Virtualization

Ruini Xue

School of Computer Science and Engineering

University of Electronic Science and Technology of China

2016

Problem

- Enterprise IT centers support many service applications
  - Microsoft Exchange
  - Oracle
  - SAP
  - Web servers
  - Citrix
  - …
- Each service application demands its own environment
  - Specific version of operating system
  - Multiple processors and disks
  - Specialized configurations
  - …

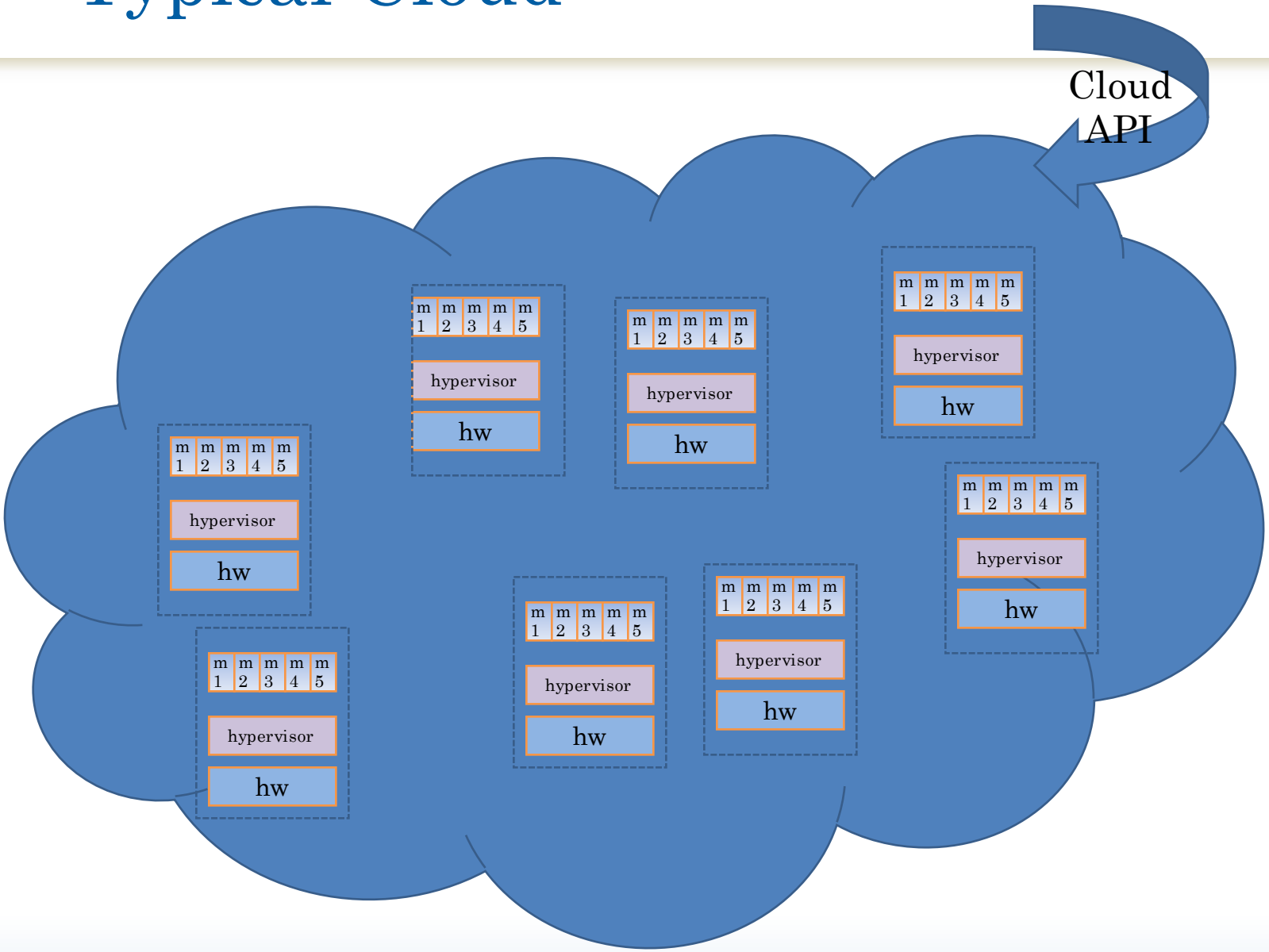
Problem (continued)

- Combining services on same server host is difficult (at best)
  - Conflicting demands
  - Incompatible loads
  - …
- Upgrading or commissioning a service is very difficult
  - Shadow server machines for debugging & testing
  - Complicated changeover tactics
  - …

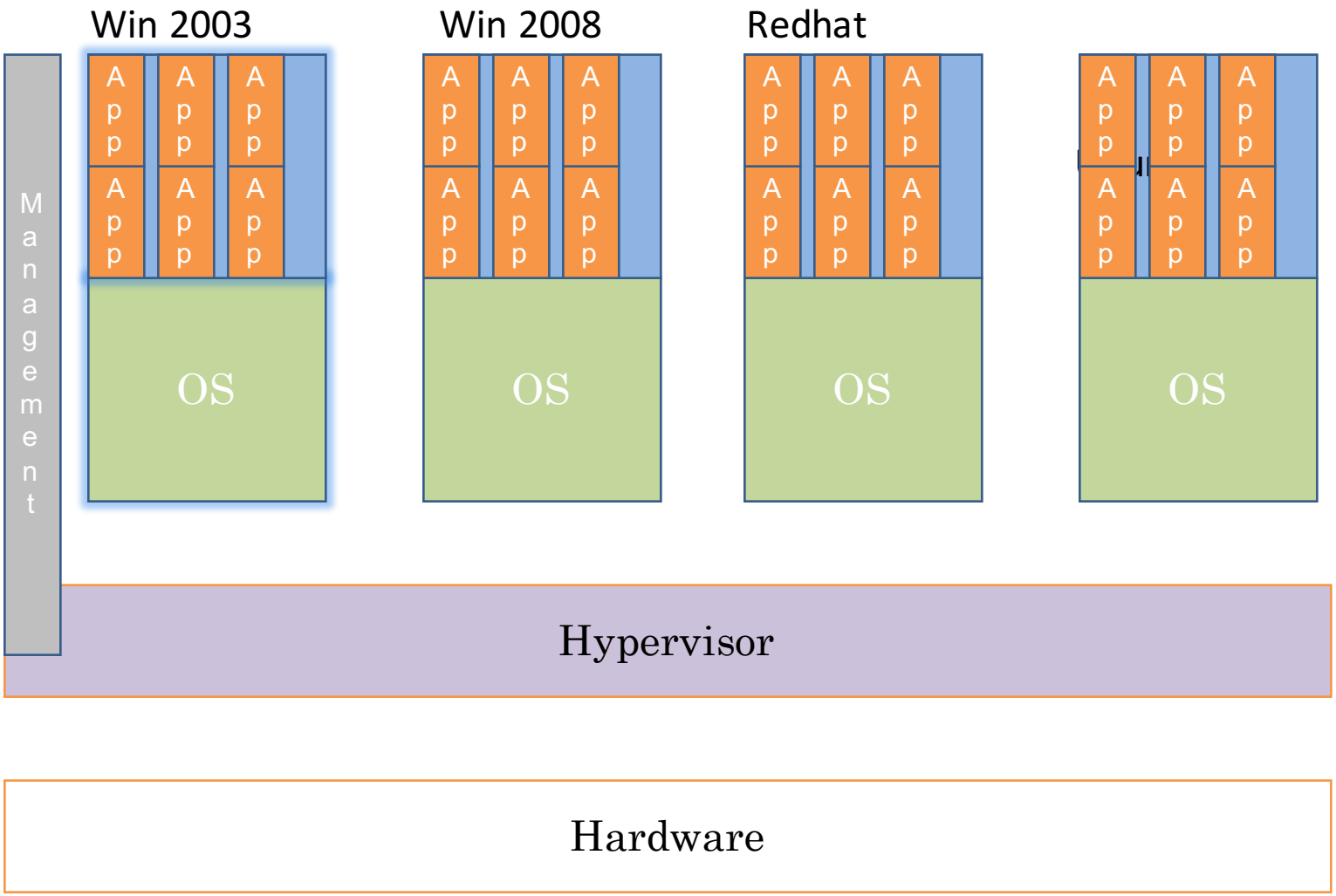
Problem (continued)

- Adding or upgrading hardware or OS is difficult
  - Testing and refitting active service
  - Complicated changeover tactics
  - …
- Load balancing is impossible
  - Services tied to own systems
  - Some underused, some overused
- Provisioning

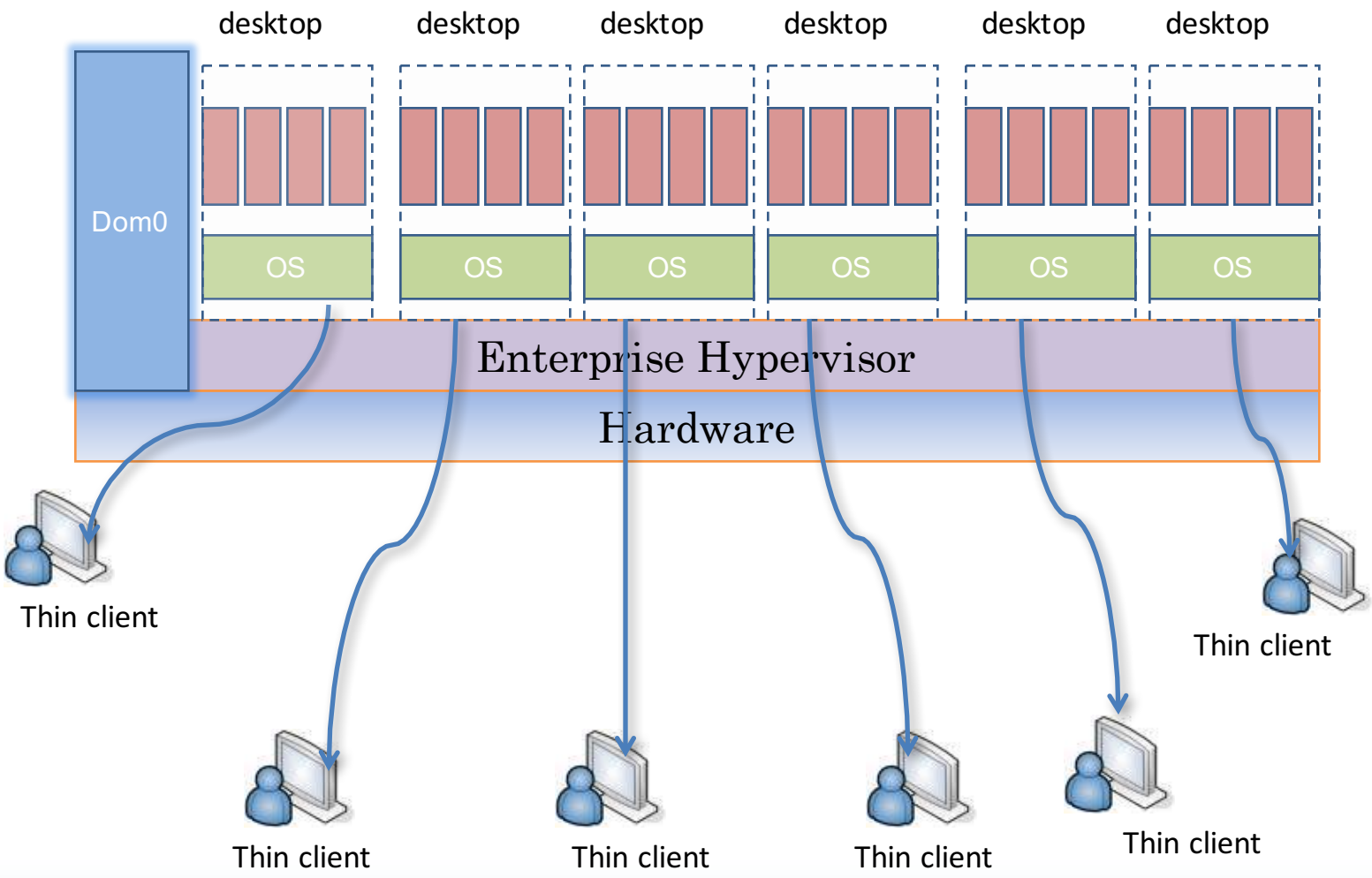
Typical Cloud



Typical Server Virtualization Deployment



Typical VDI Deployment



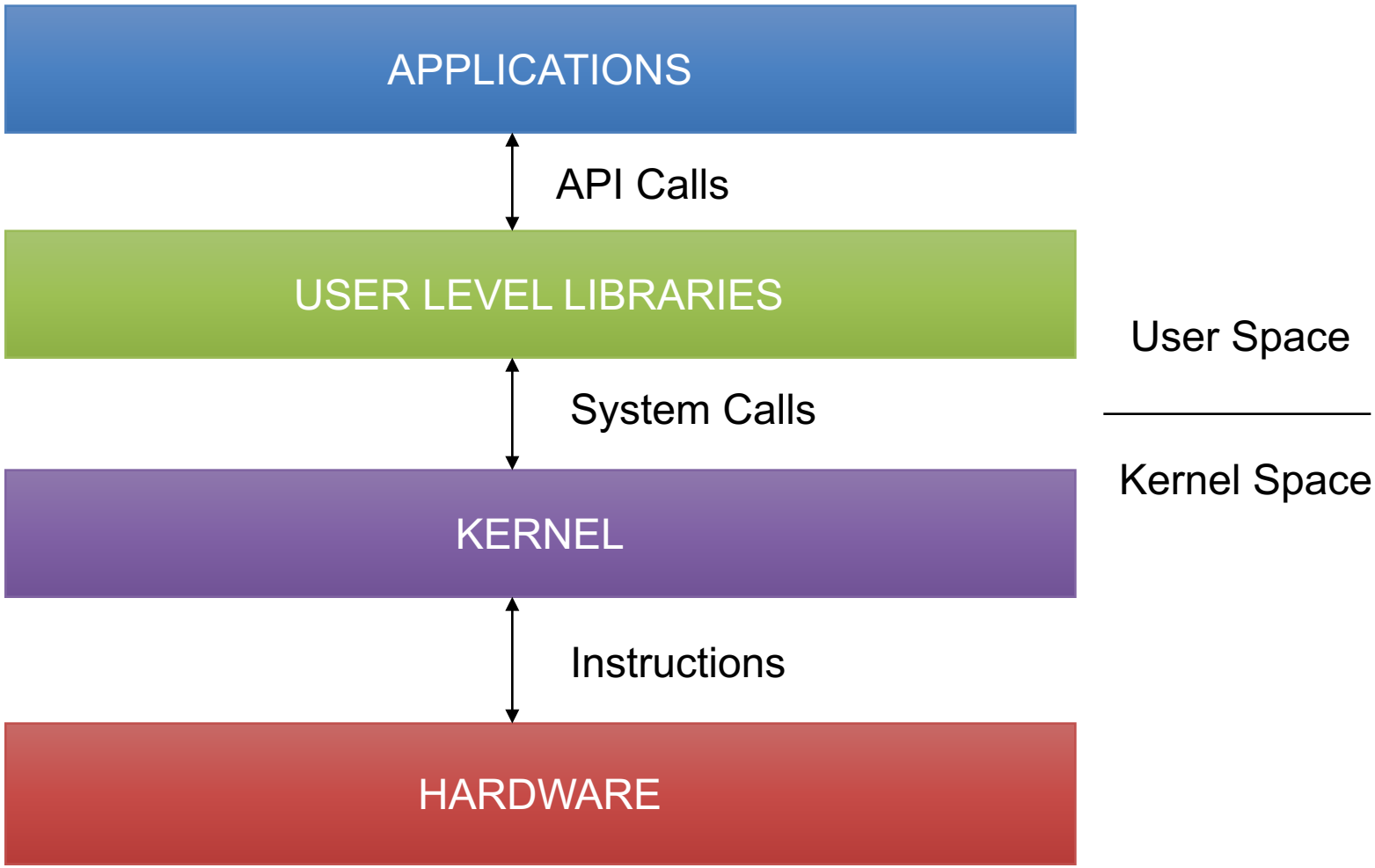
Virtualization: What is it?

- A framework that combines or divides [computing] resources to present a transparent view of one or more environments
  - Virtualizaion != VM
  - Hardware/software partitioning (or aggregation)
  - Partial or complete machine simulation and emulation
  - Time-sharing (in fact, sharing in general)
  - In general, can be M-to-N mapping (M "real" resources, N "virtual" resources)

Virtualization: Why?

- Server consolidation
- Application consolidation
- Sandboxing → security
- Multiple execution environments
- Debugging
- Software migration (Mobility)
- Testing/Quality Assurance
- Cost/Price
- Provisiong flexibility

Virtualization Levels



Categories

- Instruction Set Architecture Level
  - Bochs/Crusoe/Qemu/BIRD/Dynamo → Binary translation
- Hardware Abstraction Layer (HAL) Level
  - VMWare/Virtualbox/Denali/Xen/L4/Plex86/User-mode Linux/Cooperative Linux
- Operating System Level
  - Jail(chroot)/Virtual Environment/Ensim's VPS/FVM
- Library (user-level API) Level
  - Wine/WABI/LxRun/Visual MainWin
- Application (Programming Language) Level
  - JVM/.NET CLI/Parrot

ISA Level Virtualization

- Technologies
  - Emulation: translates guest ISA to native ISA
  - Emulates h/w specific IN/OUT instructions to mimic a device
- Problem
  - large overhead
- Solution or optimization
  - Translation cache: optimizes emulation by making use of similar recent instructions

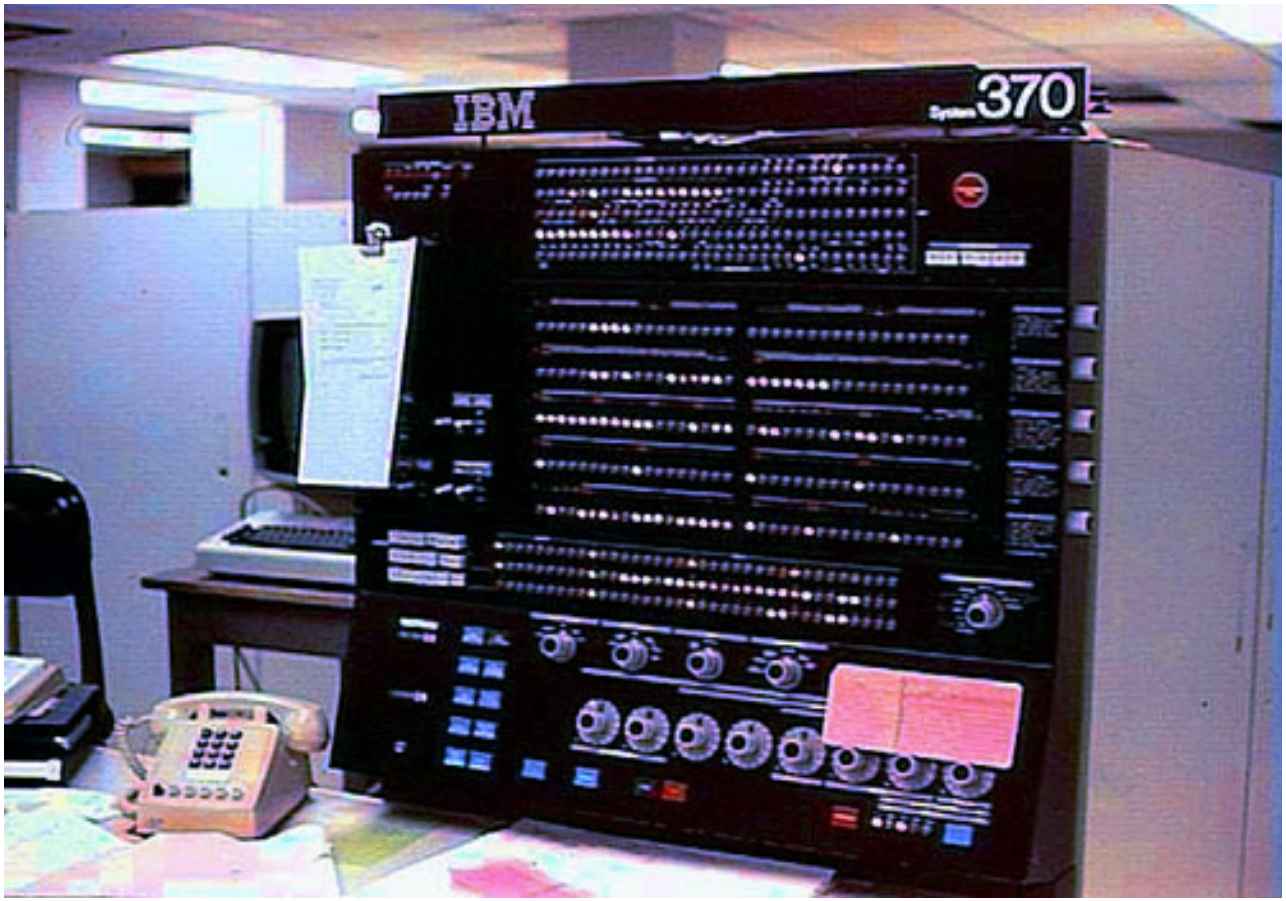
CPU Emulator

- Bochs: Open source x86 emulator
  - Emulates whole PC environment
  - x86 processor and most of the hardware (VGA, disk, keyboard, mouse, …)
  - Custom BIOS, emulation of power-up, reboot
  - Host ISAs: x86, PowerPC, Alpha, Sun, and MIPS
- QEMU:
  - Full Implementation: running OS on top of it
    - Multiple target/host ISAs: x86, ARM, PowerPC, Sparc
    - Supports self-modifying code (write-protected)
  - User-space only: useful for cross-compilation and cross-debugging

HAL Level Virtualization

- Basic requirements:
  - Multilevel privilege domains: VMM and guest OS
  - A trap needs to occur during privileged instruction execution
  - VMM takes over and keep each VM isolated
- VM: A virtual machine (VM) is a software implementation of a machine that executes programs like a physical machine.
- VMM: A hypervisor or VMM is a software to provide a hardware emulation interface including CPU, memory, I/O by multiplexing host resources.

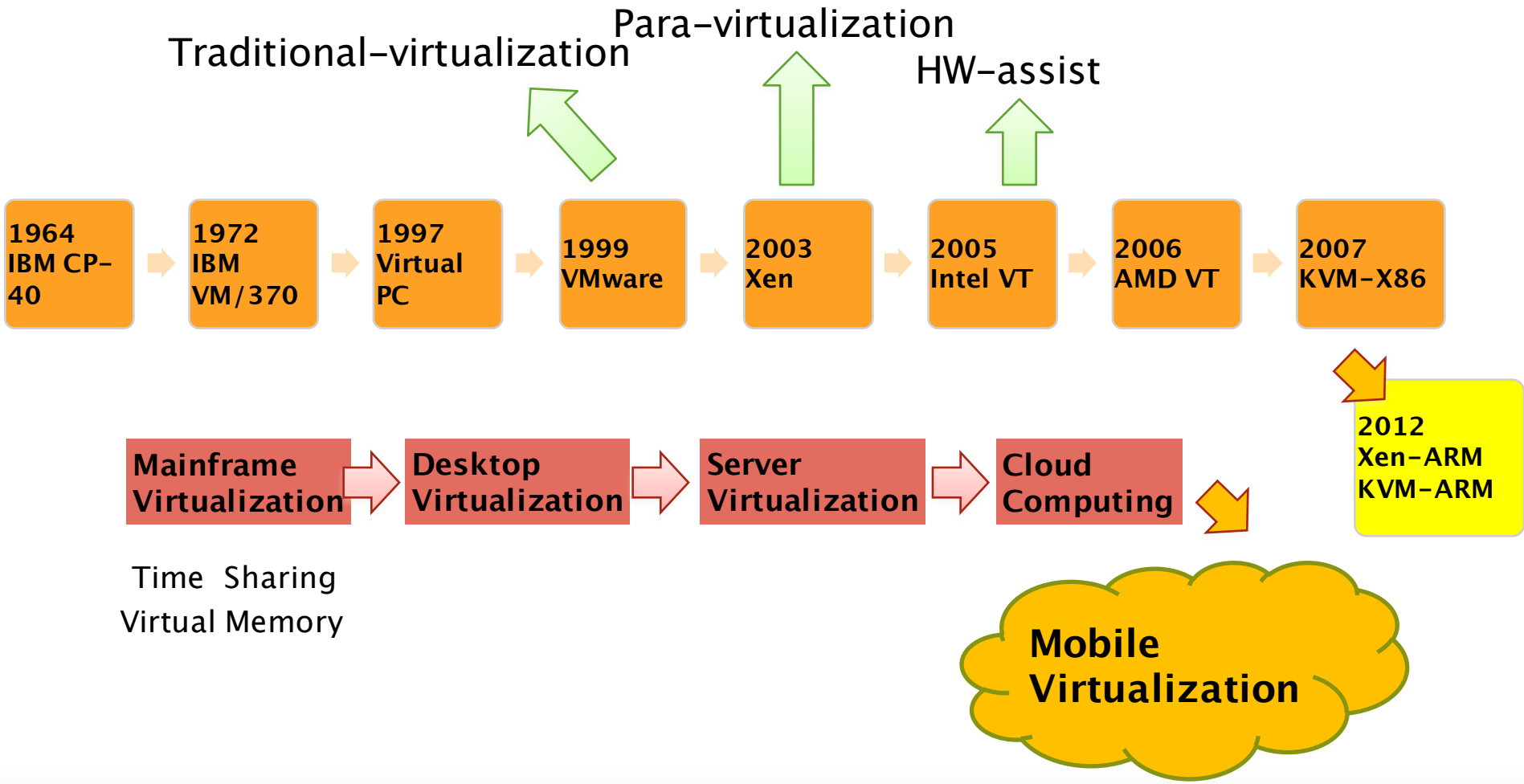
IBM-370



History

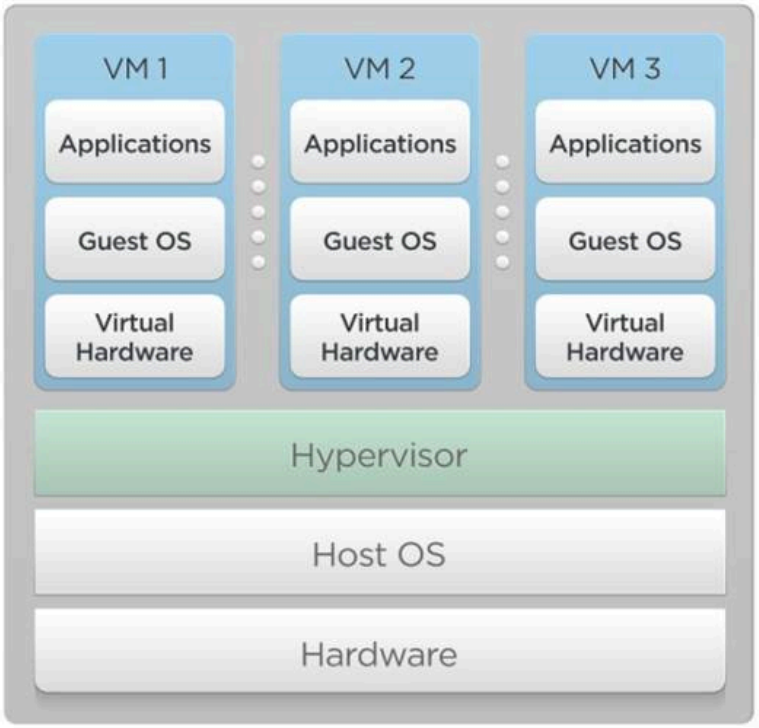
- IBM VM/370 – A VMM for IBM mainframe
  - Multiple OS environments on expensive hardware
  - Desirable when few machine around
- Popular research idea in 1960s and 1970s
  - Entire conferences on virtual machine monitor
  - Hardware/VMM/OS designed together
- Interest died out in the 1980s and 1990s
  - Hardware got cheap
  - OS became more powerful (e.g multi-user)
- Today:
  - Processors very fast; but administration, reliability, security is costly

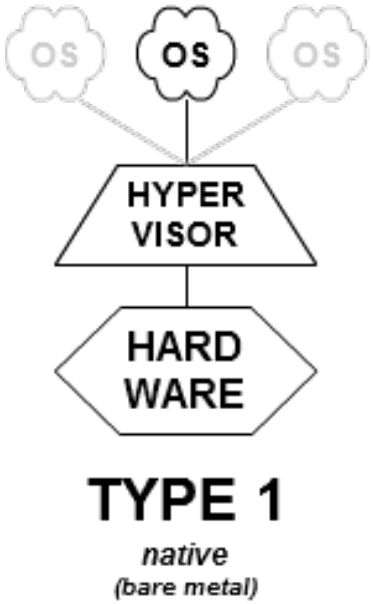
History of Virtualization

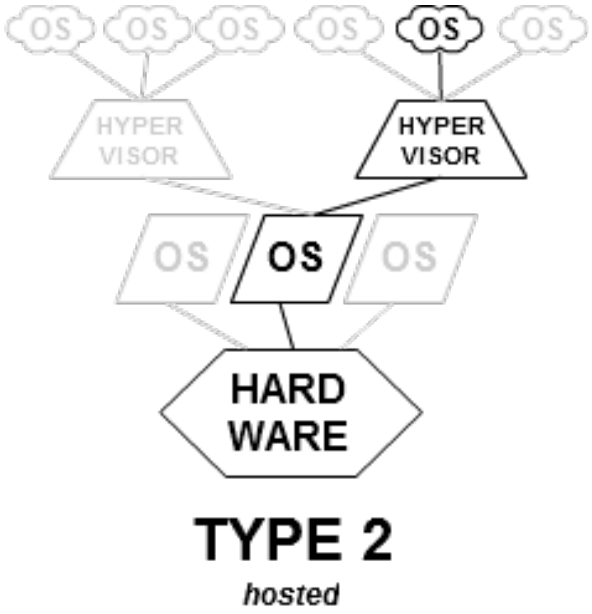


Virtual Machine

Hypervisor


VM 1
Applications
Guest OS
Virtual Hardware
VM 2
Applications
Guest OS
Virtual Hardware
VM 3
Applications
Guest OS
Virtual Hardware
Hypervisor
Host OS
Hardware


OS
OS
OS
HYPER VISOR
HARD WARE
TYPE 1
native
(bare metal)


OS
OS
OS
OS
OS
OS
HYPER VISOR
HYPER VISOR
OS
OS
OS
HARD WARE
TYPE 2
hosted

Hypervisor

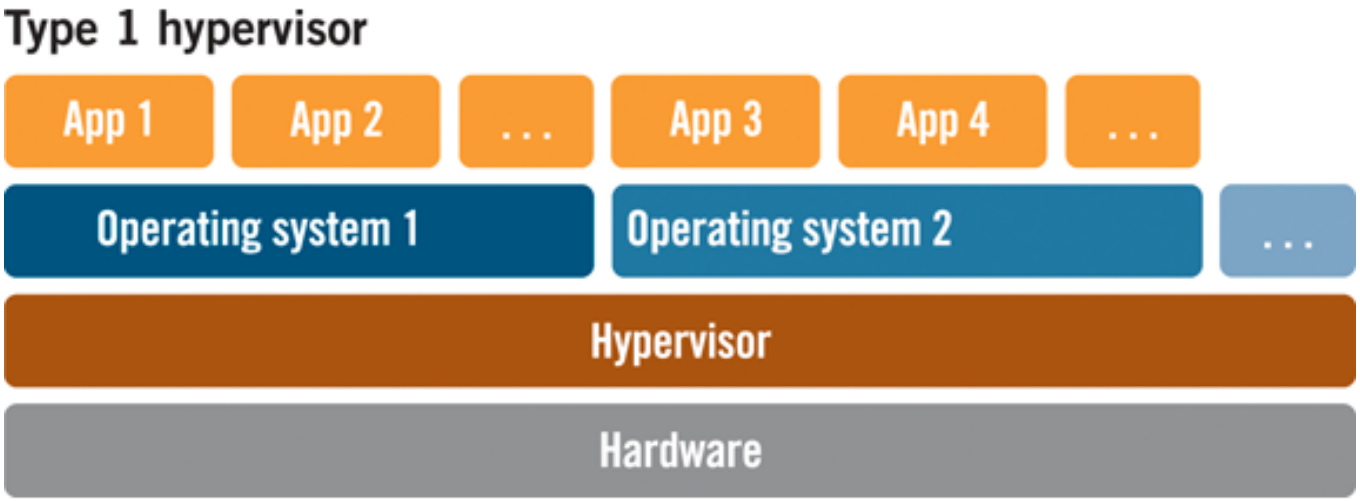


Figure 2. A Type 1 or bare-metal hypervisor sits directly on the host hardware.

