Lecture 8.1 – Virtualisation

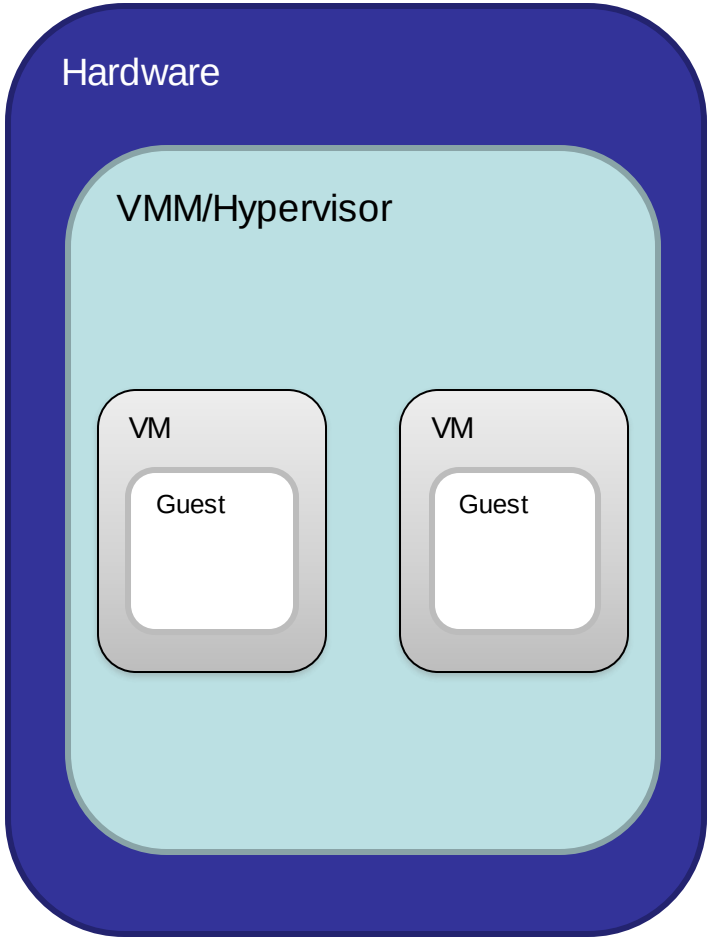
Professor Richard O. Sinnott
Director, eResearch
University of Melbourne
rsinnott@unimelb.edu.au

Overview

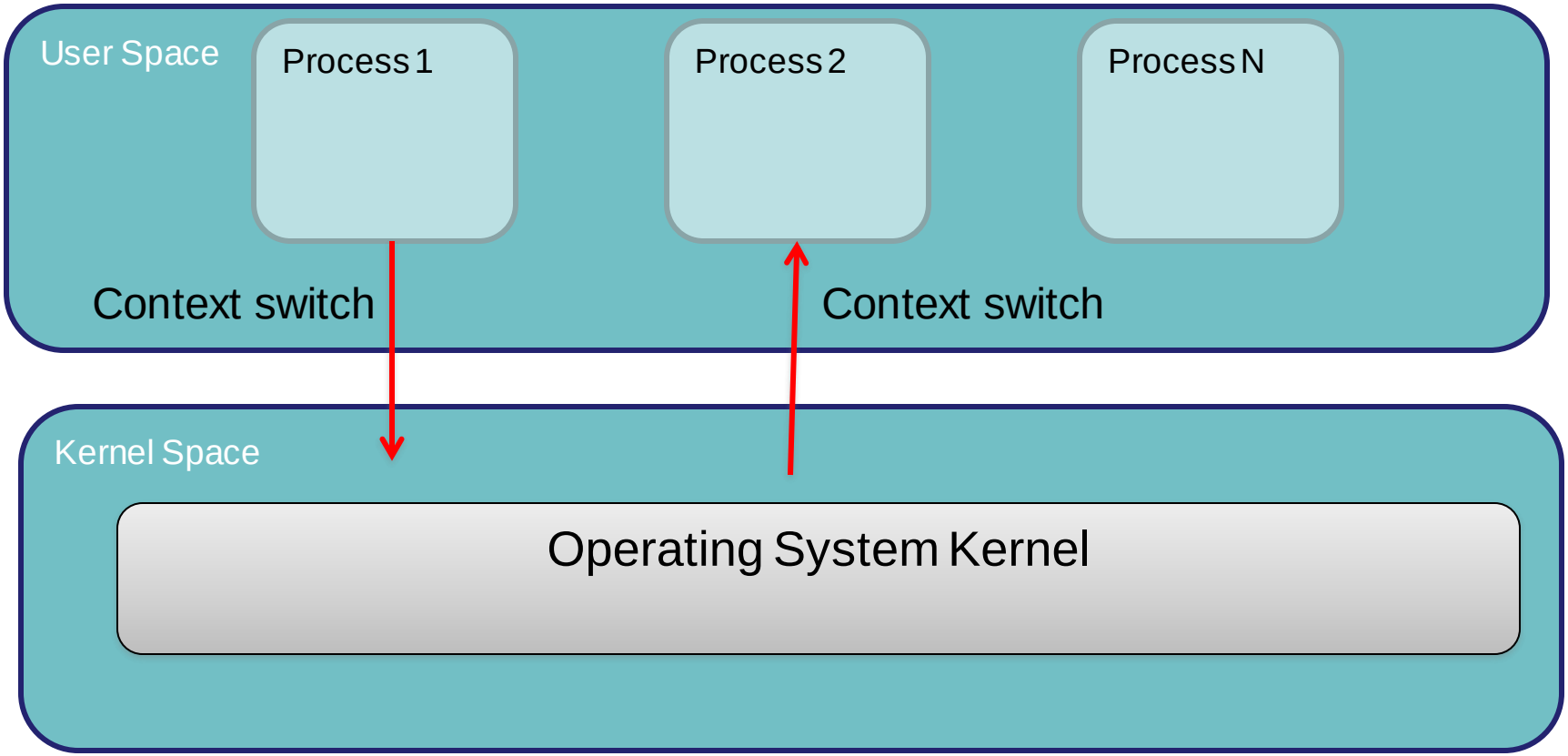
- Virtualisation
  - What happens in a VM?
  - Motivation & History
  - What are the requirements for virtualisation?
  - Virtualisation approaches
  - Memory management
  - Live migration

