Performance Analysis with Periscope

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

periscope@lrr.in.tum.de

April 2012

Outline

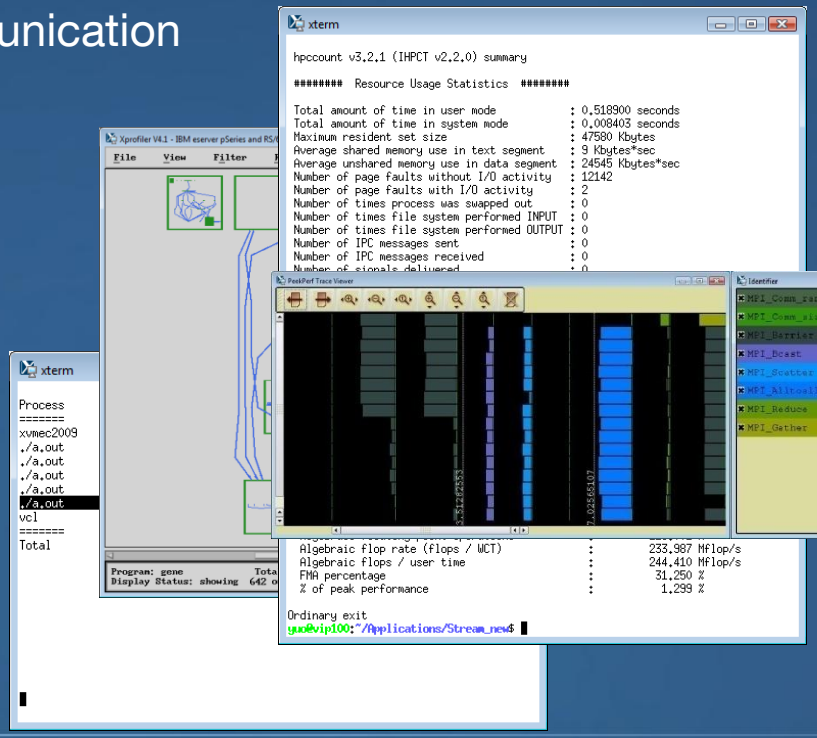
- Motivation
- Periscope overview
- Periscope performance analysis model
- Performance analysis automation
- Periscope GUI

Motivation

- Performance analysis procedure on POWER6 as an 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

