Houssam Haitof
Technische Univeristät München
haitof@in.tum.de

AutoTune

AUTOTUNE

www.autotune-project.eu

- AutoTune objective is to combine performance analysis and tuning into a single tool
- AutoTune will extend Periscope with automatic online tuning plugins for performance and energy efficiency tuning.
- The result would be the Periscope Tuning Framework
- The whole tuning process (automatic performance analysis and automatic tuning) will be executed online

AutoTune

AUTOTUNE

www.autotune-project.eu

- Automatic application tuning
  - Performance and energy
- Parallel architectures
  - HPC and parallel servers
  - Homogeneous and heterogeneous
  - Multicore and GPU accelerated systems
  - Reproducable execution capabilities
- Variety of parallel pardigms
  - MPI, HMPP, parallel patterns

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
Performance Analysis Toolkit

AUTOTUNE
www.autotune-project.eu

- Distributed architecture
  - Analysis performed by multiple distributed hierarchical agents
  - Enables locality: Processing of performance data close to its point of generation
- Iterative online analysis
  - Measurements are configured, obtained & evaluated while the program is running
  - no tracing
- Automatic bottlenecks search
  - Based on performance optimization experts' knowledge
- Enhanced GUI
  - Eclipse based integrated development and performance analysis environment
- Instrumentation
  - Fortran, C/C++
  - Automatic overhead control

