
Disco: Running Commodity Operating Systems on Scalable Multiprocessors

Bugnion, Devine, Govil, Rosenblum,
Stanford University, 1997



Outline

- Virtualization
- Disco description
- Disco performance
- Discussion

Virtualization

- A software environment that behaves ~identically to real hardware.
 - Run most instructions natively for speed
 - VMM has full control of hardware
 - Intercept and emulate “dangerous” instructions to fool guest OS
- Many architectures are not virtualizable



Virtualizability

- “Formal Requirements for Virtualizable Third Generation Architectures”, Popek and Goldberg, 1974
- An architecture is virtualizable if the set of *sensitive* instructions is a subset of the set of *privileged* instructions

Virtualizability

- Behavior sensitive
 - Instructions whose behavior depends on processor mode/"configuration"
- Control sensitive
 - Instructions that change processor mode/"configuration"
- Privileged
 - Instructions that trap while in user mode

Non-Virtualizable Architectures

- x86
 - Several instructions can read system state in CPL=3 without trapping
 - AMD-V traps into host mode when such operations are attempted in guest mode
- MIPS
 - KSEG0 bypasses TLB, reads physical memory directly

Non-Virtualizable?

- Binary rewriting
 - At load/run time, patch offending instructions with a trap or emulation
 - VMWare (x86)
- Paravirtualization
 - Modify guest OS to not use offending instructions
 - Xen (x86), Disco (MIPS)
- Emulation, Dynamic recompilation, ...
 - Not really virtualization



Disco: Motivation

- Cache-coherent Non-Uniform Memory Access system
 - Single address space
- Stanford FLASH
 - MIPS R10000 processor and RAM per node
 - “Low-latency, high-bandwidth” interconnection network
 - Designed to scale to 1000+ nodes
 - Real hardware still doesn't exist
- To utilize “innovative” hardware, in this case, the FLASH CC-NUMA machine.

Goals

- Use the machine with **minimal effort**
 - Handle NUMA efficiently
- Overcome traditional VM overheads
 - Memory and disk use due to duplication
 - Slow inter-VM communication
- Fault tolerance
 - Extend the Virtual Machine Monitor to a cellular architecture. Not part of this work.