Xen™ vmware ESXi Microsoft Hyper-V

Hypervisor

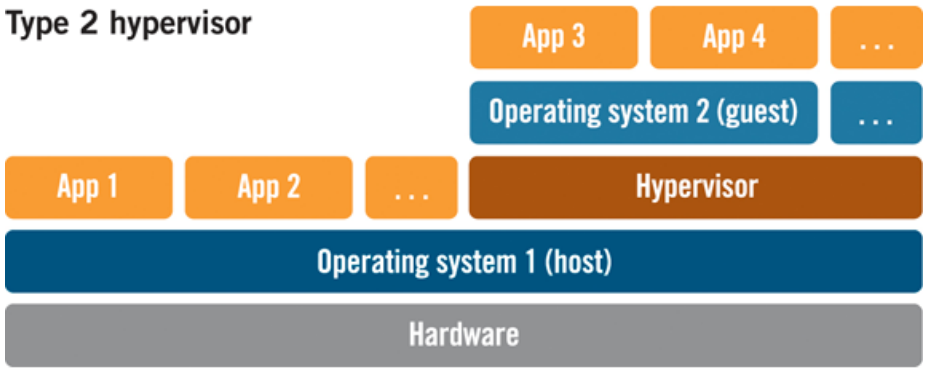


Figure 1. A Type 2 hypervisor runs as an application on a host operating system.

vmware®

Parallels™

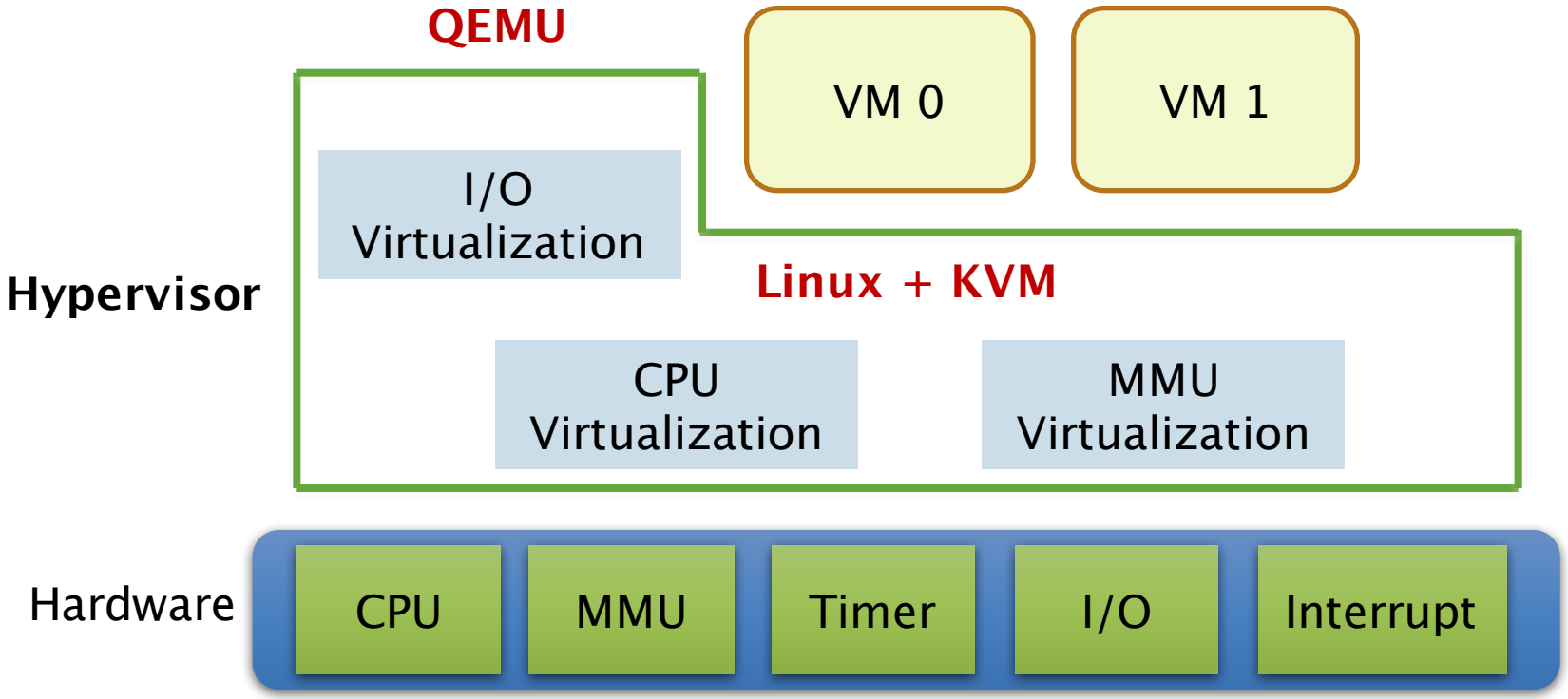
Implementations of Hypervisor

- Full Virtualization
  - A wholly emulated virtual machine makes guest operating system binary can be executed directly without modifying guest source code
  - For efficiency, it needs hardware-assisted virtualization
- Para-Virtualization
  - Hypercalls are defined and used in a guest operating system to make a virtual machine abstraction
- Pre-Virtualization
  - By compiling technique, guest operating system binary or source could be compiled for virtualization

KVM

- KVM (Kernel-based Virtual Machine)
  - Linux host OS
    - The kernel component of KVM is included in mainline Linux, as of 2.6.20.
  - Full-virtualization
    - KVM is a full virtualization solution for Linux on x86 hardware containing virtualization extensions (Intel VT or AMD-V).
    - Using KVM, one can run multiple virtual machines running unmodified Linux or Windows images.
  - IO device model in KVM :
    - KVM requires a modified QEMU for IO virtualization framework.
    - Improve IO performance by virtio para-virtualization framework.

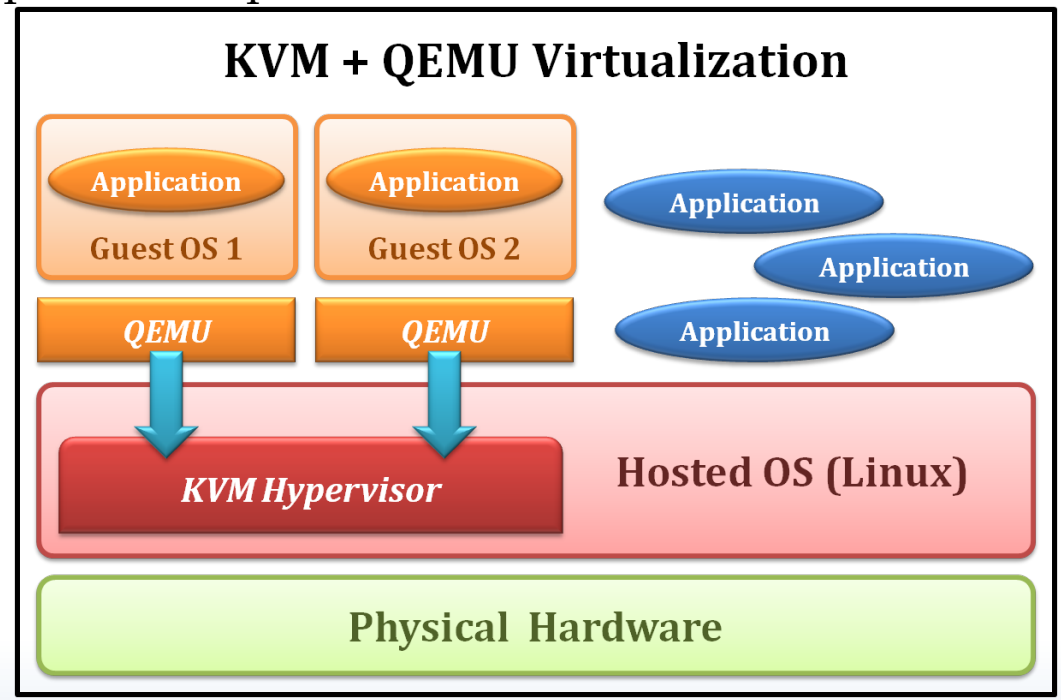
KVM



1. CPU and memory virtualizations are handled in the Linux Kernel Space
2. I/O virtualization is handled in the Linux User Space by QEMU
3. It's a type 2 virtual machine
4. It's a full virtualization implementation

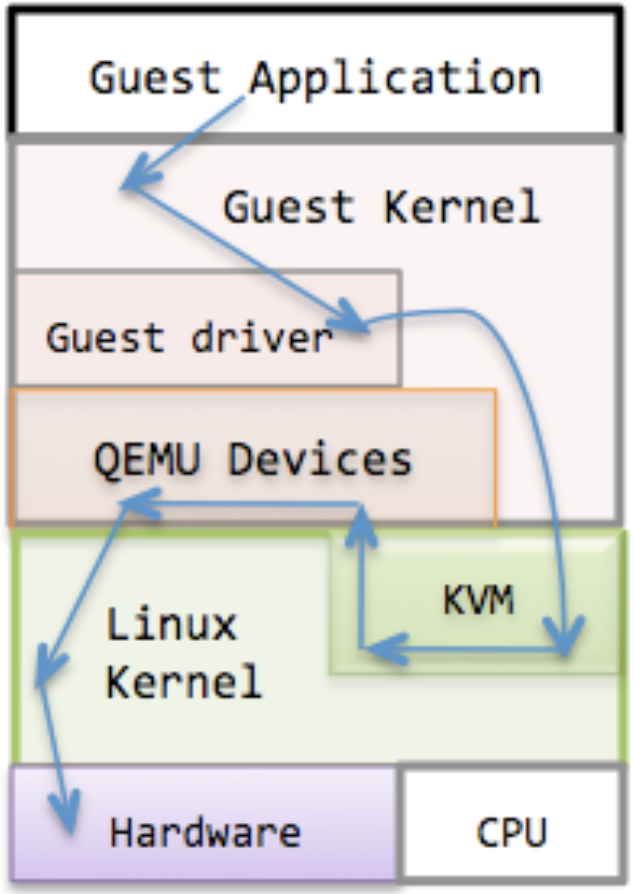
KVM Full Virtualization

- It consists of a loadable kernel module
  - **`kvm.ko`**
    - provides the core virtualization infrastructure
  - **`kvm-intel.ko / kvm-amd.ko`**
    - processor specific modules



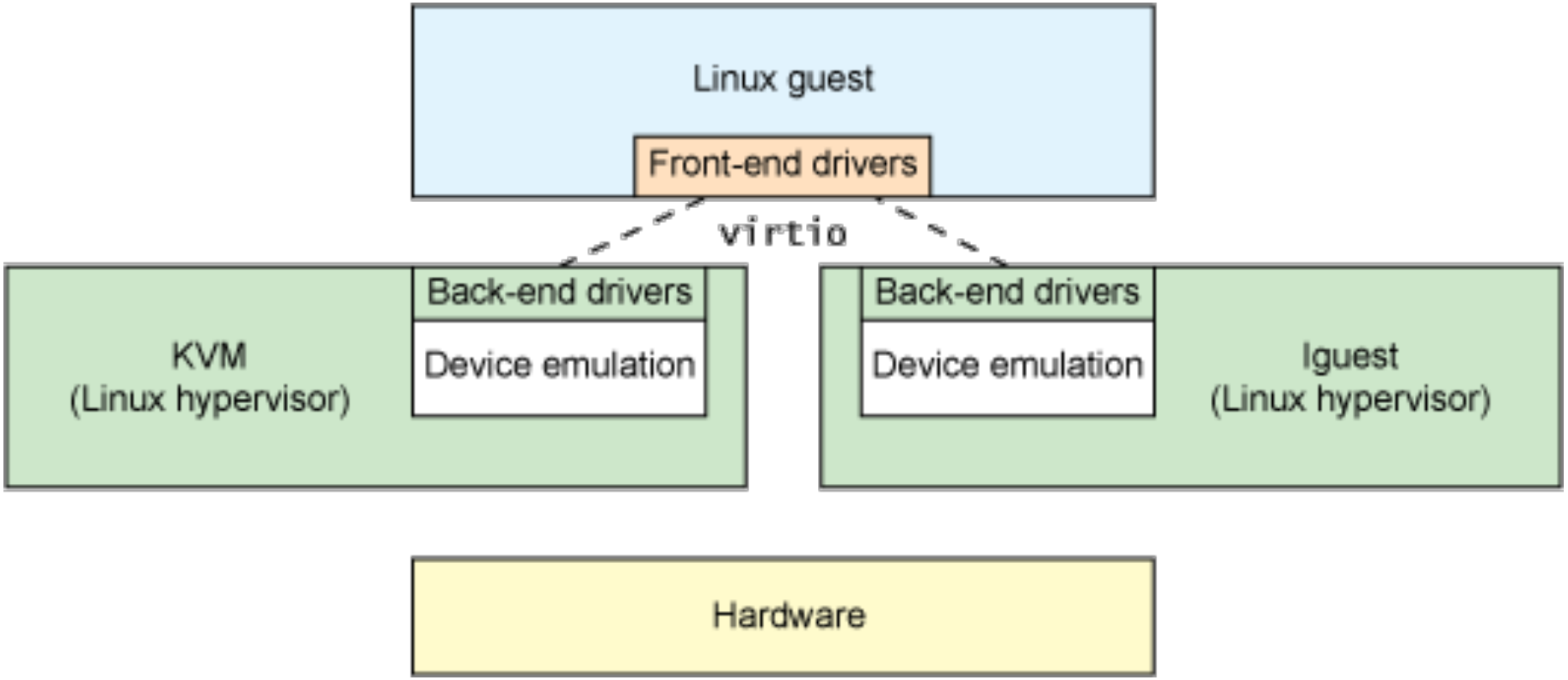
IO Device Model in KVM

- Original approach with full-virtualization
  - Guest hardware accesses are intercepted by KVM
  - QEMU emulates hardware behavior of common devices
    - RTL 8139
    - PIIX4 IDE
    - Cirrus Logic VGA



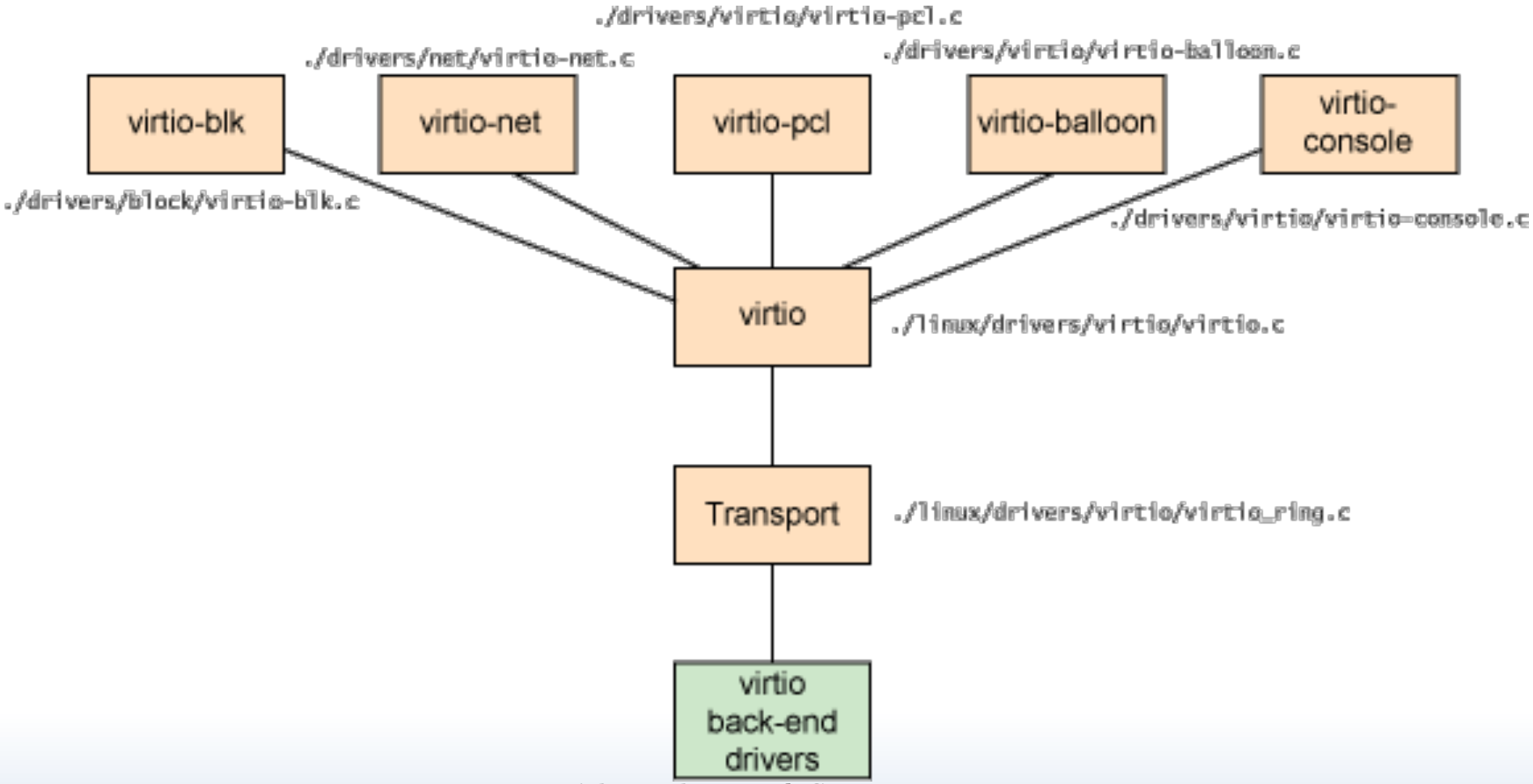
IO Device Model in KVM

- New approach with para-virtualization

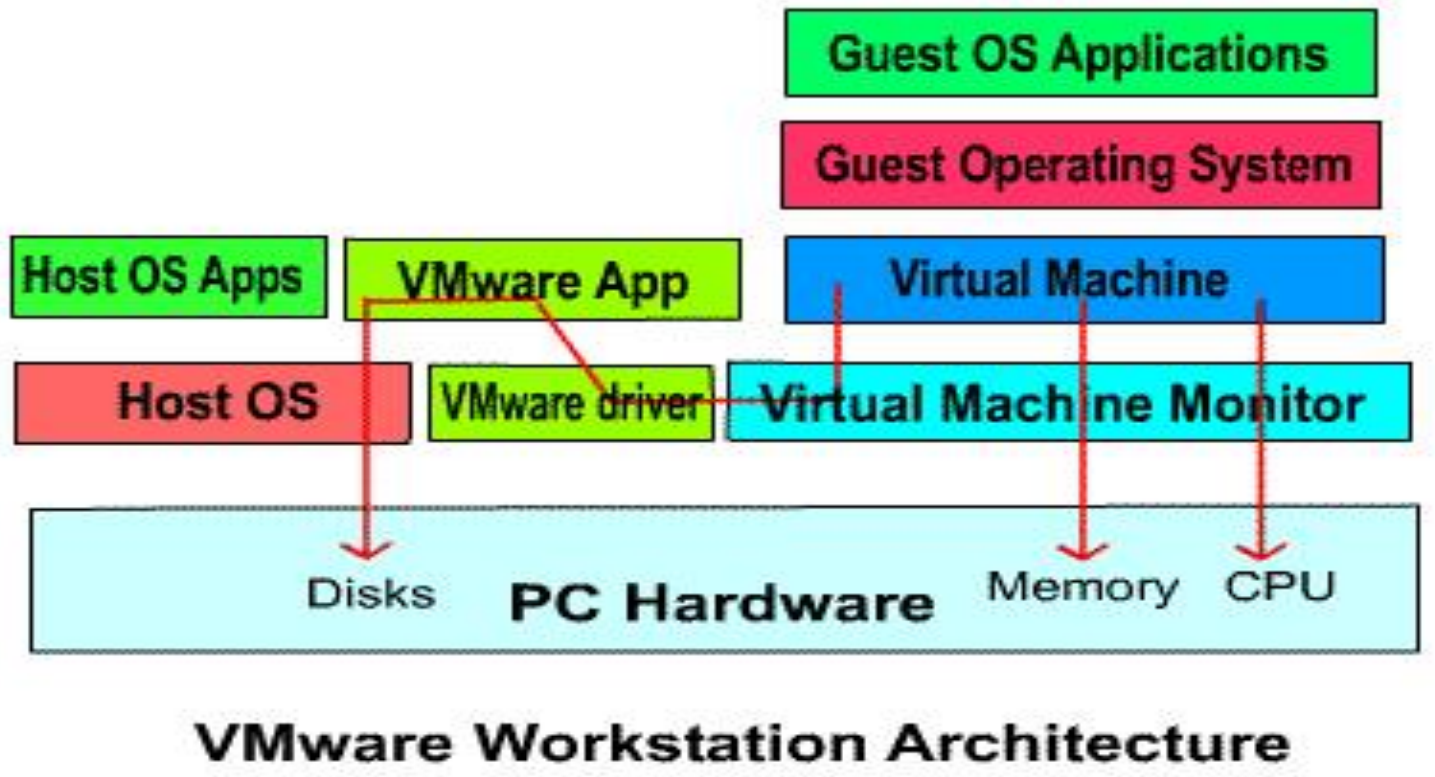


IO Device Model in KVM

- **virtio** architecture

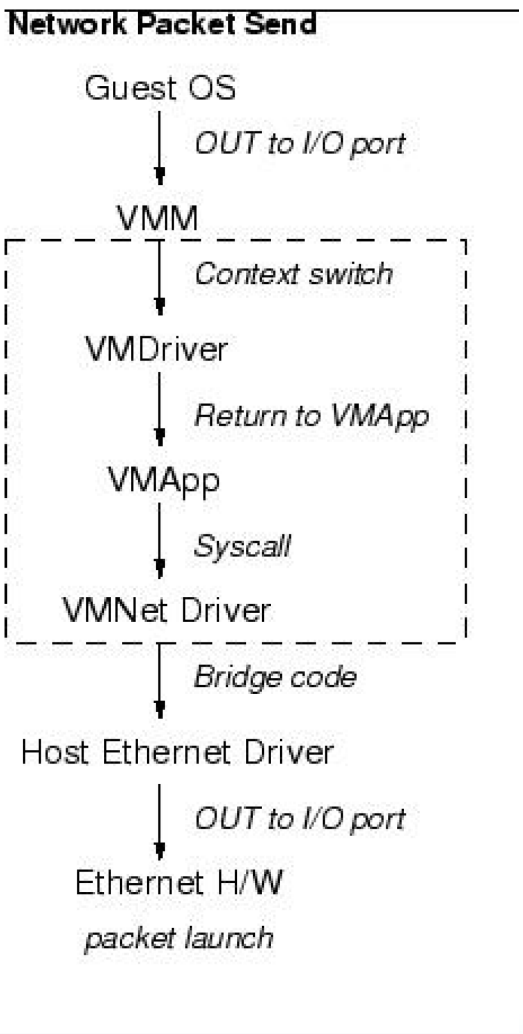


VMware Workstation



VMware Workstation Architecture

VMware: I/O Virtualization



- VMM does not have access to I/O
- I/O in "host world"
  - Low level I/O instructions (issued by guest OS) are merged to high-level I/O system calls
  - VM Application executes I/O SysCalls
- VM Driver works as the communication link between VMM and VM Application

Virtual PC

- Similar to VMWare workstation
- Additional Features
  - Undo disks
  - Binary translation: provide x86 machines on Macintosh-based machines
- Shortcomings
  - Not support Linux, FreeBSD, OpenBSD, Solaris as guest OSes
  - Not support SCSI devices
  - Not support hardware upgrades
  - Not support Linux as host OS

  - "Everything is about Microsoft"

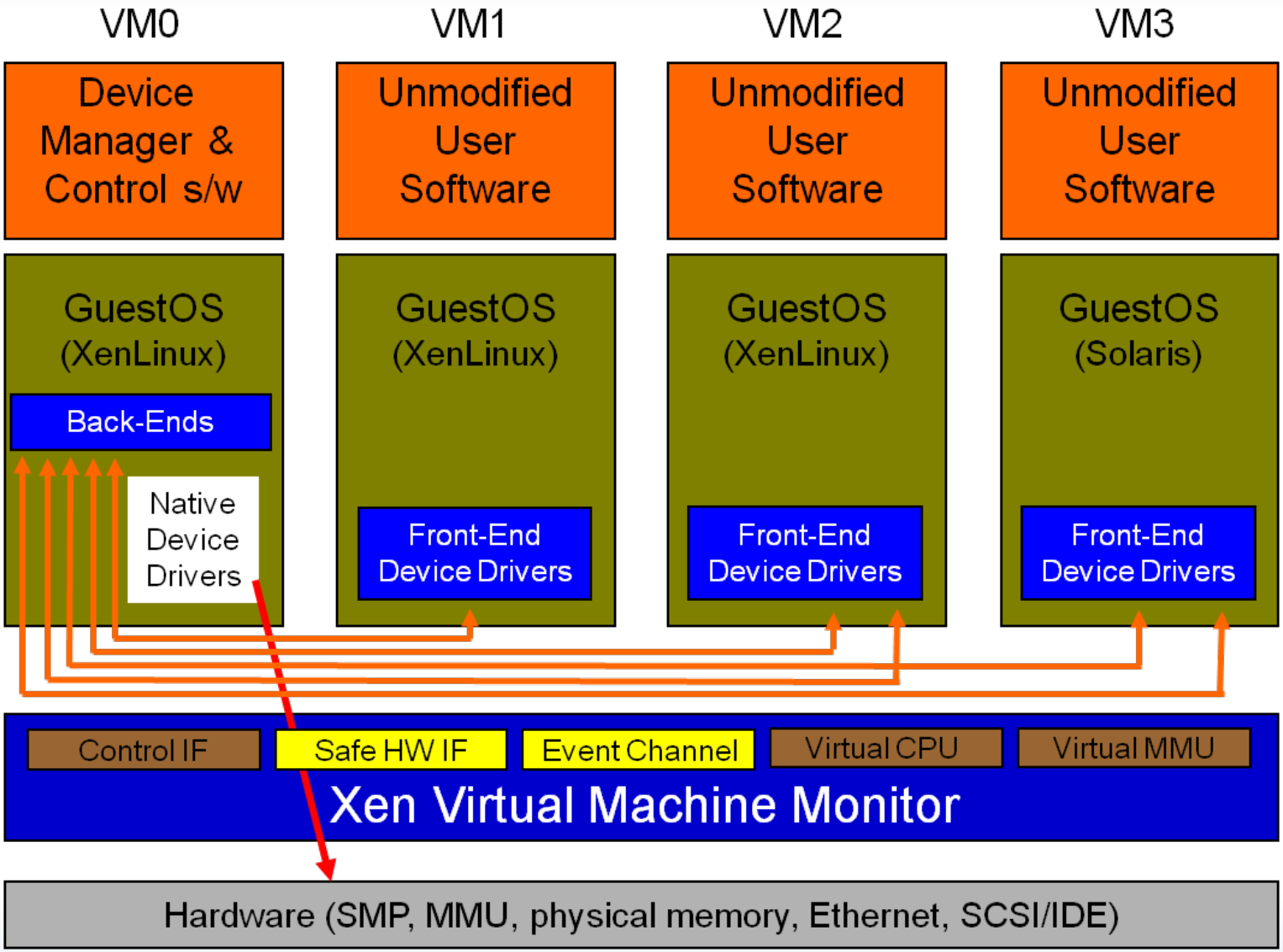
Paravirtualization

- Full: VMWare/Virtual PC
  - Guest OS lives in a complete virtual world with no knowledge about the real machine
  - Support legacy OS
  - Difficult to scale to high numbers
    - Interrupt handling/memory management/world switching
    - Solution?
- Paravirtualization
  - Provides a much simpler architecture interface for the customized guest OSes
    - Virtual I/O and CPU instructions, registers, …
  - Trade portability for performance and scalability
  - Denali/Xen

Xen

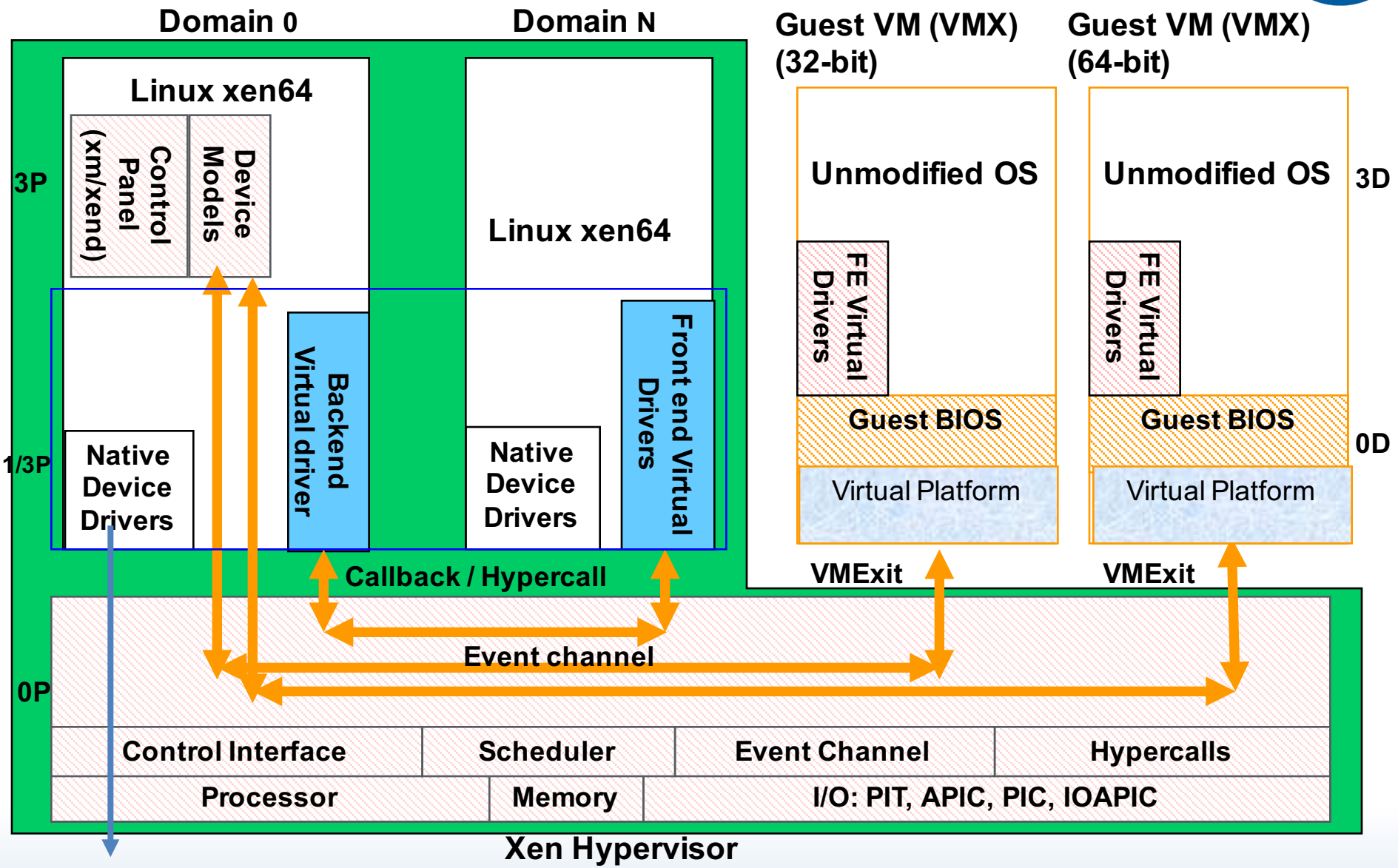
- Exposes some “real” hardware
  - e.g. clock, physical memory address
- Maintain the same application binary interface (ABI)
- The effort of porting OSes is minimal
- Examples:
  - Page table
    - Guest OS have the direct access to hardware page tables, but updates are batched and validated by Xen
  - Timer Interface
    - Guest OS is aware both “real” and “virtual” time