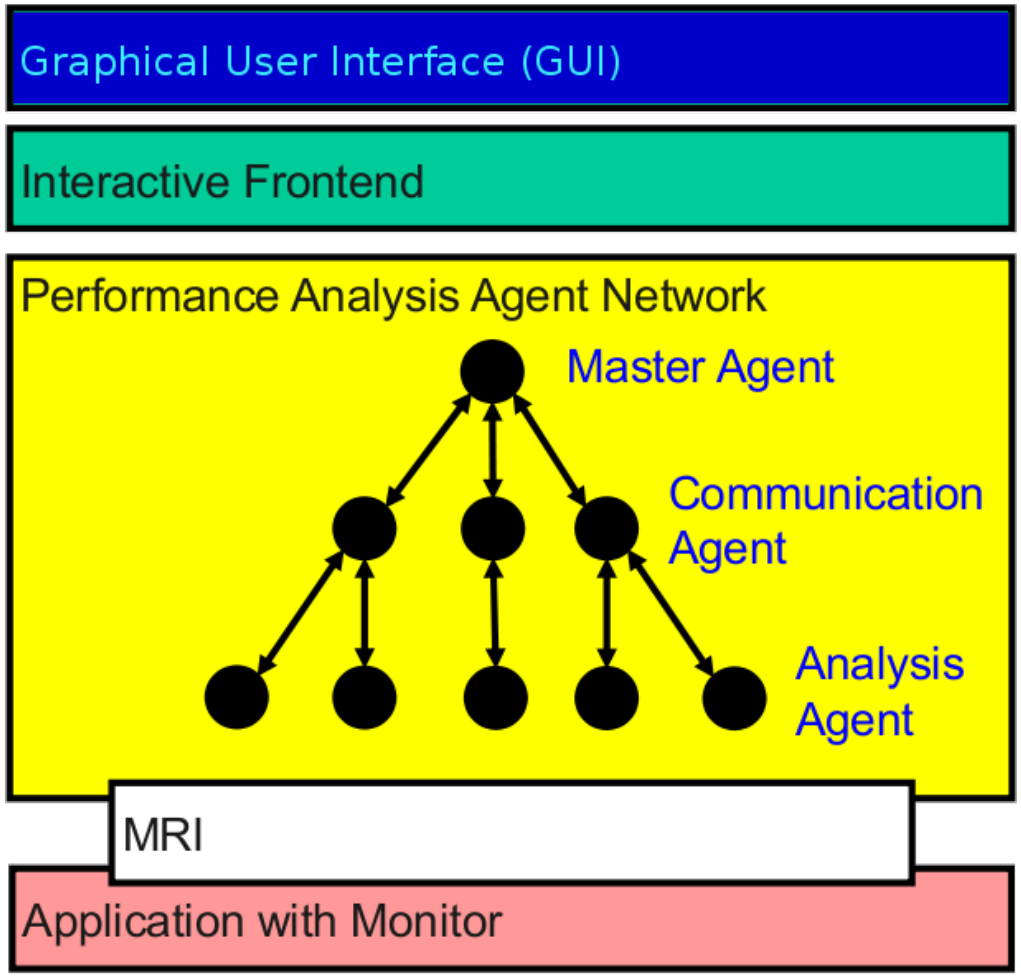
Periscope
Performance Analysis Toolkit

AUTOTUNE
www.autotune-project.eu

- Automation in Periscope is based on formalized *performance properties*
  - e.g., inefficient cache usage or load imbalance
  - Automatically check which measurements are required, which properties were found and which are the properties to look for in the next step
- The overall search for performance problems is determined by search strategies
  - A *search strategy* defines in which order an analysis agent investigates the multidimensional search space of properties, program regions, and processes.
- Periscope provides search strategies for single node performance (e.g., searching for inefficient use of the memory hierarchy), MPI and OpenMP.

Periscope
Performance Analysis Toolkit

AUTOTUNE
www.autotune-project.eu



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

TUM universität wien CAPS UAB Universitat Autònoma de Barcelona lrz ICHEC SEVENTH FRAMEWORK PROGRAMME

Integration in Eclipse (PTP)

AutoTune

www.autotune-project.eu

Where is the problem?

Filter problems for region

What is the most severe problem?

lel_Jugene/velo.f - Eclipse

Navigate Search Project Run Window Help

properties256.psc | velo.f | crecvxs.f

```
334     *                 nh,600,
335     *                 MPI_COMM_WORLD, error)
336      endif
337
338      IF (NV.GE.0) then
339          call mpi_recv(wn(1,1,0),LX*LDIM2, MPI_DOUBLE_PRECISION,
340     *                  MPI_ANY_SOURCE,
341     *                  600, MPI_COMM_WORLD,status,error)
342      endif
343
344C     NACH RECHTS SENDEN(VN)
345
346      IF (NR.GE.0)
347     *  CALL CSENDXS(77,LDIM3,VN(1,LDIM2,1),8*LX,8*LX*(LDIM2+1),NR,0)
348      IF (NL.GE.0)
349     *  CALL CRECVXS(77,LDIM3,VN(1,0,1),8*LX,8*LX*(LDIM2+1),NL,0)
350
```

Project Explorer | SIR Outline View

- SIR File: /gpfs/automountdir/hom
  - subroutine: CRECVXS (0/4)
    - call: MPI_RECV (4/4)
  - subroutine: VELO (0/6)
    - call: MPI_RECV (4/4)
    - call: MPI_ALLREDUCE (2/2)

