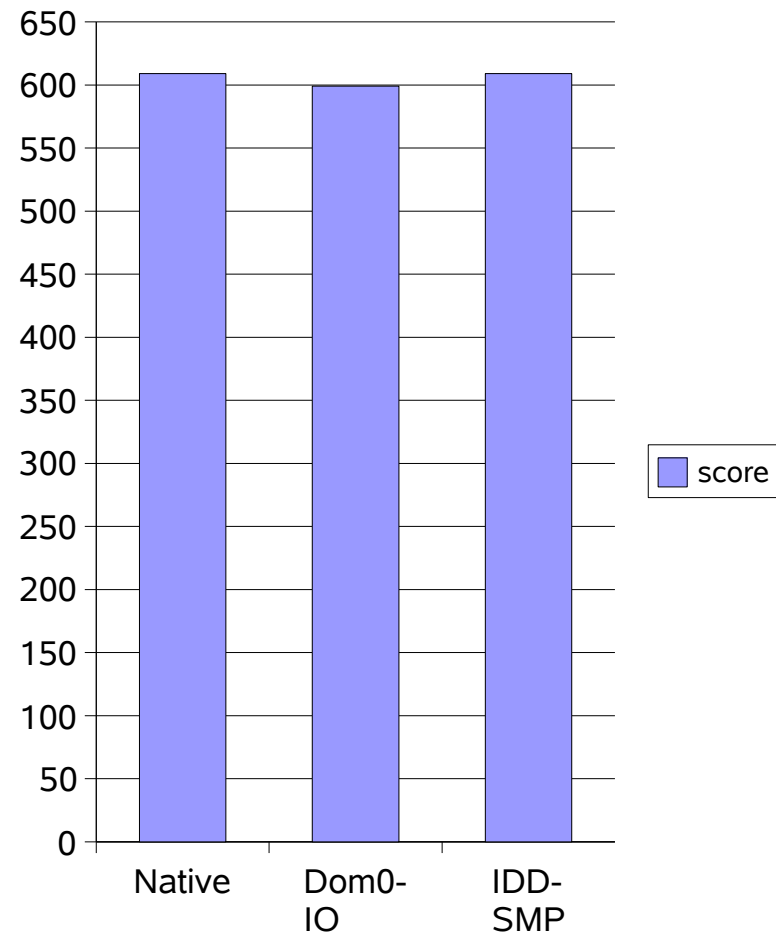
httpperf-0.8

- Apache server
- 200Mb/s server



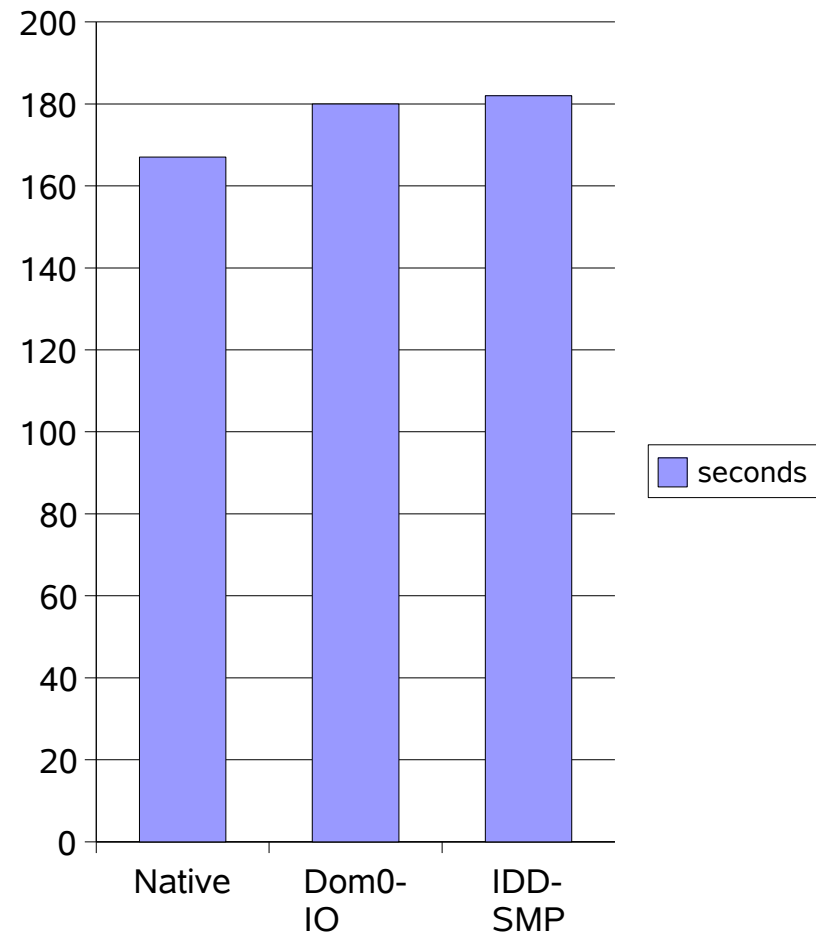
# Performance – Spec WEB 99

- Benchmark score reflects QoS supplied by webserver



# Impact On Performance

## Linux Build Time



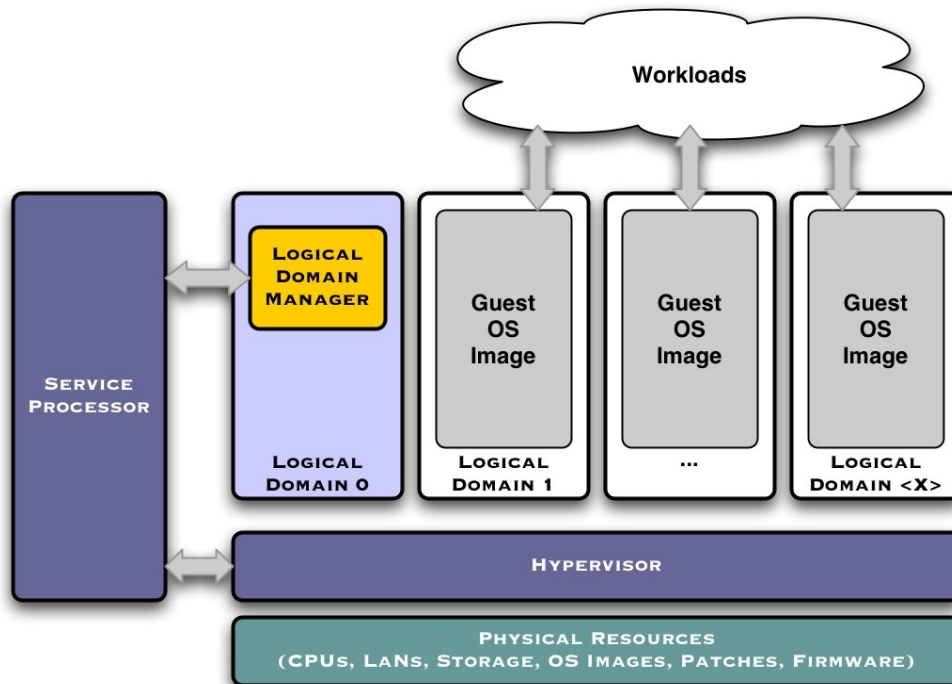
# Solaris on Xen Prototype

- Solaris 11 domU on Xen 2.0 x86
  - > Kernel in ring 1
  - > Privilege state via Hypercalls, virtual I/O
  - > Traps, interrupts, events
  - > Capable of booting over NFS or from virtual block device, virtual console config
  - > kmdb is working
- Port to Xen 3.0 underway

# Extending the Xen Platform

- Change Xen