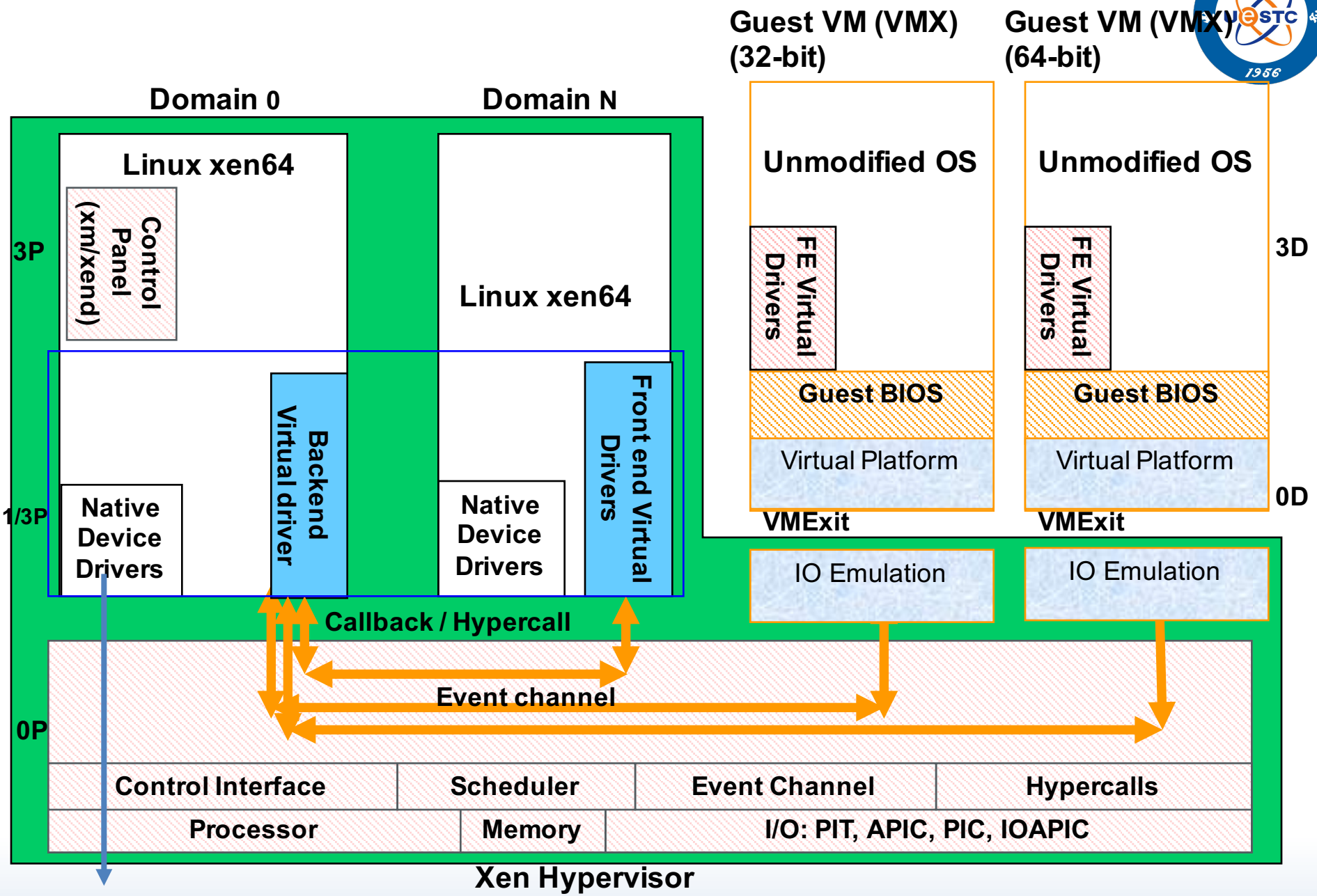
Original Xen Architecture



VT-x / Pacifica: hvm

- Enable Guest OSes to be run without modification
  - E.g. legacy Linux, Windows XP/2003
- CPU provides vmexits for certain privileged instrs
- Shadow page tables used to virtualize MMU
- Xen provides simple platform emulation
  - BIOS, apic, iopaic, rtc, Net (pcnet32), IDE emulation
- Install paravirtualized drivers after booting for high-performance IO
- Possibility for CPU and memory paravirtualization
  - Non-invasive hypervisor hints from OS


Domain 0
Domain N
Guest VM (VMX) (32-bit)
Guest VM (VMX) (64-bit)
Linux xen64
Control Panel (xm/xend)
Device Models
Linux xen64
Unmodified OS
Unmodified OS
3P
3D
FE Virtual Drivers
FE Virtual Drivers
Backend Virtual driver
Front end Virtual Drivers
Guest BIOS
Guest BIOS
0D
1/3P
Native Device Drivers
Native Device Drivers
Virtual Platform
Virtual Platform
Callback / Hypercall
VMExit
VMExit
Event channel
0P
Control Interface
Scheduler
Event Channel
Hypercalls
Processor
Memory
I/O: PIT, APIC, PIC, IOAPIC
Xen Hypervisor


Guest VM (VMX)
(32-bit)
Guest VM (VMX)
(64-bit)
Domain 0
Domain N
Linux xen64
Control Panel (xm/xend)
Linux xen64
Unmodified OS
Unmodified OS
FE Virtual Drivers
FE Virtual Drivers
3P
3D
Guest BIOS
Guest BIOS
Virtual Platform
Virtual Platform
0D
1/3P
Native Device Drivers
Backend Virtual driver
Native Device Drivers
Front end Virtual Drivers
VMExit
VMExit
IO Emulation
IO Emulation
Callback / Hypercall
Event channel
0P
Control Interface
Scheduler
Event Channel
Hypercalls
Processor
Memory
I/O: PIT, APIC, PIC, IOAPIC
Xen Hypervisor

Denali

- Design to support thousands of VM instance running network services
- Hosting a single application, single-user unprotected guest OS
- Not support ABI compatibility
- Not support virtual memory

Others

- User-mode Linux: run Linux on top of Linux
  - Virtual device: port Linux kernel to the Linux system call interface rather than a hardware interface
  - Using ptrace facility to track system calls and trap them into user-space kernel
- Cooperative Linux: run Linux as an unprivileged VM in kernel mode on top of another OS, e.g., Windows
  - Host OS needs to support loading drivers
  - Each kernel has its own complete CPU context and address space, and decide when to give the control back to its partner

OS Level Virtualization

- Containers (operating environments) on top of OS
  - Processes, File System, Network resource (IP address), Environment variables, System call interface
- Technologies
  - chroot(): File system virtualization on Unix; change the root directory of a process and its children
  - Name spaces: Each container is tagged and new entities (fork()) generated from a container remains inside
- Usages:
  - Sandboxing
  - Fine grain access control (root in the container)

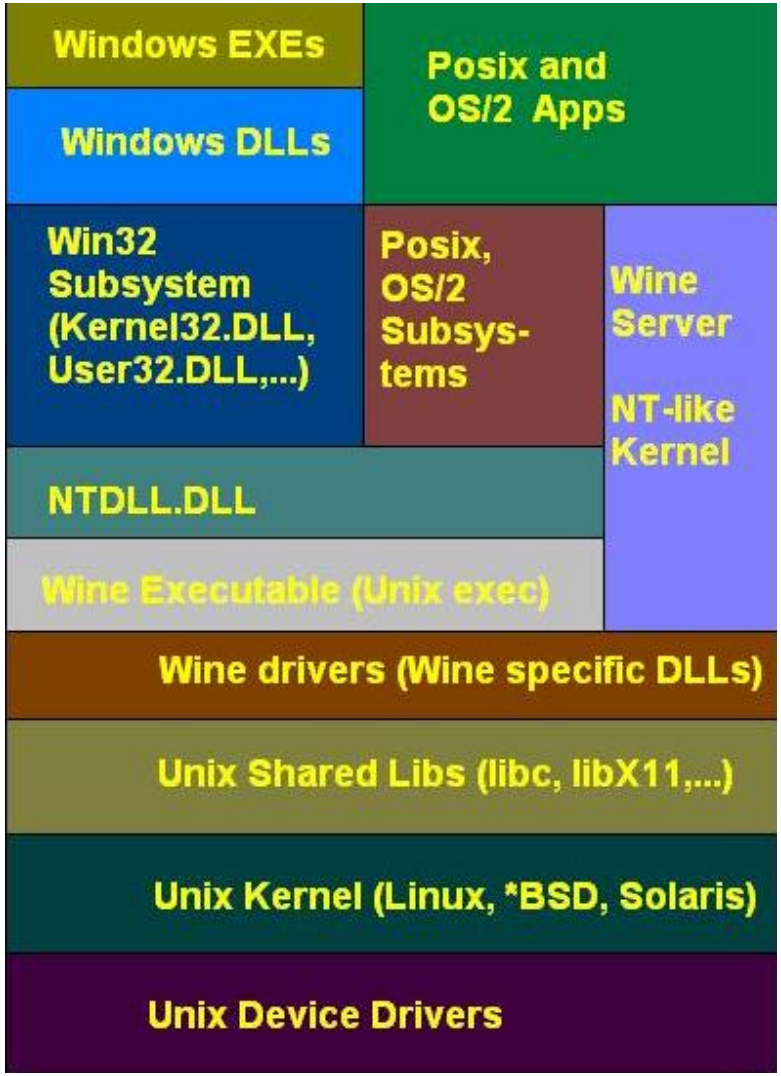
Examples

- Jail
  - FreeBSD based virtualization using “chroot()”
  - Scope is limited to the jail
  - Curtailed access to resources and operations
  - A file-system sub-tree, one IP address, one “root”
- Ensim’s “Virtual Private Server”
- Linux “Virtual Environment” (VE)

Library Level Virtualization

- Goal: ABI/API compatibility
- Technologies
  - API interception through DLL hooking
  - Partial/complete implementation of APIs
  - Emulate low level kernel implementations in user-space
    - Useful when the host OS does not provide required support (e.g. Win32 threads vs. pthreads)
- Examples
  - WINE: Win32 API implementation on Unix/X
  - LxRun: Linux API implementation on SCO UnixWare, Solaris
  - WABI: Sun's implementation similar to WINE

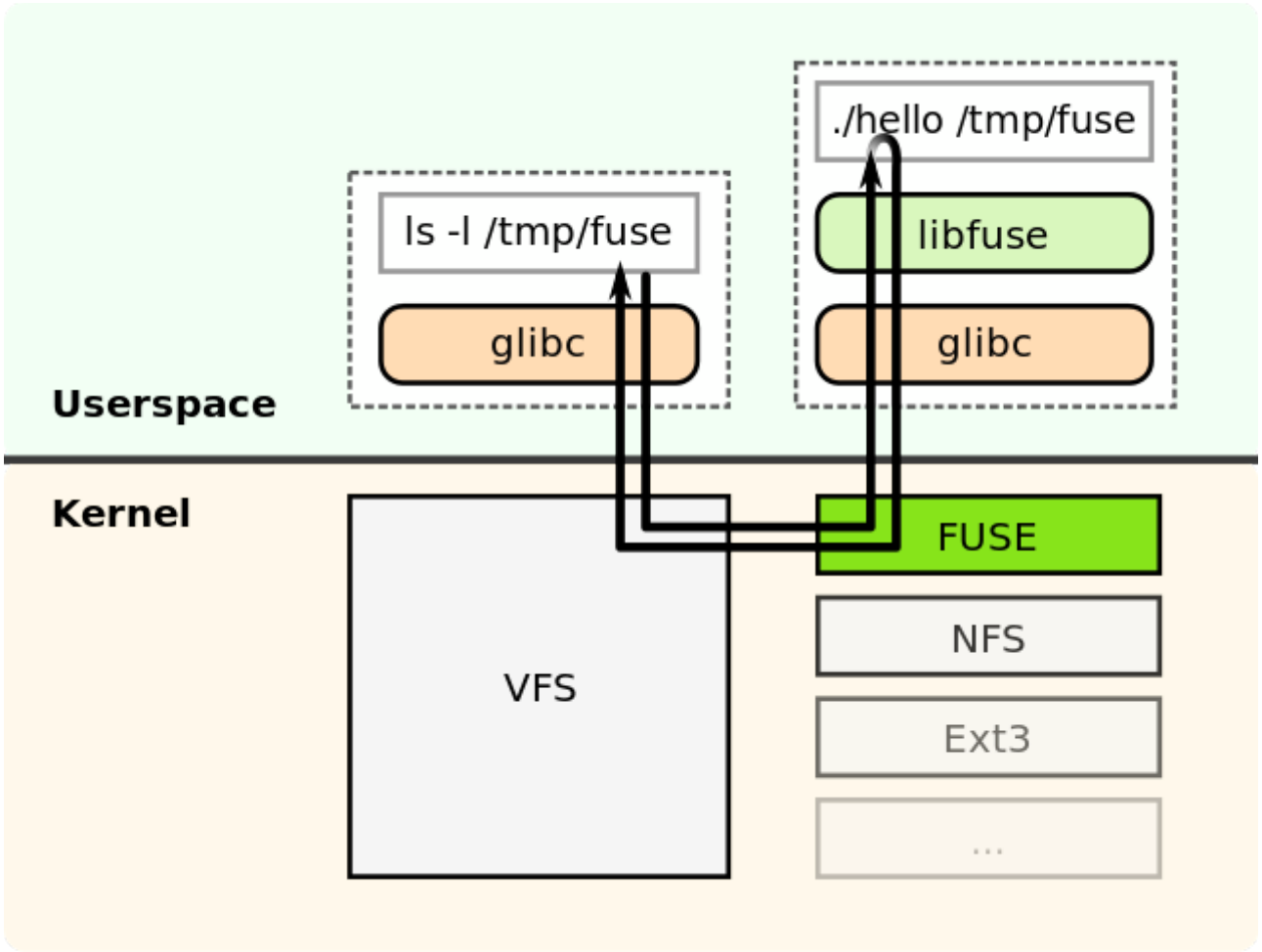
WINE – Wine Is Not an Emulator



- Closely follows NT
  - Re-implements all the "core" DLLs (ntdll, user32, kernel32)
- Wine server provides the NT backbone
  - Message passing
  - Synchronization
  - Object handles
- Native DLL support for non-core libraries
- Hardware access through Unix device drivers

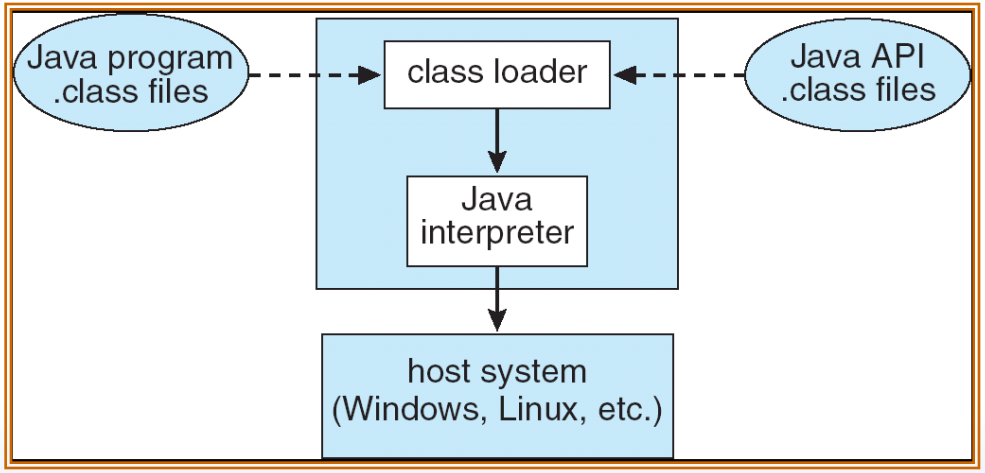
FUSE: Filesystem in Userspace

- SSHFS
- FTPFS
- GmailFS
- GlusterFS
- MooseFS

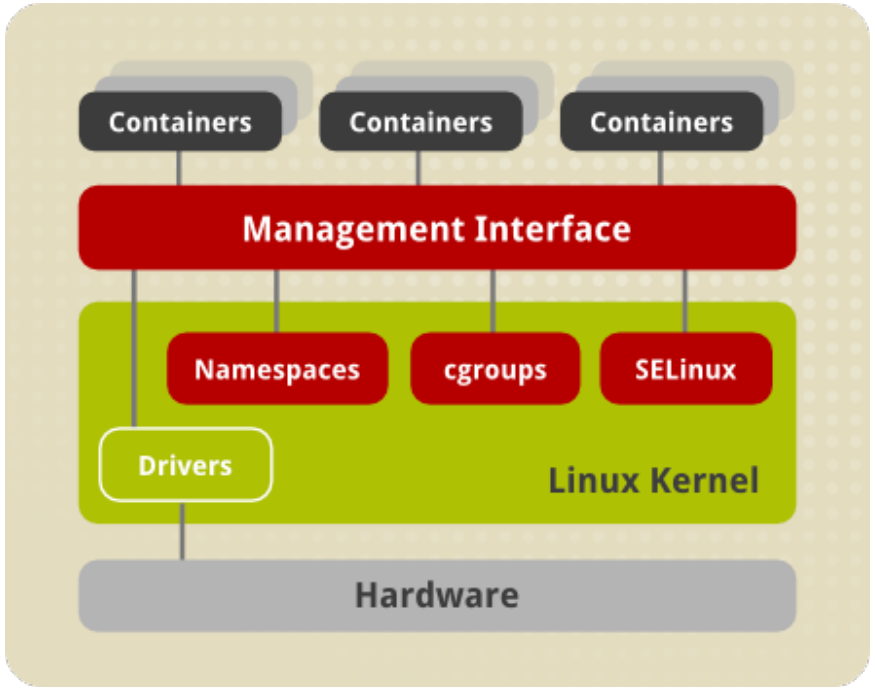


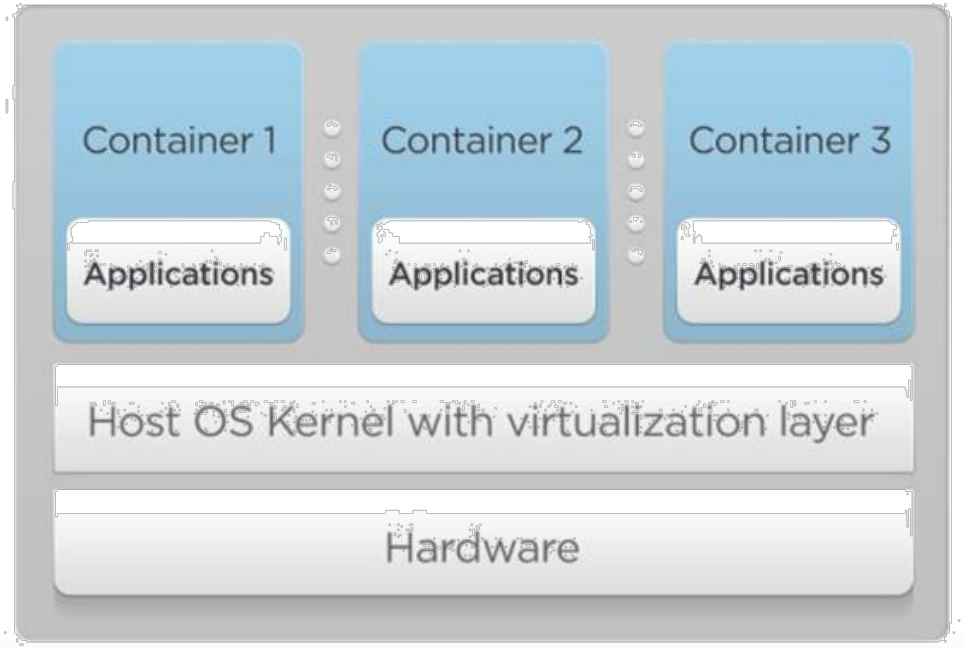
Application Level Virtualization

- Java Virtual Machine (JVM)
  - Executes Java byte code (virtual instructions)
  - Provides code verification, garbage collection
  - Hardware access through underlying OS
  - Virtual hardware: PC, register-set, heap, method (code) areas
- Other Examples: .NET CLI, Parrot (PERL 6)

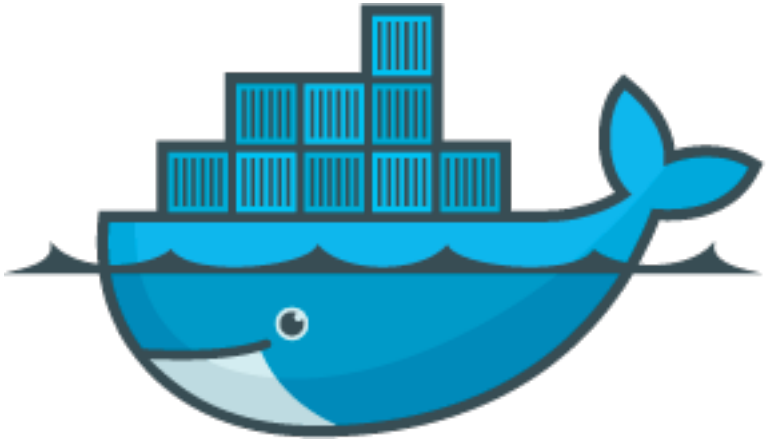


State of the art: Container





Container

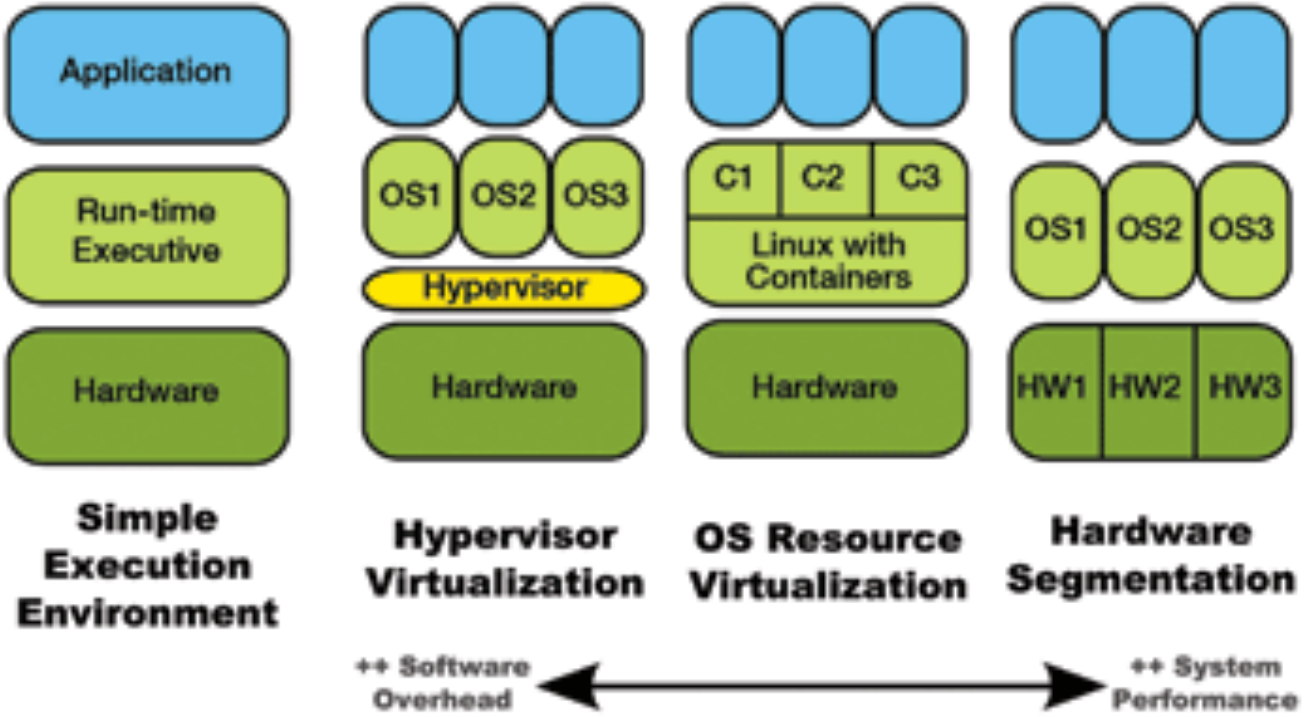


docker

kubernetes

CoreOS

Tradeoff


Application
Run-time Executive
Hardware
Simple Execution Environment
OS1
OS2
OS3
Hypervisor
Hardware
Hypervisor Virtualization
C1
C2
C3
Linux with Containers
Hardware
OS Resource Virtualization
OS1
OS2
OS3
HW1
HW2
HW3
Hardware Segmentation
++ Software Overhead
++ System Performance

Even more

Unikernels are specialised, single-address-space machine images constructed by using library operating systems.

https://www.wikiwand.com/en/Unikernel

DOI:10.1145/2541883.2541895

Article development led by queue.acm.org

What if all the software layers in a virtual appliance were compiled within the same safe, high-level language framework?

BY ANIL MADHAVAPEDDY AND DAVID J. SCOTT

Unikernels: The Rise of the Virtual Library Operating System

CLOUD COMPUTING HAS been pioneering the business of renting computing resources in large data centers to multiple (and possibly competing) tenants. The basic enabling technology for the cloud is operating-system virtualization such as Xen[1] or VMWare, which allows customers to multiplex virtual machines (VMs) on a shared cluster of physical machines. Each VM presents as a self-contained computer, booting a standard operating-system kernel and running unmodified applications just as if it were executing on a physical machine.

A key driver to the growth of cloud computing in the early days was *server consolidation*. Existing applications were often installed on physical hosts that were individually underutilized, and virtualization made it feasible to pack them onto fewer hosts without requiring any modifications or code recompilation. VMs are also managed via software APIs rather than physical actions. They can be centrally backed up and migrated across different physical hosts without interrupting service. Today commercial providers such as Amazon and Rackspace maintain vast data centers that host millions of VMs. These cloud providers relieve their customers of the burden of managing data centers and achieve economies of scale, thereby lowering costs.

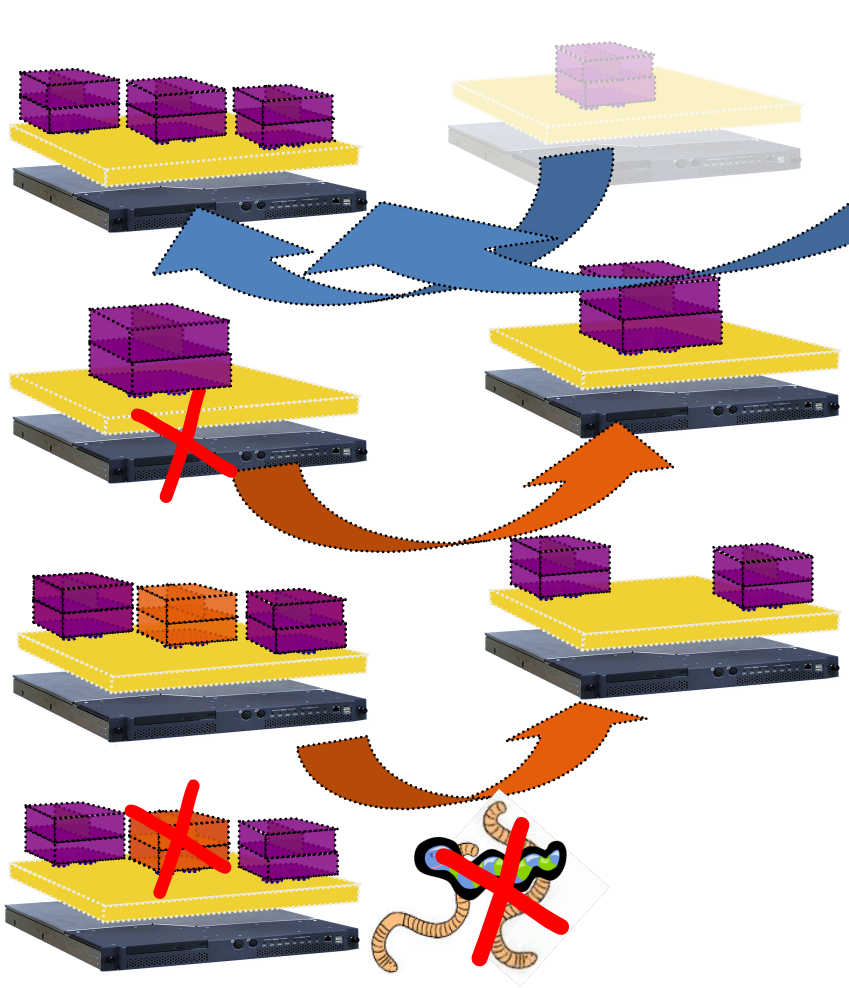
While operating-system virtualization is undeniably useful, it adds yet another layer to an already highly layered software stack now including: support for old physical protocols (for example, disk standards developed

Overall Picture

	ISA	HAL	OS	Library	PL
Performance					
Flexibility					
Ease of Impl					
Degree of Isolation					

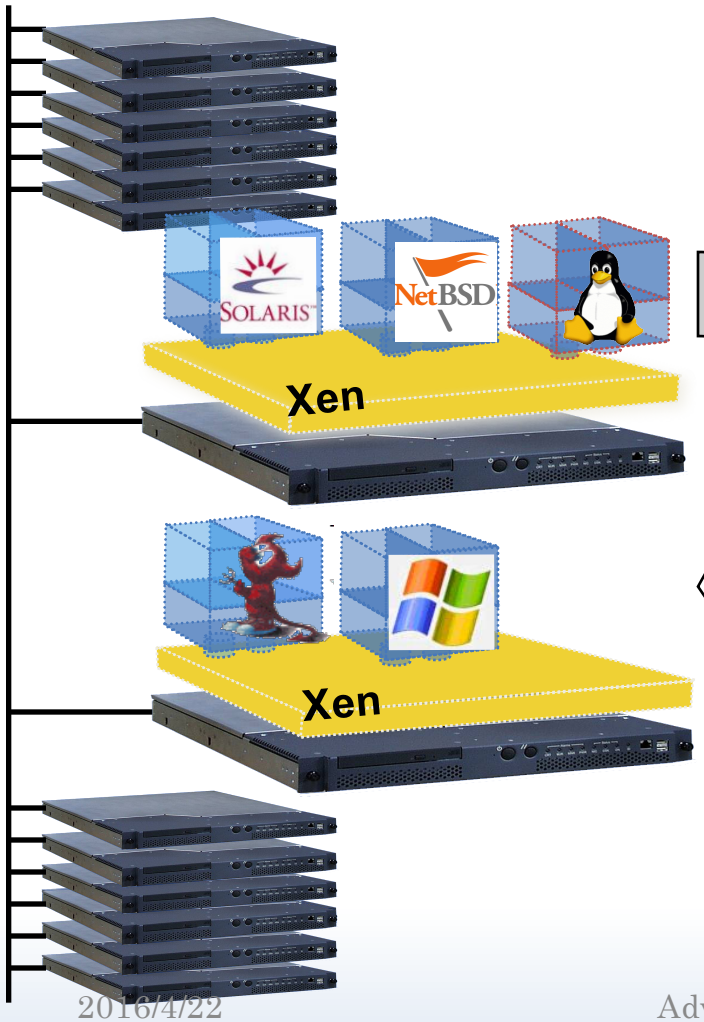
(more stars are better)

