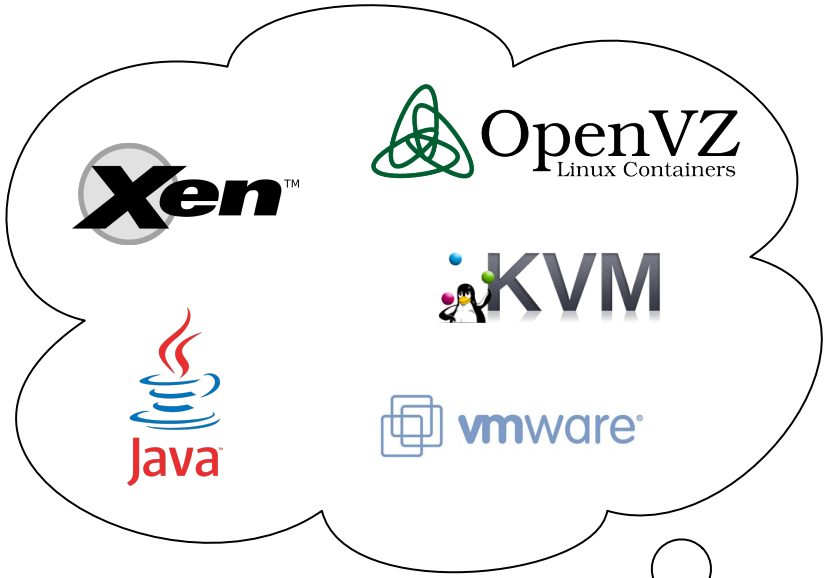
An Overview of Virtualization Technologies

Pierre Riteau

University of Rennes 1, IRISA
INRIA Rennes - Bretagne Atlantique

June 29, 2011 / Contrail Summer School 2011

Introduction



Virtualization
???

Outline

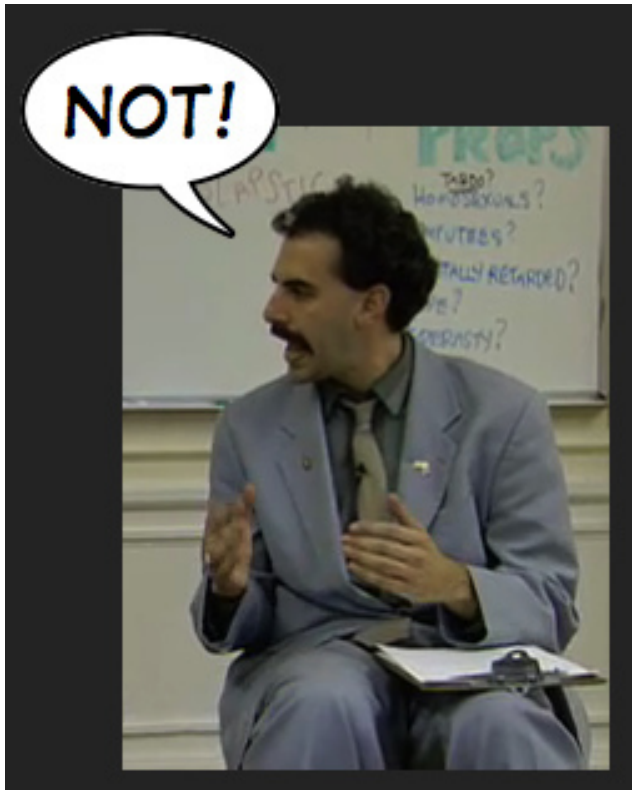
Outline

Virtualization vs Abstraction

Virtualization is abstraction.

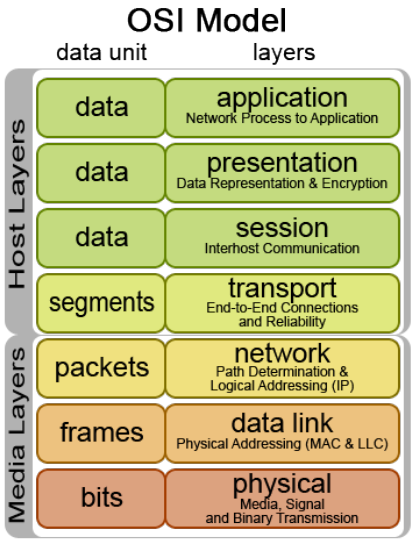
Virtualization vs Abstraction

~~Virtualization is abstraction.~~



Abstraction

- Abstraction → Offer a simplified interface
- Computing systems organized as layers of abstraction
  → each layer helps to simplify the system
- Example of abstractions
  - A file is an abstraction of disk storage
  - A TCP stream is an abstraction of network packets
    . . . which are abstraction of electrical signals



