ADVANCED OPERATING SYSTEMS

Disco, Virtual Machines

Disco: Running Commodity Operating Systems on Scalable Multiprocessors (1997)

- What kind of paper is this?
  - New big idea?
  - Measurement paper?
  - Experiences/lessons learnt paper?
  - A system description?
  - Performance study?
  - Refute-conventional wisdom?
  - Survey paper?
  - Something else?

Disco: Running Commodity Operating Systems on Scalable Multiprocessors (1997)

- What kind of paper is this?
  - New application of old technology
  - History repeats itself
  - A new twist on an old idea

What was the problem they were trying to solve?

What was the problem they were trying to solve?

- A quick and dirty way to implement a multi-processor OS
- Developing multi-processor operating systems is hard and time-consuming
- What were the alternatives?

What was the problem they were trying to solve?

- A quick and dirty way to implement a multi-processor OS
- Developing multi-processor operating systems is hard and time-consuming
- What were the alternatives?
  - Port uni-processor OS to MP
  - Partition HW and run uni-processor OS per partition
  - Disco
    - "reduces the gap between hardware innovation and the adaptation of system software"

What are the new ideas?

- Run multiple single-processor operating sytems over DISCO VMM
  - How is this different from the hardware partition option of previous slide?
- Use distributed system facilities to provide a single-system image to the user
- Eliminates inefficiencies
  - E.g., allow transparent buffer cache sharing among virtual machines
- Use page placement and dynamic page migration to hide non-uniformity of memory access
  - What is a NUMA machine?

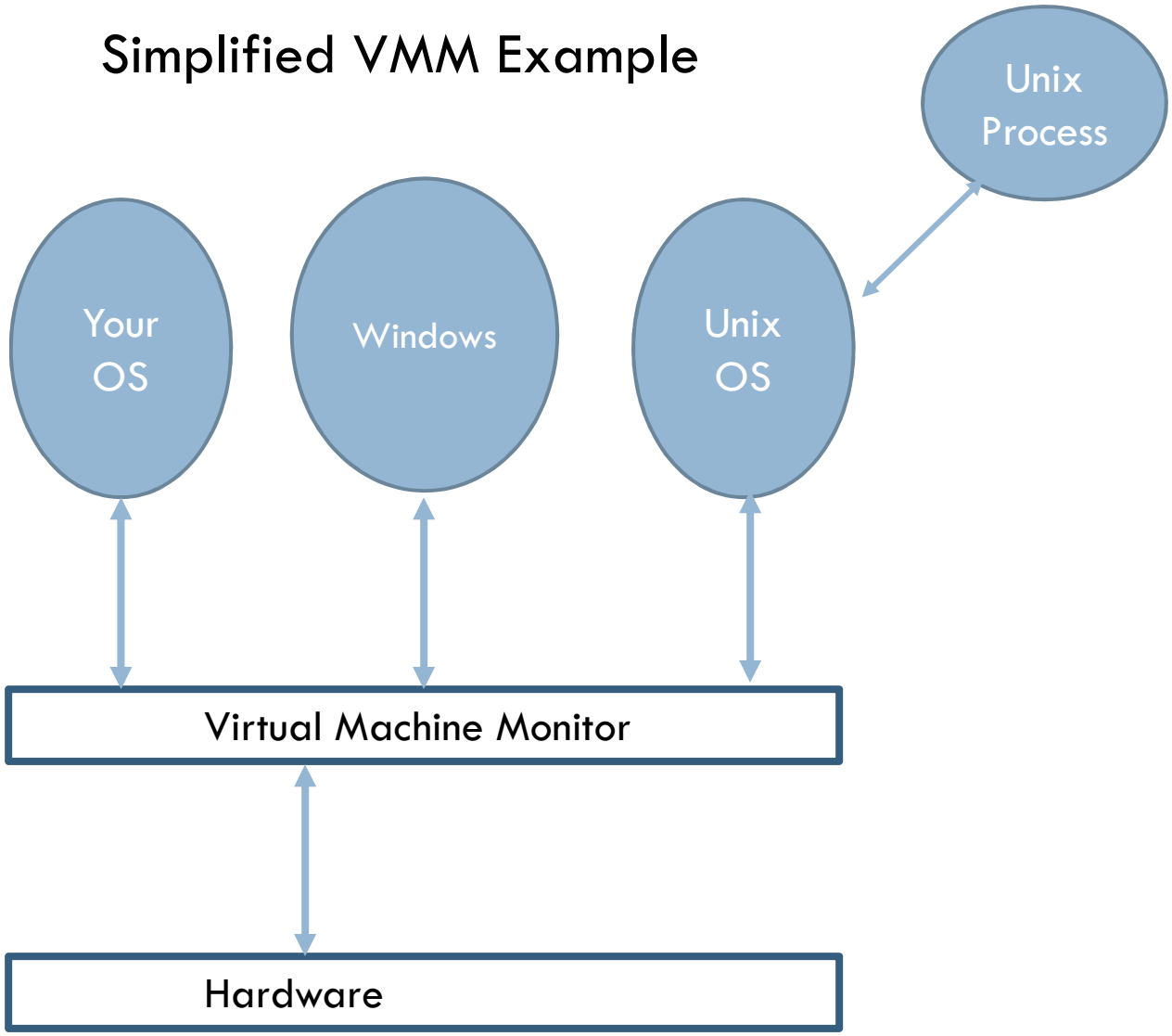
What is a NUMA?