Virtualization in the Enterprise



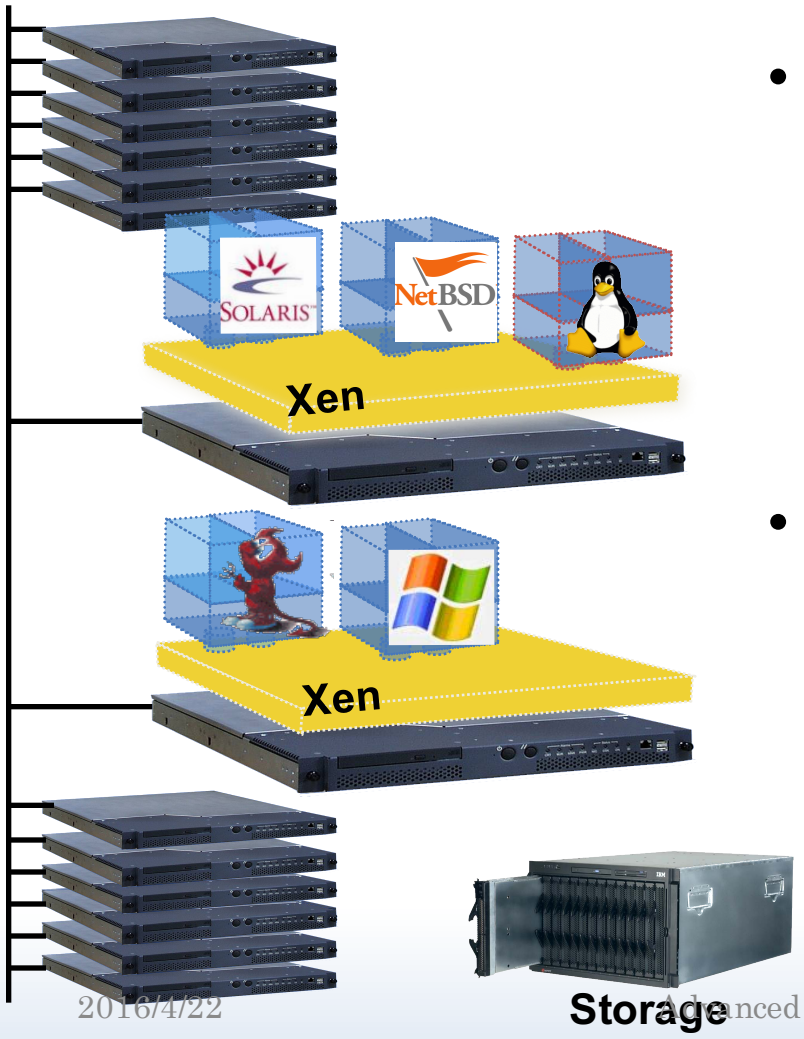
- Consolidate under-utilized servers
- Avoid downtime with VM Relocation
- Dynamically re-balance workload to guarantee application SLAs
- Enforce security policy

VM Relocation : Motivation



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

Assumptions



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

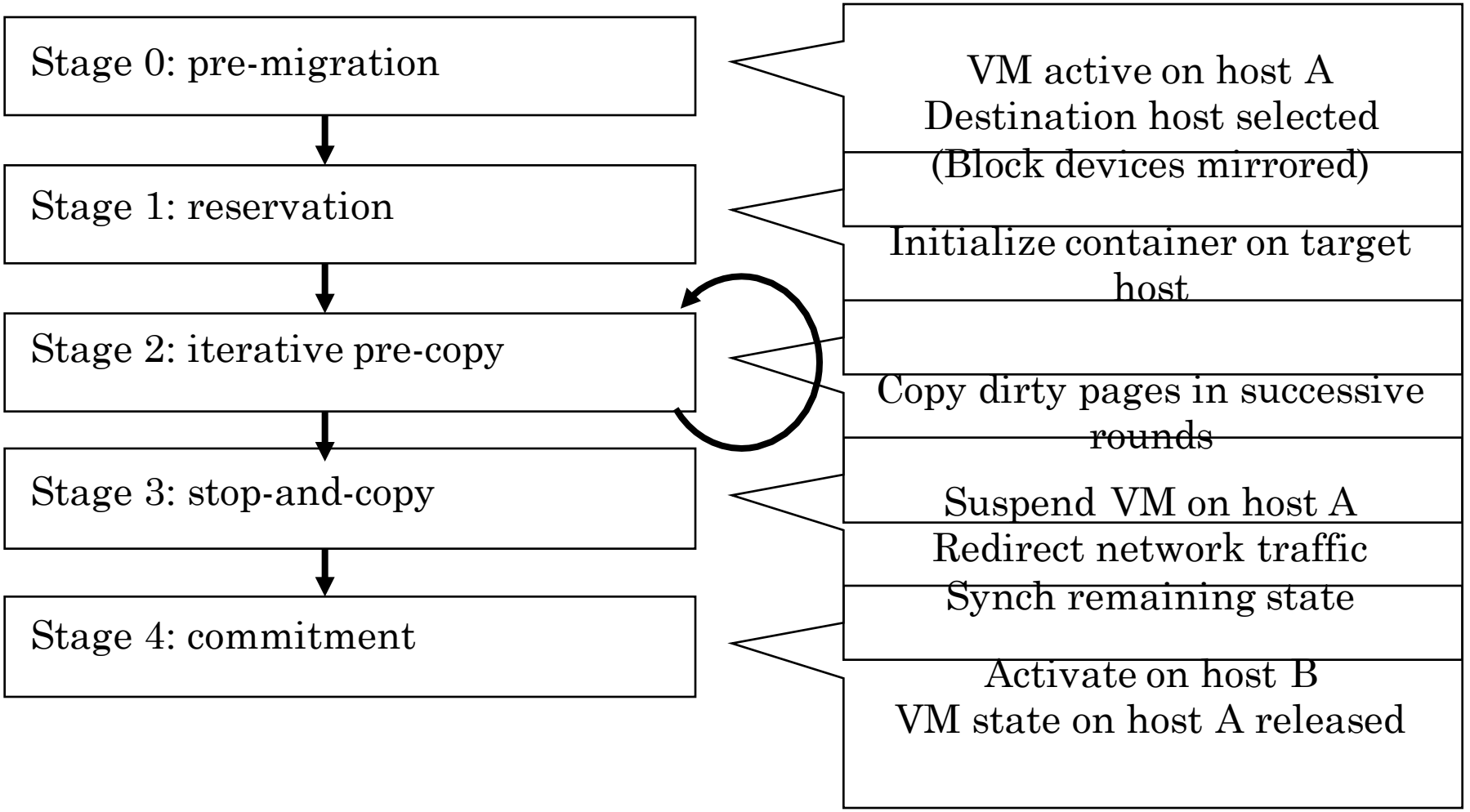
Challenges

- VMs have lots of state in memory
- Some VMs have soft real-time requirements
  - E.g. web servers, databases, game servers
  - May be members of a cluster quorum

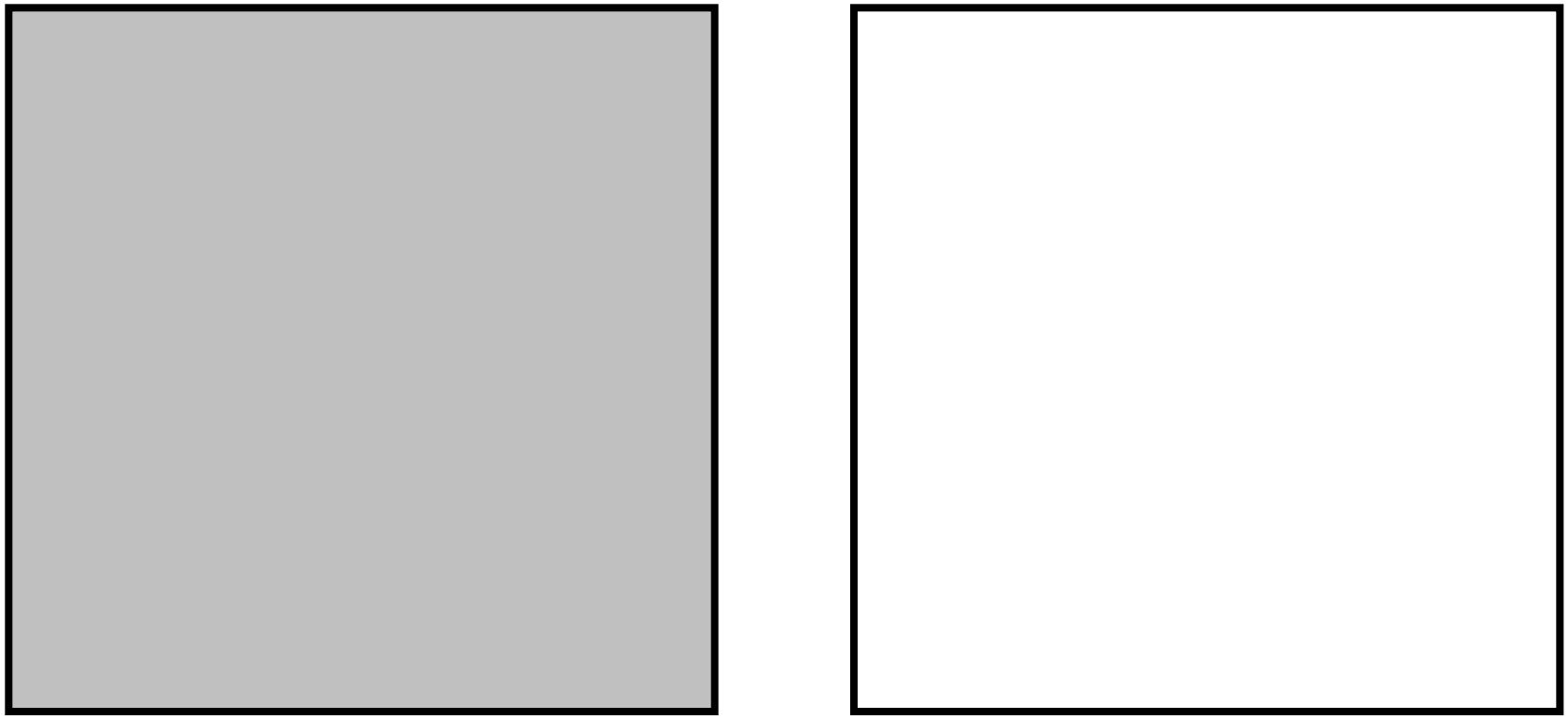
  ➔ **Minimize down-time**
- Performing relocation requires resources

