The Periscope Application Tuning Framework

Prof. Dr. Michael Gerndt
Technische Univeristät München
gerndt@in.tum.de

Performance Analysis and Tuning is Essential

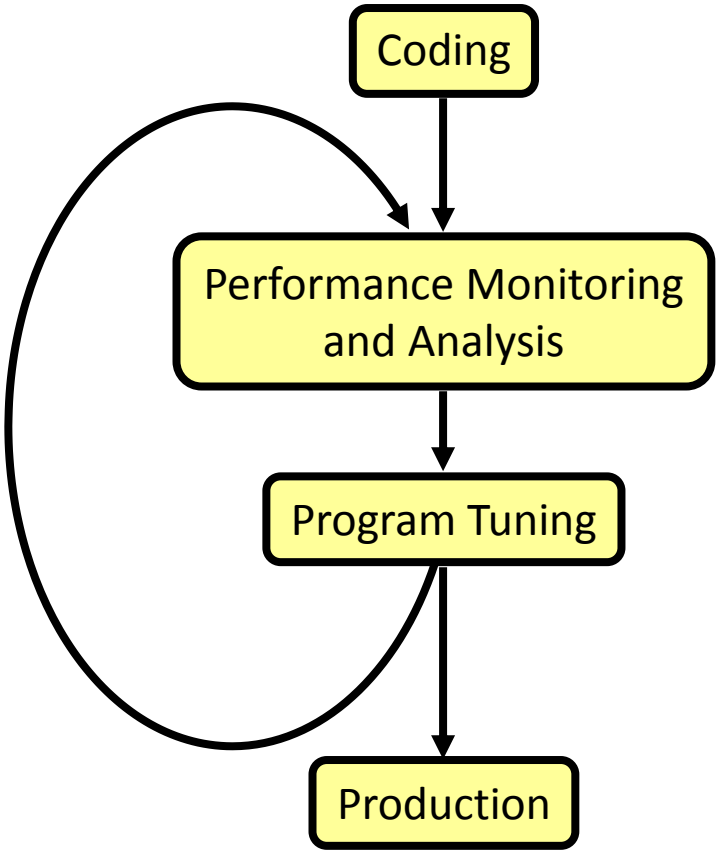
AUTOTUNE

www.autotune-project.eu

Performance Analysis for Parallel Systems

AUTOTUNE
www.autotune-project.eu

- 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



AutoTune FP7 Project Goals

AUTOTUNE
www.autotune-project.eu

- Extend Periscope for automatic tuning
  - Performance and energy
- Support wide spectrum of HPC systems
  - Homogeneous and herterogeneous
  - Focus on SuperMUC
- Provide an easily extensible tuning framework
  - Tuning plugins
  - Interface hides Periscope details but provides support by Periscope‘s rich infrastructure

Partners

AUTOTUNE

www.autotune-project.eu

- Technische Universität München
- Universität Wien
- CAPS Entreprises
- Universitat Autònoma de Barcelona
- Leibniz Computing Centre
- National University of Galaway, ICHEC