Terminology

- Virtual Machine Monitor/Hypervisor: The virtualisation layer between the underlying hardware (e.g. the physical server) and the virtual machines and guest operating systems it supports.
  - The environment of the VM should appear to be the same as the physical machine
  - Minor decrease in performance only
  - Appears as though in control of system resources
- Virtual Machine: A representation of a real machine using hardware/software that can host a guest operating system
- Guest Operating System: An operating system that runs in a virtual machine environment that would otherwise run directly on a separate physical system

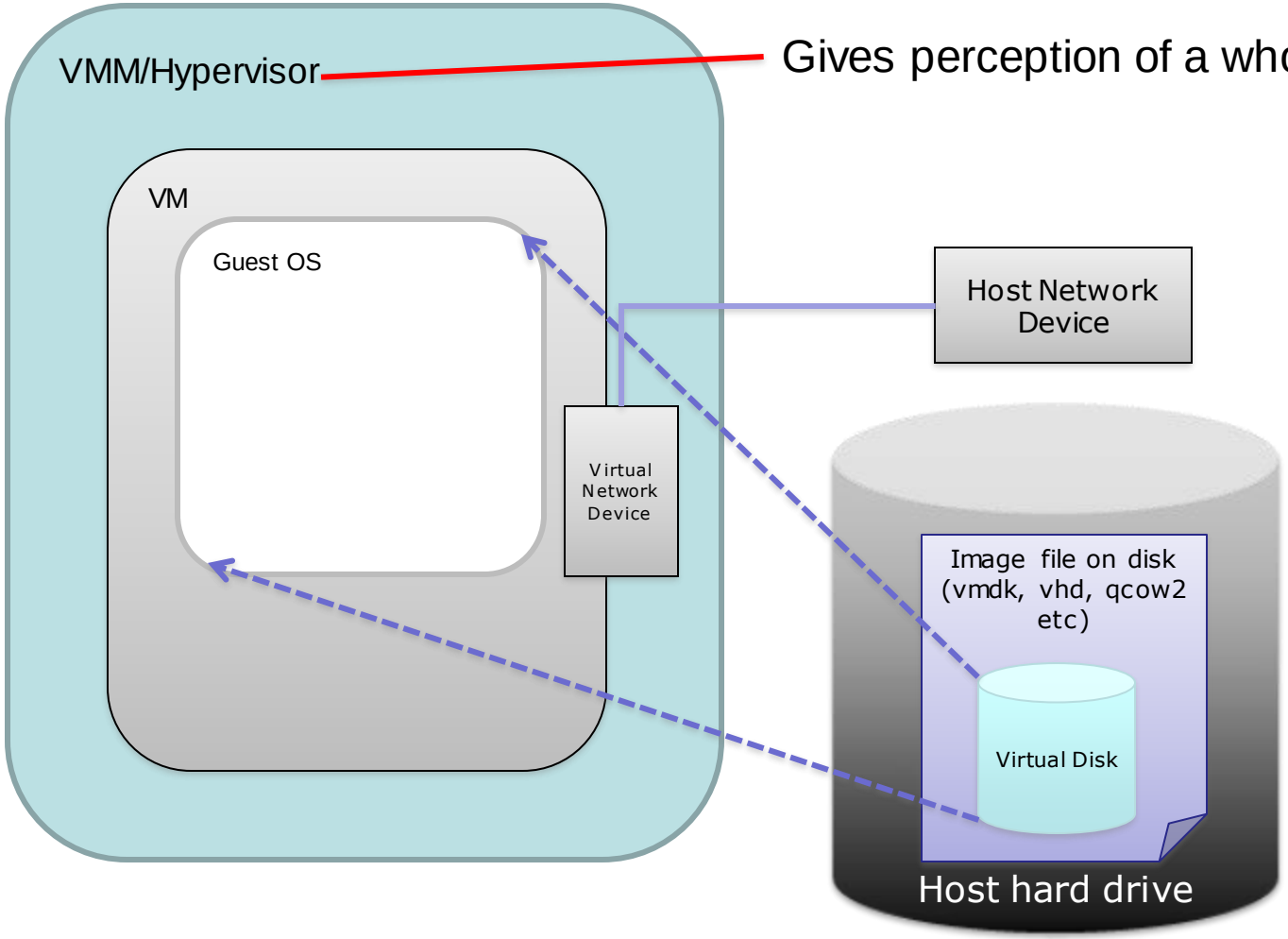


Recap on Kernel-User mode separation



- Processes run in lower privileged (user) mode
- OS Kernel runs in privileged Kernel mode
- OS typically virtualises memory, CPU, disk etc giving appearance of complete access to CPU/memory/disk to application processes
  - Each process has illusion of access to some/all of the memory or the CPU (but actually shared across multiple processes)
- Context switches can catch (trap) "sensitive" calls
  - e.g. add two numbers vs change bios settings;
  - Sensitive calls -> instruction sets are typically device specific, e.g. ARM vs x86 vs …

What Happens in a VM?



VHD (Virtual Hard Disk represents a virtual hard disk drive (HDD). May contain what is found on a physical hard disk, such as disk partitions and a file system, which in turn can contain files and folders.

VMDK (Virtual Machine Disk) describes containers for virtual hard disk drives to be used in virtual machines like VMware.

qcow2 (QEMU Copy On Write) file format for disk image files used by QEMU. It uses a disk storage optimization strategy that delays allocation of storage until it is actually needed.

- Guest OS apps “think” they write to hard disk but translated to virtualised host hard drive by VMM
- Which one is determined by image that is launched

Motivation

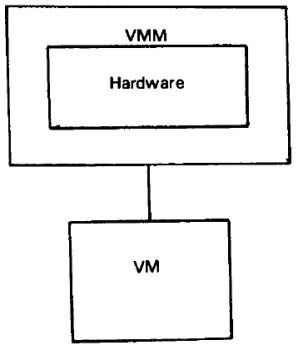
- Server Consolidation
  - Increased utilisation
  - Reduced energy consumption
- Personal virtual machines can be created on demand
  - No hardware purchase needed
  - Public cloud computing
- Security/Isolation
  - Share a single machine with multiple users
- Hardware independence
  - Relocate to different hardware

History

- Virtualisation concept goes back to 1960s
- IBM System/370
- System/370 featured hardware support for *interpretive* execution
- Formal requirements laid down by Popek and Goldberg (1974)
- Why then?
  - Mainframes needed ability to run multiple kinds of applications
    - Hardware was very expensive "back in the day"

Origins - Principles

Fig. 1. The virtual machine monitor.