  ➔ **Bound and control resources used**

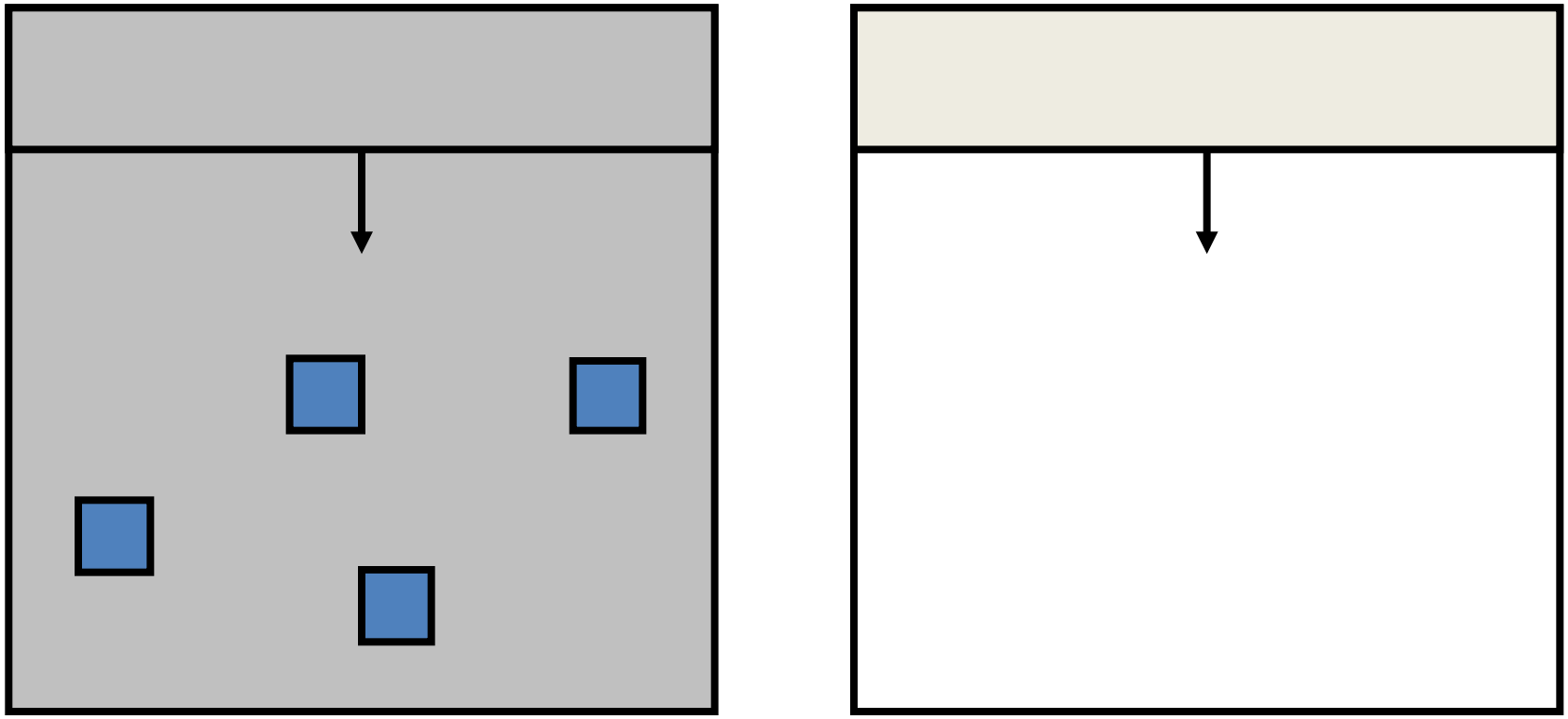
Relocation Strategy



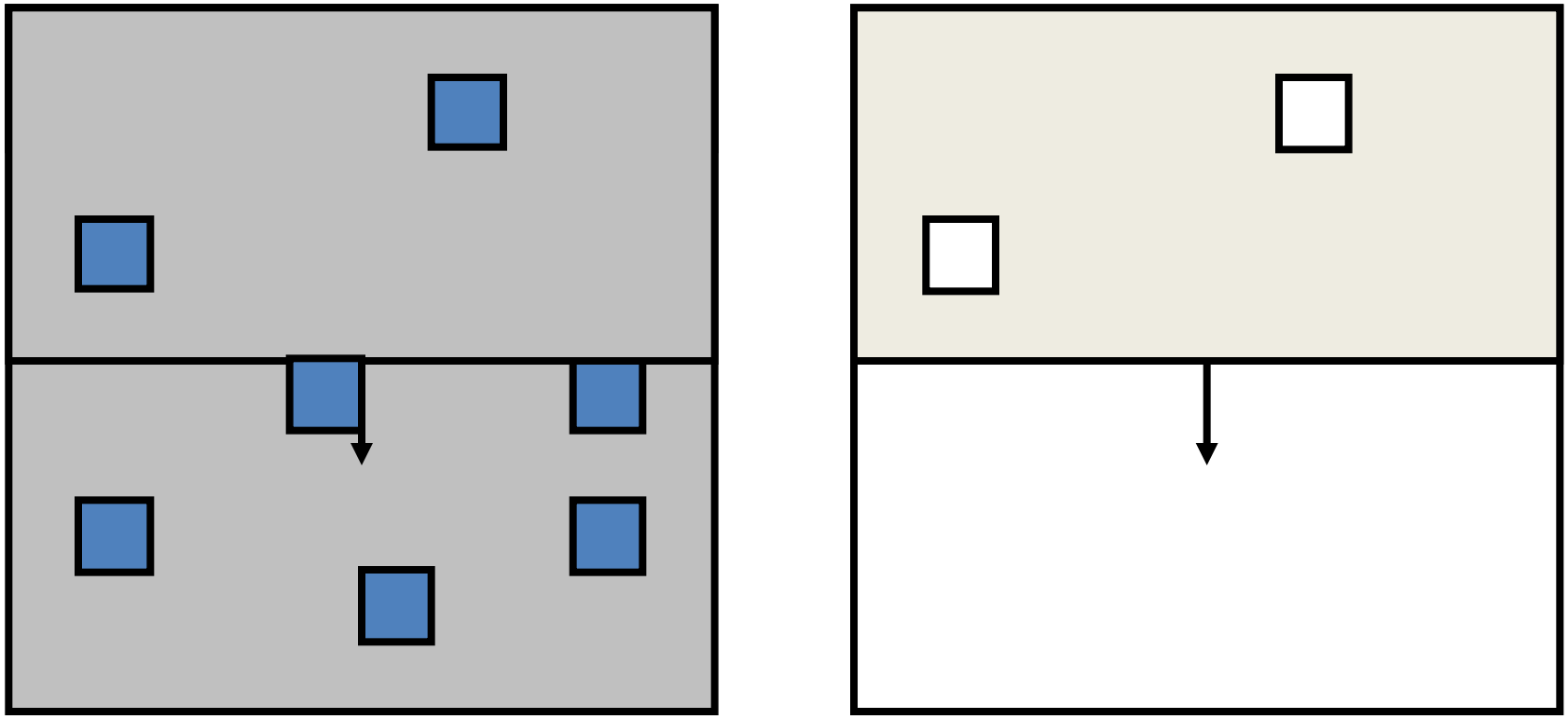
Pre-Copy Migration: Round 1



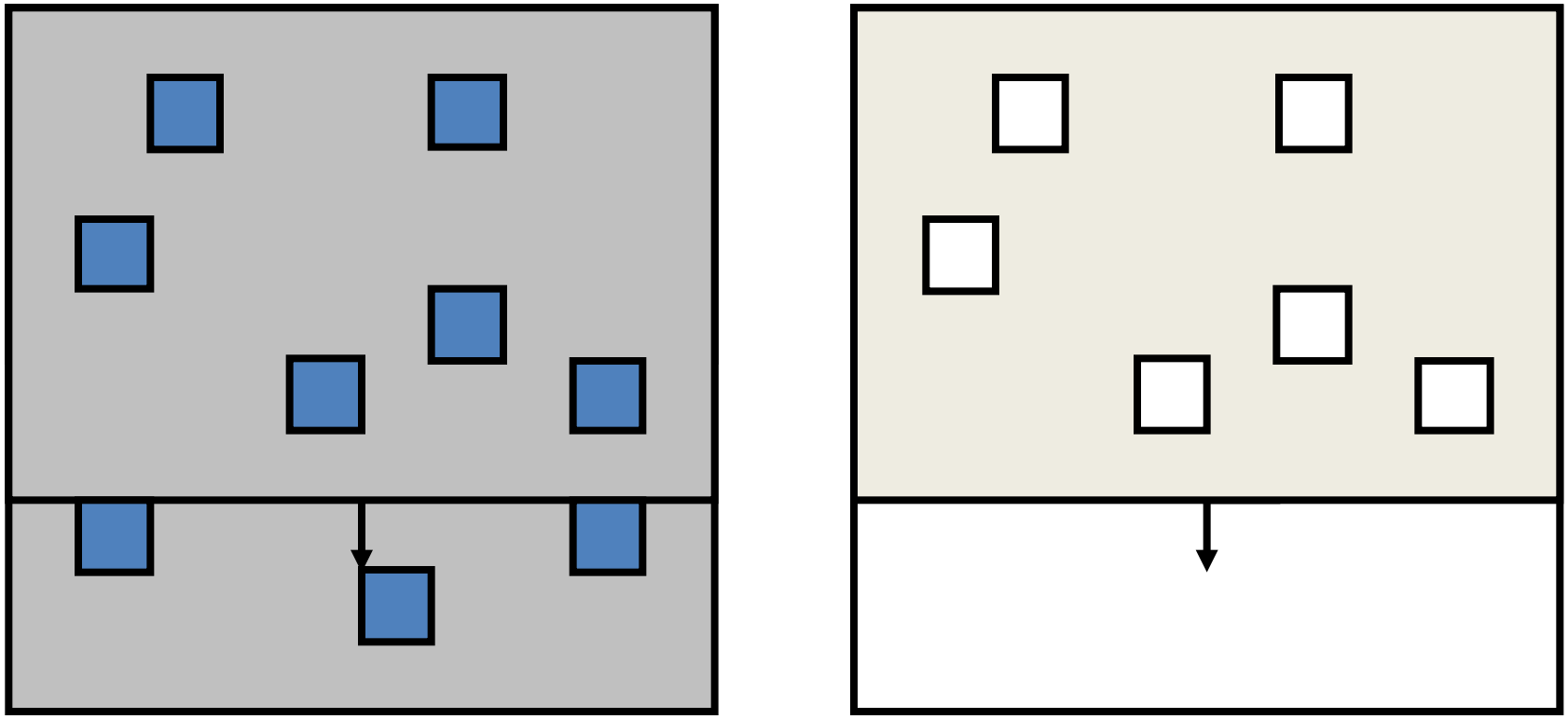
Pre-Copy Migration: Round 1



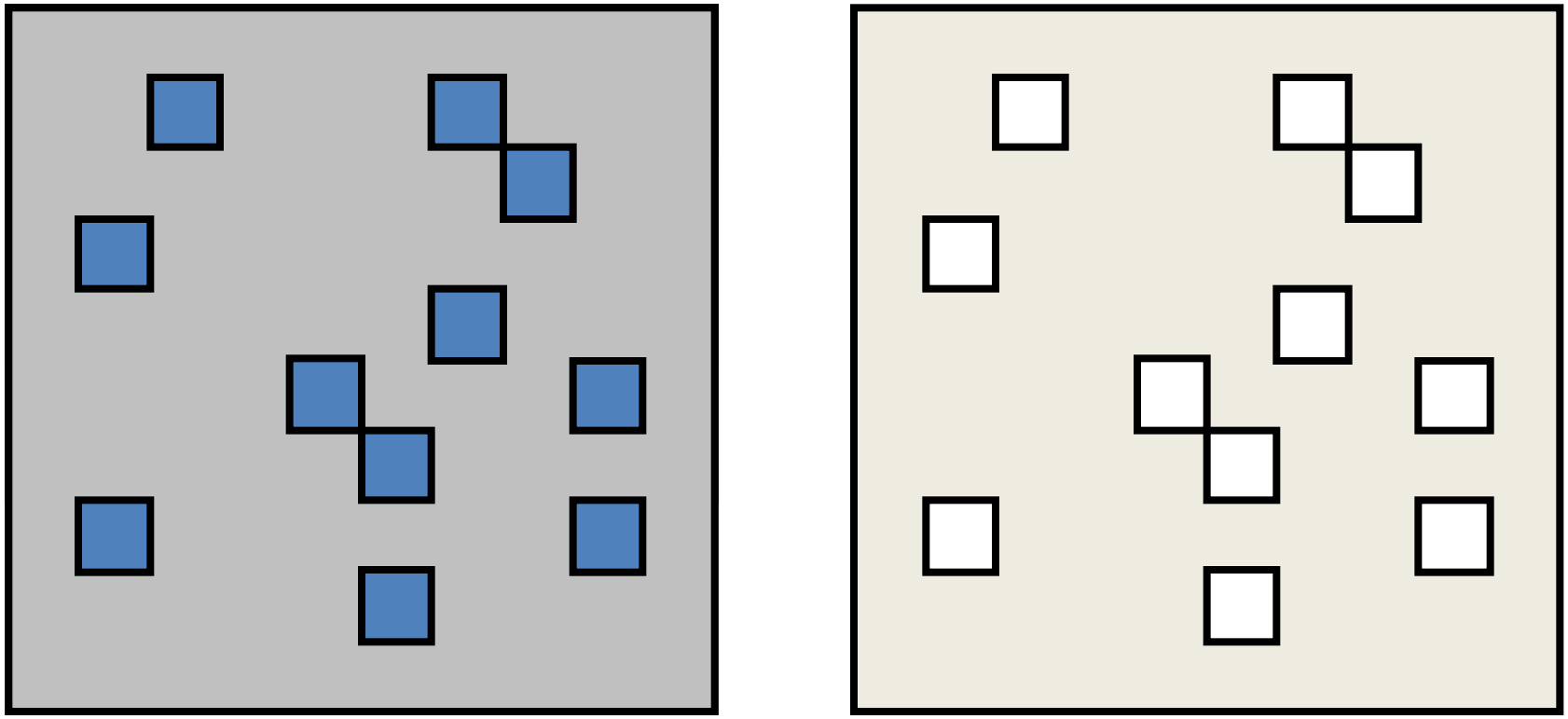
Pre-Copy Migration: Round 1



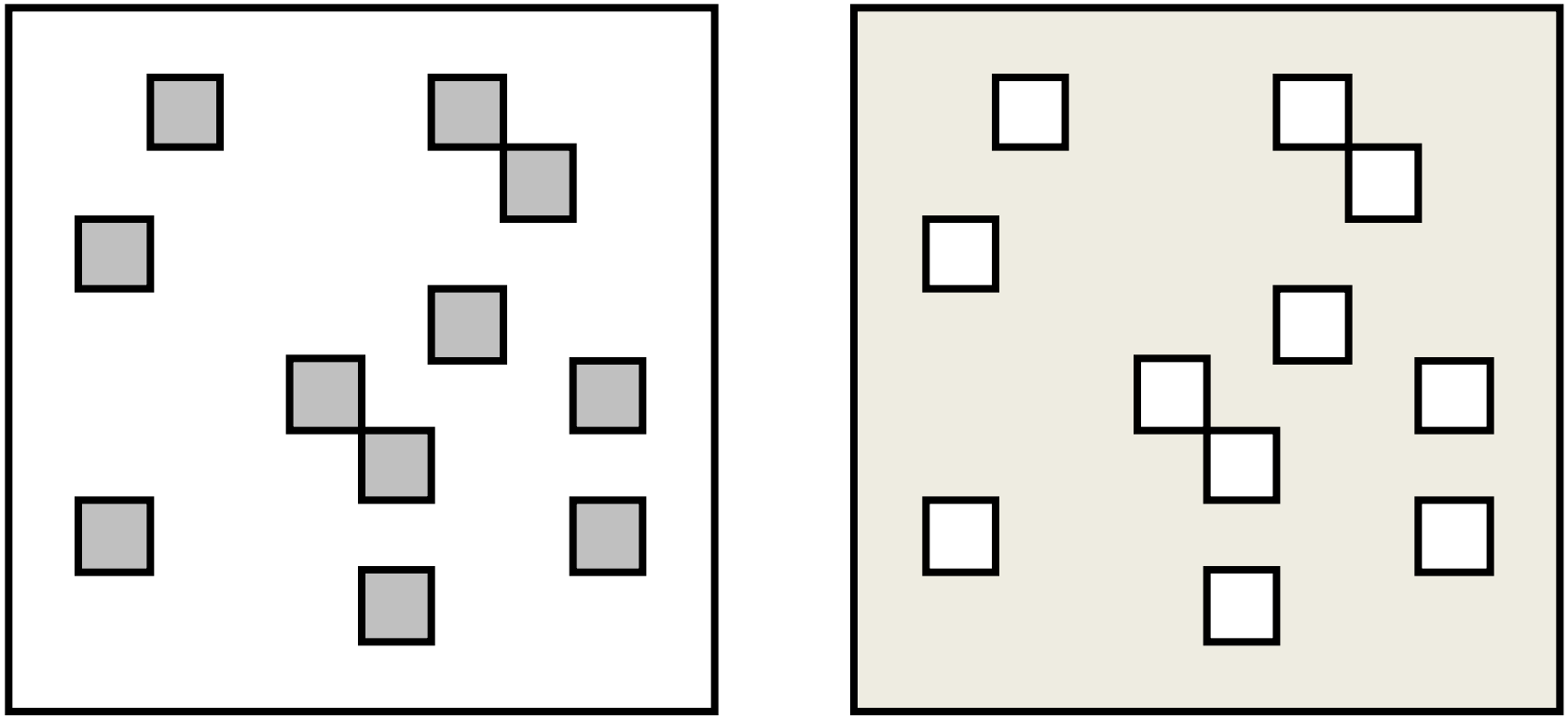
Pre-Copy Migration: Round 1



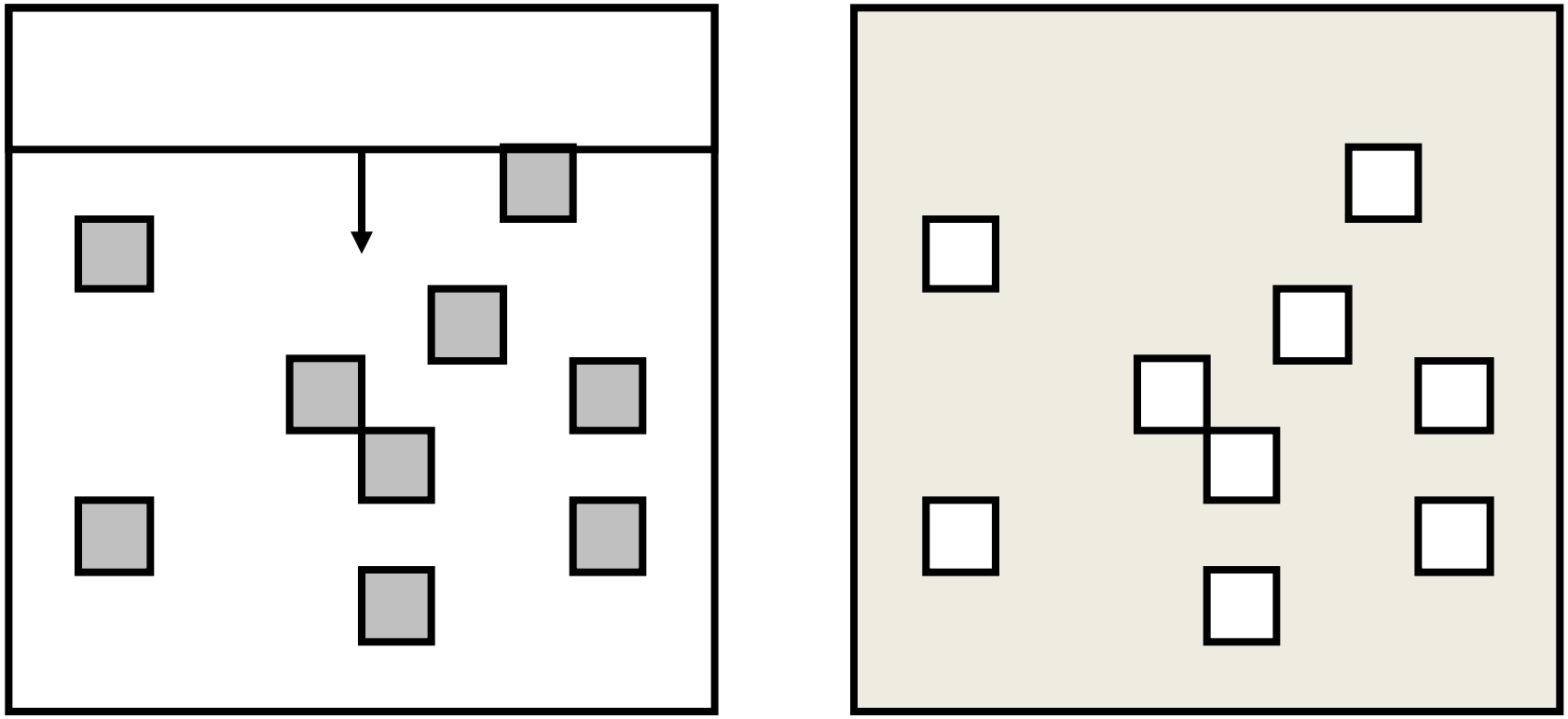
Pre-Copy Migration: Round 1



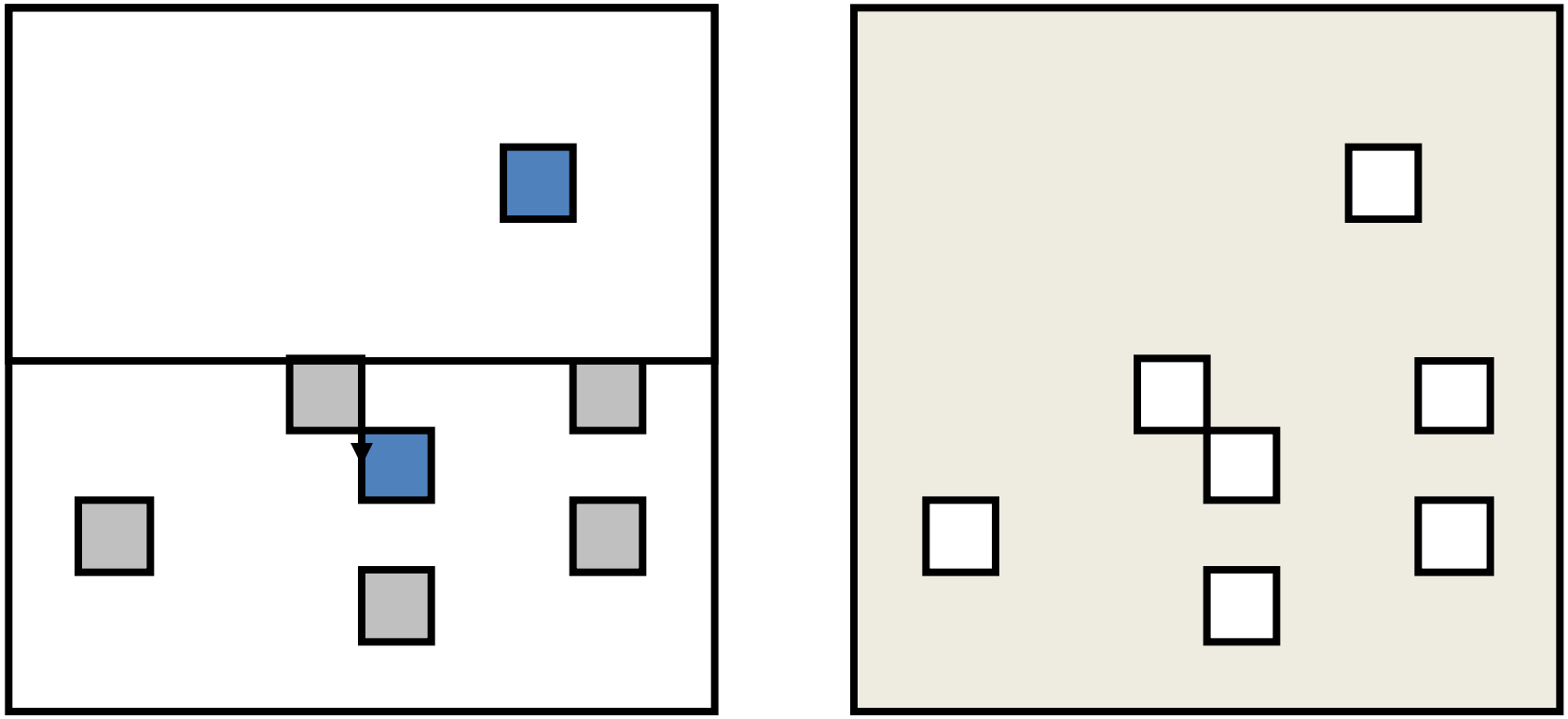
Pre-Copy Migration: Round 2



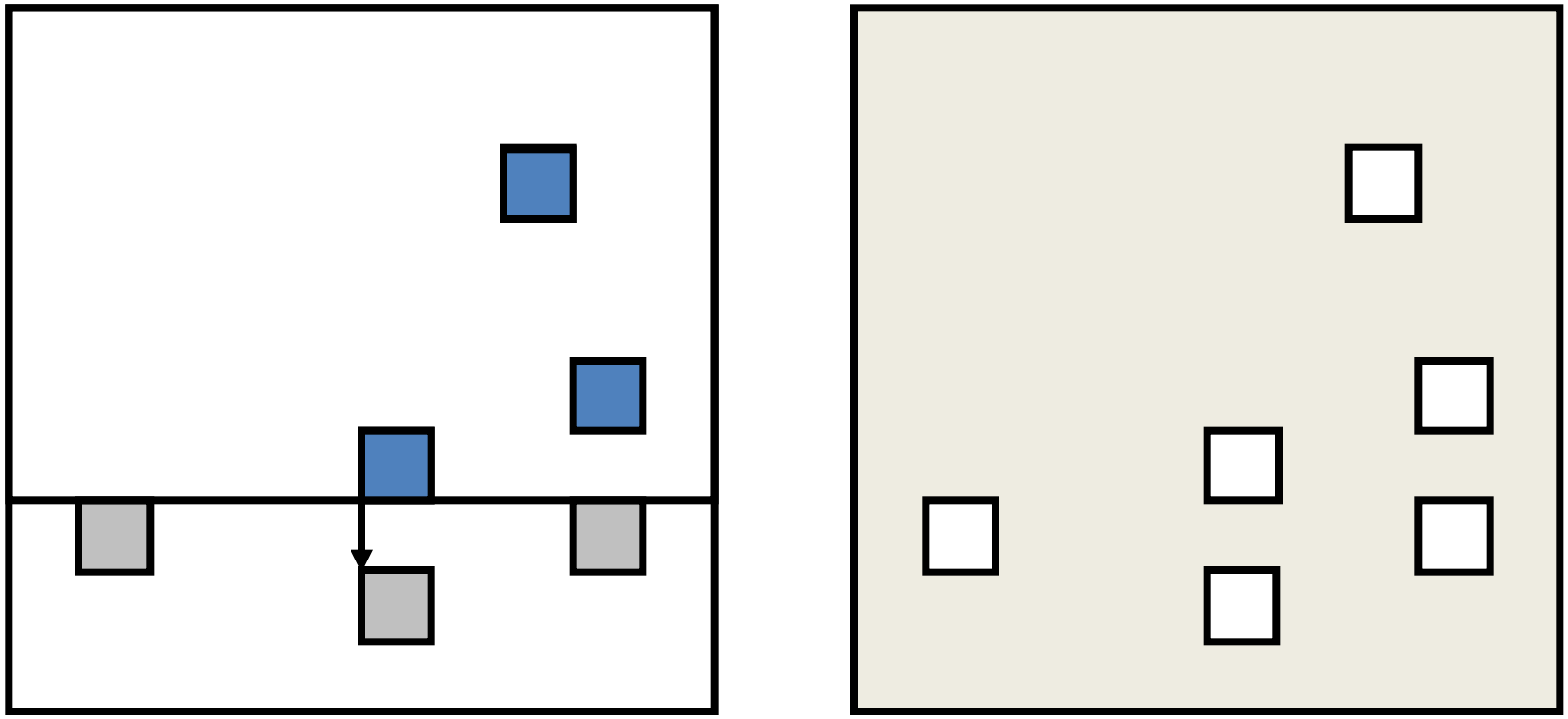
Pre-Copy Migration: Round 2



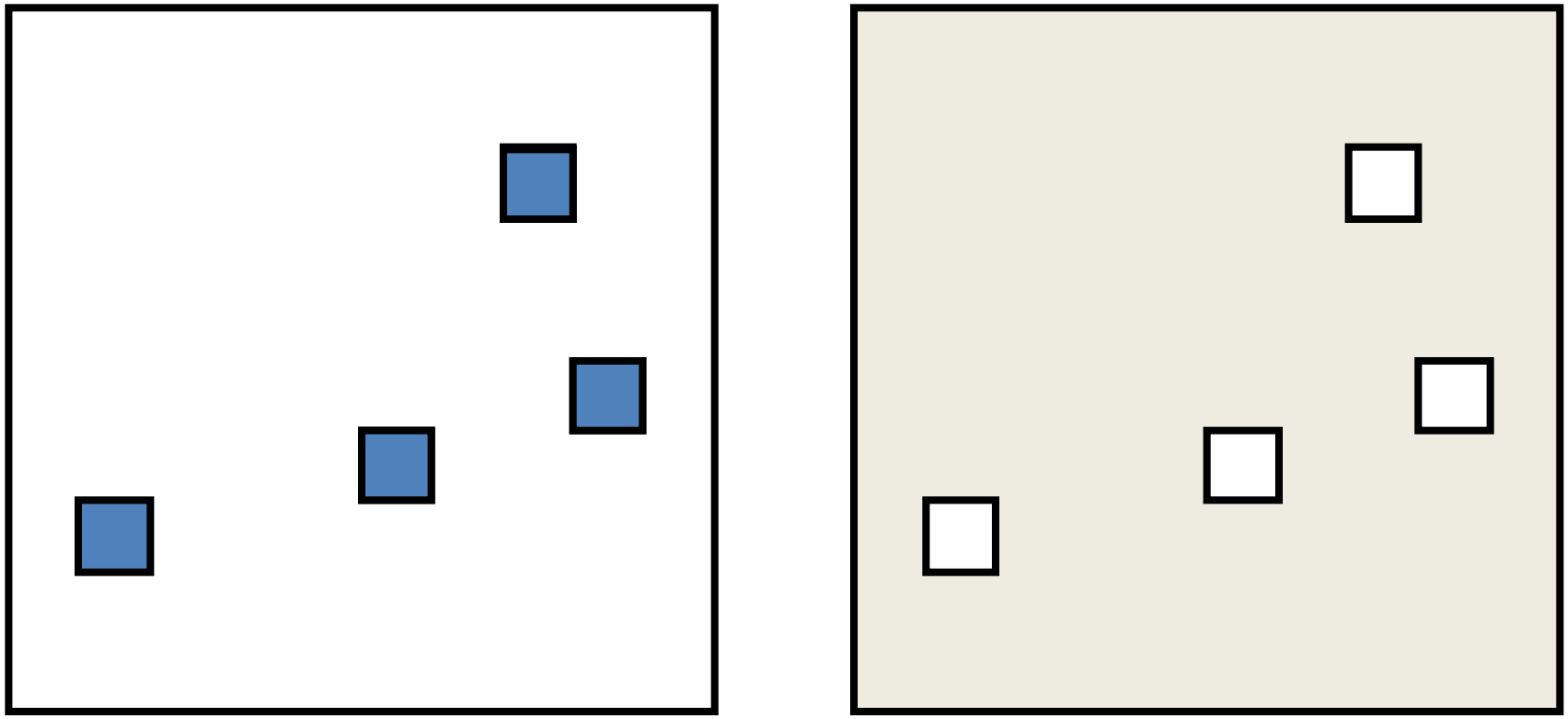
Pre-Copy Migration: Round 2



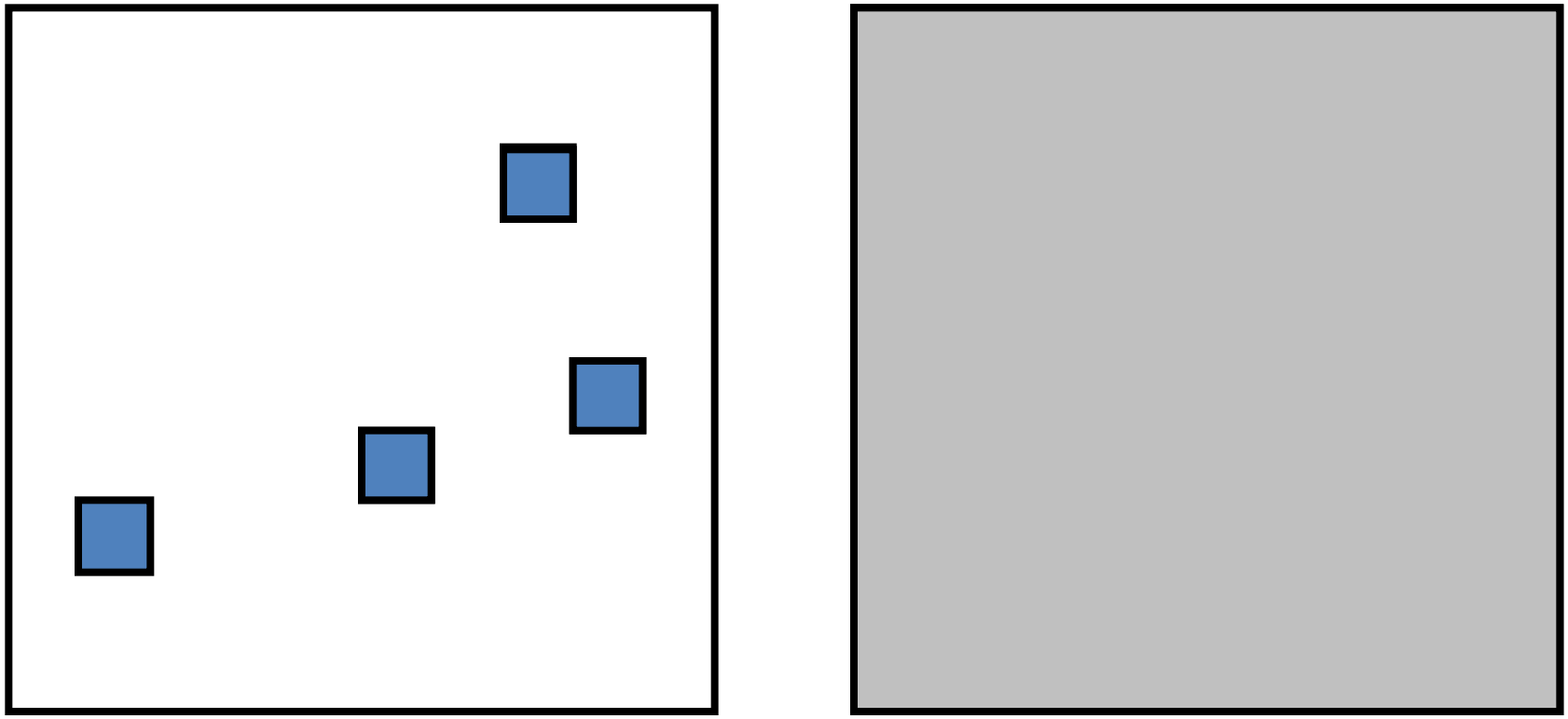
Pre-Copy Migration: Round 2



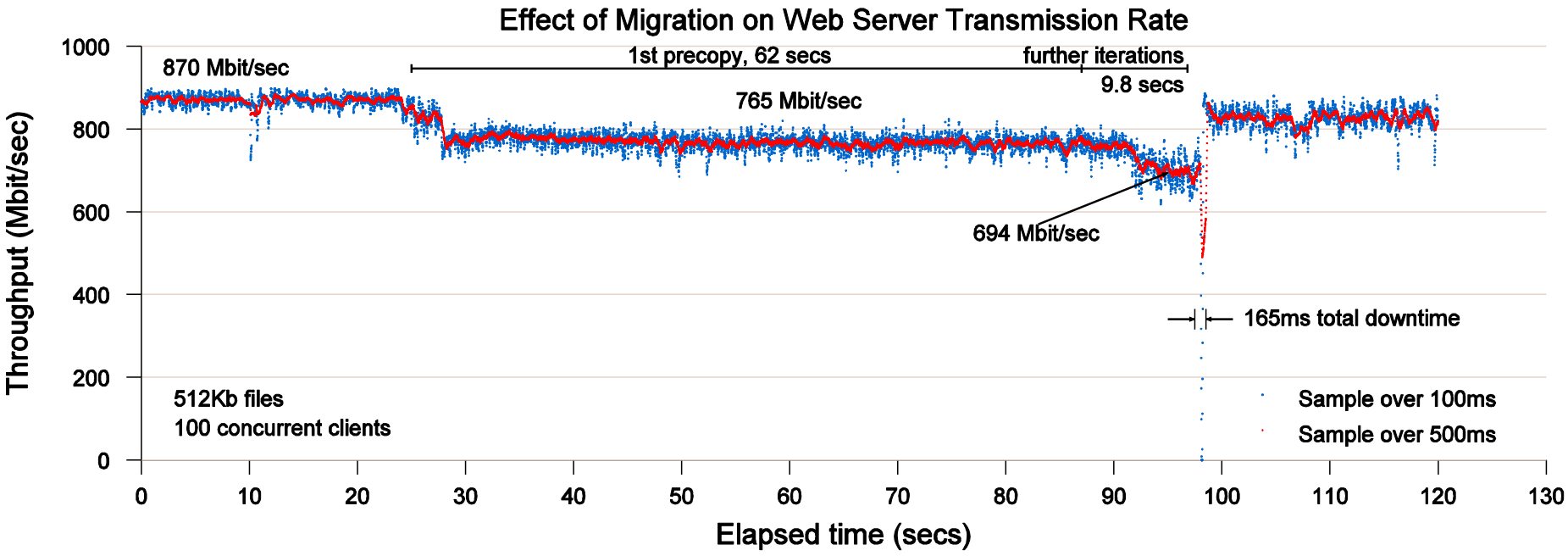
Pre-Copy Migration: Round 2



Pre-Copy Migration: Final

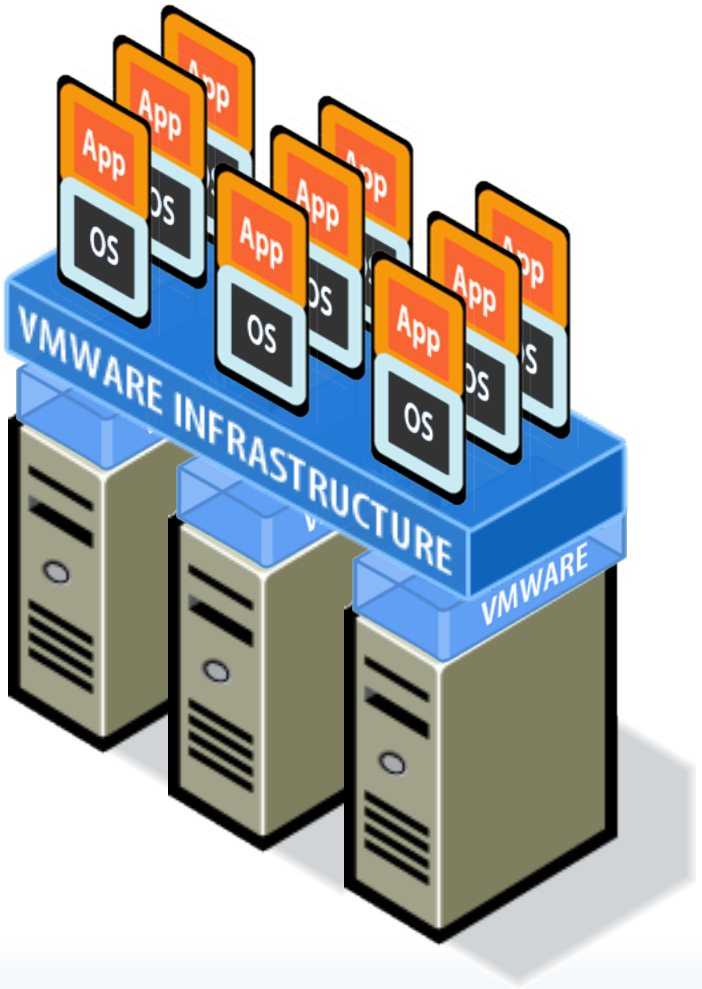


Web Server Relocation



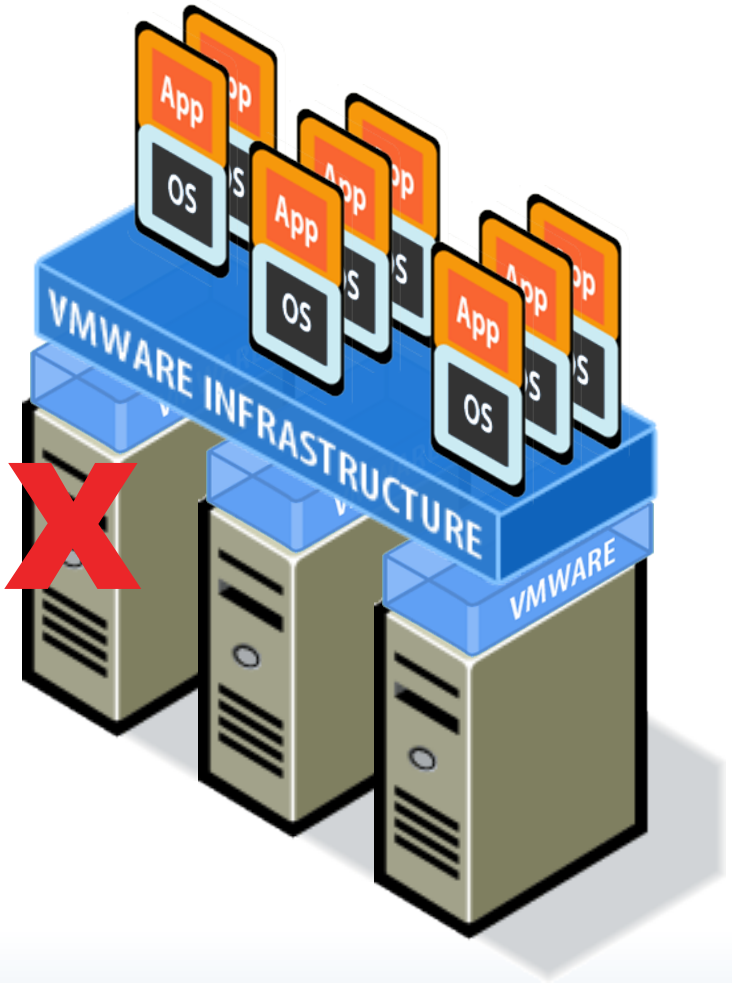
Scalability in Virtualization

- Scalability implement by VMware:
  - VMware VMotion, makes it possible to move Virtual Machines, without interrupting the applications running inside.
  - Dynamically scale up virtual machine system among physical servers.



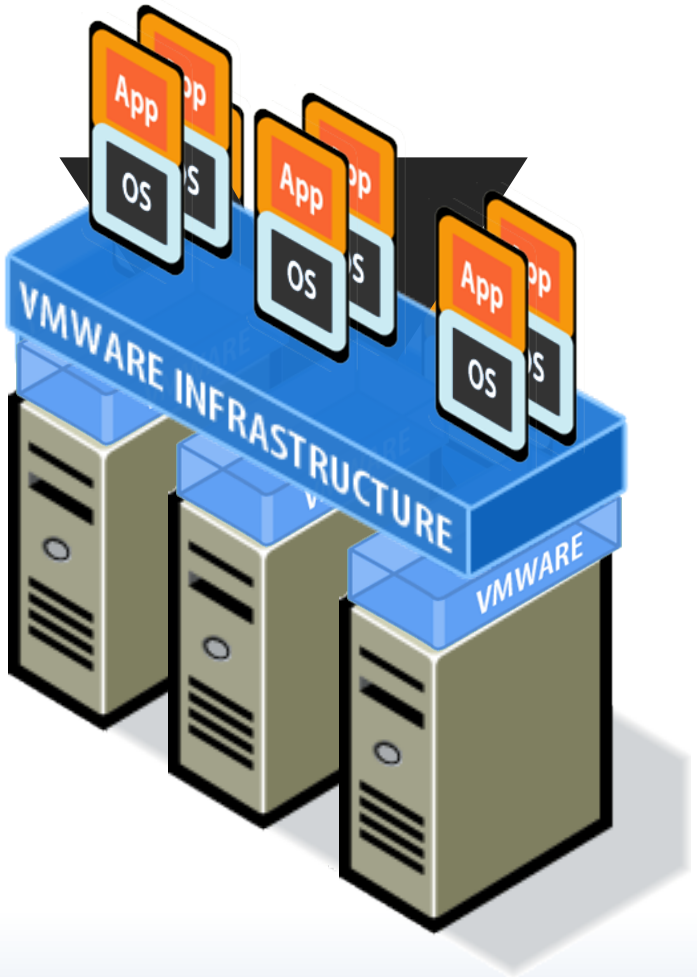
Availability in Virtualization

- Fault tolerance system :
  - VMware makes all Servers and Applications protected against component and complete system failure.
  - When system failure occurs, virtual machines will be automatic restarted on other physical servers.



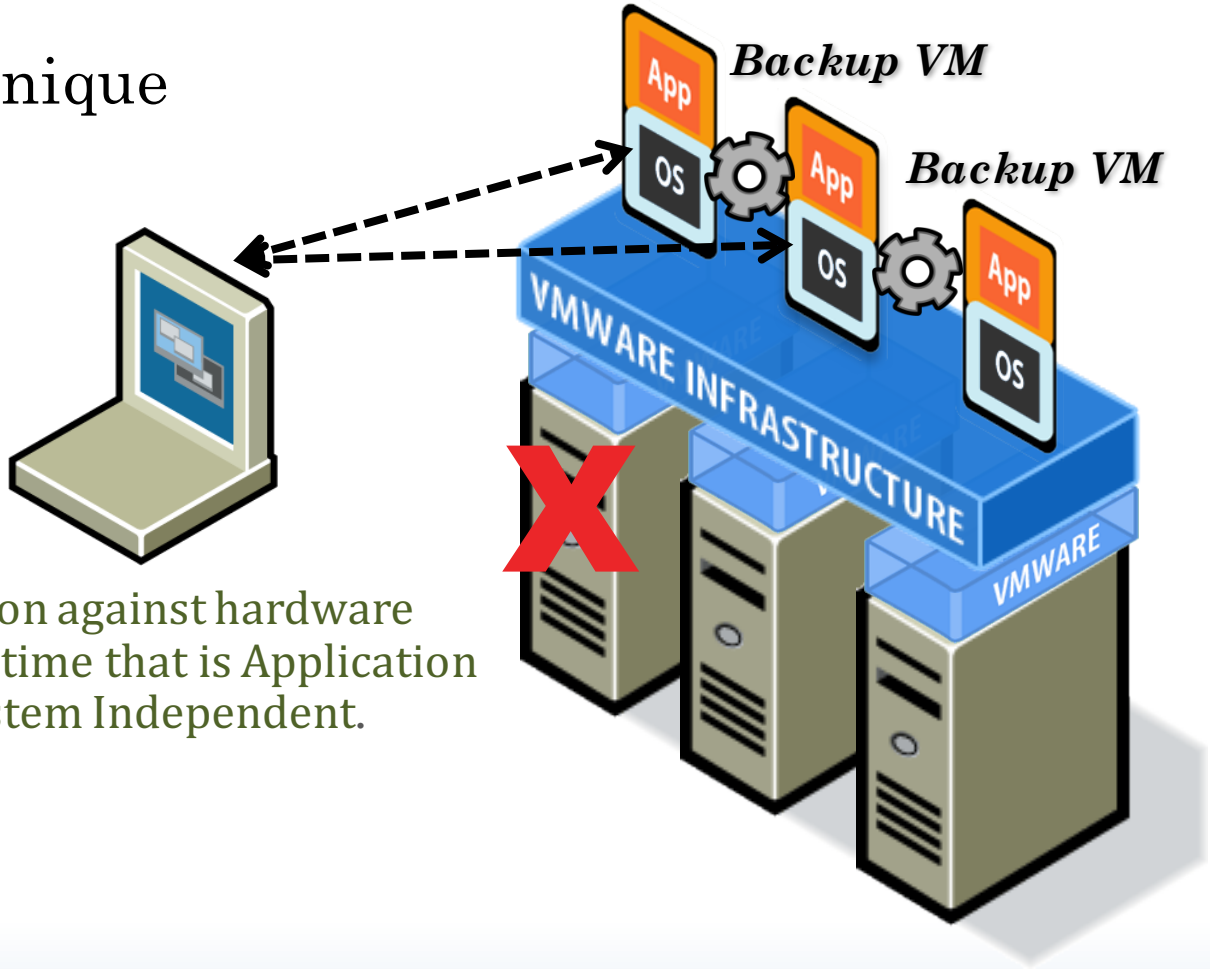
Availability in Virtualization

- Disaster recovery :
  - VMware Site Recovery Manager enables an easy transition from a production site to a Disaster Recovery site.
  - Easy Execution for real Disaster
  - Easy Testing for good night sleep



Availability in Virtualization

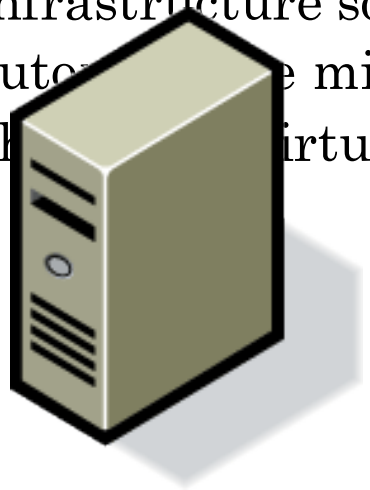
- Fail over technique

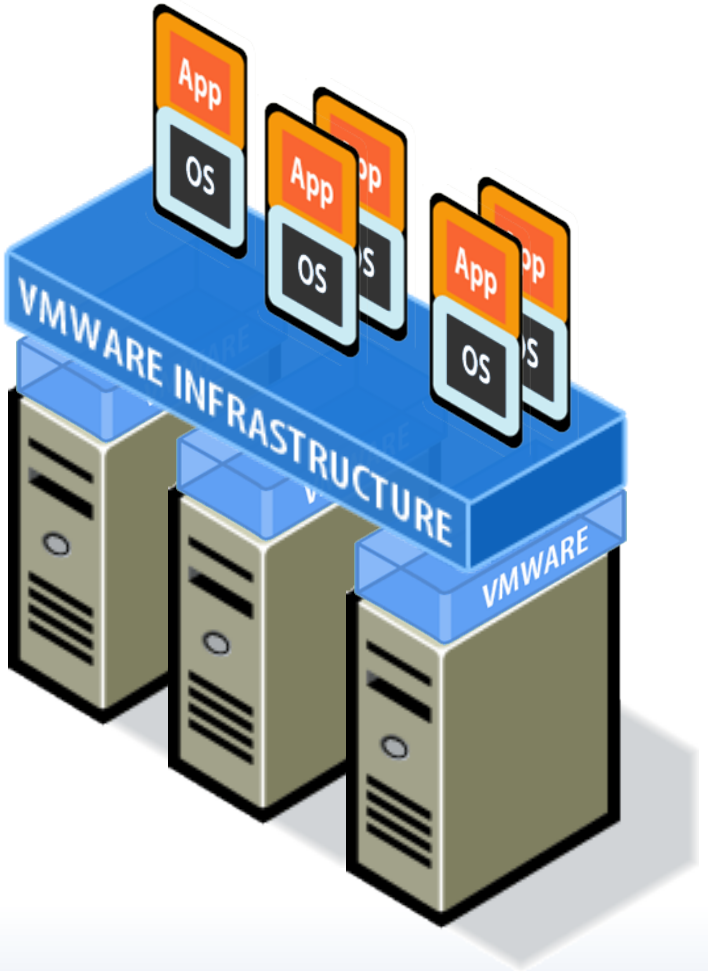


Application protection against hardware failures, with NO down time that is Application and Operating System Independent.

