

Virtual Machine Monitors: The Original Execution Environment

Mendel Rosenblum

***Associate Professor of Computer Science
Stanford University***

***Co-founder, Chief Scientist
VMware Inc.***

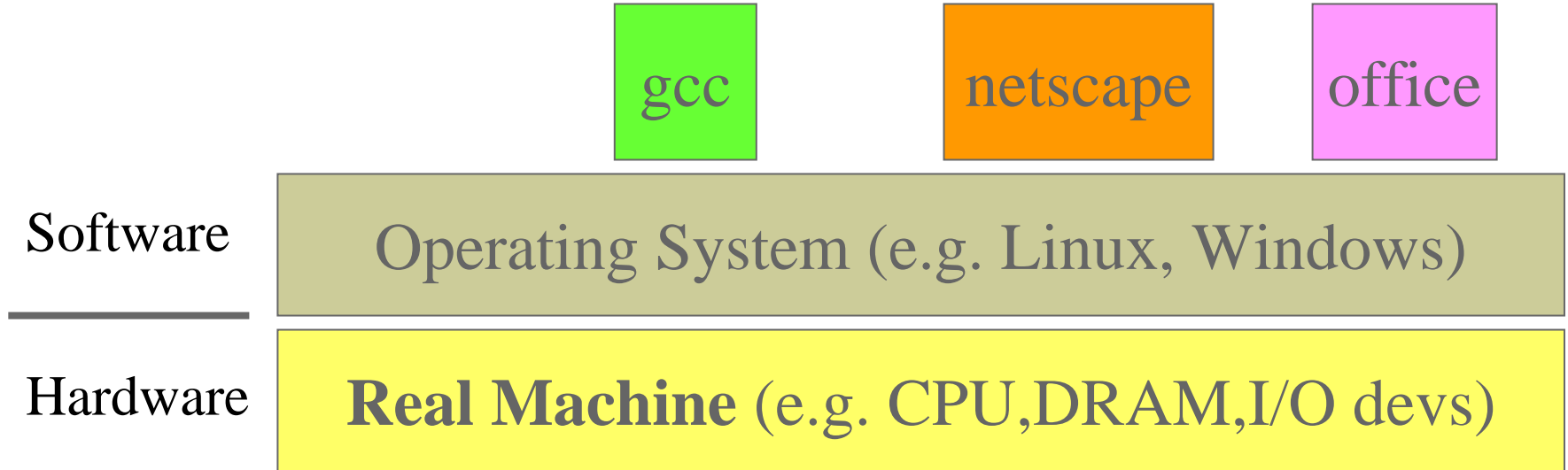
Sept 2004

Talk Outline

- What is a virtual machine?
- Similarities and differences between VMs
- Common attributes of virtual machines
- **Low-level (?)** virtual machine monitors (VMMs)
 - Hardware-level VM or Classic VM
- Modern virtual machines monitors – VMware™ Inc.
- Some predictions for the future