Periscope

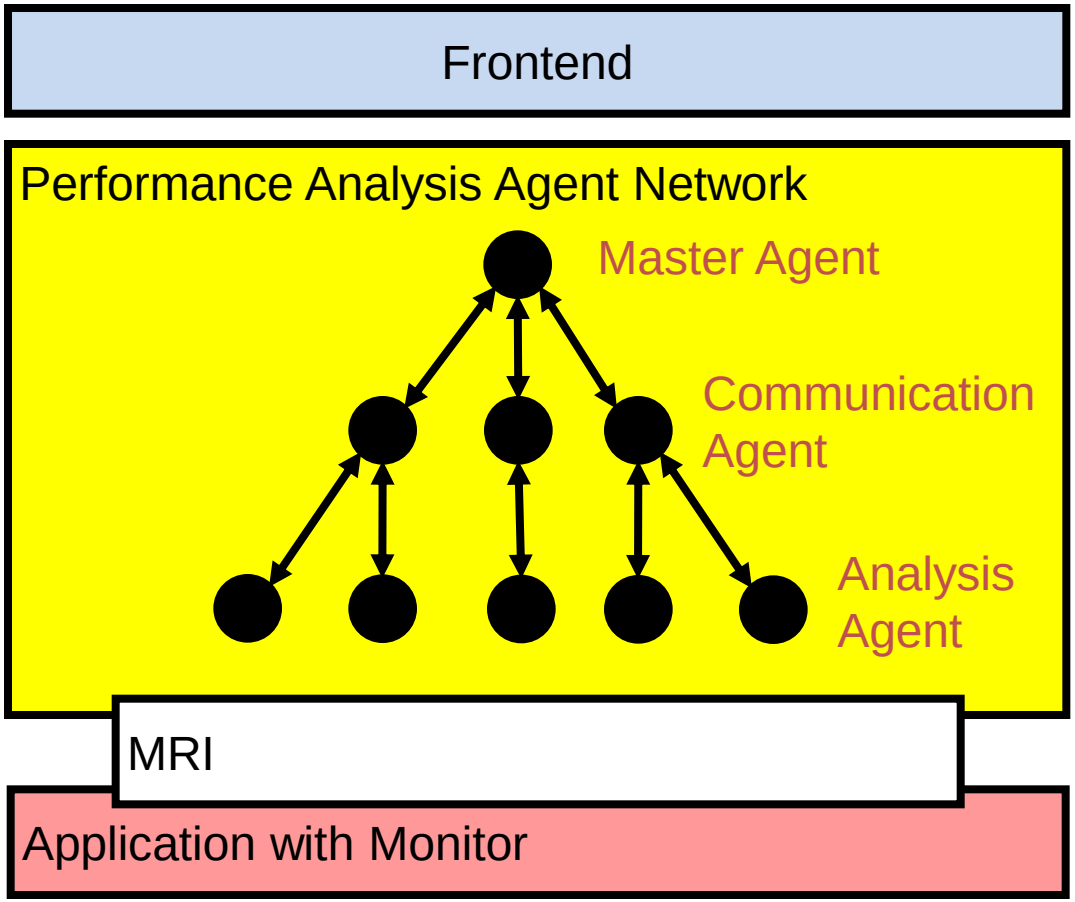
- Automated search
  - Based on formalized performance properties
- Online analysis
  - Search performed while application is executing
- Distributed search
  - User specified number of analysis agents
  - Additional cores for agents
- Profile data only
  - even for MPI Waittime analysis

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

AUTOTUNE
www.autotune-project.eu



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)

