Performance Analysis with Periscope

M. Gerndt, V. Petkov, Y. Oleynik
Technische Universität München

periscope@lrr.in.tum.de

September 2011

Outline

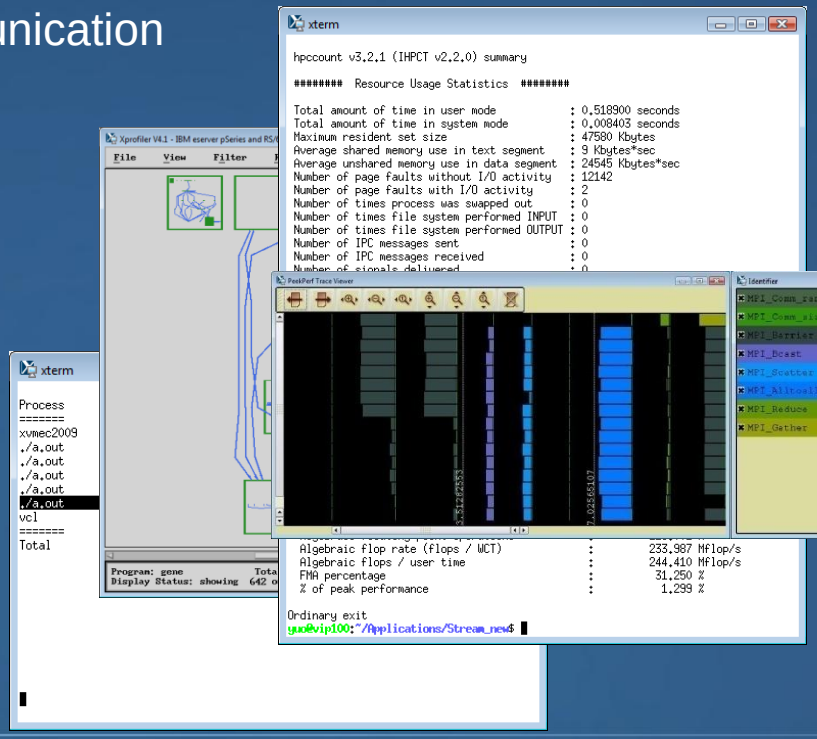
- Motivation
- Periscope overview
- Periscope performance analysis model
- Performance analysis automation
- Periscope GUI

Motivation

- Performance analysis procedure on POWER6 as a example:
  - Use Tprof to pinpoint time consuming subroutines
  - Use Xprofiler (GUI for gprof) to understand call graph
  - Use hpmcount (libhpm) to measure Hardware Counters
  - Use mpitrace to investigate mpi communication

- Problems:
  - Time consuming
  - Error prone
  - Not scalable
  - Requires deep hardware knowledge

