Tracing and Visualization of Energy Related Metrics
8th Workshop on High-Performance, Power-Aware Computing 2012, Shanghai

Timo Minartz, Julian Kunkel, Thomas Ludwig

timo.minartz@informatik.uni-hamburg.de

Scientific Computing
Department of Informatics
University of Hamburg

21-05-2012
Motivation

- HPC is a cost-intensive tool
  - Several Megawatts per installation
- Greening of HPC attracts many scientists and raises unconventional approaches
  - E.g. usage of performance and sleep states of hardware
  - **But:** Energy-performance trade off is still difficult to analyze
1. Introduction

2. Software environment
   - HDTrace
   - Sunshot

3. Energy analysis
   - Hardware environment
   - Exemplary visualization

4. Conclusion
State-of-the-art

- Various unconventional hardware architectures evaluated
- New measurement infrastructure on all levels
  - Infrastructure, systems and components
- Allows evaluation of software approaches like energy-efficiency tuning of libraries and applications
- **But:** Hardware mechanisms like performance or sleep states make it difficult to evaluate measurements
  - High potential for wrong decisions
  - Fast and frequent state transitions make it difficult to view changes
- Conventional approach is to conduct several measurements over larger time frames to smooth the usage of hardware states
Approach

- Correlate MPI applications with hardware utilization, hardware states and power consumption using off-line tracing
- Evaluate the quality of energy-saving mechanisms
  - Identification of wait times in the application and relate them to hardware states
  - Point out wrong decisions about hardware states
- Enhance already existing tracing environment HDTrace to trace energy-related metrics
- Use visualization tool Sunshot to correlate application and new metrics
1 Introduction

2 Software environment

3 Energy analysis

4 Conclusion
**HDTrace**

- Experimental tracing environment developed under the GPL
- Events (like MPI function calls) are stored in XML files
- Statistics (like system activity) are stored in a binary format with XML description header
- Project file links together events and statistic files

**Available statistics**

- Component utilization (using *libgtop*)
- Processor performance counters (using *likwid*)
- Power consumption
Sampling asynchronous hardware states

**Processor**

- P-State frequency via `cpufreq` and/or `cpufreq-stats`
- C-State usage via `cpuidle`
- Socket voltage via `lm-sensors` and `IPMI`

**Hard disk**

- Power saving mode via `hdparm`

**Network Interface Card**

- Speed and Duplex mode via `ethtool`
Tracing overhead

![Graph showing the relationship between trace interval (microseconds) and runtime (seconds) and trace size (kB). The graph indicates a downward trend as the trace interval increases.](image-url)
Sunshot

- Timeline-based Java-Swing application to visualize trace files
- Based on Jumpshot
- Supports profiles, histograms, user-defined derived metrics...

User-defined derived metrics

- Create new statistic timelines based on traced statistics and user-defined operations
- Possible operations are add, mul, sub, div, avg, min and max
User-derived statistics

- MPI Application
- Average processor frequency per node
- Node power consumption
- Average processor frequency per application
- Total power consumption per application
1 Introduction

2 Software environment

3 Energy analysis

4 Conclusion
Hardware

Details

- 3 × LMG 450 power meter
  - 4 channels each
  - up to 20 samples per second
- 5 × AMD Opteron 6168
  - Dual socket
  - 24 cores per node
- 5 × Intel Xeon X5560
  - Dual socket
  - 8 cores per node
  - SMT disabled
MPI barrier with ondemand governor for all cores

- Core frequency increases when entering barrier
- Power consumption increases
- MPI implementation seems to use busy-waiting
MPI barrier at fixed max frequency for all cores

- C-State usage changes from C3 to C0
- Main reason for power consumption increase
Switching processor states under load

- Socket bandwidth decreases when decreasing core frequency
- Socket voltage decreases when all cores on a socket are running at decreased frequency
- Node power decreases when socket voltage decreases
Switching hardware states from applications

**Processor**
- Reduce core frequency on memory-bound application phases
- Reduce core frequency in communication and I/O phases

**Disk and NIC**
- Sleep / reduce speed if unused

**Problems: Wrong decisions**
- Application behavior changes
- Library or OS interaction
MPI barrier with switching devices

- Switching DISK and NIC mode
- Visualization of effect in hardware states
- Utilization allows to identify wrong decisions
Conclusions and future work

Conclusions

- Correlation of MPI application and device utilization is helpful to detect performance issues
- Visualization of idle states and power consumption provides further insights
- Very helpful for evaluating (existing) energy saving strategies

Future work

- Detailed studies about power saving potential of scientific applications
## Trace file size dependent on runtime

<table>
<thead>
<tr>
<th>Runtime (s)</th>
<th>Filesize (kB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>1000</td>
<td>100</td>
</tr>
<tr>
<td>10000</td>
<td>1000</td>
</tr>
<tr>
<td>100000</td>
<td>10000</td>
</tr>
<tr>
<td>1000000</td>
<td>100000</td>
</tr>
</tbody>
</table>

**Single node**

À graph showing the relationship between runtime and trace file size for different runtimes (1 ms, 10 ms, 100 ms). The graph indicates a linear increase in file size with increasing runtime.