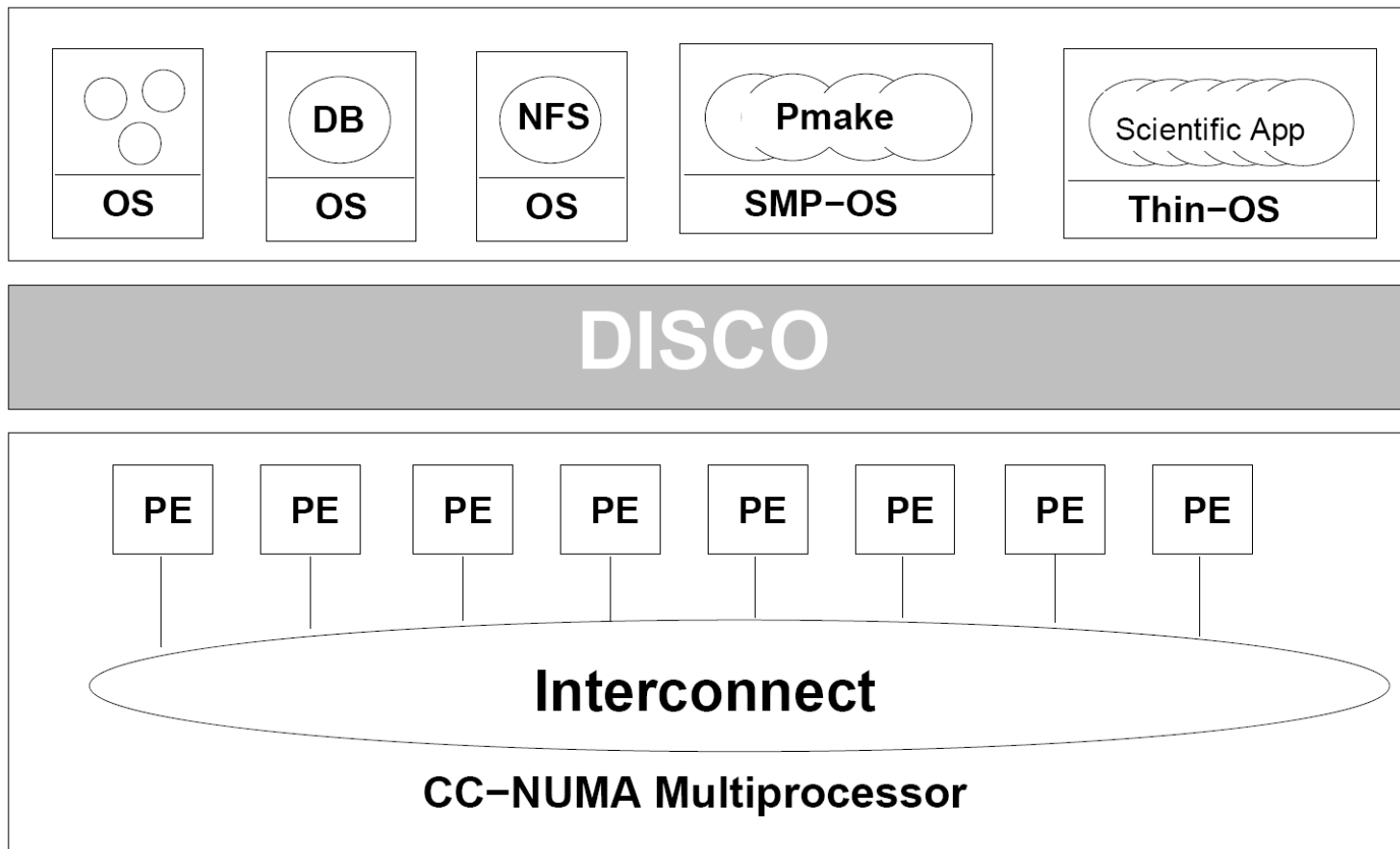
Why a Virtual Machine Monitor?

- Smaller and simpler
 - Thus easier to be correct
 - Less effort
 - Easier to scale, adapt to new hardware
- More efficient than unoptimized OS
 - If it handles NUMA and large-scale multiprocessing
 - Although less efficient than a well-optimized OS

Why a Virtual Machine Monitor?

- Secondary benefits
 - Keeps existing application base
 - Run multiple operating systems simultaneously
 - Software isolation: “security”
- **Efficiency with minimal effort**

Disco



- Disco is a Virtual Machine Monitor

Disco

- Virtual CPU
 - Paravirtualization
- Virtual Memory system
 - NUMA optimizations
 - Dynamic page migration and replication
- Virtual Disks
- Virtual Network Interface
 - Fast communication by memory mapping



Virtual CPU

- MIPS isn't virtualizable
 - KSEG0 bypasses TLB: Guest OS reads physical memory
- Solution: Paravirtualization by modifying IRIX
 - This is the only *necessary* change to make virtualization work
- Optimization: Guest OS CPU needs?
 - OS idle loop uses WAIT instruction to enter low power mode. Handled by Disco to deschedule virtual CPU
- Optimization: affinity scheduling

Virtual Memory

- Two-level mapping
 - Virtual addresses to “Physical” address via Guest OS
 - “Physical” to “Machine” address via Disco *pmap*
 - Real TLB stores Virtual -> Machine mapping
 - Optimization: Software cache of Virtual->Machine mappings (“l2tlb”)
- TLB flush when virtual CPU changes
 - Can be avoided if there was a “physical” to “machine” ASID map