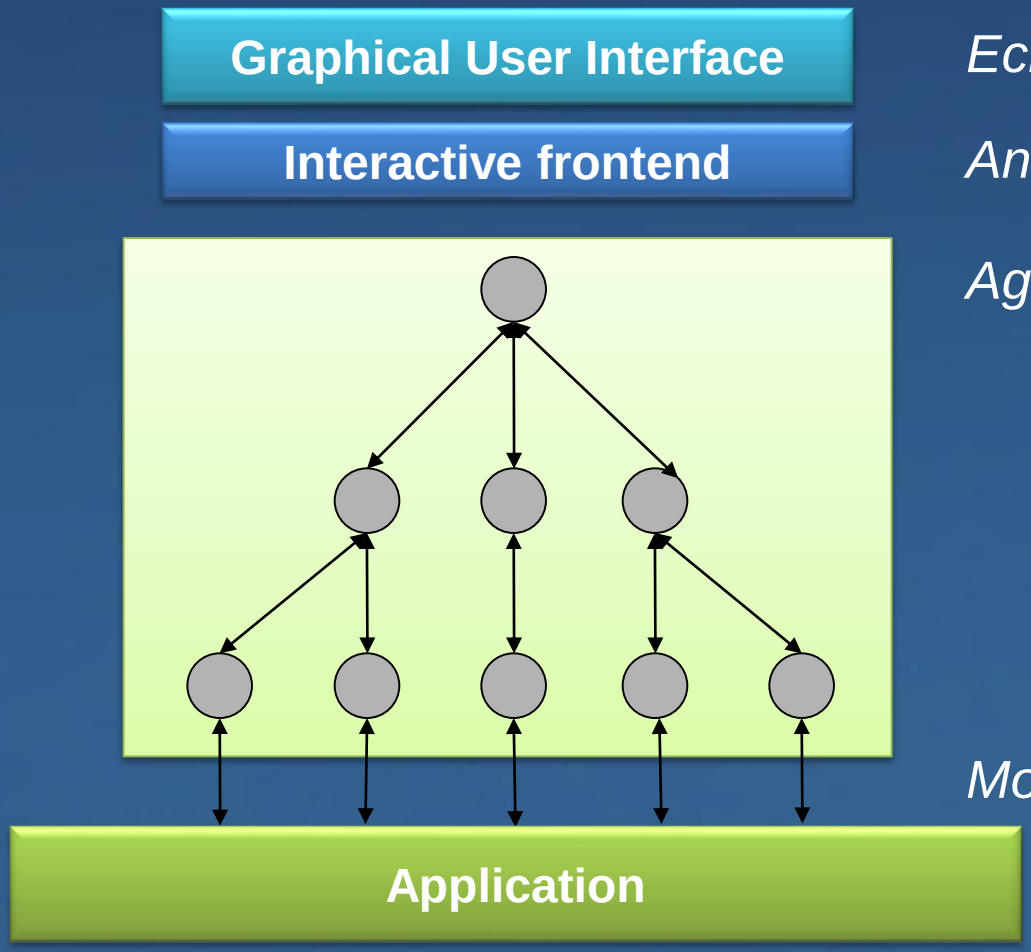
- Solution:
  - Performance analysis automation



Periscope

- Iterative online analysis
  - Measurements are configured, obtained and evaluated on the fly
  - no tracing!
- Distributed architecture
  - Analysis performed by multiple distributed hierarchical agents
- Automatic bottlenecks search
  - Based on performance optimization experts' knowledge
- Enhanced GUI
  - Eclipse based integrated development and performance analysis environment
- Instrumentation
  - Fortran, C/C++

