Styles of Virtualization

Jeff Savit
Solutions Architect, OS Ambassador
Sun Microsystems

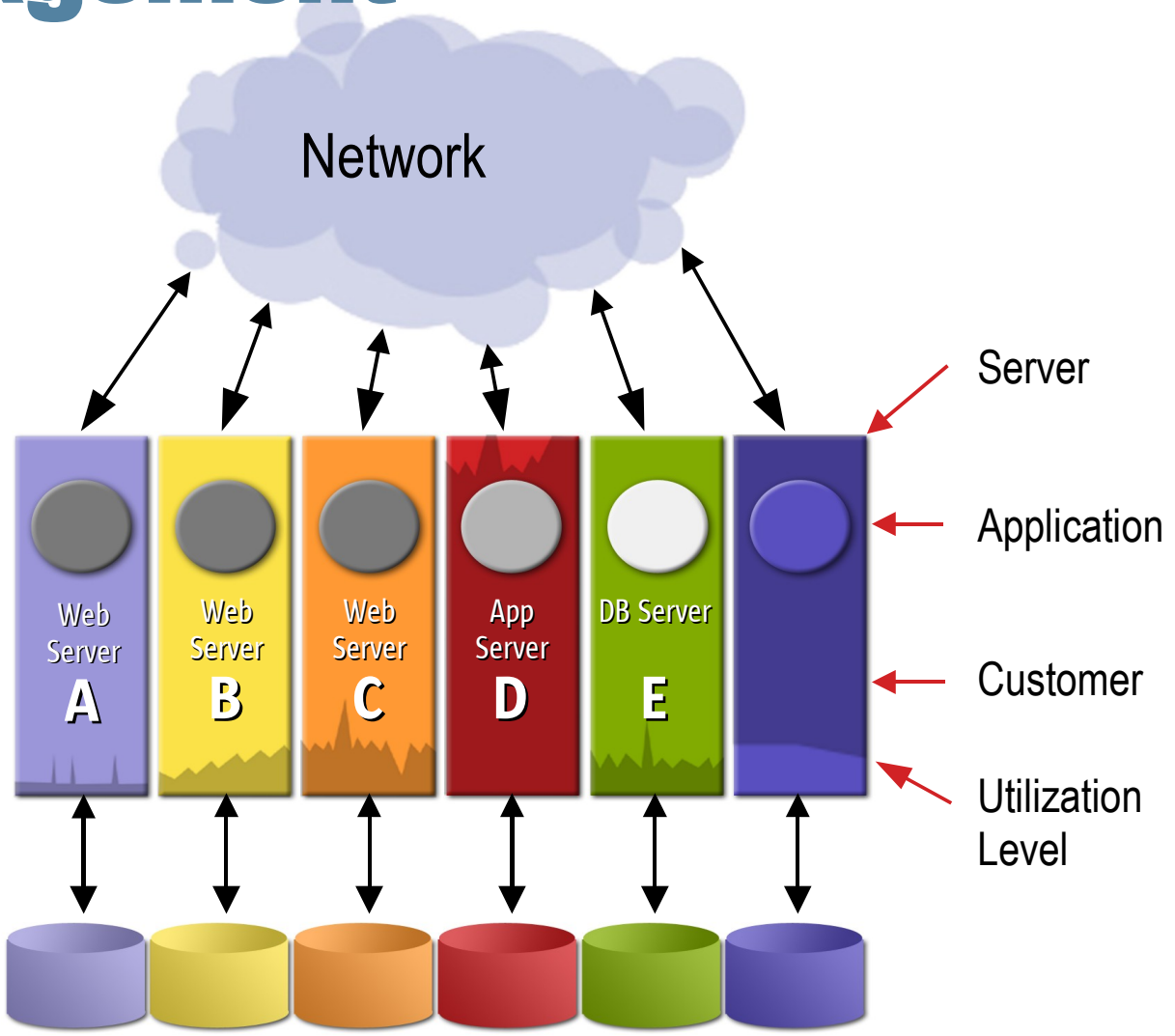
A survey of virtualization technologies

Situation Today - Driving Virtualization

- 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

Traditional Distributed Systems Resource Management

- One application per server
- One or more servers per customer
- Every server sized for peak workload
- Low average utilization rates
- Each server must be provisioned, powered, cooled, licensed, and managed.



