Integrating Parallel Application Development with Performance Analysis in Periscope

V. Petkov, M. Gerndt

Technische Universität München

19 April 2010

Atlanta, GA, USA

Motivation

Common performance analysis procedure on Power6 systems

- Use *Tprof* to pinpoint time-consuming subroutines
- Use *Xprofiler* to understand call graph
- Use *hpmcount (libhpm)* to measure HW Counters

Problem

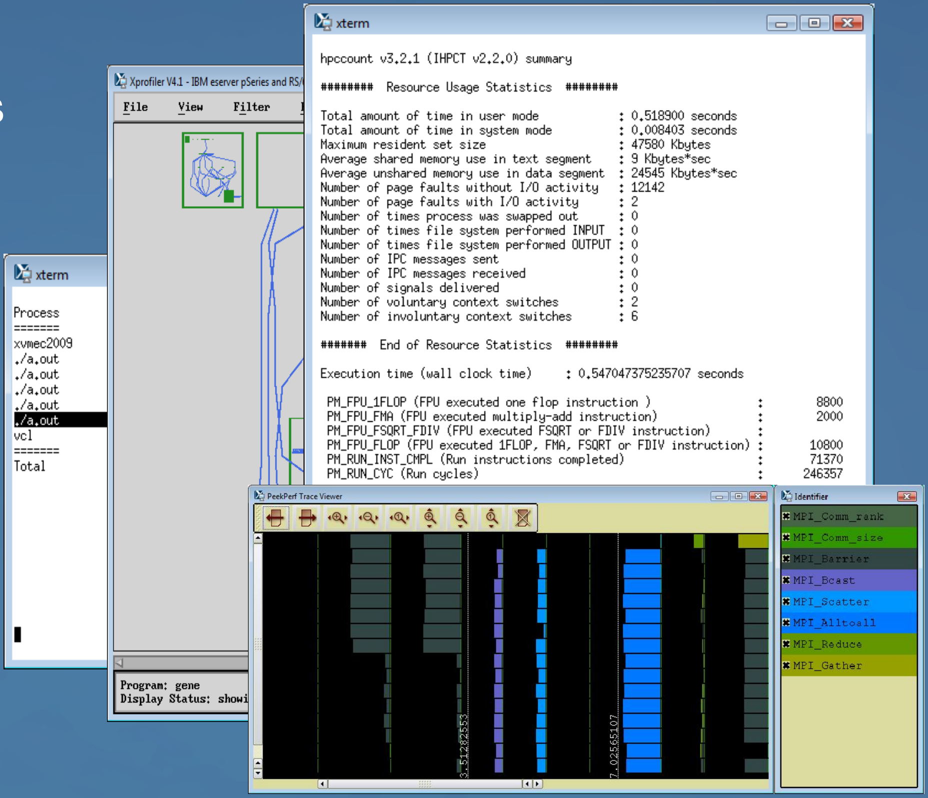
- Mostly post-development process

  → *Learning new tools required*

  → *Hard to map bottlenecks to their source code location*
- Routine, error-prone and time-consuming

Solution

- Automate performance analysis
- Integrate parallel application development and performance analysis within the same IDE



```
Process
=======
xvmec2009
./a.out
./a.out
./a.out
./a.out
./a.out
vcl
=======
Total
```

```
Program: gene
Display Status: showi
```

```
hpccount v3.2.1 (IHPCT v2.2.0) summary

########  Resource Usage Statistics  ########

Total amount of time in user mode            : 0.518900 seconds
Total amount of time in system mode          : 0.008403 seconds
Maximum resident set size                    : 47580 Kbytes
Average shared memory use in text segment    : 9 Kbytes*sec
Average unshared memory use in data segment  : 24545 Kbytes*sec
Number of page faults without I/O activity   : 12142
Number of page faults with I/O activity      : 2
Number of times process was swapped out      : 0
Number of times file system performed INPUT  : 0
Number of times file system performed OUTPUT : 0
Number of IPC messages sent                  : 0
Number of IPC messages received              : 0
Number of signals delivered                  : 0
Number of voluntary context switches         : 2
Number of involuntary context switches       : 6

#######  End of Resource Statistics  ########

Execution time (wall clock time)      : 0.547047375235707 seconds

 PM_FPU_1FLOP (FPU executed one flop instruction )                  :        8800
 PM_FPU_FMA (FPU executed multiply-add instruction)                 :        2000
 PM_FPU_FSQRT_FDIV (FPU executed FSQRT or FDIV instruction)         :
 PM_FPU_FLOP (FPU executed 1FLOP, FMA, FSQRT or FDIV instruction)   :       10800
 PM_RUN_INST_CMPL (Run instructions completed)                      :       71370
 PM_RUN_CYC (Run cycles)                                            :      246357
```