"an *efficient, isolated duplicate* of the real machine"

- Properties of interest
  - Fidelity: Software on the VMM executes behaviour identical to that demonstrated when running on the machine directly, barring timing effects
  - Performance: An overwhelming majority of guest instructions executed by hardware without VMM intervention
  - Safety: The VMM manages all hardware resources

Formal Requirements for Virtualizable Third Generation Architectures

Gerald J. Popek
University of California, Los Angeles
and
Robert P. Goldberg
Honeywell Information Systems and
Harvard University

Virtual machine systems have been implemented on a limited number of third generation computer systems, e.g. CP-67 on the IBM 360/67. From previous empirical studies, it is known that certain third generation computer systems, e.g. the DEC PDP-10, cannot support a virtual machine system. In this paper, model of a third-generation-like computer system is developed. Formal techniques are used to derive precise sufficient conditions to test whether such an architecture can support virtual machines.

Communications of the ACM, vol 17, no 7, 1974, pp.412-421

Classification of Instructions

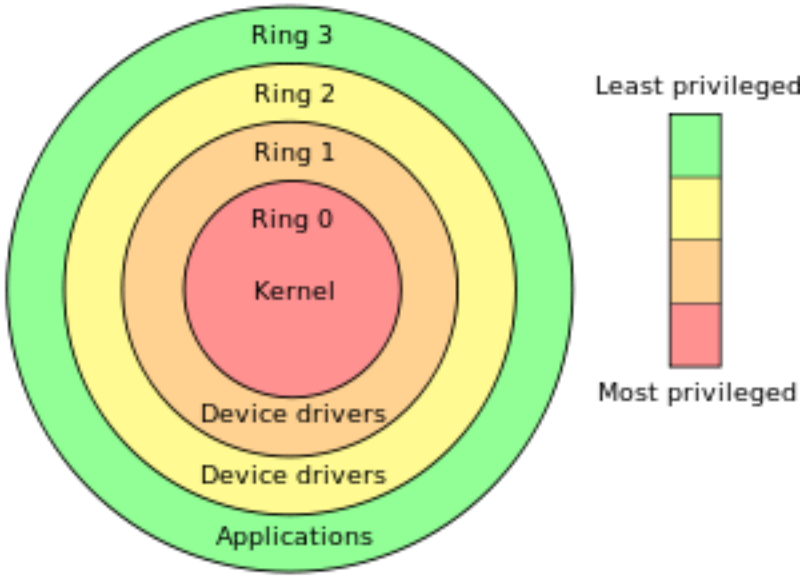
- Privileged Instructions: instructions that trap if the processor is in user mode and do not trap in kernel mode
- Sensitive Instructions: instructions whose behaviour depends on the mode or configuration of the hardware
  - Different behaviours depending on whether in user or kernel mode
    - e.g. POPF interrupt (for interrupt flag handling)
- Innocuous Instructions: instructions that are neither privileged nor sensitive
  - Read data, add numbers etc

Origins - Principles

- Theorem (Popek and Goldberg)
  - For any conventional third generation computer, a virtual machine monitor may be constructed if the set of sensitive instructions for that computer is a subset of the set of privileged instructions
    - i.e. have to be trappable

Example of Privilege Rings

- Ring 0: Typically hardware interactions
- Ring 1: Typically device drivers
- Specific gates between Rings (not ad hoc)
- Allows to ensure for example that spyware can't turn on web cam or recording device etc



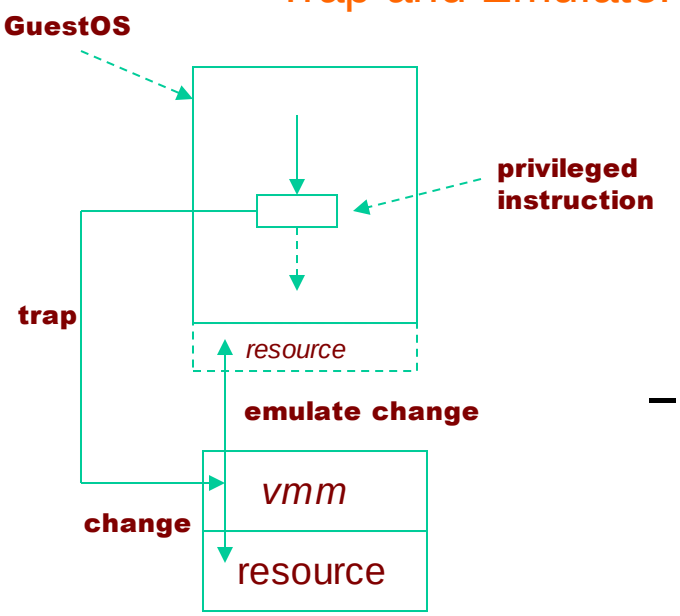
- Significance
  - The IA-32/x86 architecture was not originally virtualisable

x86 Virtualisability

- x86 architecture was historically not virtualisable, due to sensitive instructions that could not be trapped, e.g. instructions such as:
  - *SMSW* – storing machine status word
  - *SGDT, SLDT* – store global/local descriptor table register
  - *POPF* – interrupt flag (user/kernel mode)
    - Robin and Irvine, Analysis of an Intel Pentium's Ability to Support a Secure Virtual Machine Monitor, Usenix, 2000
- Intel and AMD introduced extensions to make x86 virtualisable
  - AMD SVM (Secure Virtual Machine)
  - Intel VT (Virtualisation Technology)
- These two are very similar, but use slightly different machine instructions

Typical Virtualisation Strategy

Trap and Emulate!