Virtualization

- Virtualization → Offer a different interface
- Virtualized interface is not necessarily simpler
- Can be applied to many types of resources
  - Compute (CPU)
  - Storage (disk)
  - Network
- Concept of virtual machine

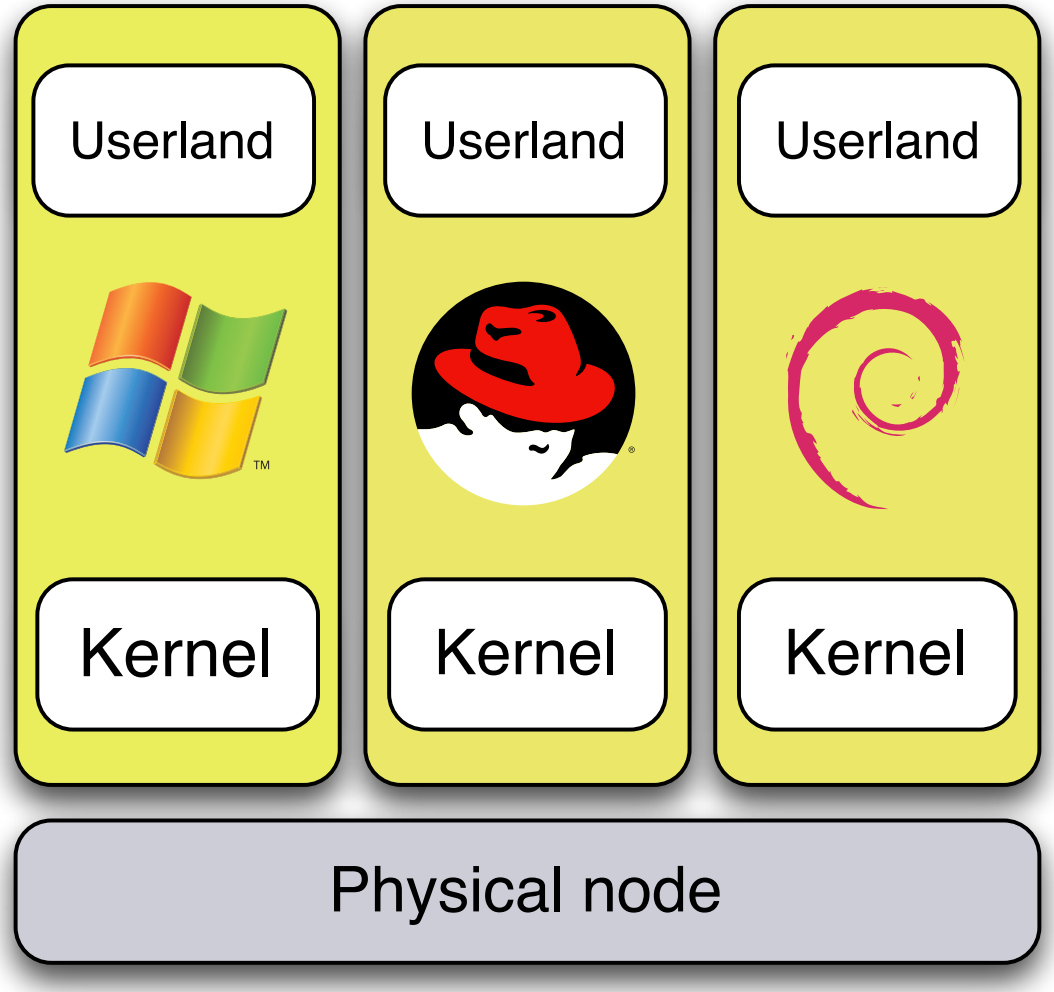
Different Types of Virtualization

- System-level virtualization
- Process-level Virtualization
- OS Virtualization

System-level Virtualization

- Emulates a computer similar to a real physical one
- With CPU(s), memory, disk(s), network interface(s), etc.
- The virtual machine runs a full OS
- Full Virtualization vs Paravirtualization
- Examples: VMware, Xen, KVM