Related Work

- Tools having separate user interfaces
  - Tailored to gain maximum flexibility when presenting the collected data
  - Often hard to map the detected bottlenecks to their exact source location
  - External to user's development environment
    - → impose greater learning overhead
    - → require switching of applications (development/analysis tools)
  - Examples are Vampir, SCALASCA, IBM HPCS Toolkit, etc.

- Tools being integrated in existing IDEs
  - Provide smooth transition between the analysis results and their source code regions
  - Tend to be easier to use as the developers do not have to learn new user interfaces and/or different tools
  - Examples are VTune, TAU, HPCToolkit, PPW, etc.

Periscope performance analysis toolkit

On-line

- no need to store trace files

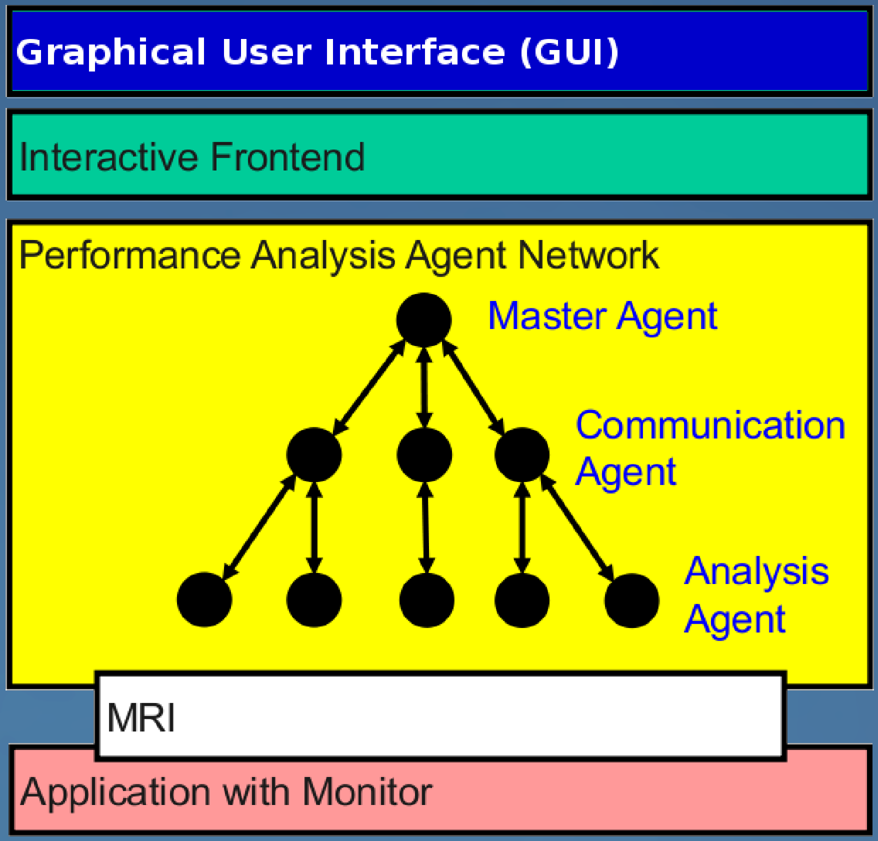
Distributed

- reduced network overhead
- based on autonomous cooperating agents

Analyzes:

- Single-node Performance
  - Intel Itanium2
  - IBM Power6
  - x86-based Systems
- MPI Communication
- OpenMP Performance