VMM needs to support:

- De-privileging
  - VMM emulates the effect on system/hardware resources of privileged instructions whose execution traps into the VMM
    - aka trap-and-emulate
  - Typically achieved by running GuestOS at a lower hardware priority level than the VMM
  - Problematic on some architectures where privileged instructions do not trap when executed at de-privileged level
- Primary/shadow structures
  - VMM maintains "shadow" copies of critical structures whose "primary" versions are manipulated by the GuestOS, e.g. memory page tables
  - Primary copies needed to insure correct versions are visible to GuestOS
- Memory traces
  - Controlling access to memory so that the shadow and primary structure remain coherent
  - Common strategy: write-protect primary copies so that update operations cause page faults which can be caught, interpreted, and addressed
    - Someones app/code doesn't crash the server you are using!!!

Major VMM and Hypervisor Providers

VMM Provider	Host CPU	Guest CPU	Host OS	Guest OS	VM Architecture
VMWare Workstation	x86, x86-64	x86, x86-64	Windows, Linux	Windows, Linux, Solaris, FreeBSD, OS/2	Full Virtualization
VMWare ESX Server	x86, x86-64	x86, x86-64	No Host OS	Same as VMWare workstation	Baremetal hypervisor
Xen	x86, x86-64, IA-64	x86, x86-64, IA-64	NetBSD, Linux, Solaris	Windows, Linux, Solaris, FreeBSD, OS/2, NetBSD	Para-virtualisation
KVM			Linux	Linux, Windows, FreeBSD, Solaris	Hardware virtualisation

Aspects of VMMs

- Full virtualisation – allow an unmodified guest OS to run in isolation by simulating full hardware (e.g. VMWare)
  - Guest OS has no idea it is not on physical machine

vs

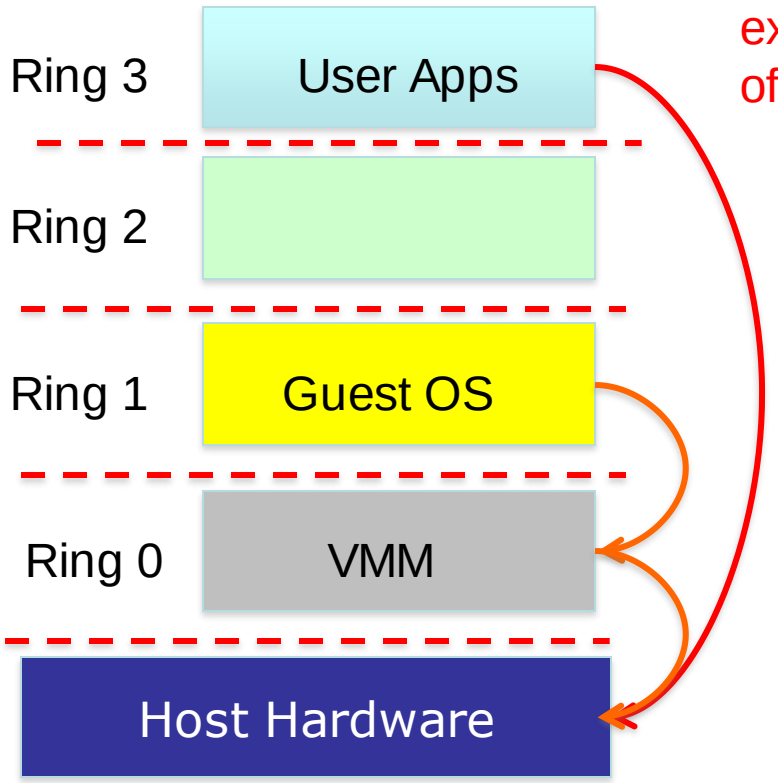
- Para-virtualisation – VMM/Hypervisor exposes special interface to guest OS for better performance. Requires a modified/hypervisor-aware Guest OS (e.g. Xen)
  - Can optimise systems to use this interface since not all instructions need to be trapped/dealt with

Full Virtualisation

- Advantages
  - Guest is unaware it is executing within a VM
  - Guest OS need not be modified
  - No hardware or OS assistance required
  - Can run legacy OS
- Disadvantages
  - can be less efficient

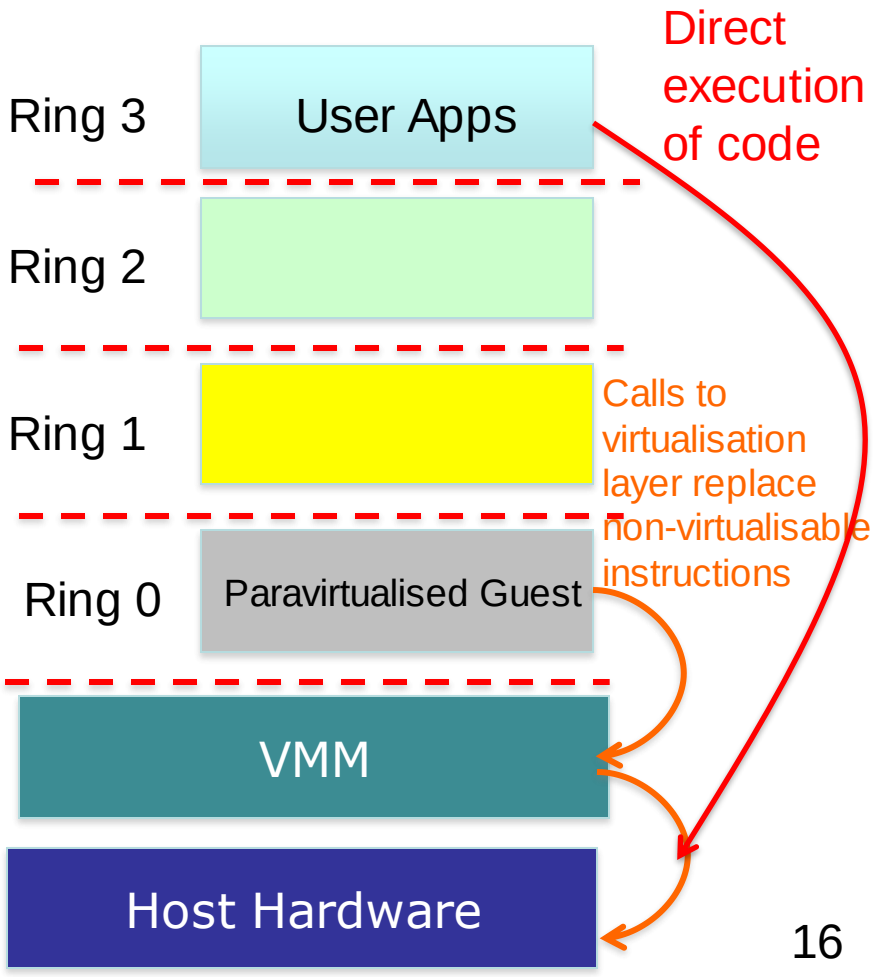
User/kernel split typically
- VMM run Ring 0
- Apps run in Ring 3

