Automated Knowledge-Based Tuning of Parallel Applications

Michael Gerndt
Technische Universität München
Contents

1. Performance analysis and tuning for HPC
2. Automatic performance analysis with Periscope
3. Periscope Tuning Framework
4. Existing tuning plugins
Faculty of Informatics

- Founded in 1975
- 3500 students
- 30% foreign students
- 40 Professors (22 Chairs)
HPC Challenges

• HPC systems
  – Large scale
  – Complex node architectures
  – Huge energy demand
  – Significant power variations
Leibniz Supercomputer Center

www.autotune-project.eu
### TOP 500 – November 2014

<table>
<thead>
<tr>
<th>Name</th>
<th>Site</th>
<th>Architecture</th>
<th>Peak PFlop/s</th>
<th>Power MW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tianhe/2</td>
<td>NUDT</td>
<td>NUDT, Cluster, Intel Xeon + Xeon Phi</td>
<td>54.9</td>
<td>17.8</td>
</tr>
<tr>
<td>Titan</td>
<td>ONL, USA</td>
<td>Cray, 560640 cores, Opteron + Nvidia K20</td>
<td>27.1</td>
<td>8.2</td>
</tr>
<tr>
<td>Sequoia</td>
<td>LLNL, USA</td>
<td>IBM BlueGene/Q, 1.6 M cores</td>
<td>20.1</td>
<td>7.9</td>
</tr>
<tr>
<td>K Computer</td>
<td>RIKEN, Japan</td>
<td>Fujitsu, 705 K cores, SPARC 64</td>
<td>11.3</td>
<td>12.6</td>
</tr>
<tr>
<td>Mira</td>
<td>ANL, USA</td>
<td>IBM BlueGene/Q, 786 K cores</td>
<td>10.1</td>
<td>3.9</td>
</tr>
<tr>
<td>Piz Daint</td>
<td>CSCS</td>
<td>Cray Xeon + K20</td>
<td>7.7</td>
<td>2.3</td>
</tr>
<tr>
<td>Stampede</td>
<td>Univ. Texas</td>
<td>Dell Power + Xeon Phi</td>
<td>5.1</td>
<td>4.5</td>
</tr>
</tbody>
</table>
Pace to Exascale

- Exascale: $10^{18}$ Flops/s
- Expected to be reached after 2018
- Massive parallelism and energy consumption

<table>
<thead>
<tr>
<th>Name</th>
<th>Cores</th>
<th>Power MW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Titan</td>
<td>20 M</td>
<td>303</td>
</tr>
<tr>
<td>Sequoia</td>
<td>80 M</td>
<td>395</td>
</tr>
<tr>
<td>K Computer</td>
<td>64 M</td>
<td>1145</td>
</tr>
<tr>
<td>SuperMUC</td>
<td>49 M</td>
<td>1000</td>
</tr>
</tbody>
</table>
Energy Efficient HPC

- Reduce the power losses in the power supply chain
- Exploit your possibilities for using compressor-less cooling and use energy-efficient cooling technologies (e.g., direct liquid cooling)
- Re-use waste heat of IT systems
- Use newest semiconductor technology
- Use of energy saving processor and memory technologies
- Consider using special hardware or accelerators tailored for solving specific scientific problems or numerical algorithms
- Monitor the energy consumption of the compute systems and the cooling infrastructure
- Use energy aware system software to exploit the energy saving features of your target platform
- Monitor and optimize the performance of your scientific applications
- Use most efficient algorithms
- Use best libraries
- Use most efficient programming paradigm
- Use all performance tuning techniques

Energy efficient infrastructure | Energy efficient hardware | Energy aware software environment | Energy efficient applications
Performance Engineering

Information

Tuning

Insight
Challenges

- Scalability demands
- Performance dynamics
- Non-intuitive tuning parameters

Solutions:
- Visualization
- Automation
Automatic Performance Analysis

• Performance properties
  – Condition, confidence, severity

• Examples
  – Load imbalance
  – Late sender
  – Memory access: L1, L2, L3, Memory, TLB
Performance Dynamics Properties

- Ph.D. thesis of Yury Oleynik (LMAC project)

- Properties
  - Significant Variability Property
  - Degredation Peaks Property
  - Degredation Trends Property
Periscope

- **Online**
  - no need to store trace files
- **Distributed**
  - reduced network utilization
- **Scalable**
  - Up to 100000s of CPUs
- **Multi-scenario analysis**
  - Single-node Performance
  - MPI Communication
  - OpenMP
- **Portable**
  - Fortran, C with MPI & OMP
  - Intel Itanium2, x86 based systems
  - IBM Power6, BlueGene P, Cray

---

**Graphical User Interface (GUI)**

**Interactive Frontend**

**Performance Analysis Agent Network**

- **Master Agent**
- **Communication Agent**
- **Analysis Agent**

**Application with Monitor**

http://www.lrr.in.tum.de/periscope
Pathway

- **Performance Engineering Workflows**
  - Ph.D. thesis of Ventsislav Petkov (LMAC project)