Supports: Fortran, C/C++



Periscope GUI Overview

Integrates with the Eclipse Development Platform

- Open-source, extensible and very popular IDE
- Supports different programming languages: C/C++, Fortran, etc.
- Uses the Eclipse Parallel Tools Platform (PTP) which provides a higher-level abstraction of the underlying parallel system

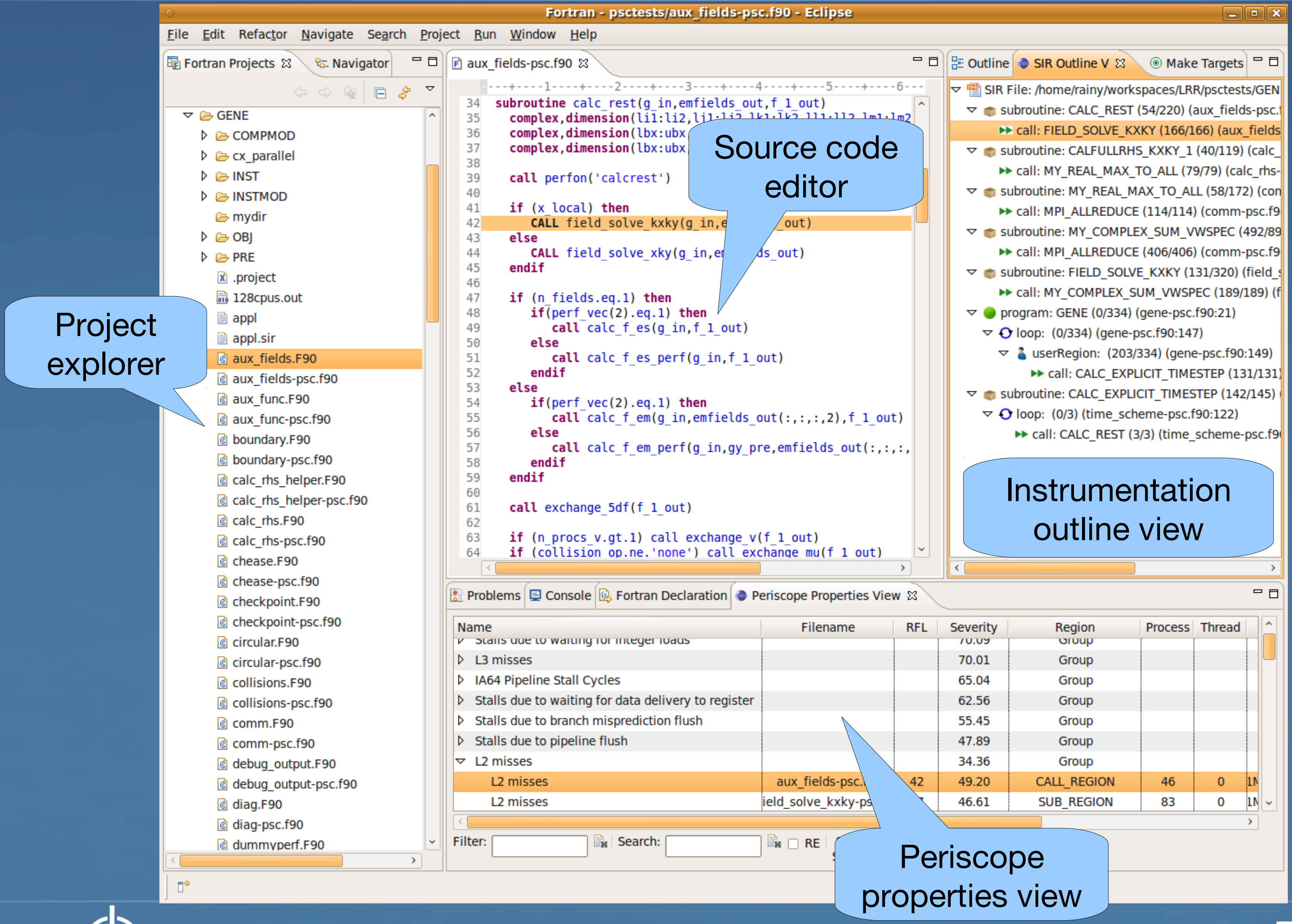
Designed to combine

- Performance measurement functionality of Periscope
- Advanced IDE functions like code indexing, refactoring, etc.

