Manual and Automatic Energy Tuning for HPC Codes

The Projects Score-E and READEX

Kai Diethelm
GNS Gesellschaft für numerische Simulation mbH
Michael Gerndt
TU München
Outline

- Motivation
- Score-E and READEX
- Energy Monitoring for HPC applications
- Application analysis
- Automatic energy tuning
- Applications
## Exascale Energy Wall

<table>
<thead>
<tr>
<th>#</th>
<th>Site</th>
<th>System</th>
<th>Cores (M)</th>
<th>Rmax (PFlop/s)</th>
<th>Rpeak (PFlop/s)</th>
<th>Power (MW)</th>
<th>Exascale (Factor)</th>
<th>Exascal (MW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>National Supercomputing Center, Wuxi, China</td>
<td>Sunway TaihuLight - Sunway 1.45GHz, Sunway, NRCPC</td>
<td>10.6</td>
<td>93.0</td>
<td>125.4</td>
<td>15.4</td>
<td>8</td>
<td>123</td>
</tr>
<tr>
<td>2</td>
<td>National Super Computer Center in Guangzhou, China</td>
<td>Tianhe-2 (MilkyWay-2), Intel Xeon E5-2.2GHz, Intel Xeon Phi 31S1P, NUDT</td>
<td>3.1</td>
<td>33.9</td>
<td>54.9</td>
<td>17.8</td>
<td>18</td>
<td>324</td>
</tr>
<tr>
<td>3</td>
<td>DOE/SC/Oak Ridge National Laboratory, United States</td>
<td>Titan - Cray XK7 , Opteron 2.2GHz, NVIDIA K20x, Cray Inc.</td>
<td>0.5</td>
<td>17.6</td>
<td>27.1</td>
<td>8.2</td>
<td>37</td>
<td>303</td>
</tr>
<tr>
<td>4</td>
<td>DOE/NNSA/LLNL, United States</td>
<td>Sequoia - BlueGene/Q, Power BQC 1.60 GHz, IBM</td>
<td>1.6</td>
<td>17.2</td>
<td>20.1</td>
<td>7.9</td>
<td>50</td>
<td>392</td>
</tr>
<tr>
<td>5</td>
<td>DOE/SC/LBNL/NERSC, United States</td>
<td>Cori - Cray XC40, Intel Xeon Phi 7250 68C 1.4GHz, Cray Inc.</td>
<td>0.6</td>
<td>14.0</td>
<td>27.9</td>
<td>3.9</td>
<td>36</td>
<td>141</td>
</tr>
</tbody>
</table>
Score-E

- **Partners:**
  - GNS Gesellschaft für numerische Simulation mbH, Braunschweig (Coordinator)
  - RWTH Aachen
  - German Research School for Simulation Sciences, Aachen (until February 2015), TU Darmstadt (from March 2015)
  - TU Dresden
  - FZ Jülich
  - TU München

- **Associated Partners:**
  - Engys GmbH (Rostock), Munters Euroform GmbH (Aachen),
  - U of Oregon

- **Funded by BMBF (Grant No. 01IH13001)**
- **October 2013 – September 2016**
Semi-automatic and Automatic Energy Tuning

REDEX
Runtime Exploitation of Application Dynamism for Energy-efficient eXascale computing

Funded by the European Union’s Horizon 2020 research and innovation programme under Grant Agreement No. 671657.
READEX Project partners

- TU Dresden (Coordinator), Germany
- NTNU, Norway
- IT4I, Czech Republic
- TUM, Germany
- Intel Exascale Lab, France
- GNS Braunschweig, Germany
- ICHEC, Ireland
Score-E: Energy Monitoring with Score-P

Metric Plugins:
- Energy: Intel RAPL, AMD APM, HDEEM
- Temperature
- C-/P-states

Synchronization Adapter
- Shared resources
Manual Analysis of Energy Usage with Vampir

- MPI parallel program
- Tracing information on function level
- Power consumption information for nodes
- Processors
- DRAM
- Effective processor frequency

NAS Parallel Benchmarks (MG)
Manual Analysis with Cube on JURECA
Energy Consumption Projection

Goal: Prediction of energy consumption on larger systems

- Based on Extra-P (GRS, TU Darmstadt, ETH Zürich, LLNL, FZ Jülich)
- Analytical model deduced from application profiles
- Visualization of predicted energy consumption in Cube

MILC: QCD code
Resource-Aware Visualization
Geometry-Aware Visualization
Geometry-Aware Visualization
Energy Tuning with Periscope Tuning Framework

Automatic application analysis & tuning
- Tune performance and energy (statically)
- Plug-in-based architecture
- Evaluate alternatives online
- Scalable and distributed framework

Support variety of parallel paradigms
- MPI, OpenMP, OpenCL, Parallel pattern

Developed in the AutoTune EU-FP7 project
- Integrated with Score-P in Score-E
Tuning Plugins for Static Energy Tuning

- **PCAP**: Optimization of the thread number
  - Exploits scalability limitations
  - Reduces static and dynamic energy
  - Supports tuning of individual parallel regions

- **MPICAP**: Optimizes the number of MPI tasks
  - In addition: reduces communication energy

- **DVFS**: Computes the optimal core frequency
  - Exploits wait cycles (IO-/Memory Bound)
  - Reduces dynamic energy

