Autotuning the Energy Consumption
Content

- Overview
- The enopt library (libenopt)
  - Features
  - SandyBridge microarchitecture
  - Components
  - Metrics
  - Validation
  - Tests and Results
- Plugin for the Energy Consumption via CPUFreq
- IBM Energy model
Content

- Overview
- The enopt library (libenopt)
  - Features
  - SandyBridge microarchitecture
  - Components
  - Metrics
  - Validation
  - Tests and Results
- Plugin for the Energy Consumption via CPUFreq
- IBM Energy model
Overview

- Processors operating at lower clock speed consume proportionately less power and generate less heat.
- Dynamic scaling of the clock speed gives some control in power consumption, when not operating at full capacity.
- Lower processor frequency does not necessarily reduce energy consumption (application will take longer)
Overview

Energy consumption vs. CPU parameters set

Garching. 12.09.2013

Autotuning the energy consumption
Content

- Overview
- The enopt library (libenopt)
  - Features
  - SandyBridge microarchitecture
  - Components
  - Metrics
  - Validation
  - Tests and Results
- Plugin for the Energy Consumption via CPUFreq
- IBM Energy model
Features

■ Written in C++
■ Bindings for C and Fortran codes
■ Support for:
  ● Parallel codes: MPI, OpenMP and Hybrid.
  ● Sequential codes.
■ Socket and node level counter measurements.
■ Compatible with PAPI v4 and PAPI v5 headers
■ Provides accesses to kernel mode operations:
  ● Changing CPUFreq infrastructure parameters.
  ● Accesses to the MSR devices.
Content

- Overview
- The enopt library (libenopt)
  - Features
  - SandyBridge microarchitecture
  - Components
  - Metrics
  - Validation
  - Tests and Results
- Plugin for the Energy Consumption via CPUFreq
- IBM Energy model
SandyBridge microarchitecture

- SandyBridge sensors
Content

■ Overview

■ The enopt library (libenopt)
  ● Features
  ● SandyBridge microarchitecture
  ● Components
  ● Metrics
  ● Validation
  ● Tests and Results

■ Plugin for the Energy Consumption via CPUFreq

■ IBM Energy model
Components

Garching, 12.09.2013

Autotuning the energy consumption
Components

- **Language wrappers layer**
  - Set of functions with different mangling.
  - Allow access the library kernel from different languages.

- **Computing strategy layer**
  - Discover of processes topology: register and handshake.
  - Election of the master process per node: node level counters.

- **Counter layer: Interface for counter commands**
  - PAPI counters: at process level.
  - RAPL counters: at node level. A *refresh* function is needed.
  - PaddleCard counters (AC/DC sensors): node level. HWMON.
  - **CAP_SYS_RAWIO** permissions to access the MSR devices.
Components

- CPUFreq layer
  - Starting with the 2.6.0 Linux kernel, the CPU frequency can be dynamically scaled through the cpufreq subsystem.
  - The cpufreq subsystem lies in the sysfs entries.
  - **Superuser** privileges to change sysfs entries are needed.
  - Governors (policies):
    - Static: performance, powersave, userspace.
    - Dynamic: conservative, ondemand.
  - Frequencies SB: (1.2GHz : 2.7 GHz : 100MHz).
Components – Servers and Daemons

- CPUFreqServer / CPUFreqDaemon
  - Since kernel 2.6, there is no daemon to change governors and frequencies.
  - Granted permissions are needed to access the sysfs entries.
  - Server: Root process that performs a chmod operation to the required permissions over the corresponding sysfs entry per core.
  - Daemon: waits for CPUFreq requests. As soon as one is received, it instanciates the server process.
  - Communication between server and library done through a special file.
Components – Servers and Daemons

- **PAPIServer / PAPIDaemon**
  - Under non-privileged users, only binaries with the CAP_SYS_RAWIO flag set can access the MSR devices. (sudo is NOT a solution).
  - Server: Root process that access the MSR devices, perform measurements and send them back to the library.
  - Daemon: waits for requests. As soon as one is received, it triggers the server process.
  - Communication between server and library done through a special file.
Components – Notifications

- Inotify library
  - Linux kernel subsystem that notice operations over the filesystem.
  - A watcher can be configured to monitor a file or folder.
  - Monitorized operations: open, close, read, write, append...
  - It report those changes to applications, triggering the start of a command / binary.
  - Notifications are launched in real time.
  - Takes the advances of the polling mechanisms of kernel.
Content

- Overview
- The enopt library (libenopt)
  - Features
  - SandyBridge microarchitecture
  - Components
  - Metrics
  - Validation
  - Tests and Results
- Plugin for the Energy Consumption via CPUFreq
- IBM Energy model
Metrics

- **PAPI - RAPL**
  - PAPI_TOT_CYC
  - PAPI_TOT_INS
  - PAPI_L3_TCM
  - PACKAGE_ENERGY:PACKAGEx
  - PPo_ENERGY:PACKAGEx
  - DRAM_ENERGY:PACKAGEx