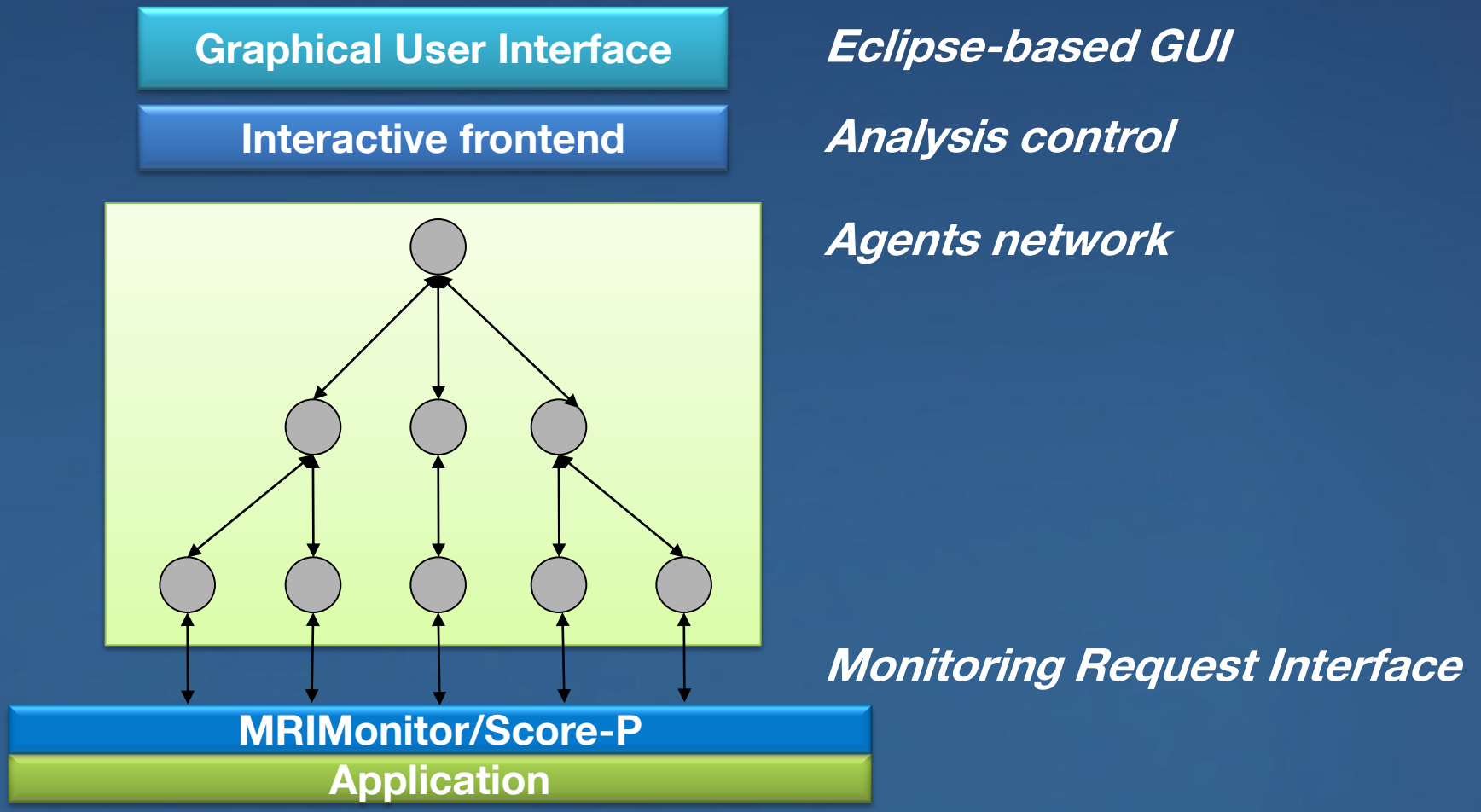
- Solution:
  - Performance analysis automation



Periscope

- Distributed architecture
  - Analysis performed by multiple distributed hierarchical agents
- Iterative online analysis
  - Measurements are configured, obtained and evaluated on the fly
  - no tracing files needed
- 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