  - > Scaling, Partitioning
  - > RAS & observability infrastructure
- Change Solaris-on-Xen
  - > RAS reporting & fault diagnosis across machine
  - > Networking & Security
- Leverage Trusted Solaris, DR, FMA, and SMF.

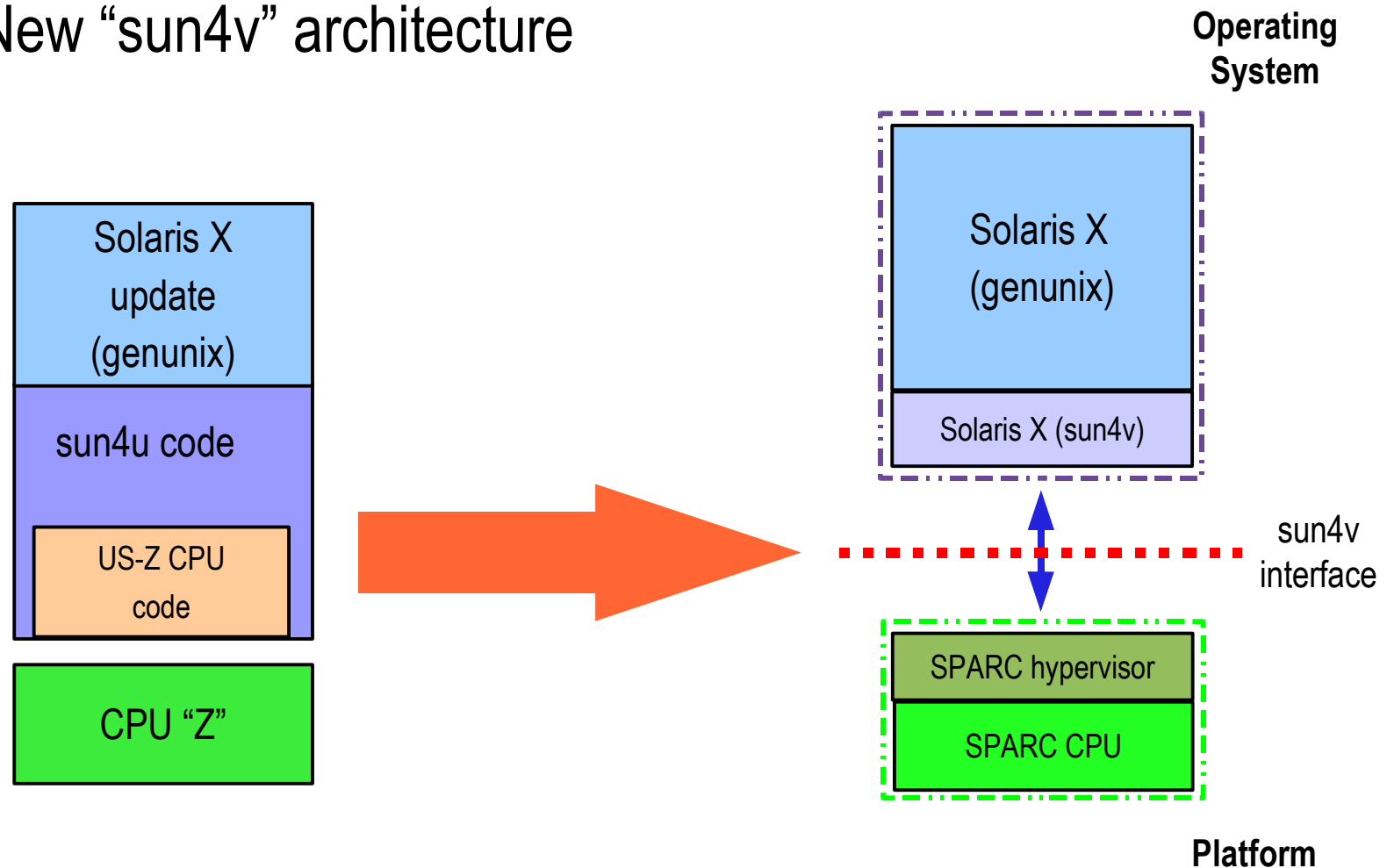
# Server Virtualization – Logical Domains



- A hardware virtualization solution
- Partitions a single physical system into one or more fully isolated “logical domains”
- Enables fine-grained “physical to virtual” resource mapping and physical resource sharing
- Physical resource can be dynamically reassigned without impact on running OS images

