Lecture 23: Virtual Memory, Multiprocessors

• Today’s topics:
  ▪ Virtual memory
  ▪ Multiprocessors, cache coherence
Virtual Memory

• Processes deal with virtual memory – they have the illusion that a very large address space is available to them

• There is only a limited amount of physical memory that is shared by all processes – a process places part of its virtual memory in this physical memory and the rest is stored on disk (called swap space)

• Thanks to locality, disk access is likely to be uncommon

• The hardware ensures that one process cannot access the memory of a different process
Address Translation

- The virtual and physical memory are broken up into pages

8KB page size

Virtual address

Translated to physical page number

Physical address
Memory Hierarchy Properties

- A virtual memory page can be placed anywhere in physical memory (fully-associative)

- Replacement is usually LRU (since the miss penalty is huge, we can invest some effort to minimize misses)

- A page table (indexed by virtual page number) is used for translating virtual to physical page number

- The page table is itself in memory
TLB

• Since the number of pages is very high, the page table capacity is too large to fit on chip

• A translation lookaside buffer (TLB) caches the virtual to physical page number translation for recent accesses

• A TLB miss requires us to access the page table, which may not even be found in the cache – two expensive memory look-ups to access one word of data!

• A large page size can increase the coverage of the TLB and reduce the capacity of the page table, but also increases memory waste
TLB and Cache

• Is the cache indexed with virtual or physical address?
  - To index with a physical address, we will have to first look up the TLB, then the cache → longer access time
  - Multiple virtual addresses can map to the same physical address – must ensure that these different virtual addresses will map to the same location in cache – else, there will be two different copies of the same physical memory word

• Does the tag array store virtual or physical addresses?
  - Since multiple virtual addresses can map to the same physical address, a virtual tag comparison can flag a miss even if the correct physical memory word is present
Cache and TLB Pipeline

Virtually Indexed; Physically Tagged Cache
Bad Events

• Consider the longest latency possible for a load instruction:
  ▪ TLB miss: must look up page table to find translation for v.page P
  ▪ Calculate the virtual memory address for the page table entry that has the translation for page P – let’s say, this is v.page Q
  ▪ TLB miss for v.page Q: will require navigation of a hierarchical page table (let’s ignore this case for now and assume we have succeeded in finding the physical memory location (R) for page Q)
  ▪ Access memory location R (find this either in L1, L2, or memory)
  ▪ We now have the translation for v.page P – put this into the TLB
  ▪ We now have a TLB hit and know the physical page number – this allows us to do tag comparison and check the L1 cache for a hit
  ▪ If there’s a miss in L1, check L2 – if that misses, check in memory
  ▪ At any point, if the page table entry claims that the page is on disk, flag a page fault – the OS then copies the page from disk to memory and the hardware resumes what it was doing before the page fault … phew!
Multiprocessor Taxonomy

- SISD: single instruction and single data stream: uniprocessor
- MISD: no commercial multiprocessor: imagine data going through a pipeline of execution engines
- SIMD: vector architectures: lower flexibility
- MIMD: most multiprocessors today: easy to construct with off-the-shelf computers, most flexibility
Memory Organization - I

- Centralized shared-memory multiprocessor or Symmetric shared-memory multiprocessor (SMP)
- Multiple processors connected to a single centralized memory – since all processors see the same memory organization → uniform memory access (UMA)
- Shared-memory because all processors can access the entire memory address space
- Can centralized memory emerge as a bandwidth bottleneck? – not if you have large caches and employ fewer than a dozen processors
SMPs or Centralized Shared-Memory
Memory Organization - II

• For higher scalability, memory is distributed among processors → distributed memory multiprocessors

• If one processor can directly address the memory local to another processor, the address space is shared → distributed shared-memory (DSM) multiprocessor

• If memories are strictly local, we need messages to communicate data → cluster of computers or multicomputers

• Non-uniform memory architecture (NUMA) since local memory has lower latency than remote memory
Distributed Memory Multiprocessors

Processor & Caches

Memory

I/O

Memory

Interconnection network

Processor & Caches

Memory

I/O

Memory

Processor & Caches

Memory

I/O

Processor & Caches

Memory

I/O
SMPs

• Centralized main memory and many caches $\rightarrow$ many copies of the same data

• A system is cache coherent if a read returns the most recently written value for that word

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th>Value of X in Cache-A</th>
<th>Cache-B</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>-</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>CPU-A reads X</td>
<td>1</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>CPU-B reads X</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>CPU-A stores 0 in X</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Cache Coherence

A memory system is coherent if:

- P writes to X; no other processor writes to X; P reads X and receives the value previously written by P

- P1 writes to X; no other processor writes to X; sufficient time elapses; P2 reads X and receives value written by P1

- Two writes to the same location by two processors are seen in the same order by all processors – write serialization

- The memory consistency model defines “time elapsed” before the effect of a processor is seen by others
Cache Coherence Protocols

• Directory-based: A single location (directory) keeps track of the sharing status of a block of memory

• Snooping: Every cache block is accompanied by the sharing status of that block – all cache controllers monitor the shared bus so they can update the sharing status of the block, if necessary

  ➢ Write-invalidate: a processor gains exclusive access of a block before writing by invalidating all other copies
  ➢ Write-update: when a processor writes, it updates other shared copies of that block
Design Issues

- Three states for a block: invalid, shared, modified
- A write is placed on the bus and sharers invalidate themselves
Title

• Bullet