Last Time: Synchronization

Before:

Tail

Higher Addresses

| y | x |

Head

After:

Tail

Higher Addresses

| y |

Higher Addresses

T1 code (producer)

ORi R1, R0, x ; Load x value into R1
LW R2, tail(R0) ; Load queue tail into R2
SW R1, 0(R2) ; Store x into queue
ADDi R2, R2, 4 ; Shift tail by one word
SW R2 0(tail) ; Update tail memory addr

T2 & T3 (2 copies of consumer thread)

spin: LW R4, tail(R0) ; Load queue tail into R4
BEQ R4, R3, spin ; If queue empty, wait
LW R5, 0(R3) ; Read x from queue into R5
ADDi R3, R3, 4 ; Shift head by one word
SW R3 head(R0) ; Update head memory addr

Critical section: T2 and T3 must take turns running red code.
Today: Memory System Design

Multiprocessor memory systems: Consequences of cache placement.

Write-through cache coherency: Simple, but limited, approach to multiprocessor memory systems.

NUMA and Clusters: Two different ways to build very large computers.
Two CPUs, two caches, shared DRAM ...

CPU0:
LW R2, 16(R0)

CPU1:
LW R2, 16(R0)
SW R0, 16(R0)

View of memory no longer "coherent".

Loads of location 16 from CPU0 and CPU1 see different values!

Today: What to do ...
The simplest solution ... one cache!

CPUs do not have internal caches.

Only one cache, so different values for a memory address cannot appear in 2 caches!

Multiple caches banks support read/writes by both CPUs in a switch epoch, unless both target same bank.

In that case, one request is stalled.
Not a complete solution ... good for L2.

For modern clock rates, access to shared cache through switch takes 10+ cycles.

Using shared cache as the L1 data cache is tantamount to slowing down clock 10X for LWs. Not good.

This approach was a complete solution in the days when DRAM row access time and CPU clock period were well matched.
Modified form: Private L1s, shared L2

Thus, we need to solve the cache coherency problem for L1 cache.

Advantages of shared L2 over private L2s:

- Processors communicate at cache speed, not DRAM speed.
- Constructive interference, if both CPUs need same data/instr.

Disadvantage: CPUs share BW to L2 cache ...
supports a 1.875-Mbyte on-chip L2 cache. Power4 and Power4+ systems both have 32-Mbyte L3 caches, whereas Power5 ... unit, LSU = load/store unit, IFU = instruction fetch unit, FPU = floating-point unit, and MC = memory controller).
Sequentially Consistent Memory Systems
Recall: Sequential Consistency

Sequential Consistency: As if each thread takes turns executing, and instructions in each thread execute in program order.

<table>
<thead>
<tr>
<th>T1 code (producer)</th>
<th>T2 code (consumer)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORi R1, R0, x</td>
<td>LW R3, head(R0)</td>
</tr>
<tr>
<td>LW R2, tail(R0)</td>
<td>LW R4, tail(R0)</td>
</tr>
<tr>
<td>SW R1, 0(R2)</td>
<td>BEQ R4, R3, spin</td>
</tr>
<tr>
<td>ADDi R2, R2, 4</td>
<td>LW R5, 0(R3)</td>
</tr>
<tr>
<td>SW R2 0(tail)</td>
<td>ADDi R3, R3, 4</td>
</tr>
<tr>
<td></td>
<td>SW R3 head(R0)</td>
</tr>
</tbody>
</table>

Legal orders: 1, 2, 3, 4 or 1, 3, 2, 4 or 3, 4, 1, 2 ... but not 2, 3, 1, 4!
Sequential consistency requirements ...

1. Only one processor at a time has write permission for a memory location.

The “sequential” part of sequential consistency.

2. No processor can load a stale copy of a location after a write.

The “consistent” part of sequential consistency.
Implementation: Snoopy Caches

Each cache has the ability to "snoop" on memory bus transactions of other CPUs.

The bus also has mechanisms to let a CPU intervene to stop a bus transaction, and to invalidate cache lines of other CPUs.
Writes from 10,000 feet ...

For write-thru caches ...

1. Writing CPU takes control of bus.
2. Address to be written is invalidated in all other caches.
   Reads will no longer hit in cache and get stale data.
3. Write is sent to main memory.
   Reads will cache miss, retrieve new value from main memory.

To a first-order, reads will "just work" if write-thru caches implement this policy.

A "two-state" protocol (cache lines are "valid" or "invalid").
Limitations of the write-thru approach

Every write goes to the bus.

Total bus write bandwidth does not support more than 2 CPUs, in modern practice.

Write-back big trick: keep track of whether other caches also contain a cached line. If not, a cache has an “exclusive” on the line, and can read and write the line as if it were the only CPU. For details, take CS 252 ...

CS 152 L26: Synchronization
Other Machine Architectures
Each CPU has part of main memory attached to it.

To access other parts of main memory, use the interconnection network.

For best results, applications take the non-uniform memory latency into account.

Good for applications that match the machine model...
Connect large numbers of 1-CPU or 2-CPU rack mount computers together with high-end network technology.

Instead of using hardware to create a shared memory abstraction, let an application build its own memory model.

University of Illinois, 650 2-CPU Apple Xserve cluster, connected with Myrinet (3.5 µs ping time - low latency network).
Clusters also used for web servers

In some applications, each machine can handle a net query by itself.

Example: serving static web pages. Each machine has a copy of the website.

Load manager is a special-purpose computer that assigns incoming HTTP connections to a particular machine. Image from Eric Brewer’s IEEE Internet Computing article.
Clusters also used for web services

In other applications, many machines work together on each transaction.

Example: Web searching. The search is partitioned over many machines, each of which holds a part of the database.

Altavista web search engine did not use clusters. Instead, Altavista used shared-memory multiprocessors. This approach could not scale with the web.
CS 152: What’s left ...

- **Monday 5/2:** Final report due, 11:59 PM
  Watch email for final project peer review request.

- **Thursday 5/5:** Midterm II, 6 PM to 9 PM, 320 Soda.
  No class on Thursday. Review session in Tuesday 5/2, + HKN (???).

- **Tuesday 5/10:** Final presentations.
  Deadline to bring up grading issues: Tues 5/10@ 5PM. Contact John at lazzaro@eecs
Tomorrow: Final Project Checkoff

TAs will provide “secret” MIPS machine code tests.

Bonus points if these tests run by 2 PM. If not, TAs give you test code to use over weekend