# The Hypervisor: Virtualisation for SPARC Platforms

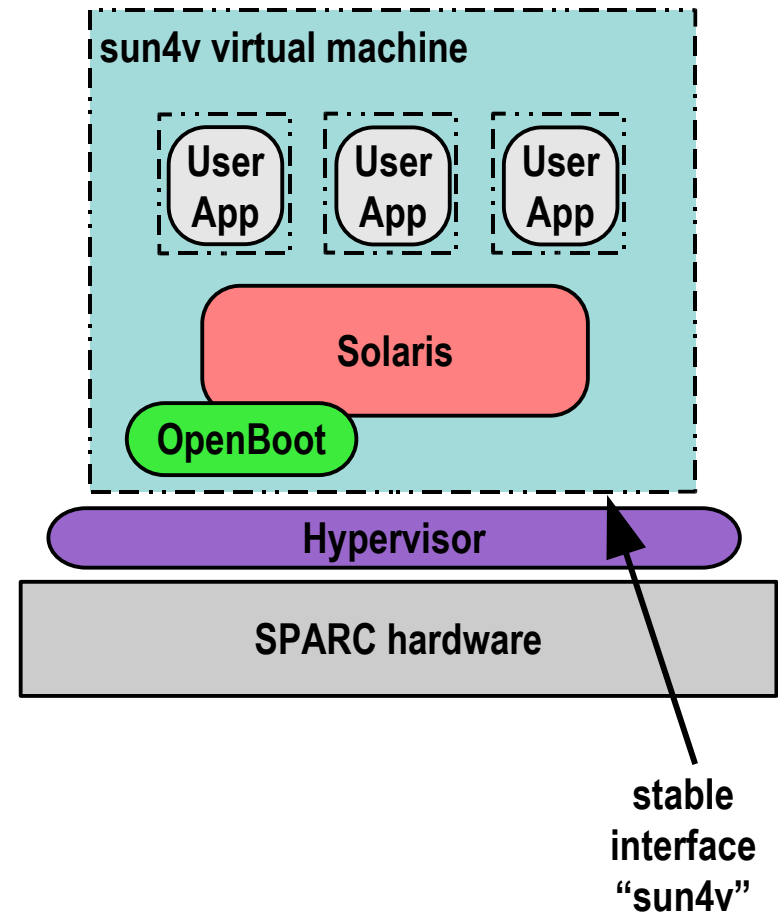
- New “sun4v” architecture





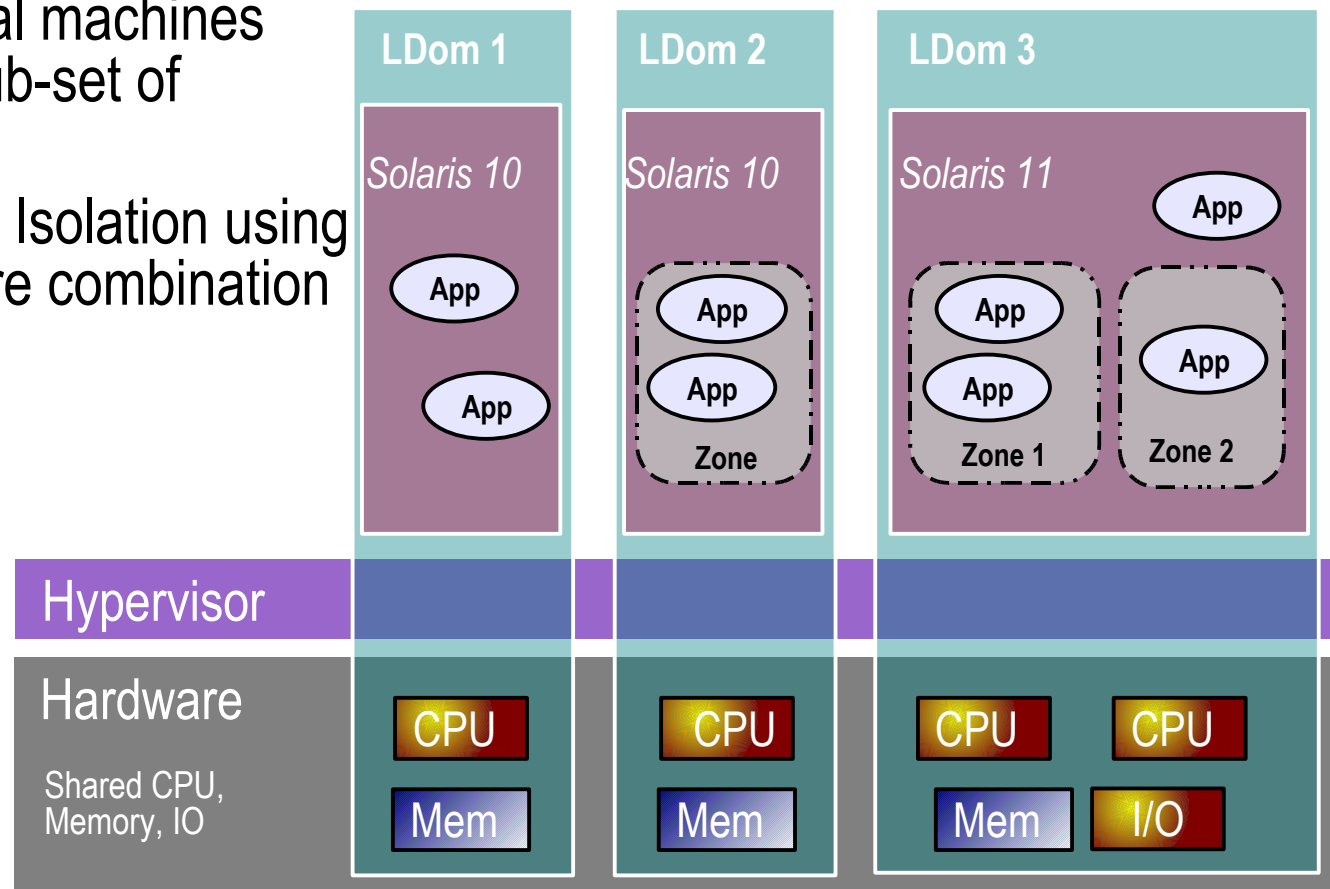
# Virtual Machine for SPARC

- Thin software layer between OS and platform hardware
- Hypervisor + sun4v interface
  - > Virtualises machine HW and isolates OS from register-level
  - > Delivered with platform not OS
  - > Not itself an OS



