Optimizing Power-Performance Trade-off for Parallel Applications through Dynamic Core and Frequency Scaling

Satoshi Imamura  Hiroshi Sasaki  Naoto Fukumoto  Koji Inoue  Kazuaki Murakami

Kyushu University
Many-core Processors

- Multi-core processor is currently mainstream
- Core counts on a chip increase as technology size shrinks
- Many-core processor era is coming
  - 10s and 100s of cores on a chip
  - Execute a multi-threaded program for high performance

TILERA “TILE-Gx100” ブロック図
http://www.tilera.com/products/processors/TILE-Gx_Family
Challenge of Many-core

• Demand for low power consumption
  o Ex: Large scale data centers
    • Reduce peak power consumption by power capping

Programs need to be efficiently executed under power consumption constraint
Two Knobs to Determine Performance

- CPU frequency & the number of cores

- Characteristics of multi-threaded programs differ among/within programs
  - Sensitivity to CPU frequency
  - Parallelism

Need to choose the proper configuration according to the kind of programs and their behaviors
Experimental Environment

32-core AMD four socket system

Configuration of processor

<table>
<thead>
<tr>
<th>Processor</th>
<th>AMD Opteron 6136</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of cores</td>
<td>8</td>
</tr>
<tr>
<td>L1 I/D cache</td>
<td>128 KB</td>
</tr>
<tr>
<td>L2 cache</td>
<td>512 KB</td>
</tr>
<tr>
<td>Shared L3 cache</td>
<td>12 MB</td>
</tr>
<tr>
<td>Main memory</td>
<td>16 GB (DDR3-1333)</td>
</tr>
<tr>
<td>Bus speed</td>
<td>6.4 GT/s</td>
</tr>
<tr>
<td>Technology size</td>
<td>45 nm</td>
</tr>
</tbody>
</table>

Conventional execution & Power constraint:
The power when all 32 cores run on 0.8 GHz

<table>
<thead>
<tr>
<th>Number of cores</th>
<th>CPU frequency [GHz]</th>
<th>Supply voltage [V]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – 5</td>
<td>2.4</td>
<td>1.30</td>
</tr>
<tr>
<td>6 – 8</td>
<td>1.9</td>
<td>1.21</td>
</tr>
<tr>
<td>9 – 12</td>
<td>1.5</td>
<td>1.13</td>
</tr>
<tr>
<td>13 – 19</td>
<td>1.1</td>
<td>1.04</td>
</tr>
<tr>
<td>20 – 32</td>
<td>0.8</td>
<td>0.95</td>
</tr>
</tbody>
</table>
Characteristics among Programs

- blackscholes
  - 0.8GHz
  - 1.1GHz
  - 1.5GHz
  - 1.9GHz
  - 2.4GHz

- x264
  - 0.8GHz
  - 1.1GHz
  - 1.5GHz
  - 1.9GHz
  - 2.4GHz

- dedup
  - 0.8GHz
  - 1.1GHz
  - 1.5GHz
  - 1.9GHz
  - 2.4GHz

Performance graphs showing normalized performance against the number of cores for different programs and processor frequencies.
Characteristics within a Program

**x264**

- 4cores@2.4GHz
- 8cores@1.9GHz
- 12cores@1.5GHz
- 16cores@1.1GHz
- 32cores@0.8GHz

IPS : Instructions Per Second
Our Goal

• Maximize performance of parallel programs on many-core under power consumption constraint
  o Variety of characteristics among/within programs
    • Sensitivity to CPU frequency
    • Scalability to core counts
  o Choose the optimal trade-off point between core counts and CPU frequency dynamically
Overview of DCFS
(Dynamic Core and Frequency Scaling)

• Optimize core counts and CPU frequency dynamically according to characteristics of program
  o High parallelism
    • Parallel processing with the maximum available core counts
  o Medium/low parallelism
    • Restrict the number of active cores
    • Reallocate the power budget to increase CPU frequency
DCFS Algorithm

• Two phases
  o In Training phase
    • Change the configuration of core counts and CPU frequency periodically
    • Measure IPS during execution with each configuration
    • Estimate the optimal configuration using measured IPS
  o In Execution phase
    • Execute with the optimal configuration
    • Detect behavior changes of executed program
How to find the best configuration

- Find the best core counts for each CPU frequency
  - Decrement core counts until IPS declines
- Select the configuration with the highest IPS
Evaluation Result

- **DCFS-3, DCFS-10:**
  - Our proposed technique without detecting behavior changes
  - Execution with the configuration estimated in Training phase for constant 3 or 10 seconds

- **DCFS-WD:**
  - Our proposed technique with detecting behavior changes
Evaluation Result

- Almost no performance improvement for high parallelism programs
  - Execution with all cores maximizes performance
  - Performance degradation due to overhead of Training phase
Evaluation Result

- Almost no performance improvement despite of middle/low parallelism
  - Two most memory-bound programs in PARSEC*
  - Small performance improvement by increasing CPU frequency

Evaluation Result

- Performance improvement for middle/low parallelism programs
  - 35% improvement for dedup
  - 20% improvement on average for four programs
  - 6% improvement on average for all programs
Conclusions

• Challenge of many-core processors
  o Maximizing performance under power constraint

• Proposed technique: DCFS
  o Optimize core counts and CPU frequency dynamically
  o Detect behavior changes of executed program

• Evaluation
  o Max 35% performance improvement
    • 6% performance improvement for ten benchmarks
  o No performance improvement for high parallelism and memory-bound programs
Future Work

- Improve the algorithm of our technique to find the best configuration and to detect behavior changes
- Evaluate under different power consumption constraints
- Evaluate on different platforms
Thank you for your attention.

I would appreciate if you could ask me questions slowly.