Introduction
Our Implementation
Results
Possible Extensions / Improvements

Process Cruise Control
on Intel Atom Processors

Martin Farkas-Schmid    Sebastian Klaas

Department of Computer Science
Karlsruhe Institute Of Technologie (KIT)

Power Management Lab Winter Term 2009/2010
Outline

1. Introduction
   - Process Cruise Control

2. Our Implementation

3. Results
   - Benchmarking Results
   - Power Consumption

4. Possible Extensions / Improvements
General Idea

- power management policy based on frequency scaling
- using embedded hardware monitors
- individual clock frequency for each task
- performance should only suffer slightly
Trade-offs / Restrictions

- power consumption vs. QoS
- does only work on architectures providing performance counters
The Policy

- find set of countable events, characterizing tasks
- make partitions (frequency domains) - each according to an optimal speed
- determine the optimal frequency at each task-switch
Settings

- performance counters count:
  - retired instructions
  - memory bus activity
- maximum performance loss: 15%
Performance Measurement

- evaluate performance counter data at each task-switch
- calculate ratio: \((\text{retired instructions}) \div \text{(bus activity)}\)
- store ratio in task\_struct
- store new frequency value in task\_struct
- values are stored in task\_struct
Frequency Switch

- immediately before a task is scheduled
- set frequency according to value in task_struct
- switch functions provided by a new governor
### Frequency Domains

<table>
<thead>
<tr>
<th>Instr. per Mem.act.</th>
<th>According Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 - 50</td>
<td>800 Mhz</td>
</tr>
<tr>
<td>50 - 100</td>
<td>1067 Mhz</td>
</tr>
<tr>
<td>100 - 200</td>
<td>1333 Mhz</td>
</tr>
<tr>
<td>200 - ...</td>
<td>1600 Mhz</td>
</tr>
</tbody>
</table>
## Some Examples

<table>
<thead>
<tr>
<th>Program</th>
<th>Avg. Instr. per Mem.act.</th>
<th>Best Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>cksum</td>
<td>68-86</td>
<td>1333 MHz</td>
</tr>
<tr>
<td>gzip</td>
<td>108-160</td>
<td>1600 MHz</td>
</tr>
</tbody>
</table>

(values are examples - recorded with an old buggy kernel version ...)

Farkas-Schmid, Klaas  
Process Cruise Control
Linux 2.6.30 with PCC

Benchmarking Results
Power Consumption

Introduction
Our Implementation
Results
Possible Extensions / Improvements

Farkas-Schmid, Klaas
Process Cruise Control
Linux 2.6.30 with PCC

![Graph showing benchmarking results for Linux 2.6.30 with Process Cruise Control (PCC).]
Possible Extensions / Improvements

- let the user decide about the acceptable loss of performance
- frequency decision based on more than one timeslice
- improve frequency domains (more dimensions, more detailed)
Andreas Weissel, Frank Bellosa: Process Cruise Control: Event-driven clock scaling for dynamic power management
Cases 2002, October 8-11, 2002, Grenoble France
http://doi.acm.org/10.1145/581630.581668