Distributed Architecture

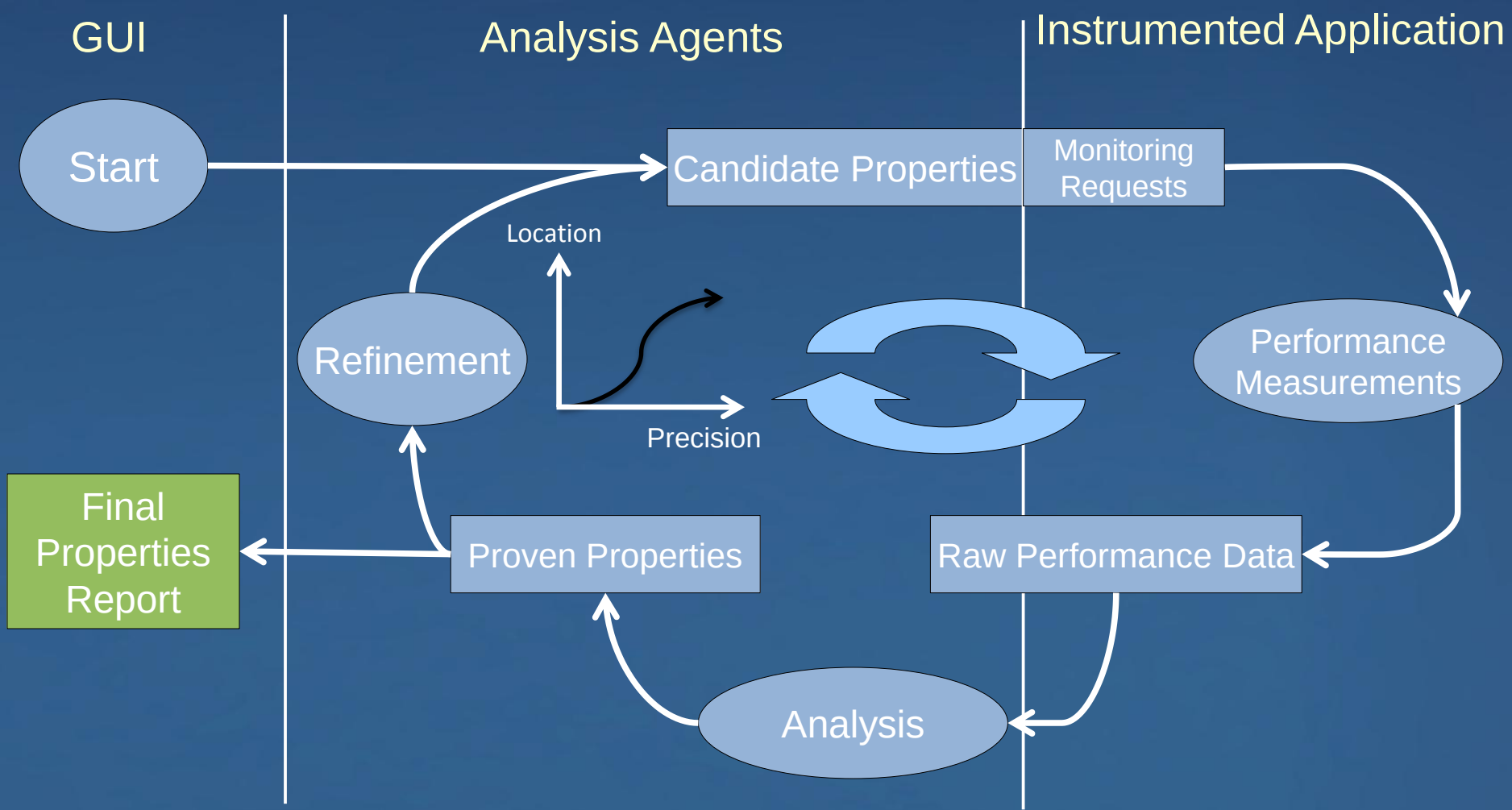


Eclipse-based GUI

Analysis control

Agents network

Monitoring Request Interface


GUI
Analysis Agents
Instrumented Application
Start
Candidate Properties
Monitoring Requests
Location
Refinement
Precision
Performance Measurements
Final Properties Report
Proven Properties
Raw Performance Data
Analysis

Periscope Phases

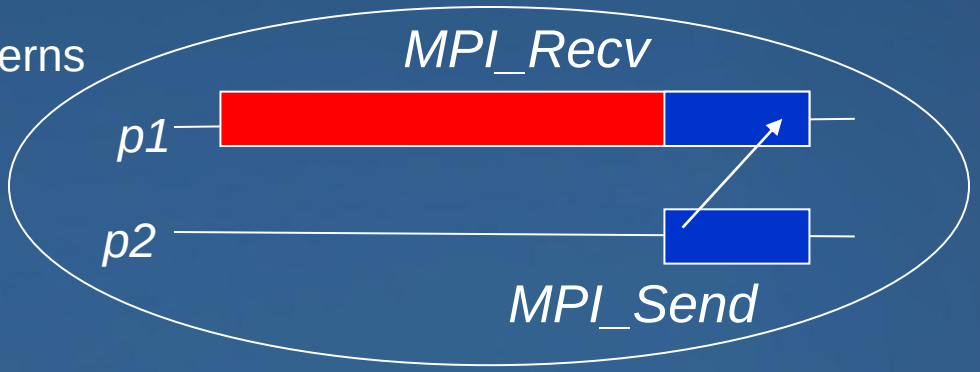
- Periscope performs multiple iterative performance measurement experiments on the basis of *Phases:*
  - All measurements are performed inside phase
  - Begin and end of phase are global synchronization points
- By default phase is the whole program
  - Needs restart if multiple experiments required (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
  - IBM POWER6 Single Core Performance Analysis
  - MPI Communication Pattern Analysis
  - Generic Memory Strategy
  - OpenMP-based Performance Analysis
  - Scalability Analysis – OpenMP codes