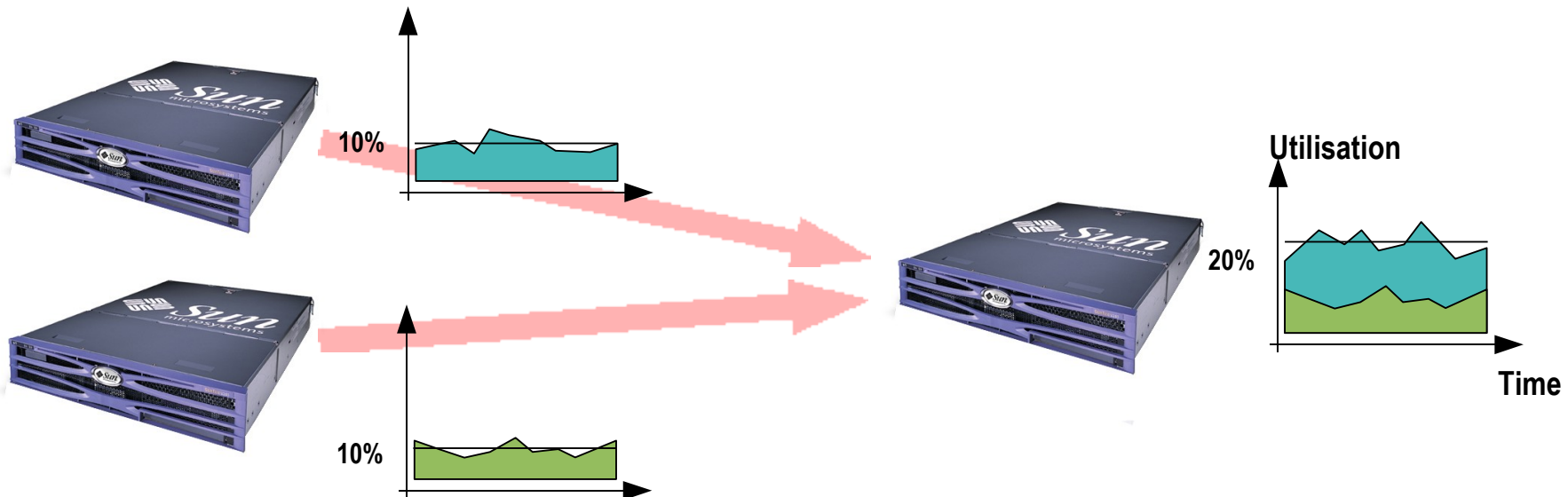
# Logical Domains

- Partitioning capability
  - > Create virtual machines each with sub-set of resources
  - > Protection & Isolation using HW+firmware combination



# Consolidation

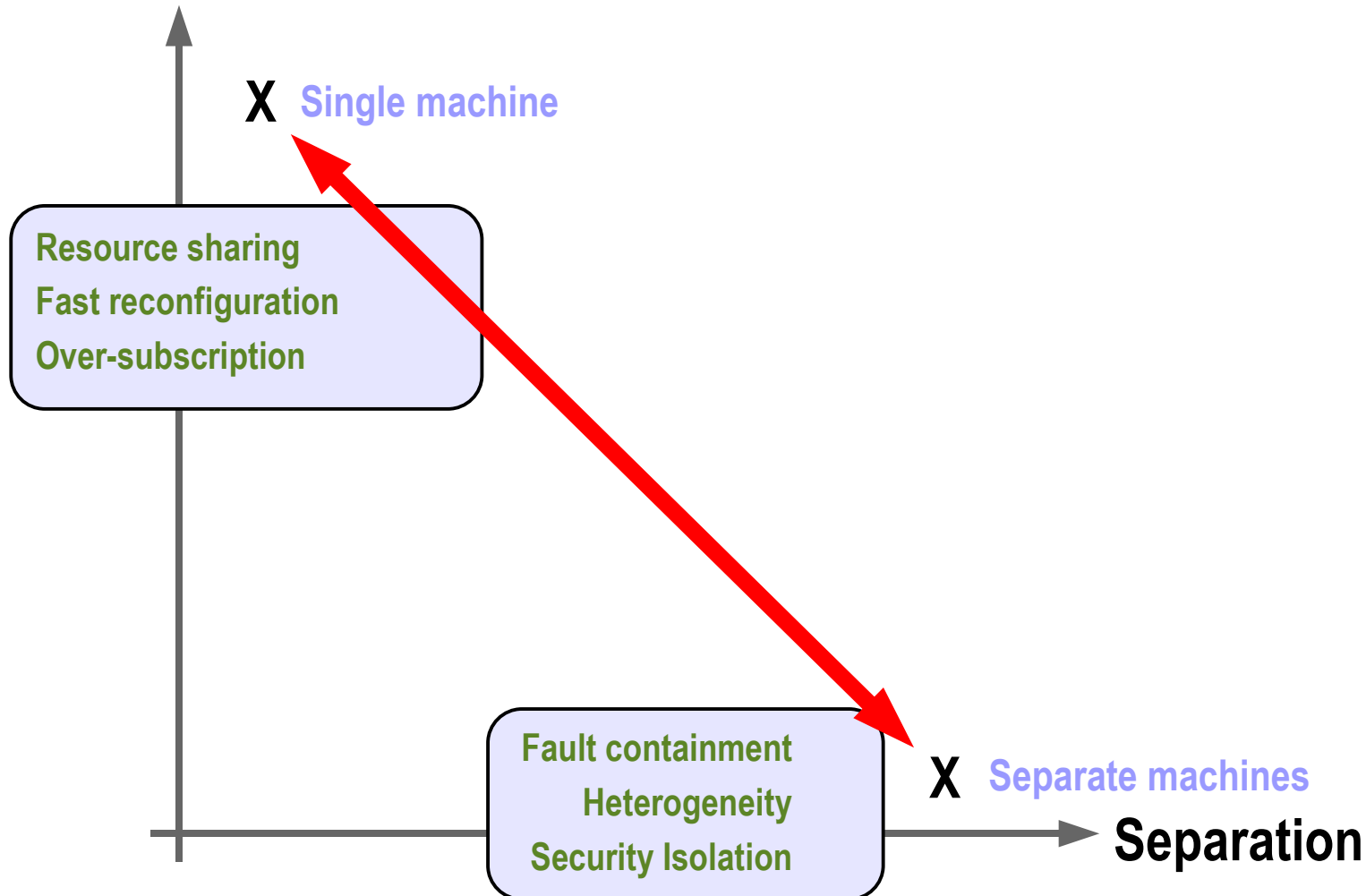
- Conventional Unix server utilisation is 7 to 15%
- Data center compression
  - > Goal: Increase utilisation of equipment