Virtual Memory

- Optimization: Page replication and migration hides NUMA-ness from guest OS

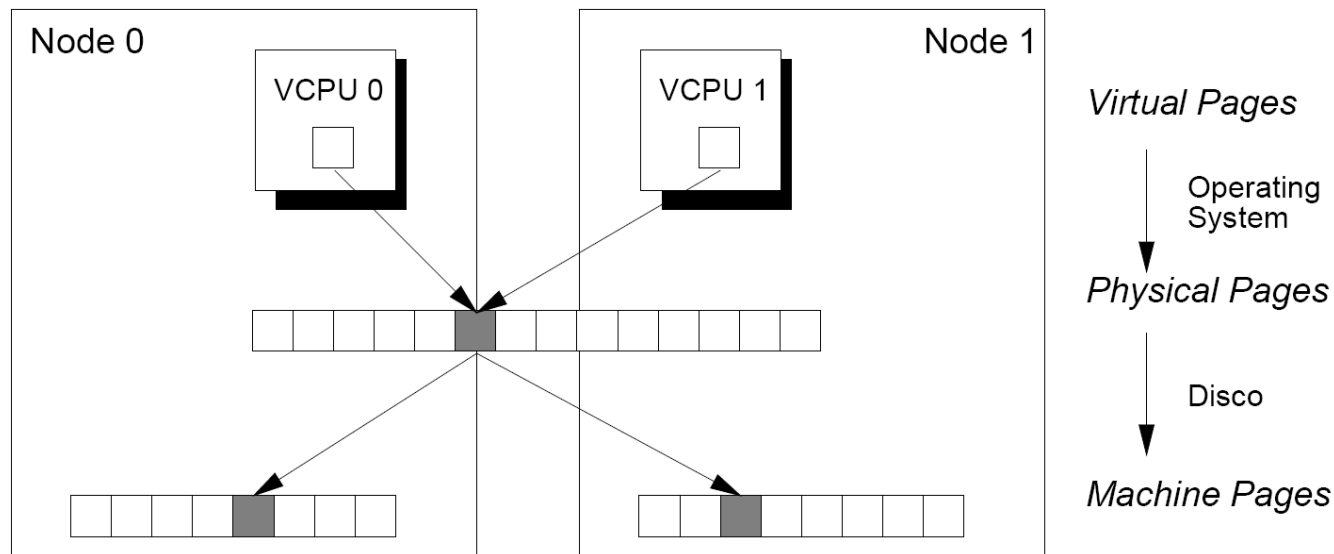


Fig. 2. Transparent Page Replication

Virtual I/O

- Virtual I/O Devices
 - Special device drivers written for guest OS rather than emulating the hardware
- Virtual DMA
 - DMA requests are also mapped from “Physical” to “Machine” addresses by Disco
 - Optimization: Disco keeps track of pages for sharing between nodes

Virtual Disk

- Persistent disks are not shared
 - Sharing done using NFS
- Non-persistent disks are shared copy-on-write
 - Disco watches DMA requests to disk devices
 - Blocks previously read are mapped instead of re-read
 - Disk buffer cache shared because DMA from disk to buffer cache results in a remap instead

Virtual Network

- When sending data between nodes, Disco intercepts DMA and remaps when possible
 - IRIX NFS implementation changed to ask for remap when copying received network data to buffer cache