Example Properties

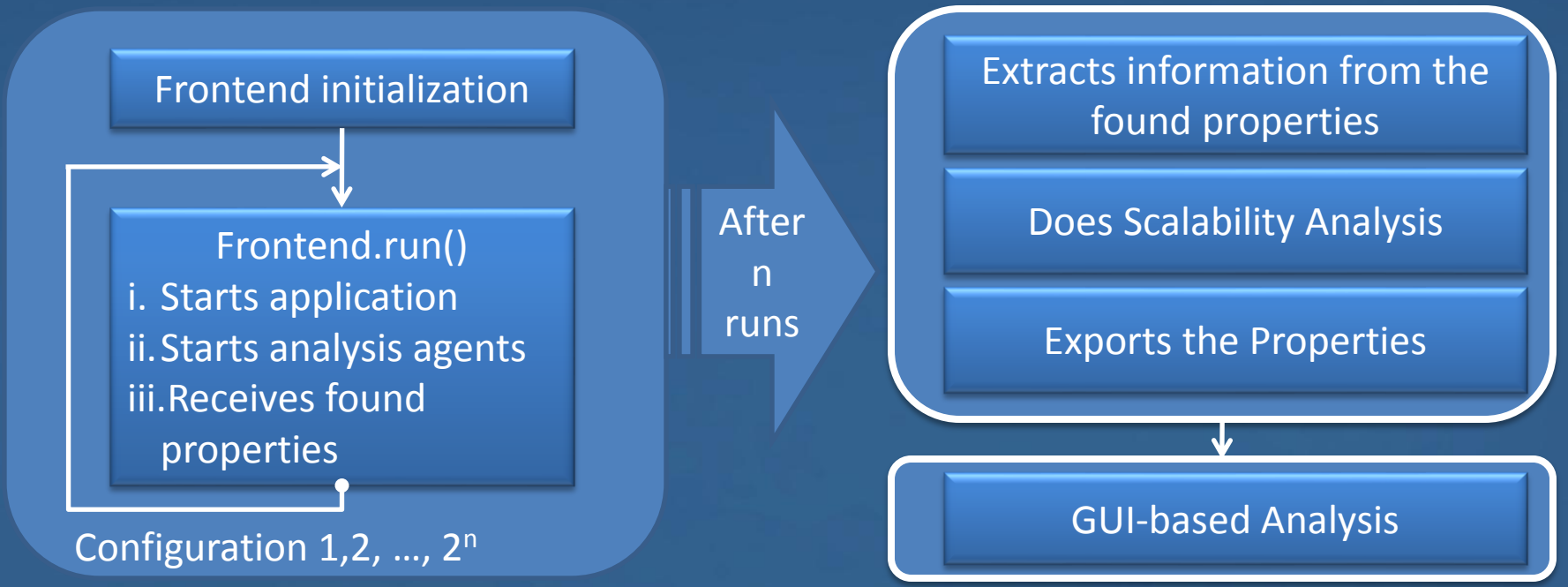
- StallCycles (Region, Rank, Thread, Metric, Phase)
  - Condition
    - Percentage of lost cycles >30%
  - Severity
    - Percentage of lost cycles
- MPI Late Sender
  - Automatic detection of wait patterns
  - Measurement on the fly
  - No tracing required!