- **Taurus-Energy**: Combined tuning (3D)
READEX: Beyond Static Tuning

**HPC**
- Automatic Tuning

**Embedded**
- System Scenarios
Systems Scenario based Methodology

**Design-time**

- Applications
- Platform
- Identify system scenarios
  - Develop prediction mechanism
  - Develop switching mechanism
  - Develop calibration mechanism
- Optimize scenario exploitation

**Run-time**

- System integration
- Cost
- Time

- Without scenarios
- With scenarios
  - Exploitation and prediction
  - Switching
  - Calibration
Exploit Intra- and Inter-Phase Dynamism

```c
int main(void) {
    // Initialize application
    // Initialize experiment variables
    int num_iterations = 2;
    for (int iter = 1; iter <= num_iterations;
        // Start phase region
        // Read PhaseCharct
        laplace3D(); // significant regi
        residue = reduction(); // insign
        fftw_execute(); // significant r
        // End phase region
    }
    // Post-processing:
    // Write noise matrices to disk fc
    // Terminate application
    MPI_Finalize();
    return 0;
}
```
Scenario-Based Tuning

Design Time Analysis

Tuning Model

Runtime Tuning

Periscope Tuning Framework (PTF)

READEX Runtime Library (RRL)
## Lulesh: Individual, 3 cores per task

<table>
<thead>
<tr>
<th>individual, 3 cores/task</th>
<th>Worst</th>
<th>Static Tuning</th>
<th>READEX</th>
<th>#Threads</th>
<th>Freq</th>
</tr>
</thead>
<tbody>
<tr>
<td>CalcCourantConstraintForElems</td>
<td>6075</td>
<td>5375</td>
<td>5232</td>
<td>10</td>
<td>2.4</td>
</tr>
<tr>
<td>CalcKinematicsForElems</td>
<td>11170</td>
<td>8260</td>
<td>8250</td>
<td>9</td>
<td>2.4</td>
</tr>
<tr>
<td>CalcMonotonicQGradients</td>
<td>4977</td>
<td>3920</td>
<td>3370</td>
<td>6</td>
<td>2.4</td>
</tr>
<tr>
<td>CalcMonotonicQRegionForElems</td>
<td>11714</td>
<td>12470</td>
<td>11714</td>
<td>16</td>
<td>2.4</td>
</tr>
<tr>
<td>CalcVolumeForceForElems</td>
<td>56821</td>
<td>48111</td>
<td>47804</td>
<td>9</td>
<td>2.4</td>
</tr>
<tr>
<td>EvalEOSForElems</td>
<td>28335</td>
<td>18311</td>
<td>17082</td>
<td>4</td>
<td>2.4</td>
</tr>
<tr>
<td>SUM</td>
<td>119092</td>
<td><strong>96447</strong></td>
<td><strong>93452</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>19.0%</td>
<td>3.1%</td>
</tr>
<tr>
<td>Energy for phase</td>
<td>163891</td>
<td>138508</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>15.5%</td>
<td></td>
</tr>
</tbody>
</table>
Score-E Application Example: Analysis of Indeed

- Finite Element code for sheet metal forming simulation
- SMP version (OpenMP) & DMP version (hybrid: OpenMP + MPI) available
- Two different application scenarios:
  - Classical use case: direct simulation of forming process
  - Recent variation: determination of optimal process parameters (in particular, tool geometry)
- Fundamental difference: mesh size for discretization of tools
- Consequences:
  - Significance of contact search algorithms changes drastically
  - Classification changes between “compute bound” and “memory bound”
Score-E Application Example: Analysis of Indeed

- Potential tuning parameters:
  - CPU frequency
  - Number of OpenMP threads
  - Number of MPI processes
  - Code path switching
Score-E Application Example: Analysis of Indeed

- Potential tuning parameters:
  - CPU frequency
  - Number of OpenMP threads
  - Number of MPI processes
  - Code path switching

Accepted by customers
Undesired by customers
Measurement Results for *Indeed*

Relative values w. r. t. default clock frequency (Sandy Bridge, classical example)
Measurement Results for *Indeed*

Relative values w. r. t. default clock frequency (Haswell, classical example)
Measurement Results for Indeed

Relative values w. r. t. default clock frequency (Haswell, nonclassical example)
Conclusions for Indeed

- Choose high clock frequency
- Exact optimal value depends on
  - Processor architecture
  - Characteristics of input deck
  - Concrete choice of objective function
  - Static tuning very attractive on Sandy Bridge;
- further improvements possible with dynamic tuning
- Optimization on Haswell requires further methods and tools
  (automatic dynamic tuning at runtime → READEX)
- Viability of platform-dependent approach from commercial point of view?
Future Work in READEX and Expected Outcome

- Use tools based on PTF for design-time analysis (search for significant regions, dynamism detection, ...)
- Runtime application tuning based on READEX Runtime Library (RRL)
- Alpha version for full READEX tool suite scheduled for spring 2017
- Later step: Programming paradigm for expressing application dynamism → further improve automatic dynamic energy tuning
- Significant increase in energy efficiency without prohibitively high effort for software developers
Acknowledgements

We thank

- BMBF for providing the funding for Score-E,
- DLR PT-SW for the support and cooperation with respect to Score-E,
- the European Union for the funding for READEX.

Further information

- [www.vi-hps.org/projects/score-e](http://www.vi-hps.org/projects/score-e)
- [www.readex.eu](http://www.readex.eu)