Clustering Results View | Periscope Properties View

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

Filter: | Search: | RE | 0 Loaded - 13 Shown - 1 Selected -

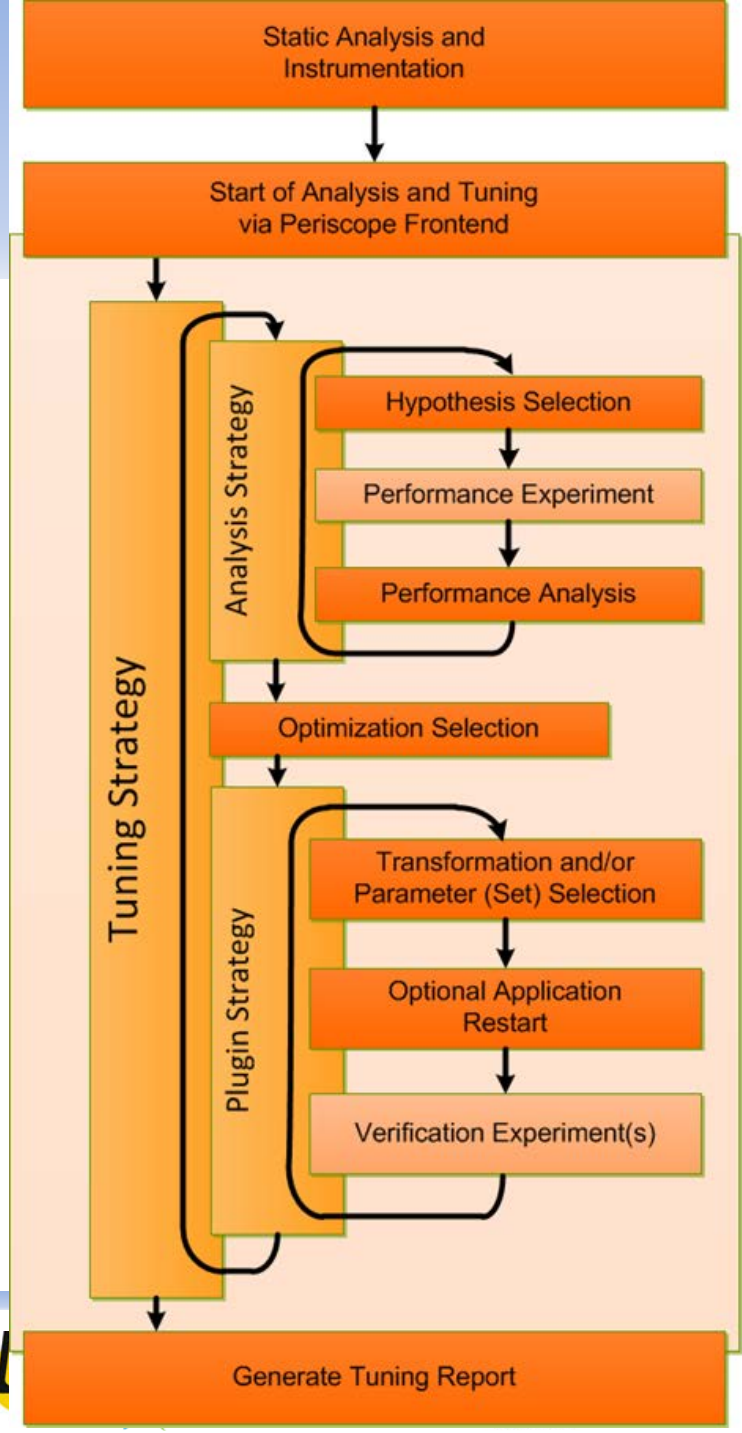
AutoTune Approach

AUTOTUNE
www.autotune-project.eu

- AutoTune will follow Periscope principles
  - Predefined tuning strategies combining performance analysis and tuning, online search, distributd processing...
- Plugins (online and semi-online)
  - 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



Example Plugin

- A demo plugin was developed to demonstrate and validate the auto-tuning extensions to Periscope.

```
do k=1,20
     variant=k
     !$MON USERREGION TP name(Test) variable(variant) variants(10)
     tstart=MPI_Wtime()
     !<user compute code depending on the value of variable variant.>
     tend=MPI_Wtime()
     !$MON END USERREGION
enddo
```

Compiler Flag Selection

- The tuning objective is to reduce the execution time of the application's phase region by selecting the best compiler flags for the application
- Compilers can automatically apply many optimization to the code (e.g., loop interchange, data prefetching, vectorization, and software pipelining)
- It is very difficult to predict the performance impact and to select the right sequence of transformations