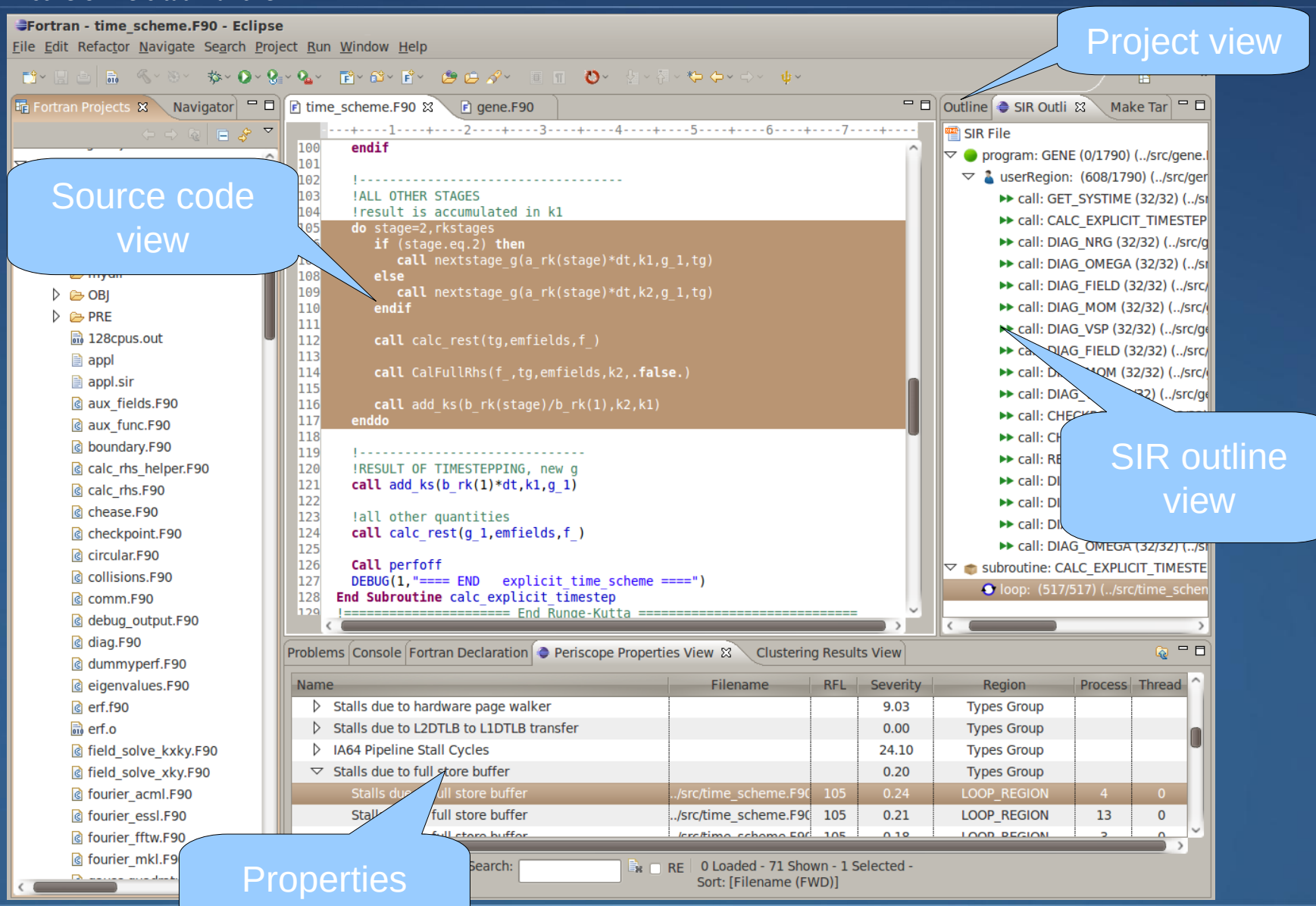


- OpenMP Synchronization properties
  - Critical section overhead property
  - Frequent atomic property

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!**




Fortran - time_scheme.F90 - Eclipse
File Edit Refactor Navigate Search Project Run Window Help
Project view
Source code view
SIR outline view
Properties view
Fortran Projects
Navigator
time_scheme.F90
gene.F90
Outline
SIR Outli
Make Tar
SIR File
Problems
Console
Fortran Declaration
Periscope Properties View
Clustering Results View
Name
Filename
RFL
Severity
Region
Process
Thread
Stalls due to hardware page walker
Stalls due to L2DTLB to L1DTLB transfer
IA64 Pipeline Stall Cycles
Stalls due to full store buffer
Search:
RE
0 Loaded - 71 Shown - 1 Selected -
Sort: [Filename (FWD)]

Thank you for your attention!

- Current version 1.4
  - Available under: http://www.lrr.in.tum.de/periscope/Download
- Supported architectures
  - SGI Altix 4700 Itanium2
  - IBM Power575 POWER6
  - IBM BlueGene/P
  - x86-based architectures
- Further information:
  - Periscope web page: http://www.lrr.in.tum.de/periscope
  - Contact us directly at: periscope@lrr.in.tum.de