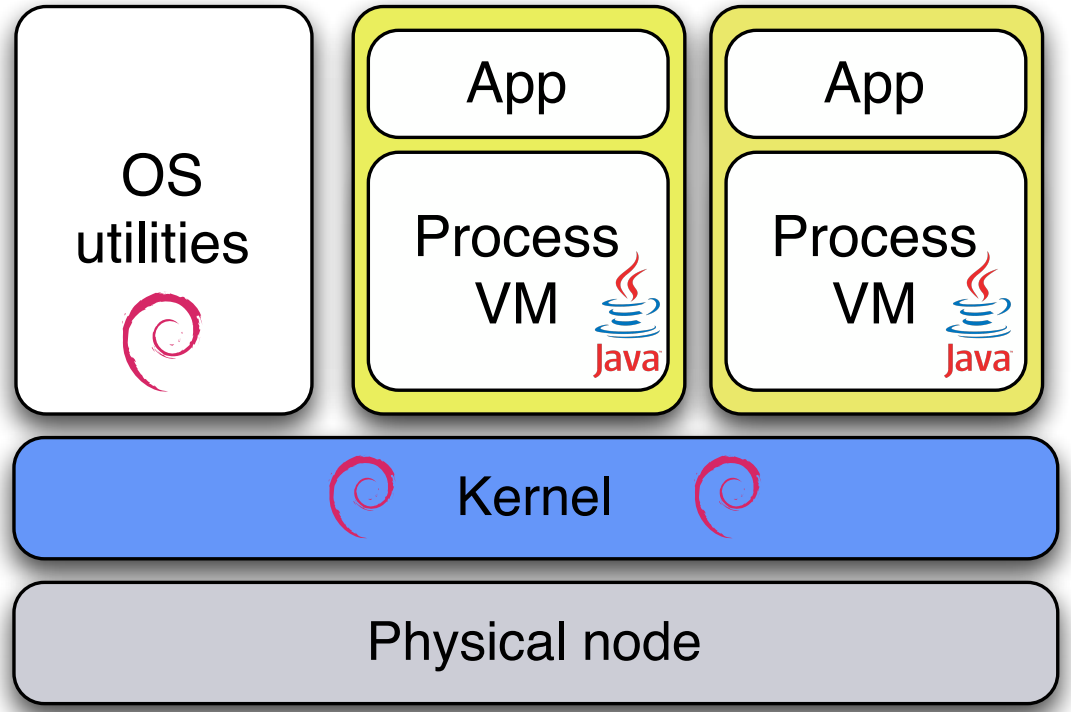
System-level Virtualization



Process-level Virtualization

- The virtual machine runs one application (one process)
- Application has to be written specifically for the VM
- Usually implemented on top of an operating system
- Example: Java Virtual Machine
- Advantage
  - Application is portable among all platforms supporting the VM
    $\rightarrow$ JVM on Windows, Linux, OS X, PDAs, phones ...
- Disadvantage
  - Legacy applications have to be rewritten for the VM

