Lecture 13 – Parallelism in Software IV
Credits

• Most of the slides courtesy Dr. Rodric Rabbah (IBM)
  – Taken from 6.189 IAP taught at MIT in 2007.
Patterns for Parallelizing Programs

4 Design Spaces

Algorithm Expression

• Finding Concurrency
  – Expose concurrent tasks

• Algorithm Structure
  – Map tasks to processes to exploit parallel architecture

Software Construction

• Supporting Structures
  – Code and data structuring patterns

• Implementation Mechanisms
  – Low level mechanisms used to write parallel programs

Quick recap

• Decomposition
  – High-level and fairly abstract
  – Consider machine scale for the most part
  – Task, Data, Pipeline
  – Find dependencies

• Algorithm structure
  – Still abstract, but a bit less so
  – Consider communication, sync, and bookkeeping
  – Task (collection/recursive)
  – Data (geometric/recursive)
  – Dataflow (pipeline/event-based-coordination)

• Supporting structures
  – Loop
  – Master/worker
  – Fork/join
  – SPMD
  – MapReduce
Map/Reduce Pattern

• Two phases in the program
• Map phase applies a single function to all data
  – Each result is a tuple of value and tag
• Reduce phase combines the results
  – The values of elements with the same tag are combined to a single value per tag -- \textit{reduction}
  – Semantics of combining function are associative
  – Can be done in parallel
  – Can be pipelined with map
• Google uses this for \textit{all} their parallel programs
Communication and Synchronization Patterns

• Communication
  – Point-to-point
  – Broadcast
  – Reduction
  – Multicast

• Synchronization
  – Locks (mutual exclusion)
  – Monitors (events)
  – Barriers (wait for all)
    • Split-phase barriers (separate signal and wait)
      – Sometimes called “fuzzy barriers”
    • Named barriers allow waiting on subset
## Algorithm Structure and Organization (from the Book)

<table>
<thead>
<tr>
<th></th>
<th>Task parallelism</th>
<th>Divide and conquer</th>
<th>Geometric decomposition</th>
<th>Recursive data</th>
<th>Pipeline</th>
<th>Event-based coordination</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SPMD</strong></td>
<td>****</td>
<td>***</td>
<td>****</td>
<td>**</td>
<td>***</td>
<td>**</td>
</tr>
<tr>
<td><strong>Loop Parallelism</strong></td>
<td>****</td>
<td>**</td>
<td>***</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Master/Worker</strong></td>
<td>****</td>
<td>**</td>
<td>*</td>
<td>*</td>
<td>****</td>
<td>*</td>
</tr>
<tr>
<td><strong>Fork/Join</strong></td>
<td>**</td>
<td>****</td>
<td>**</td>
<td>****</td>
<td>****</td>
<td>****</td>
</tr>
</tbody>
</table>

- Patterns can be hierarchically composed so that a program uses more than one pattern.
Algorithm Structure and Organization (my view)

<table>
<thead>
<tr>
<th>Pattern Type</th>
<th>Task parallelism</th>
<th>Divide and conquer</th>
<th>Geometric decomposition</th>
<th>Recursive data</th>
<th>Pipeline</th>
<th>Event-based coordination</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPMD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Loop Parallelism</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Master/Worker</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fork/Join</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Patterns can be hierarchically composed so that a program uses more than one pattern
## Algorithm Structure and Organization (my view)

<table>
<thead>
<tr>
<th></th>
<th>Task parallelism</th>
<th>Divide and conquer</th>
<th>Geometric decomposition</th>
<th>Recursive data</th>
<th>Pipeline</th>
<th>Event-based coordination</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SPMD</strong></td>
<td>****</td>
<td>**</td>
<td>****</td>
<td>**</td>
<td>****</td>
<td>*</td>
</tr>
<tr>
<td><strong>Loop Parallelism</strong></td>
<td>**** when no dependencies</td>
<td>*</td>
<td>****</td>
<td>*</td>
<td>****</td>
<td>SWP to hide comm.</td>
</tr>
<tr>
<td><strong>Master/Worker</strong></td>
<td>****</td>
<td>***</td>
<td>***</td>
<td>***</td>
<td>**</td>
<td>****</td>
</tr>
<tr>
<td><strong>Fork/Join</strong></td>
<td>****</td>
<td>****</td>
<td>**</td>
<td>****</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

- Patterns can be hierarchically composed so that a program uses more than one pattern