Periscope
Tutorial Exercise
NPB-MPI/BT

M. Gerndt, Y. Oleynik, V. Petkov

Technische Universität München

periscope@in.tum.de

September 2011

NPB-BT Exercise

- Intermediate-level tutorial example

- Available in **MPI**, OpenMP, hybrid OpenMP/MPI variants
  - also MPI File I/O variants (collective & individual)

- Automatic performance properties search with Periscope:
  - Source code instrumentation
    - Loops, MPI & application function calls
  - Automatic search for slow MPI communication patterns
  - Results exploration with Eclipse based GUI

- Manual instrumentation optimization

Exercise steps

0. Configuration of Periscope
1. Program instrumentation: `psc_instrument`
2. Periscope analysis: `psc_frontend`
3. Performance properties exploration: `Periscope GUI`

Configuring Periscope

- Before first use of Periscope, one has to create the configuration file `.periscope` in the home directory. Configuration could be copied from `$PERISCOPE_ROOT`:

```
% cp $PERISCOPE_ROOT/etc/periscope.sample ~/.periscope
```

- It should look like:

```
MACHINE          = localhost       // hostname
SITE             = VIHPS
REGSERVICE_HOST = cluster-beta     // host where the registry is running
REGSERVICE_PORT = 50001            // please choose a random port!
APPL_BASEPORT    = 51000           // first port for application
AGENT_BASEPORT   = 50002           // first port agent hierarchy
```

- Install GUI into eclipse from http://www.lrr.in.tum.de/periscope/eclipse/ or use the eclipse with pre-installed GUI available with `module load periscope`

Instrumenting application

- The Periscope agents and the application processes register with a `registry`. It is started in batch via:

```
% psc_regsrv &
```

- To enable performance measurement, the program has to be instrumented. This is done with `psc_instrument`:

```
% psc_instrument
Periscope Source-to-Source Instrumentation Wrapper
Usage: psc_instrument [-t regions] [-n] [-s sir] [-v] [-d] compiler
                 [options] file [libs]
        -t Types of regions to instrument separated by spaces
                 (e.g. -t "user loop call")
        -s Filename for the resulting SIR file (default: appl.sir)
        -v Verbose output
        -d Debug mode: keeps the instrumented source files
                 after the compilation
        -n Prints each step of the compilation instead of executing them
        -i Force Intel compilers
```

Instrumenting NPB-MPI-BT

- Substitute compile/link commands in Makefile definitions (`config/make.def`) with `psc_instrument`:

```
MPIF77 = psc_instrument -i -s ${PROGRAM}.sir -t user,mpi mpif77
FLINK = $(MPIF77)
FFLAGS = -O

mpi-bt: $(OBJECTS)
        $(FLINK) $(FFLAGS) -o mpi-bt $(OBJECTS)
.f.o:
        $(MPIF77) $(FFLAGS) -c $<
```

NPB-MPI-BT instrumentation and build

- Return to root directory and clean-up

```
% make clean
```

- Re-build BT with the original command (B or W version)

```
% make bt CLASS=B NPROCS=16
   =========================================
   =      NAS Parallel Benchmarks 3.3      =
   =      MPI/F77/C                        =
   =========================================
cd BT; make NPROCS=16 CLASS=B SUBTYPE= VERSION=
make[1]: Entering directory `BT'
...
psc_instrument -i -s ... -t user,mpi mpif77 -c  -O -g bt.f
psc_instrument -i -s ... -t user,mpi mpif77 -c  -O -g make_set.f
…
psc_instrument -i -s ... -t "user loop call" mpif77 -O \
-o ../bin.periscope/bt_B.16 bt.o ...
Built executable ../bin.periscope/bt_B.16
make[1]: Leaving directory `BT'
```

- Change directory to bin.periscope

```
% cd bin.periscope
```

Running Periscope

- Periscope is started via the frontend. It automatically starts application and hierarchy of analysis agents.
- Run `psc_frontend --help` for brief usage information

```
% psc_frontend --help
Usage: psc_frontend <options>
  [--help]                (displays this help message)
  [--quiet]               (do not display debug messages)
  [--registry=host:port]  (address of the registry service, optional)
  [--port=n]              (local port number, optional)
  [--maxfan=n]            (max. number of child agents, default=4)
  [--timeout=secs]        (timeout for startup of agent hierarchy)
  [--delay=n]               (search delay in phase executions)
  [--appname=name]
  [--apprun=commandline]
  [--mpinumprocs=number of MPI processes]
  [--ompnumthreads=number of OpenMP threads]
…
  [--strategy=name]
  [--sir=name]
  [--phase=(FileID,RFL)]
  [--debug=level]
```

Running Periscope

- Run Periscope analysis by executing `psc_frontend` with the following command in the batch script psc.lsf/.msub

```
% bsub < psc.lsf   or    msub psc.msub
Check the job output:
% bpeek
[psc_frontend][DBG0:fe] Agent network UP and RUNNING. Starting search.

 NAS Parallel Benchmarks 3.3 -- BT Benchmark
 [...]
 Time step 200
 BT Benchmark Completed.


----------
End Periscope run! Search took 37.57 seconds (33.09 seconds for startup)
```

- Frontend will write the detected properties into the file `properties_MPI_<PID>.psc` in the current directory. It should be copied into the BT source directory

