Virtual Machines

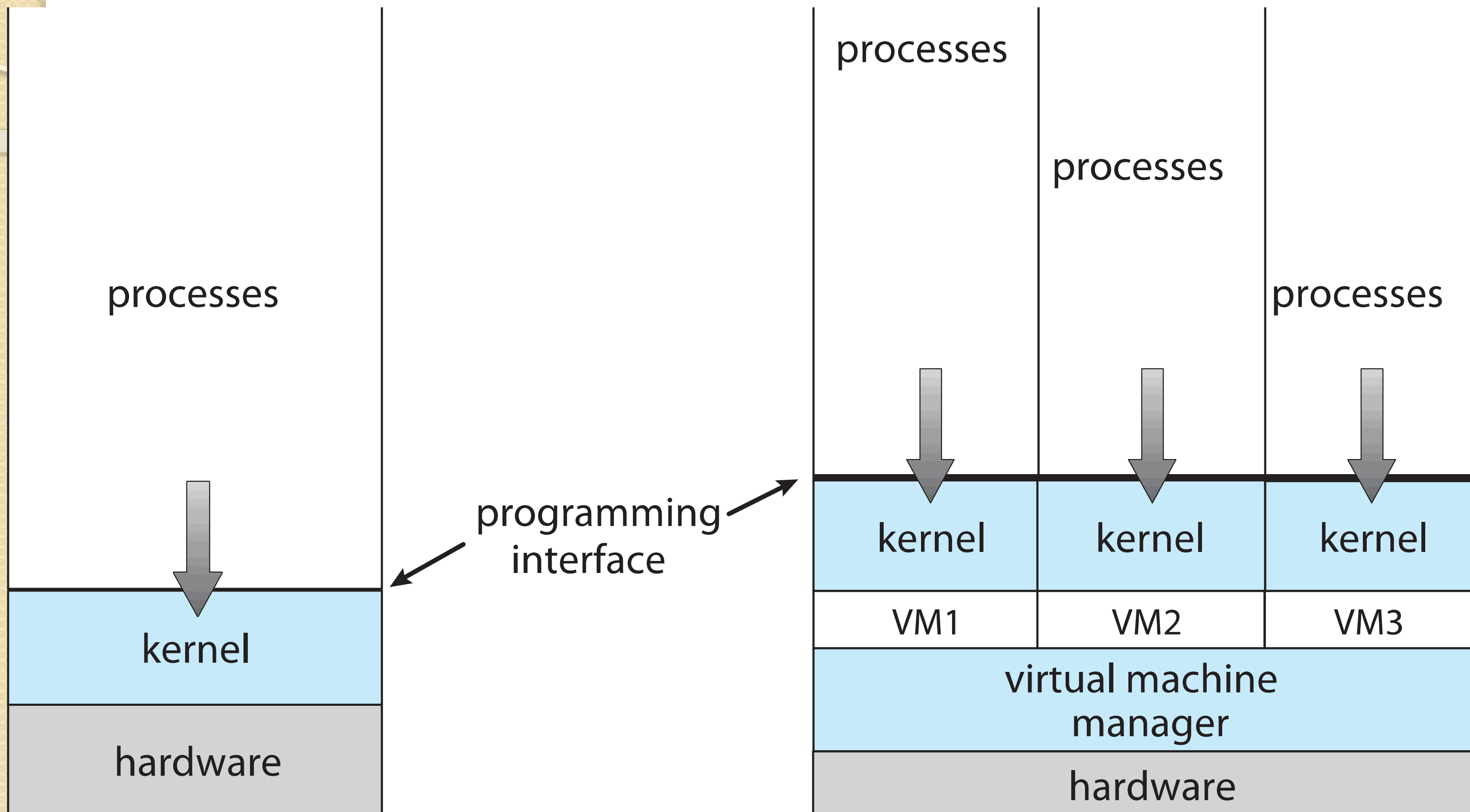
Outline

- Fundamental idea
- History
- Benefits and features
- Building Blocks
- Types of virtual machines and their Implementations
- Virtualization and operating system components
- Examples

Fundamental Idea

- Abstract hardware of a single computer into several different execution environments
  - Similar to layered approach
  - But layer creates virtual system (virtual machine, or VM) on which operation systems or applications can run
- Several components
  - Host – underlying hardware system
  - Virtual machine manager (VMM) or hypervisor – creates and runs virtual machines by providing interface that is identical to the host
    - Except in the case of paravirtualization
  - Guest – process provided with virtual copy of the host
    - Usually an operating system
- Single physical machine can run multiple operating systems concurrently, each in its own virtual machine