Virtualisation (Guest OS) uses extra rings; VMM traps privileged instructions and translates to hardware specific instructions



Para-Virtualisation

- Advantages
  - Lower virtualisation overheads, so better performance, e.g. Xen
- Disadvantages
  - Need to modify guest OS
    - Can't run arbitrary OS!
  - Less portable
  - Less compatibility



Aspects of VMMs (contd…)

- Hardware-assisted virtualisation – Hardware provides architectural support for running a Hypervisor (e.g. KVM)
  - New processors typically have this
  - Requires that all sensitive instructions trappable

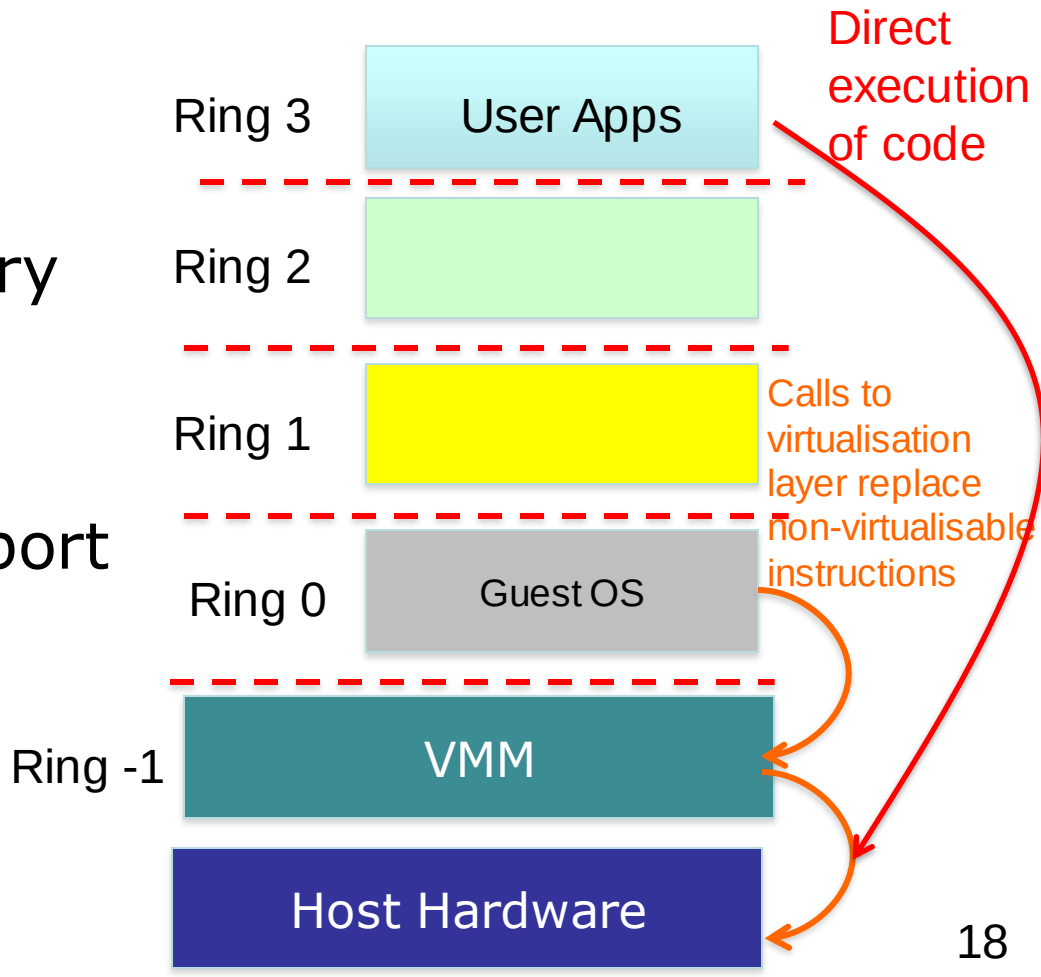
vs

- Binary Translation – Trap and execute occurs by scanning guest instruction stream and replacing sensitive instructions with emulated code (e.g. VMWare)
  - Don’t need hardware support, but can be much harder to achieve
    - Rarely ever 1:1 mapping between instruction sets

Hardware-assisted virtualisation

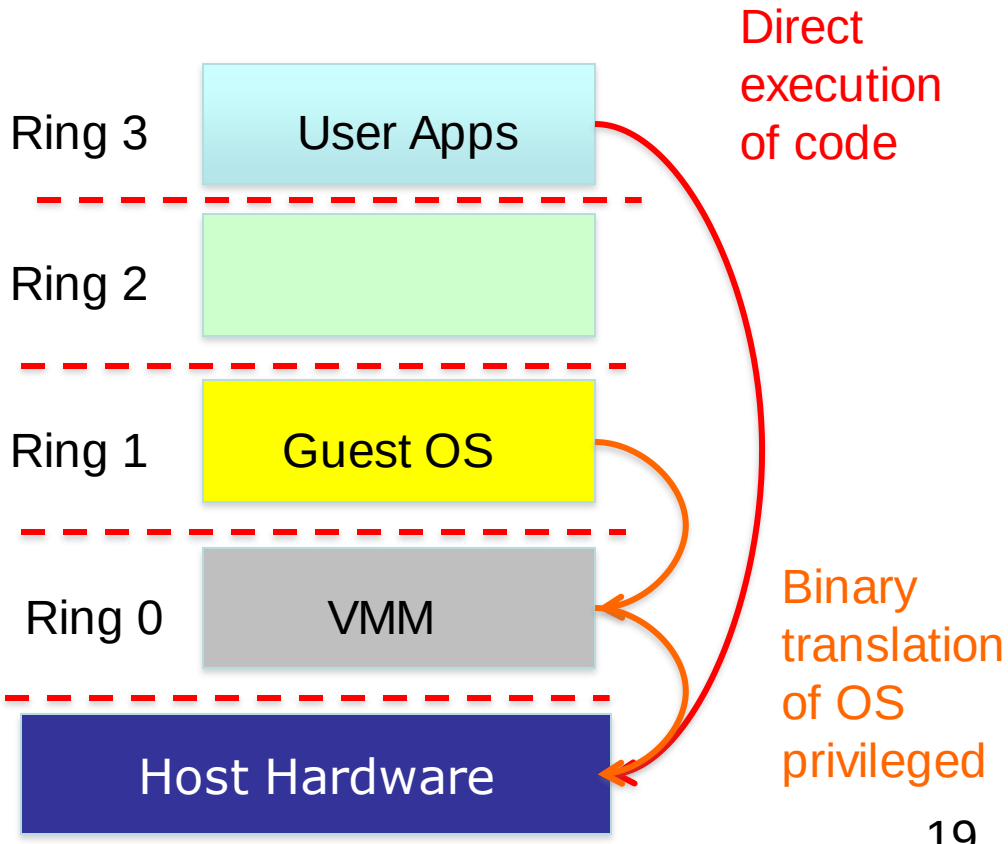
- Advantages
  - Good performance
  - Easier to implement
  - Advanced implementation supports hardware assisted DMA, memory virtualisation, …
- Disadvantages
  - Needs hardware support