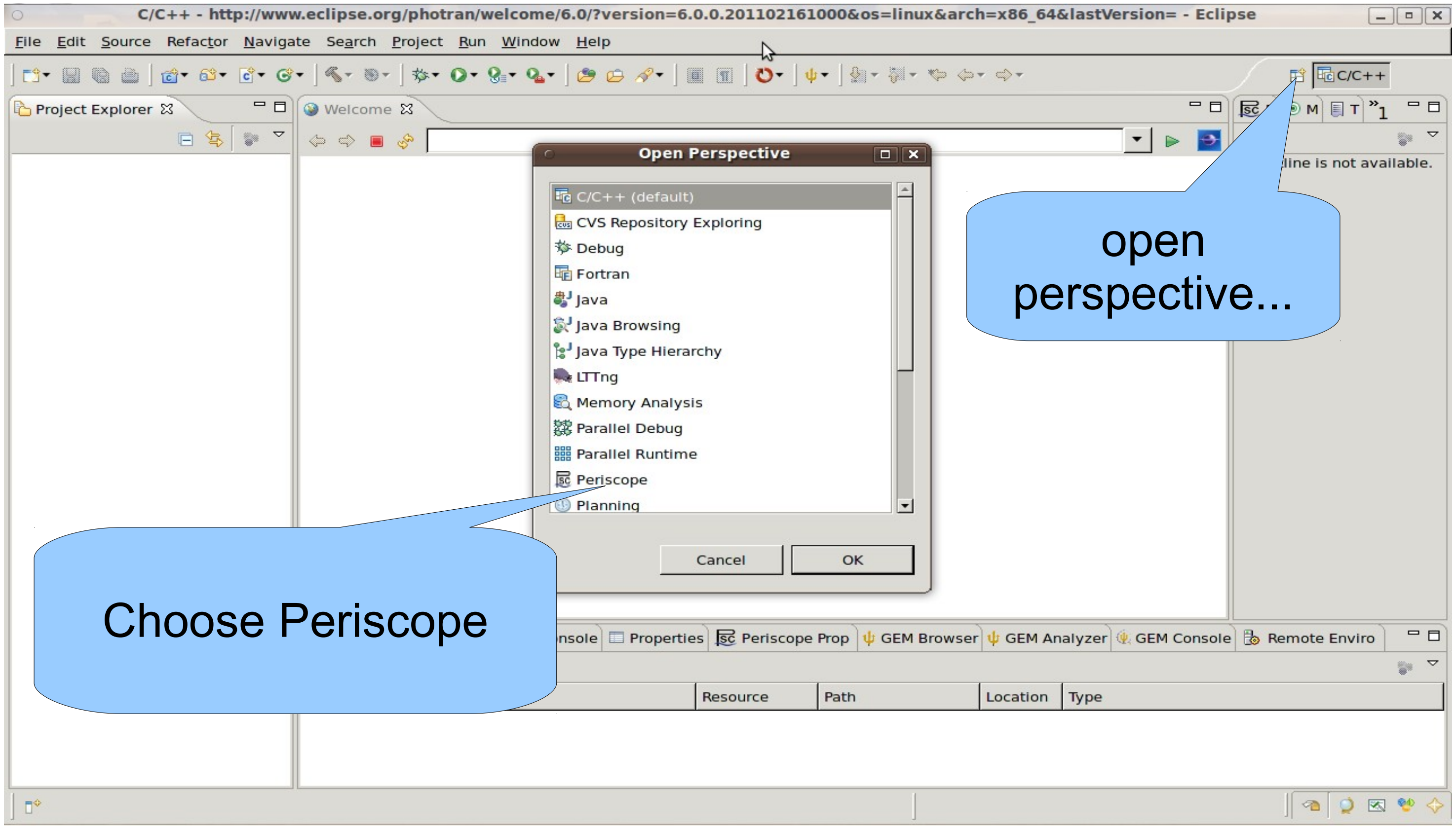
```
% cp properties_MPI_*.psc ../BT
```

Starting Periscope GUI

- Start `Eclipse` with Periscope GUI from console

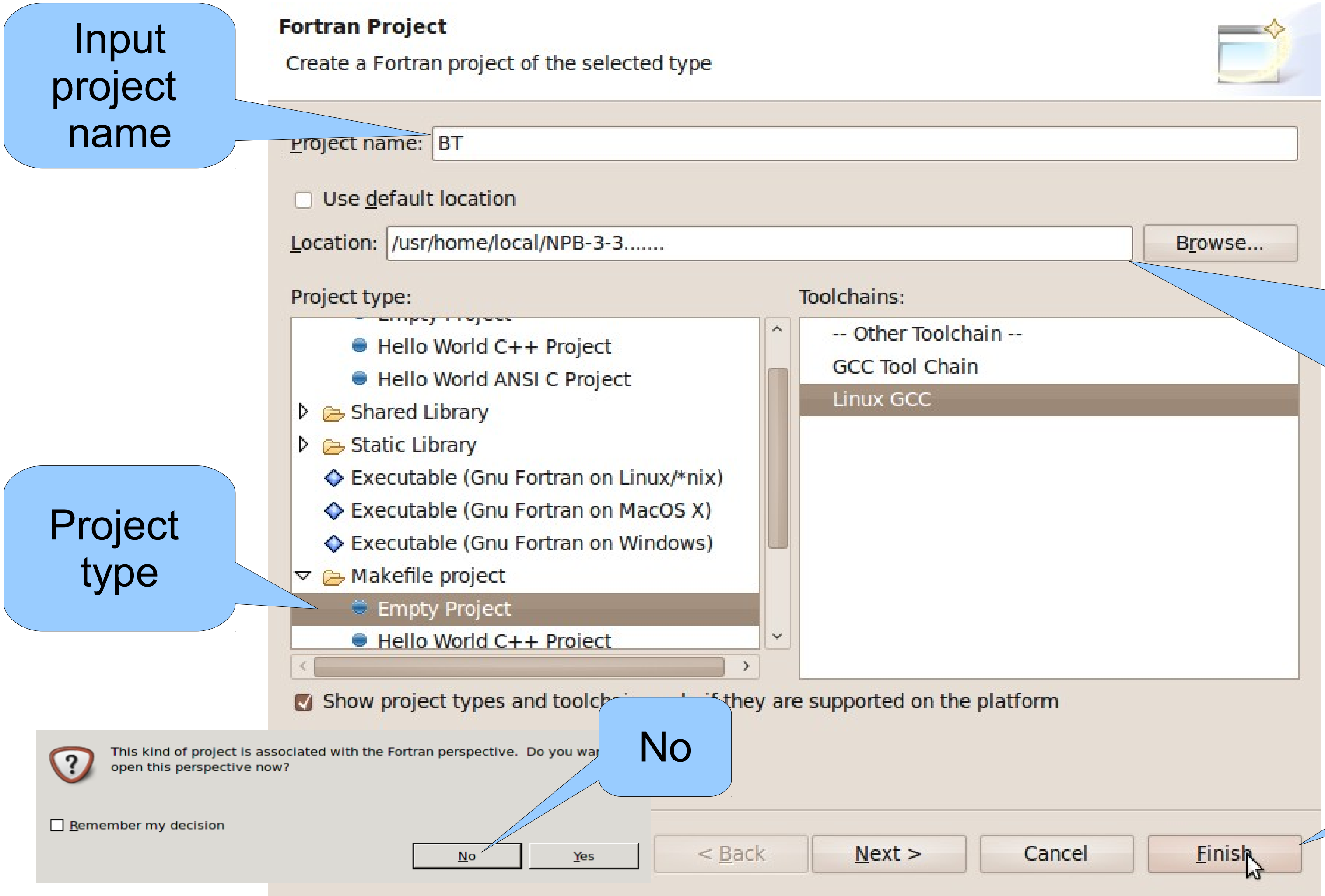
```
% eclipse &
```

- Or by double-click on Eclipse pictogram on the Desktop



Creating Fortran Project

- File->New->Project... → Fortran->Fortran Project



Fortran Project

Create a Fortran project of the selected type

Project name: BT

☐ Use default location

Location: /usr/home/local/NPB-3-3.......

Project type:

- Empty Project
- Hello World C++ Project
- Hello World ANSI C Project