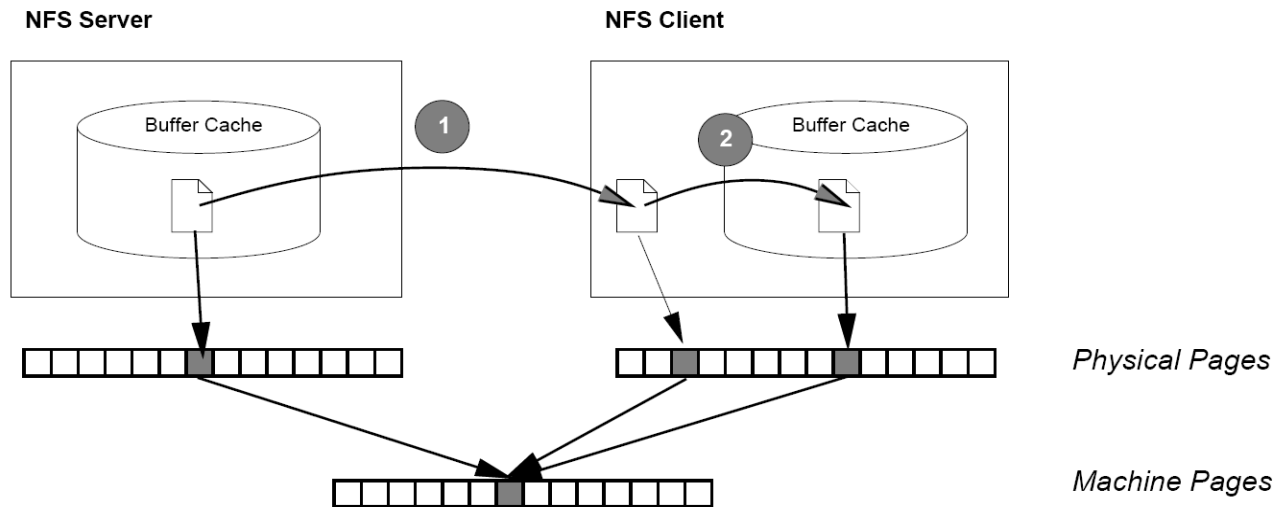


Fig. 5. Example of transparent sharing of pages Over NFS

Performance: Virtual CPU

- “Modest” virtualization overhead
 - Pmake overhead due to OS services, others due to TLB mapping