Features

- Multi-functional table to display the detected bottlenecks
- Outline of the instrumented code regions
- Clustering techniques to get classes of similarly behaving processes
- Supports both local and remote projects
- Higher-level configuration and execution of performance experiments

Periscope GUI Overview



Fortran - psctests/aux_fields-psc.f90 - Eclipse
File Edit Refactor Navigate Search Project Run Window Help
Fortran Projects
Navigator
aux_fields-psc.f90
Outline
SIR Outline V
Make Targets
Problems
Console
Fortran Declaration
Periscope Properties View

Project explorer

- GENE
  - COMPMOD
  - cx_parallel
  - INST
  - INSTMOD
  - mydir
  - OBJ
  - PRE
  - .project
  - 128cpus.out
  - appl
  - appl.sir
  - aux_fields.F90
  - aux_fields-psc.f90
  - aux_func.F90
  - aux_func-psc.f90
  - boundary.F90
  - boundary-psc.f90
  - calc_rhs_helper.F90
  - calc_rhs_helper-psc.f90
  - calc_rhs.F90
  - calc_rhs-psc.f90
  - chease.F90
  - chease-psc.f90
  - checkpoint.F90
  - checkpoint-psc.f90
  - circular.F90
  - circular-psc.f90
  - collisions.F90
  - collisions-psc.f90
  - comm.F90
  - comm-psc.f90
  - debug_output.F90
  - debug_output-psc.f90
  - diag.F90
  - diag-psc.f90
  - dummyperf.F90

Source code editor

```
---+----1----+----2----+----3----+----4----+----5----+----6---
subroutine calc_rest(g_in,emfields_out,f_1_out)
  complex,dimension(li1:li2,lj1:lj2,lk1:lk2,ll1:ll2,lm1:lm2
  complex,dimension(lbx:ubx,
  complex,dimension(lbx:ubx,

  call perfon('calcrest')

  if (x_local) then
     CALL field_solve_kxky(g_in,e          _out)
  else
     CALL field_solve_xky(g_in,er       ds_out)
  endif

  if (n_fields.eq.1) then
     if(perf_vec(2).eq.1) then
        call calc_f_es(g_in,f_1_out)
     else
        call calc_f_es_perf(g_in,f_1_out)
     endif
  else
     if(perf_vec(2).eq.1) then
        call calc_f_em(g_in,emfields_out(:,:,:,2),f_1_out)
     else
        call calc_f_em_perf(g_in,gy_pre,emfields_out(:,:,:,
     endif
  endif

  call exchange_5df(f_1_out)

  if (n_procs_v.gt.1) call exchange_v(f_1_out)
  if (collision_op.ne.'none') call exchange_mu(f_1_out)
```

Instrumentation outline view

- SIR File: /home/rainy/workspaces/LRR/psctests/GEN
  - subroutine: CALC_REST (54/220) (aux_fields-psc.f
    - call: FIELD_SOLVE_KXKY (166/166) (aux_fields
  - subroutine: CALFULLRHS_KXKY_1 (40/119) (calc_
    - call: MY_REAL_MAX_TO_ALL (79/79) (calc_rhs-
  - subroutine: MY_REAL_MAX_TO_ALL (58/172) (con
    - call: MPI_ALLREDUCE (114/114) (comm-psc.f9
  - subroutine: MY_COMPLEX_SUM_VWSPEC (492/89
    - call: MPI_ALLREDUCE (406/406) (comm-psc.f9
  - subroutine: FIELD_SOLVE_KXKY (131/320) (field_s
    - call: MY_COMPLEX_SUM_VWSPEC (189/189) (f
  - program: GENE (0/334) (gene-psc.f90:21)
    - loop: (0/334) (gene-psc.f90:147)
      - userRegion: (203/334) (gene-psc.f90:149)
        - call: CALC_EXPLICIT_TIMESTEP (131/131)
  - subroutine: CALC_EXPLICIT_TIMESTEP (142/145) (
    - loop: (0/3) (time_scheme-psc.f90:122)
      - call: CALC_REST (3/3) (time_scheme-psc.f9