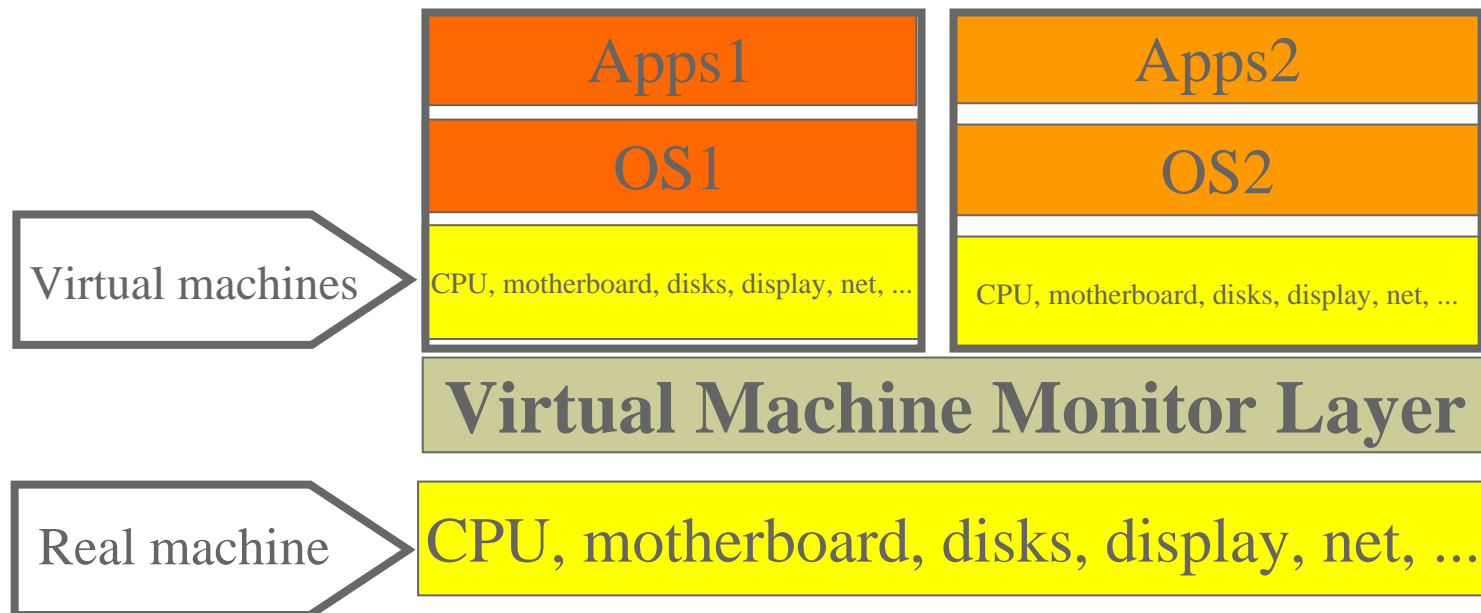
What is a real machine?

- Some hardware with layers of software on it.



1960s: What is a virtual machine?

- Virtual Machine Monitor (VMM)
 - Thin layer of software running on the raw hardware.
 - Exports an abstraction that looks like raw hardware.
 - *A Virtual Machine*



The Rise and Fall of the VMM

-- 1970s: Looks like a good idea --

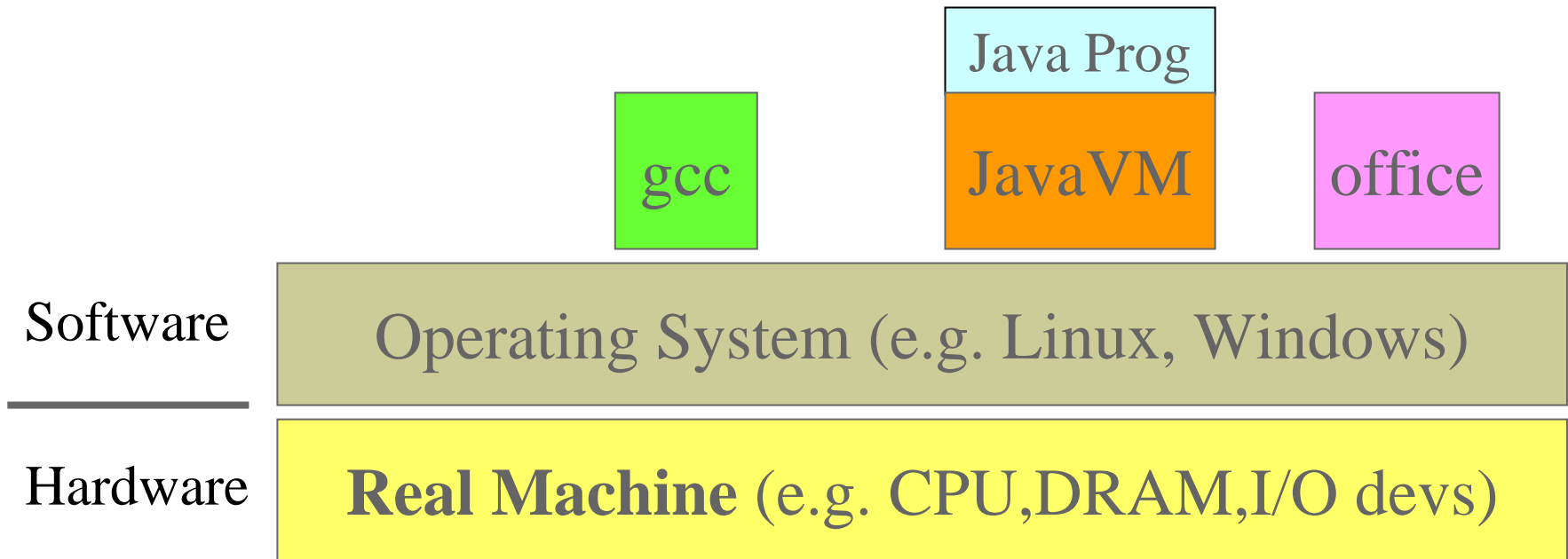
- IBM VM/370 – A VMM for IBM mainframes
 - Multiplex multiple OS environments on expensive hardware.
 - Desirable when few machine around.
- Popular idea in the 1960s and 1970s
 - Entire conferences on virtual machine monitors