Motivating factors

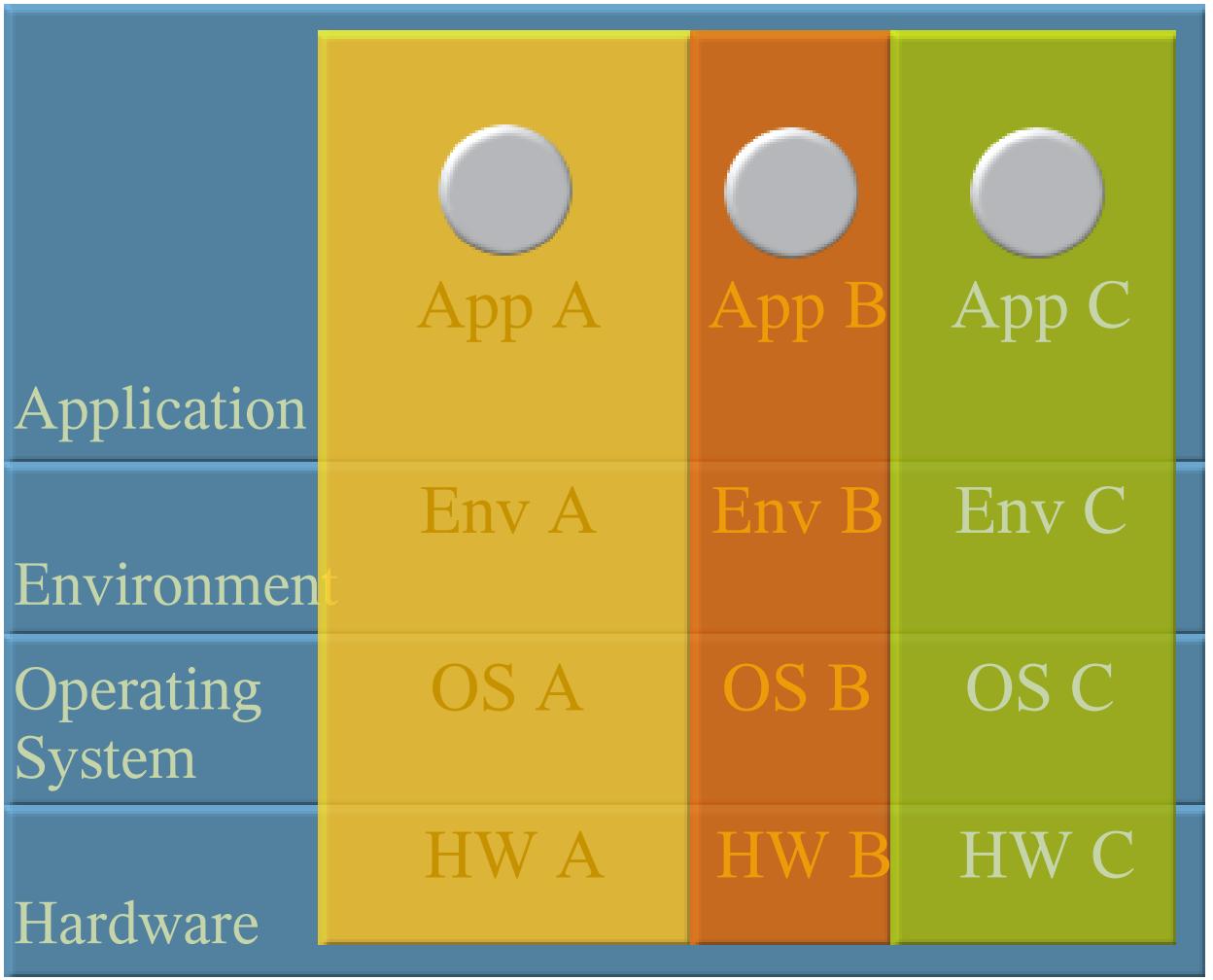
- Marketing-speak “Customers are interested in improving the utilization of their computing resources”
  - Translation: they want better return on the money they've spent
- One way to do this is server consolidation
- Consolidation requires (at least) the following
  - **1. Non-interference of independent 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...)

Styles of virtualization

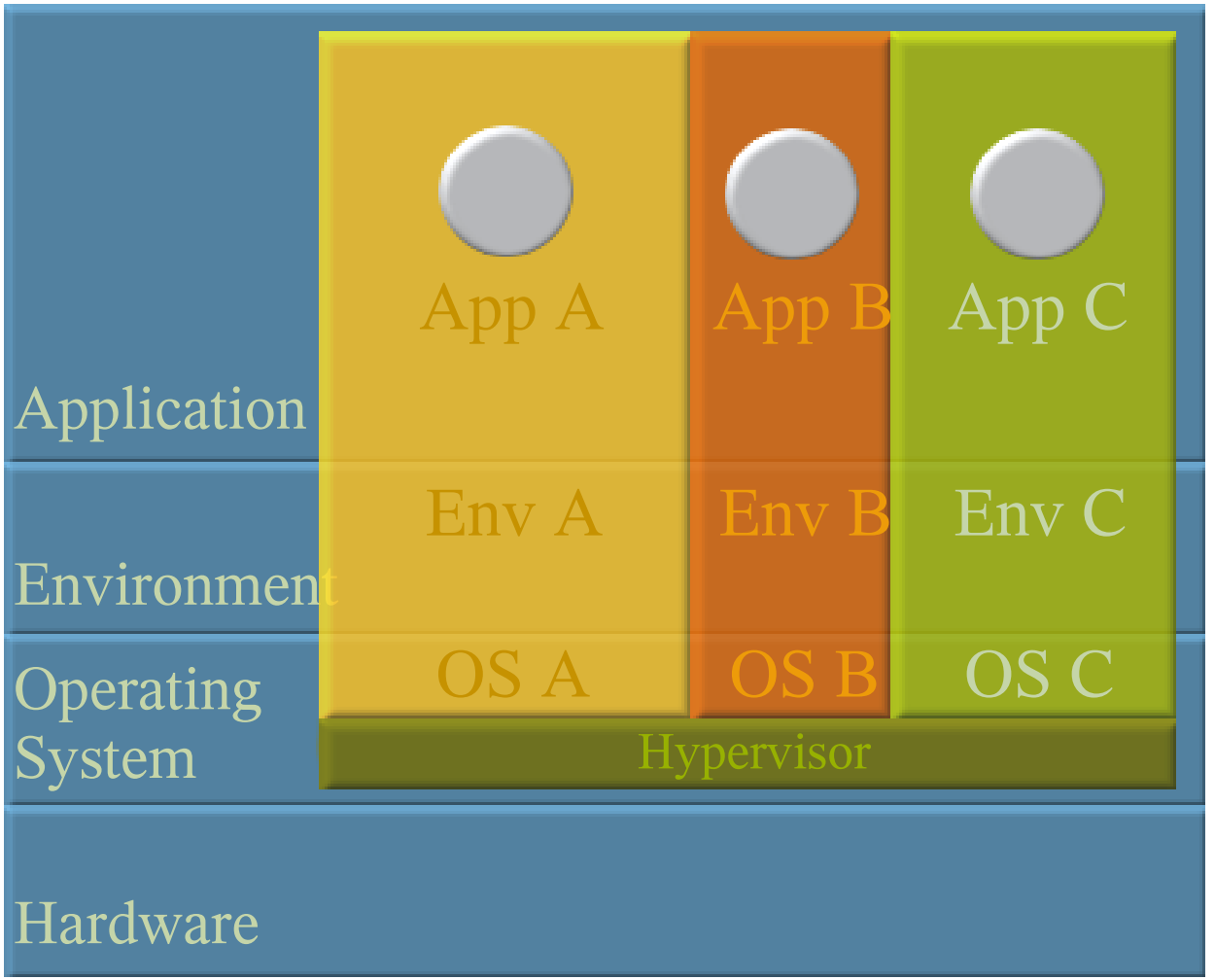
- Virtualization, present since the 1960s, now a mainstream technology available on multiple platforms
- There are different styles of virtualization with different benefits and limitations
  - this presentation does not discuss “outside the host” virtualization of network and storage
- Partitioning and virtual machines: hardware or a host OS (“hypervisor”) provides the illusion of a dedicated computer for each guest OS.
- Solaris provides its own virtualization: Containers
- This presentation surveys and discusses, with emphasis placed on recently created virtualization technology