Autotuning Extension in HMPP

AUTOTUNE
www.autotune-project.eu

- Directive-based programming model supporting OpenHMPP and OpenACC standards.
- The directives allow the programmer to write hardware independent applications that can significantly speed up C and Fortran code.
- Directives to provide optimization space to explore
  - Parameterized loop transformations and *gridification*
  - 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);
  }
 }
```

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

Parallel Patterns for GPGPU

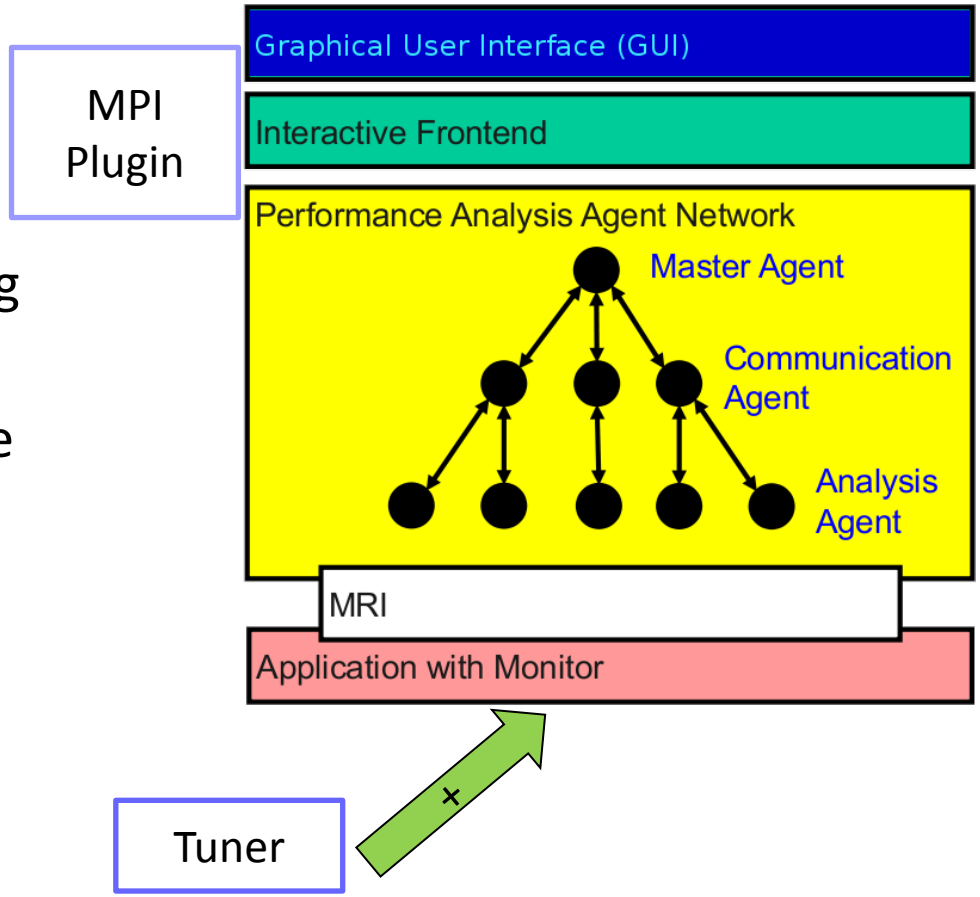
AUTOTUNE
www.autotune-project.eu

- Tuning of high-level patterns for single-node heterogeneous manycore architectures comprising CPUs and GPUs.
- Focus on pipeline patterns, which are expressed in C/C++ programs with annotated while-loops
- Pipeline stages correspond to functions for which different implementation variants may exist
- The associated programming framework has been developed in the EU project PEPPHER

MPI Tuning

AUTOTUNE
www.autotune-project.eu

- MPI Plugin encloses a performance model (ex. M/W) based on Periscope-provided performance data
- Automatically generated tuning decisions are sent to Tuner
- Tuner dynamically modifies the application before next experiment



Energy Efficiency Plugin (LRZ)

AUTOTUNE
www.autotune-project.eu

1. Preparation phase:
   - Selection of possible core frequencies
   - Selection of regions for code instrumentation
2. "While"-loop (until region refinement):
   "For"-loop (all frequencies) :
   a) Set new frequency of tuned region
   b) Periscope analysis (instrumentation, (re-)compiling , start and stop experiment)
   c) Measure execution time + energy of tuned region
   d) Evaluate experiment
   "End for"
3. Evaluate results of all experiments in refinement loop
4. Store best frequencies-region combination

Expected Impact

- Improved performance of applications
- Reduced power consumption of parallel systems
- Facilitated program development and porting
- Reduced time for application tuning
- Leadership of European performance tools groups
- Strengthened European HPC industry

THANK YOU

BACKUP SLIDES

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 Phases

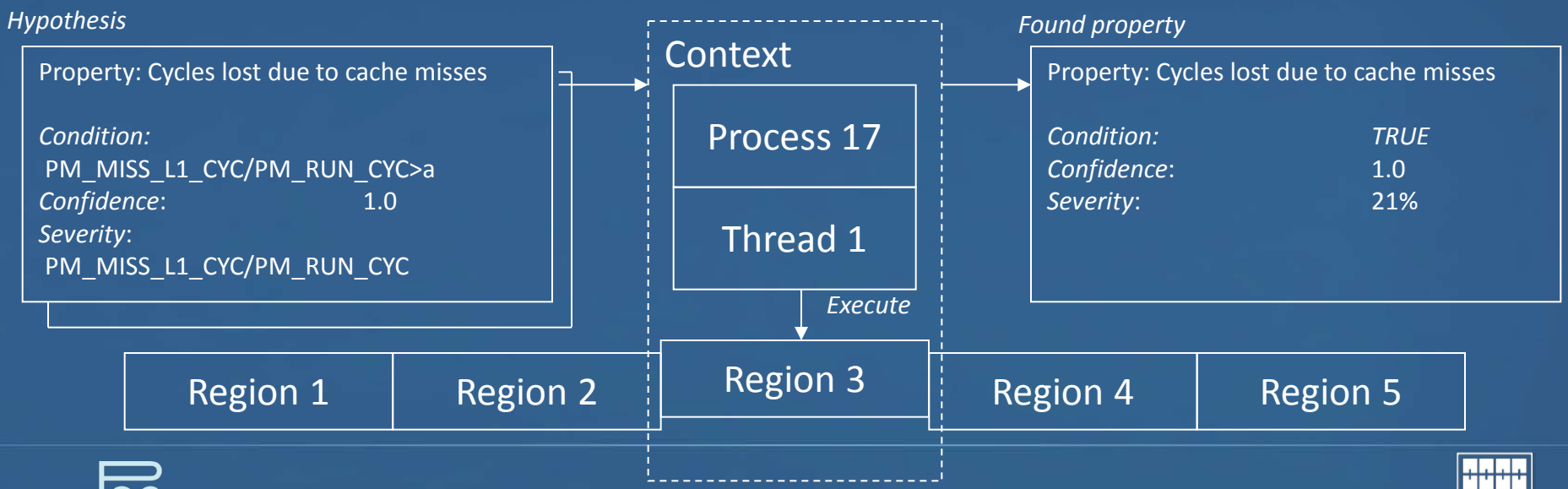
- Periscope performs multiple iterative performance measurement experiments on the basis of *Phases:*
  - All measurements are performed inside phase
  - Begin and end of phase are global synchronization points
- By default phase is the whole program
  - Needs restart if multiple experiments required (single core performance analysis strategies require multiple experiments)
