Implications of shared data synchronization techniques on multi-core energy efficiency

Ashok Gautham\(^{\gamma}\), Kunal Korgaonkar\(^{\gamma\phi}\), Patanjali Slpsk\(^{\gamma}\), Shankar Balachandran\(^{\gamma}\), Kamakoti Veezhinathan\(^{\gamma}\)

\(^{\gamma}\) IIT Madras, Chennai, India

\(^{\phi}\) IBM Research, Bangalore, India
What We Have Done

• We evaluated
  – Mutex, Spinlock and Software Transactional memory (STM) on a commodity multi-core
  – Sequential implementation also considered

• Synchronization techniques differ based on how they execute Critical Sections (CSs)

[Diagram showing Lock based Code and STM based Code with context notes: Thread waiting or spinning, Threads run critical sections in parallel as transactions]
Glimpse of Results

- Metrics used - Performance Per Watt (PPW), Energy Delay Product (EDP), and not just Performance

Results:
- The extent of savings by using STM (for one of the benchmarks) over best performing lock-based code
  - 224% speedup
  - 653% reduction in EDP
  - 213% improvement in PPW
# Behavior and Trends

<table>
<thead>
<tr>
<th></th>
<th>STM</th>
<th>Mutex</th>
<th>Spinlock</th>
<th>Sequential</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cores Utilization</strong></td>
<td>All cores kept busy at all times</td>
<td>Cores are underutilized; Presence of idle periods</td>
<td>Cores kept busy, but spinning (wasteful cycles)</td>
<td>Only one core used</td>
</tr>
<tr>
<td><strong>Core Sleep State Usage (energy saving states)</strong></td>
<td>None</td>
<td><strong>Power saving sleep states are used, but not the deepest sleep state</strong></td>
<td>None</td>
<td>Other cores in deepest sleep state at all times</td>
</tr>
<tr>
<td><strong>Duration</strong></td>
<td>Short</td>
<td>Long</td>
<td>Long</td>
<td>Long</td>
</tr>
<tr>
<td><strong>Power</strong></td>
<td>High</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td><strong>Energy</strong></td>
<td>Low</td>
<td>Still high</td>
<td>High</td>
<td>Low</td>
</tr>
</tbody>
</table>
Experimental setup

Linux – OS
(with core sleep state manager/governor)

Intel Sandy Bridge
Commodity Processor
With per-core sleep state support.

Intel

Standard Pthread
Library for Mutex

École Polytechnique
Fédérale de Lausanne

State of the art STM – Swiss TM
Experimentally proven to be best

STAMP TM Benchmark
Range of ‘time inside CS’ – 2% to 90%
Duration = Performance

Swiss vs MutexLock vs CGL vs Sequential - Time

CGL=Spinlock

Duration - Locks Better
EDP (= Performance and Energy)
Trading off Performance for Reduction in Avg. and Peak Power

Figure 3: STM Performance-Energy trade-off
Future Work

• We plan to cover a wider set of benchmarks
  – One of the questions we want to answer - Does there exist applications for which both lock and STM scale, but STM scale better?

• We also want to check how programs using mutex be made more energy efficient
  – by exploiting core ‘sleep states’ in a more fine grained and controlled manner

• For a program using STM, conflict rate still is a key workload parameter dictating the resultant performance and energy
  – A thorough quantitative analysis of such workload parameters is needed
Questions