Disco: Running Commodity Operating Systems on Scalable Multiprocessors

Edouard Bugnion, Scott Devine, and Mendel Rosenblum, Stanford University, 1997

Presented by James Loope
Outline

- Virtual Machine Monitors
- Disco description
- Disco performance
- Conclusions
What's a NUMA?

- Cache-coherent Non-Uniform Memory Access
- CPUs see “local” and “remote” memory space
Only prototype OS's run on a NUMA
  - Hive, Hurricane, Cellular IRIX
OS development lags behind hardware
Extensive modifications required to existing OS's to run NUMA
That's hard, and could make things buggy
Goals

- Handle NUMA without extensive OS changes
- Particularly SGI IRIX on prototype Stanford FLASH
- Reduce the overheads
  - Memory and disk duplication
  - Slow cross VM communication
  - Emulation overhead
Virtual Machine Monitor

- A software layer that behaves like hardware
  - Intercepts and emulates instructions
  - Hides atypical system architecture from OS
- Commonly called a “Hypervisor”
Virtual Machine Monitor

- It's not perfect, not everything is virtualizable
- Binary patching
  - At load/run time, patch offending instructions with a trap or emulation
- VMWare Server/Desktop
- Paravirtualization
  - Modify guest OS to not use unvirtualizable instructions
- Xen, ESX, Disco, KVM
Disco

PE = Processor + Memory
Direct execution
- Guest OS runs in supervisor mode
  - Access to a “supervisor” memory segment
  - No privileged mode or physical memory access
- Virtual processors time shared across physical
- Data structure stored for each Virtual processor
  - Process state, TLB
- Privileged mode instructions trap to disco monitor for emulation of instruction
Virtual Memory

- Memory mapping
  - Virtual addresses to “Physical” address by Guest
  - “Physical” to “Machine” address via Disco
    \textit{pmap}
  - TLB stores Virtual to Machine mapping
  - Software cache of Virtual to Machine mappings (second layer TLB)
- TLB flush when virtual CPU changes
  - Avoids having to keep track of and virtualize ASIDs
Virtual Memory

- Page replication and migration hides NUMA from guests

Fig. 2. Transparent Page Replication
It puts the kernel where?

- Oops! MIPS isn't fully virtualizable
  - KSEG0 segment access bypasses TLB
  - Can't access from supervisor mode
  - IRIX puts the kernel code and data there
- Solution is to modify IRIX
  - Put the kernel in a place we can virtualize
  - Paravirtualized
Virtual I/O

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

- User disks are not shared
  - Sharing done via NFS
- Root disk is shared copy-on-write
  - Disco watches DMA requests to disk devices
  - Blocks previously read are mapped instead of re-read
- Disk buffer cache shared
Virtual Network

- When sending data via NFS, Disco intercepts DMA and remaps to avoid duplication

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

- Virtualization Overhead
  - Pmake overhead due to OS services; TLB emulation cost

![Graph showing overhead of virtualization](image-url)
Memory Performance

- Remapping reduces machine memory use

Fig. 7. Data sharing in Disco between virtual machines
NUMA vs UMA Performance

- Disco optimizes IRIX on NUMA through page migration and replication

![Graph showing performance comparison between Disco and UMA]

- Engineering: 16% IRIX (100), Disco (67), UMA (48)
- Raytrace: 6% IRIX (100), Disco (62), UMA (49)
Scalability Performance

- Disco outperforms an unoptimized OS

![Graph showing performance comparison between Disco and an unoptimized OS](image)

Fig. 8. Workload scalability under Disco
Conclusion

- Disco hides NUMA from OS
- Disco hides large-scale multiprocessor from OS
- Disco is fairly simple, no huge OS modifications
- Performance data may be sketchy
  - It was all simulated – FLASH doesn't exist
References

- http://lse.sourceforge.net/numa/faq/