System Models



Non-virtual machine

Virtual machine

Implementation of VMMs

Vary greatly, with options including:

- *Type 0 hypervisors* - Hardware-based solutions that provide support for virtual machine creation and management via firmware
  - IBM LPARs and Oracle LDOMs are examples
- *Type 1 hypervisors* - Operating-system-like software built to provide virtualization
  - including VMware ESX, Joyent SmartOS, and Citrix XenServer
- *Type 1 hypervisors* – Also includes general-purpose operating systems that provide standard functions as well as VMM functions
  - including Microsoft Windows Server with HyperV and RedHat Linux with KVM
- *Type 2 hypervisors* - Applications that run on standard operating systems but provide VMM features to guest operating systems
  - including VMware Workstation and Fusion, Parallels Desktop, and Oracle VirtualBox

Implementation of VMMs – Other Variations

- Other variations include:
  - Paravirtualization - Technique in which the guest operating system is modified to work in cooperation with the VMM to optimize performance
  - Programming-environment virtualization - VMMs do not virtualize real hardware but instead create an optimized virtual system
    - used by Oracle Java and Microsoft.Net
  - Emulators – Allow applications written for one hardware environment to run on a very different hardware environment, such as a different type of CPU
  - Application containment - Not virtualization at all but rather provides virtualization-like features by segregating applications from the operating system, making them more secure, manageable
    - including Oracle Solaris Zones, BSD Jails, and IBM AIX WPARs
- Much variation due to breadth, depth and importance of virtualization in modern computing

History

- First appeared in IBM mainframes in 1972
- Allowed multiple users to share a batch-oriented system
- Formal definition of virtualization helped move it beyond IBM
  - A VMM provides an environment for programs that is essentially identical to the original machine
  - Programs running within that environment show only minor performance decreases
  - The VMM is in complete control of system resources
- In late 1990s Intel CPUs fast enough for researchers to try virtualizing on general purpose PCs
  - Xen and VMware created technologies, still used today
  - Virtualization has expanded to many OSes, CPUs, VMMs

Benefits and Features (1)

- Host system protected from VMs
- VMs protected from each other
  - Sharing is provided via shared file system volume, network communication
- Freeze, suspend, running VM
  - Then can move or copy somewhere else and resume
  - Snapshot of a given state, able to restore back to that state
    - some VMMs allow multiple snapshots per VM
  - Clone by creating copy and running both original and copy
- Run multiple, different OSes on a single machine
  - Consolidation, app dev, …

Benefits and Features (2)

- Enables OS research
- Can improve system development efficiency
- *Templating* – create an OS + application VM, provide it to customers, use it to create multiple instances of that combination
- *Live migration* – move a running VM from one host to another!
  - No interruption of user access
- All those features taken together → cloud computing
  - Using APIs, programs tell cloud infrastructure (servers, networking, storage) to create new guests, VMs, virtual desktops, …

Building Blocks

Generally difficult to provide an exact duplicate of underlying machine

- Especially if only dual-mode operation available on CPU
- But getting easier over time as CPU features and support for VMM improves
- Most VMMs implement virtual CPU (VCPU) to represent state of CPU per guest as guest believes it to be
  - when guest context switched onto CPU by VMM, information from VCPU loaded and stored
- Several techniques, as described in next slides

Trap and Emulate (1)