Various forms of virtualization

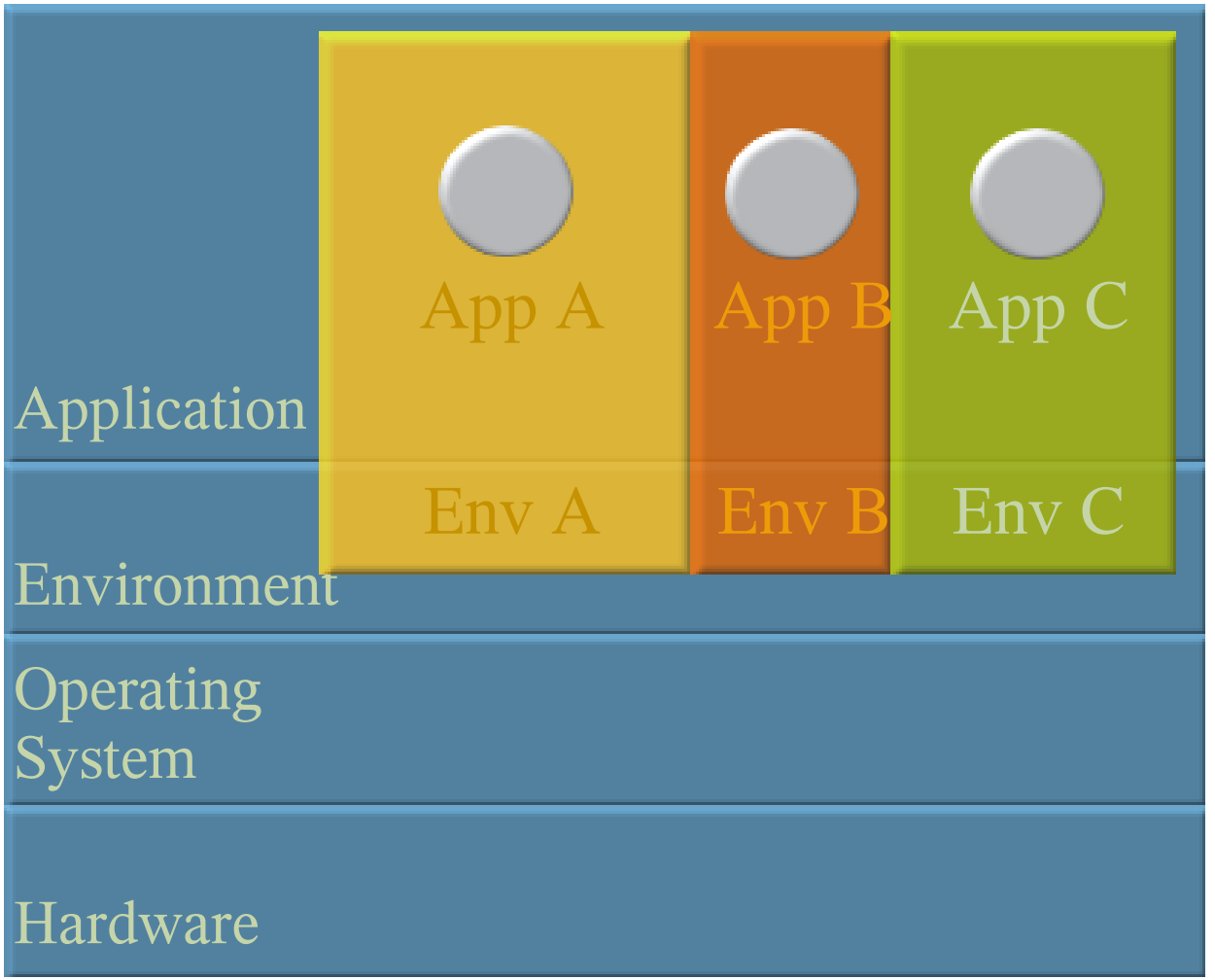
Hard Partitioning



Virtual Machines



OS Virtualization

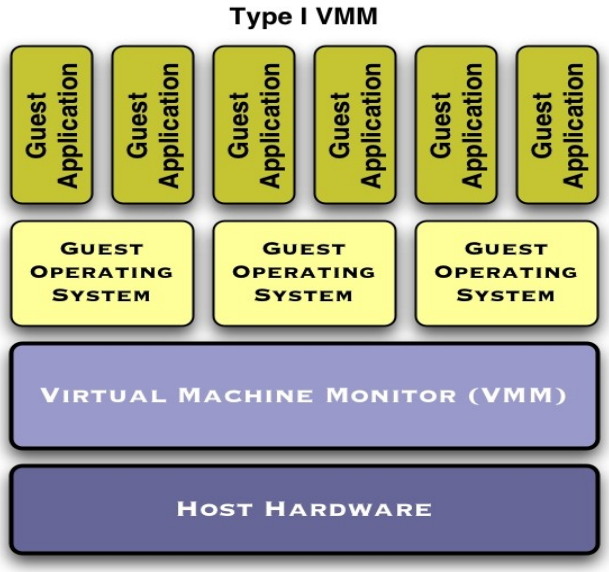


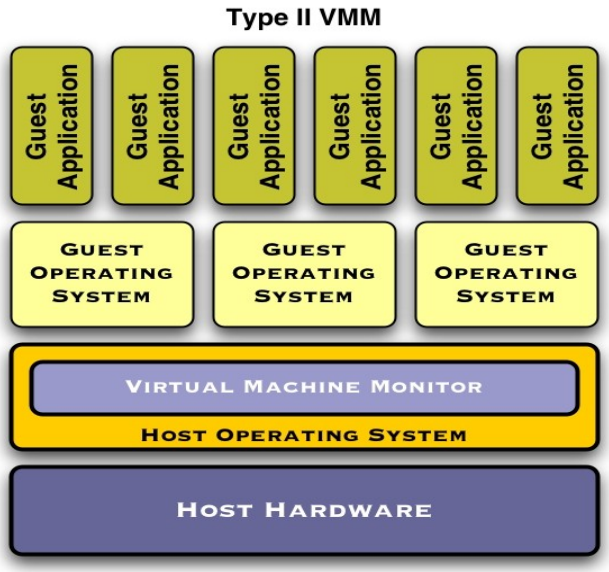
Partitioning

- Hardware/firmware oriented, less virtualization
  - Some offer virtual NIC, virtual disk
- Different degrees of isolation, granularity, and overhead
- On mainframe: Amdahl MDF, then IBM LPAR
  - CPU engines either dedicated to or shared by domains/partitions
  - Timer vs. event dispatching, differing schedulers, affect interrupt service latency and cache/TLB reuse
- LPARs on pSeries and iSeries (not same as mainframe LPAR)
- Sun domains: board-level granularity, scriptable reconfiguration (move board between domains), fault isolation
- HP's SuperDome vPARS, xPARS
- All: Typically fewer OS instances than with virtual machines

Different Virtual Machine systems

Virtualization – Architectural Patterns





- VMM Host ("hypervisor") creates virtual machines for "guest" operating systems
  - Guest OS runs in "user mode". VMM traps attempts to use privileged instructions, and emulates as needed
- Can run multiple, different, guest systems
  - Transparency to application, and often to guest OS as well
- Separate security, authentication, and failure contexts
  - aids security: well-defined interfaces (guest to guest, guest to VMM)
- VMM can be simple (no apps or userland)
- Synthetic or abstract resources (possibly better primitives for I/O, etc, than the "real machine")

Virtual Machine challenges

- Overhead depends on hardware platform, hypervisor, guest OS, and workload
  - Some architectures have complex semantics for privileged instructions; can be expensive to virtualize
  - I/O (disk, network), memory intensivity (or overcommittment), high context switch rate can make this expensive
- CPU, RAM footprint of multiple OS copies
  - Per-guest copies of binaries, daemons, and kernel processes (eg: housekeeping like buffer flush or Linux jiffies)
- Overall system management and observability
  - Guest OS in virtual machine is a black box; can be challenge for performance and debugging questions
  - Multiple OS instances to maintain, patch, administer

Tricky Virtual Machine topics

- Timing and dispatching
  - Example: guest OS “CPU consumed” clock shouldn't advance when guest not dispatched
  - Anti-social behavior: guests should not spin on locks or when idle
- Memory management challenges
  - Most OSes use all available RAM and don't release “guest real” memory. Poor locality of reference: WSS approaches VM's “size”
  - Map RAM (real memory, level 1) to “guest thinks it's real” (level 2) and guest's virtual memory (level 3) via *shadow page tables*