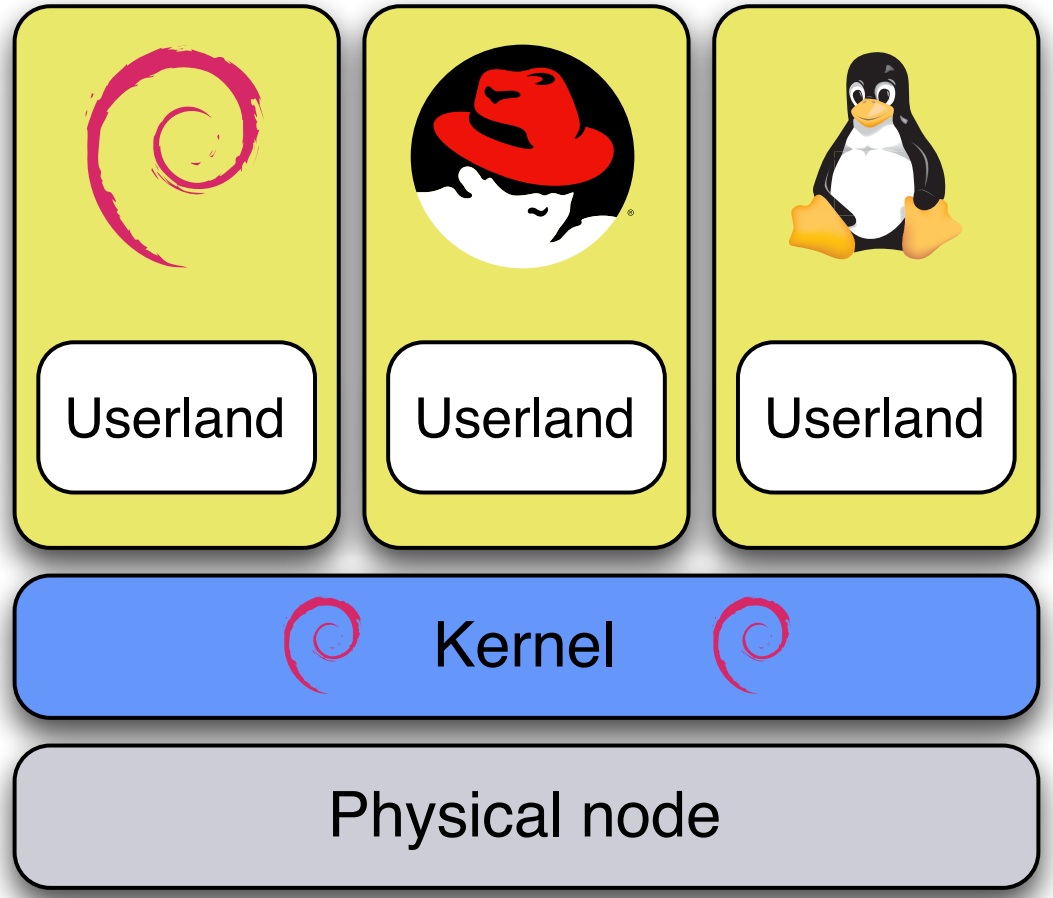
Process-level Virtualization



OS Virtualization

- The virtual machine runs a set of userland processes
- Userland domains are separated
- Kernel is the same for all userland domains
- Example: OpenVZ, Solaris zones, FreeBSD jails