Name	Filename	RFL	Severity	Region	Process	Thread
Stalls due to waiting for integer loads			70.09	Group		
L3 misses			70.01	Group		
IA64 Pipeline Stall Cycles			65.04	Group		
Stalls due to waiting for data delivery to register			62.56	Group		
Stalls due to branch misprediction flush			55.45	Group		
Stalls due to pipeline flush			47.89	Group		
L2 misses			34.36	Group		
L2 misses	aux_fields-psc.	42	49.20	CALL_REGION	46	0
L2 misses	ield_solve_kxky-ps		46.61	SUB_REGION	83	0

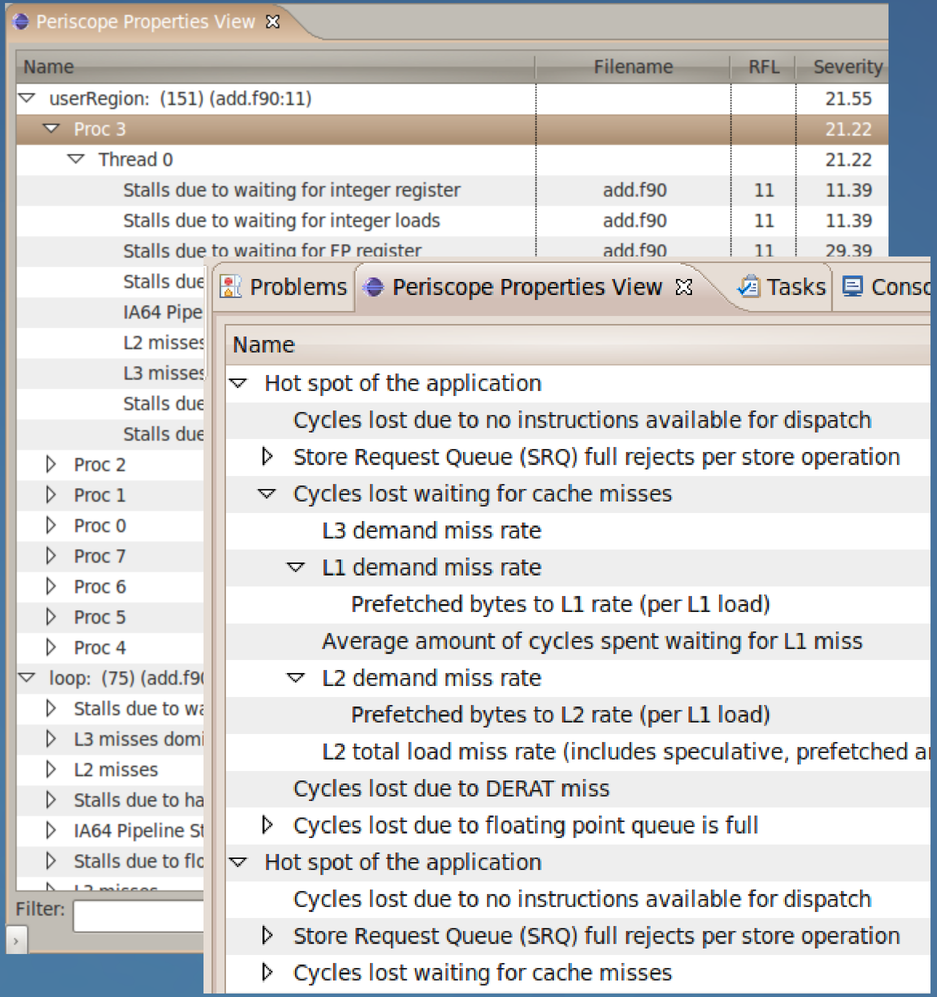
Filter: ______ Search: ______ □ RE

Periscope properties view

Periscope GUI: *Properties Table*

Multi-functional table based on the *OSEE XViewer*

- Simple and clean tree-based overview
- Multi-level grouping
- Complex data filtering
- Multiple criteria sorting algorithm
- Navigation from the properties to their source code location