    - Traditional way (since VM/370, now VMware too) to manage “host real” to “host virtual == guest real” to “guest virtual” mapping. Can be costly
    - Valid translations must be present for HW. Translations flushed on both host and guest context switch – cost at hypervisor, cache, and TLB levels

Nested Resource Managers

A problem seen in several VM systems

- "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 of guest to host's load
  - 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 power (which reduces cost savings)

Mainframe VM

From VM/370 to current date

- S/370 architecture was virtualizable
  - Supervisor state instructions issued in problem state generate an exception, ("privop exception") which the hypervisor intercepts and emulates
  - Elegant and extensible (eg: could add synthetic instructions)
- Performance considerations
  - Privops and system calls (SVC) cost much more virtually than real
  - Context switch to hypervisor could dominate time. Flushes cache and TLB
  - Some things hard to do (example, the "interval timer"). Some workloads designed for "real" machine difficult to do efficiently (eg: MVS)
- Alleviated by microcoded assists, and subsequently by "interpretive execution" (firmware and hardware)
  - There is still cost for virtualization, but many exceptions are now handled without software intervention

VMware

- Provides virtual machines on x64/x86
  - Host OS Windows or Linux (VMWare Workstation or GSX)
  - VMware on bare metal (VMWare ESX)
  - Guest can be most unmodified x86 OSes
    - Linux, Windows, Solaris, BSD...
    - Virtual uniprocessor except with ESX, which allows 2-CPU guest
  - Solaris works as a guest. Sun partnering with VMware
- Current x86 architecture hard to virtualize
  - In some modes, privops are ignored, rather than causing a trap.
    - Silent “no operation” not acceptable to proper operation
    - ESX dynamically scans and rewrites portions of the guest's code, with caching of patched locations
  - Consequence is CPU overhead (with variable cost)

VMware – clever memory tricks

- Uses shadow tables method mentioned previously
- “Balloon” technique to handle storage contention at host level - reduces guest storage footprint:
  - guest thinks more of its RAM is in use, so doesn't assign to processes, but it isn't really, so guest working set can decrease
- ESX has hash-computed sharing of identical guest pages, with “copy on write” (COW) in case of alteration
- VMotion permits movement of a running virtual machine to a different real machine (for load balancing, or migration before a maintenance window)
- Can checkpoint a virtual machine to disk while its running

