Scalable Dynamic Task Scheduling on Adaptive Many-Cores

Vanchinathan Venkataramani, Anuj Pathania, Muhammad Shafique, Tulika Mitra, Jörg Henkel

Introduction: Many-Core Paradigm [Our Definition]

- Multi-Core
  - Few Cores
  - Memory Bus
  - Context Switching
  - Centralized Scheduler

- Many-Cores
  - Dozens of Cores
  - Network on Chip (NoC)
  - One Thread per Core
  - Distributed Scheduler

Introduction: Adaptive Many-Core

- Adaptive many-cores allow speedup for both single-threaded and multi-threaded tasks.

Introduction: Performance Maximization Problem

- Cores are limited; multiple active tasks want them.
- Goal is to maximize performance (Total Speedup).
- Give whom, how many?
- Solvable by Dynamic Programming optimally; but centralized.
- Our contribution: A Distributed Solution.
Introduction: Varying Speedup
- Speedup extracted by a task changes over its execution lifetime based on requirements.
- For malleable tasks improve performance by moving cores from low speedup tasks to high speedup tasks.

Introduction: Motivational Example
- Dynamic core reallocation of 4 cores between two tasks every scheduling epoch results in higher throughput compared to a static equal core allocation.
- Throughput –
  - 3.205 (Static)
  - 3.238 (Dynamic)

Introduction: Concavity in Speedup
- Speedup in task is monotonically increasing and submodular, due to TLP and ILP saturation.
- No stability without concavity; cyclic oscillations otherwise.

Algorithm: DPMS
- We present an algorithm called DPMS.
  - Uses a Multi-Agent System.
  - Uses a regression-based performance-prediction model also introduced in this work.
Algorithm: Convergence [Theorem 1] and Optimal [Theorem 2]

- System is guaranteed to converge to a solution in \( O(\text{Tasks}) \) number of steps.
- Converged solution is optimal.

Regression-Based Performance Prediction for ILP Tasks

- CPI = Steady-State CPI + Miss CPI

\[
P_S(C_p) = \left(\gamma_{\text{inst}} I_{\text{inst}} + \gamma_{\text{fp}} I_{\text{fp}} + \gamma_{\text{busy}} I_{\text{busy}}\right) / I
\]

- Estimate Steady-State CPI

\[
P_M(C_p) = \left(\gamma_{\text{inst}} I_{\text{inst}} + \gamma_{\text{fp}} I_{\text{fp}} + \gamma_{\text{mem}} I_{\text{mem}}\right) / I
\]

- Estimate Miss CPI

\[
I_{\text{busy}} = \beta_1 I_{\text{busy}} + \beta_2 I_{\text{inst}} + \beta_3 I_{\text{fp}} + \beta_4 I_{\text{br}} + \beta_5 I_{\text{mem}} + \beta_6
\]

- Estimate Steady CPI on Different Size CPU Allocation

\[
I_{\text{steady}} = \beta_1 I_{\text{steady}} + \beta_2 I_{\text{inst}} + \beta_3 I_{\text{fp}} + \beta_4 I_{\text{br}} + \beta_5 I_{\text{mem}} + \beta_6
\]

- Estimate Miss CPI on Different Size CPU Allocation

\[
I_{\text{miss}} = \beta_1 I_{\text{miss}} + \beta_2 I_{\text{inst}} + \beta_3 I_{\text{fp}} + \beta_4 I_{\text{br}} + \beta_5 I_{\text{mem}} + \beta_6
\]

Accuracy: %Error

<table>
<thead>
<tr>
<th>Name</th>
<th>2-Core</th>
<th>3-Core</th>
</tr>
</thead>
<tbody>
<tr>
<td>Art</td>
<td>6.2</td>
<td>4.4</td>
</tr>
<tr>
<td>Astar</td>
<td>1.0</td>
<td>4.6</td>
</tr>
<tr>
<td>Bwaves</td>
<td>1.77</td>
<td>12.2</td>
</tr>
<tr>
<td>Bzip2</td>
<td>8.0</td>
<td>14.3</td>
</tr>
<tr>
<td>Disparity</td>
<td>4.5</td>
<td>12.9</td>
</tr>
<tr>
<td>Equake</td>
<td>16.6</td>
<td>6.7</td>
</tr>
<tr>
<td>H264ref</td>
<td>3.5</td>
<td>5.7</td>
</tr>
<tr>
<td>Mcf</td>
<td>2.5</td>
<td>4.6</td>
</tr>
<tr>
<td>Omnetpp</td>
<td>10.3</td>
<td>7.9</td>
</tr>
<tr>
<td>Perlbench</td>
<td>3.5</td>
<td>3.3</td>
</tr>
<tr>
<td>Tracking</td>
<td>4.3</td>
<td>8.7</td>
</tr>
<tr>
<td>Svm</td>
<td>6.5</td>
<td>13.4</td>
</tr>
</tbody>
</table>

**Training Set**

**Testing Set**

DPMS

- Begin
- Equal Core Distribution
- New Scheduling Epoch
- New MAS Round
- Execute Tasks
- Moves Made = 0?
- Max Epoch?
- Myopic Performance Improving Moves
- Agents Make Positive Utility Moves
- Stop
- No
- Yes

Agent Per-Tasks
### Worst-Case Complexity: Dynamic Prog. vs DPMS

- **C**: Number of Cores
- **T**: Number of Tasks

<table>
<thead>
<tr>
<th></th>
<th>Dynamic Programming</th>
<th>DPMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Calculations Overhead</td>
<td>$O(T^2 C)$</td>
<td>$O(T^2 C)$</td>
</tr>
<tr>
<td>Per-Core Calculations Overhead</td>
<td>$O(T^2 C)$</td>
<td>$O(T^2)$</td>
</tr>
<tr>
<td>Communication Overhead</td>
<td>$O(T)$</td>
<td>$O(T^2)$</td>
</tr>
<tr>
<td>Space Overhead</td>
<td>$O(T C)$</td>
<td>$O(1)$</td>
</tr>
</tbody>
</table>

### Results: Experimental Setup

- **Gem5, Bahurupi Adaptive Cores and ARM v7 ISA**
- **36 Benchmarks from SPEC 2000 & 2006, PARSEC and SD-VBS**

### Results: Rounds to Convergence

- Full System Reconfiguration
- Partial System Reconfiguration

### Results: Performance

- On a 256-core system, **same** performance for DPMS and Dynamic Programming.
Results: Changes in Overheads

- 1000x Decrease in Per-Core Processing Overhead
- 10x Increase in Communication Overhead

Thank You

- Unanswered Question: Is 1000x decrease in processing overhead better than 10x increase in communication overhead?
- There also exists an efficient centralized greedy scheduler for the problem.

http://invasic.de

Questions?