- **Paddle Card (HWMON kernel driver)**
  - AC Counter
  - DC Counter

- **Time metrics**

- **Future: other counters: temperature, fan, network counters...**
Validation

- Overview
- The enopt library (libenopt)
  - Features
  - SandyBridge microarchitecture
  - Components
  - Metrics
  - Validation
  - Tests and Results
- Plugin for the Energy Consumption via CPUFreq
- IBM Energy model
### Validation

**Comparison of measurements with three external tools.**

<table>
<thead>
<tr>
<th>Tool</th>
<th>DRAM</th>
<th>SOCKET</th>
<th>NODE</th>
<th>RACK</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIKWID</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>PAPI-RAPL</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>PaddleCard</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>PDU</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tool</th>
<th>Technology</th>
<th>Resolution</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIKWID</td>
<td>MSR</td>
<td>1ms</td>
<td>Socket, DRAM</td>
</tr>
<tr>
<td>PAPI-RAPL</td>
<td>MSR</td>
<td>1ms</td>
<td>Socket, DRAM</td>
</tr>
<tr>
<td>PaddleCard</td>
<td>Ibmaem-HWMON</td>
<td>300 ms</td>
<td>Node</td>
</tr>
<tr>
<td>PDU</td>
<td>Power meter</td>
<td>10 min</td>
<td>Rack</td>
</tr>
</tbody>
</table>

Garching. 12.09.2013
Validation

- sleep(10) command

<table>
<thead>
<tr>
<th></th>
<th>LIKWID</th>
<th>RAPL</th>
<th>IBMAEM</th>
<th>IBMAEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>PKG0</td>
<td>105 J</td>
<td>103 J</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PKG1</td>
<td>-</td>
<td>104 J</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DRAM0</td>
<td>-</td>
<td>25.5 J</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DRAM1</td>
<td>-</td>
<td>25.5 J</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DC</td>
<td>-</td>
<td>-</td>
<td>491 J</td>
<td>449 J</td>
</tr>
<tr>
<td>E/Node</td>
<td>210 J</td>
<td>257 J</td>
<td>245.5 J</td>
<td>224.5 J</td>
</tr>
</tbody>
</table>
Validation

- MSR and Paddle Card comparison

MSR and Paddle Cards Power utilization

Garching, 12.09.2013  
Autotuning the energy consumption
Validation

■ Power utilization average

Garching, 12.09.2013

Autotuning the energy consumption
Content

■ Overview

■ The enopt library (libenopt)
  ● Features
  ● SandyBridge microarchitecture
  ● Components
  ● Metrics
  ● Validation
  ● Tests and Results

■ Plugin for the Energy Consumption via CPUFreq

■ IBM Energy model
Tests and Results

APEX-MAP benchmark
- Generates artificial calculations and memory accesses with measurement purposes.
- Assumes that performance behavior of scientific apps can be modeled by a set of specific performance factors
- Simulate compute and memory bound applications
- Developed by the Laurence Berkeley National Laboratory
- Specific performance factors: memory bandwidth and FLOPS
Tests and Results

Runtime vs. Frequency (userspace)

\[ y = 7.124x^2 - 25.44x + 1609.6 \]

\[ R^2 = 0.9634 \]
Tests and Results

Power used vs. Frequency (userspace)

\[ y = -0.0285x^2 - 0.1617x + 50.14 \]

\[ R^2 = 0.9215 \]
Tests and Results

**MB/s vs MFlops**

![Graph showing MB/s vs MFlops](image)
Tests and Results

Socket power (W) vs MB/s

P.Socket vs MB/s

- 2.7 GHz
- 2.6 GHz
- 2.5 GHz
- 2.4 GHz
- 2.3 GHz
- 2.2 GHz
- 2.1 GHz
- 2.0 GHz
- 1.9 GHz
- 1.8 GHz
- 1.7 GHz
- 1.6 GHz
- 1.5 GHz
- 1.4 GHz

Garching, 12.09.2013

Autotuning the energy consumption
Content

- Overview
- The enopt library (libenopt)
  - Features
  - SandyBridge microarchitecture
  - Components
  - Metrics
  - Validation
  - Tests and Results
- Plugin for the Energy Consumption via CPUFreq
- IBM Energy model
Aim

- Optimize the energy consumption of an arbitrary application, by choosing the best combination of frequencies for each code region.

Integration with periscope

- The start of each code region calls (per callback) the corresponding library function to change:
  - The CPU governor
  - The CPU frequency
- The code is executed for each combination of frequencies and governors, looking for the minima energy consumption.
Content

■ Overview

■ The enopt library (libenopt)
  ● Features
  ● SandyBridge microarchitecture
  ● Components
  ● Metrics
  ● Validation
  ● Tests and Results

■ Plugin for the Energy Consumption via CPUFreq

■ IBM Energy model
IBM Energy model

- Will be used by loadleveler to select the best frequency to minimize the energy consumption (energy_to_solution)

\[
PWR_{Fn} = A_{Fn} * GIPS_{F0} + B_{Fn} * GBS_{F0} + C_{Fn}
\]

- GIPS and GBS are measured at nominal frequency.
- \(A_n, B_n\) and \(C_n\) are measured for the given platform at all possible frequencies.
- Hides the dependency of GIPS and GBS of a given clock frequency
Your turn!

Questions?

carmen.navarrete@lrz.de