- **Formalization and automation**

- **Support for**
  - Definition of workflows in Eclipse
  - Automatic/manual tasks
  - Transparent experiments
  - Experiment archive and documentation
AutoTune Project

- Automatic tuning based on expert knowledge
- Beyond auto-tuning like Active Harmony
- Tuning plugins for Periscope
- Expert knowledge
  - When is it necessary to tune?
  - Where to tune?
  - What are the tuning parameters?
  - How to shrink the search space?
Periscope Tuning Framework

- Extension of Periscope
- Online tuning process
  - Application phase-based
- Extensible via tuning plugins
  - Single tuning aspect
  - Combining multiple tuning aspects
- Rich framework for plugin implementation
- Automatic and parallel experiment execution
Tuning Plugin Interface

- Plugin
  - Search Space Exploration
  - Tuning Step
- Periscope Frontend
  - Scenario execution
  - Tuning actions
  - Analysis strategies
- Application with Monitor

www.autotune-project.eu
Tuning Plugins

• MPI parameters
  – Eager Limit, Buffer space, collective algorithms
  – Application restart or MPIT Tools Interface

• DVFS
  – Frequency tuning for energy delay product
  – Model-based prediction of frequency
  – Region level tuning

• Parallelism capping
  – Thread number tuning for energy delay product
  – Exhaustive and curve fitting based prediction
Tuning Plugins

• Master/worker
  – Partition factor and number of workers
  – Prediction through performance model based on data measured in pre-analysis

• Parallel Pattern
  – Tuning replication and buffer between pipeline stages
  – Based on component distribution via StarPU

• OpenHMPP/OpenACC
  – Tuning parameters based on CAPS tune directives
  – Optimizing data transfer and kernel execution time
Tuning Plugins

• MPI IO
  – Tuning data sieving and number of aggregators
  – Exhaustive and model based

• Compiler Flag Selection
  – Automatic recompilation and execution
  – Selective recompilation based on pre-analysis
  – Exhaustive and individual search
  – Scenario analysis for significant routines
  – Combination with Pathway
Tuning Workflow

- Tune for different data sets
Search Algorithms

• Tuning Parameters
  TP1: O2, O3, O4         TP2: -xhost, none

• Exhaustive
  Single tuning step
  O2 -xhost     O2
  O3 -xhost     O3
  O4 -xhost     O4

• Individual
  1. Tuning Step
    O2
  2. Tuning Step
    O2 -xhost
## Individual Tuning NPB (C)

<table>
<thead>
<tr>
<th></th>
<th>O</th>
<th>xhost</th>
<th>unroll</th>
<th>ip</th>
<th>ipo</th>
<th>opt-pref</th>
<th>time</th>
<th>min</th>
<th>max</th>
<th>incr. %</th>
</tr>
</thead>
<tbody>
<tr>
<td>BT</td>
<td>02</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>5,19</td>
<td>6,09</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>LU</td>
<td>04</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>3,54</td>
<td>3,92</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>SP</td>
<td>03</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>1,77</td>
<td>3,23</td>
<td>83</td>
<td></td>
</tr>
<tr>
<td>CG</td>
<td>02</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>4,66</td>
<td>4,80</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>FT</td>
<td>04</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>13,10</td>
<td>15,09</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>MG</td>
<td>03</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4,06</td>
<td>4,06</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>IS</td>
<td>02</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4,54</td>
<td>4,62</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
k-NN – training error
SeisSol application code-region optimization

<table>
<thead>
<tr>
<th>Region ID</th>
<th>1.6 GHz</th>
<th>1.7 GHz</th>
<th>1.8 GHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1132</td>
<td>1143</td>
<td>1172</td>
</tr>
<tr>
<td>2</td>
<td>185</td>
<td>185</td>
<td>186</td>
</tr>
<tr>
<td>3</td>
<td>912</td>
<td>928</td>
<td>971</td>
</tr>
</tbody>
</table>

Table showing energy consumption in Joules.

Percentage of energy savings on the outer most region (Region ID 1)
## MPI Parameters Tuning
*(fish school simulation-64K fishes)*

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Best Value</th>
<th>Execution Time (Distribution)</th>
</tr>
</thead>
<tbody>
<tr>
<td>eager_limit</td>
<td>4196 B</td>
<td>65560 B</td>
<td>65560</td>
<td>0.587957 (with default values 0.831273)</td>
</tr>
<tr>
<td>use_bulk_xfer</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>bulk_min_msg_size</td>
<td>4096 B</td>
<td>1048576 B</td>
<td>268288</td>
<td></td>
</tr>
<tr>
<td>task_affinity</td>
<td>CORE</td>
<td>MCM</td>
<td>CORE</td>
<td></td>
</tr>
<tr>
<td>pe_affinity</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>cc_scratch_buf</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>
Periscope Tuning Framework

- periscope.in.tum.de
- Version: 1.0 (1.1 coming in April)
- Supported systems:
  - x86-based clusters
  - Cray XT
  - IBM Power
- License: New BSD
Summary

• Manual and automatic performance engineering are important.

• Periscope supports automatic online analysis

• Periscope Tuning Framework adds automatic tuning plugins

• Pathway formalizes and support the overall performance engineering process