- Shared Library
- Static Library
- Executable (Gnu Fortran on Linux/*nix)
- Executable (Gnu Fortran on MacOS X)
- Executable (Gnu Fortran on Windows)
- Makefile project
  - Empty Project
  - Hello World C++ Project

Toolchains:

- -- Other Toolchain --
- GCC Tool Chain
- Linux GCC

☑ Show project types and toolch if they are supported on the platform

This kind of project is associated with the Fortran perspective. Do you war open this perspective now?

☐ Remember my decision

Input project name

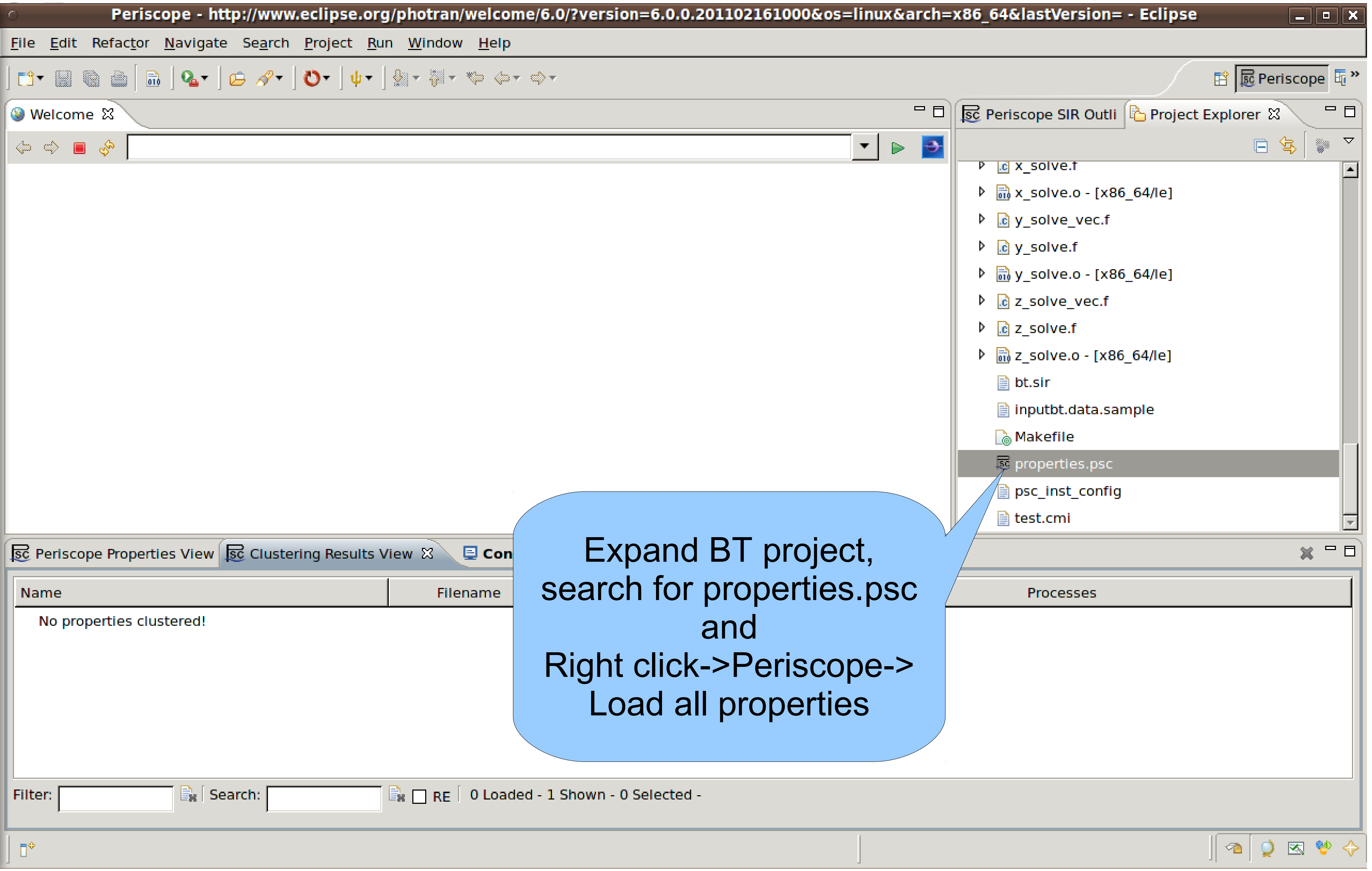
Unmark “Use default location” and provide path to *BT* folder