Distributed Architecture



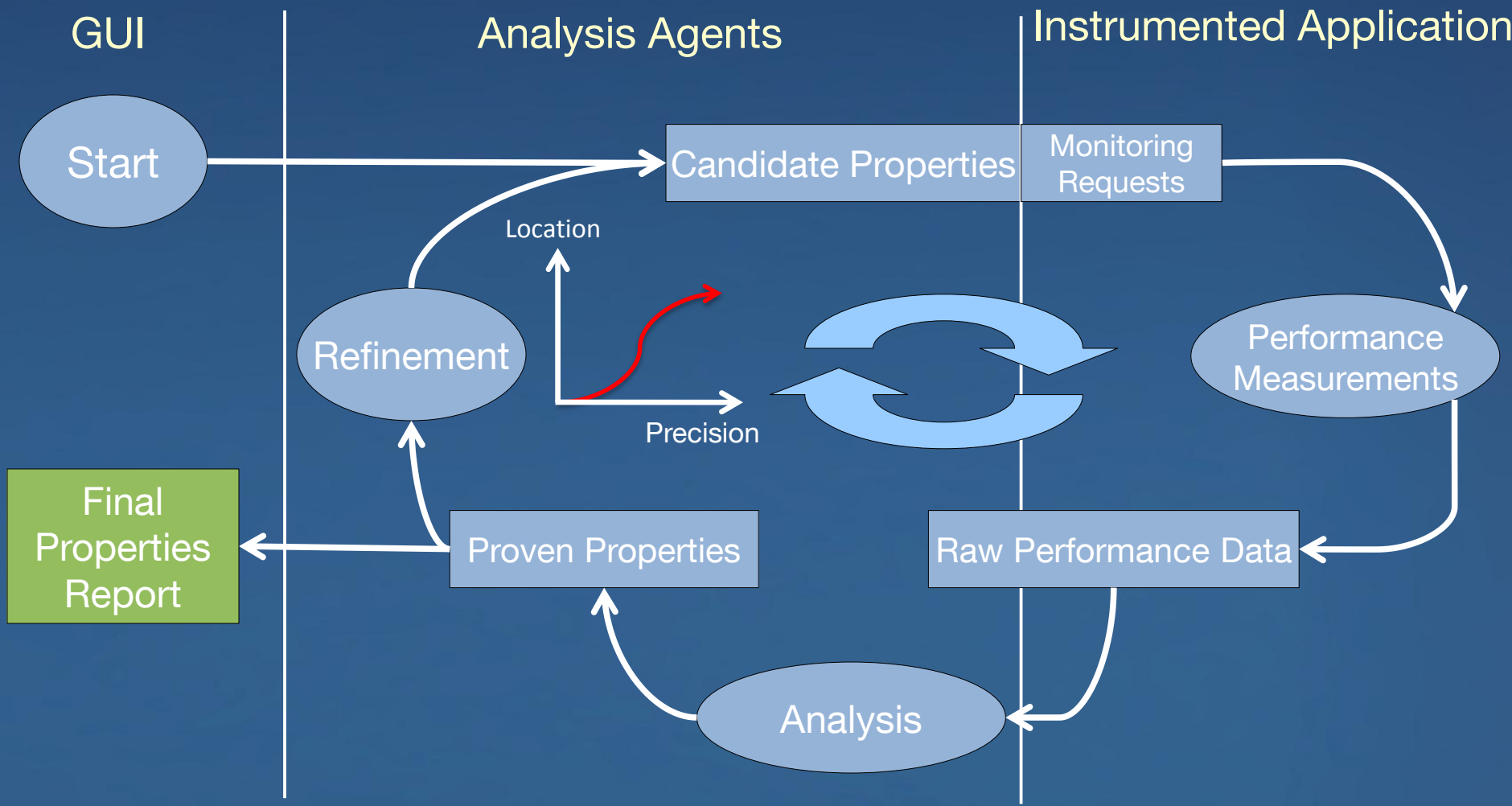
Eclipse-based GUI

Analysis control

Agents network

Monitoring Request Interface

Iterative Online 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
  - Automatic restart might be necessary
- Region needs to me marked as an Online Access Phase to use the Score-P Online Access Interface
  - 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

Definition of Online Access Phases

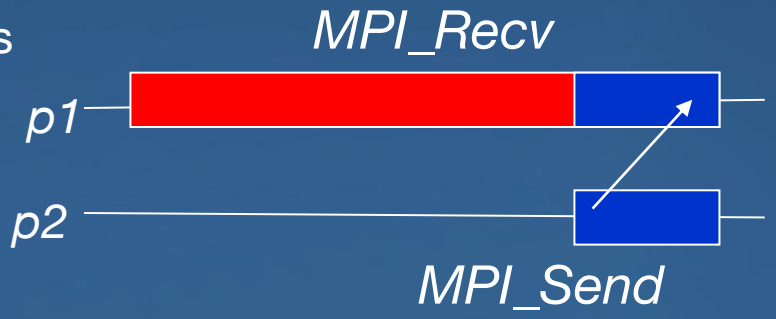
```
#include <scorep/SCOREP_User.h>
void foo()
{
    SCOREP_USER_REGION_DEFINE( my_region_handle )
    for(i=0;...
    {
        SCOREP_USER_OA_PHASE_BEGIN( my_region_handle, \
        "foo",SCOREP_USER_REGION_TYPE_COMMON )
        // do something

        ...

        SCOREP_USER_OA_PHASE_END( my_region_handle )
    }
}
```

