COS 318: Operating Systems

Virtual Machine Monitors

Prof. Margaret Martonosi
Computer Science Department
Princeton University

http://www.cs.princeton.edu/courses/archive/fall11/cos318/

Announcements

- Project 6 due Tuesday Jan 17
- Final Exam: Sunday Jan 22 at 1pm
  - in CS104 (LARGE AUDITORIUM! NOT THIS ROOM!)
  - 90 minutes long.
  - Cumulative, but biased toward material after the midterm.
  - 1-page (front and back) cheat sheet if you desire.
  - Otherwise, the exam is closed-book, closed-notes.

Introduction

- Have been around since 1960's on mainframes
  - used for multitasking
  - Good example – VM/370
- Have resurfaced on commodity platforms
  - Server Consolidation
  - Web Hosting centers
  - High-Performance Compute Clusters
  - Managed desktop / thin-client
  - Software development / kernel hacking

Why do we care?

- Manageability
  - Ease maintenance, administration, provisioning, etc.
- Performance
  - Overhead of virtualization should be small
- Isolation
  - Activity of one VM should not impact other active VMs
  - Data of one VM is inaccessible by another
- Scalability
  - Minimize cost per VM

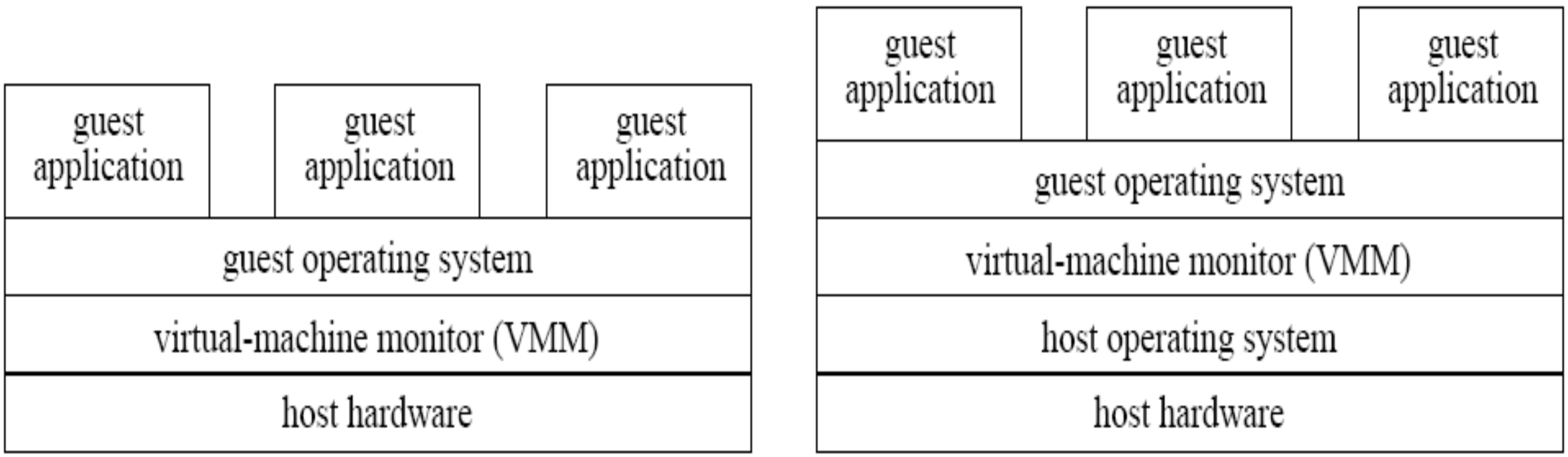
Virtual Machine Monitor (VMM)

- Resides as a layer below the operating system
- Presents a hardware interface to an OS
- Multiplexes resources between several virtual machines (VMs)
- Performance Isolates VMs from each other

Virtualization Styles

- Fully virtual*izing* VMM
  - Virtual machine looks exactly like some physical machine.
  - (But maybe not the one you're running on right now.)
  - Run OS or other software unchanged (from the machine the VM mimics)
- Para- virtual*izing* VMM
  - Some architecture features are hard to virtualize, so exact copy is too difficult (or slow).
  - Instead, punt on a few features.
  - VMM provides idealized view of hardware and then fixes under the covers.
  - Since the VMM doesn't match any real hardware, an OS running on it MUST be changed, not legacy.