Project type

No

Press Finish

Loading properties

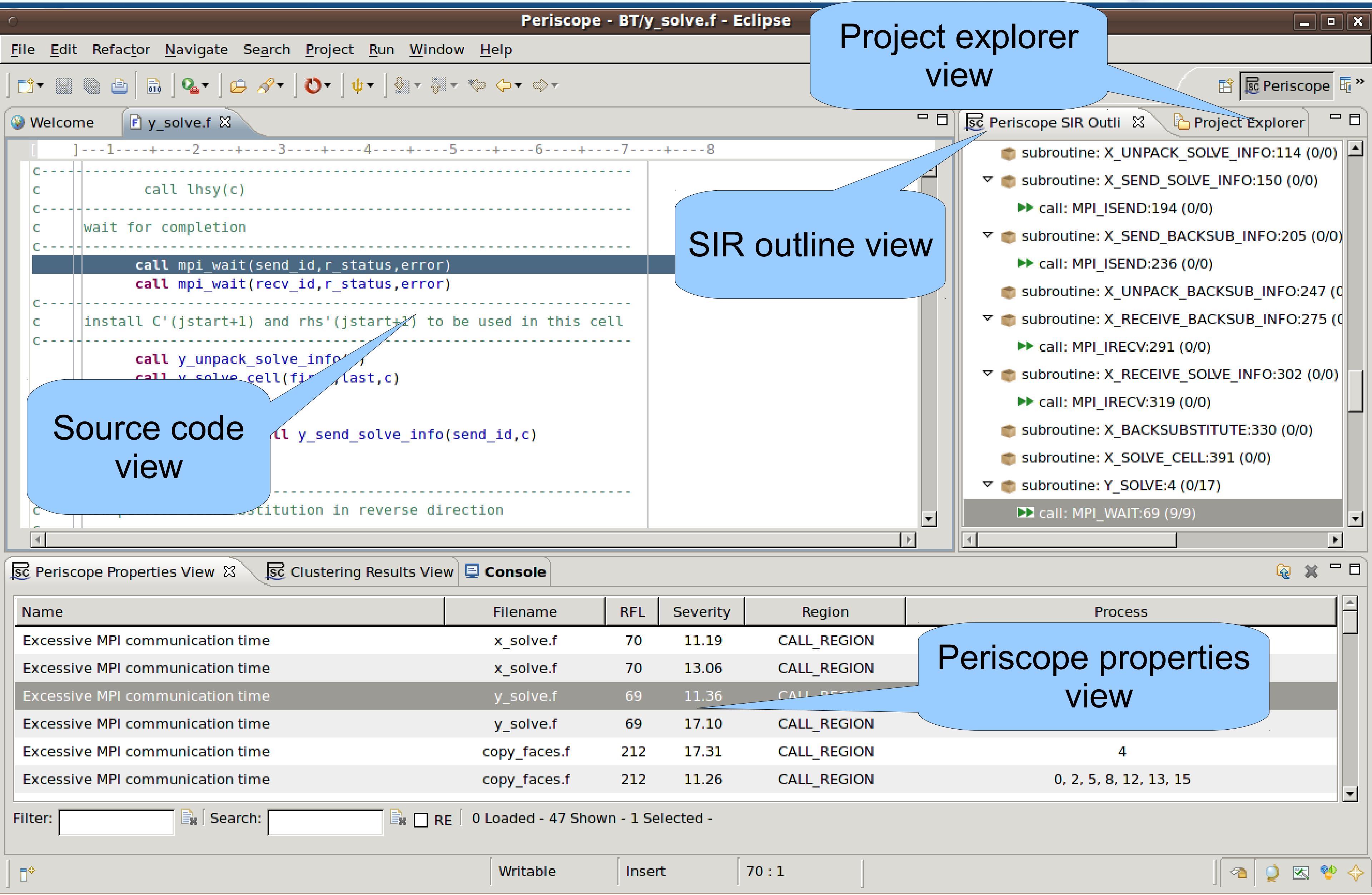


Name	Filename	Processes
No properties clustered!		

0 Loaded - 1 Shown - 0 Selected -

Expand BT project,
search for properties.psc
and
Right click->Periscope->
Load all properties

Periscope GUI


Periscope - BT/y_solve.f - Eclipse
File Edit Refactor Navigate Search Project Run Window Help
Periscope
Welcome
y_solve.f
Periscope SIR Outli
Project Explorer
Periscope Properties View
Clustering Results View
Console

Project explorer view

SIR outline view

Source code view

Periscope properties view

```
c-------------------------------------------------------------------
c         call lhsy(c)
c-------------------------------------------------------------------
c     wait for completion
c-------------------------------------------------------------------
         call mpi_wait(send_id,r_status,error)
         call mpi_wait(recv_id,r_status,error)
c-------------------------------------------------------------------
c     install C'(jstart+1) and rhs'(jstart+1) to be used in this cell
c-------------------------------------------------------------------
         call y_unpack_solve_info(
         call y_solve_cell(fi    ,last,c)

                  ll y_send_solve_info(send_id,c)

                  ------------------------------------------------
c                 stitution in reverse direction
```

- subroutine: X_UNPACK_SOLVE_INFO:114 (0/0)
- subroutine: X_SEND_SOLVE_INFO:150 (0/0)
  - call: MPI_ISEND:194 (0/0)
- subroutine: X_SEND_BACKSUB_INFO:205 (0/0)
  - call: MPI_ISEND:236 (0/0)
- subroutine: X_UNPACK_BACKSUB_INFO:247 (0
- subroutine: X_RECEIVE_BACKSUB_INFO:275 (0
  - call: MPI_IRECV:291 (0/0)
- subroutine: X_RECEIVE_SOLVE_INFO:302 (0/0)
  - call: MPI_IRECV:319 (0/0)
- subroutine: X_BACKSUBSTITUTE:330 (0/0)
- subroutine: X_SOLVE_CELL:391 (0/0)
- subroutine: Y_SOLVE:4 (0/17)
  - call: MPI_WAIT:69 (9/9)

Name	Filename	RFL	Severity	Region	Process
Excessive MPI communication time	x_solve.f	70	11.19	CALL_REGION	
Excessive MPI communication time	x_solve.f	70	13.06	CALL_REGION	
Excessive MPI communication time	y_solve.f	69	11.36	CALL_REGI	
Excessive MPI communication time	y_solve.f	69	17.10	CALL_REGION	
Excessive MPI communication time	copy_faces.f	212	17.31	CALL_REGION	4
Excessive MPI communication time	copy_faces.f	212	11.26	CALL_REGION	0, 2, 5, 8, 12, 13, 15

