Disco

Robert Grimm
New York University
The Three Questions

- What is the problem?
- What is new or different?
- What are the contributions and limitations?
Background: ccNUMA

- Cache-coherent non-uniform memory architecture
  - Multi-processor with high-performance interconnect
- Non-uniform memory
  - Global address space
  - But memory distributed amongst processing elements
- Cache-coherence
  - Issue: How to ensure that memory in processor caches is consistent
  - Solutions: Bus snooping, directory
- Targeted system: FLASH, Stanford’s own ccNUMA
The Challenge

- Commodity OS’s not well-suited for ccNUMA
  - Do not scale
    - Lock contention, memory architecture
  - Do not isolate/contain faults
    - More processors ➞ more failures

- Customized operating systems
  - Take time to build, lag hardware
  - Cost a lot of money
The Disco Solution

- Add a virtual machine monitor (VMM)
  - Commodity OS’s run in their own virtual machines (VMs)
    - Communicate through distributed protocols
  - VMM uses global policies to manage resources
    - Move memory between VMs to avoid paging
    - Schedules virtual processors to balance load
Virtual Machines: Challenges

- **Overheads**
  - Instruction execution, exception processing, I/O
  - Memory
    - Code and data of hosted operating systems
    - Replicated buffer caches

- **Resource management**
  - Lack of information
    - Idle loop, lock busy-waiting
    - Page usage

- **Communication and sharing**
  - Not really a problem anymore b/c of distributed protocols
Disco in Detail
Interface

- MIPS R10000 processor
  - All instructions, the MMU, trap architecture
  - Extension to support common operations through memory
    - Enabling/disabling interrupts, accessing privileged registers

- Physical memory
  - Contiguous, starting at address 0

- I/O devices
  - Virtual devices exclusive to VM
  - Physical devices multi-plexed by Disco
  - Special abstractions for SCSI disks and network interfaces
    - Virtual subnet across all virtual machines
Virtual CPUs

- **Three modes**
  - **Kernel mode:** Disco
    - Provides full access to hardware
  - **Supervisor mode:** Guest operating system
    - Provides access to special memory segment (used for optimizations)
  - **User mode:** Applications

- **Emulation by direct execution**
  - Not for privileged instructions, direct access to physical memory and I/O devices
    - Emulated in VMM
    - Recorded in per VM data structure (registers, TLB contents)
    - Some traps (syscall, page fault) handled by guest OS’s trap handlers
Virtual Physical Memory

- Adds level of translation: Physical-to-machine
  - Performed in software-reloaded TLB
  - Based on pmap data structure: Entry per physical page
- Requires changes in IRIX memory layout
- Flushes TLB when scheduling different virtual CPUs
  - MIPS TLB is tagged (address space ID)
    - Avoids virtualizing ASIDs
- Increases number of misses, but adds software TLB
  - Guest operating system also mapped through TLB
  - TLB is flushed on virtual CPU switches
  - Virtualization introduces overhead
NUMA Memory Management

- Heavily accessed pages moved to using node
- Read-only shared pages duplicated across nodes
- Based on cache miss counting facility of FLASH
- Supported by memmap data structure
  - Entry per machine page
    - Points to virtual machines & addresses using page, also copies
Virtual I/O Devices

- Specialized interface for common devices
  - Special drivers for guest OS’s: rely on single trap
- DMA requests are modified
  - Physical to machine memory
- Copy-on-write disks
  - Remap pages that are already in memory
    - Decreases memory overhead, speeds up access
Virtual Network Interface

- **Issue:** Different VMs communicate through standard distributed protocols (here, NFS)
  - May lead to duplication of data in memory
- **Solution:** Virtual subnet
  - Ethernet-like addresses, no maximum transfer unit
  - Read-only mapping instead of copying
  - Supports scatter/gather
- What about NUMA?
Running Commodity Operating Systems

- IRIX 5.3
  - Changed memory layout to make all pages mapped
  - Device drivers for special I/O devices
    - Disco’s drivers are the same as IRIX’s. Sound familiar?
  - Patched HAL to use memory loads/stores instead of privileged instructions
  - Added new calls
    - To request zeroed-out memory pages
    - To inform Disco that page has been freed
  - Changed mbuf management to be page-aligned
  - Changed bcopy to use remap (with copy-on-write)
Evaluation
Experimental methodology

- FLASH machine “unfortunately not yet available”
- Use SimOS
  - Models hardware in enough detail to run unmodified OS
  - Supports different levels of accuracy, checkpoint/restore
- Workloads
  - pmake, engineering, scientific computing, database
Execution Overheads

- Uniprocessor configuration comparing Irix & Disco
  - Disco overheads between 3% and 16% (!)
    - Mostly due to trap emulation and TLB reload misses
### Diggin’ Deeper

<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>execve</td>
<td>6%</td>
<td>437 µs</td>
<td>1.60</td>
<td>0.97</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>TLB Write Emulation</th>
<th>Other Privileged Instructions</th>
<th>Monitor Calls &amp; Page Faults</th>
<th>Kernel TLB Faults</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEMAND_ZERO</td>
<td>0.21</td>
<td>0.16</td>
<td>0.47</td>
<td>0.16</td>
</tr>
<tr>
<td>QUICK_FAULT</td>
<td>0.80</td>
<td>0.56</td>
<td>0.00</td>
<td>0.53</td>
</tr>
<tr>
<td>open</td>
<td>0.08</td>
<td>0.06</td>
<td>0.02</td>
<td>0.30</td>
</tr>
<tr>
<td>UTLB_MISS</td>
<td>1.22</td>
<td>0.05</td>
<td>0.00</td>
<td>0.02</td>
</tr>
<tr>
<td>write</td>
<td>0.24</td>
<td>0.21</td>
<td>0.31</td>
<td>0.17</td>
</tr>
<tr>
<td>read</td>
<td>0.13</td>
<td>0.09</td>
<td>0.01</td>
<td>0.20</td>
</tr>
<tr>
<td>execve</td>
<td>0.03</td>
<td>0.05</td>
<td>0.17</td>
<td>0.40</td>
</tr>
</tbody>
</table>

- **What does this table tell us?**
- **What is the problem with entering/exiting the kernel?**
- **What is the problem with placing the OS into mapped memory?**
Memory Overheads

- Workload: Eight instances of basic pmake
- Memory partitioned across virtual machines
  - NFS configuration uses more memory than available
Scalability

- IRIX: High synchronization and memory overheads
  - memlock: spinlock for memory management data structures
- Disco: Partitioning reduces overheads
- What about RADIX experiment?
Page Migration and Replication

- What does this figure tell us?
What Do You Think?