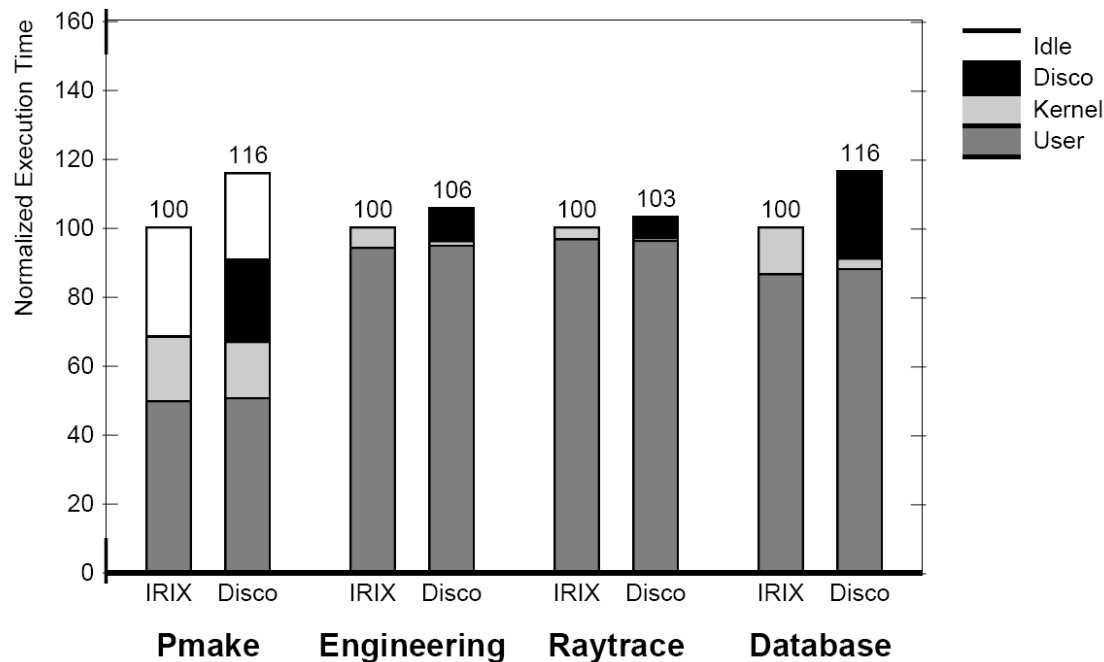


Fig. 6. Overhead of virtualization

Performance: Virtual Memory