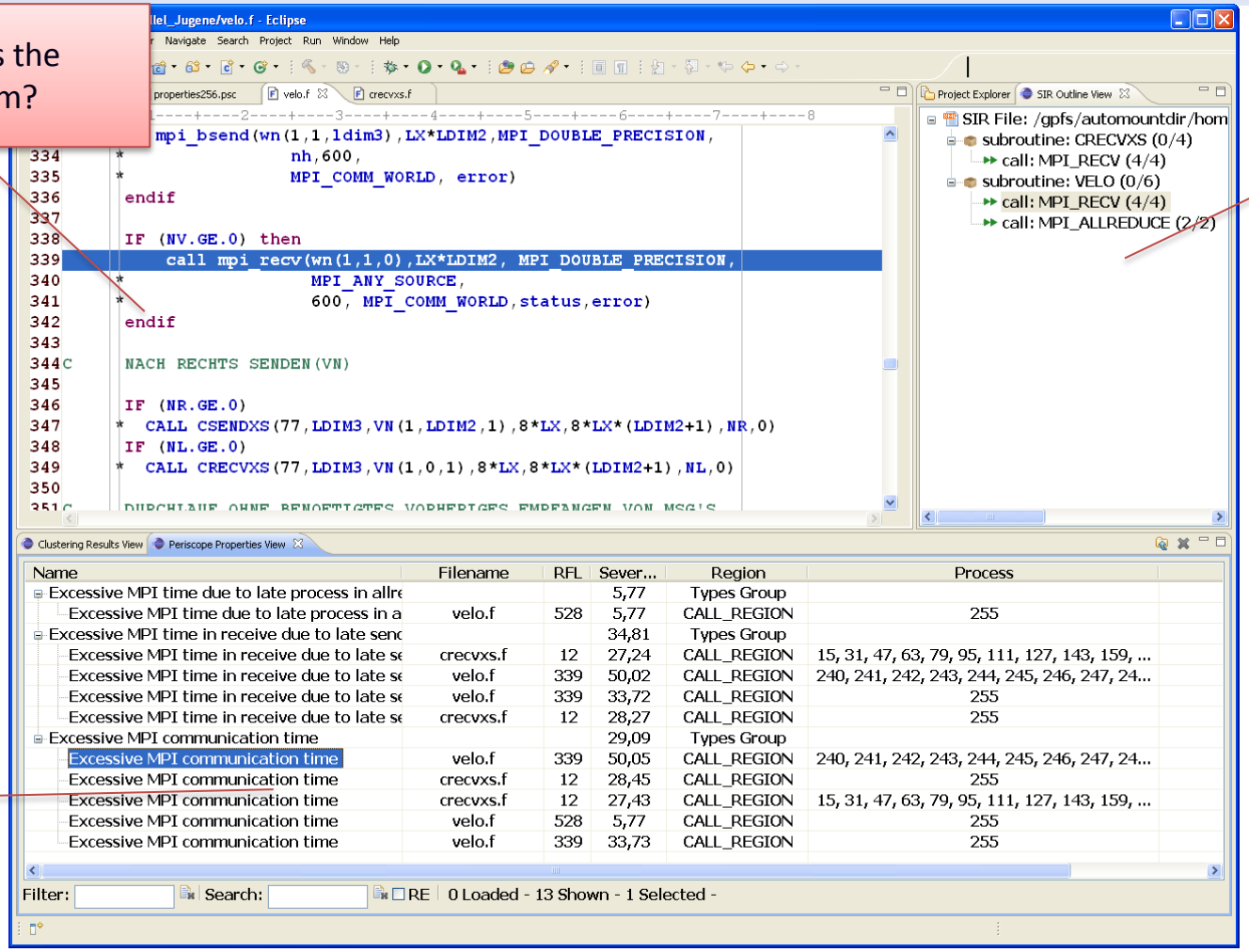
AUTOTUNE

www.autotune-project.eu

Where is the problem?

Filter problems for region

What is the most severe problem?



Name	Filename	RFL	Sever...	Region	Process
Excessive MPI time due to late process in allre			5,77	Types Group	
Excessive MPI time due to late process in a	velo.f	528	5,77	CALL_REGION	255
Excessive MPI time in receive due to late senc			34,81	Types Group	
Excessive MPI time in receive due to late se	crecvxs.f	12	27,24	CALL_REGION	15, 31, 47, 63, 79, 95, 111, 127, 143, 159, ...
Excessive MPI time in receive due to late se	velo.f	339	50,02	CALL_REGION	240, 241, 242, 243, 244, 245, 246, 247, 24...
Excessive MPI time in receive due to late se	velo.f	339	33,72	CALL_REGION	255
Excessive MPI time in receive due to late se	crecvxs.f	12	28,27	CALL_REGION	255
Excessive MPI communication time			29,09	Types Group	
Excessive MPI communication time	velo.f	339	50,05	CALL_REGION	240, 241, 242, 243, 244, 245, 246, 247, 24...
Excessive MPI communication time	crecvxs.f	12	28,45	CALL_REGION	255
Excessive MPI communication time	crecvxs.f	12	27,43	CALL_REGION	15, 31, 47, 63, 79, 95, 111, 127, 143, 159, ...
Excessive MPI communication time	velo.f	528	5,77	CALL_REGION	255
Excessive MPI communication time	velo.f	339	33,73	CALL_REGION	255