VMM Types



Type I VMM

Type II VMM

For VM approaches you have used, which type are they?

VMM Classification

	Type I	Type II
Fully-virtualized	VMware ESX	VMware Workstation
Para-virtualized	Xen	User Mode Linux

VMM Implementation

Should efficiently virtualize the hardware

- Provide illusion of multiple machines
- Retain control of the physical machine

Subsystems

- Processor Virtualization
- I/O virtualization
- Memory Virtualization

Processor Virtualization

Popek and Goldberg (1974)

- Sensitive instructions: only executed in kernel mode
- Privileged instructions: trap when run in user mode
- CPU architecture is virtualizable only if sensitive instructions are subset of privileged instructions

- When guest OS runs a sensitive instruction, must trap to VMM so it maintains control

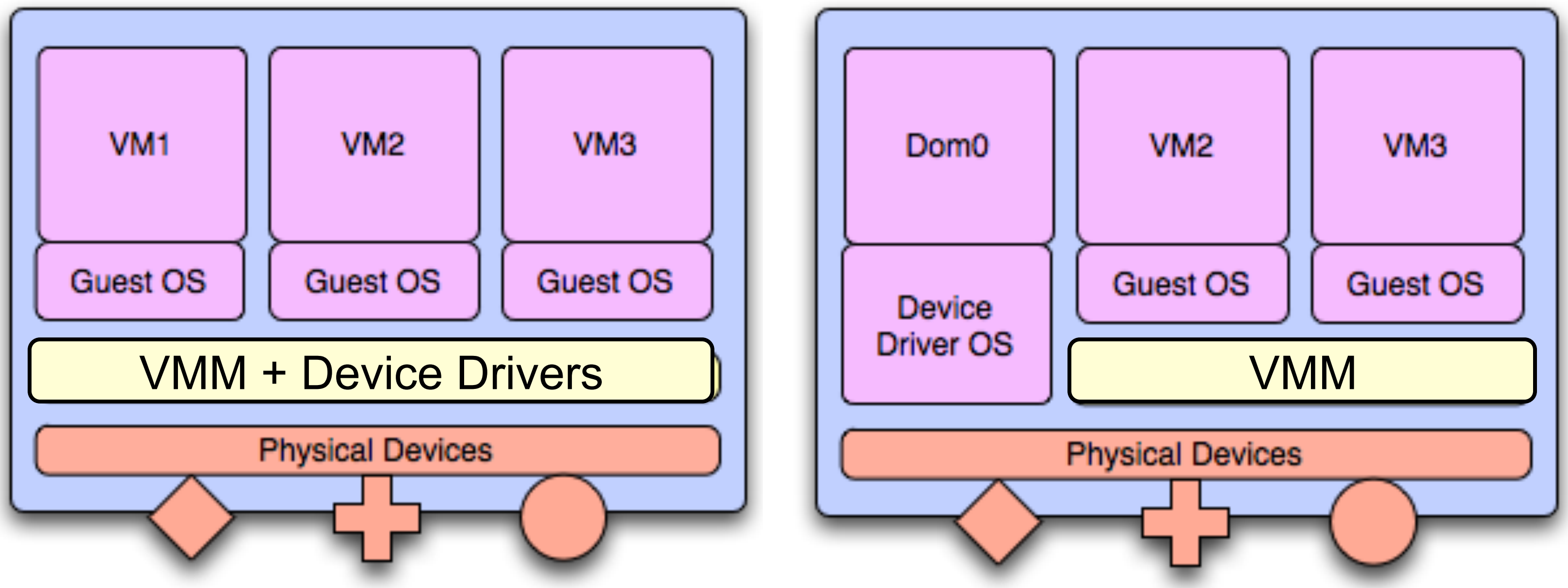
x86 Processor Virtualization

- x86 architecture is not fully virtual*izable*
  - Certain privileged instructions behave differently when run in unprivileged mode
    - POPF instruction that is used to set and clear the interrupt-disable flag. If run in user mode, it has no effect: it's a NO-OP.
  - Certain unprivileged instructions can access privileged state
- Techniques to address inability to virtualize x86
  - Replace non-virtualizable instructions with easily virtualized ones statically (Paravirtualization)
  - Perform Binary Translation (Full Virtualization)