- Non-uniform memory access
- Some memory is closer than other
- ccNUMA is cache-coherent NUMA

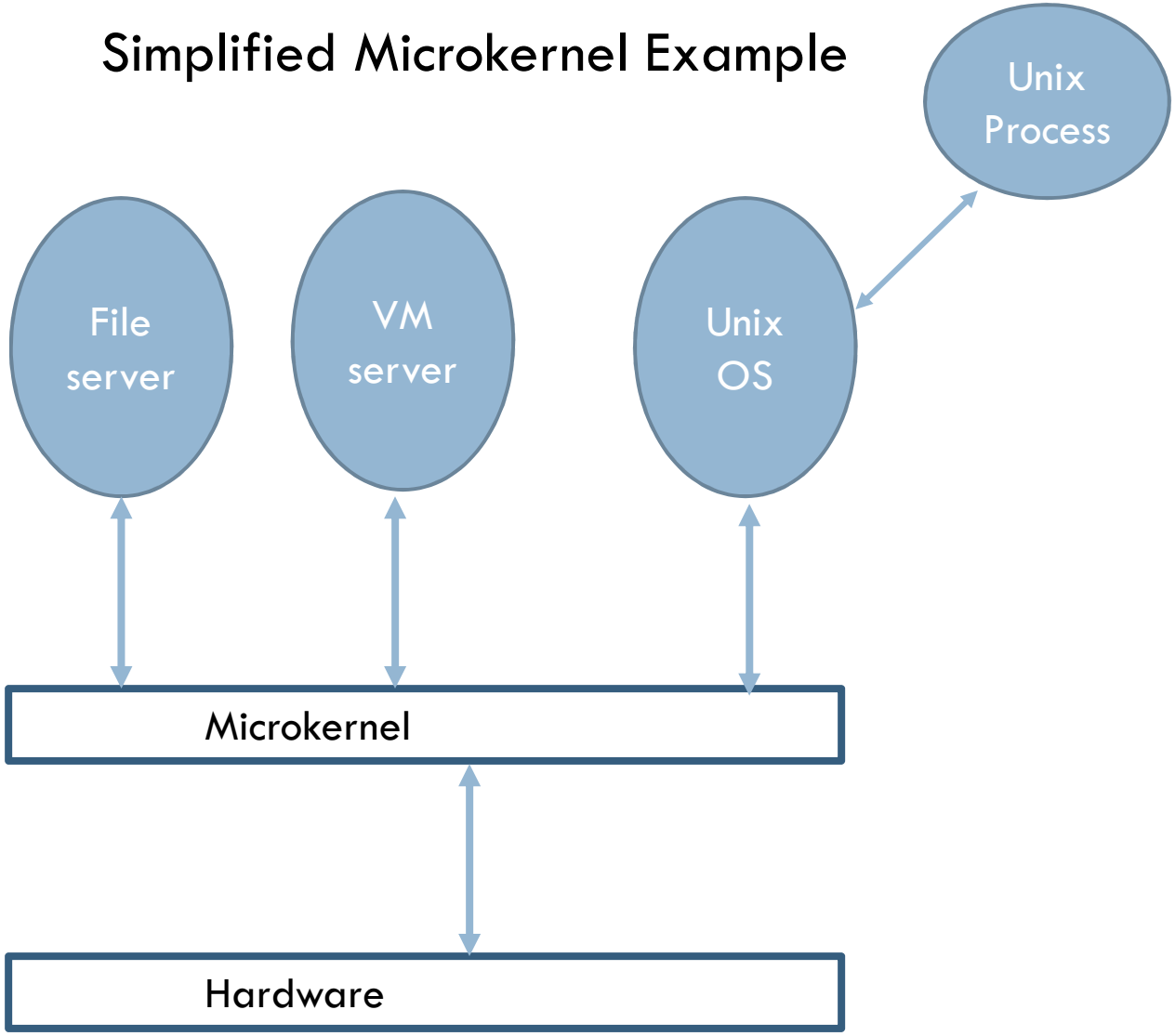
Note

- This was binary-compatible virtualization
  - OSs were compiled for the same hardware that was virtualized
  - Can you have an OS written for different hardware running on top of the VMM?

