The Periscope Application Tuning Framework

Prof. Dr. Michael Gerndt
Technische Universität München
gerndt@in.tum.de
Performance Analysis and Tuning is Essential
Performance Analysis for Parallel Systems

- Development cycle
  - Assumption: Reproducibility

- Instrumentation
  - Static vs Dynamic
  - Source-level vs binary-level

- Monitoring
  - Software vs Hardware
  - Statistical profiles vs event traces

- Analysis
  - Source-based tools
  - Visualization tools
  - Automatic analysis tools

Diagram:

1. Coding
2. Performance Monitoring and Analysis
3. Program Tuning
4. Production

Flow:
- Coding → Performance Monitoring and Analysis → Program Tuning → Production
AutoTune FP7 Project Goals

• Extend Periscope for automatic tuning
  – Performance and energy

• Support wide spectrum of HPC systems
  – Homogeneous and heterogeneous
  – Focus on SuperMUC

• Provide an easily extensible tuning framework
  – Tuning plugins
  – Interface hides Periscope details but provides support by Periscope’s rich infrastructure
Partners

Technische Universität München

Universität Wien

CAPS Entreprises

Universitat Autònoma de Barcelona

Leibniz Computing Centre

National University of Galaway, ICHEC
• Automated search
  – Based on formalized performance properties
• Online analysis
  – Search performed while application is executing
• Distributed search
  – User specified number of analysis agents
  – Additional cores for agents
• Profile data only
  – even for MPI Waittime analysis
Properties

• StallCycles(Region, Rank, Thread, Metric, Phase)
  – Condition: Percentage of lost cycles >30%
  – Confidence: 1.0
  – Severity: Percentage of lost cycles

• StallCyclesIntegerLoads
  – Requires access to two counters

• L3MissesDominatingMemoryAccess
  – Condition: Importance of L3 misses (theoretical latencies)
  – Severity: Importance multiplied by actual stall cycles
Periscope Design

Frontend

Performance Analysis Agent Network

Master Agent

Communication Agent

Analysis Agent

MRI

Application with Monitor
Agent Search Strategies

• Application phase is a period of program’s execution
  – Phase regions
    • 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
  – Single and multi step strategies
Integration in Eclipse (PTP)

Where is the problem?

What is the most severe problem?

Filter problems for region
Autotune Approach

• Predefined tuning plugins combining performance analysis and tuning

• Plugins
  – Compiler based optimization
  – HMPP tuning for GPUs
  – Parallel pattern tuning
  – MPI tuning
  – Energy efficiency tuning
Periscope Tuning Framework

• Online
  – Analysis and evaluation of tuned version in single application run
  – Multiple versions in single step due to parallelism in application

• Result
  – Tuning recommendation
  – Adaptation of source code and/or execution environment
  – Impact on production runs
Autotuning Extension in HMPP

• Directives to provide optimization space to explore
  – Parameterized loop transformations
  – Alternative/specialized code declaration to specify various implementations

• HMPP static information
  – Optimization space description
  – Static code information collect

• Dynamic information collect (i.e. timing, parameter values)

```c
#pragma hmppcg(CUDA) unroll(RANGE), jam
for( i = 0 ; i < n; i++ ) {
  for( j = 0 ; j < n; j++ ) {
    ...
    VC(j,i) = alpha*prod+beta * VC(j,i);
  }
}
```
Extensions to Periscope

Frontend
- Tuning Plugin
- Search Strategies
- Scenario Execution Engine

Analysis Agent
- Tuning Strategy

Monitor Request Interface
- Tuning Action Requests

Monitor
- Tuning Actions
Tuning Plugin

• Defines tuning space
  – Crossproduct of tuning points

• Goes through single/multiple plugin steps
  – Selection of a variant space
  – Find best variant in this space by generating and executing tuning scenarios

• Searching the variant space can make use of predefined search algorithms.

• Provides functions that can be called by
  – Frontend
  – Meta Tuning Plugins
do k=1,20
  variant=k
  !$MON USERREGION TP name(Test) variable(variant) variants(10)
  tstart=MPI_Wtime()
    call sleep(5-variant+1)
  tend=MPI_Wtime()
  write (*,*), myrank, variant, tend-tstart
  !$MON END USERREGION
enddo
Tuning Objectives

• Tuning searches for variant(s) with best value for a single or multiple objectives

• Objectives are implemented as Periscope properties.
  – Properties specify measurements and return a severity, i.e. the objective value.
  – They are automatically evaluated by the analysis agents based on the AA Tuning Strategy
Tuning Scenarios

• Specify a single variant
  – Region to be tuned
  – Tuning action/value pairs
  – Objective Ids

• Life cycle
  1. Creation by search algorithm -> Scenario Pool
  2. Preparation -> Prepared Scenario Pool
  3. Selection for experiment -> Experiment Scenario Pool
  4. Evaluation -> Finished Scenario Pool

• Steps 1-3 provided by plugin functions
• Step 4 executed by Scenario Execution Engine
Tuning Actions

• Monitor Request Interface (MRI)
  – Configuration of monitor
  – Application control

• MRI tuning actions
  – Variable tuning action
  – Function tuning action

• General tuning actions
  – During preparation of scenarios by tuning plugin
  – During restart of the application
  – During execution
Development of Plugins

• Determine tuning points with tuning actions
• Define (intelligent) search algorithm
  – Predefined search algorithm
  – Plugin-specific search algorithm
  – Combination of both
• Provide functions for
  – Creation and preparation of scenarios
  – Optional recompilation
  – Optional restart parameters
  – Selection of scenarios for next experiment
  – Evaluation of experiment results
Status

• Demo tuning plugin provided for first year review
• Prototypes of AutoTune plugins provided by partners in next year
• Integration with other projects
  – InvasIC (TRR 89)
  – Score-E proposal