Parallel Processors

Forecast

- Motivation
- Connect Processors At
  - Processor
  - I/O
  - Memory
- Symmetric Multiprocessors & Cache Coherence
- Taxonomies & Examples

Motivation

So far we have had one processor in a computer system

Why not use \textbf{N} processors to get

- Higher throughput via many jobs in parallel
- Improved cost-effectiveness (e.g., adding 3 processors may quadruple throughput for twice the system cost)
- To get lower latency from shrink-wrapped software (e.g., databases & web servers today, but more tomorrow)
- Lower latency through parallelizing your application (but this is not easy)

Where Should Processors Be Connected?

At Processor

- E.g., Single-Instruction Multiple-Data (SIMD)

At I/O System

- Clusters or Multicomputers

At Memory System

- Shared-Memory Multiprocessors
- Especially Symmetric Multiprocessors (SMPs)

Connect at Processor: SIMD
**Connect at Processor**

**SIMD Assessment**
- Amortizes cost of control unit over many datapaths
- Does not leverage current microprocessors
- Programming model has limited flexibility (compare w/ Multiple-Instruction Multiple-Data (*MIMD*))

Other Designs that join at processor
- Systolic, etc., have not been commercial success

---

**Connect at I/O**

Connect with Standard Network (e.g., with Ethernet)
- Called a **Cluster** (e.g., Beowulf Cluster)
- Adequate bandwidth, but high-latency communication
- Cheap, but “you get what you pay for”

Connect with Custom Network (e.g., IBM SP/[1-3])
- Sometimes called a **Multicomputer**
- Higher cost than cluster & poorer communication than multiprocessors (coming next)

---

**Connect at Memory: Multiprocessors**

**Shared Memory Multiprocessors**
- All processors can address all physical memory
- Demands evolutionary operating systems changes
- Higher throughput with no application changes
- Lower-latency, however, requires parallelization with synchronization

Most Successful: **Symmetric Multiprocessor (SMP)**
- 4-32 microprocessors on a bus
- Too much bus traffic so add caches
Cache Coherence

Snooping in SMP
- All requests broadcast on bus
- All processors & memory snoop & respond appropriately
- Cache blocks read/write at one processor or read-only at several

Larger Multiprocessors
- Eschew bus
- Use “level of indirection” through directory

Taxonomies

<table>
<thead>
<tr>
<th>In General [Flynn 1966]</th>
<th>single data</th>
<th>multiple data</th>
</tr>
</thead>
<tbody>
<tr>
<td>single instrn</td>
<td>SISD</td>
<td>SIMD</td>
</tr>
<tr>
<td>multiple instrn</td>
<td>MISD</td>
<td>MIMD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>For Shared Memory Multiprocessors</th>
<th>uniform memory access</th>
<th>non-uniform memory access</th>
</tr>
</thead>
<tbody>
<tr>
<td>cache coherence</td>
<td>CC-UMA</td>
<td>CC-NUMA</td>
</tr>
<tr>
<td>no cache coherence</td>
<td>NCC-UMA</td>
<td>NCC-NUMA</td>
</tr>
</tbody>
</table>

Example Commercial Systems

CC-UMA (SMP)
- Sun E10000: http://computer.org/micro/mi1998/m1039abs.htm

CC-NUMA

NCC-NUMA
- Cray T3E: http://www.cs.wisc.edu/~markhill/Misc/asplos96_t3e_comm.pdf

Clusters
- ASCI: http://www.llnl.gov/asci/platforms