New Ring -1 supported
Page tables, virtual memory mgt, direct memory access for high speed reads etc



Binary Translation

- Advantages
  - Guest OS need not be modified
  - No hardware or OS assistance required
  - Can run legacy OS
- Disadvantages
  - Overheads
  - Complicated
  - Need to replace instructions "on-the-fly"
  - Library support to help this, e.g. vCUDA



Aspects of VMMs (contd…)

- Bare Metal Hypervisor – VMM runs directly on actual hardware (e.g. VMWare ESX Server)
  - Boots up and runs on actual physical machine
  - VMM has to support device drivers, all HW mgt

vs

- Hosted Virtualisation – VMM runs on top of another operating system (E.g. VMWare Workstation,…)

Operating System Level Virtualisation

- *Lightweight* VMs
- Instead of whole-system virtualisation, the OS creates mini-containers
  - A subset of the OS is often good enough for many use cases
    - Can't use for running Windows on Linux etc, but often not needed!
  - Akin to an advanced version of "*chroot*"
    - operation that changes apparent root directory for current running process and subprocesses. Program run in such a modified environment cannot access files and commands outside that environmental directory tree. Aka a **chroot** jail
- Examples:
  - LXC, Docker, OpenVZ, FreeBSD Jails etc

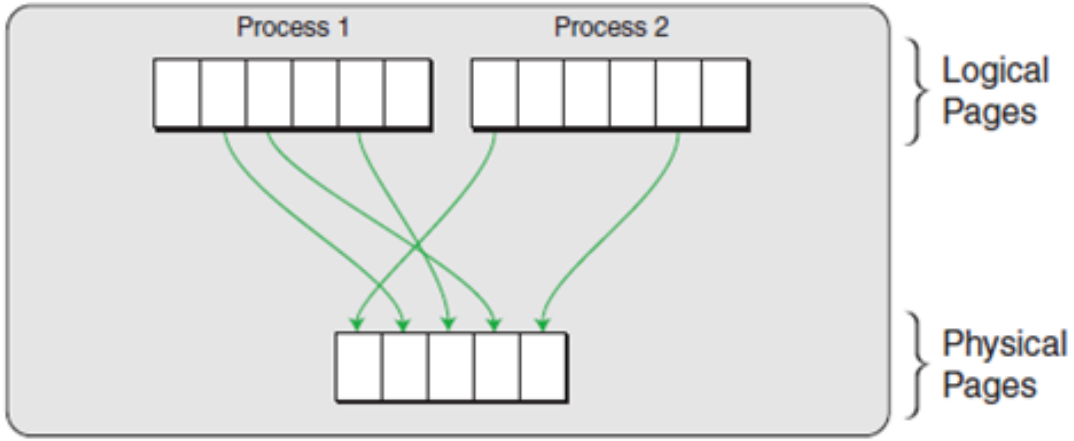
Operating System Level Virtualisation

- Advantages
  - Lightweight
  - Many more VMs on same hardware
  - Can be used to package applications and all OS dependencies into container

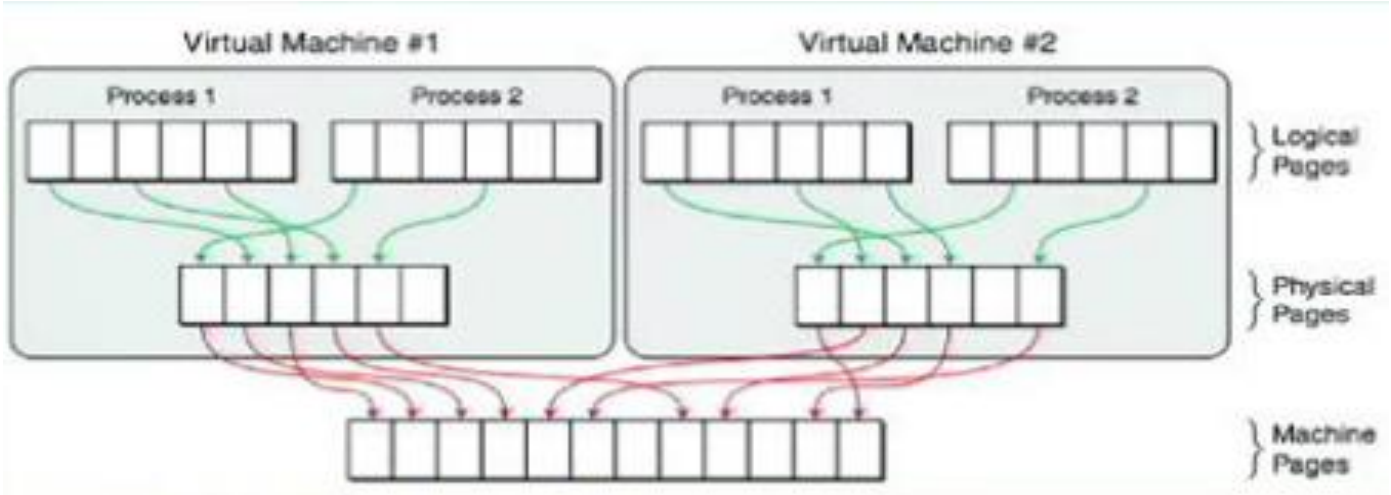
- Disadvantages
  - Can only run apps designed for the same OS
  - Cannot host a different guest OS
  - Can only use native file systems
  - Uses same resources as other containers