Filter: Search: □ RE 0 Loaded - 47 Shown - 1 Selected -

Writable Insert 70 : 1

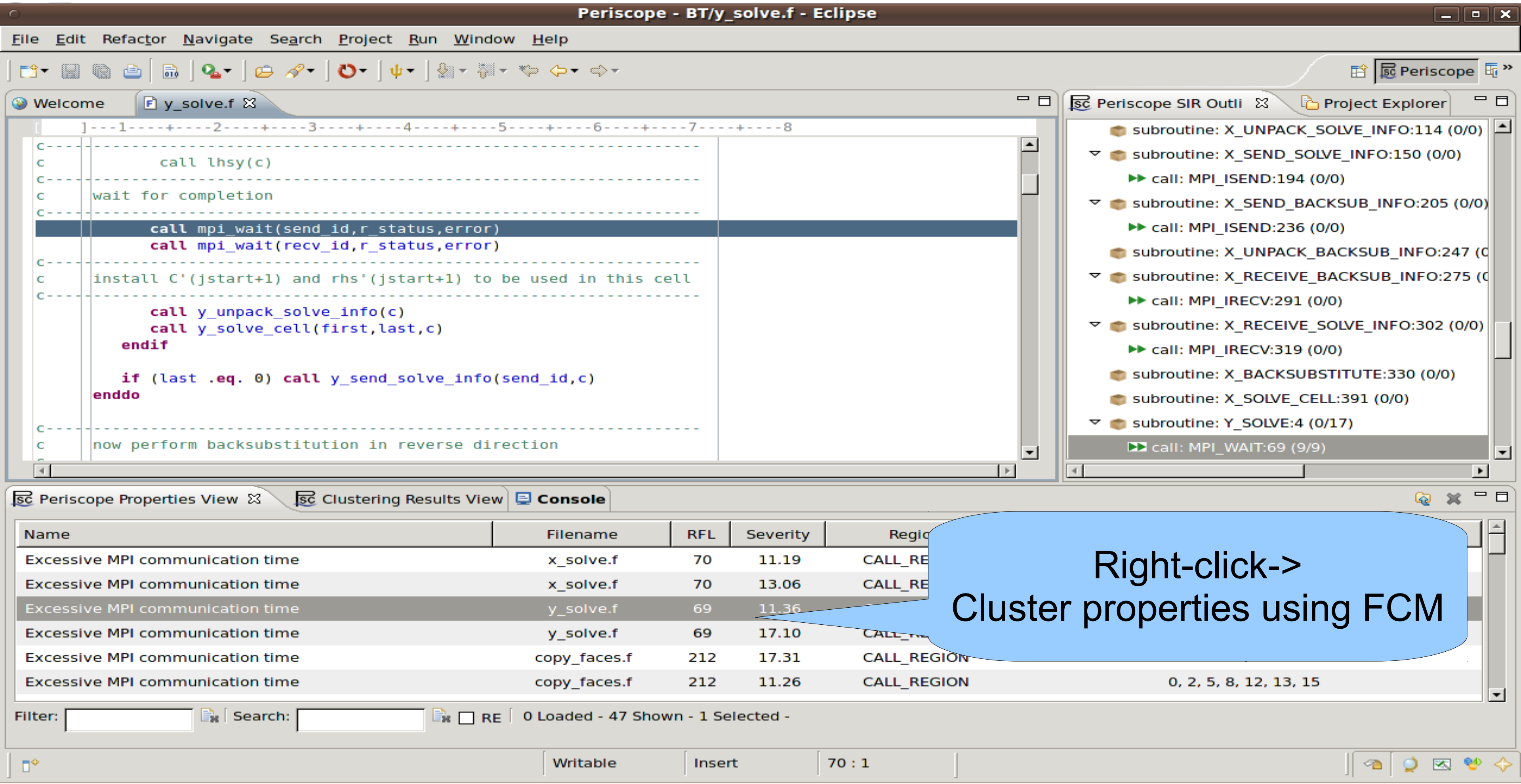
Periscope GUI report exploration features

- Multi-functional table is used in the GUI for Eclipse for the visualization of bottlenecks
  - Multiple criteria sorting algorithm
  - Complex categorization utility
  - Searching engine using Regular Expressions
  - Filtering operations
  - Direct navigation from the bottlenecks to their precise source location using the default IDE editor for that source file type (e.g. CDT/Photran editor).

- SIR outline view shows a combination of the standard intermediate representation (SIR) of the analysed application and the distribution of its bottlenecks. The main goals of this view are to assist the navigation in the source code and attract developer's attention to the most problematic code areas.

Properties clustering

- Clustering can effectively summarize displayed properties and identify a similar performance behaviour possibly hidden in the large amount of data


Periscope - BT/y_solve.f - Eclipse
File Edit Refactor Navigate Search Project Run Window Help
Periscope
Welcome
y_solve.f
Periscope SIR Outli
Project Explorer
Periscope Properties View
Clustering Results View
Console

```
c---------------------------------------------------------------------
c         call lhsy(c)
c---------------------------------------------------------------------
c     wait for completion
c---------------------------------------------------------------------
            call mpi_wait(send_id,r_status,error)
            call mpi_wait(recv_id,r_status,error)
c---------------------------------------------------------------------
c     install C'(jstart+1) and rhs'(jstart+1) to be used in this cell
c---------------------------------------------------------------------
            call y_unpack_solve_info(c)
            call y_solve_cell(first,last,c)
         endif

         if (last .eq. 0) call y_send_solve_info(send_id,c)
      enddo

c---------------------------------------------------------------------
c     now perform backsubstitution in reverse direction
```

```
subroutine: X_UNPACK_SOLVE_INFO:114 (0/0)
subroutine: X_SEND_SOLVE_INFO:150 (0/0)
    call: MPI_ISEND:194 (0/0)
subroutine: X_SEND_BACKSUB_INFO:205 (0/0)
    call: MPI_ISEND:236 (0/0)
subroutine: X_UNPACK_BACKSUB_INFO:247 (0
subroutine: X_RECEIVE_BACKSUB_INFO:275 (0
    call: MPI_IRECV:291 (0/0)
subroutine: X_RECEIVE_SOLVE_INFO:302 (0/0)
    call: MPI_IRECV:319 (0/0)
subroutine: X_BACKSUBSTITUTE:330 (0/0)
subroutine: X_SOLVE_CELL:391 (0/0)
subroutine: Y_SOLVE:4 (0/17)
    call: MPI_WAIT:69 (9/9)
```