  - Unnecessary code parts also measured
- User specified region in Fortran files that is marked with **`!$MON USER REGION`** and **`!$MON END USER REGION`** will be used as phase:
  - Typically main loop of application → no need for restart, faster analysis
  - Unnecessary code parts are not measured → less measurements overhead
  - Severity value is normalized on the main loop iteration time → more precise performance impact estimation

Automatic search for bottlenecks

- Automation based on formalized expert knowledge
  - Potential performance problems → properties
  - Efficient search algorithm → search strategies
- Performance property
  - Condition
  - Confidence
  - Severity
- Performance analysis strategies
  - Itanium2 Stall Cycle Analysis
  - Westmere-EX Single Core Performance
  - IBM POWER6 Single Core Performance Analysis
  - MPI Communication Pattern Analysis
  - Generic Memory Strategy
  - OpenMP-based Performance Analysis
  - Scalability Analysis – OpenMP codes

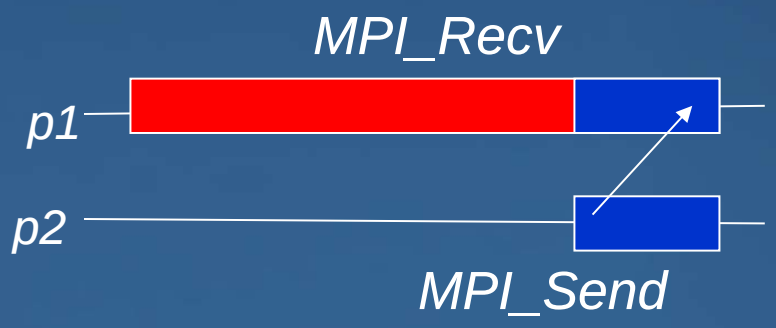
Periscope Performance Property

- Based on APART Specification Language:
  - *“ **Performance Property:** A performance property (e.g. load imbalance, communication, cache misses, redundant computations, etc.) characterizes a specific performance behavior of a program (or part) and can be checked by a set of *conditions*. Conditions are associated with a *confidence value* (between 0 and 1) indicating the degree of confidence about the existence of a performance property. In addition, for every performance property a *severity measure* is provided the magnitude of which specifies the importance of the property. The severity can be used to focus effort on the important performance issues during the (manual or automatic) performance tuning process. Performance properties, confidence and severity are defined over performance-related data. “*



Example Properties

- Long Latency Instruction Exception
  - Condition
    - Percentage of exceptions weighted over region time >30%
  - Severity
    - Percentage of exceptions weighted over region time
- MPI Late Sender
  - Automatic detection of wait patterns
  - Measurement on the fly
  - No tracing required!



- OpenMP Synchronization properties
  - Critical section overhead property
  - Frequent atomic property

Scalability Analysis – OpenMP codes

- Identifies the OpenMP code regions that do not scale well
- Scalability Analysis is done by the frontend / restarts the application /
- **No need to manually configure the runs and find the speedup!**