Semantically Sequential, Parallel Execution of Programs on Multiprocessors

Gagan Gupta (Advisor: Gurindar S. Sohi)

Research Summary

Multiprocessors are ubiquitous, but programming them continues to be challenging.

**Our Goal**: Simplify multiprocessor programming without compromising performance

<table>
<thead>
<tr>
<th>Conventional Wisdom</th>
<th>Our Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order in programs obstructs parallelism</td>
<td>Order can help to expose parallelism!</td>
</tr>
<tr>
<td>Use non-deterministic programs, or make dataflow in programs explicit</td>
<td>Use <em>ordered</em> programs; maintain precise program-order execution semantics</td>
</tr>
<tr>
<td>Programmer should expose parallelism</td>
<td>Use <em>run-time dataflow</em> and <em>speculative</em> techniques to expose parallelism</td>
</tr>
</tbody>
</table>

**Benefits**:
- Simplified programming; Simplified system design; Better reliability
- Performance at par or better (5% to 288%) than conventional methods

Proposed System: ParaKram

- **A Runtime system** to manage the multiprocessor program’s execution
- Performs *out-of-order superscalar processor-like execution* on multicores
  - Uses data dependences to perform *dataflow* execution
  - Supports program-wide **precise exceptions** (overall order)
  - **Speculates** when dependences are unknown, but keeps ordered semantics

ParaKram

- **Sequencer**
  - Unrolls dynamic instances of tasks
  - Computes data set dynamically (user assisted)

- **Precise-restart Engine**
  - Tracks tasks and their order in a Reorder List
  - Checkpoints mod set in History Buffer
  - Retires task in program order

- **Dataflow Engine**
  - Uncovers parallelism past blocked tasks in the program
  - Constructs dynamic data dependence graph using write and read sets
  - Executes tasks out-of-order
  - If task dependences/order are unknown, speculates tasks are independent

- **Order-aware Scheduler**
  - Execute out-of-order speculatively
  - Declares dataset => misspeculated => rolled back using History Buffer
  - Execute out-of-order non-speculatively
  - Re-execute

Example speculative dataflow execution on 3 processors

ParaKram speedup is 288% higher than non-deterministic OpenMP, 75% over Cilk

ParaKram speedup (harmonic mean) is 20% higher than non-deterministic Pthreads (excludes Cholesky)

ParaKram scales with system size; Non-deterministic method does not scale

ParaKram speedup is up to 77% higher than non-deterministic Cilk and TL2 STM

Precisely restarting misspeculated task (F3 from above)