Periscope Properties View

Name	Filename	RFL	Severity
▽ userRegion: (151) (add.f90:11)			21.55
▽ Proc 3			21.22
▽ Thread 0			21.22
Stalls due to waiting for integer register	add.f90	11	11.39
Stalls due to waiting for integer loads	add.f90	11	11.39
Stalls due to waiting for FP register	add.f90	11	29.39
Stalls due			
IA64 Pipe			
L2 misses			
L3 misses			
Stalls due			
Stalls due			
▷ Proc 2			
▷ Proc 1			
▷ Proc 0			
▷ Proc 7			
▷ Proc 6			
▷ Proc 5			
▷ Proc 4			
▽ loop: (75) (add.f9			
▷ Stalls due to wa			
▷ L3 misses domi			
▷ L2 misses			
▷ Stalls due to ha			
▷ IA64 Pipeline St			
▷ Stalls due to flo			
▷ L2 misses			

Filter:

Problems | Periscope Properties View | Tasks | Conso

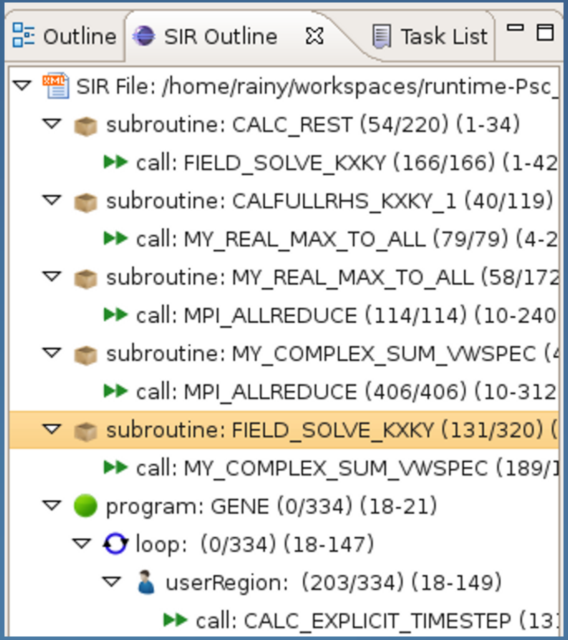
Name

- ▽ Hot spot of the application
  - Cycles lost due to no instructions available for dispatch
  - ▷ Store Request Queue (SRQ) full rejects per store operation
  - ▽ Cycles lost waiting for cache misses
    - L3 demand miss rate
    - ▽ L1 demand miss rate
      - Prefetched bytes to L1 rate (per L1 load)
    - Average amount of cycles spent waiting for L1 miss
    - ▽ L2 demand miss rate
      - Prefetched bytes to L2 rate (per L1 load)
    - L2 total load miss rate (includes speculative, prefetched a
  - Cycles lost due to DERAT miss
  - ▷ Cycles lost due to floating point queue is full
- ▽ Hot spot of the application
  - Cycles lost due to no instructions available for dispatch
  - ▷ Store Request Queue (SRQ) full rejects per store operation
  - ▷ Cycles lost waiting for cache misses

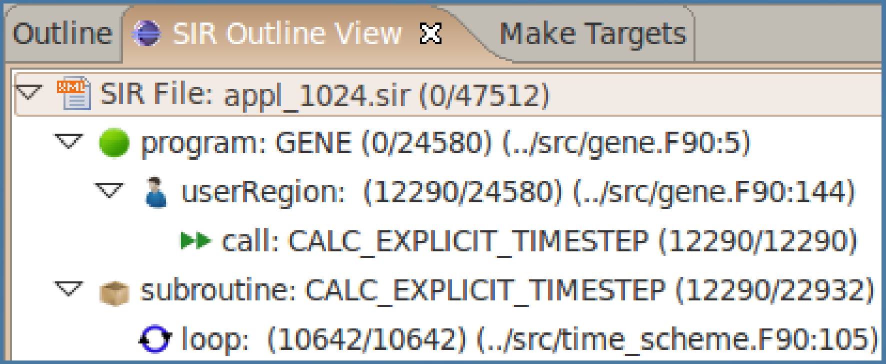
Periscope GUI: *Instrumentation Outline*