Simplified VMM Example



Simplified Microkernel Example



History of Virtual Machines

- Started at IBM in 1970s
  - Were used for time-sharing to multiplex expensive hardware
- Following development of Multics, IBM hurried to announce plans to build TSS, its time-sharing system
- Multics and TSS were late
- But IBM released a system CP/CMS
  - CP stands for "Control Program", i.e., a virtual machine monitor
  - CMS was a single-user operating system
  - This was their quick-and-dirty way to implement time-sharing
- CP/CMS was a precursor of IBM OS/360 and OS/390 – OS/390 still in use today on IBM mainframes

What are VMs used for?

- Time-sharing (1970s)
- Operating system debugging when hardware is expensive
- Running Windows office software
- Security: honeypots
- Multi-platform OS development (Solaris):
  - develop OS for virtual hardware platform
  - run on top of hypervisor
  - simplifies development and reduces the size of code tree
- What's the big one I've left out??

Back to Disco – System Implementation

- Multiple OSs run over Disco. *Don't have to be all the same OSs.*
- Single-system image is accomplished by *configuring the systems* as a cluster
- Machine resources are managed by the VMM and are *dynamically* allocated among virtual machines
- A virtual machine is the unit of scalability and the unit of fault containment: contains both software and hardware faults
- VMM code base is very small
  - 13K lines, 72 KB – it's replicated

DISCO system implementation

- Machine-wide data structures are partitioned such that they are located on the processor where they are likely to be accessed more often
- Wait-free synchronization used to improve scalability
- Inter-VM communication is done through shared memory
- Devices are virtualized: what does this mean?

DISCO system implementation

- Machine-wide data structures are partitioned such that they are located on the processor where they are likely to be accessed more often
- Wait-free synchronization used to improve scalability
- Inter-VM communication is done through shared memory
- Devices are virtualized: what does this mean?
  - All operations on devices are intercepted and emulated
- Non-privileged instructions are run directly on hardware, privileged instructions are emulated: what does this mean?

Definition of virtualizability

- By Goldberg, the father of virtual machines, 1974
- *For efficiency, most instructions execute natively. "Privileged instructions must be trapped and emulated: accessing processor state: status registers, TLB, I/O instructions"*

Disco system implementation (cont'd)

- Memory pages are migrated and replicated to ensure better locality
  - they use FLASH hardware counters to find out whether a page should be migrated or replicated
- Memory and disk are transparently shared
  - Block cache is shared
  - Disco uses copy-on-write disks

Performance results

- To get the overhead to its minimum, modifications to operating systems were required
- Main sources of performance overhead:
  - TLB reloading for scientific mostly user-level workloads
  - High TLB fault rate for unpredictable database workloads
  - Emulation overhead for pmake
  - Nonetheless, max overhead observed was 16%
- They showed better scalability with Disco compared to IRIX, a commercial SMP operation system
  - Is this a fundamental property of VMs? Or can IRIX be fixed by using better synchronization primitives?

Discussion

- What are the traditional problems with virtual machines
  - Virtualization overhead
    - E.g., memory usage was a problem (code and data of hosted OSs were replicated on the machine)
  - Resource management
    - don't know when a resource is no longer in use and can be taken away from a VM
  - Communication and sharing
    - old VMs could not communicate: a user could not start two virtual machines that accessed files on the same disk
- How did they solve these problems in Disco?

Discussion

- What are the traditional problems with virtual machines
  - Virtualization overhead
    - Code+data of identical OSs shared
  - Resource management
    - Change HAL to give hints to monitor about resource utilization (idle loop, page reclamation hints)
  - Communication and sharing
    - Use DS protocols to communicate; one VM mounts disk, others grab files via NFS

Discussion

- What performance optimizations helped the scalability of Disco?
  - Software TLB cache
  - Shared-memory communication for VMs
  - Replicated code
  - Partitioned data structures
  - Wait-free synchronization
  - Page migration and replication
  - Transparently shared memory buffer cache

Discussion

- Why didn't the world adopt this idea? Why are people building SMP operating systems?
  - One reason is that it is difficult to run parallel applications
  - Disco had to provide special support for parallel applications run on different virtual machines and share memory through these segments
- Why is Disco a good idea?
- Why is this a not-so-good idea?

