Autotuning the Energy Consumption

{carmen.navarrete, carla.guillen, wolfram.hesse, matthias.brehm, david.brayford}@lrz.de
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

5 Governors:

- **Static (no thresholds)**
  - Performance: Max(frequency).
  - Powersave: Min(frequency).
  - Userspace: User defined frequency.

- **Dynamic**:
  - Ondemand: Single threshold increases and decreases the frequency step size.
  - Conservative: Dual threshold (up & down frequency) reduces the possibility of oscillation between frequency steps.
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.
SandyBridge microarchitecture

- SandyBridge sensors

1-6: RAPL (Running Average Power Limit) Counters.


1 & 2: Energy of the 8 cores.
3 & 4: Energy of the complete Package (core + uncore).
5 & 6: DRAM Energy.

7: DC Counter.
8: AC Counter.
Components

Allow access the library kernel from different languages.

Discover of processes topology: register and handshake. Election of the master process per node: node level counters.

Counter layer: Interface for counter commands

Communication between server and library done through a special file.

Communication with the Linux kernel subsystem
Metrics

- PAPI - RAPL
  - PAPI_TOT_CYC
  - PAPI_TOT_INS
  - PAPI_L3_TCM
  - PACKAGE_ENERGY:PACKAGEEx
  - PPo_ENERGY:PACKAGEEx
  - DRAM_ENERGY:PACKAGEEx

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

- Time metrics

- Future: other counters: temperature, fan, network counters...
## 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>X</td>
<td></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>
</tr>
</thead>
<tbody>
<tr>
<td>LIKWID</td>
<td>MSR</td>
<td>1ms</td>
</tr>
<tr>
<td>PAPI-RAPL</td>
<td>MSR</td>
<td>1ms</td>
</tr>
<tr>
<td>PaddleCard</td>
<td>Ibmaem-HWMON</td>
<td>300 ms</td>
</tr>
<tr>
<td>PDU</td>
<td>Power meter</td>
<td>1 min</td>
</tr>
</tbody>
</table>
# 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>25 J</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>260 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 graph](image-url)
Tests and Results

APEX-MAP benchmark

- Generates artificial calculations and memory accesses for measurement purposes.
- Assumesthat 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
Plugin for the Energy Consumption via CPUFreq

- **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 minimum energy consumption.
Energy model

- Used by loadleveler to minimize the energy-to-solution

\[
PWR_{Fn} = PWR_{F0} \times \text{func}_0(CPI_{F0}, L2_{F0}, L3_{F0}, GIPS_{F0}, GBS_{F0}, \ldots)
\]

\[
T_{Fn} = T_{F0} \times \text{func}_1(CPI_{F0}, L2_{F0}, L3_{F0}, GIPS_{F0}, GBS_{F0}, \ldots)
\]

- CPI, L2, L3, GIPS, GBS... are measured at nominal frequency Fo.
- Coefficients of functions are measured for the given platform at all possible frequencies.
- Hides the dependency of GIPS and GBS of a given clock frequency
Energy model

- Predicted energy use (model) vs Observed energy used (measured)
Your turn!

Questions?