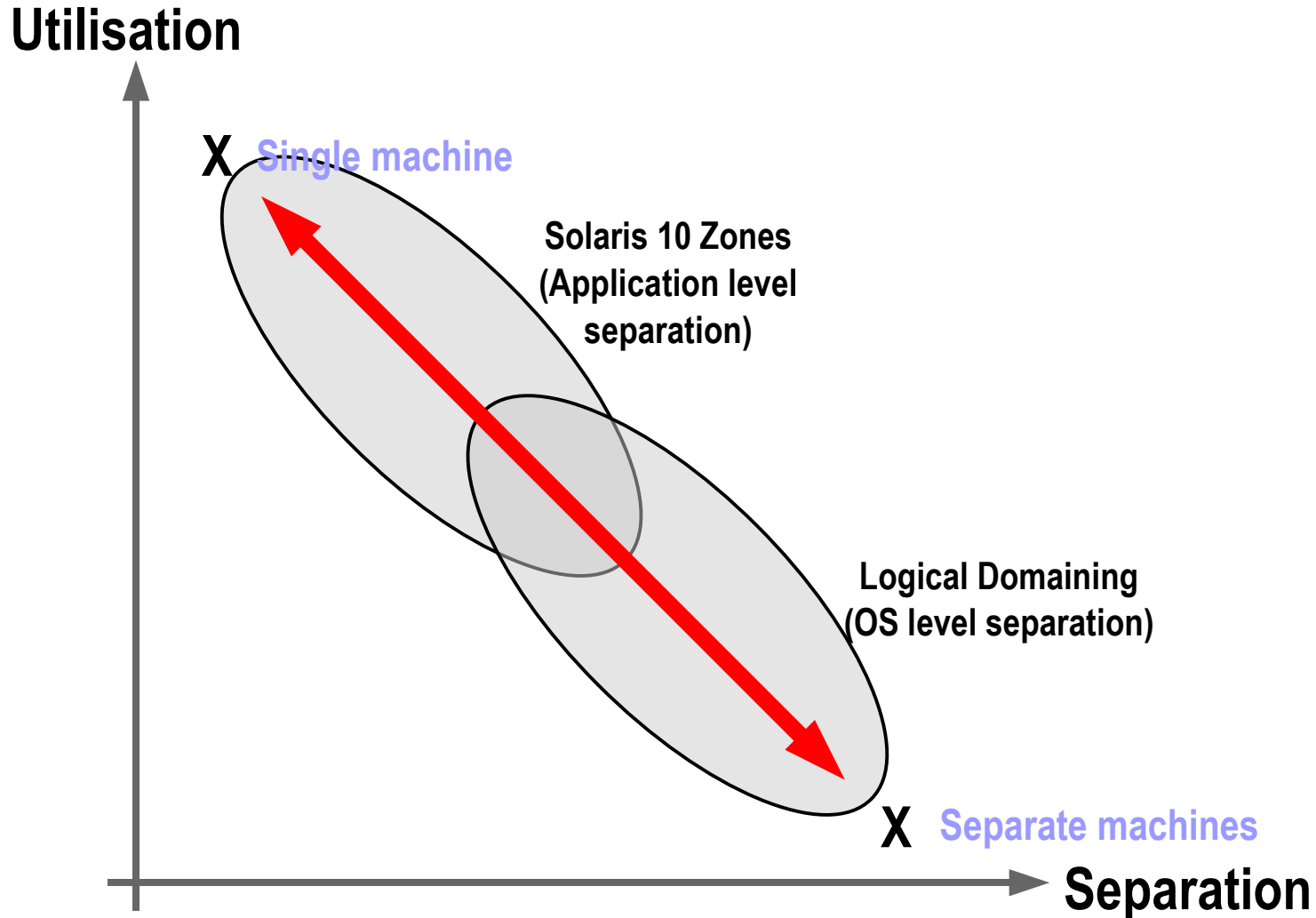
# The Voice of the Customer

- I want strong security!
- I need different OS environments!
  - > Different OS's or different versions for different apps
- I need fault & service containment!
  - > minimal downtime to other services for faults and maintainance