-- 1980s: Looks like a dumb idea --

- Hardware got cheap but wimpy
 - VMM neither desirable nor possible.
- IBM kills VM/CMS in favor of MVS
 - Multi-user OS is better than N single-user + VMM.

1990s: The Rebirth of the Virtual Machine

- Define an abstract (i.e. different from hardware) machine specification – A Virtual Machine (HLL VM)
 - Typically emulate machine within an OS process
 - Examples: p-system, Smalltalk, Java, Microsoft's CLR

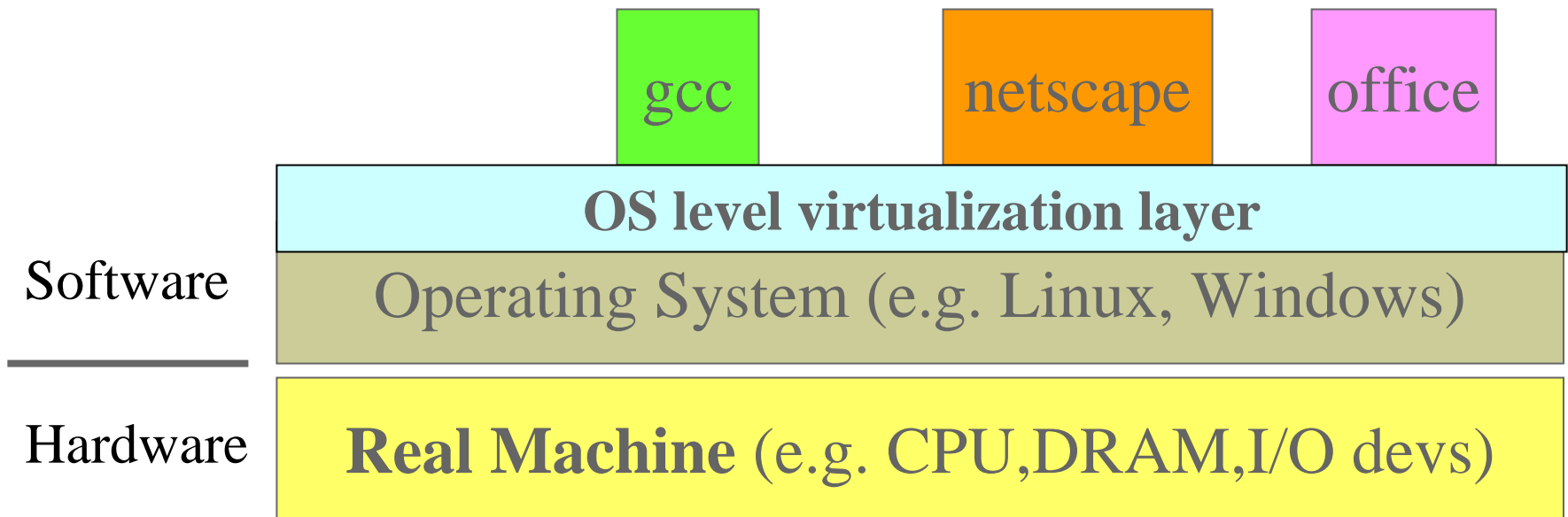


Virtual Machines a good idea again

- Unlike old virtual machines, no existing software.
- Market it as better than real machines:
 - Faster development time.
 - Write once, run anywhere.
 - Type and memory safety.
 - Fewer bugs, better security, etc.
- Popular idea in the 2000s
 - Entire conferences on virtual machines