OS Virtualization

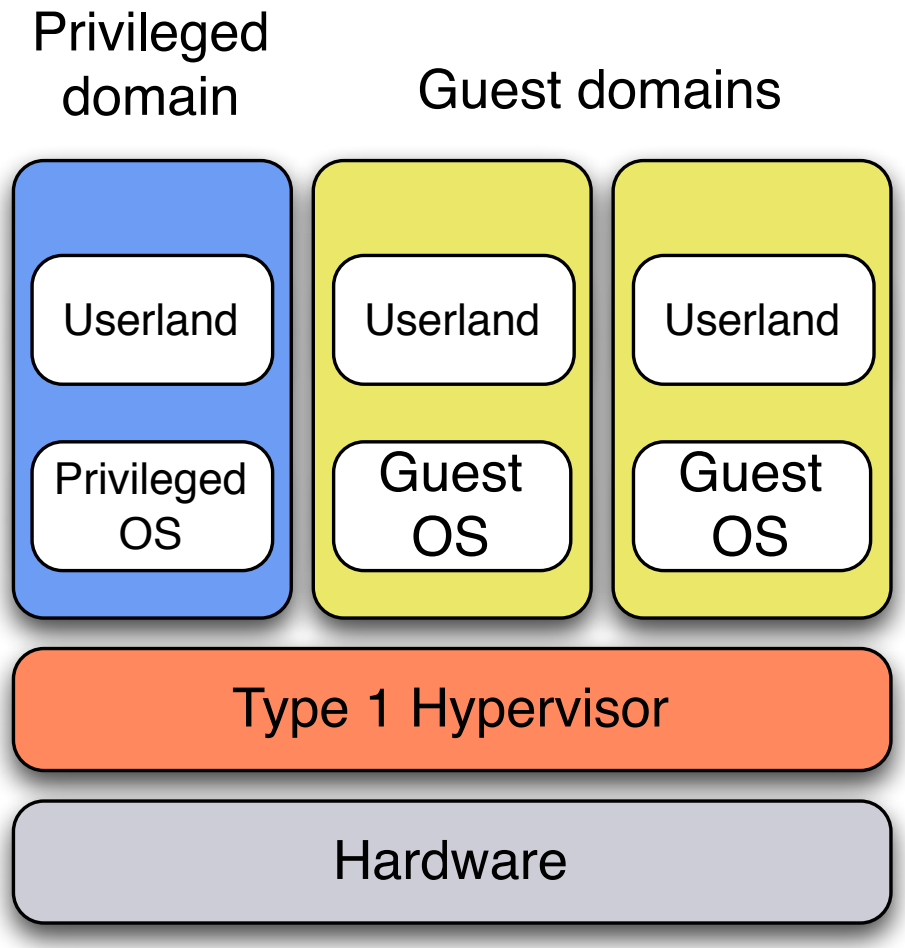


Outline

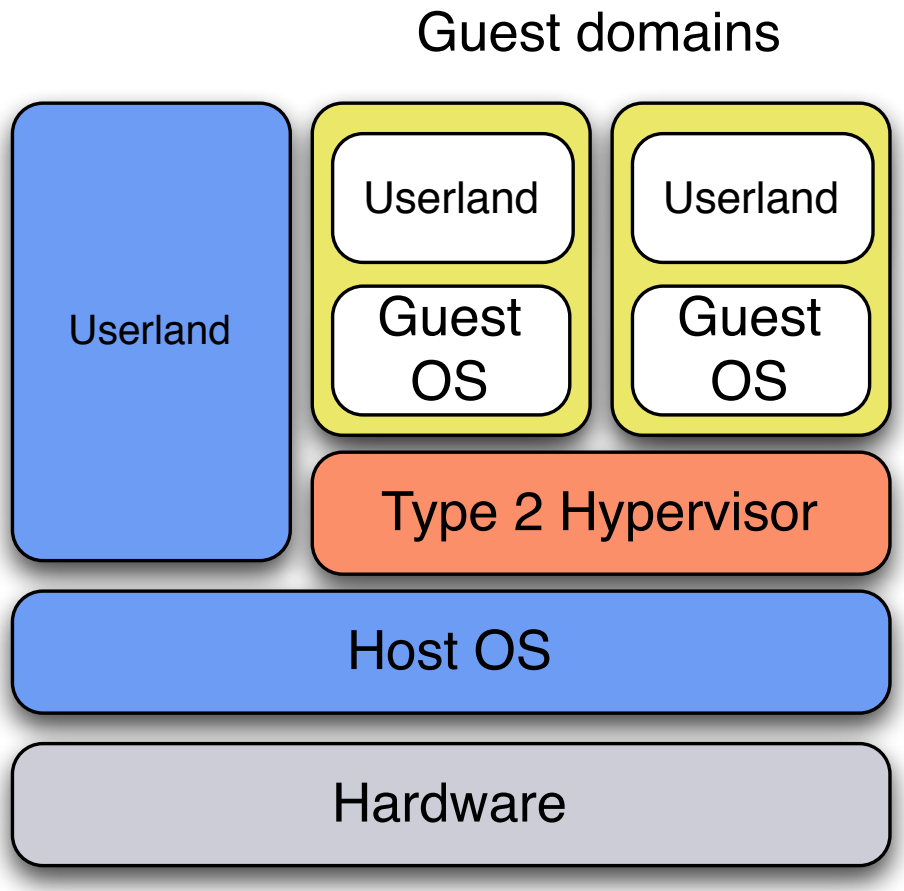
System-level Virtualization

- Virtual machines are managed by another software layer
- Hypervisor / Virtual Machine Manager (VMM)
- Can be of two different types
  - Type 1: native, runs directly on hardware
  - Type 2: hosted on top of another operating system
    $\rightarrow$ Host OS

Type 1 Hypervisor



Type 2 Hypervisor



Full Virtualization

- Full Virtualization → Run an OS without modification
- Initiated by IBM in 1967 with the CP-40 system
- Run natively most processor instructions
- Trap and emulate privileged instructions (I/O access, system CPU registers, . . .)
- Example
  - Virtual machine application runs a ADD instruction
    → runs directly on processor without hypervisor being involved
  - Virtual machine kernel reads the current system level
    → trapped by hypervisor and emulated to show a fake value

Paravirtualization

- Modify the guest OS to improve performance
- Make the guest OS aware that it is being virtualized
- Modify privileged instructions in the guest OS to avoid traps
- Replace by an interaction between the guest OS and the hypervisor interface
- Examples
  - Disco (Stanford University, 1997)
  - Xen (University of Cambridge, 2003)

Paravirtualized drivers

- Keep the guest OS unmodified ...
- ... but write drivers that know the system is virtualized
- ~~Emulation of a real device~~ → simple virtual device
- Examples
  - virtio in KVM for Linux guests
  - VMware Tools for Windows/Linux guests
- Used for I/O devices requiring high performance
  - Network I/O
  - Disk I/O

Problems with virtualizing the Intel x86

- Classical x86 architecture is not virtualizable
- Some privileged instructions don't generate traps
  $\rightarrow$ sensitive instructions
- Concept of ring levels
  - Normal system
    - OS runs in ring 0
    - applications in ring 3
  - Virtualized setting
    - hypervisor runs in ring 0
    - guest OS in ring 3

