 AutoTune Workshop

Michael Gerndt
Technische Universität München
AutoTune Project

• Automatic Online Tuning of HPC Applications
  – High PERFORMANCE Computing
  – HPC application developers
  – Compute centers: Energy Efficiency

• Partners
  – Coordinator: Technische Universität München
  – Universität Wien
  – Universitat Autonoma de Barcelona
  – Leibniz Rechenzentrum
  – University of Galway, ICHEC

• October 2011-April 2015
Project Participants Today

- **Michael Gerndt**, TUM Coordinator

- **Michael Lysaght**, ICHEC WP5: Evaluation

- **Yury Oleynik**, TUM ECS-Group
Goals of Meeting

• Present the Periscope Tuning Framework
• Present post-project exploitation plans
  – AutoTune Demonstration Center
  – ECS-Group spin-off
• Collect feedback on post-project exploitation
Agenda for Today

11:00 – 12:30 Introduction to the project

Lunch

14:00 – 14:30 AutoTune Demonstration Center
14:30 – 15:00 ECS-Group
15:00 – 15:45 Discussion, Questionnaire

Coffee
Todays Parallel Architectures

• Zoo of architectures
  – Multicore processors
  – NUMA multiprocessors
  – Accelerators
  – Clusters
  – Supercomputers

• and programming models
  – OpenMP, PGAS, TBB
  – MPI
  – CUDA, OpenCL, OpenACC
Multicore Processors

• Ingredients
  – Superscalar cores
  – SIMD units
  – Multithreading
  – Multiple cores
  – Memory hierarchy

• Challenges
  – Long latency, low-bandwidth memory access
  – Multiple levels of parallelism: ILP, SIMD, thread
  – Compilation, vectorization, parallelization (OpenMP)
NUMA Multiprocessors

• Ingredients
  – Multiple multicore processors
  – Shared address space

• Challenges
  – Different latencies to memory, local vs remote access
  – Contention for memory and synchronization
  – OS support for thread affinity
Accelerators

- **Ingredients**
  - Many cores
  - Wide SIMD
  - Specialized Hardware

- **Challenges**
  - Special programming model
    CUDA, OpenCL, OpenACC
  - Computation and data offload
  - Application tuning
Clusters

• Ingredients
  – Many nodes
  – Scalable high performance network
  – Distributed memory

• Challenges
  – Explicit data transfers
  – Global parallelization with MPI
  – Scalability
  – Energy consumption

SuperMUC @ LRZ
Development Cycle

- Coding
- Performance Analysis
- Program Tuning
- Production
- Refinement
- Measurement
- Analysis
- Ranking

www.autotune-project.eu
Performance Analysis

• Goal: Identify inefficient regions and root cause
• Analysis techniques:
  – Time profile
  – Single core: counter profiles
  – Multiple cores: overhead profiles
  – Accelerators: counter and overhead profiles
  – Clusters: Communication profiles and traces
• Measurement techniques
  – Sampling
  – Instrumentation
Performance Analysis Tools

• **Profiling tools**
  – Single core, accelerators
  – Multiple cores, clusters
  – IDE integration

• **Tracing tools**
  – Multiple cores, clusters: visualization of dynamic behavior.

• **Automatic**
  – Beyond measurements
Application Tuning

• Objectives
  – Performance
  – Cost
  – Energy

• When
  – Installation
  – Design time
  – Runtime
VI-HPS Virtual Institute for High Productivity Supercomputing

- Aiming at development of tools and training of users to enable higher productivity in using supercomputers
- Partners

FZJ

GRS

RWTH Aachen

UNI Stuttgart

ZIH Dresden

TUM

UNI Oregon

UNI Tennessee
Periscope Performance Analysis

- On-line
  - no need to store trace files
- Distributed
  - reduced network utilization
- Scales to 100000s of CPUs
- Analyzes:
  - Single-node Performance
  - MPI Communication
  - OpenMP
- Supports: Fortran, C
  - Intel Itanium based systems
  - IBM Power
  - BlueGene P
  - x86 based systems
Performance Properties

• APART specification language
• Performance property
  – Condition
  – Severity
  – Confidence
• Agent analysis strategies
  – Determine hypothesis refinement
    • Region nesting
    • Property hierarchy-based refinement
  – Experiment for a phase execution
Agent Search Strategies

• Application phase is a period of program’s execution
  – Phase region
    • Full program
    • Single user region assumed to be repetitive
  – Phase boundaries have to be global (SPMD programs)

• Search strategies
  – Determine hypothesis refinement
    • Region nesting
    • Property hierarchy-based refinement
Periscope Tuning Framework

Periscope

Pathway

Tuning Plugins
AutoTune Project

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

• Tuning plugins for Periscope
  – Dynamically loaded
  – < few hundred lines of code
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 Plugins

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

• DVFS (LRZ)
  – Frequency tuning for energy delay product
  – Model-based prediction of frequency, pre-analysis
  – Region level tuning

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

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

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

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

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

• Compiler Flag Selection (TUM)
  – Automatic recompilation and execution
  – Selective recompilation based on pre-analysis
  – Exhaustive and individual search
  – Scenario analysis for significant routines
  – Combination with Pathway
  – Machine learning support
Pathway Goals

- Formalize performance engineering workflows.
- Automate usage of analysis and tuning tools.
- Remove Monkey Work from code developers.
- Support overall tuning effort by bookkeeping.
Pathway Features

• Formal, graphical view on workflows
  – Pre-defined workflows available

• Transparently serve different HPC systems
  – Define connection parameters for each system
  – Pathway automatically generates batch scripts for different schedulers

• Automatic tool invocation

• Experiment browser

• Notebook
Scalability Workflow
## 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>incr. %</th>
</tr>
</thead>
<tbody>
<tr>
<td>BT</td>
<td>O2</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>5.19</td>
<td>6.09</td>
</tr>
<tr>
<td>LU</td>
<td>O4</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>3.54</td>
<td>3.92</td>
</tr>
<tr>
<td>SP</td>
<td>O3</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>1.77</td>
<td>3.23</td>
</tr>
<tr>
<td>CG</td>
<td>O2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.66</td>
<td>4.80</td>
</tr>
<tr>
<td>FT</td>
<td>O4</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>13.10</td>
<td>15.09</td>
</tr>
<tr>
<td>MG</td>
<td>O3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.06</td>
<td>4.06</td>
</tr>
<tr>
<td>IS</td>
<td>O2</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.54</td>
<td>4.62</td>
</tr>
</tbody>
</table>
k-NN – training error
DVFS Tuning Plugin

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></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>

0.587957  
(with default values 0.831273)
Periscope Tuning Framework

- periscope.in.tum.de
- Version: 1.0 (1.1 coming in April)
- Supported systems:
  - x86-based clusters
  - Bluegene
  - 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 supports the overall performance engineering process

• Future work: Integration with Score-P