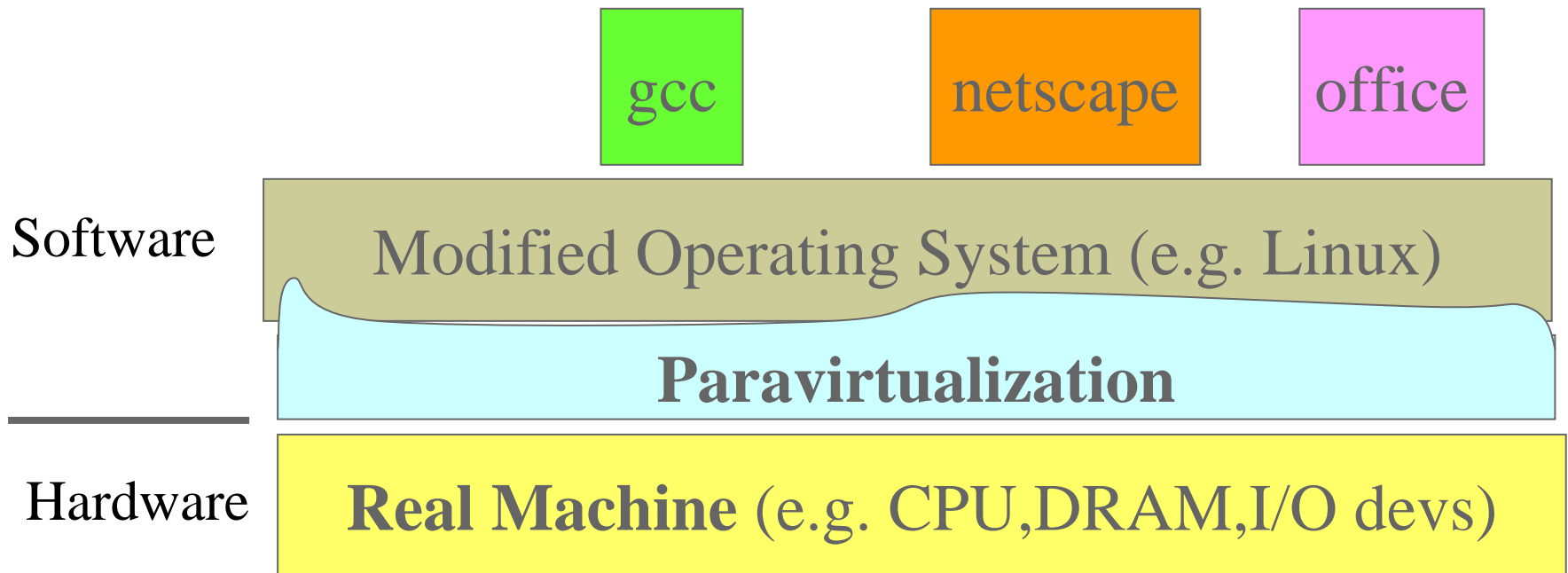
More VMs: OS level virtual machines

- Squeeze in between OS and applications
 - Done at libraries or system call interface.
- Each application or set of apps run in a virtual machine.



Yet more VMs: Slice it inside the OS

- Can make the cut inside of the operating system
 - Example: HAL level
- Make it like a VMM only slightly different in places
 - Example: Paravirtualization



Commonality across things called **virtual machines**

- All are targets of a programmer or compiler
 - Someone writes software for it.
 - Examples:
 - *VMMs run software authored for real machine*
 - *Java VMs run software written in Java.*
- All benefit from the “level of indirection”
 - All problems can be solved by a level of indirection.
 - Use the layer to improve the software running in the VM.

Common virtual machine attributes

- **Compatibility**
 - VM provide a compatible interface to the software.
- **Isolation**
 - VM acts as a sandbox to isolate execution.
- **Encapsulation**
 - VM provides an ability to manipulate and control the execution.
- **Performance**
 - All VM systems argue the benefits far outweigh any overheads.

Software compatibility

- VM will run all the software that target it.
- Lower-level VMs have advantages here:
 - Hardware-level VMM:
 - All** software (app & OS) written for hardware.
 - Paravirtualization:
 - All applications for the ported OSes.
 - OS-level VM:
 - All application for that OS/hardware combination.
 - HLL VM:
 - Programs compiled to the byte code.