Manageability in Virtualization

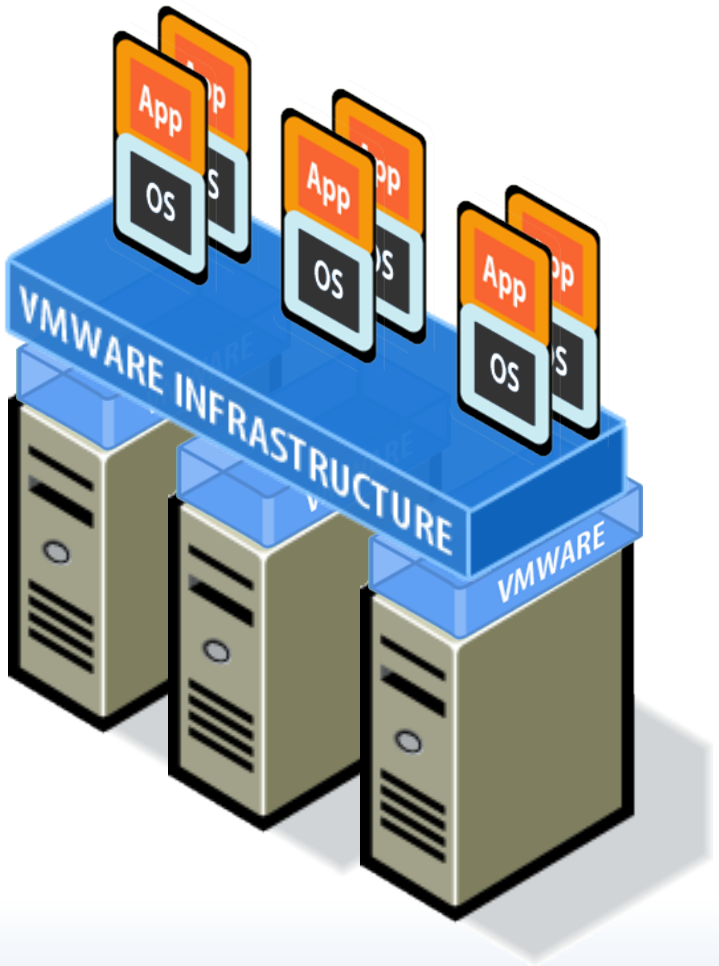
- Provide physical to virtual translation :
  - Consolidation Management with the VMware Infrastructure software will auto[illegible]e migration from ph[illegible]irtual machines.





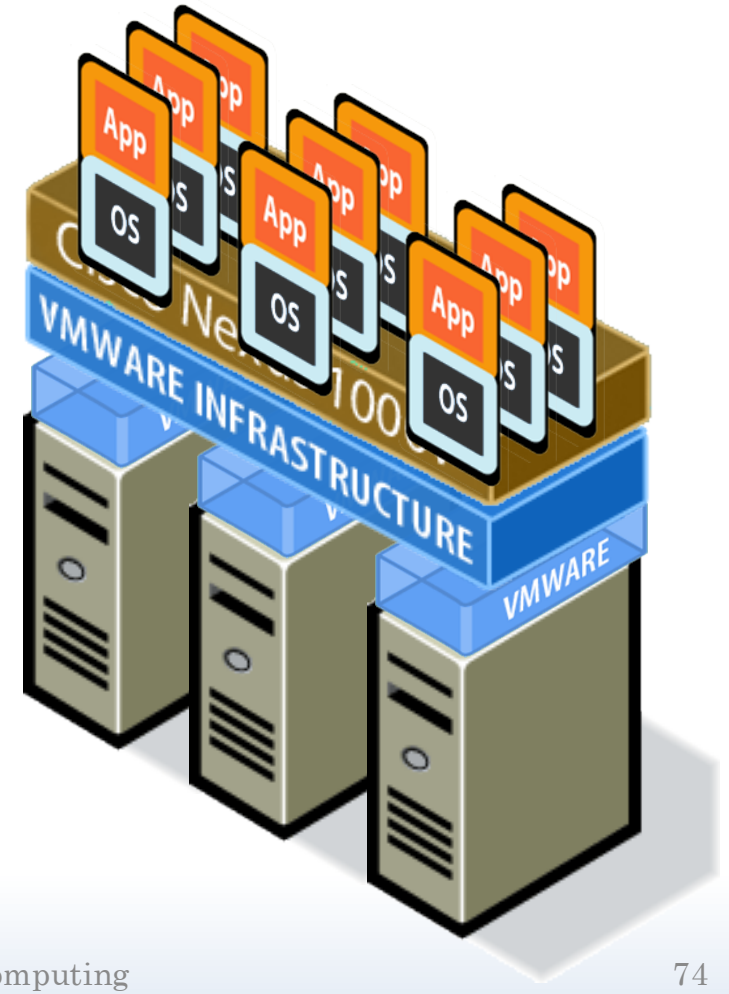
Performance in Virtualization

- Dynamic load balancing :
  - VMware Distributed Resource Scheduler automatically balances the Workloads according to set limits and guarantees.
  - Removing the need to predict resource assignment.



Performance in Virtualization

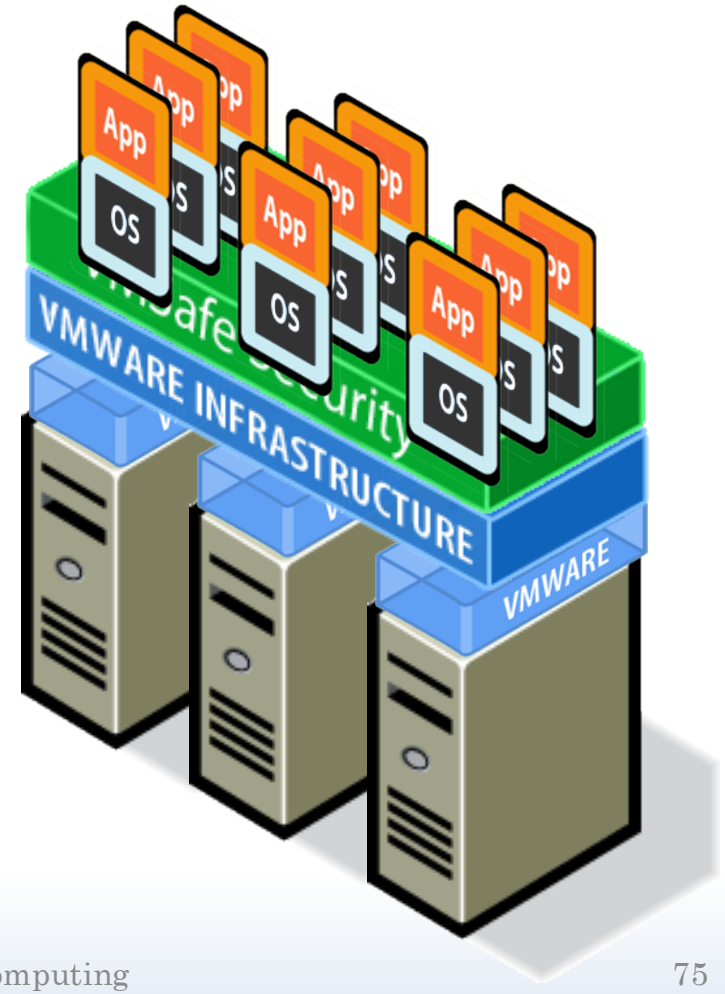
- Optimize network access :
  - VMware and Cisco are collaborating to enhance workload mobility and simpler management with virtualization-aware networks.



CISCO **Nexus 1000V**

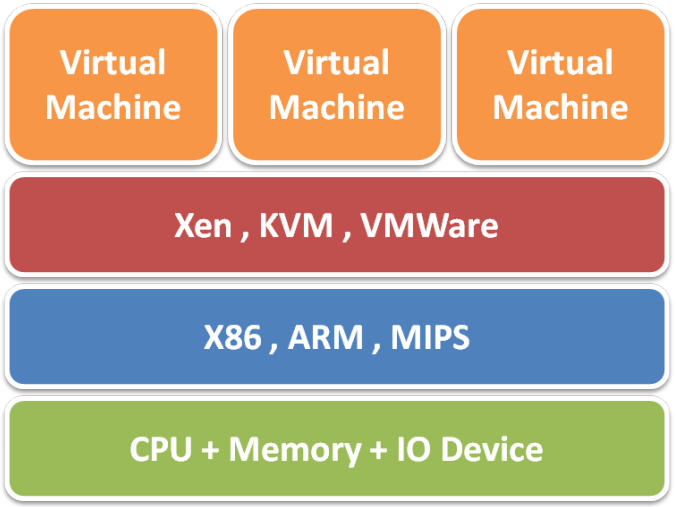
Security in Virtualization

- Enhance virtual machine security protection :
  - The Application vService VMSafe allows security vendors to add superior security solutions inside the VMware Infrastructure.



Summary

- Server virtualization technique :
  - CPU virtualization
    - Ring compression, Intel VT-x, …etc
  - Memory virtualization
    - Shadow page table, Intel EPT, …etc
  - IO virtualization
    - Device model, Intel VT-d, PCIe SR-IOV, .

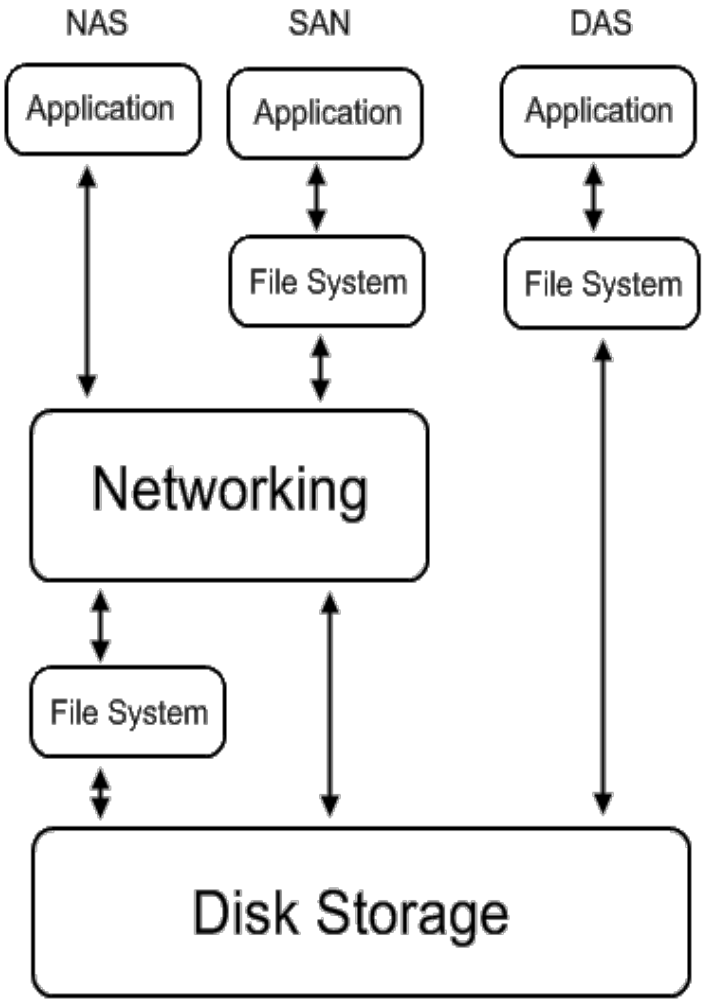


- Ecosystem :
  - VMware implements both type-1 & type-2 virtualization
  - Xen implements both para and full virtualization
  - KVM implements in Linux mainstream kernel
- Cloud properties :
  - Enabled by live migration technique
  - Scalability, Availability, Manageability and Performance

Storage Virtualization

Storage Architecture

- DAS - Direct Attached Storage
  - Storage device was directly attached to a server or workstation, without a storage network in between.
- NAS - Network Attached Storage
  - File-level computer data storage connected to a computer network providing data access to heterogeneous clients.
- SAN - Storage Area Network
  - Attach remote storage devices to servers in such a way that the devices appear as locally attached to the operating system.



Storage Virtualization

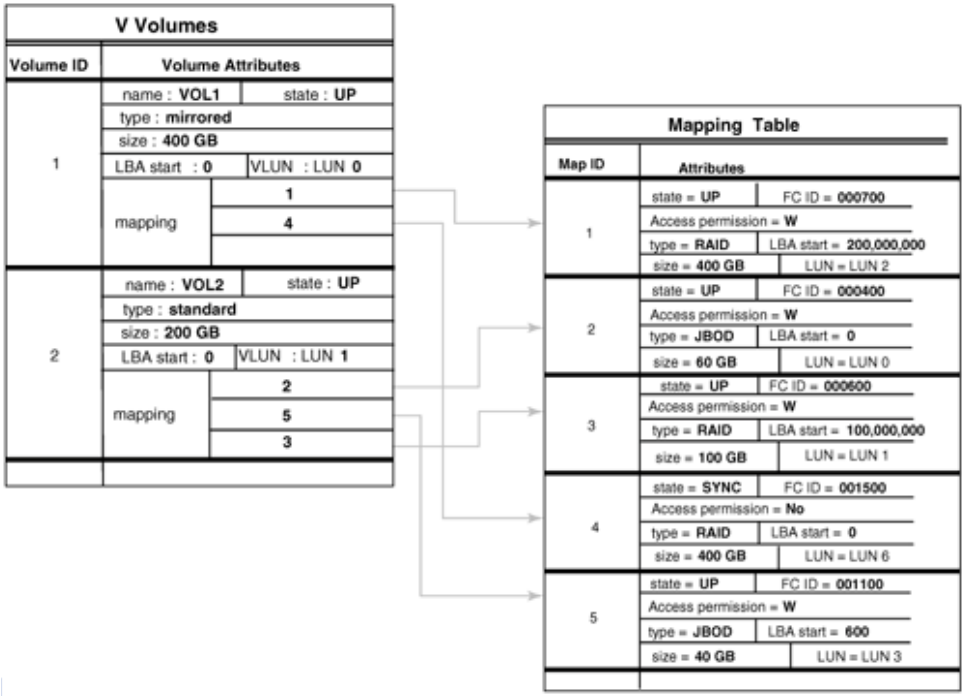
- Desirable properties of storage virtualization:
  - Manageability
    - Storage resource should be easily configured and deployed.
  - Availability
    - Storage hardware failures should not affect the application.
  - Scalability
    - Storage resource can easily scale up and down.
  - Security
    - Storage resource should be securely isolated.

Storage Virtualization

- Storage concept and technique
  - Storage resource mapping table
  - Redundant data
  - Multi-path
  - Data sharing
  - Tiering

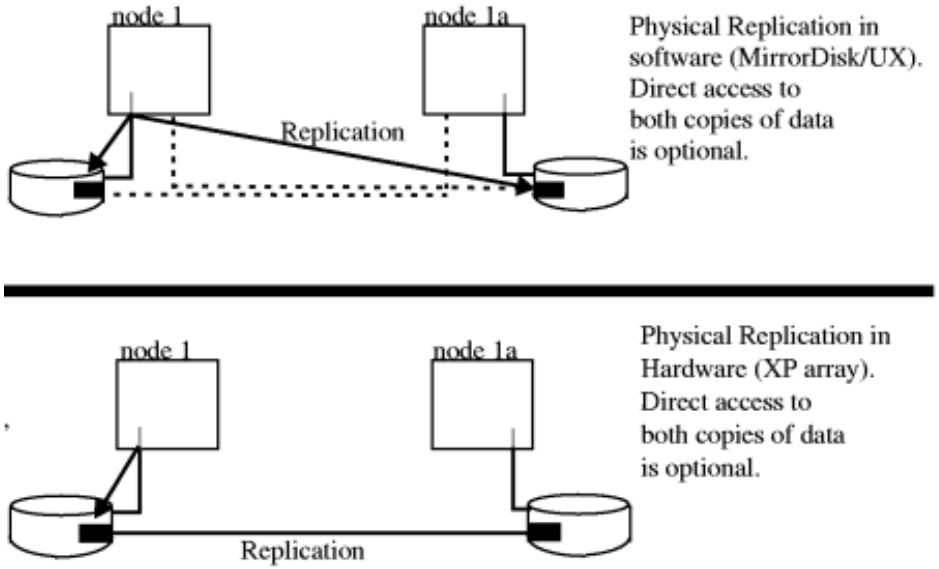
Concept and Technique

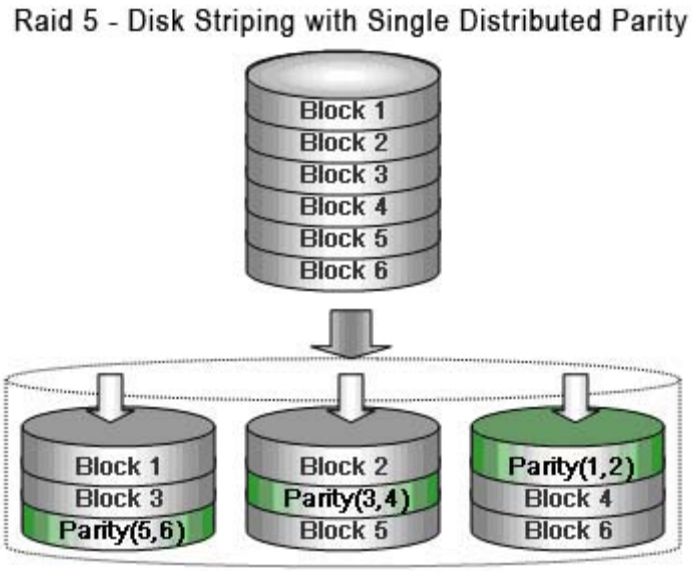
- Storage resource mapping table
  - Maintain tables to map storage resource to target.
  - Dynamic modify table entries for thin provisioning.
  - Use table to isolate different storage address space.



Concept and Technique

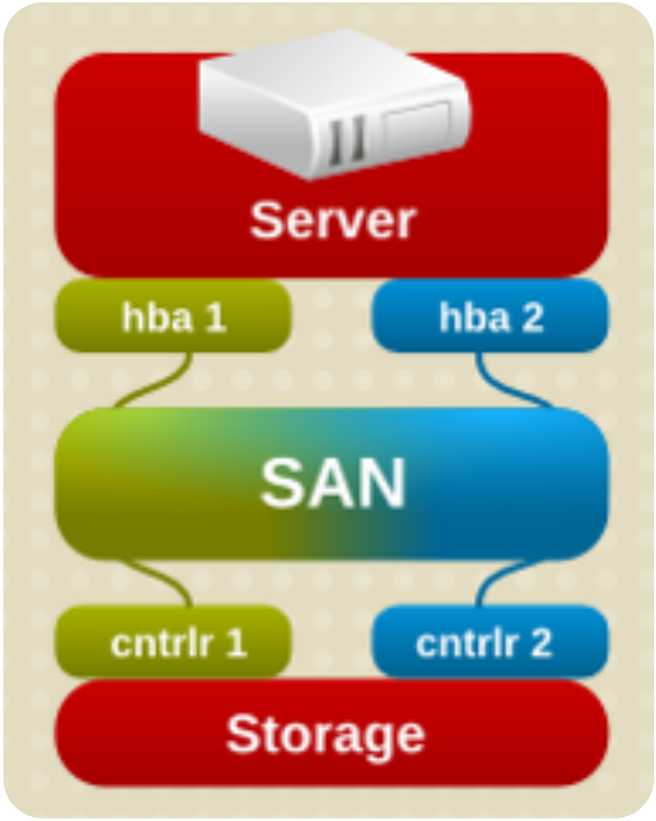
- Redundant data
  - Maintain replicas to provide high availability.
  - Use RAID technique to improve performance and availability.





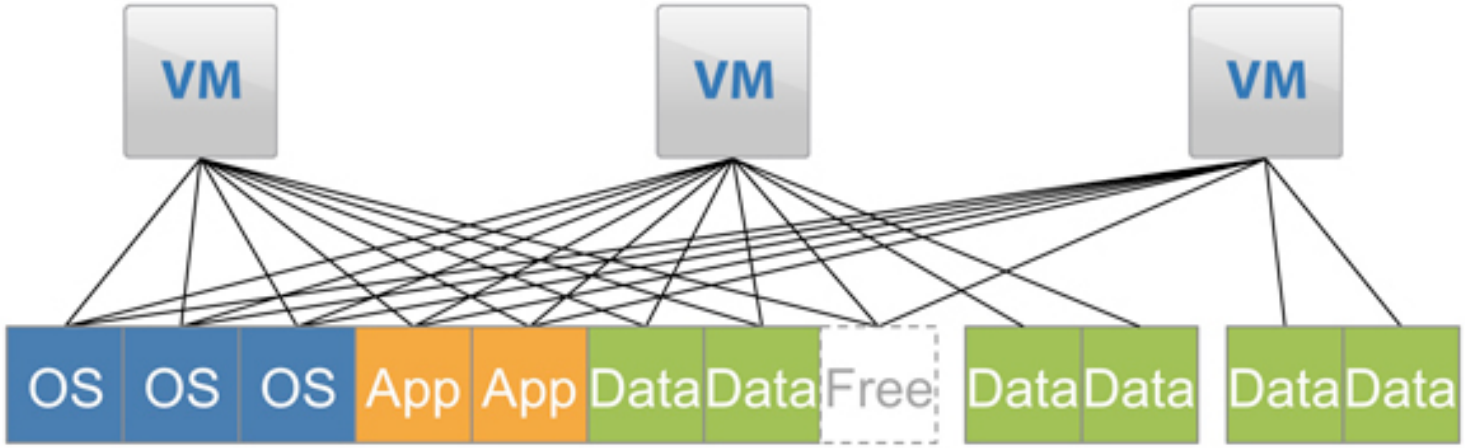
Concept and Technique

- Multi-path
  - A fault-tolerance and performance enhancement technique.
  - There is more than one physical path between the host and storage devices through the buses, controllers, switches, and bridge devices connecting them.



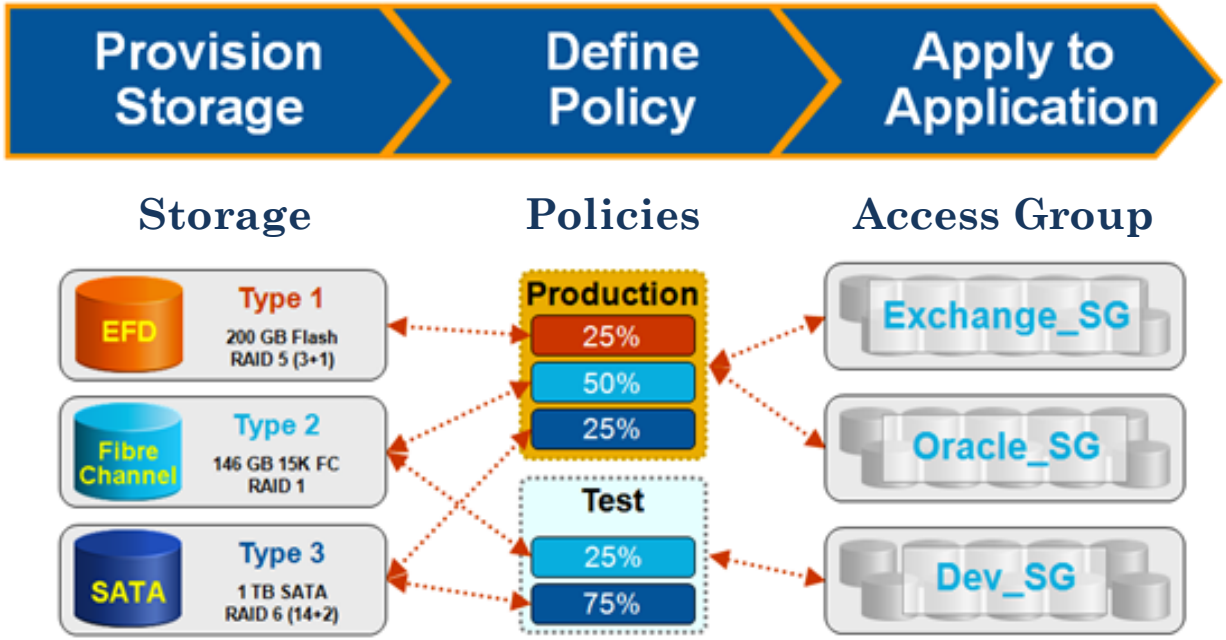
Concept and Technique

- Data sharing
  - Use data de-duplication technique to eliminate duplicated data.
  - Saving and improving the usage of storage space



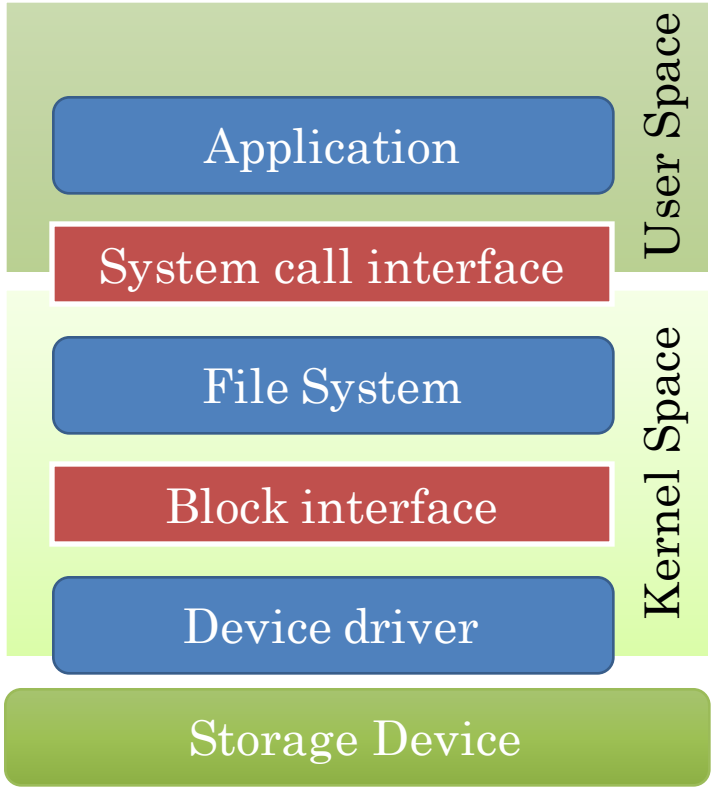
Concept and Technique

- Tiering
  - Automatic migrate data across storage resources with different properties according to the significance or access frequency of data.



What To Be Virtualized

- Layers can be virtualized
  - File system
    - Provide compatible system call interface to user space applications.
  - Block device
    - Provide compatible block device interface to file system.
    - Through the interface such as SCIS, SAS, ATA, SATA, etc.



File System Level

- Data and Files
  - What is data ?
    - Data is information that has been converted to a machine-readable, digital binary format.
    - Control information indicates how data should be processed.
    - Applications may embed control information in user data for formatting or presentation.
    - Data and its associated control information is organized into discrete units as files or records.
  - What is file ?
    - Files are the common containers for user data, application code, and operating system executables and parameters.

File System Level

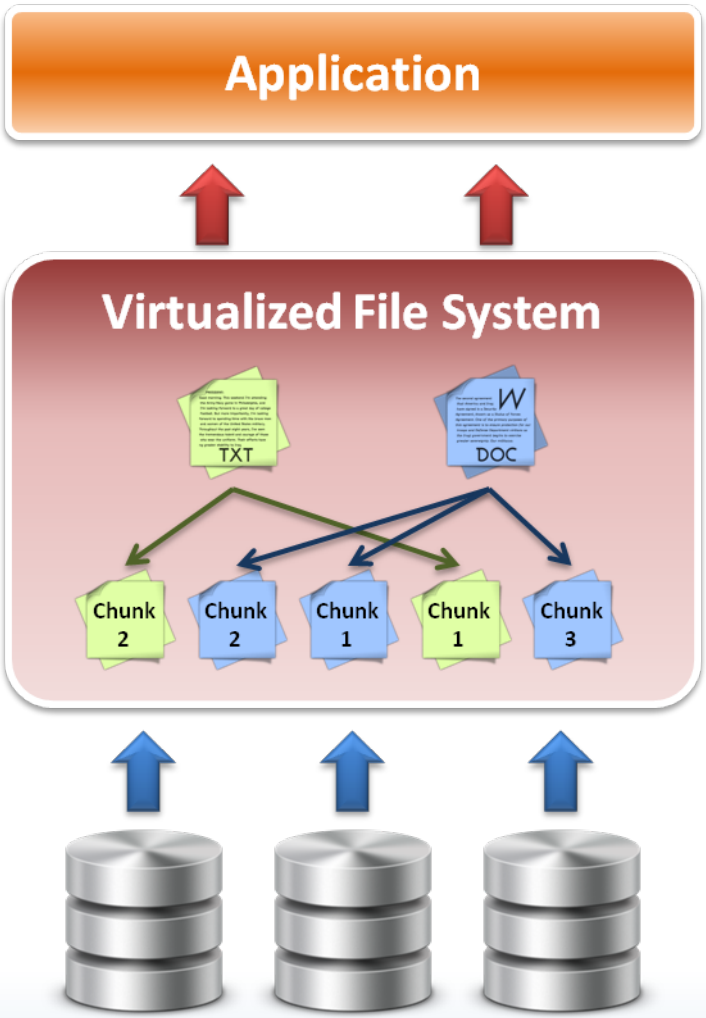
- About the files
  - Metadata
    - The control information for file management is known as metadata.
    - File metadata includes file attributes and pointers to the location of file data content.
    - File metadata may be segregated from a file's data content.
    - Metadata on file ownership and permissions is used in file access.
    - File timestamp metadata facilitates automated processes such as backup and life cycle management.
  - Different file systems
    - In Unix systems, file metadata is contained in the i-node structure.
    - In Windows systems, file metadata is contained in records of file attributes.

File System Level

- File system
  - What is file system ?
    - A file system is a software layer responsible for organizing and policing the creation, modification, and deletion of files.
    - File systems provide a hierarchical organization of files into directories and subdirectories.
    - The B-tree algorithm facilitates more rapid search and retrieval of files by name.
    - File system integrity is maintained through duplication of master tables, change logs, and immediate writes off file changes.
  - Different file systems
    - In Unix, the super block contains information on the current state of the file system and its resources.
    - In Windows NTFS, the master file table contains information on all file entries and status.

File System Level

- File system level virtualization
  - File system maintains metadata (i-node) of each file.
  - Translate file access requests to underlining file system.
  - Sometime divide large file into small sub-files (chunks) for parallel access, which improves the performance

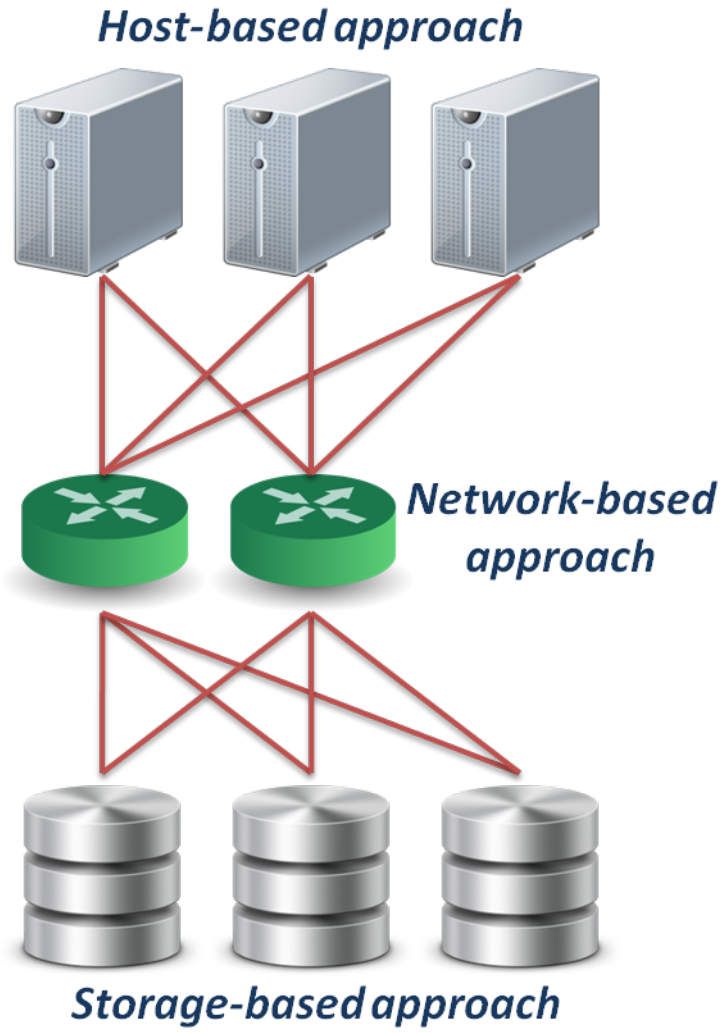


Block Device Level

- Block level data
  - The file system block
    - The atomic unit of file system management is the file system block.
    - A file's data may span multiple file system blocks.
    - A file system block is composed of a consecutive range of disk block addresses.
  - Data in disk
    - Disk drives read and write data to media through cylinder, head, and sector geometry.
    - Microcode on a disk translates between disk block numbers and cylinder/head/sector locations.
    - This translation is an elementary form of virtualization.

