Disco: Running Commodity Operating Systems on Scalable Multiprocessors

Bugnion et al.

Presented by: Ahmed Wafa
Overview

- Shared Memory Multiprocessor Machines
- System Software for Shared Memory Multiprocessor
- Disco as a virtual machine monitor
- Performance
- Conclusions
Shared Memory Multiprocessors

- UMA (Uniform Memory Access)
- NUMA (Non Uniform Memory Access)
- ccNUMA (Cache-Coherent NUMA)
The Stanford FLASH Multiprocessor

Fig1. The FLASH system architecture
System Software for Shared Memory Multi-processors

• Invest a large development effort for designing/developing custom OS

• Statically Partition the machine and run commodity systems that communicate together using distributed protocols

• Run multiple commodity systems on top of a virtual machine monitor that provide dynamic resource sharing
Disco: A Virtual Machine Monitor

Fig. 2 Disco's Architecture
Challenges facing Virtual Machines

- Overheads.
- Resource Management.
- Communication and Sharing.
Disco

- Interface.
- Implementation.
- Running Commodity Operating systems.
Disco's Interface

- Processors
  - Virtual CPUs that abstracts the MIPS R10000
- Physical Memory
  - Contiguous space starting at address 0
  - Near Uniform memory access time
- I/O Devices
  - Virtualize access to devices.
  - Virtual subnet to allow communication
Disco's Implementation

- Virtual CPUs
- Virtual Physical Memory
- NUMA Memory Management
- Virtual I/O Devices
- Copy-on-write Disks
- Virtual Network Interfaces
Implementation: Virtual CPUs

• Direct Execution when possible
• MIPS modes
  – Kernel
  – Supervisor
  – User
Implementation: Virtual Physical Memory

- Fast and Correct Mapping from the VM virtual address space to the real machine address space
- Handling of TLB misses and the pmap data structures
- TLB miss cost and Disco's Second-Level TLB
Implementation: NUMA Memory Management

- Satisfying Cache misses from local memory
- Page Migration and Replication
- FLASH cache miss counter and Hot Pages
- Disco memmap and TLB shootdowns

Fig. 3 Page Replication
Implementation: I/O

- **Virtual I/O Devices**
  - Intercepting device access and DMA requests
  - Special device drivers, single trap per request

- **Copy-on-Write Disks**
  - Used for shared disks
  - Speed up disk reads by sharing memory

---

![Fig. 4 Copy-on-Write Page Sharing](image-url)
Virtual Network Interface

- Virtual subnet to allow VM communication
- No MTU limit for packets
- Aligned messages that span complete pages will be re-mapped rather than copied
- Sharing vs Replicating for maximizing data locality

Fig. 5 NFS sharing, replacing bcopy to remap data
Running Commodity Operating Systems

- IRIX 5.3 a SVR4 based UNIX from SG.
- Changes for IRIX
  - MIPS and KSEG0
  - Device Drivers
  - HAL
  - Network mbuf sharing
Experimental Results

- Setup
- Execution Overheads
- Memory Overheads
- Scalability
- Dynamic Page Migration and Replication
Experimental Setup

- The FLASH machine was not available at the time of development
- Hardware emulation using SimOS
Execution Overheads

- 3-16% slowdown
  - TLB misses
  - System calls

![Fig.2 Virtualization Overhead](image)

![Fig.6 Service Breakdown for pmake workload](image)
Memory Overheads

Fig. 7 Service Breakdown for pmake workload
Scalability

- IRIX and memory management
- Using 8 VM(s) reduces delay by 40%

Fig. 8 Workload Scalability Under Disco
Dynamic Page Migration and Replication

- The NUMA problem
- IRIX (non NUMA aware) vs DISCO

Fig.9 Performance Benefits of Page Migration
Conclusions

- Developing system for software for Shared Memory Multiprocessor machines can cost much.
- Disco can present such a machine as a cluster of connected machines through virtualization.
- Disco will allow commodity operating system to be used on such machines.
Conclusions

• Virtualization overhead can be as low as 3%-16%
• Optimization techniques can enhance scalability and hide NUMAness from commodity OS
References

• “The Stanford FLASH multiprocessor”, kuskin et al.