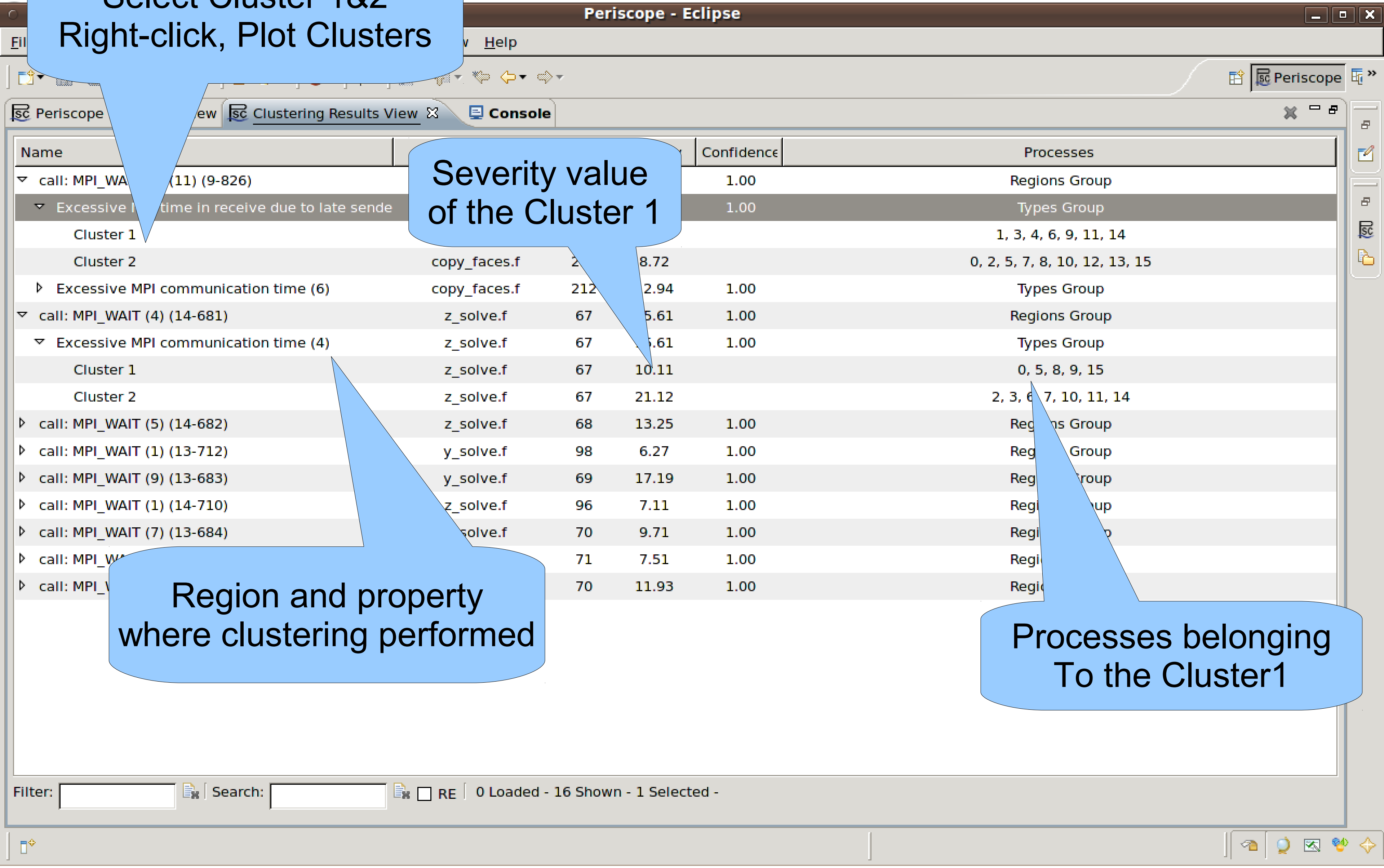
Name	Filename	RFL	Severity	Regi	
Excessive MPI communication time	x_solve.f	70	11.19	CALL_RE	
Excessive MPI communication time	x_solve.f	70	13.06	CALL_RE	
Excessive MPI communication time	y_solve.f	69	11.36		
Excessive MPI communication time	y_solve.f	69	17.10		
Excessive MPI communication time	copy_faces.f	212	17.31	CALL_REGION	
Excessive MPI communication time	copy_faces.f	212	11.26	CALL_REGION	0, 2, 5, 8, 12, 13, 15

Filter: Search: ☐ RE 0 Loaded - 47 Shown - 1 Selected -

Writable Insert 70 : 1

Right-click->
Cluster properties using FCM

Properties clustering



Name				Confidence	Processes
call: MPI_WA (11) (9-826)				1.00	Regions Group
Excessive I time in receive due to late sende				1.00	Types Group
Cluster 1					1, 3, 4, 6, 9, 11, 14
Cluster 2	copy_faces.f	2	8.72		0, 2, 5, 7, 8, 10, 12, 13, 15
Excessive MPI communication time (6)	copy_faces.f	212	2.94	1.00	Types Group
call: MPI_WAIT (4) (14-681)	z_solve.f	67	5.61	1.00	Regions Group
Excessive MPI communication time (4)	z_solve.f	67	.61	1.00	Types Group
Cluster 1	z_solve.f	67	10.11		0, 5, 8, 9, 15
Cluster 2	z_solve.f	67	21.12		2, 3, 6 7, 10, 11, 14
call: MPI_WAIT (5) (14-682)	z_solve.f	68	13.25	1.00	Reg ns Group
call: MPI_WAIT (1) (13-712)	y_solve.f	98	6.27	1.00	Reg Group
call: MPI_WAIT (9) (13-683)	y_solve.f	69	17.19	1.00	Reg roup
call: MPI_WAIT (1) (14-710)	z_solve.f	96	7.11	1.00	Regi up
call: MPI_WAIT (7) (13-684)	solve.f	70	9.71	1.00	Regi
call: MPI_W		71	7.51	1.00	Regi
call: MPI_		70	11.93	1.00	Regi