How to virtualize the Intel x86

- Binary translation recompilation of code
  $\rightarrow$ hypervisor analyzes guest code and replaces it with emulated code
- Paravirtualization
- Hardware support
  - creates ring -1 for hypervisor
  - guest OS can run in ring 0
  - AMD-V & VT-x

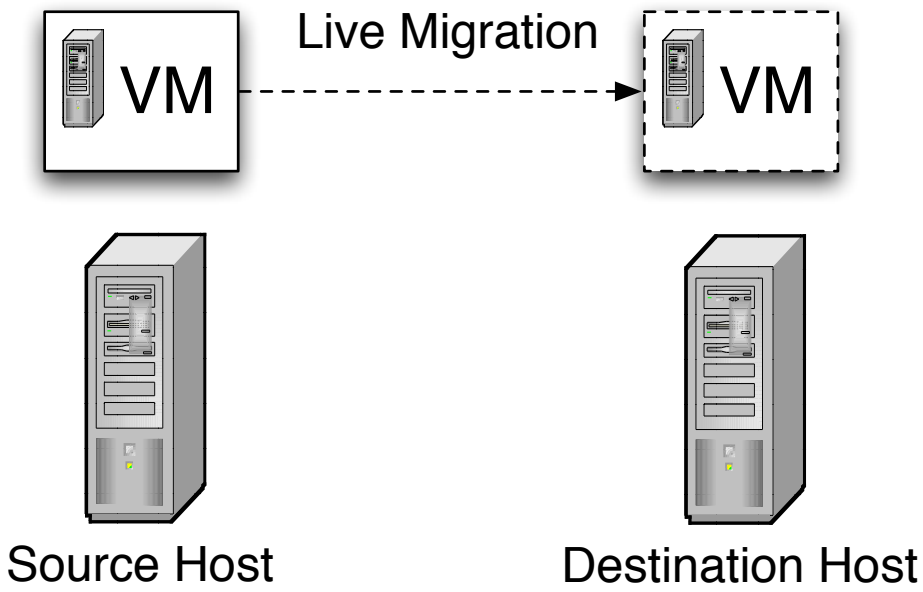
Outline

Live Migration of Processes

- Migration of processes has long been researched
- Offers many advantages
  - Load balancing
  - Power efficiency
  - Transparent infrastructure maintenance
- Problems
  - Complex implementations required to migrate all system resources
  - Residual dependencies

Live Migration of Virtual Machines

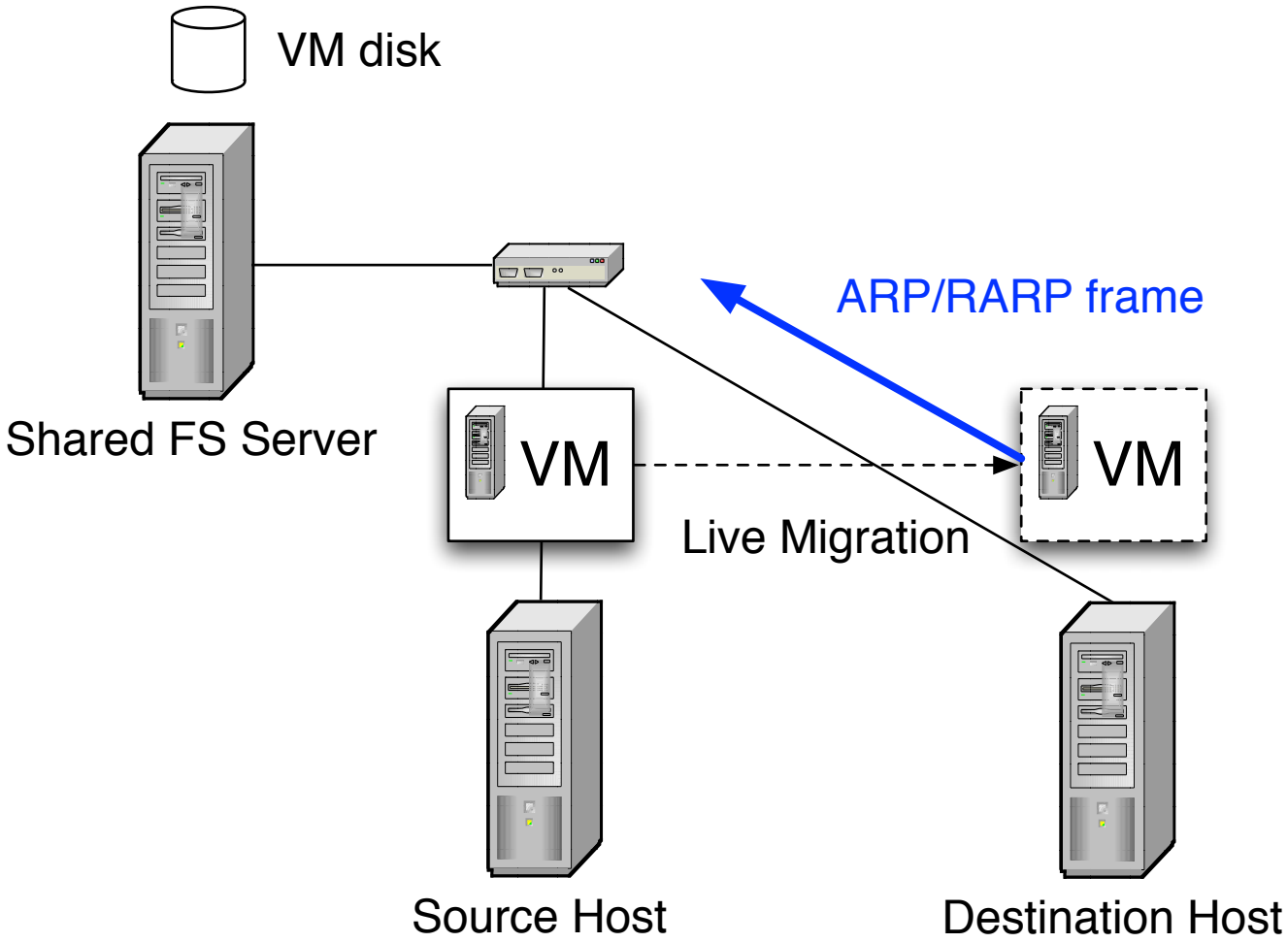
- Virtual machines provide complete encapsulation of
  - Applications
  - Libraries
  - Operating system
- Possible to serialize the state of a VM between physical hosts