Standard Intermediate Representation (SIR) View

- Resembles the code outline view of the Eclipse C/C++ Development Tooling
- Outlines the instrumented code regions and their nesting
- Shows the number of properties in each region
- Assists code navigation
- Filters the displayed properties



```
▽ SIR File: /home/rainy/workspaces/runtime-Psc_
    ▽ subroutine: CALC_REST (54/220) (1-34)
          call: FIELD_SOLVE_KXKY (166/166) (1-42
    ▽ subroutine: CALFULLRHS_KXKY_1 (40/119)
          call: MY_REAL_MAX_TO_ALL (79/79) (4-2
    ▽ subroutine: MY_REAL_MAX_TO_ALL (58/172
          call: MPI_ALLREDUCE (114/114) (10-240
    ▽ subroutine: MY_COMPLEX_SUM_VWSPEC (
          call: MPI_ALLREDUCE (406/406) (10-312
    ▽ subroutine: FIELD_SOLVE_KXKY (131/320) (
          call: MY_COMPLEX_SUM_VWSPEC (189/1
    ▽ program: GENE (0/334) (18-21)
        ▽ loop:  (0/334) (18-147)
            ▽ userRegion:  (203/334) (18-149)
                  call: CALC_EXPLICIT_TIMESTEP (13
```



```
▽ SIR File: appl_1024.sir (0/47512)
    ▽ program: GENE (0/24580) (../src/gene.F90:5)
        ▽ userRegion:  (12290/24580) (../src/gene.F90:144)
              call: CALC_EXPLICIT_TIMESTEP (12290/12290)
    ▽ subroutine: CALC_EXPLICIT_TIMESTEP (12290/22932)
          loop:  (10642/10642) (../src/time_scheme.F90:105)
```

Periscope GUI: *RDT and EFS*

Eclipse File System (EFS)

- Abstracts the underlying file system details

  → *Any supported file system can be used: Remote projects using SSH/FTP/DStore, Local, Zip, etc.*

- Source files of the analyzed application reside only on the remote

  → *no need for synchronization*

Remote Development Tools (RDT)

- Part of Eclipse Parallel Tools Platform (PTP) Project
- Remote Compilation
- Remote Indexing
- Currently supports only C/C++ applications