I/O Virtualization

- Issue: lots of I/O devices
- Problem: Writing device drivers for all I/O device in the VMM layer is not a feasible option
- Insight: Device driver already written for popular Operating Systems
- Solution: Present *virtual* I/O devices to *guest* VMs and channel I/O requests to a trusted *host* VM running popular OS

I/O Virtualization



Higher performance, but PITA to write all the drivers

Lower performance, but reuses drivers guest OS already has.

Memory Virtualization

- Traditional way is to have the VMM maintain a shadow of the VM's page table
- The shadow page table controls which pages of machine memory are assigned to a given VM
- When guest OS updates its page table, VMM updates the shadow

Case Study: VMware ESX Server

- Type I VMM - Runs on bare hardware
- Full-virtualized – Legacy OS can run unmodified on top of ESX server
- Fully controls hardware resources and provides good performance

ESX Server – CPU Virtualization

- Most user code executes in Direct Execution mode; near native performance

- Uses *runtime* Binary Translation for x86 virtualization
  - Privileged mode code is run under control of a Binary Translator, which emulates problematic instructions
  - Fast compared to other binary translators as source and destination instruction sets are nearly identical

ESX Server – Memory Virtualization

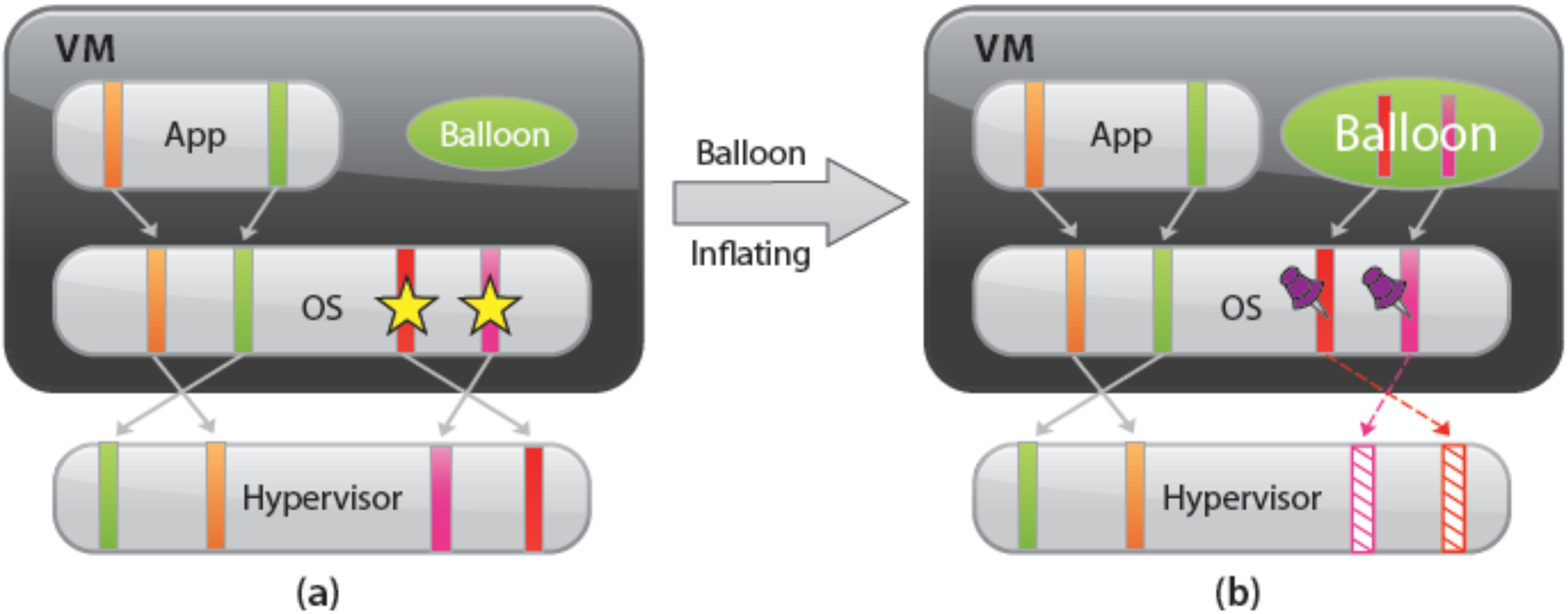
- Maintains shadow page tables with virtual to machine address mappings.
- Shadow page tables are used by the physical processor
- Guest OS page table: maps virtual addresses to “physical” addresses (note quotes)
- ESX maintains the pmap data structure per VM: maps “physical” to machine address mappings
- Shadow page table holds the combined effects of these two map steps
- ESX can easily remap a machine page when needed