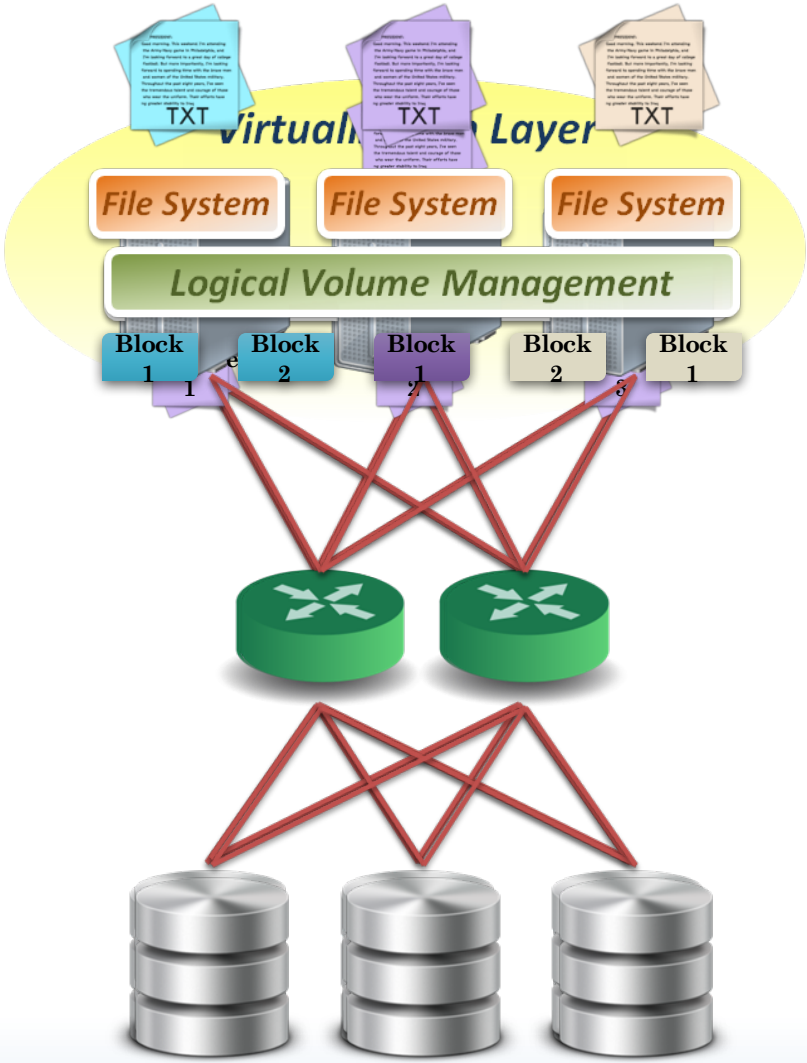
Where To Be Virtualized

- Different approaches :
  - Host-based approach
    - Implemented as a software running on host systems.
  - Network-based approach
    - Implemented on network devices.
  - Storage-based approach
    - Implemented on storage target subsystem.



Host-based Virtualization

- File level
  - Run virtualized file system on the host to map files into data blocks, which distributed among several storage devices.
- Block level
  - Run logical volume management software on the host to intercept I/O requests and redirect them to storage devices.
- Provide services
  - Software RAID

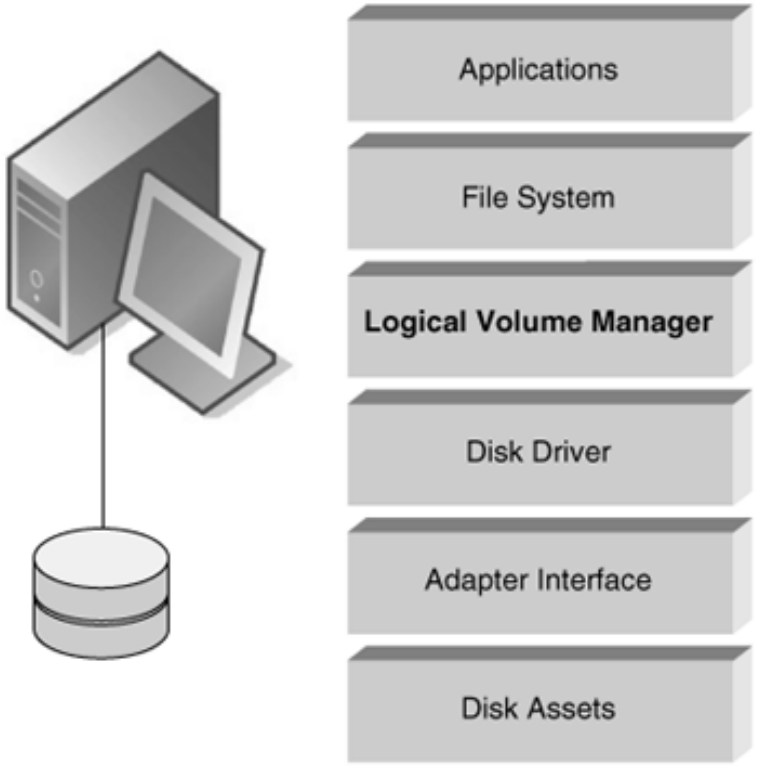


Host-based Virtualization

- Important issues
  - Storage metadata servers
    - Storage metadata may be shared by multiple servers.
    - Shared metadata enables a SAN file system view for multiple servers.
    - Provides virtual to real logical block address mapping for client.
    - A distributed SAN file system requires file locking mechanisms to preserve data integrity.
  - Host-based storage APIs
    - May be implemented by the operating system to provide a common interface to disparate virtualized resources.

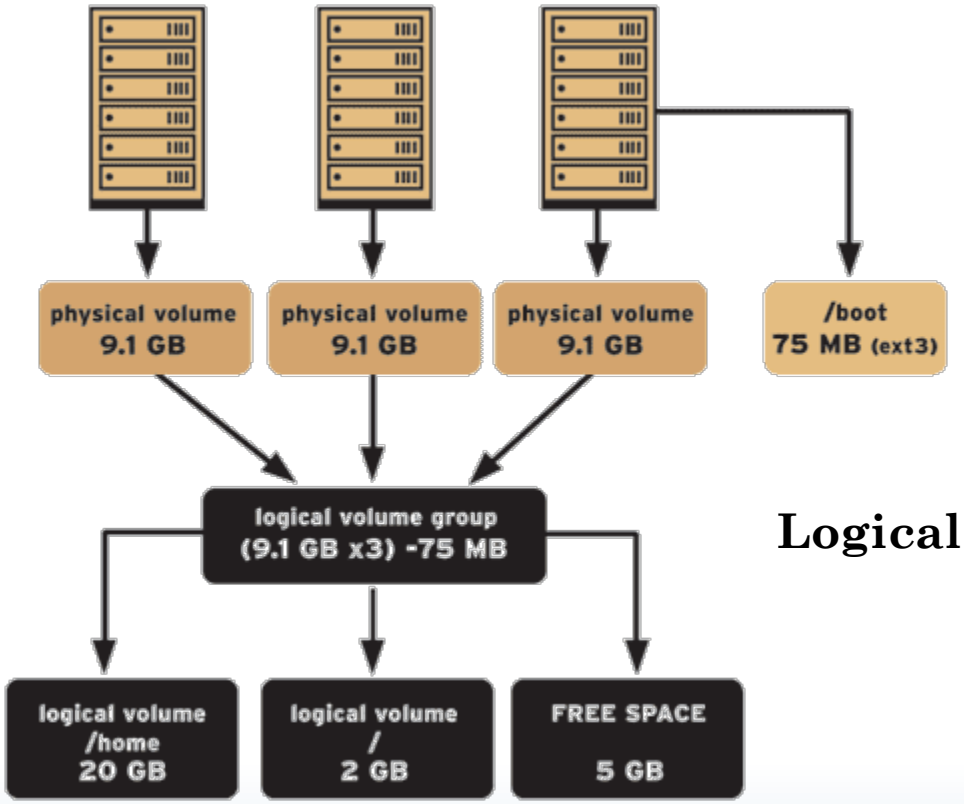
Host-based Virtualization

- A typical example :
  - LVM
    - Software layer between the file system and the disk driver.
    - Executed by the host CPU.
    - Lack hardware-assist for functions such as software RAID.
    - Independence from vendor-specific storage architectures.
    - Dynamic capacity allocation to expand or shrink volumes.
    - Support alternate pathing for high availability.



LVM(1/2)

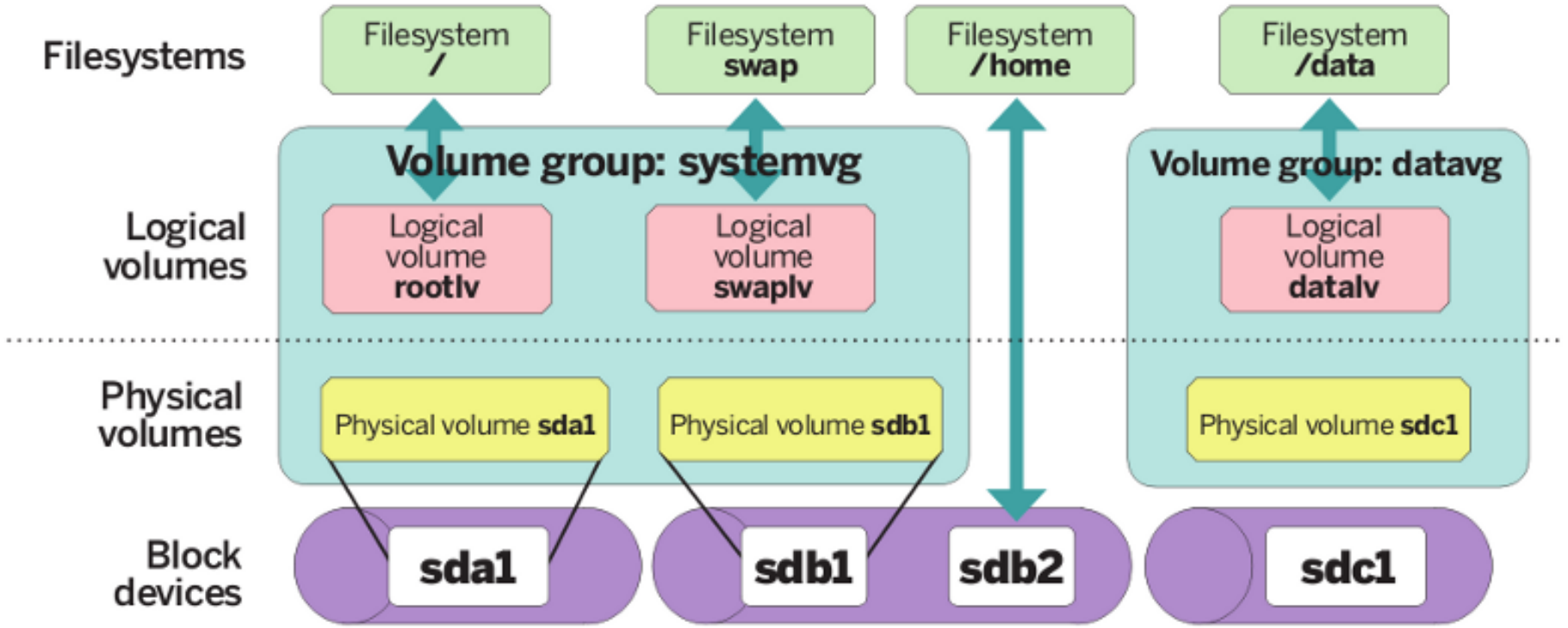
- LVM is a logical volume manager for the Linux kernel; it manages disk drives and similar mass-storage devices.



Logical Volume Manager

LVM(2/2) : Example

- Disk partition → physical volumes → volume group → logical volumes → file systems

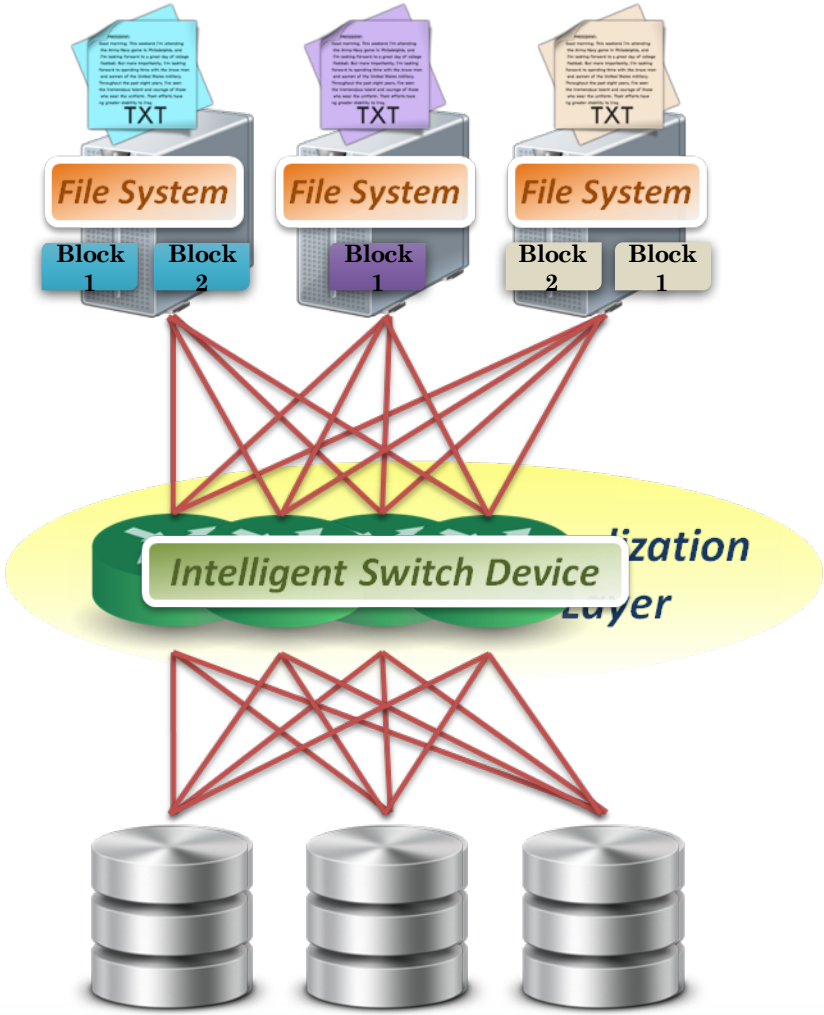


Host-based Virtualization

- Host-based implementation
  - Pros
    - No additional hardware or infrastructure requirements
    - Simple to design and implement
    - Improve storage utilization
  - Cons
    - Storage utilization optimized only on a per host base
    - Software implementation is depending on each operating system
    - Consume CPU clock cycle for virtualization
  - Examples
    - LVM, NFS

Network-based Virtualization

- Network-based approach
  - File level
    - Seldom implement file level virtualization on network device.
  - Block level
    - Run software on dedicated appliances or intelligent switches and routers.
  - Provide services
    - Multi-path
    - Storage pooling

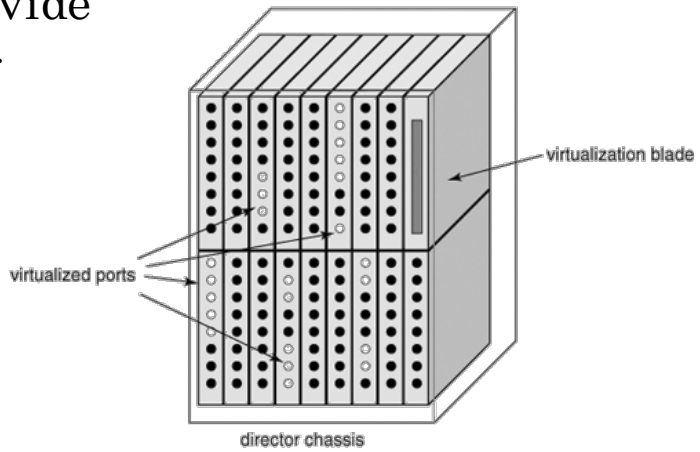


Network-based Virtualization

- Requirements of storage network
  - Intelligent services
    - Logon services
    - Simple name server
    - Change notification
    - Network address assignment
    - Zoning
  - Fabric switch should provide
    - Connectivity for all storage transactions
    - Interoperability between disparate servers, operating systems, and target devices

Network-based Virtualization

- Techniques for fabric switch virtualization
  - Hosted on departmental switches
    - A PC engine provisioned as an option blade.
  - Data center directors
    - Should be able to preserve the five nines availability characteristic of director-class switches.
    - Dedicated virtualization ASICs provide high-performance frame processing and block address mapping.
  - Interoperability between different implementations will become a priority.

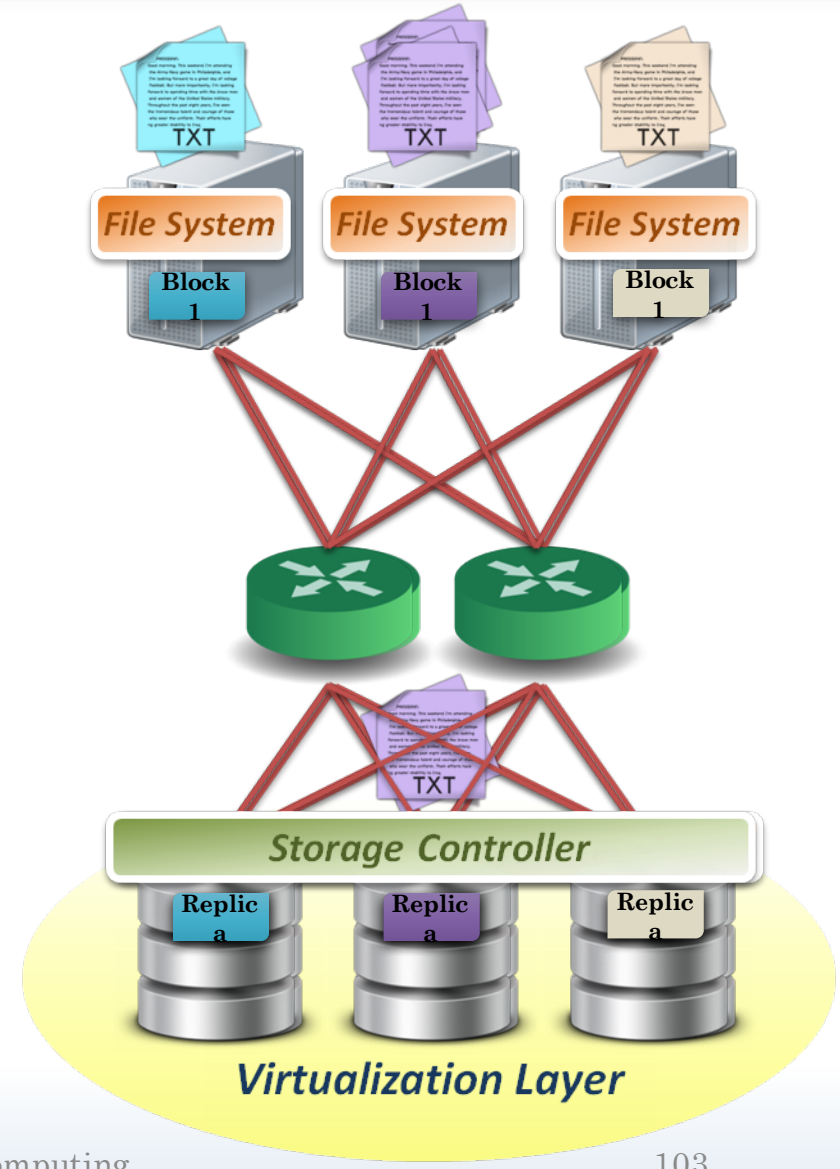


Network-based Virtualization

- Pros
  - True heterogeneous storage virtualization
  - No need for modification of host or storage system
  - Multi-path technique improve the access performance
- Cons
  - Complex interoperability matrices - limited by vendors support
  - Difficult to implement fast metadata updates in switch device
  - Usually require to build specific network equipments (e.g., Fibre Channel)
- Examples
  - IBM SVC (SAN Volume Controller), EMC Invista

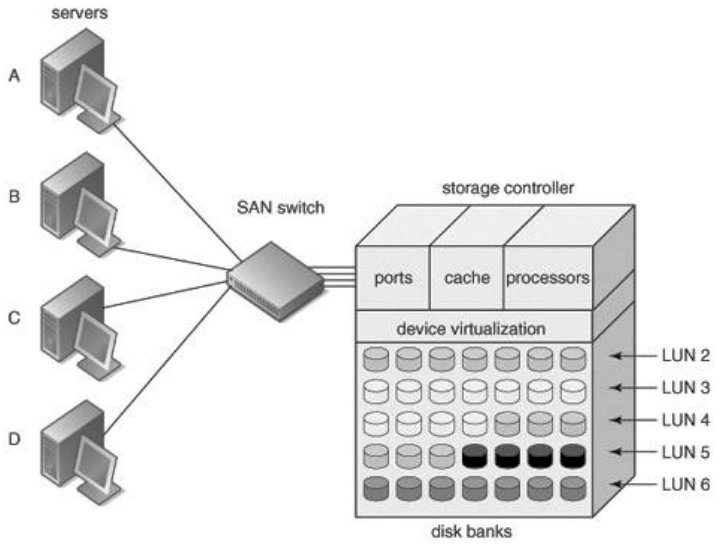
Storage-based Virtualization

- Storage-based approach
  - File level
    - Run software on storage device to provide file based data storage services to host through network.
  - Block level
    - Embeds the technology in the target storage devices.
  - Provide services
    - Storage pooling
    - Replication and RAID
    - Data sharing and tiering



Storage-based Virtualization

- Array-based virtualization
  - Storage controller
    - Provide basic disk virtualization in the form of RAID management, mirroring, and LUN mapping or masking.
    - Allocate a single LUN to multiple servers.
    - Offer Fibre Channel, iSCSI, and SCSI protocol.
  - Cache memory
    - Enhance performance.
  - Storage assets coordination
    - Coordination between multiple storage systems is necessary to ensure high availability.



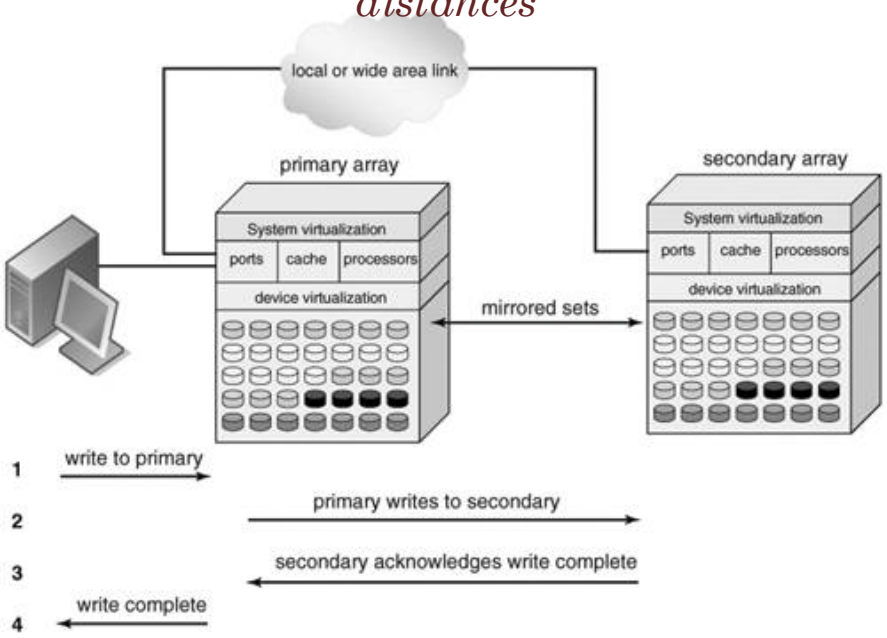
Storage-based Virtualization

- Data replication
  - Array-based data replication
    - Referred to as disk-to-disk replication.
    - Requires that a storage controller function concurrently as both an initiator and target.
  - Synchronous vs. Asynchronous
    - Synchronous data replication ensures that a write operation to a secondary disk array is completed before the primary array acknowledges task completion to the server.
    - Asynchronous data replication provides write completion by the primary array, although the transaction may still be pending to the secondary array.

Storage-based Virtualization

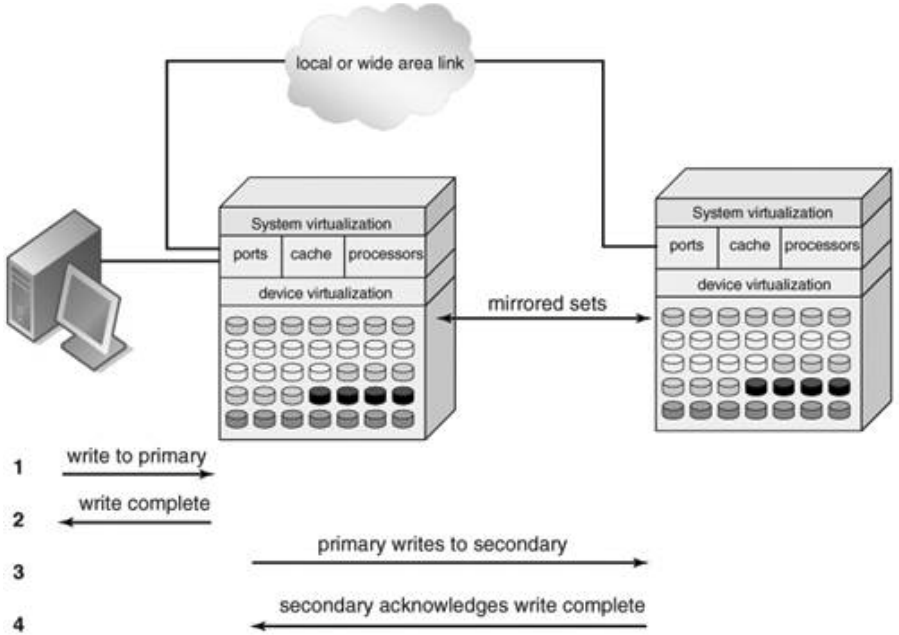
Synchronous

To preserve performance, synchronous data replication is limited to metropolitan distances



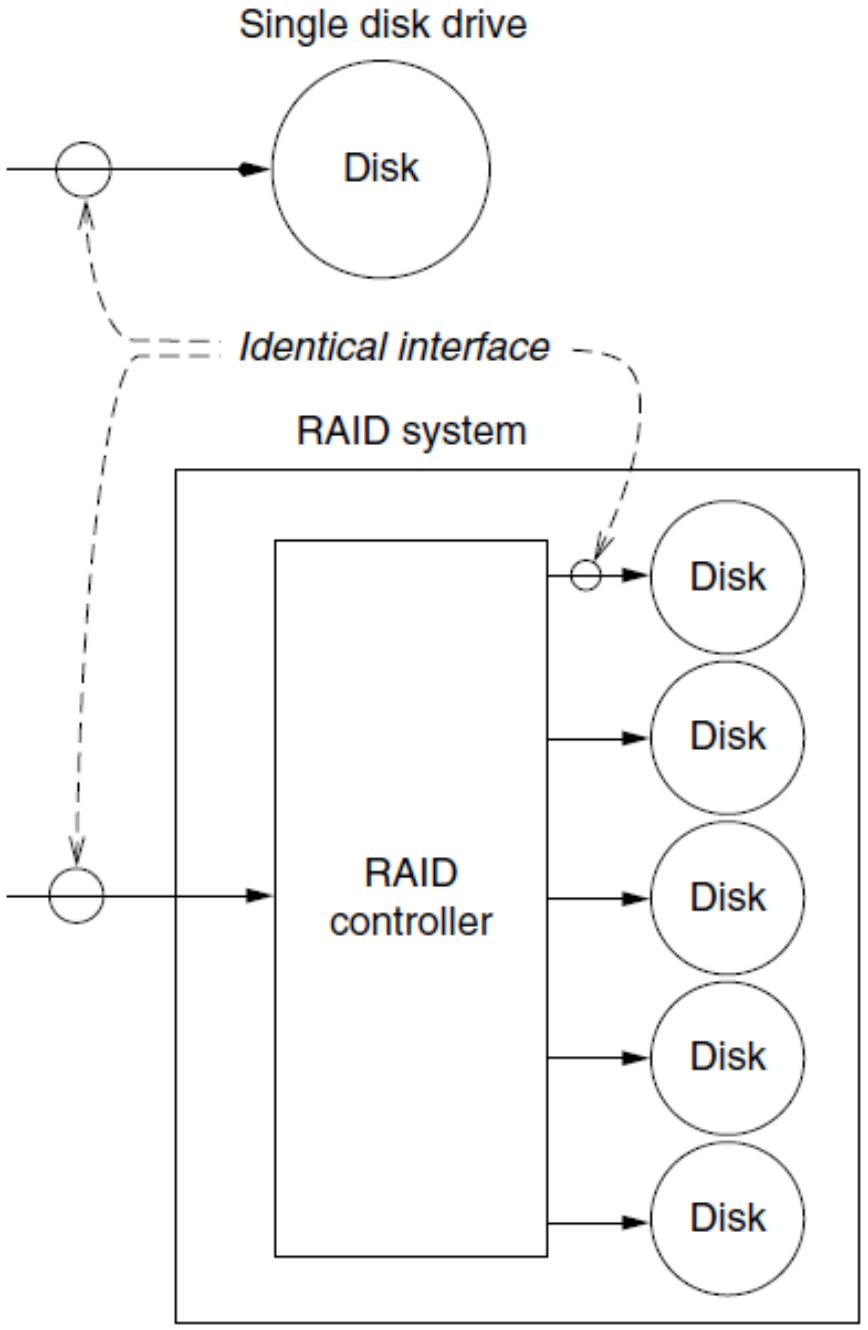
Asynchronous

Asynchronous data replication is largely immune to transmission latency

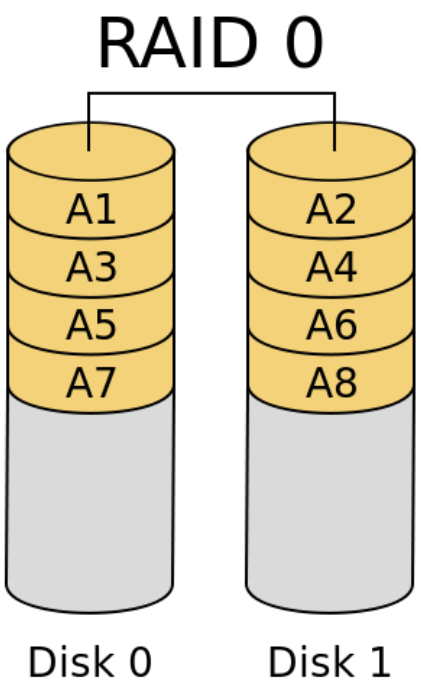


RAID

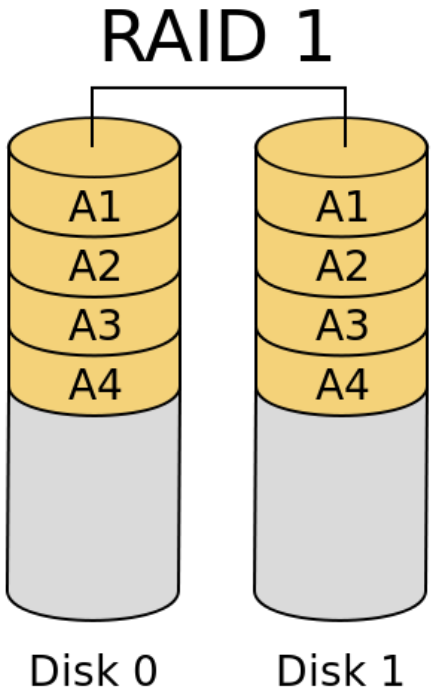
- RAID (redundant array of independent disks), a storage technology that combines multiple disk drive components into a logical unit.
- Data is distributed across the drives in one of several ways called *RAID levels*, depending on the level of redundancy and performance required.