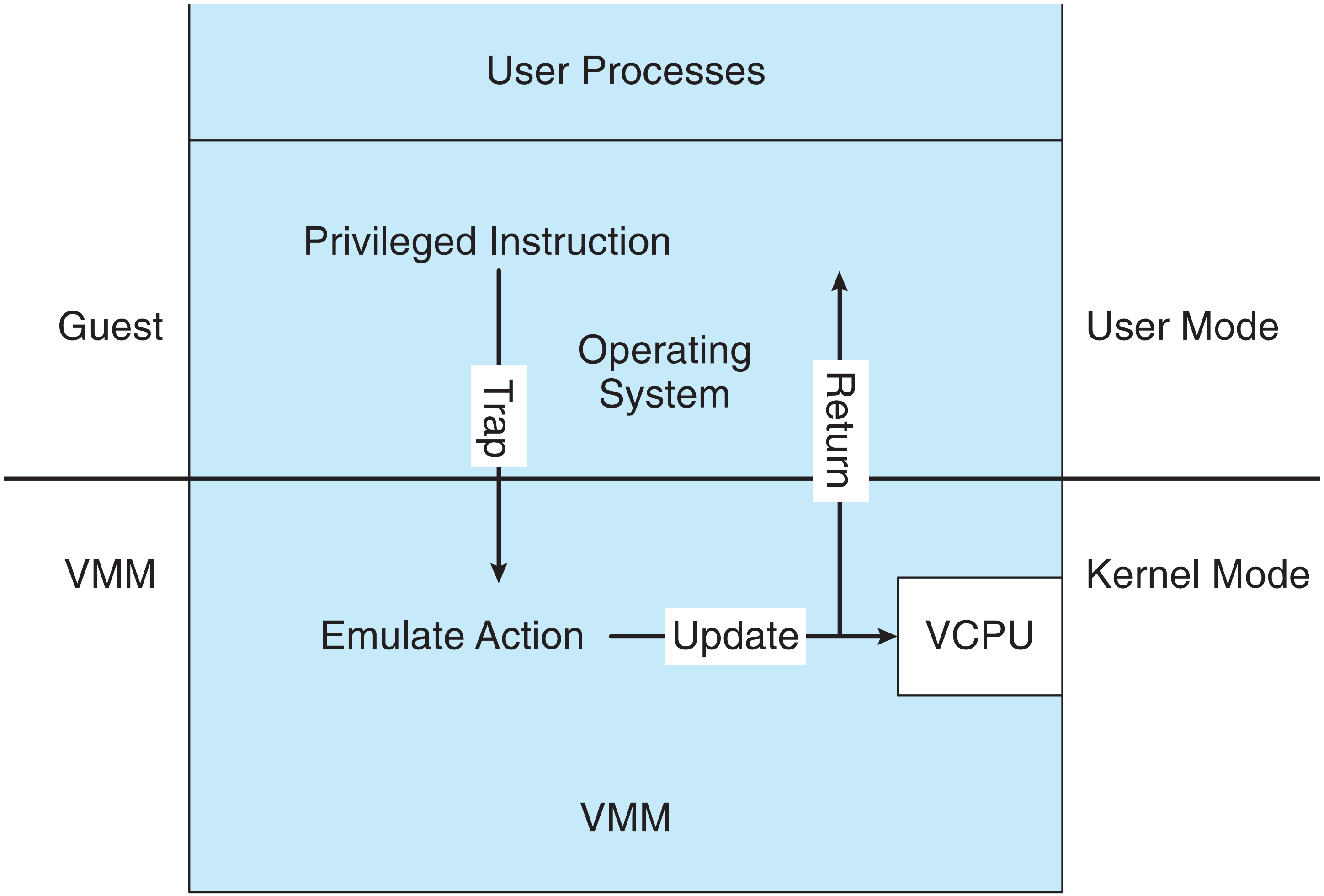
Dual mode CPU means guest executes in user mode

- Kernel runs in kernel mode
- Not safe to let guest kernel run in kernel mode too
- So VM needs two modes – virtual user mode and virtual kernel mode
  - both of which run in real user mode
- Actions in guest that usually cause switch to kernel mode must cause switch to virtual kernel mode

Trap and Emulate (2)

- How to switch from virtual user mode to virtual kernel mode?
  - Attempting a privileged instruction in user mode causes an error -> trap
  - VMM gains control, analyzes error, executes operation as attempted by guest
  - Returns control to guest in user mode
  - Known as trap-and-emulate
  - Most virtualization products use this at least in part
- User mode code in guest runs at same speed as if not a guest
- But kernel mode privileged code runs slower due to trap-and-emulate
  - Especially a problem when multiple guests running, each needing trap-and-emulate
- CPUs adding hardware support, more CPU modes to improve virtualization performance

Trap and Emulate Implementation



Binary Translation (1)

Some CPUs don’t have clean separation between privileged and nonprivileged instructions

- Earlier Intel x86 CPUs are among them
  - earliest Intel CPU designed for a calculator
- Backward compatibility means difficult to improve
- Consider Intel x86 popf instruction
  - loads CPU flags register from contents of the stack
  - if CPU in privileged mode -> all flags replaced
  - if CPU in user mode -> some flags replaced
    - No trap is generated