ESX Server – Memory Mgmt

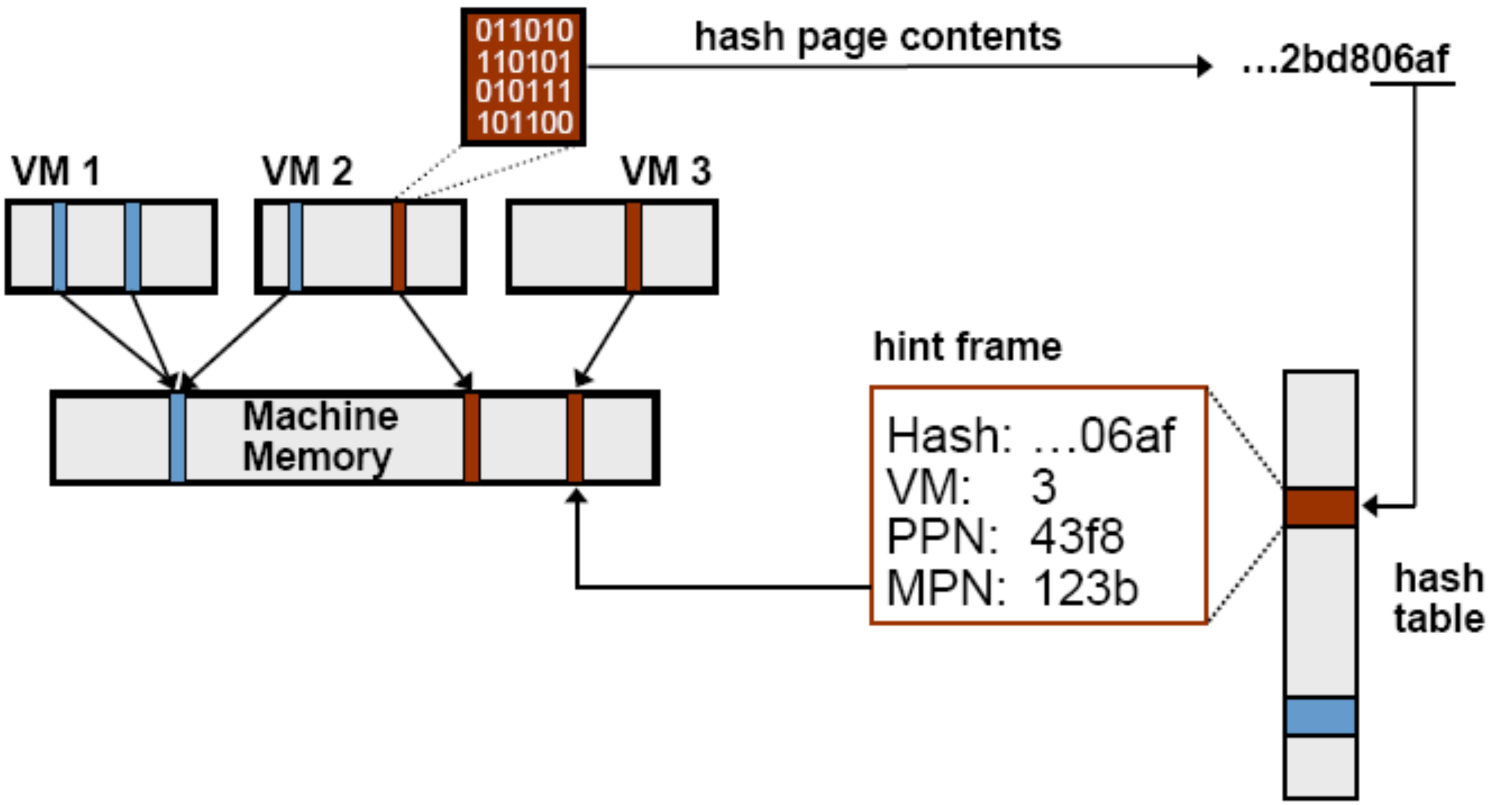
- Page reclamation – Ballooning technique
  - VMM reclaims memory when it detects thrashing/ overcommitment
  - VMM controls shadow page table, so it could just arbitrarily take a few pages away.
  - But the guest OS has better info on which pages are used or not-→ want *it* to make the decision.