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
  - Westmere Single-node Analysis
  - Itanium2 Stall Cycle Analysis
  - IBM POWER6 Single Core Performance Analysis
  - MPI Communication Pattern Analysis
  - Generic Memory Analysis
  - OpenMP-based Performance Analysis
  - Scalability Analysis – OpenMP codes

Example Properties

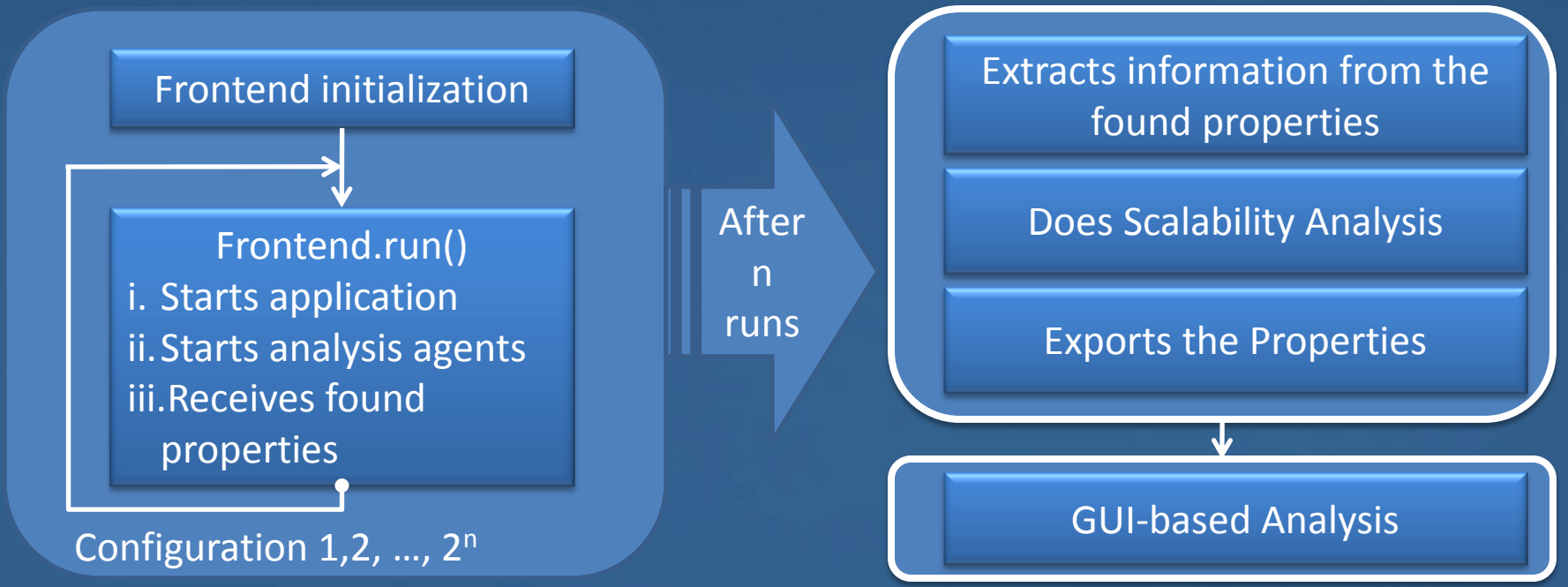
- StallCycles (Region, Rank, Thread, Metric, Phase)
  - Condition
    - Percentage of lost cycles >30%
  - Severity
    - Percentage of lost cycles
- MPI Late Sender
  - Automatic detection of wait patterns
  - Measurement on the fly
  - No tracing required