- Fancy remapping works to share code and buffer cache

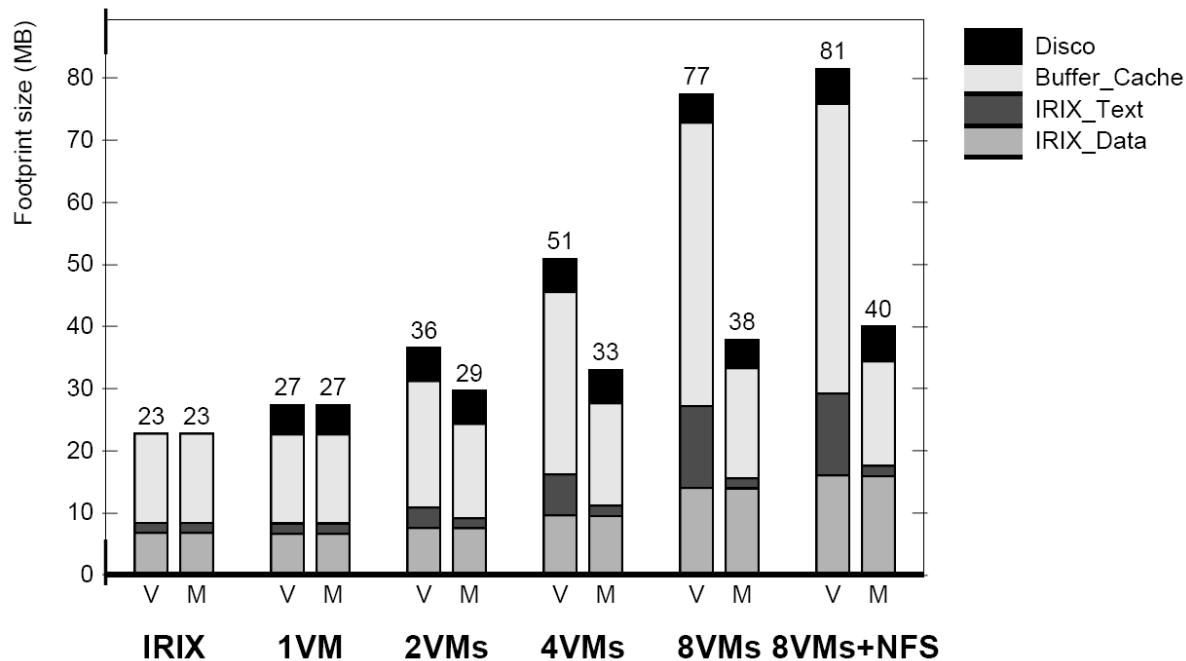
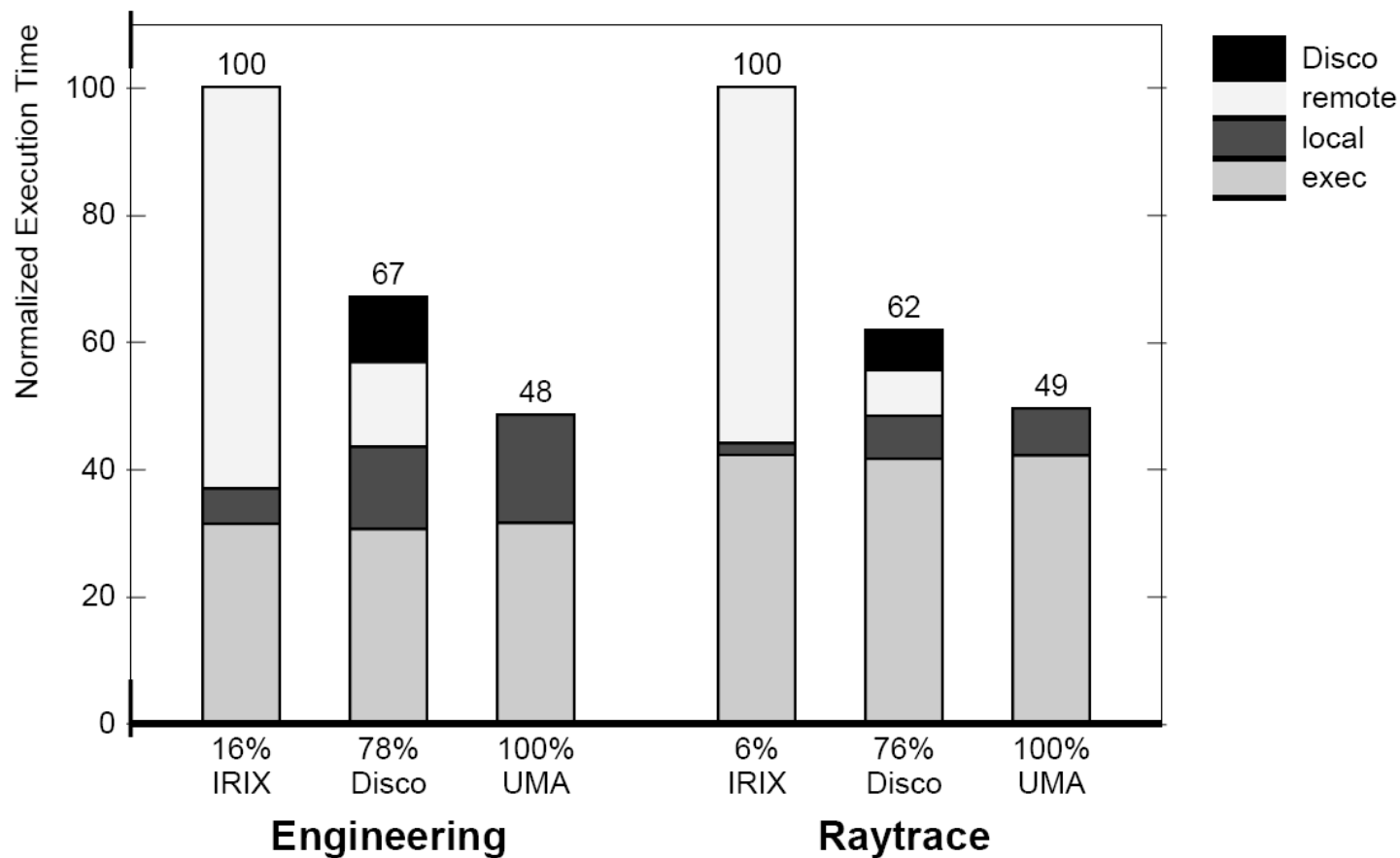