- Page sharing – Content based sharing
  - Eliminates redundancy and saves memory pages when VMs use same operating system and applications

ESX Server- Ballooning



ESX Server – Page Sharing



Real World Page Sharing

		Total	Saved	
Workload	Guest Types	MB	MB	%
Corporate IT	10 Windows	2048	673	32.9
Nonprofit Org	9 Linux	1846	345	18.7
VMware	5 Linux	1658	120	7.2

Corporate IT – database, web, development servers (Oracle, Websphere, IIS, Java, etc.)

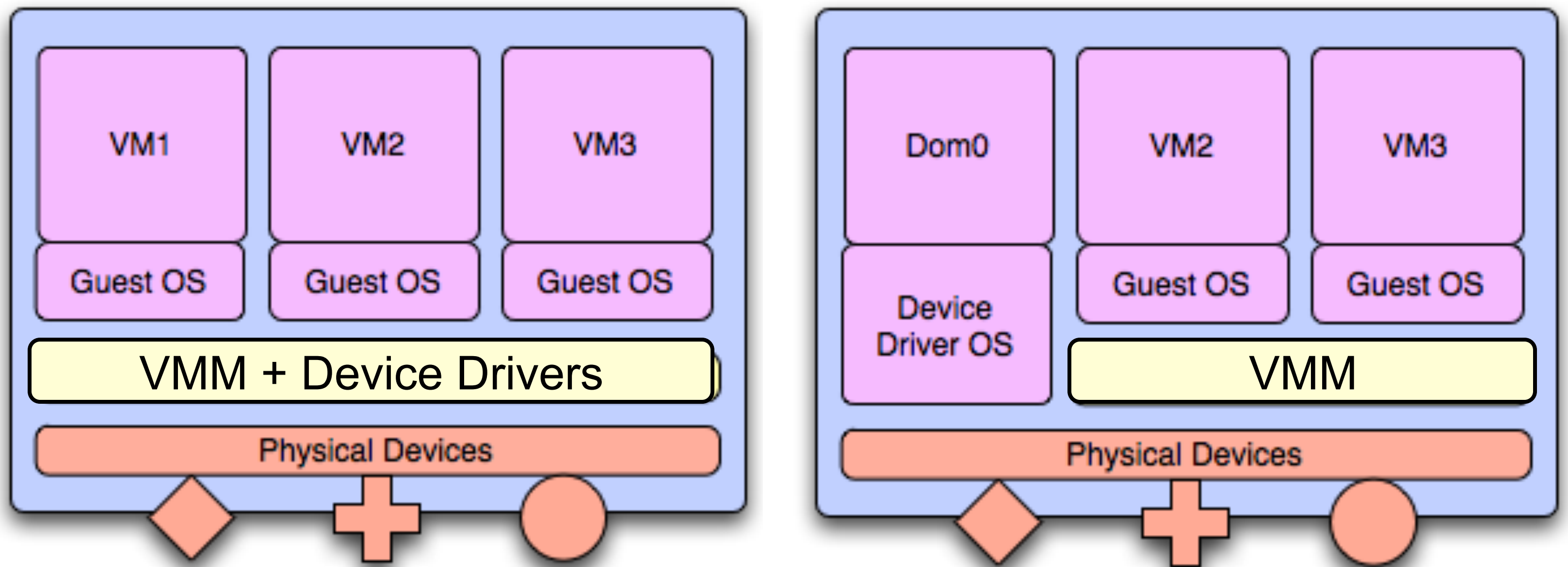
Nonprofit Org – web, mail, anti-virus, other servers (Apache, Majordomo, MailArmor, etc.)