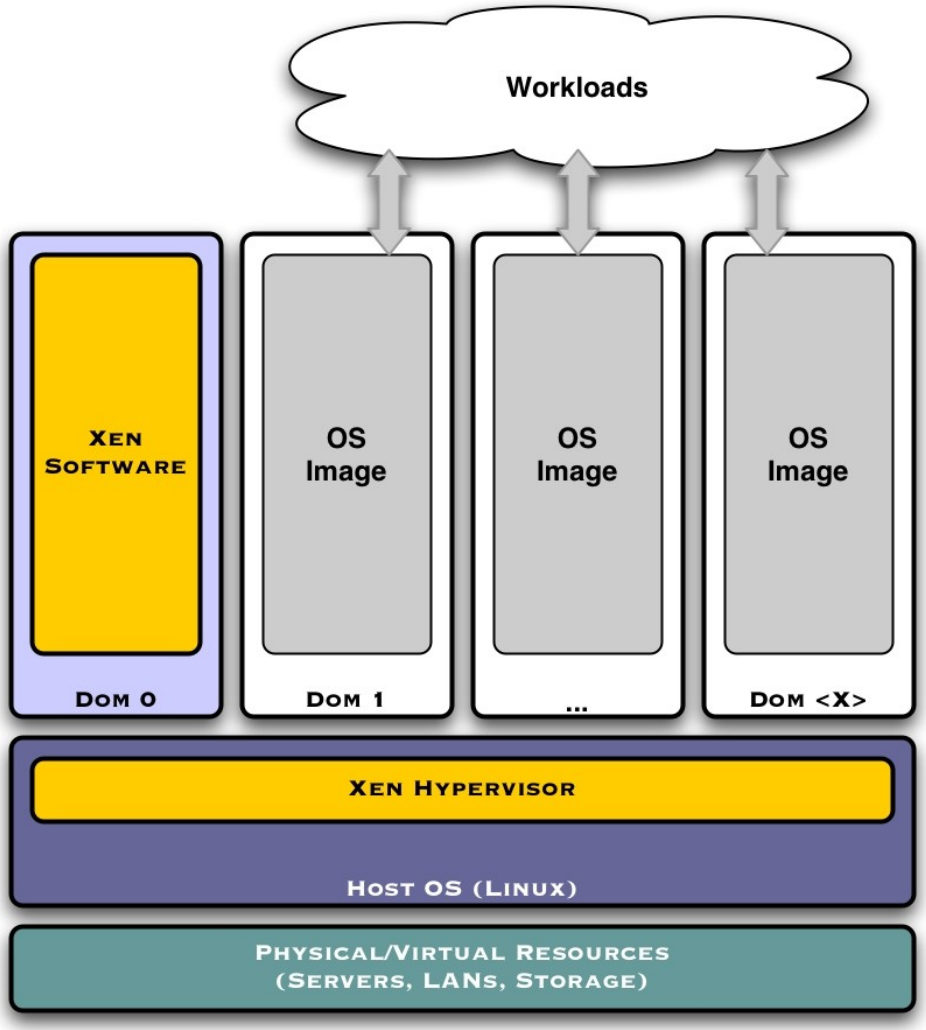
Microsoft's Virtualization

- *Virtual PC* (desktop) and *Virtual Server* products
- Virtual Machine Additions (VMA) for guest video driver and clock sync
- Windows-only for both guest and host
  - > Linux has been tested and seen to work, but is not supported
- Guests are 32-bit uniprocessors only
  - > Virtual Server 2005 R2 will support 64-bit
- Targeted for consolidation and for hosting legacy NT4 on current computers
- Windows System Resource Manager (WSRM) can manage CPU and RAM (Windows Server 2003 Enterprise Edition and Datacenter Editions)

Xen project

- For low cost hypervising on x86
- Runs modified guests in x86 ring 1 (normally ring 0)
  - Runs Linux, FreeBSD, NetBSD, Plan 9. Ports underway for Windows XP and Solaris (has been booted under Xen - see opensolaris.org)
- “Hypercalls”; synthetic I/O and memory management
  - Privops sent to Xen as calls (eg: TLB changes: guest sees TLB R/O, and updates via hypercall)
  - Reduce transitions to hypervisor; can batch requests for efficiency
  - Avoids scan and binary rewrites of guest OS code done by VMware
  - Understand virtual (VM dispatched) vs real time
  - Synthetic I/O architecture and I/O virtualization

Server Virtualization – Virtual Machines



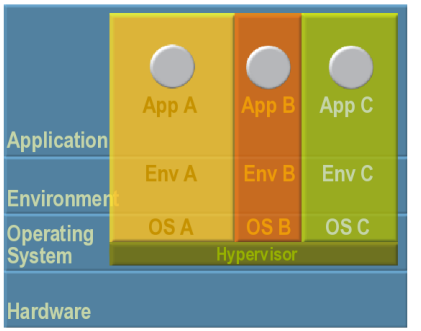
- An virtualized deployment environment for operating platforms
- Xen hypervisor is part of the host OS kernel
- Xen software resides in a special domain (Dom 0)
- Additional domains are instantiated on top of resources abstracted by the host OS/Xen hypervisor
- Guest OS's have to be modified in order to be hosted