# Meeting Requirements: The Partitioning Continuum



# Complementary Solutions

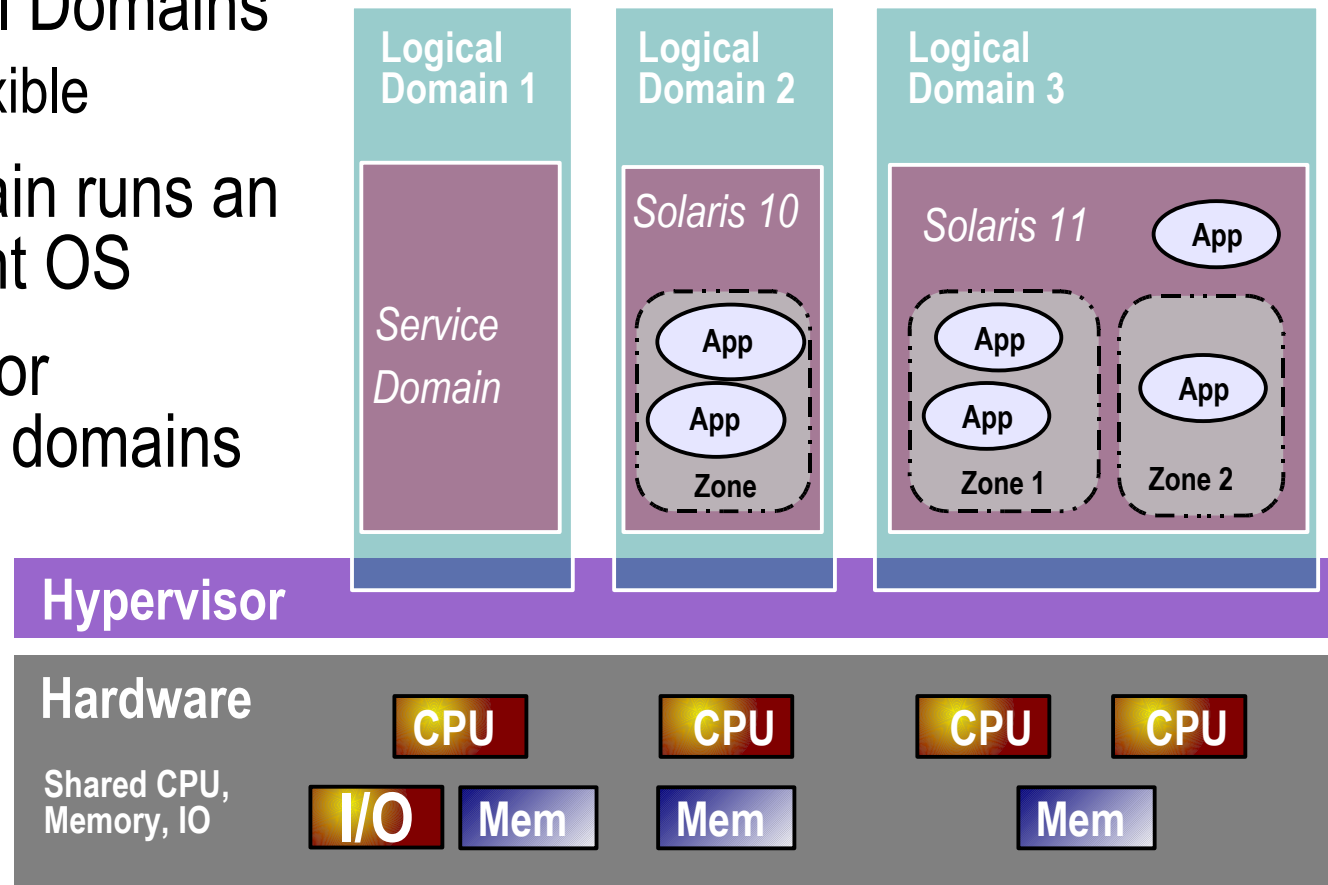


# Architectural Requirements

- Virtualisation
  - > Virtualised SPARC v9 compliant CPU
  - > MP support
  - > Virtualised memory & (shared) devices
  - > Physical device (driver) support
  - > Legacy OS support
  - > NUMA support
  - > Hot-spare & physical DR support
  - > Availability & survive-ability support
  - > Operations idempotent, no implied state
    - > Time-slice & Freeze/thaw capable
- Partitioning
  - > Multiple partitions
  - > Heterogeneous environment
  - > Zero trust
  - > Security
- Hypervisor
  - > KISS
  - > Fast
  - > Non-blocking
  - > Secure
  - > Non-trusting
  - > Robust

# The Sun4v/HV/LDoms Model

- Replace HW domains with Logical Domains
  - > Highly flexible
- Each Domain runs an independent OS
- Capability for specialised domains

