Running Commodity Operating Systems on Scalable Multiprocessors

Edouard Bugnion, Scott Devine and Mendel Rosenblum

Presentation by Mark Smith
Agenda

- Introduction
- Virtual Machine Monitor
- Virtual Machine Design
- Performance and Validation
- Conclusions
Introduction

- Stanford University project, late 90’s
- Paper’s authors went on to found VMWare
- Goal: efficiently run operating systems on large-scale shared memory multi-processor
  - Porting entire OS for each implementation is too expensive
  - Costs of OS porting limits hardware innovation
  - NUMA (non-uniform memory access) architecture
    - Maintaining cache coherence across shared memory has high overhead
Microkernel to Virtual Machine

Microkernel exports OS abstractions
- threads, IPC, memory -

Virtual machine exports HW abstractions
- processor, memory, i/o devices -

Images courtesy Wikipedia
Virtualizes machine resources and exports a conventional hardware interface to the OS

Global policies for resource management and load balancing

A single unit of fault containment

Other benefits of Virtual Machine Monitors
  ◦ Multiple, independent OS running side-by-side
  ◦ Support older OS and their legacy applications
Disco Architecture

Designed for FLASH multiprocessor system
MIPS R10000 processor
Nodes connected with high-performance interconnects
Directory based cache-coherency
Virtual Physical Memory

- Present guest OS with a virtual memory space starting at address 0
  - Provides a simulation of machine memory
  - Disco maintains the physical-to-machine address mapping

- MIPS software-reload TLB (performance)
  - Initial seed of TLB physical-to-machine address managed by Disco
  - Further updates automatically managed by hardware
Virtual Physical Memory

Establish initial TLB entry

Guest OS

Insert virtual-physical into TLB

Disco

Translate Address
- pmap lookup table contains each vm’s page entries

Insert modified entry into TLB

Hardware

Subsequent memory access

Guest OS

Virtual memory address

Disco

Normal TLB Lookup

Hardware
Performance aspects

- Team did not want to virtualize MIPS TLB address space identifiers so Disco flushes TLB on MMU context switch (i.e. scheduling a new virtual CPU)

- Second-level software TLB caches the recently used address translations
  - TLB miss handler first consults cache and if found inserts cached entry into TLB
  - If not found, forward TLB miss exception to OS
Virtual CPU

- Virtual CPU directly executes instructions on real CPU

- To schedule a virtual CPU
  - Set PC (program counter) on real machine registers to the virtual CPU PC
  - Jump to current PC of the virtual CPU
  - Start direct execution on physical CPU

- Simple scheduler allows virtual CPU to be time-shared across multiple physical CPU

- Guest OS not allowed privileged instructions
  - Processor placed in supervisor mode for OS execution
  - Privilege operations include TLB modifications, physical memory access, device I/O
Virtual CPU

Emulation of privileged instructions

- A Disco data structure maintains virtual CPU state:
  - Saved registers
  - Privileged registers
  - TLB contents of the virtual CPU

- Traps: system call, page fault, bus errors
  - Handled by Disco for virtual machine emulation
  - Disco updates virtual CPU privilege registers and jumps to the virtual machine’s trap vector
NUMA Memory Management

- Dynamic page migration and page replication
- Locality improved by moving frequently accessed pages closer to the originating node
- Monitored with FLASH cache miss counters
  - Identify “hot” page
  - Algorithm avoids moving pages too frequently
  - Choose migrate or replicate strategy
NUMA Memory Management

- Migrate page
  - Invalidate TLB entries for old machine
  - Move data to the local machine page
  - Change physical–machine mapping

- Replicate page
  - Downgrade all TLB entries to read–only status
  - Copy page to the local node
  - Update relevant TLB entries
- Single virtual machine
- Virtual CPU 0 and 1 map to same “physical” page
- Disco transparently replicates page to local node for better locality
Copy-on-Write Disk (Network Storage)

- Special I/O device drivers added to OS
  - Monitor calls pass command args in single trap
  - All DMA requests require translation of “physical” to “machine” addresses
  - DMA intercepts allow sharing of disk and memory across multiple virtual machines

- Disk access requests
  - If new read request is already in memory, can bypass disk access – fulfilled by page map updates
  - Flags page as read-only
  - Disk requests that are multiples of a page are mapped directly into the vm’s “physical” memory
Copy-on-Write Disk (Network Storage)

Global buffer cache filled by first read
Performance & Validation

- SimOS machine simulator to evaluate
- Virtualization overhead
- Memory overhead
- Scalability
- Page Migration
Performance & Validation
Virtualization Overhead

![Graph showing normalized execution time for different workloads (Pmake, Engineering, Raytrace, Database) under IRIX and DISCO.]
# Performance & Validation

## Memory Overhead

<table>
<thead>
<tr>
<th>Operating System Service</th>
<th>% of System Time (IRIX)</th>
<th>Avg Time per Invocation (IRIX)</th>
<th>Slowdown on Disco</th>
<th>Relative Execution Time on Disco</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Kernel Execution</td>
</tr>
<tr>
<td>DEMAND_ZERO</td>
<td>30%</td>
<td>21 µs</td>
<td>1.42</td>
<td>0.43</td>
</tr>
<tr>
<td>QUICK_FAULT</td>
<td>10%</td>
<td>5 µs</td>
<td>3.17</td>
<td>1.27</td>
</tr>
<tr>
<td>open</td>
<td>9%</td>
<td>42 µs</td>
<td>1.63</td>
<td>1.16</td>
</tr>
<tr>
<td>UTLB_MISS</td>
<td>7%</td>
<td>0.035 µs</td>
<td>1.35</td>
<td>0.07</td>
</tr>
<tr>
<td>write</td>
<td>6%</td>
<td>12 µs</td>
<td>2.14</td>
<td>1.01</td>
</tr>
<tr>
<td>read</td>
<td>6%</td>
<td>23 µs</td>
<td>1.53</td>
<td>1.10</td>
</tr>
<tr>
<td>execvc</td>
<td>6%</td>
<td>437 µs</td>
<td>1.60</td>
<td>0.97</td>
</tr>
</tbody>
</table>
Data Sharing in Disco

Workload Scalability
Performance & Validation
Benefits of Page Migration

![Bar chart showing normalized execution time for Engineering and Raytrace applications with different percentages of DISCO, remote, local, and exec.
Summary

- Virtual Machine Monitors is not a new idea but Disco provides several key enhancements
  - Transparently share major data structures (global buffer cache)
  - Emulation of DMA engine
  - Abstraction of NUMA to UMA for guest OS

- Overhead from virtualization is modest

- Reliable, scalable solution for multiprocessor shared memory systems