Live Migration of VMs in LANs

- Transfer VM state from source host to destination host
- VM state
  - Processor state (CPU registers)
  - Device state (hardware registers)
  - Memory content
- What about storage and network resources?
- Shared storage (e.g. NFS) $\rightarrow$ no migration needed
- Network traffic redirected with gratuitous ARP/RARP frames

Live Migration of Virtual Machines



Pre-Copy Live Migration

- Traditional method used for migration of processes
- Iterative process
  - Copy all memory content to the destination host (while the VM continues running)
  - Do multiples iterations to copy modified memory pages during the previous period
  - When *enough* iterations have been done, stop the VM and
    - Copy the remaining modified memory pages
    - Copy the CPU and device state
  - Resume VM on destination host

Post-Copy Live Migration

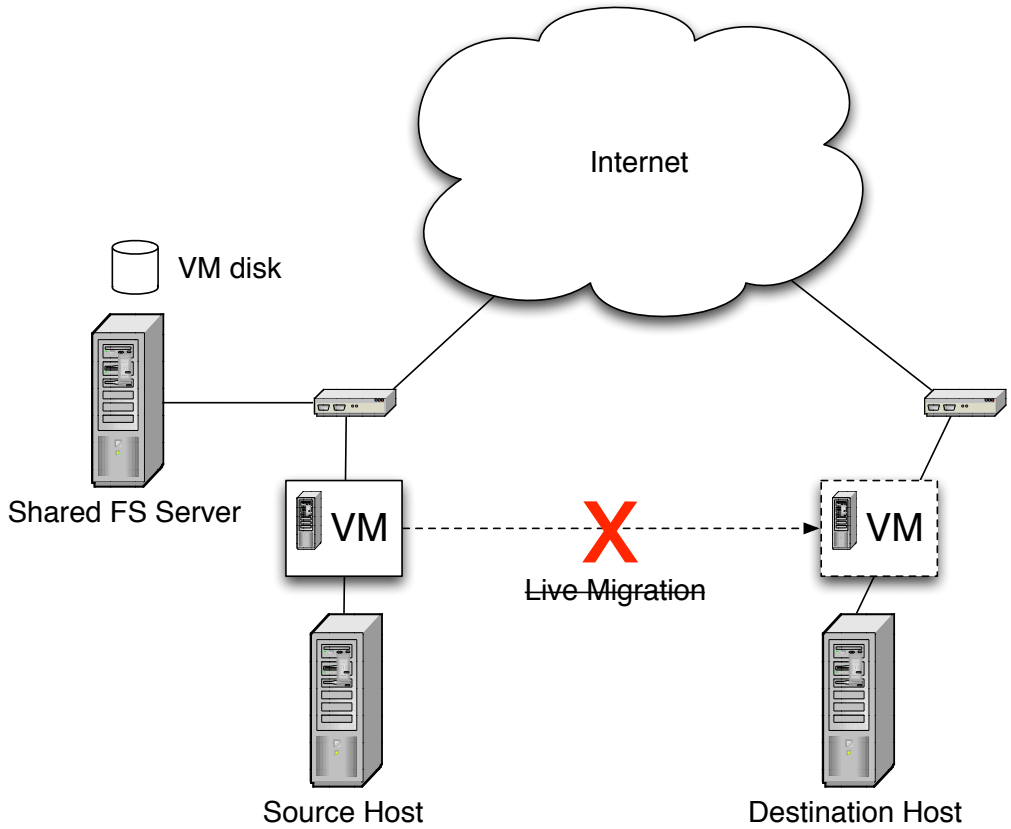
- Pre-copy: can present long downtime in the last phase
  - if the application modifies a large working set
  - if the available bandwidth is low
- Post-copy algorithm
  - Start by copying CPU and device state
  - Resume VM execution on the destination host
  - Fetch memory on demand when accessed
- Reduces downtime over pre-copy
- Can lower performance because of memory access latency

