Distributed Shared Memory and Machine Learning

CSci 8211
Chai-Wen Hsieh
11/5/2018
Overview of Distributed Shared Memory (DSM)

System performance:
- Lookup
- Action
Key Issues

1. **DSM algorithm**
   - how accesses actually executes

2. **Implementation level**
   - where the access is implemented

3. **Memory consistency model**
   - how to maintain consistent
DSM System Design Choices

- DSM algorithm
- Implementation level
- Memory consistency model
- Cluster configuration
- Interconnection network
- Structure of shared data
- Granularity of shared data
- Data compression?
DSM Systems and Algorithms

- **DSM systems**: all systems that provide shared memory abstraction on a distributed shared-memory system

- Basic problems:
  - Distribution of shared data
  - Coherent view of shared data
DSM Systems and Algorithms

- Two strategies: replication and migration
- Algorithm classifications

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
<th>Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRSW</td>
<td>Single reader/single writer</td>
<td>No replication, maybe migration</td>
</tr>
<tr>
<td>MRSW</td>
<td>Multiple reader/single writer</td>
<td>Read replication, invalidation</td>
</tr>
<tr>
<td>MRMW</td>
<td>Multiple reader/multiple writer</td>
<td>Full replication</td>
</tr>
</tbody>
</table>
# Implementation Level

<table>
<thead>
<tr>
<th>Software</th>
<th>User-level library, runtime system, OS kernel, language</th>
<th>1-8 Kb</th>
<th>More flexible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware</td>
<td>CC-NUMA COMA RMS</td>
<td>4-128 bytes</td>
<td>Faster searching and directory functions</td>
</tr>
<tr>
<td>Hybrid</td>
<td>various</td>
<td>16 bytes-8 Kb</td>
<td>Balance the cost-complexity trade-offs</td>
</tr>
</tbody>
</table>
Memory Consistency Model - The “trade-off”

- The legal ordering of memory references issued by a processor, as observed by other processors

<table>
<thead>
<tr>
<th>Memory consistency model</th>
<th>Strict</th>
<th>Loose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory consistency</td>
<td><img src="up.png" alt="increase" /></td>
<td><img src="down.png" alt="decrease" /></td>
</tr>
<tr>
<td>Access latency</td>
<td><img src="up.png" alt="increase" /></td>
<td><img src="down.png" alt="decrease" /></td>
</tr>
<tr>
<td>Bandwidth requirement</td>
<td><img src="up.png" alt="increase" /></td>
<td><img src="down.png" alt="decrease" /></td>
</tr>
<tr>
<td>Programming simplicity</td>
<td><img src="up.png" alt="increase" /></td>
<td><img src="down.png" alt="decrease" /></td>
</tr>
</tbody>
</table>
Memory Consistency Model - The “trade-off”

- Strong consistency models
  - *Sequential* consistency: the same sequence of reads and writes
  - *Processor* consistency: same sequence of writes

- More relaxed models
  - *Weak* consistency: consistent only on synchronization memory access
  - *Release* consistency: ordinary access between acquire/release pairs
  - *Lazy* release consistency: modifications wait until the next acquire
  - *Entry* consistency: use associated shared variable to protect protected shared variable
What Can We Do -1

● How to do parallelization for a particular application?
  ○ Analyze its access pattern
  ○ Split the job into several sub-jobs
  ○ Parallel, not sequential
  ○ Independent
  ○ More reads, less writes
What Can We Do -2

● Preprocessing the shared memory data
  ○ Predict next data migration/repetition in terms of
    ■ Usage
    ■ Size
    ■ Destination
  ○ Relocate/copy the data based on prediction
What Can We Do -3

- Weigh between concurrency and consistency
  - Examine application before runtime for best consistency model
  - During runtime, change model accordingly
    - Memory miss
    - Bottleneck
    - Data source
Papers


Papers