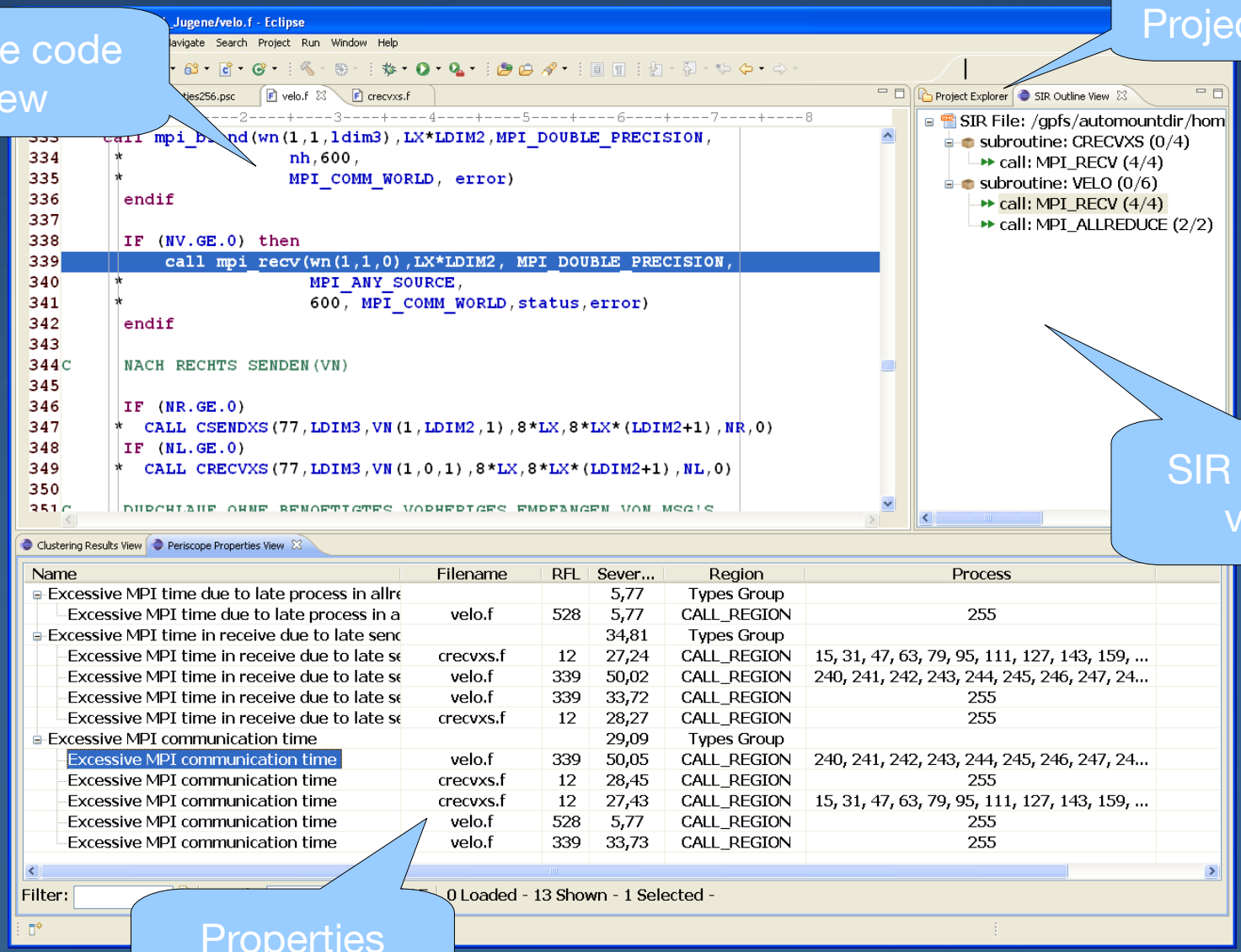


- OpenMP Synchronization properties
  - Critical section overhead property
  - Frequent atomic property

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!





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 send			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

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/x64-based architectures
- Further information:
  - Periscope web page: http://www.lrr.in.tum.de/periscope
  - Contact us directly at: periscope@lrr.in.tum.de