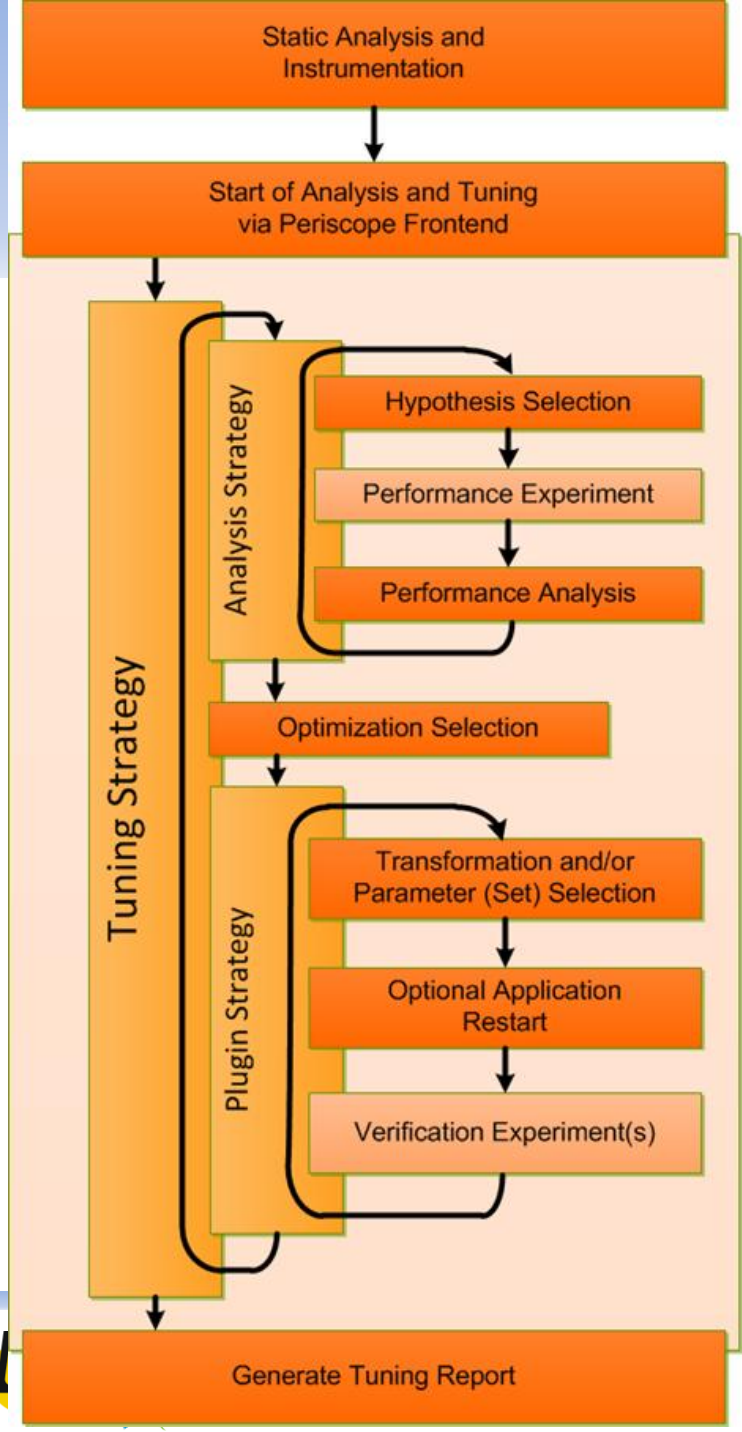
Autotune Approach

AUTOTUNE
www.autotune-project.eu

- 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

AUTOTUNE
www.autotune-project.eu

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

```
#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);
  }
 }
```

- HMPP static information
  - Optimization space description
  - Static code information collect
- Dynamic information collect (i.e. timing, parameter values)

Extensions to Periscope

AUTOTUNE
www.autotune-project.eu

Frontend	Tuning Plugin Search Strategies Scenario Execution Engine
Analysis Agent	Tuning Strategy
Monitor Request Interface	Tuning Action Requests
Monitor	Tuning Actions

Tuning Plugin

AUTOTUNE
www.autotune-project.eu

- 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

User Defined Tuning Points


```
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

AUTOTUNE
www.autotune-project.eu

- 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

AUTOTUNE
www.autotune-project.eu

- 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

AUTOTUNE
www.autotune-project.eu

- 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

AUTOTUNE
www.autotune-project.eu

- 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

AUTOTUNE

www.autotune-project.eu

THANK YOU