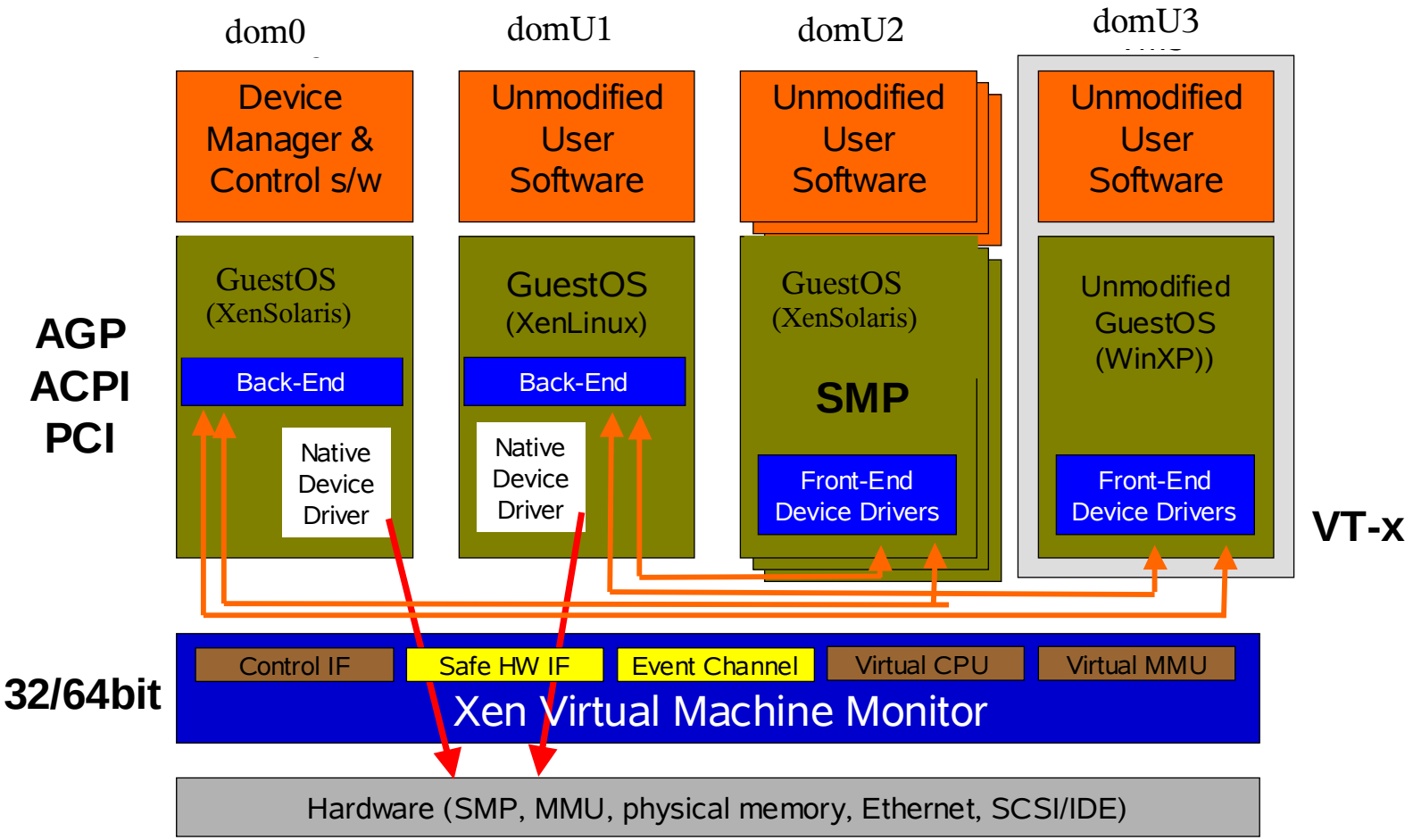
Xen project

- Seems to get 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 guest OS ports
- “Paravirtualization” can get nice effects if host and guest designed to work together
  - Like: batch multiple guest requests, state changes, or event notifications to reduce overhead
- Can relocate or migrate running virtual machines for load balancing or server maintenance
  - Incremental copy of changing pages while guest runs

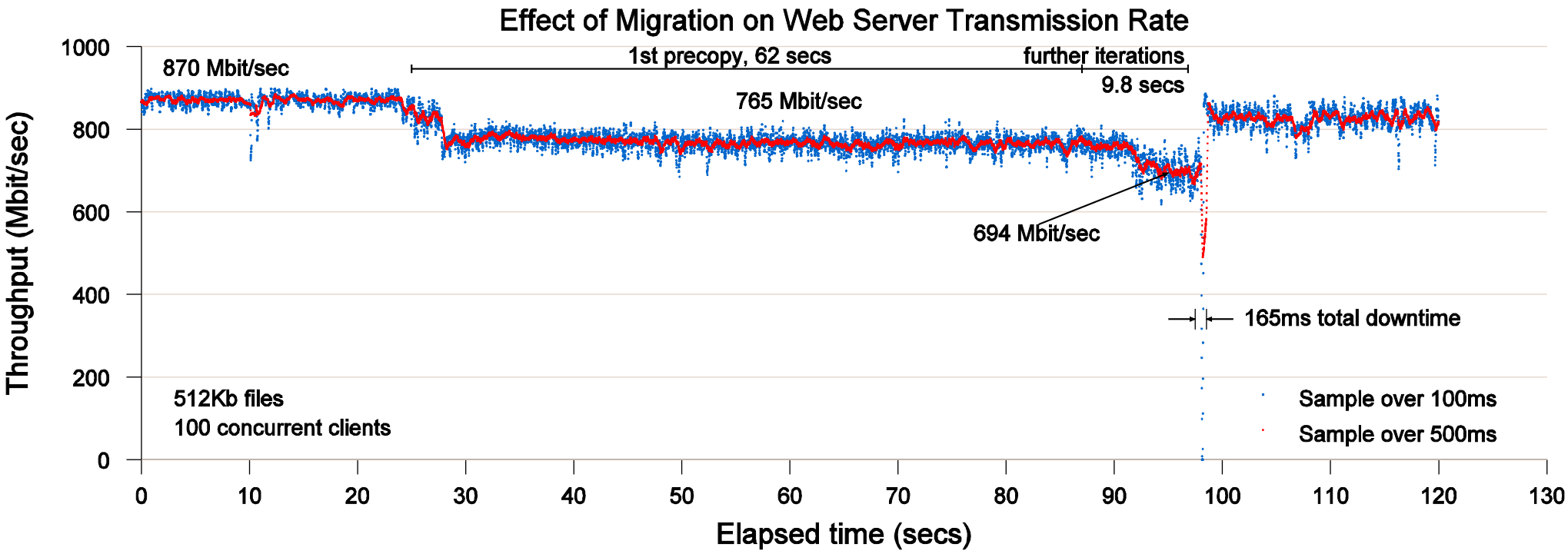
Virtual Machines

http://www.cl.cam.ac.uk/Research/SRG/netos/xen/





SPECweb99 Migration Experiment



From LinuxWorld 2005 Virtualization BoF

HP Virtual Server Environment (VSE)

- For HP Integrity (Itanium based) servers, due 12/05
- Allows multiple instances of HP-UX (initially) with Linux and Windows 2003 added in 2006 and 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

