Advanced Topics in Operating Systems, CS262

Prof. Eric A. Brewer

Lecture Notes, 10/2/98
Disco Virtual Machines for Scalable SMPs

I. Disco

Basic idea: convert a large-scale SMP into a collection of (variable sized) virtual machines that run concurrently, but can run different OSs including commercial OSs.

Why is this useful?

- A single commercial OS typically can't effectively use a large SMP, but will eventually. (takes a long time to develop highly parallel OS)
- Can mix OSs, especially a commercial one with smaller much faster specialized OSs
- Can better share resources than the big OS can, and can largely hide NUMA properties

Key concept:

- Emulate a complete machine including CPU, programmed I/O and DMA, MMU and TLB.
- Then run real OSs and applications on the VM instead of the real machine.

Issues: performance hit and ability to share among VMs

What is the OS/HW interface?

1) emulate R10000

- simulate all instructions: most are directly executed, privileged instructions must be emulated, since we won't run the OS in privileged mode (Disco runs privileged, OS runs supervisor mode, apps in user mode)
- an OS privileged instruction causes a trap which causes Disco to emulated the intended instruction
- map VCPUs onto real CPU: registers, hidden registers

2) MMU and physical memory

- virtual memory -> virtual physical memory -> machine memory
- VTLB is a Disco data structure, maps VM -> PM
- TLB holds the "net" mapping from VM -> MM, by combining VTLB mapping with Disco's page mapping, which is PM -> MM
- on TLB instruction on the VCPU, Disco gets trap, updates the VTLB, computes the real TLB entry by combined VTLB mapping with internal PM->MM page table (taking

the permission bits from the VTLB instruction)

- Must flush the real TLB on VM switch
- Somewhat slower:
  - OS now has TLB misses (not direct mapped)
  - TLB flushes are frequent
  - TLB instructions are now emulated
- Disco maintains a second-level cache of TLB entries: this makes the VTLB seem larger than a regular R10000 TLB. Disco can thus absorb many TLB faults without passing them through to the real OS

3) I/O (disk and network)

- emulated all programmed I/O instructions
- can also use special Disco-aware device drivers (simpler)
- main task: translate all I/O instructions from using PM addresses to MM addresses

Optimizations:

- larger TLB
- copy-on-write disk blocks
  - track which blocks already in memory
  - when possible, reuse these pages by marking all versions read-only and using copy-on-write if they are modified
  - => shared OS pages and shared executables can really be shared.
- zero-copy networking along fake "subnet" that connect VMs within an SMP. Sender and receiver can use the same buffer (copy on write)