Summary of Key Points

Architectures and Applications

- SISD, SIMD, MIMD
- Shared Memory and Atomic actions
- Network Topologies
- Application classes
  * Multithreaded Systems
  * Distributed Systems
  * Parallel Computation
- Programming Patterns
  * Iterative parallelism
  * Recursive parallelism
  * Producers and consumers
  * Client/Server
  * Peers
Typical Applicability

<table>
<thead>
<tr>
<th>Programming Pattern</th>
<th>Application Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterative Parallelism</td>
<td>MT    Dist</td>
</tr>
<tr>
<td>Recursive parallelism</td>
<td>✔      ✔</td>
</tr>
<tr>
<td>Producer/Consumer</td>
<td>✔      ✔</td>
</tr>
<tr>
<td>Client/Server</td>
<td>✔      ✔</td>
</tr>
<tr>
<td>Peers</td>
<td>✔      ✔</td>
</tr>
</tbody>
</table>

Processes and Synchronization

- states, atomic actions, history
- partial and total correctness
- Independence
- Await statements — mutual exclusion and conditional synchronization
- Hoare Logic and Proof Outline Logic
- Semantics of co-statements and await-statements
- Interference of statements with assertions
- Avoiding interference
  - Disjoint variables
Concurrent Programming—Slide Set 14 Summary

Theodore Norvell

- Weakened assertions
- Global Invariants
- Ghost Variables

- Properties
  - Safety property
  - Liveness property
  - Fairness assumption

- Peterson’s algorithm as a case-study

**Locks and Barriers**

- Critical Sections
  - Coarse-grained solution
  - Hardware Solution
  - Peterson’s algorithm

- Barrier Synchronization
- Flag Synchronization Principles
- Data parallel algorithms.

**Semaphores**

- Semaphores as repositories for “rights” or “permits”.
Monitors

- Monitors are thread safe objects
- Islands of sequentiality
  - Mutual exclusion: At most one occupant.
- Condition variables (Signal and wait discipline)
  - wait: Allow threads to wait until a specific condition is true
  - signal: Allow one waiting thread in.
  - Formal rules for signal and wait.
- Other signalling disciplines
  - Signal and leave
  - Signal and urgent wait
  - Signal and continue

Message Passing

- Synchronous and asynchronous
- Used in
  - client/server structures.
  - peer/peer structures
  - pipeline and heartbeat structures
RPC and Rendezvous

- Remote procedure calls (method invocations)
  * Allow distribution without changing the structure of an application
  * Compared to local procedure calls
    - Take more time
    - No pass by reference
    - Change of data formats (if machines are inhomogeneous)
    - Subject to partial failure
    - May introduce concurrency and/or extra threads

- Rendezvous
  * Looks like PC/RPC from caller’ POV
  * Handled by daemon, when it is ready.

Interaction Paradigms

- Probe/echo
- Broadcast
- Logical clocks
Transaction Processing

- ACID
  - Atomicity
  - Consistency
  - Isolation
  - Durability

- Concurrent execution

- Error recovery via write ahead logs
  - Concurrency control
    - by locks
    - by timestamping
    - optimistic

- Deadlock: detection, prevention, resolution

- Distributed transactions
  - Concurrency control
  - Distributed commit

Transactional Memory

- Transactional code.
• Software Implementation
  * Optimistic: Check on commit
• Libraries for TM
• Caches for hardware support of TM

Model Checking

• Routes to trust.
  * Hand proof
  * Fully automated proof
  * Computer checked proof
  * Finite model checking
• Linear Temporal Logic, X, G, F, U
• SMV