Filter: Search: □ RE 0 Loaded - 16 Shown - 1 Selected -

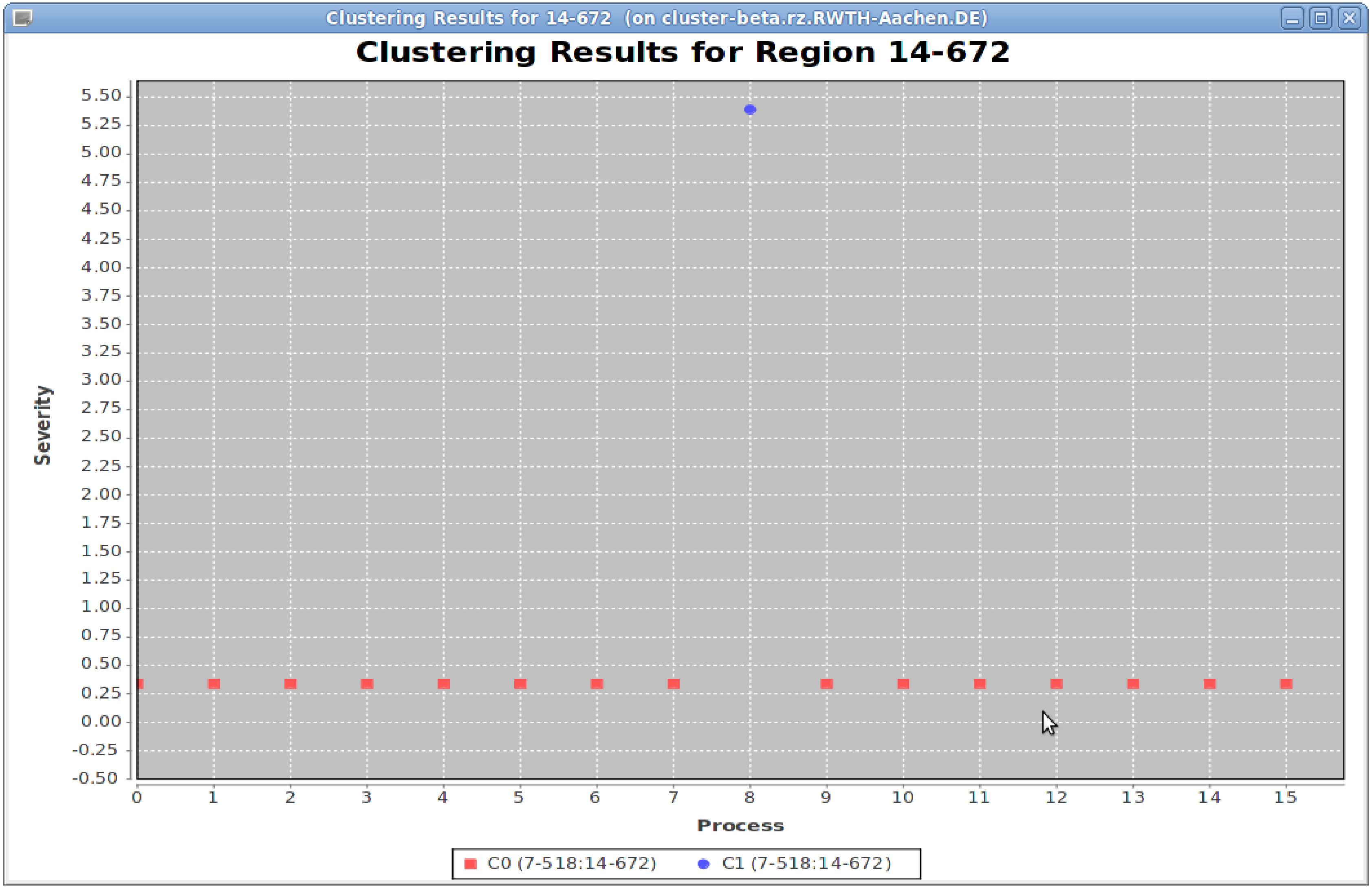
Select Cluster 1&2
Right-click, Plot Clusters