Fig. 7. Data sharing in Disco between virtual machines

Performance: NUMA

- Disco hides NUMA-ness from unoptimized OS



Performance: Scalability

- Disco outperforms an unoptimized OS

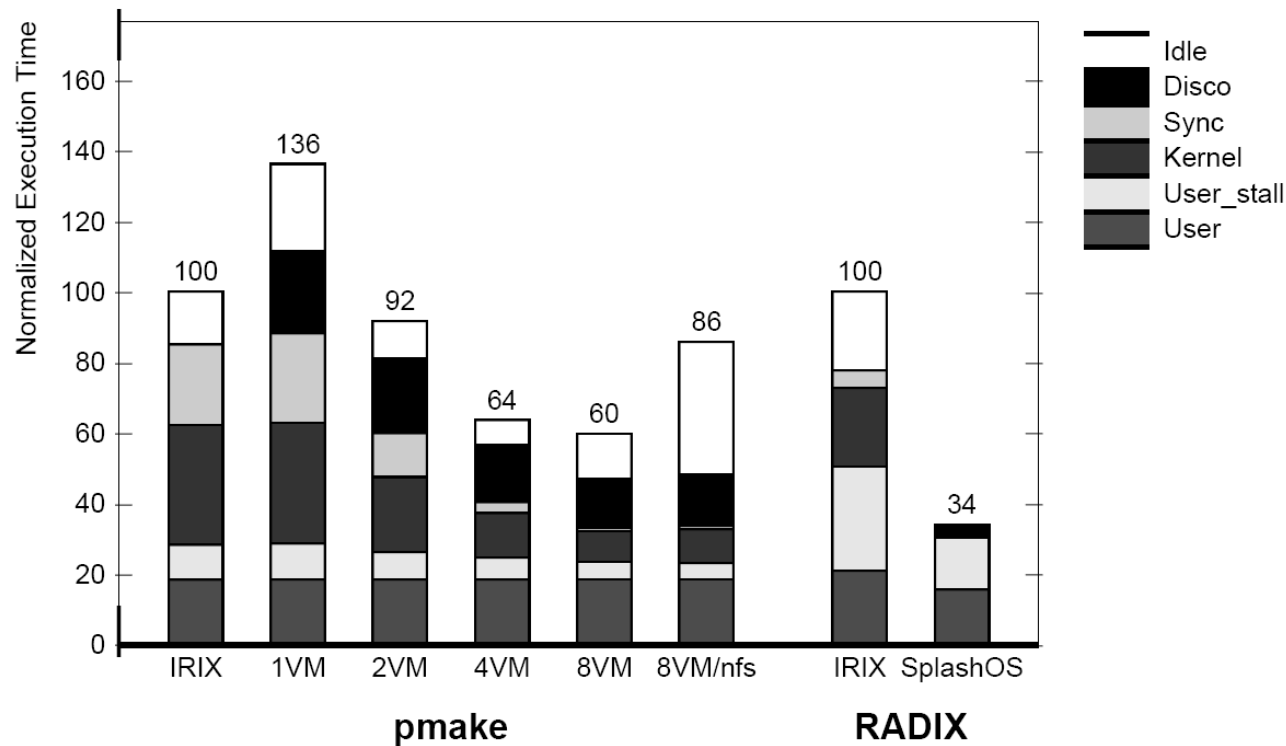


Fig. 8. Workload scalability under Disco

Conclusion


- Disco VMM hides NUMA-ness from unoptimized OS
- Disco VMM hides large-scale multiprocessor from unoptimized OS
- Disco VMM is low(er) effort
- Most changes to IRIX were for performance

Wisdom from the World of Networking

- (6) It is easier to move a problem around (for example, by moving the problem to a different part of the overall network architecture) than it is to solve it.
 - (6a) (corollary). It is always possible to add another level of indirection.
 - R. Callon, “The Twelve Networking Truths”, RFC1925, April 1 1996.

Question: Does FLASH Exist Yet?

- My guess is no.



The logo features the word "Stanford" in red, "FLASH" in blue with a jagged, lightning-bolt-like font, and "Multiprocessor" in red. A yellow lightning bolt graphic is positioned behind the "FLASH" text.

Welcome to the FLASH Home Page

[What's new?](#) (most recent: 03/11/98)

This site provides information about the design of the FLASH (FLexible Architecture for SHared memory) multiprocessor and about related work here at [Stanford](#).

Questions: Related Work

- Hive
 - Stanford cellular OS for FLASH for fault tolerance
- Hurricane
 - UofT “hierarchically clustered” OS for Hector shared memory multiprocessor
- Probably didn't compare to these because running simulations are painful.

Questions: Changes to IRIX

- Replaced some privileged register accesses in HAL to read/write to special page of memory
 - Writes directly into the “shadowed” data structures instead of trapping, which is expensive

Questions: Overcommitting

- Overcommitting resources
 - I'd say it's possible, and would have its uses, much like paging to disk when virtual memory was introduced
 - It's probably a better idea to use some disk backing store, rather than really overcommit