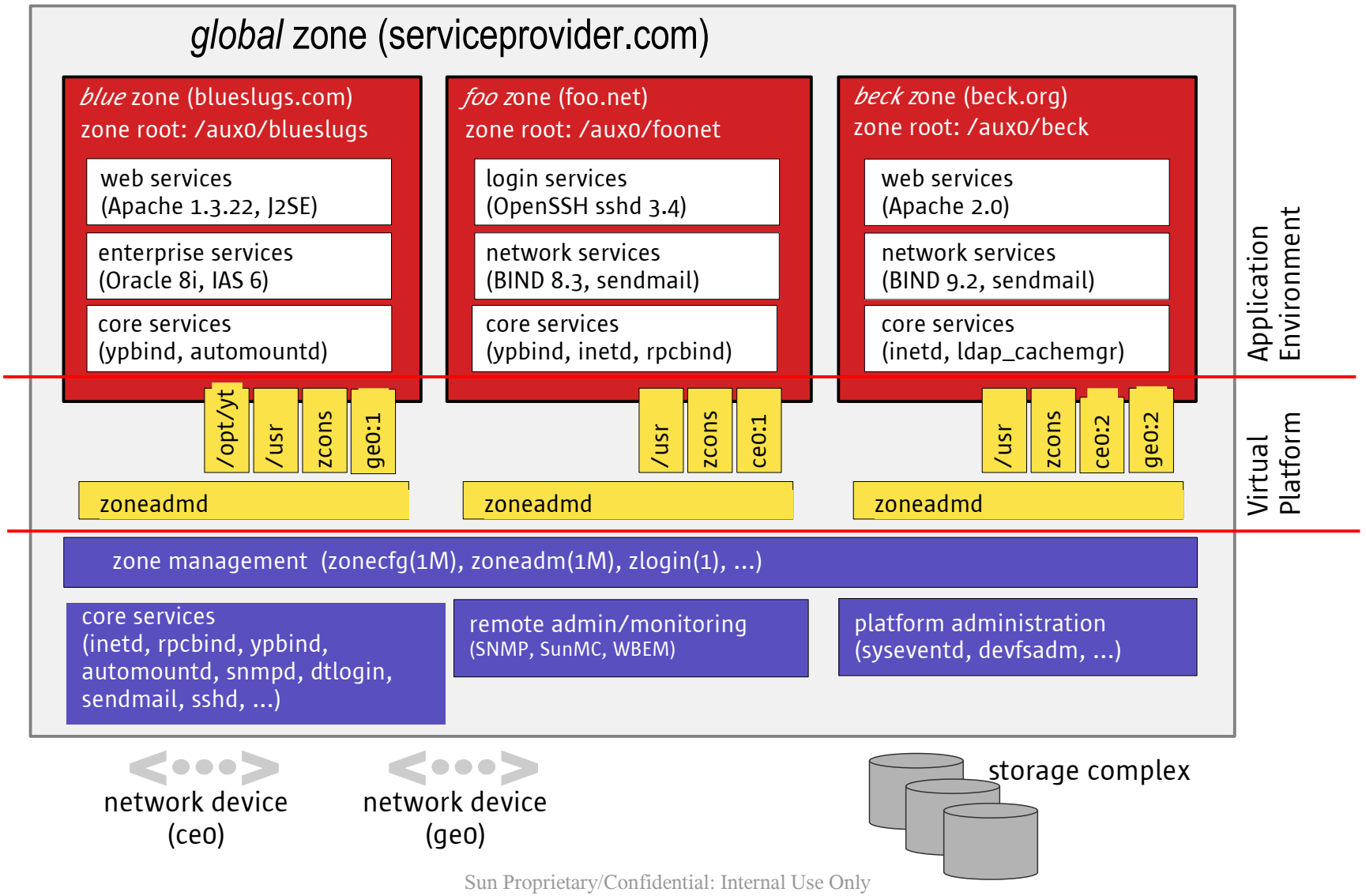
Solaris Containers ("Zones")

- Appearance of many OS instances, not many machines
  - Private name, IP address and port range
  - Private process lists and authentication (file, NIS, LDAP,...)
  - Can boot, reboot a zone, run rc.*N* scripts
  - Can create a new zone in a few minutes
  - Private file systems, with two flavors:
    - "**whole root**" - zone owns private, R/W file systems. Costs disk space, but you can separately manage patches, packages.
    - "**sparse root**" - zone inherits R/O: /usr, /sbin, /platform, /lib. Gets subset of packages from global zone. Saves disk space and RAM
- Separate security, resource management, and failure scopes
- Global zone administrator can give devices, UFS mount points, loopback filesystems, etc, to zone for its disk assets

Zone Performance

- A very lightweight process abstraction
  - 40MB of RAM per zone recommended but not necessary
  - Antique, small servers and workstations running 100s of zones
  - Zones add negligible overhead whether idle or in use
- Integrates with resource manager (granular CPU, I/O, RAM allocation to zones)
- Intra-zone networking at memory speeds
  - Solaris allows many IP addresses per network interface card for years; inter-zone communication via IP without going to NIC
  - Security benefit: doesn't transmit over wire; can avoid encryption
- Can share binaries in RAM across zones
  - Unix natively shares in-RAM copies of application binaries loaded from same file; no extra effort needed