VMM Software compatibility

- **Key: Make virtual machine abstraction match real HW**
 - All software that runs on real HW runs in VM
- **Example: VMware™'s products run**
DOS, Win 3.1,95,98,NT,2000,ME,XP,2003 Linux, FreeBSD, etc.
- **Most compatible of application compatibility solutions:**
 - Hardware interface: tractable complexity, slow rate of change
 - *Example: PC98, PC99, ...*
 - OS API interface: intractable complexity, rapid change
 - *Example: Win32 API*

Isolation capability

- Claim: VMs should not be able to get out of the sandbox to attack other VMs or virtualization software layer.
 - Layer controls what resources are accessible to each VM.
- VMs can be isolated and requests vetted.
 - Policy-based access control.
- Assurance dependent on the implementation.
 - Size/complexity of interface.
 - *Compare hardware interface vs. win32 system call*
 - Size/complexity of trusted computing base(TCB).
 - *Millions of lines of code in TCB.*

VMM Isolation capability

- Key: Use HW protection mechanisms to isolate VMs
 - Example: Protection rings, MMU protection bits
 - Simple implementation/small size.
- Complete isolation
 - Code running in a VM can't read, modify, break, etc:
Other virtual machines
The virtual machine monitor
- Isolation comparable to separate physical machines
 - Handle accidents (e.g. software bugs)
 - Malicious attacks (e.g. hackers)
- Compare with win32 API

Encapsulation

- Have ability to manipulate and control software in VM
 - Save execution state.
 - Transfer VM over networks.
 - Effect the input/outputs to the software running in VM
- Manage VM execution on machines
 - Provisioning, load balancing, high availability, etc.
 - Examples: Java application servers, VMware™'s ESX Server
- Decoupling of software from hardware.
 - Virtualization layer controls mapping
- Treat software in VM as first class object.

VMM Encapsulation

- **Key: VMM sits between VM and hardware**
 - Virtual machine is not tied to physical hardware
- **VMM can completely isolate VM from hardware:**
 - Support for checkpoint/restore operations
 - Virtual machine migration
 - Undo execution
- **Hardware independence**
 - VMM maps “standard” virtual HW to real HW
 - *VMM provides the “conversion” layer*
- **Management interface for mapping VMs to HW.**
 - Provisioning, high availability, etc.

VMM Low overhead/High Performance

- Key: Configure HW to directly run Virtual Machines
 - Use CPU to emulate a virtual CPU
 - Use real physical memory to emulate virtual physical memory
 - Emulate a disk with a disk, etc.
- Trick from 1960s:
 - Configure hardware to safely give it to virtual machine
 - VMM gets control on any privileged operation
- Virtual machine runs within a few percent of native

Which “virtual machine” will win?

- In theory you only need a single level of indirection
 - Will someone come up with the single, right VM?
- Benefits of HLL VMs and VMMs are real and distinct
 - Compelling uses for the foreseeable future.
 - HLL VMs: Better runtime environment
 - VMMs: Manage hardware and **all** software
- Unlikely to find a compromise in the middle that works
 - Lose the benefits of each
- Synergies between levels?

Future trends for VMMs

- Lost art of virtualizable architectures has been found:
 - New Power architecture
 - Intel's Vanderpool Technology (VT)
- Overheads for virtualization going down to 1970s levels
 - Few percentage slowdown – everything can run in VMs
- Compelling problems in manageability, availability, reliability and security
 - Addressable with modern VMMs
- Extend VMM to enhance software in side of VM
 - Example: Make more secure

Modern virtual machine monitors

- VMware™ Workstation
 - Virtual Machine matches an modern x86 desktop PC.
 - Hosted architecture – Adds VMs to existing OS (Host OS)
 - Virtual I/O devices implemented using host OS services.
- VMware™ ESX server with Virtual Center
 - Virtual Machine matches an modern SMP x86 server
 - Traditional VMM architecture
 - *High performance I/O for network and disk*
 - *Sophisticated resource management (CPU, MEM, I/O)*
 - Remote control and provisioning of VMs.
 - Cluster management with hot virtual machine migration.