VMware – web proxy, mail, remote access (Squid, Postfix, RAV, ssh, etc.)

ESX Server – I/O Virtualization

- Has highly optimized storage subsystem for networking and storage devices
  - Directly integrated into the VMM
  - Uses device drivers from the Linux kernel to talk directly to the device
- Low performance devices are channeled to special “host” VM, which runs a full Linux OS

I/O Virtualization



ESX uses both models: LHS for high-perf devices, RHS for rest.

Xen

- Type I VMM
- Para-virtualized
  - Linux->Xen: alters 3000 lines or about 1% of code
- Open-source
- Designed to be efficient & scalable:
  - run about 100 virtual machines on a single machine
- Used in Amazon Web Services EC2

Xen – CPU Virtualization

- Privileged instructions are para-virtualized by requiring them to be validated and executed with Xen
- Processor Rings
  - Guest applications run in Ring 3
  - Guest OS runs in Ring 1
  - Xen runs in Ring 0

Xen – Memory Virtualization(1)

- Initial memory allocation is specified and memory is statically partitioned
- A maximum allowable reservation is also specified.
- Balloon driver technique similar to ESX server used to reclaim pages

Xen – Memory Virtualization(2)

- Guest OS is responsible for allocating and managing hardware page table
- Xen involvement is limited to ensure safety and isolation
- Xen exists in the top 64 MB section at the top of every address space
  - Because if there were process switches, when entering and leaving the VMM, some (not all) CPUs would need TLB flushes at those points

Xen – I/O Virtualization

- Xen exposes a set of clean and simple device abstractions
- I/O data is transferred to and from each domain via Xen, using shared memory, asynchronous buffer descriptor rings
- Xen supports lightweight event delivery mechanism used for sending asynchronous notifications to domains

VMMs the only way to Virtualize?

- Alternative: Container-based OS (COS)
  - Eg., Solaris 10, Linux-Vserver, OpenVZ

Features	**VMM**	**COS**
Multiple kernels	✔	✘
Administrative power (root)	✔	✔
Manageability	✔	✔
Scalability	✔	✔✔
Isolation	✔✔	✔
Efficiency	✔	✔✔

The Big Finish

Key OS Topics

- Abstraction
- Resource Management
- Protection

- The first topic, abstraction, is a key enabler for the other two.
  - Think about Virtual memory…

Operating Systems are Illusionists

Physical reality

- Single CPU
- Interrupts
- Limited memory
- No protection
- Raw hard drive storage

Abstraction ("Looks like")

- Infinite number of CPUs
- Cooperating sequential threads
- Unlimited virtual memory
- Each address has its own machine
- Organized and reliable storage system

Operating Systems are Timeless!

- Example: VMs first used by IBM in early 1970s!
- Tradeoffs changed, hardware got cheap, and they went into dormancy
- Now back again: Why?

⇒ Moral of the story: Know the basics and be creative about how, where, when to apply them or variations!

Other Cool OS-related classes to take

- Spring
  - COS 461: Computer Networks
  - COS 598b: Mobile Computing
- Fall
  - COS518: Advanced Computer Systems
  - COS429: Security
  - Sometimes offered: COS 598A: Parallel Arch & Prog.