Discussion

- What did you like about this paper? What are the good ideas you would use in future system designs?
- Throughout history, we have seen VMs being used as a "quick-and-simple" solution to complicated problems
  - IBM OS/360 for time-sharing, Disco for running over SMP hardware, VMWare Workstation for running Microsoft office apps by Unix geeks
- Do VMs have a place of their own, or do they simply serve as technology placeholders until the better technology comes around?

Are Virtual Machine Monitors Microkernels Done Right? (2005)

- What kind of paper is this?
  - Position paper
  - VMMs are the practical approach to systems research
  - VMMs enable innovation
  - VMMs produce the virtue of microkernels

Differences between VMMs and Microkernels

- Microkernels are prone to liability inversions
  - The OS depends on user-level components (e.g., pagers)
  - Avoiding deadlock is hard
- VMMs are designed to avoid such problems
  - Resource management is done in the VMM
  - VMs are isolated
- IPC performance dominates microkernel design, but isn't relevant to VMM
  - Inter-server communication is paramount in microkernel systems, but simply network communication in VMMs

Differences between VMMs and Microkernels

- Compatibility achieved in VMMs through guest OS's
  - In microkernels, it required huge compatibility libraries
- VMMs provide developer familiarity, which improves innovation

Similarities between VMMs and Microkernels

- The quest for narrow interfaces
- High-confidence in system security
- Investigate ilities, not performance
  - Security, reliability, extensibility

Are Virtual Machine Monitors Microkernels Done Right? (2006)

- What kind of paper is this?
  - New big idea?
  - Measurement paper?
  - Experiences/lessons learnt paper?
  - A system description?
  - Performance study?
  - Refute-conventional wisdom?
  - Survey paper?
  - Something else?

Are Virtual Machine Monitors Microkernels Done Right? (2006)

- What kind of paper is this?
  - Position
  - Response to other paper of the same name
  - Claims the other paper is erroneous and this paper corrects it

Definitions

- VMM: “software which transforms the single machine interface into the illusion of many”
- Microkernel: “to minimize the kernel and to implement whatever possible outside of the kernel”

Goals

- VMM: software reliability, data security, alternative system APIs, improved and new mechanisms
- Microkernels: flexibility, extensibility, fault isolation, mantainability, and restricted interdependencies

Revisiting Hand's Assertions

- Liability inversion
  - Claim Xen suffers from identical problem
  - Both cite Parallax as example to prove their point
  - Hand says that Parallax only affects other client VMs
  - Heiser says that this is the same as a failure of an L4 server
- IPC performance
  - Xen uses Dom0 for drivers, so all I/O requires IPC (simple asynchronous unidirectional event mechanism) with Dom0
  - CPU load of Dom0 dominates all other CPU load, so it **is** performance critical
- VMMs provide familiar environment
  - L4 Linux was presented many years ago, apparently satisfying the same criteria

Hype and Virtue (2007)

- What kind of paper is this?
  - New big idea?
  - Measurement paper?
  - Experiences/lessons learnt paper?
  - A system description?
  - Performance study?
  - Refute-conventional wisdom?
  - Survey paper?
  - Something else?

Hype and Virtue (2007)

- What kind of paper is this?
  - Position paper
  - Hypervisors are leading OS research astray
  - X86 ABI is just a bad interface for most things
  - Long-term value focus; not short-term product focus

Two kinds of VMM research

- Building a better hypervisor
- Neat VMM tricks: this entire class is simply exposing inadequacies of existing systems
  - Examples: replay debugging, honeypots, etc.

Nothing new is new (Much Ado section)

- Sharing and protection
  - VMM research here is simply rehash of old work
- Communication:
  - Normal networking or microkernel fast-paths rehashed
- Abstraction
  - HW as the new OS abstraction (it's a bad one)

HW as an API

- More code between an app and the real hardware
- HW interface is not necessarily simple (e.g., x86 MMU)

The Plan

- Use hypervisors as an excuse to really experiment with OS design
- Give up on backward compatibility and really explore

Suggested areas

- New APIs
  - Assume something other than single-threaded C programs
  - Take advantages of concurrency, transactional memory, MMUs, etc.
- Implementation techniques
  - Use theorem proving and other techniques to make precise claims about what your kernel is doing
- Find killer apps that require fundamentally new OS technology
- Metrics for OS scalability and/or isolation