Modern usage of VMMs

- Backward compatibility, innovation enabler.
 - Run legacy apps on legacy OS not new OS.
 - *Example: Microsoft's problem*
- Environment isolation
 - Personal Windows, Company Windows
 - Classified/unclassified environments.
- Logical partitioning of hardware
 - Give less than one CPU to an environment
- Enhance security of software
 - Example: Intrusion detection in VMM

VMMs in modern computing

- Old view: OS is an extension of hardware
- Problem: Modern OSes are too complex, buggy, insecure, etc. to tie to hardware and treat special (every app uses same copy).
- Use VMMs to provide mapping
 - Provisioning, Resource management, Administration
- Key: VMM can be made simple enough to:
 - Have high assurance.
 - Won't suffer same problem as modern OSes.
- Modern OS becomes a library that you link your applications with not the hardware!

Conclusions

- Many things called “virtual machines” at different levels
 - Share many of the same benefits and tradeoffs
- There are compelling reasons for two:
 - Hardware level virtualization.
 - Language level runtime.
- Hardware-level VMMs are back and exciting both commercially and in research
 - Note: Many of the lessons of the first coming still apply.

Backward compatibility with VMMs

- Backward compatibility is bane of new OSes.
 - Huge effort require to innovate but not break.
- Recent security consideration make it impossible
 - Choice: Close security hole and break apps or be insecure
- Example: Not all WinNT applications run on WinXP.
 - In spite of a huge effort to make WinXP compatible.
 - Given the number of applications that run on WinNT, practically any change will break something.
If (OS == WinNT)....
- Solution: Use a VMM to run both WinNT and WinXP
 - Obvious for OS migration as well: Windows -> Linux

Isolation: Access to Classified Networks

- Traditional tension: Security vs. Usability
 - Secure systems tend not to be that usable.
 - Flexible systems are not that secure.
- Additional information assurance requirement:
 - Data cannot flow between networks of different classification.
- Solution: Run two VMs:
 - Classified VM
 - Internet VM
- Use isolation property to isolate two VMs
 - VMM has control of the information flow between machines
 - *Declassifier mechanism*

Logical partitioning of server machines

- **Run multiple servers on same box**
 - Ability to give away less than one machine
 - *Modern CPUs more power than most services need.*
 - 0.10U rack space machine - Better power, cooling, floor space, etc.
 - Server consolidation trend: N machine -> 1 real machine
- **Isolation of environments**
 - Printer server doesn't take down Exchange server
 - Compromise of one VM can't get at data of others
- **Resource management**
 - Provide service-level agreements
- **Heterogeneous environments**
 - Linux, FreeBSD, Windows, etc.

Example: Using VMM to enhance security

- Problem Area: Intrusion Detection Systems (IDS).
- Trade-offs
 - Host-based IDS (HIDS):
 - + *Good visibility to catch intruder.*
 - *Weak isolation from intruder disabling/masking IDS.*
 - Network-based IDS (NIDS):
 - + *Good isolation from attack from intruder.*
 - *Weak visibility can allow intruder to slip by unnoticed.*
- **Would like visibility of HIDS with isolation of NIDS.**
 - Idea: Do it in the virtual machine monitor.

VMM-based Intrusion Detection System

- Strong isolation
 - VMM isolate software in VM from VMM.
 - Comprise OS in VM can't disable IDS in VMM.
- Introspection – Peer inside at software running in VM
 - VMM can see: Physical memory, registers, I/O device state, etc.
 - Signature scan of memory
 - *Look through physical memory for patterns or signs of break-in*
- Interposition – Modify VM abstraction to enhance security
 - Memory Access Enforcer
 - *Interpose on page protection.*
 - NIC Access Enforcer
 - *Interpose on virtual network device.*

Collective Project: A Compute Utility

- Distributed system where all software runs in VMs
Research with Prof. Monica Lam and students.
- Virtual Appliance abstraction
 - x86 virtual machine.
 - Target specialized environment (e.g. program development)
 - Store in a centralized persistent storage repository.
 - Cached on the machine where virtual appliances run.
- Target benefits
 - System administration
 - *Centralize and amortize administration of a virtual appliance.*
 - Mobility
 - *Computing environment follows user around.*