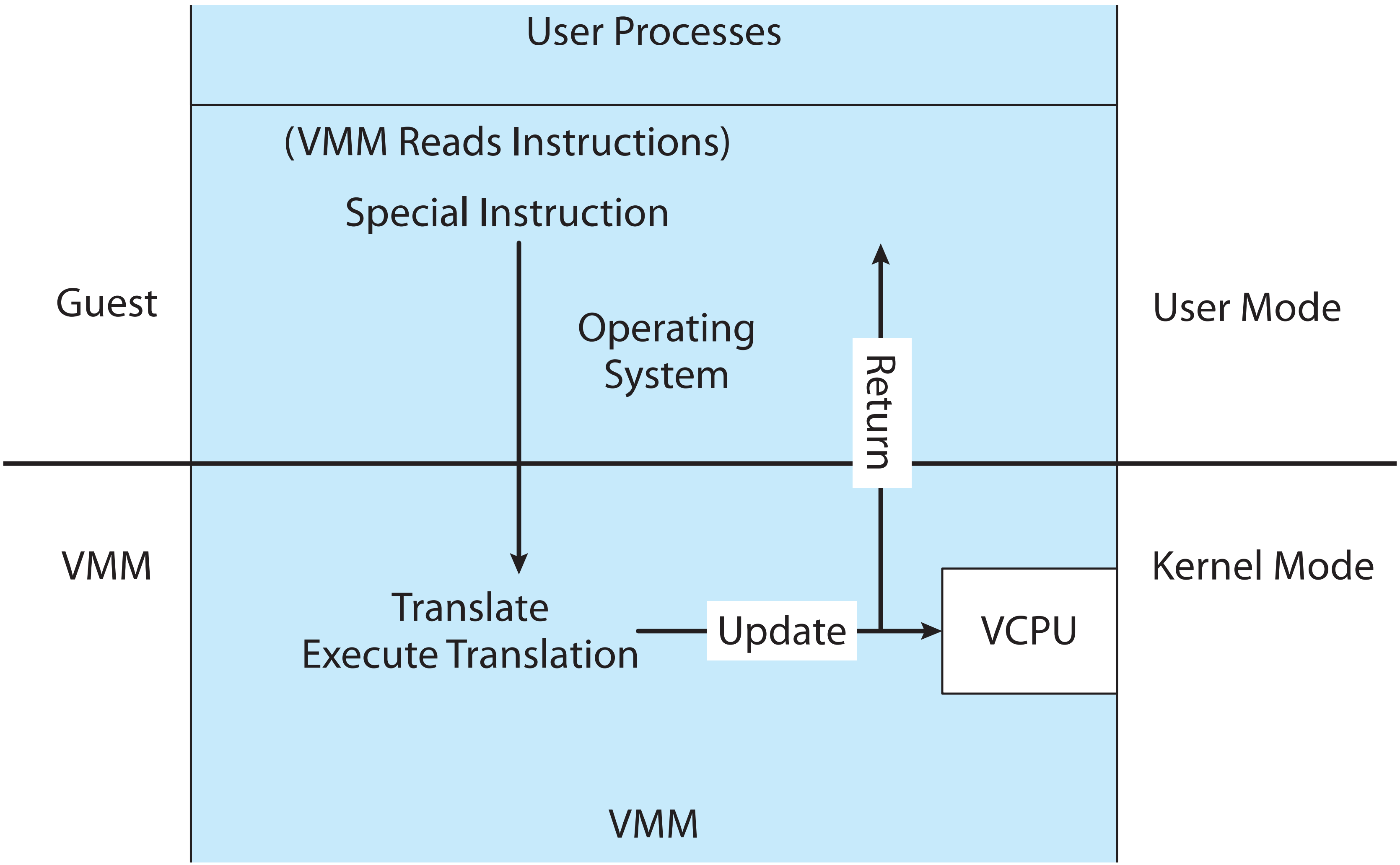
Binary Translation (2)

- Other similar problem instructions we will call special instructions
  - Caused trap-and-emulate method considered impossible until 1998
- Binary translation solves the problem
  - Basics are simple, but implementation very complex
  - If guest VCPU is in user mode, guest can run instructions natively
  - If guest VCPU in kernel mode (guest believes it is in kernel mode)
    - VMM examines every instruction guest is about to execute by reading a few instructions ahead of program counter
    - non-special-instructions run natively
    - special instructions translated into new set of instructions that perform equivalent task (for example changing the flags in the VCPU)

Binary Translation (3)