Memory Virtualisation

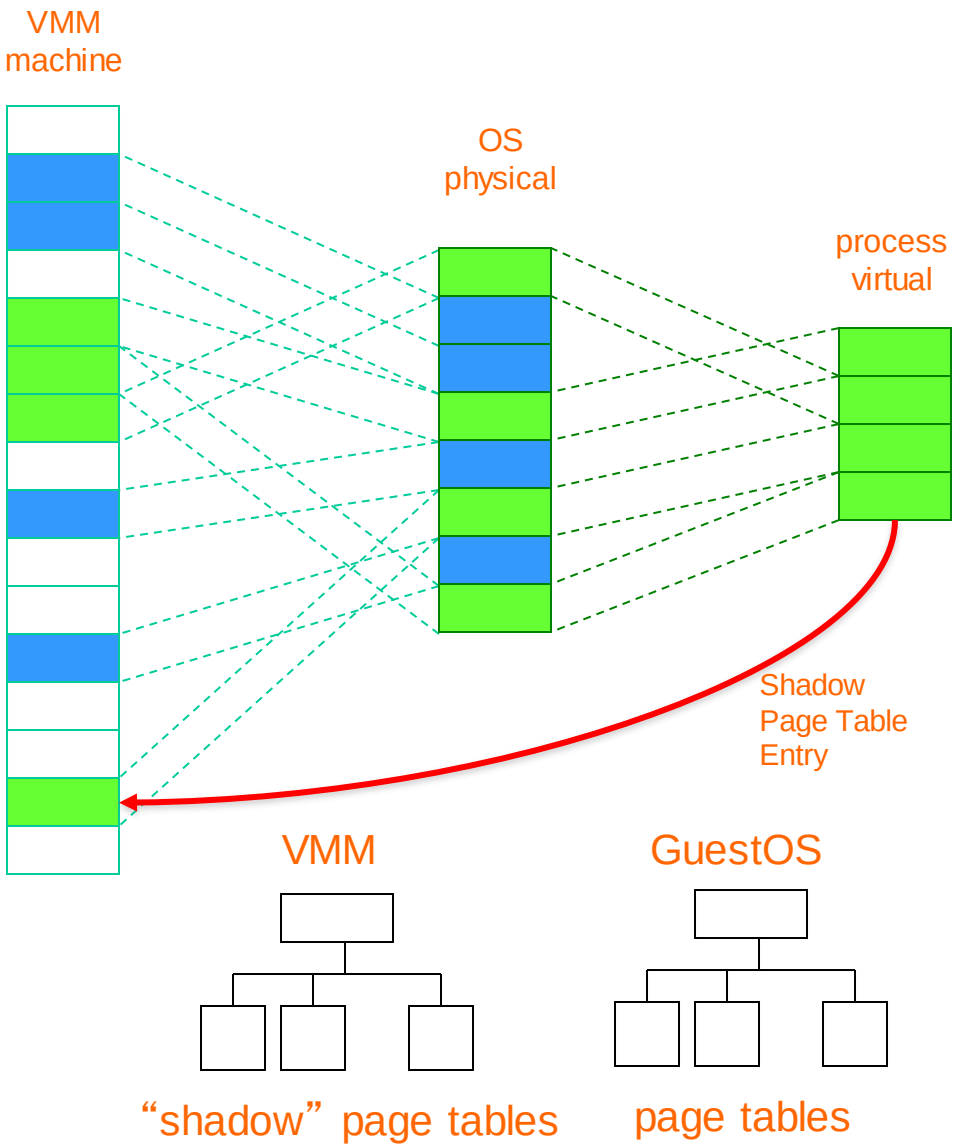
- Conventionally page tables store the logical page number -> physical page number mappings
  - Seems like more memory than actually have



- What happens in a VM?