Trace & Replay Live Migration

- Use pre-copy as the basic migration algorithm
- Instead of sending modified memory pages $\rightarrow$ send external events of the VM to replay the modifications
- e.g., network packet received $\rightarrow$ modify network card registers
- Greatly reduces amount of data to send between hosts
- Problem: not working for SMP VMs as CPU synchronization would be too costly

Live Migration over Wide Area Networks

- Live migration between different infrastructures/data centers/clouds



Live Migration of Storage

- Need to replicate data to the destination infrastructure
- Mechanism similar to pre-copy live migration
- Copy the whole disk content
- Iteratively synchronize changes
- Examples: KVM migration, DRBD

Network Support for Live Migration

- Not possible to redirect traffic with ARP/RARP frames between different IP networks
- Solutions based on encapsulating traffic in a tunnel over WAN
- Or Mobile IPv6 mechanisms

Live Migration Optimizations

- Objective: Minimize downtime
- Means: Reduce amount of data to send
- Several approaches
  - Data Compression
  - Page Delta Transfer
  - Data Deduplication

Data Compression

- Compress memory pages sent over the network
- Trivial approach: compress zero'd memory pages
- General approach: use regular compression (gzip)
- More complicated: adaptive memory compression

Page Delta Transfer

- Memory pages are 4 KB on x86
- Modify 1 byte in the page $\rightarrow$ transfer 4 KB
- Delta transfer mechanism:
  - Keep copy of original page
  - Computer differences between original and new page
  - Send diff instead of full content

Data Deduplication

- VMs can contain identical data in multiple memory pages
- Deduplication retains only one unique copy of each memory page
- Duplicate detection based on fast hash algorithm + full data comparison in case of match

Memory Management

- Virtualization properties
  - Multiplexing of several guest OS
  - Isolation

  $\rightarrow$ Consolidation: running multiple systems on one physical host
- Multiple guest OS compete for memory of host

Ballooning

- Paravirtualized driver runs in the VM
- Responds to hypervisor requests for memory
- Inflate/deflate its memory allocation
- Memory is given back to the hypervisor
  $\rightarrow$ Can be used by other VMs afterwards

Page Sharing

- Typical to run multiple times the OS on one host
- Each OS will have its own copy of code and data from
  - kernel
  - libraries
  - applications
- Detect identical pages in multiples VMs of the same host
- Merge identical pages to reduce memory consumption
- Mark shared pages as read-only to do copy on write

Snapshots

- Snapshot = save full state (memory + storage) of a VM
- Allows to return to a previous state
- Some scenarios
  - Wrong configuration change → rollback to snapshot
  - Failed OS update → rollback to snapshot
- Copy-On-Write (COW) for storing changes
  → Store only modifications made on writes

Summary

- Virtualization offers different interfaces ($\neq$ abstration).
- Concept of virtual machine as an execution platform
- Different kinds of virtual machines
- System-level virtualization allows to execute regular OS
- Features offered by system-level virtualization
  - Live migration in LAN or WAN
  - Memory balancing/sharing
  - Snapshots