Multiprocessors, Multiprocessing, and Multithreading

more is better?

Classifying Multiprocessors

- Interconnection Network
- Memory Topology
- Programming Model

Interconnection Network

- pros/cons?
### Memory Topology

- **UMA** (Memory Access)
- **NUMA** (Memory Access)
- pros/cons?

![Memory Topology Diagram](image)

### Programming Model

- **Shared Memory** -- every processor can access every address location.
- **Message Passing** -- each processor can only access its local memory. Communication is through explicit messages.
- pros/cons?

![Programming Model Diagram](image)

### Parallel Programming

- **Processor A**
  - `index = i++;

- **Processor B**
  - `index = i++;

- **Example:** find the max of 100,000 integers on 10 processors.

### Multiprocessor Caches (Shared Memory)

- the problem -- cache coherency
- the solution?

![Multiprocessor Caches Diagram](image)
Cache Coherency

- **write-update**
  - on each write, each cache holding that location its value

- **write-invalidate** <= most common
  - on each write, each cache holding that location the cache line.

- both schemes MUCH easier on a bus-based multiprocessor
- potentially requires a LOT of messages, but...

Cache Coherency

- A good cache coherency protocol can avoid sending unnecessary (and expensive) invalidate or update messages.
- Allows each cache line to be in one of several *states*.
- MESI (Illinois)
  - modified
  - exclusive
  - shared
  - invalid

<table>
<thead>
<tr>
<th>State</th>
<th>Local Read</th>
<th>Local Write</th>
<th>External Read</th>
<th>External Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modified</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exclusive</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shared</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Invalid</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Cache Coherency

- How do you know when an external read/write occurs?
- Snooping protocols
- Directory protocols

Multithreading Processors

uniprocessor? multiprocessor?

conventional CPU

multithreaded CPU
Multithreaded Processors

- Coarse-grain multithreading (Alewife-MIT)
  - context switch at long-latency operations (cache misses)
- Fine-grain multithreading (Tera Supercomputer)
  - context switch every cycle
- Simultaneous multithreading (SMT) (Tullsen, Eggers, Levy 1995)
  - execute instructions from multiple threads in the same cycle
  - is only different from fine-grain multithreading in the context of superscalar execution
  - requires surprisingly few changes to a conventional out-of-order superscalar processor
  - Was announced to be featured in the next Compaq Alpha processor (21464)
  - Will be in future Intel processors (both server and desktop lines) – announced as “Hyper-threading technology.”

Multithreading models

<table>
<thead>
<tr>
<th>Conventional Superscalar</th>
<th>Fine-Grain Multithreading</th>
<th>Simultaneous Multithreading</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SMT Performance

- Simultaneous Multithreading
- Fine-Grain Multithreading
- Conventional Superscalar

Number of Threads

Multiprocessors -- Key Points

- Network vs. Bus
- Message-passing vs. Shared Memory
- Shared Memory is more intuitive, but creates problems for both the programmer (memory consistency, requiring synchronization) and the architect (cache coherency).
- Multithreading gives the illusion of multiprocessing (including, in many cases, the performance) with very little additional hardware.