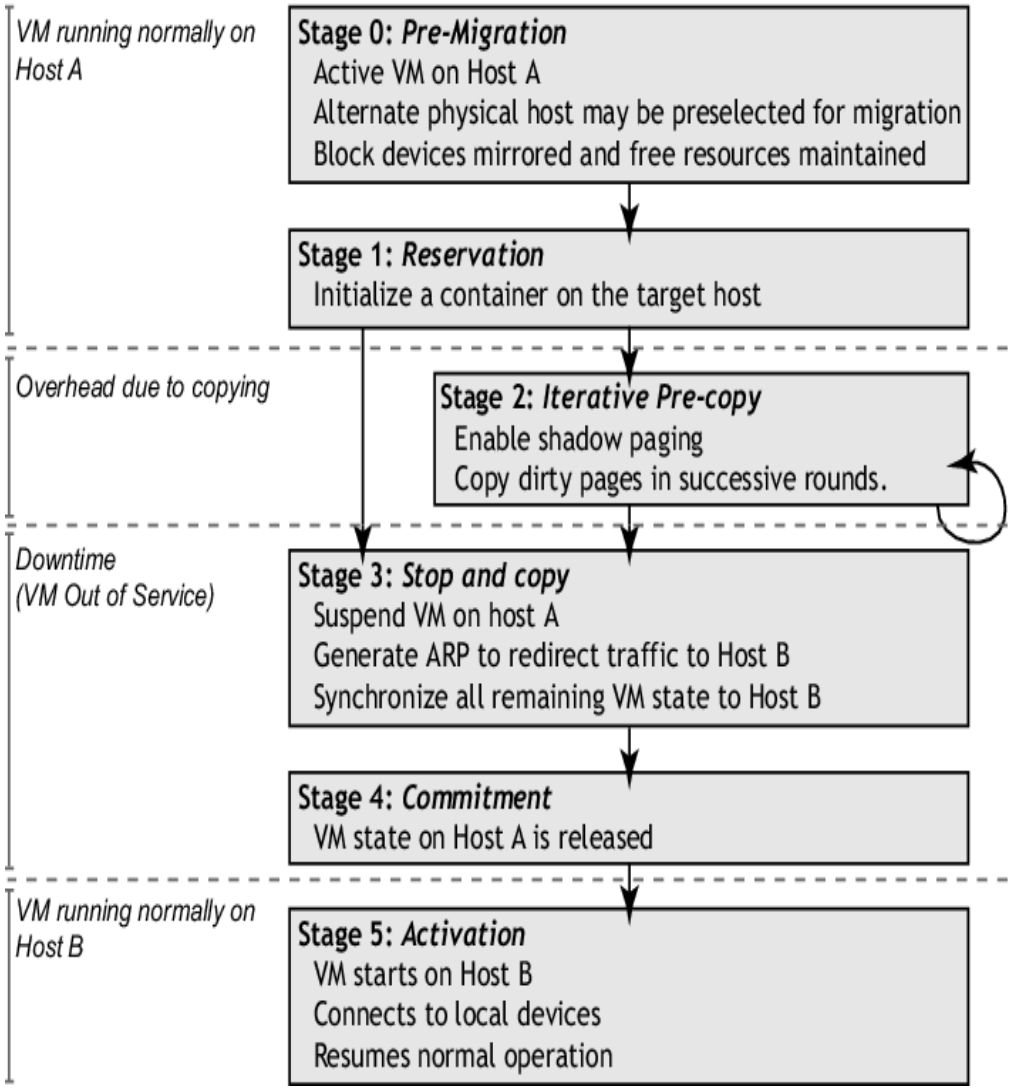


Shadow Page Tables



- VMM maintains shadow page tables in lock-step with the page tables
- Adds additional management overhead
- Hardware performs guest -> physical and physical -> machine translation

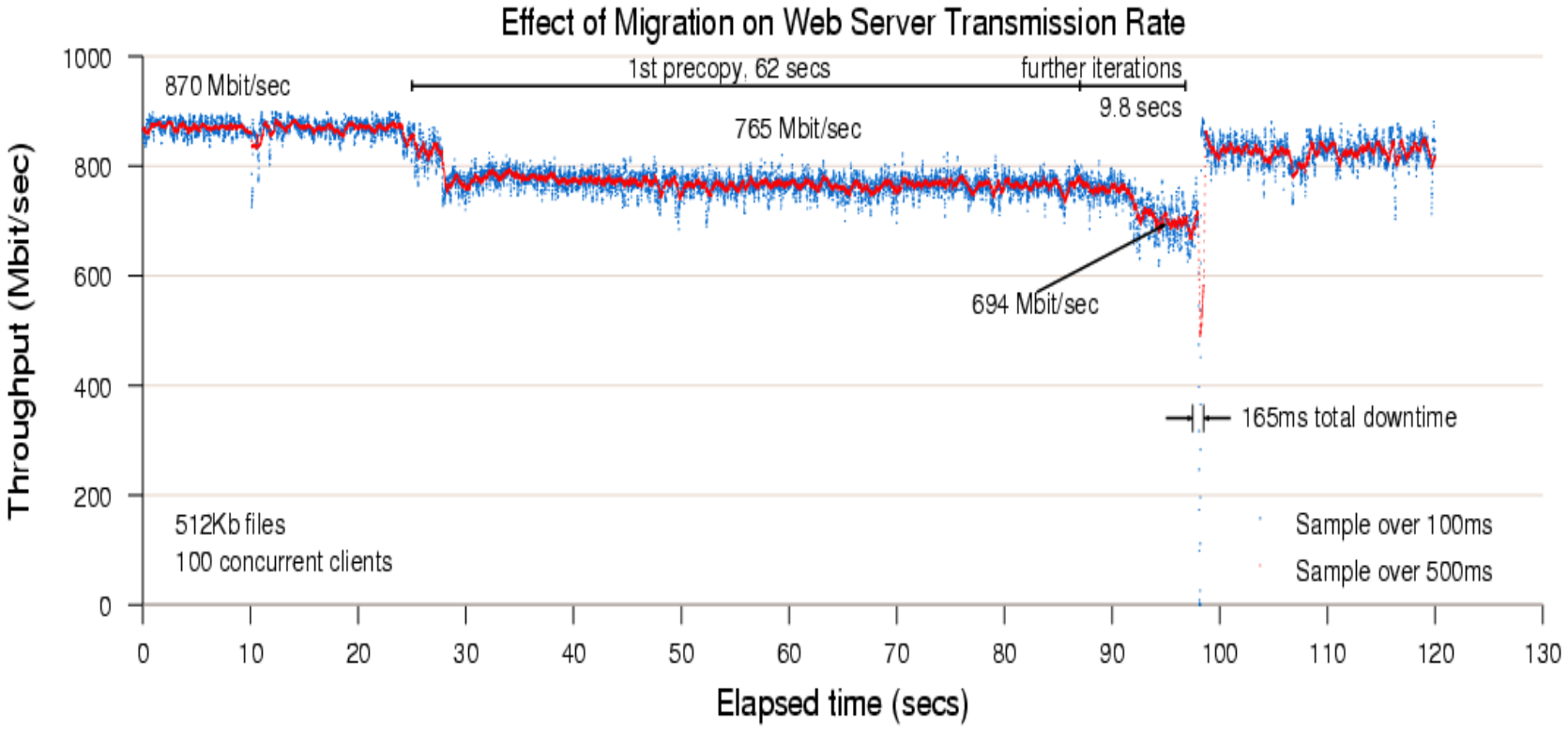
Live Migration from Virtualisation Perspective



Source: Clark et al, Live migration of virtual machines, NSDI 2005

Clark et al. Live migration of virtual machines, NSDI 2005.

Effects of Live Migration



Source: Clark et al, Live migration of virtual machines, NSDI 2005

References

- Hwang, Dongarra & Fox, 2011. Distributed and Cloud Computing, 1st Edition. Elsevier.
- Rosenblum M & Garfinkel T. 2005. Virtual machine monitors: Current technology and future trends, in IEEE Computer.
- Clark et al, 2005. Live migration of virtual machines. In Proceedings of the 2nd ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI). pp. 273-286
- K. Adams, O. Agesen, 2006. A Comparison of Software and Hardware Techniques for x86 Virtualization, ASPLOS 2006