Severity value
of the Cluster 1

Region and property
where clustering performed

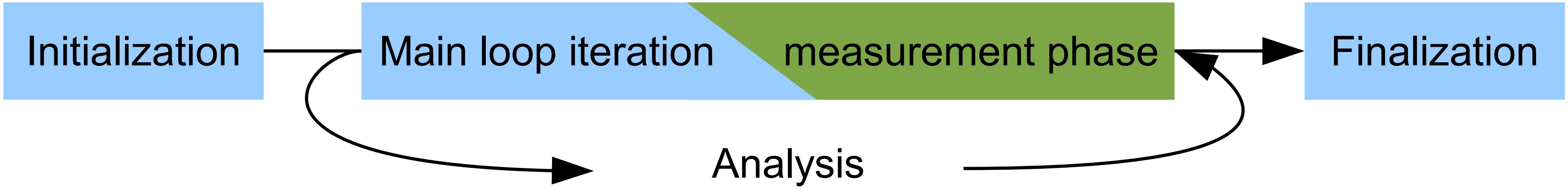
Processes belonging
To the Cluster1

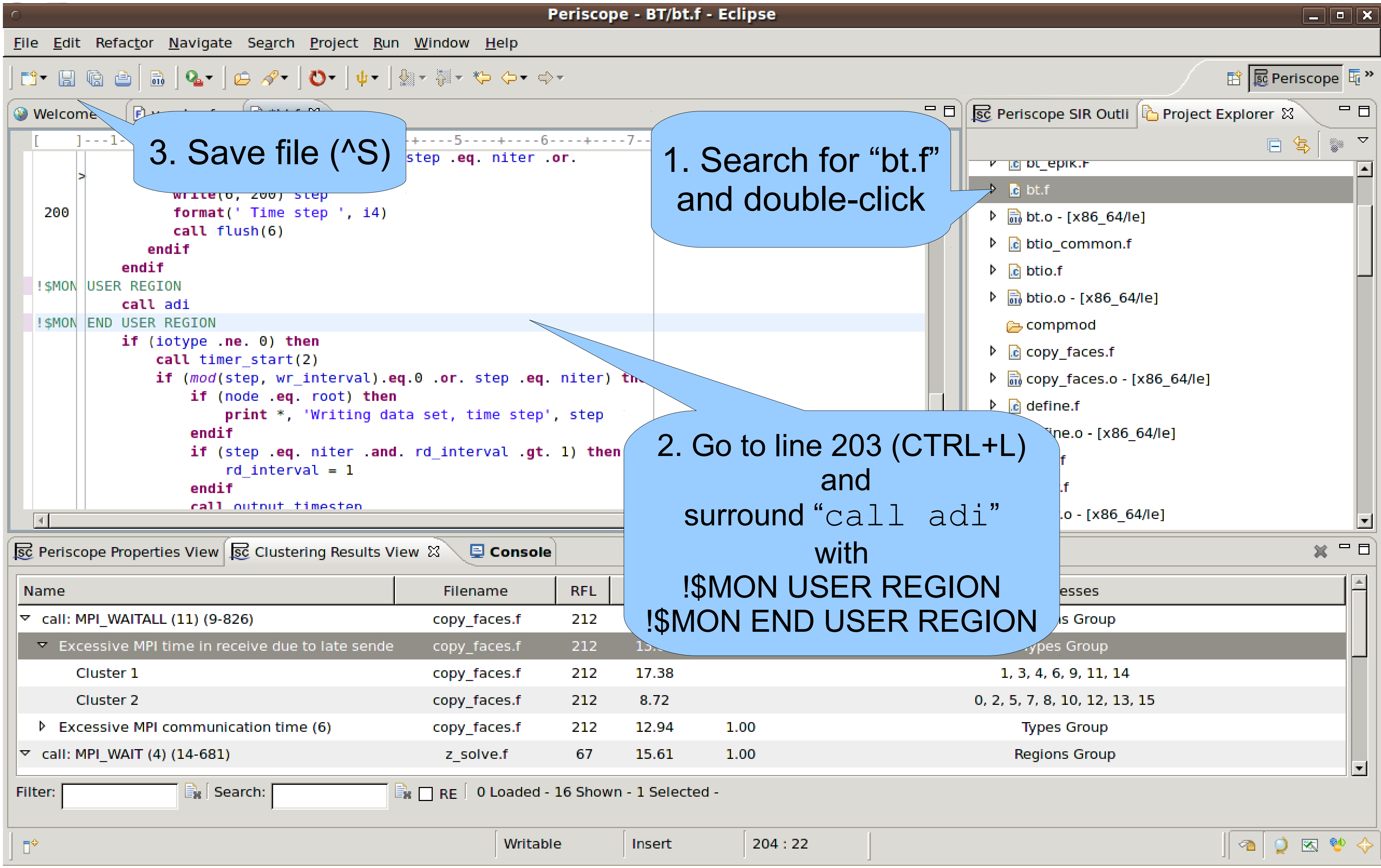
Plotting clusters



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 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





Periscope - BT/bt.f - Eclipse

1. Search for “bt.f” and double-click

2. Go to line 203 (CTRL+L) and surround “call adi” with !$MON USER REGION !$MON END USER REGION

3. Save file (^S)

```
                                                    step .eq. niter .or.
      >
                     write(6, 200) step
 200                 format(' Time step ', i4)
                     call flush(6)
                  endif
               endif
!$MON USER REGION
               call adi
!$MON END USER REGION
               if (iotype .ne. 0) then
                   call timer_start(2)
                   if (mod(step, wr_interval).eq.0 .or. step .eq. niter) th
                       if (node .eq. root) then
                           print *, 'Writing data set, time step', step
                       endif
                       if (step .eq. niter .and. rd_interval .gt. 1) then
                           rd_interval = 1
                       endif
                       call output_timestep
```

Name	Filename	RFL			esses
call: MPI_WAITALL (11) (9-826)	copy_faces.f	212			s Group
Excessive MPI time in receive due to late sende	copy_faces.f	212	13.		ypes Group
Cluster 1	copy_faces.f	212	17.38		1, 3, 4, 6, 9, 11, 14
Cluster 2	copy_faces.f	212	8.72		0, 2, 5, 7, 8, 10, 12, 13, 15
Excessive MPI communication time (6)	copy_faces.f	212	12.94	1.00	Types Group
call: MPI_WAIT (4) (14-681)	z_solve.f	67	15.61	1.00	Regions Group

Filter: Search: □ RE 0 Loaded - 16 Shown - 1 Selected -

Writable Insert 204 : 22

Repeating Periscope analysis

- Return to root directory and clean-up

```
% make clean
```

- Re-build BT with the original command

```
% make bt CLASS=B NPROCS=16
```

- Change directory into location of executable

```
% cd bin.periscope
```

Re-running Periscope

- Re-run Periscope analysis by submitting the script

```
% bsub < psc.lsf   or    msub psc.msub
Check the job output:
% bpeek
[psc_frontend][DBG0:fe] Agent network UP and RUNNING. Starting search.
 NAS Parallel Benchmarks 3.3 -- BT Benchmark
 [...]
 Time step 1
 BT Benchmark Completed.
----------
End Periscope run! Search took 37.2 seconds (33.3 seconds for startup)
```

- Only 1 iteration of BT required instead of 200 previous run!
- Frontend will overwrite the properties found into the file `properties_MPI_<PID>.psc` in the current directory, which again need to be copied into the BT source directory

```
% cp properties_MPI_*.psc ../BT
```

- Re-load `properties_MPI_<PID>.psc` in Periscope GUI. Now found properties should have more precise severities values