Example : RAID 0 and RAID 1



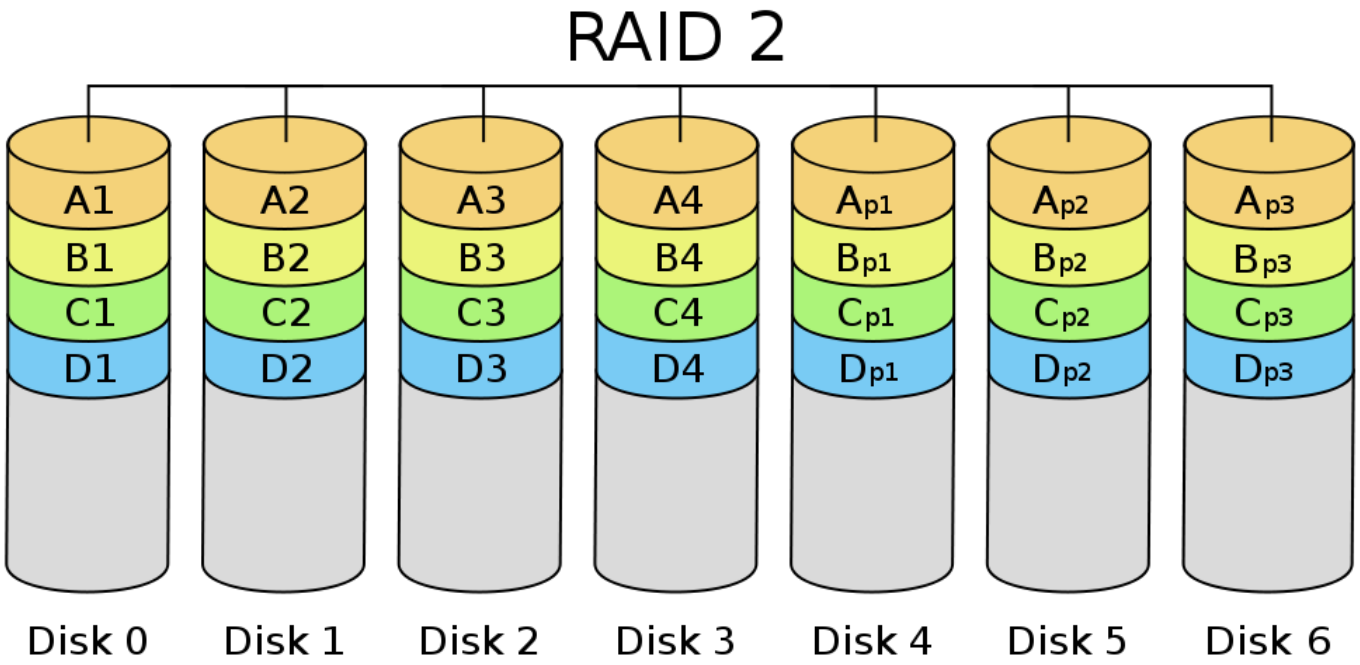
improved performance and additional storage but no fault tolerance (block-level striping without parity or mirroring) .



mirroring without parity or striping

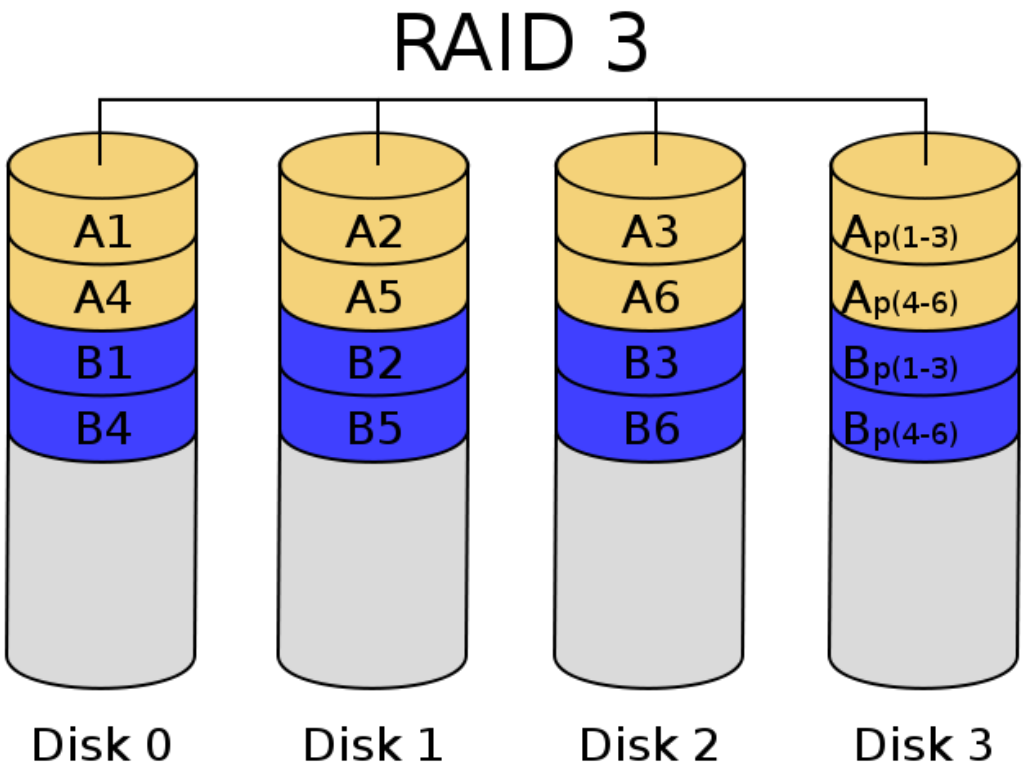
RAID 2

- bit-level striping with dedicated Hamming-code parity



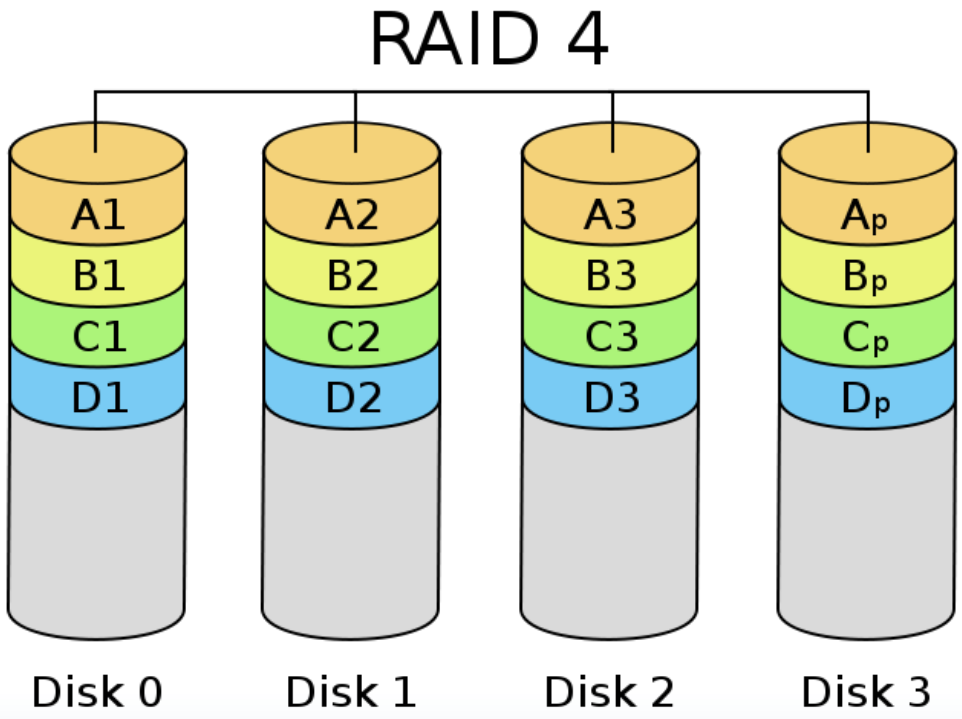
RAID 3

- Bit-level stripping parity
- Independent parity disk



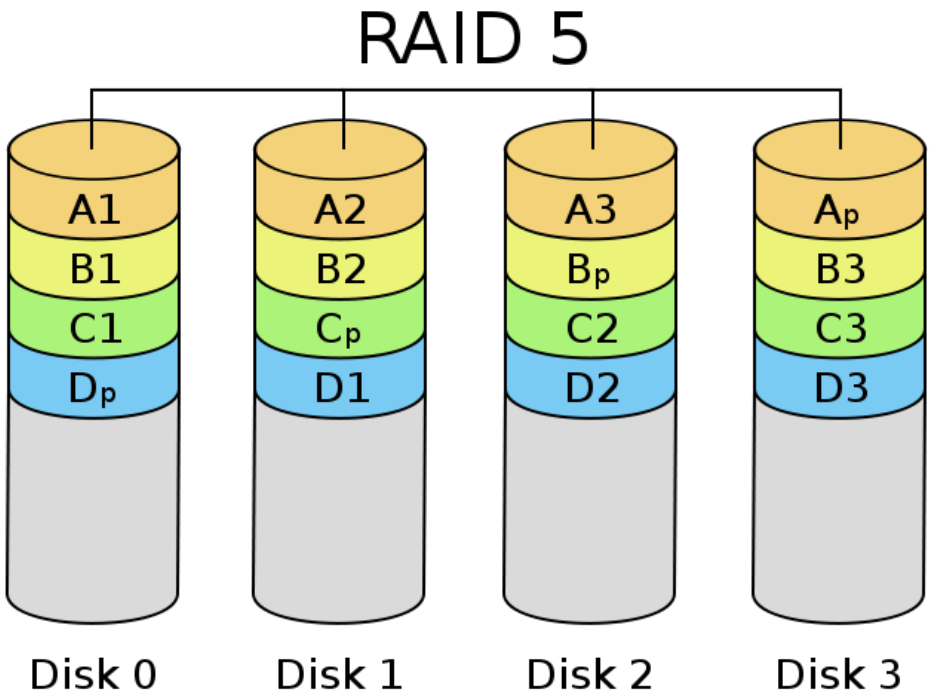
RAID 4 (Block-Level Parity)

- RAID4 block-level parity
- Bottleneck: parity disk



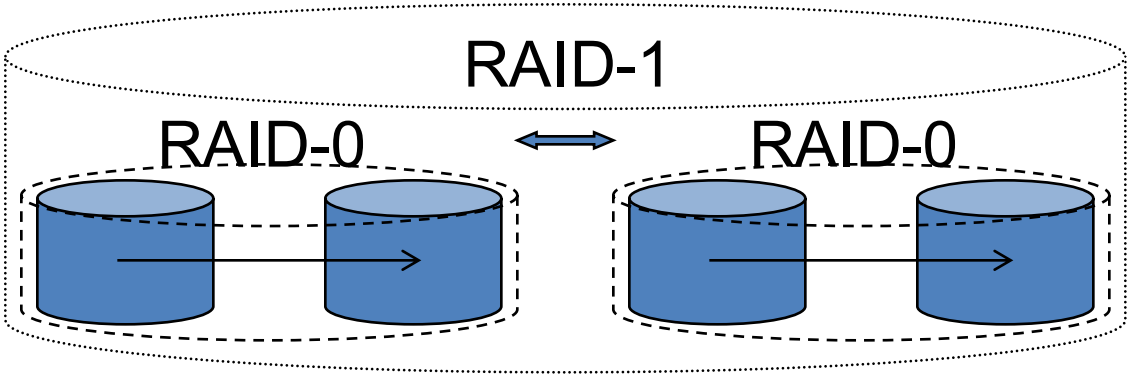
RAID 5

- Block-level striping with distributed parity.
- Spiral parity

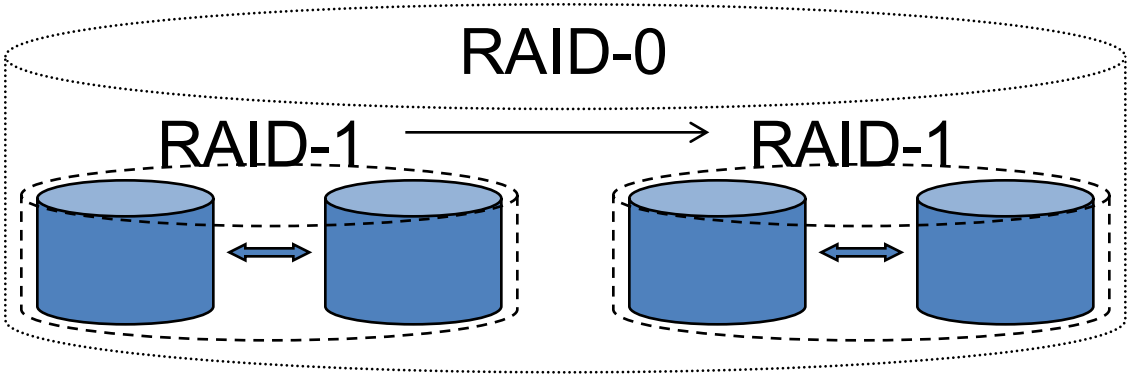


Nested RAID

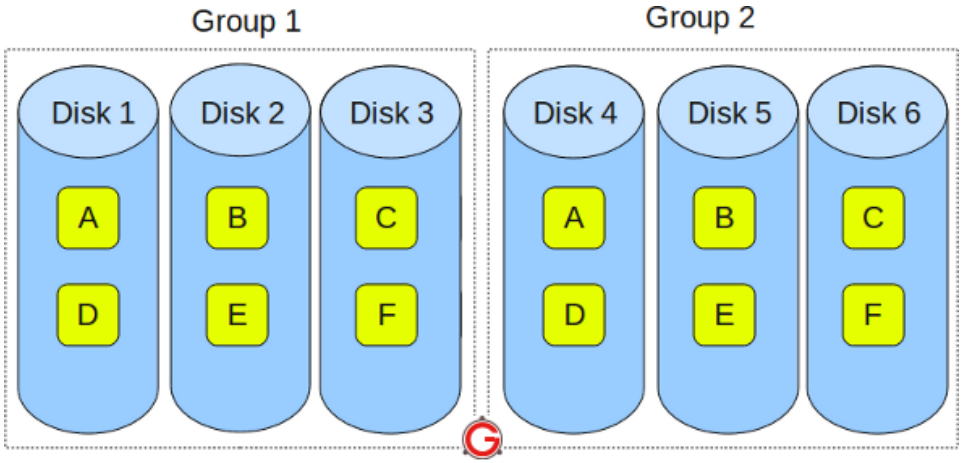
- RAID 0+1



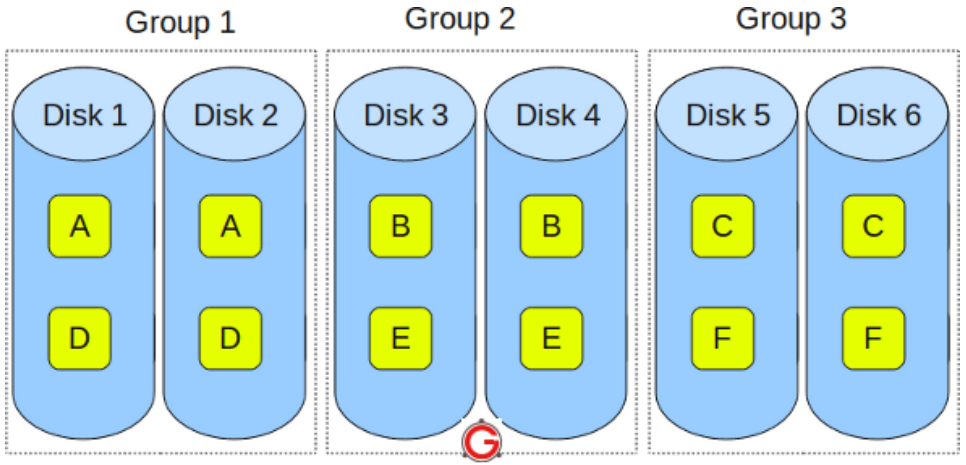
- RAID 1+0



RAID 0+1 vs. RAID 1+0



RAID 01 – Blocks Striped. (and Blocks Mirrored)

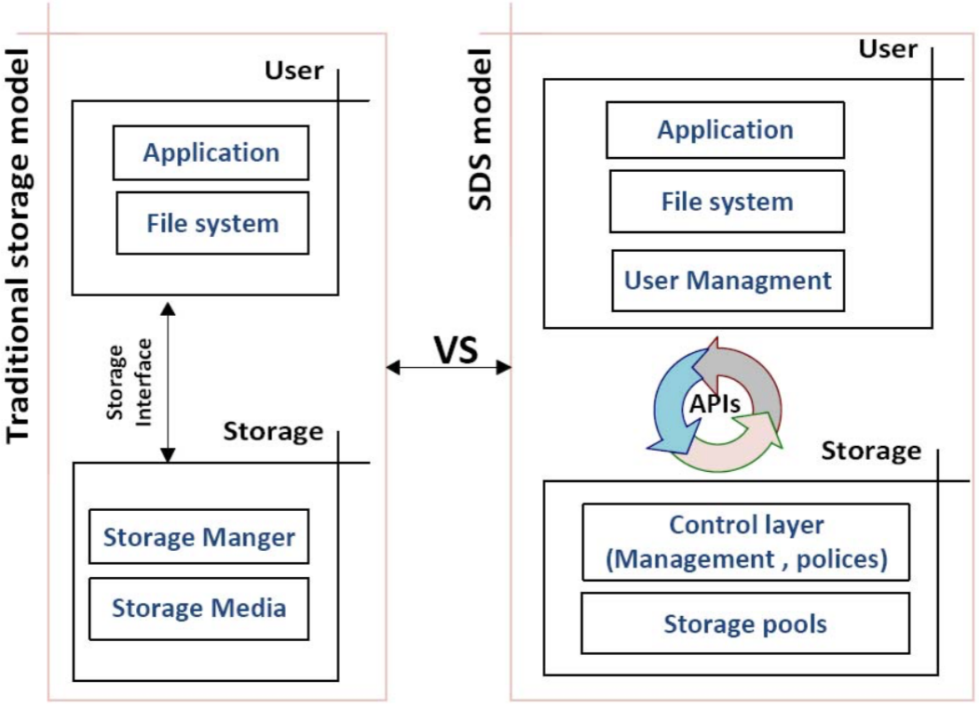


RAID 10 – Blocks Mirrored. (and Blocks Striped)

Software Defined Storage

- Flexibility, scalability, QoS, application-aware requirements, policy-based mgmt

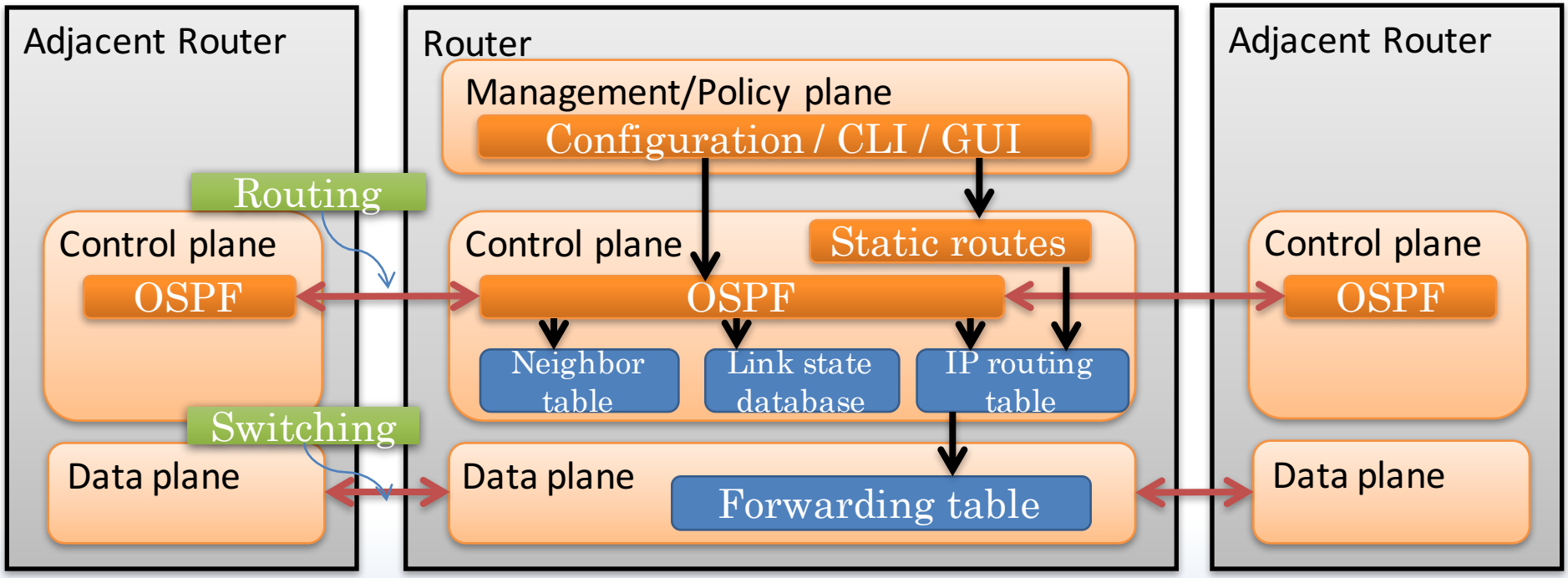
- SDDC: Software Defined Data Center
- HCI: Hyper-Converged Infrastucture



Network Virtualization

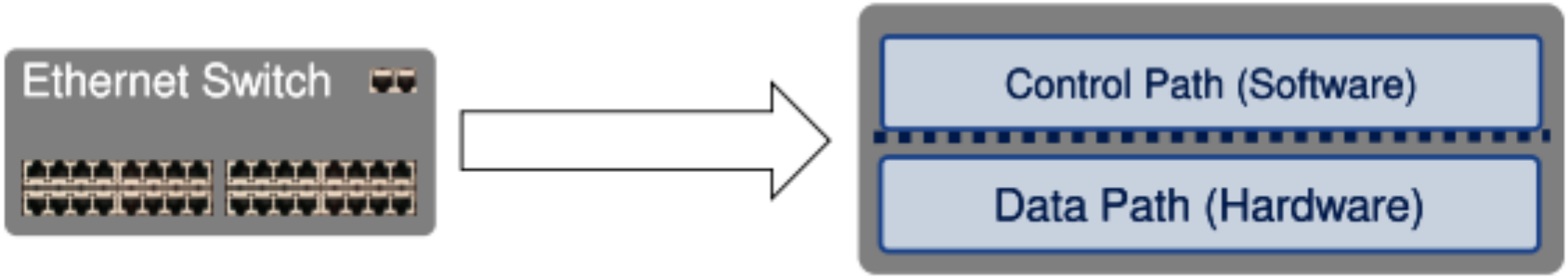
Traditional network node: Router

- Router can be partitioned into control and data plane
  - Management plane/ configuration
  - Control plane / Decision: OSPF (Open Shortest Path First)
  - Data plane / Forwarding



Traditional network node: Switch

- Typical Networking Software
  - Management plane
  - Control Plane – The brain/decision maker
  - Data Plane – Packet forwarder



Software Defined Network (1/2)

- Software defined networking (SDN) is an approach to building computer networks that separates and abstracts elements of these systems
- SDN decouples the system that makes decisions about where traffic is sent (the control plane) from the underlying system that forwards traffic to the selected destination (the data plane)
  - Sometimes we need different decisions for different apps

SDN Concept

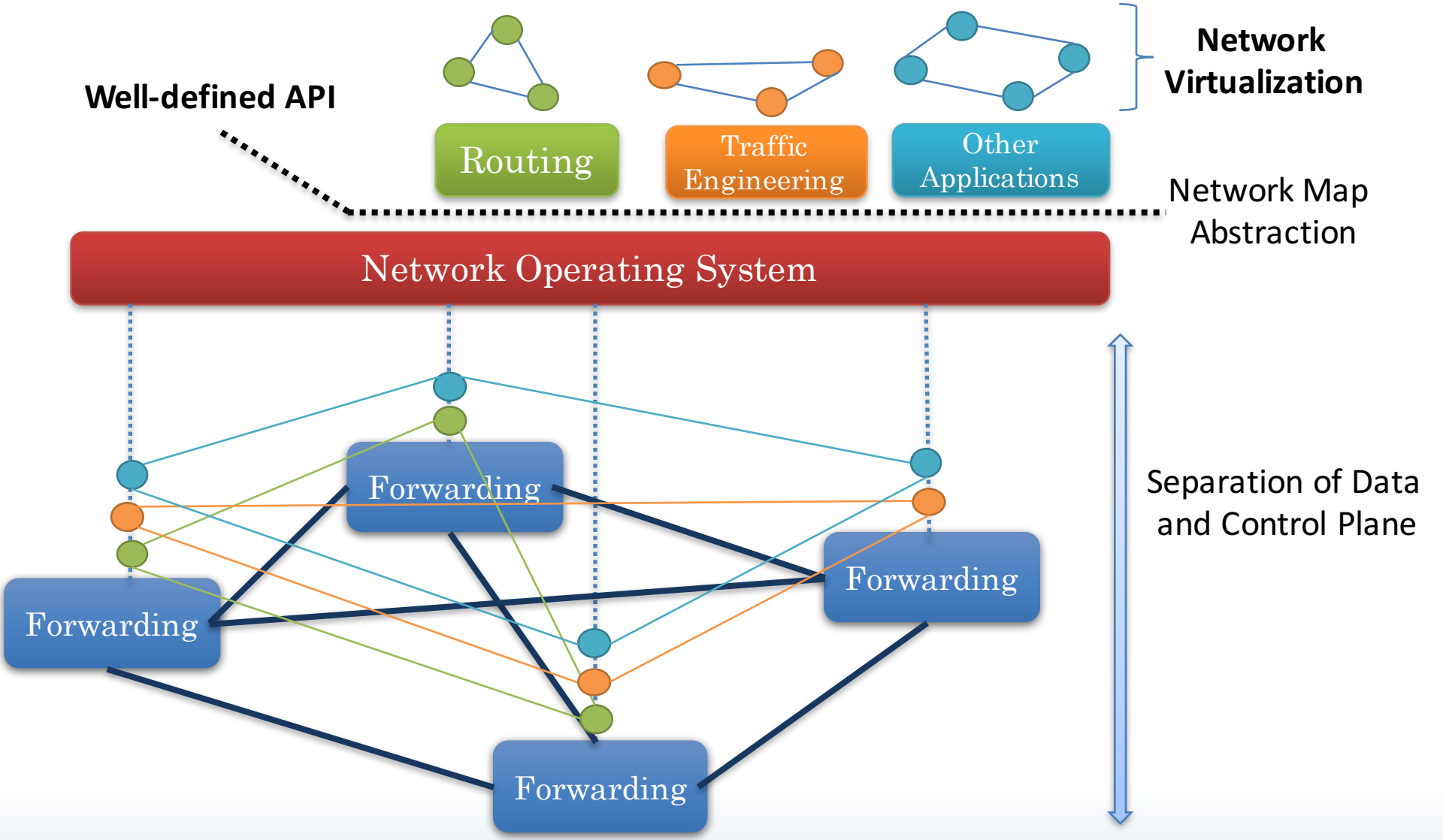
- Separate Control plane and Data plane entities
  - Network intelligence and state are logically centralized
  - The underlying network infrastructure is abstracted from the applications
- Execute or run Control plane software on general purpose hardware
  - Decouple from specific networking hardware
  - Use commodity servers
- Have programmable data planes
  - Maintain, control and program data plane state from a central entity
- An architecture to control not just a networking device but an entire network

Control Program

- Control program operates on view of network
  - Input: global network view (graph/database)
  - Output: configuration of each network device

- Control program is not a distributed system
  - Abstraction hides details of distributed state

Key Abstractions in the Control Plane

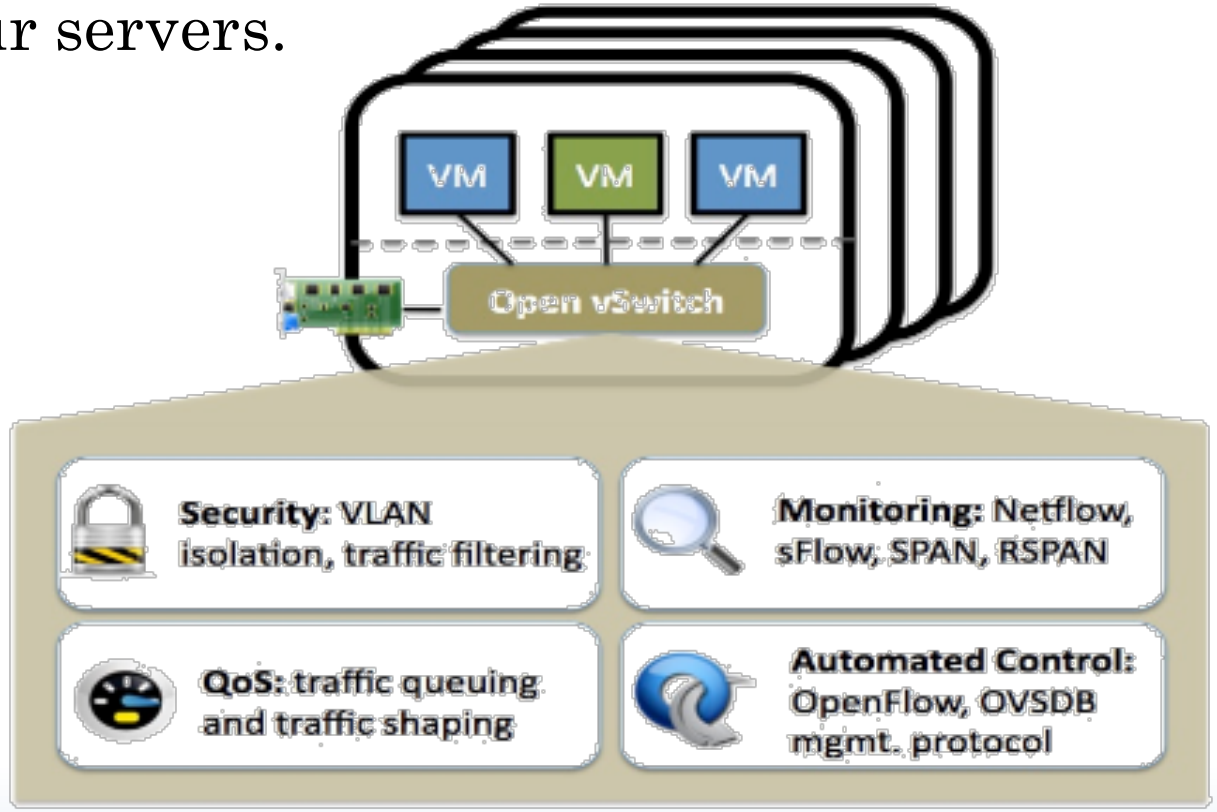


Open vSwitch (1/2)

- A flexible, multi-layer software network switch. Typically used in virtualization environments as the network switching component in the hypervisor.
- Maintains the logical state of a virtual machine's network connection across physical hosts when a virtual machine is migrated, and it can be managed and monitored by standard protocols such as: OpenFlow, NetFlow, sFlow, SPAN, RSPAN.

Open vSwitch (2/2)

- When it comes to virtualization, open vSwitch is attractive because it provides the ability for a single controller to manage your virtual network across all your servers.



InfiniBand Virtualization

- InfiniBand is a switched fabric communications link used in high-performance computing and enterprise data centers.
- It has two key features: low latency and high bandwidth
- Virtualization using InfiniBand brings big benefits to datacenters

Thanks!