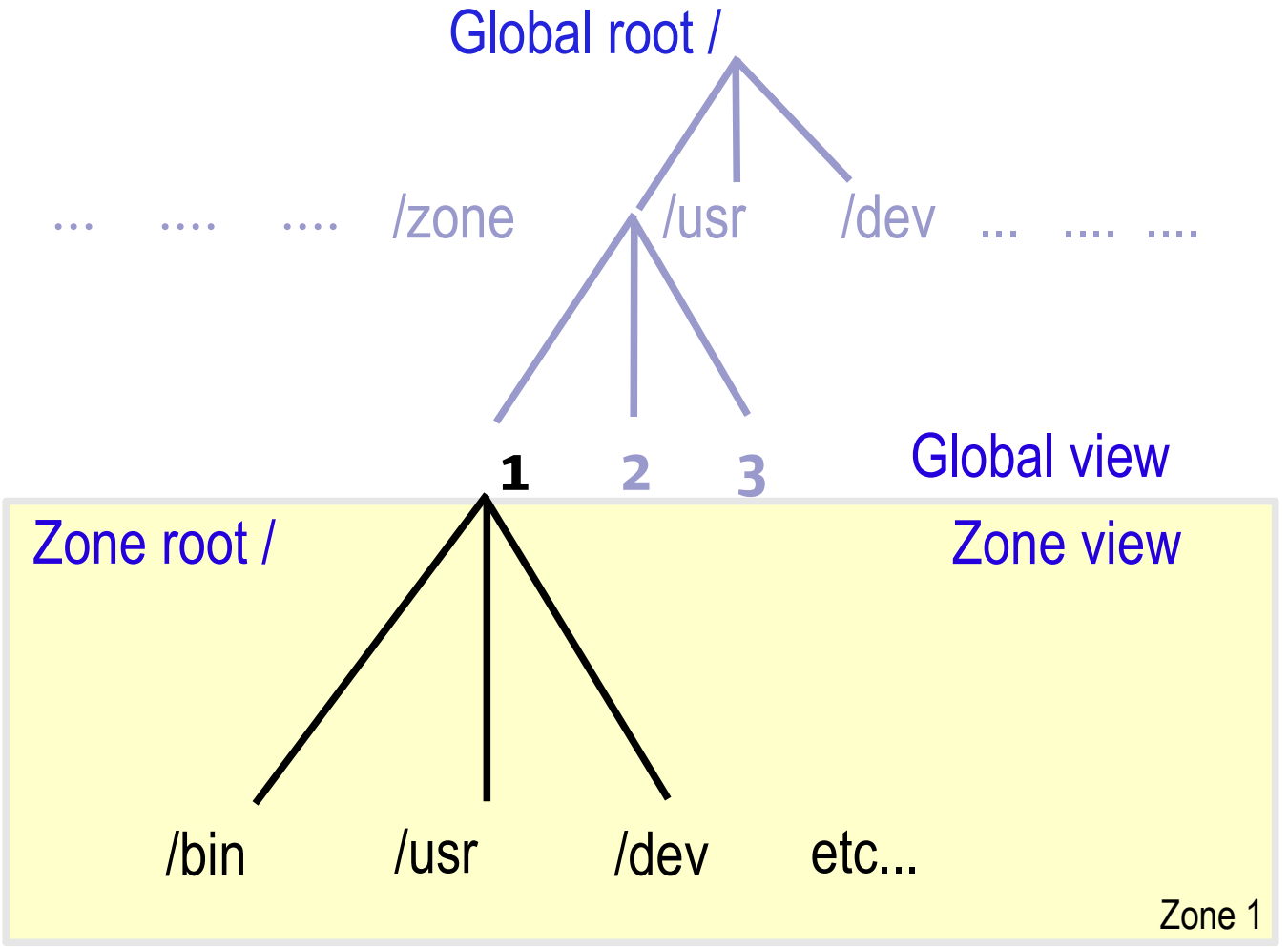
Zones Block Diagram



Zone Features

- Granularity
  - No dedicated physical devices (unless you want to)
  - Multiplexed resources
  - Arbitrary CPU, disk space, etc., granularity
  - 100+ Zones on a 1 CPU system (with load applied)
    - Throttle: disk space for unique zone files. ~100MB if sparse zone
- Visibility
  - local zone sees only its own processes (eg: “ps -ef”) and file system. Its “/” is a subtree within global zone's file system
  - root in global zone sees all processes and filesystems in each local zone, so can debug, tune, measure unimpeded
  - Suggestion: use global zone as management node

Zone File Systems



Zone Features

- Security within a zone
  - No access to other zones; in this sense it IS a different machine: doesn't see other zone's files, users, processes
  - Restricted root: local zone root has only local powers
  - Functions not allowed include
    - Reboot or shutdown of entire system
    - No access to kernel memory through /dev/kmem
    - No access to objects in other zones
- Cute trick: put web file directory in mount point read/only to the zone. Even if zone compromised the web contents can't be defaced

Zone Limitations

- Single OS “under the covers”
  - All zones share the same kernel, so can't (yet) have different levels of kernel function
  - If kernel dies lose all the zones (exactly the same as what happens if a virtual machine hypervisor dies)
- Local restrictions
  - Can't lock down real memory or load kernel modules, which some applications want to do
  - Privilege model doesn't yet allow some useful functions (eg: Dtrace) to run in a zone
  - Some install software assumes that userid==”root” means it can write to any directory – which is not true within a zone

Virtualization at the API level

Linux Application Environment

- **Linux Application Environment (LAE) planned for a Solaris update**
  - Run Linux and Solaris apps on same box without separate OS instances
  - A Linux **personality and runtime environment** for Solaris
    - Provides Linux **binary compatibility** for Solaris
    - System call, signal delivery, file system, etc, map onto Solaris native
    - Linux Standard Base (**LSB 1.3**) compliance for Solaris
  - Performance target: within 5% of Linux on same machine
  - Can utilize Solaris features like DTrace, SRM, Zones, etc, that don't exist in Linux
  - It is not an implementation of Linux or a Linux system
    - Does not support Linux device drivers or Loadable Kernel Modules
    - Not 100% Linux compatible and never will be
    - Not available for SPARC (LAE does NOT emulate x86/64)

What applications run?

- We have installed and informally tested:
  - Opera, Adobe Acrobat reader, Hancom Office, Majesty, BEA Weblogic Server, Oracle Database Server, StarOffice 7, Samba, Apache, and most of the core RedHat Advanced Server 3.0 binaries.
- Any LSB1.3 compliant commercial application
- We have tested and provided Linux installation “helper” scripts for RedHat Advanced Server 3.0
  - We will provide helper scripts for other versions of Linux in the future and these will of course, be supported by Sun
- CentOS, binary code of RHAS without copywrite encumbrance, also works (and no license fee)

How and where is the Linux runtime environment installed?

- Install into a local zone
  - Separates Linux environments from Solaris ones
- Customer must acquire a Linux distribution
  - Currently this must be the CD media for RedHat Advanced Server 3.0
  - We provide a script (install_linux_env.sh) to install the packages from the distribution into the Solaris file system name space
    - Creates LAE namespace, installs support files
    - Uses Linux **rpm**(1M) command to install the Linux image
    - Turns the LAE feature on
- Apps can be moved in by creating tar file on Linux and transferring it

Summary:

Virtualization Alternatives

Virtualization Alternatives

- Partitioning
  - Excellent separation, but constrains number of instances
  - Some implementations have substantial overhead
  - Manageability, observability can be challenging
    - Multiple OS instances not much better than maintaining multiple boxes
- Virtual Machines
  - Very good isolation and support for heterogeneous software
  - Overhead (CPU, RAM footprint) can be high (YMMV)
  - Manageability, observability can be challenging
- Containers (Zones, LAE)
  - Low overhead and small footprint permit **many** instances
  - Good isolation **in the context of a single kernel,** hence not for heterogeneous OS

Predictions

- The system vendors will enhance their virtualization products now that it's mainstream and understood
- Expect maturation of offerings, emergence of multi-vendor and open-standard APIs
- Different forms of virtualization will be applied in different use cases depending on needs for isolation, flexibility, performance, manageability. There will be overlapping areas where choices will not be clear-cut
- This area will be rapidly evolving - Watch this space!

Styles of Virtualization

Jeff Savit
jeff.savit@sun.com