- Implemented by translation of code within VMM
- Code reads native instructions dynamically from guest, on demand, generates native binary code that executes in place of original code
- Performance of this method would be poor without optimizations
  - Products like VMware use caching
    - translate once, and when guest executes code containing special instruction cached translation used instead of translating again
    - testing showed booting Windows XP as guest caused 950,000 translations, at 3 microseconds each, or 3 second (5 %) slowdown over native

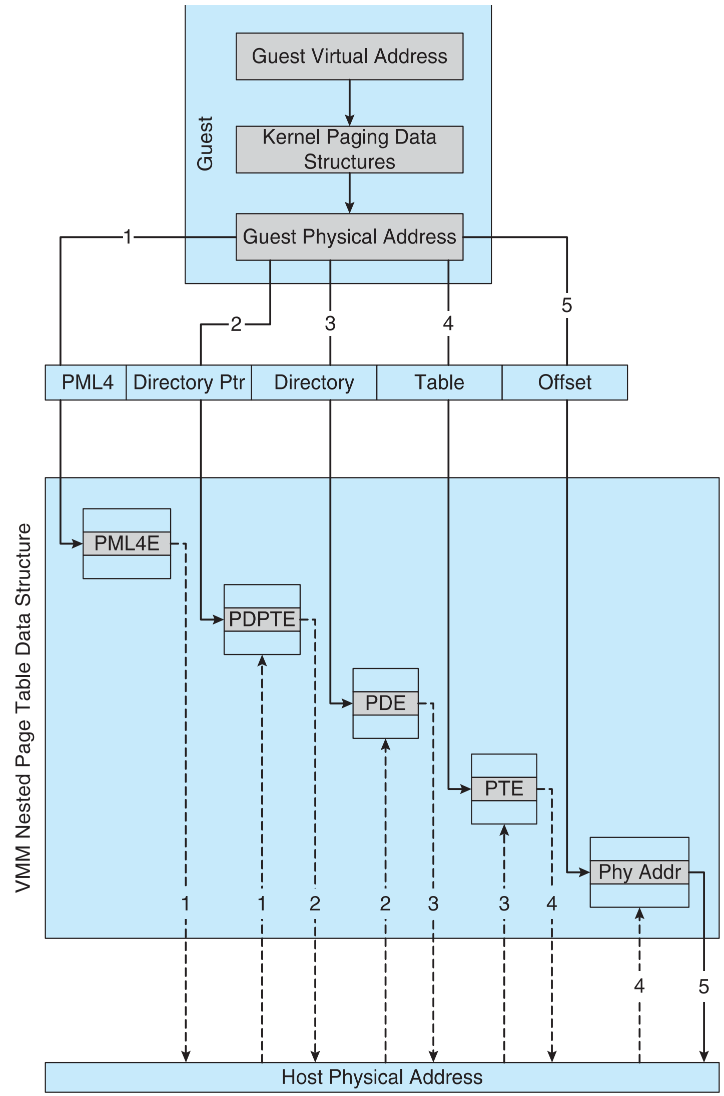
Binary Translation Implementation



Nested Page Tables

- Memory management another general challenge to VMM implementations
- How can VMM keep page-table state for both guests believing they control the page tables and VMM that does control the tables?
- Common method (for trap-and-emulate and binary translation) is nested page tables (NPTs)
  - Each guest maintains page tables to translate virtual to physical addresses
  - VMM maintains per guest NPTs to represent guest's page-table state
    - just as VCPU stores guest CPU state
  - When guest on CPU → VMM makes that guest's NPTs the active system page tables
  - Guest tries to change page table → VMM makes equivalent change to NPTs and its own page tables
  - Can cause many more TLB misses → much slower performance

Nested Page Tables



Hardware Assistance

- All virtualization needs some HW support
- More support ➔ more feature rich, stable, better performance of guests
- Intel added new VT-x instructions in 2005 and AMD the AMD-V instructions in 2006
  - CPUs with these instructions remove need for binary translation
  - Generally define more CPU modes – "guest" and "host"
  - VMM can enable host mode, define characteristics of each guest VM, switch to guest mode and guest(s) on CPU(s)
  - In guest mode, guest OS thinks it is running natively, sees devices (as defined by VMM for that guest)
    - access to virtualized device, priv instructions cause trap to VMM
    - CPU maintains VCPU, context switches it as needed
- HW support for NPTs, DMA, interrupts as well, over time