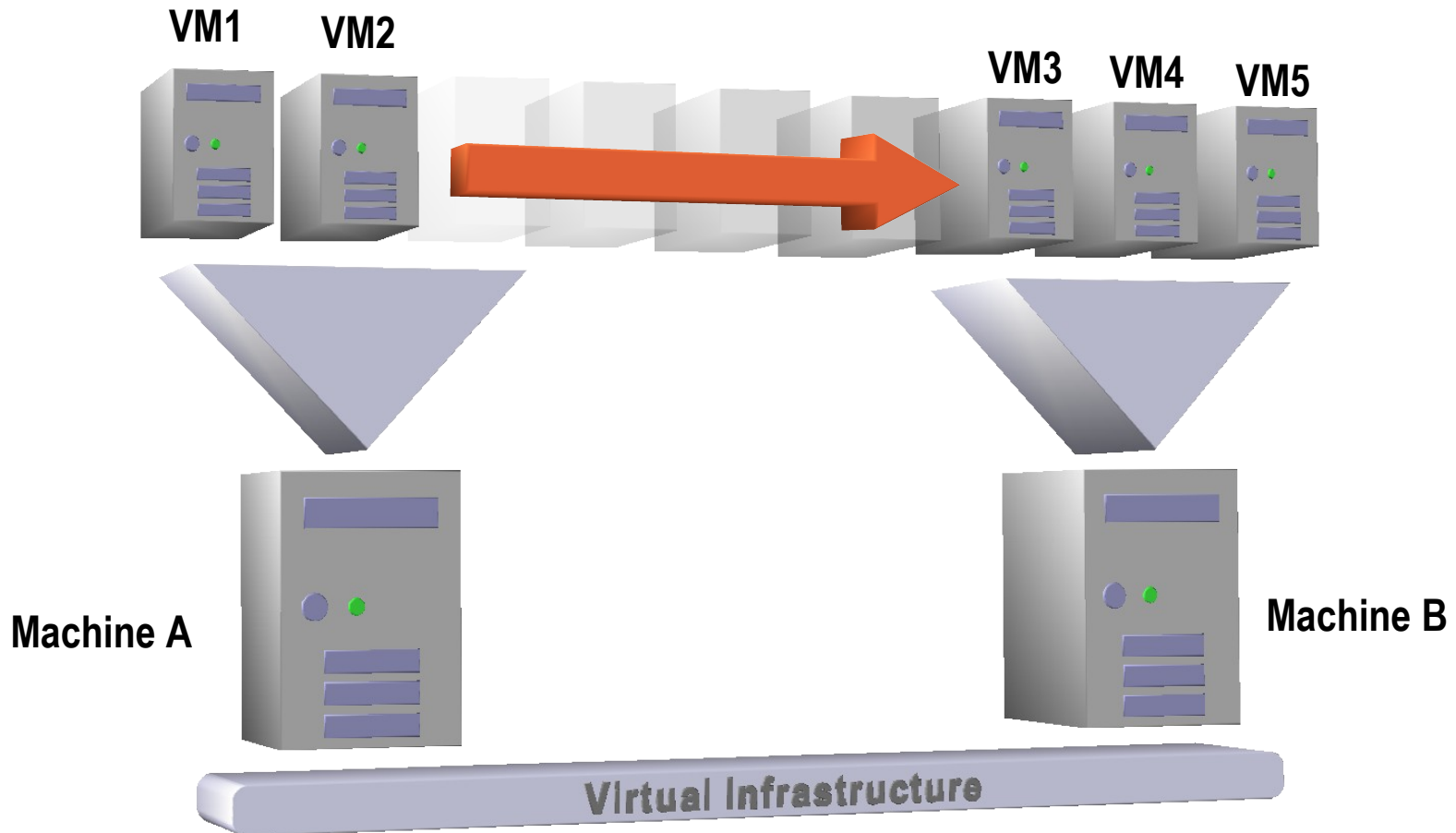




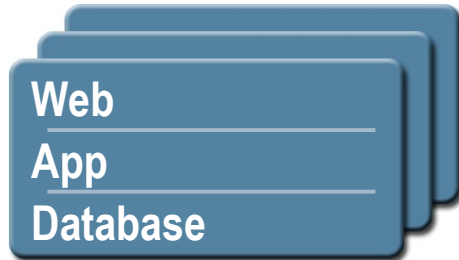
# Sun4v Error Philosophy

- Sun4v Guest only knows about the resources its been given
  - > Is never notified about errors that do not affect it
- Sun4v compatible OS must be able to handle error notifications even if running on a platform built after OS was released
  - > Highly abstracted form delivered to guest from hypervisor so guest can either panic or mitigate as it sees fit
- Hypervisor performs triage on errors
  - > Correctable errors never sent to guest
  - > All error notifications are sent in extreme detail to the “Service Entity”
    - > Essentially gets a dump of all relevant diag registers
- Error notifications are uniquely numbered and time-stamped for cross-correlation and chronic analysis

# Live Migration



# Evolution of Server Virtualization



Run multiple user's applications each in its own secure, isolated environment

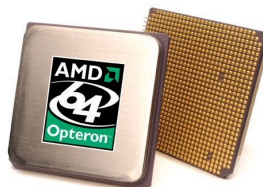
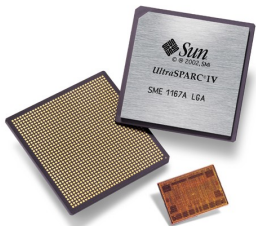


Create multiple virtual OS environments on a single OS instance

LDoms



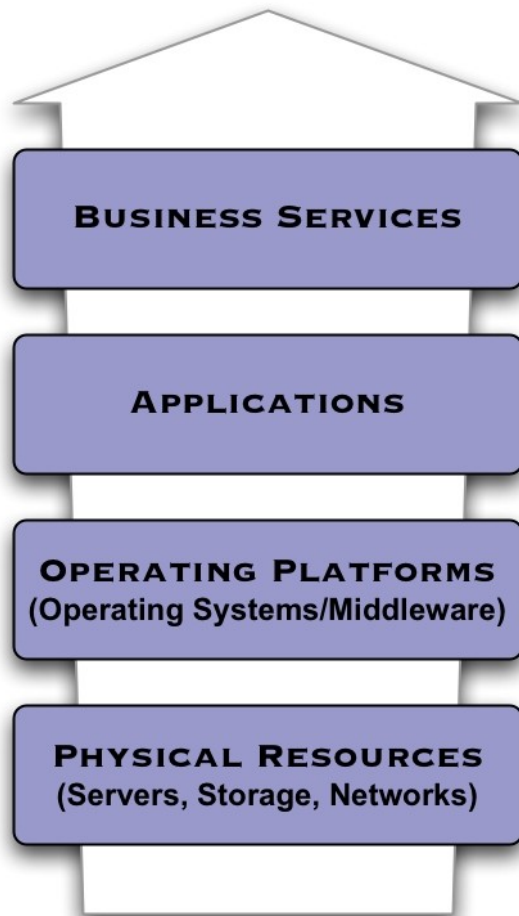
Create multiple virtual CPUs on a single physical CPU