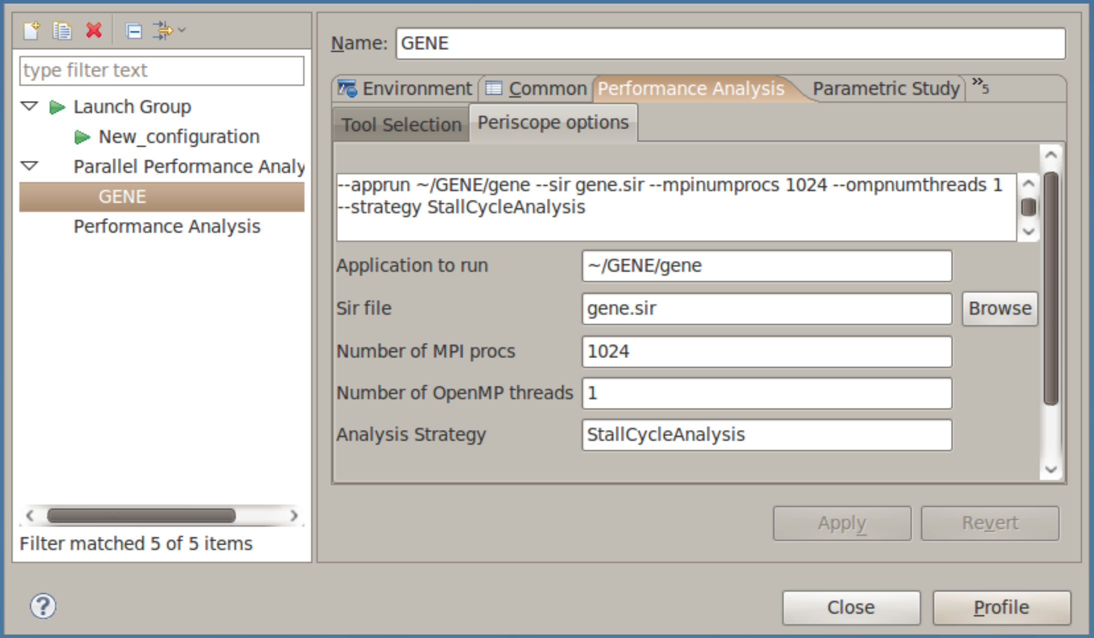
External Tools Framework Integration

External Tools Framework (ETFw)

- Part of Eclipse Parallel Tools Platform (PTP) Project
- More convenient environment using ETFw's `Profile launch configuration`

→ *no terminal access needed*

→ *higher level configuration and automation possible*



Name: GENE

```
--apprun ~/GENE/gene --sir gene.sir --mpinumprocs 1024 --ompnumthreads 1
--strategy StallCycleAnalysis
```

Application to run: ~/GENE/gene

Sir file: gene.sir

Number of MPI procs: 1024

Number of OpenMP threads: 1

Analysis Strategy: StallCycleAnalysis

Clustering support

Properties summarization

- Metaproperties

Needed for peta-scale PA

Identify *hidden* behavior

Based on the Weka workbench:

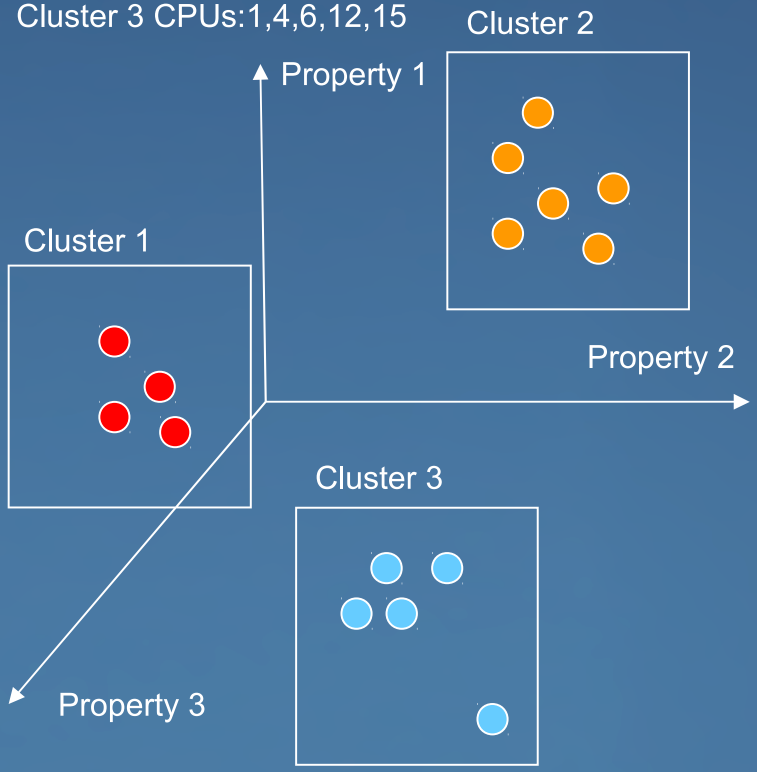
- Waikato Environment for Knowledge Analysis
- Uses K-Means algorithm
- Groups properties based on CPU distribution and code region

Results shown in a table view similar to the properties view

Cluster 1 CPUs: 7-10,16

Cluster 2 CPUs: 2-3,5,11,13-14

Cluster 3 CPUs:1,4,6,12,15



Future Work

Management and comparison of multiple experiments

Enhancing the clustering functionality

- Add pre- and post-processing steps to improve the quality of the results
- Use attribute selection techniques to highlight the most variable data points

Sharing the collected data with other performance tools

- Integrate with a generic performance database, e.g. PerfDMF (TAU)
- Allow the developer to easily apply more than one tool on the same project

Thank you for your attention!

Further information:

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