Create multiple cores on a die

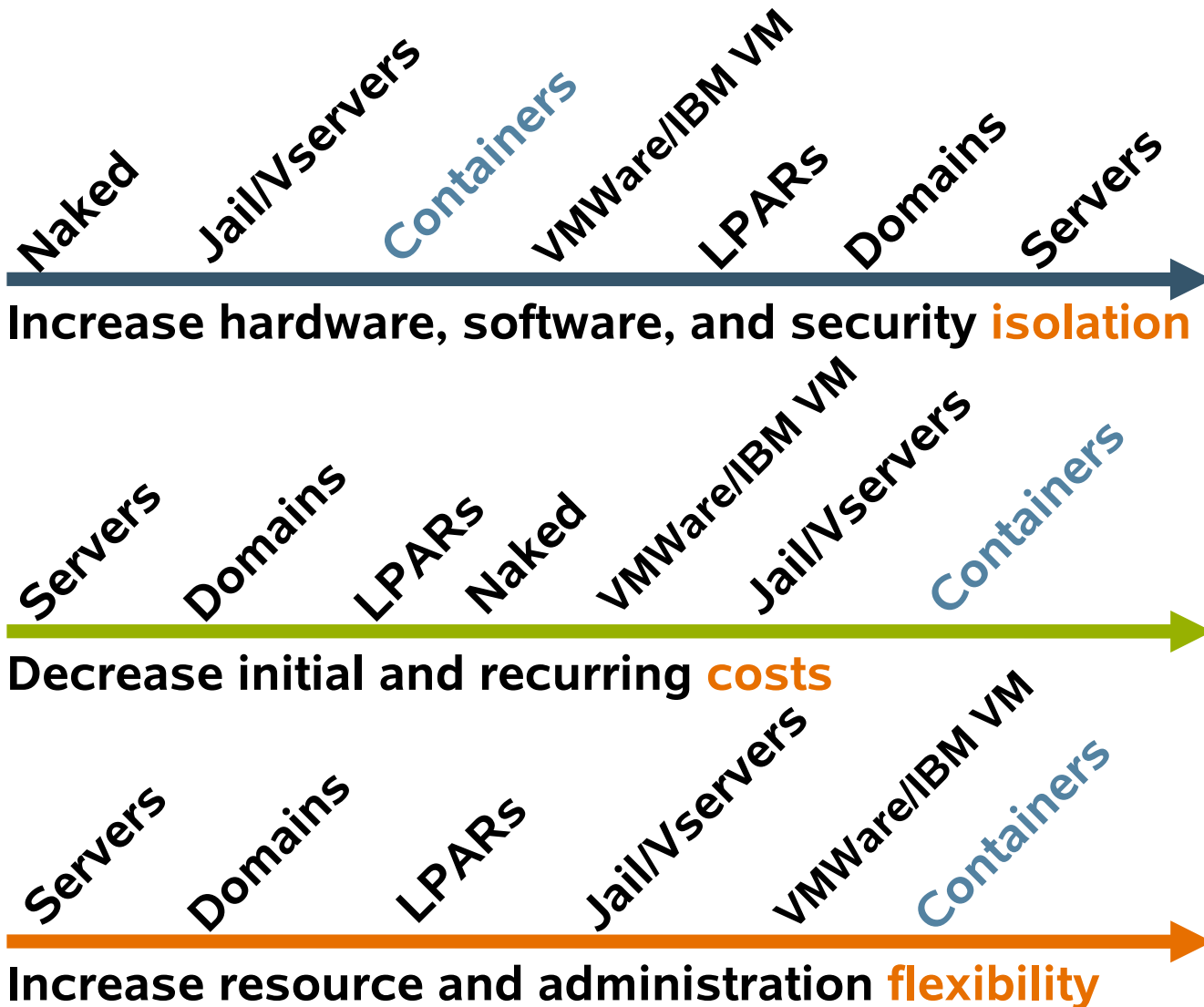
# Service Virtualization

# Dimensions Of Virtualization









- Resource Virtualization
  - > Partitioning of physical resources into fine-grained virtual parts for the purpose of improving resource utilization
  - > Abstraction of physical resources from applications for the purpose of improving service level RAS characteristics
- Application/Service Virtualization
  - > Creation of higher level aggregated application and service management abstractions for the purpose of
    - > Minimizing administrative costs
    - > Maximizing ease of administration
    - > Scaling out – creating a more sophisticated application/service deployment environments

# Current Virtualization Pitch



# Industry Virtualization Landscape

						
Virtual Execution Environments	Java Virtual Machine	X	C# CLR	X	Java Virtual Machine	X
Middleware Components/Containers	Sun J2EE JES Application Server	X	.NET Components/Containers	X	J2EE WebSphere App Server	X
OS Level Containers	Solaris 10 Containers	X	X	RPAR Resource Partitions	X	X
Server Virtualization	LDoms/Xen	VMware	Microsoft Virtual Server	VPAR Virtual Partitions	MVS VM	X
Hardware Domaining	All SunFire Servers 4900 and Higher	X	X	NPAR Virtual Partitions	LPAR Logical Partitions	X
Storage Virtualization	StorEdge 6920	Invista Symmetrix Connectrix	X	X	IBM Total Storage Tivoli	MDS 9000 VSANs
Network Virtualization	Sun Secure Application Switch N1400V	X	X	X	X	AON Application Network Switches

# Scenarios



# Mapping Onto The Use Cases

- Consolidation:
  - > Solaris Containers a great fit
  - > Lowest overhead, best observability, lowest maintenance effort
- Upgrade Solaris version:
  - > Zones not a fit, however only few instances are needed
    - > If the customer has dozens of software levels, they have other problems...
  - > Offer LDoms or Dynamic domains on SPARC, VMware on x64, and Xen when available for Solaris
- Heterogeneous OS coexistence cases (migration, etc)
  - > Zones not a fit – and only relevant on multi-OS x64/x86
  - > Offer VMware, and Xen when available for Solaris

# Mapping Onto The Use Cases

- Provide separate fault, security, admin domains of same OS level:
  - > Zones a perfect match for this use case, on either platform
  - > Provides the needed function with lowest overhead and complexity
- Relieve scalability constraints of a given OS via multiple instances
  - > Solaris more than handles x64 scale, more of a SPARC issue
  - > Few instances needed, so LDoms and Dynamic Domains fit
- Use legacy OS on newer kit (ex: run NT4 on current x86)
  - > Not applicable for us :-)

# Customer Scenario 1

- Customer requirements:
  - > Minimize (OS) management costs
  - > Fault and security isolation
  - > Fast recovery from failure
  - > Maximum utilization
  - > Low performance overhead
- Example: ISP offering web and e-mail services
- Solution: Solaris Containers

# Customer Scenario 2

- Previous example but with additional target of consolidating billing / accounting application
  - > Maximum (electrical) isolation of billing system
  - > Separate Solaris patches and release schedule
  - > High-end, mission-critical system
- Solution today: Solaris Containers + Dynamic System Domains
- Solution tomorrow: Solaris Containers + Logical Domains or Xen

# Customer Scenario 3

- Customer requirements:
  - > Multiple OS environments – Solaris, Linux, Windows
  - > Application certification and ISV support
  - > Hardware consolidation
- Example: Running older versions of the OS on newer hardware
- Solution today: VMware
- Solution tomorrow: Xen or VMware

# Customer Scenario 4

- Customer requirements:
  - > Multiple OS environments – Solaris, Linux
  - > Counteract the Linux Distro Sprawl
  - > Short term goal: Host Linux applications on consolidated system
  - > End goal: Migration to Solaris
  - > Use Solaris tools (DTrace, ...)
- Example: Migration of customer application to Solaris
- Solution: Solaris Containers for Linux Applications

# VIRTUALIZATION

**Carlos Piedrafita**

[carlos.piedrafita@sun.com](mailto:carlos.piedrafita@sun.com)