Periscope
Tutorial Exercise
NPB-MPI/BT

M. Gerndt, V. Petkov, Y. Oleynik, R. Mijakovic

Technische Universität München

periscope@in.tum.de

April 2012

NPB-BT-MZ Exercise

- Intermediate-level tutorial example

- Available in MPI, OpenMP, hybrid OpenMP/MPI variants

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

- Manual instrumentation optimization

Exercise steps

0. Loading and configuring of Periscope

```
module load 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          = curie50          //hostname
SITE             = UVSQ
REGSERVICE_HOST = curie50          //host of registry
REGSERVICE_PORT = 50001            //port of the registry
APPL_BASEPORT    = 51000            //first port for application
AGENT_BASEPORT   = 50002            //first port agent hierarchy
```

Instrumenting application

- The Periscope agents and the application processes register with a `registry`. The registry is started 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
```

Instrumenting NPB-MPI-BT

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

```
MPIF77 = psc_instrument -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-MZ instrumentation and build

- Return to root directory and clean-up

```
% make clean
```

- Re-build BT with the original command

```
% make bt-mz CLASS=B NPROCS=4
   ==========================================
   =      NAS Parallel Benchmarks 3.3      =
   =      MPI/F77/C                        =
   ==========================================
cd BT-MZ; make NPROCS=4 CLASS=B SUBTYPE= VERSION=
make[1]: Entering directory `BT-MZ'
...
psc_instrument -s bt.sir -t "user loop call" mpif77 -c  -O -g bt.f
psc_instrument -s bt.sir -t "user loop call" mpif77 -c  -O -g make_set.f
…
psc_instrument -s bt.sir -t "user loop call" mpif77 -O \
-o ../bin.periscope/bt-mz_B.4 bt.o ...
Built executable ../bin.periscope/bt-mz_B.4
make[1]: Leaving directory `BT-MZ'
```

- 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

```
% psc_frontend --sir=bt-mz_B.4.sir --apprun=./bt-mz_B.4 --strategy=MPI
--mpinumprocs=4
[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 60.5 seconds (33.3 seconds for startup)
```

Running Periscope in Batch

- Copy and change the Periscope batch script

```
#!/bin/bash
# submit from ./bin.periscope directory with "ccc_msub psc.msub"
#MSUB -r npb_btmz_psc
#MSUB -o npb_btmz_%I.oe
#MSUB -e npb_btmz_%I.oe
#MSUB -n 4      # number of MPI processes
#MSUB -c 8      # number of OpenMP threads/process
#MSUB -T 600    # max walltime in seconds
#MSUB -x        # allocate exclusive nodes
#MSUB -A tgcc0007 # project id

cd $BRIDGE_MSUB_PWD

# benchmark configuration
export OMP_NUM_THREADS=$BRIDGE_MSUB_NCORE
PROCS=$BRIDGE_MSUB_NPROC
CLASS=B
EXE=./bt-mz_$CLASS.$PROCS

# remember to check that psc_regsrv is running!
psc_frontend --apprun=$EXE --mpinumprocs=$PROCS --strategy=MPI
```

- Submit the script with `ccc_msub psc.msub`

Running Periscope GUI from LiveDVD

- When running Eclipse *from* LiveDVD copy the .psc file to your local tutorial folder:

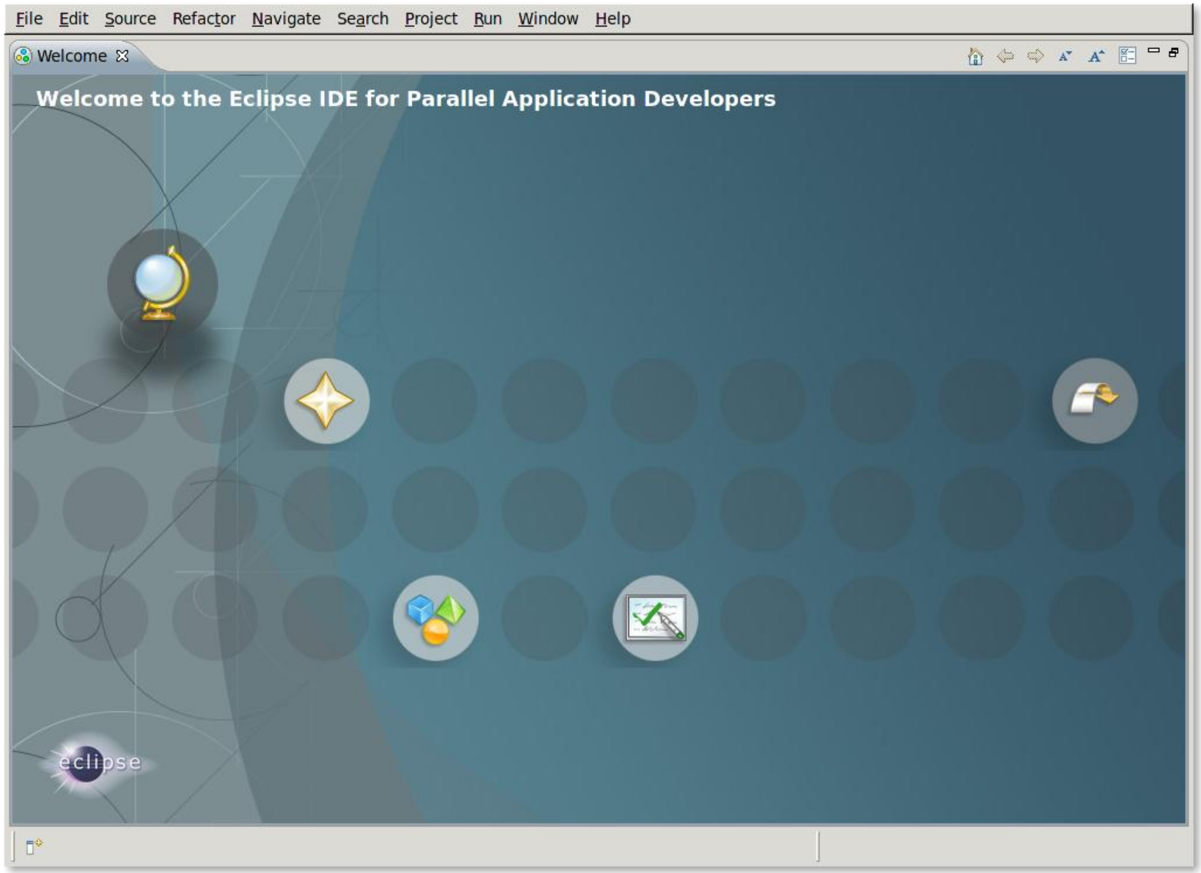
```
%scp <username>@curie.ccc.cea.fr:tutorial/NPB3.3-MZ-MPI/bin.periscope/*.psc
tutorial/NPB3.3-MZ-MPI/bin.periscope

%scp <username>@curie.ccc.cea.fr:tutorial/NPB3.3-MZ-MPI/bin.periscope/*.sir
tutorial/NPB3.3-MZ-MPI/bin.periscope
```

Starting Periscope GUI

- Start `Eclipse` with Periscope GUI from console

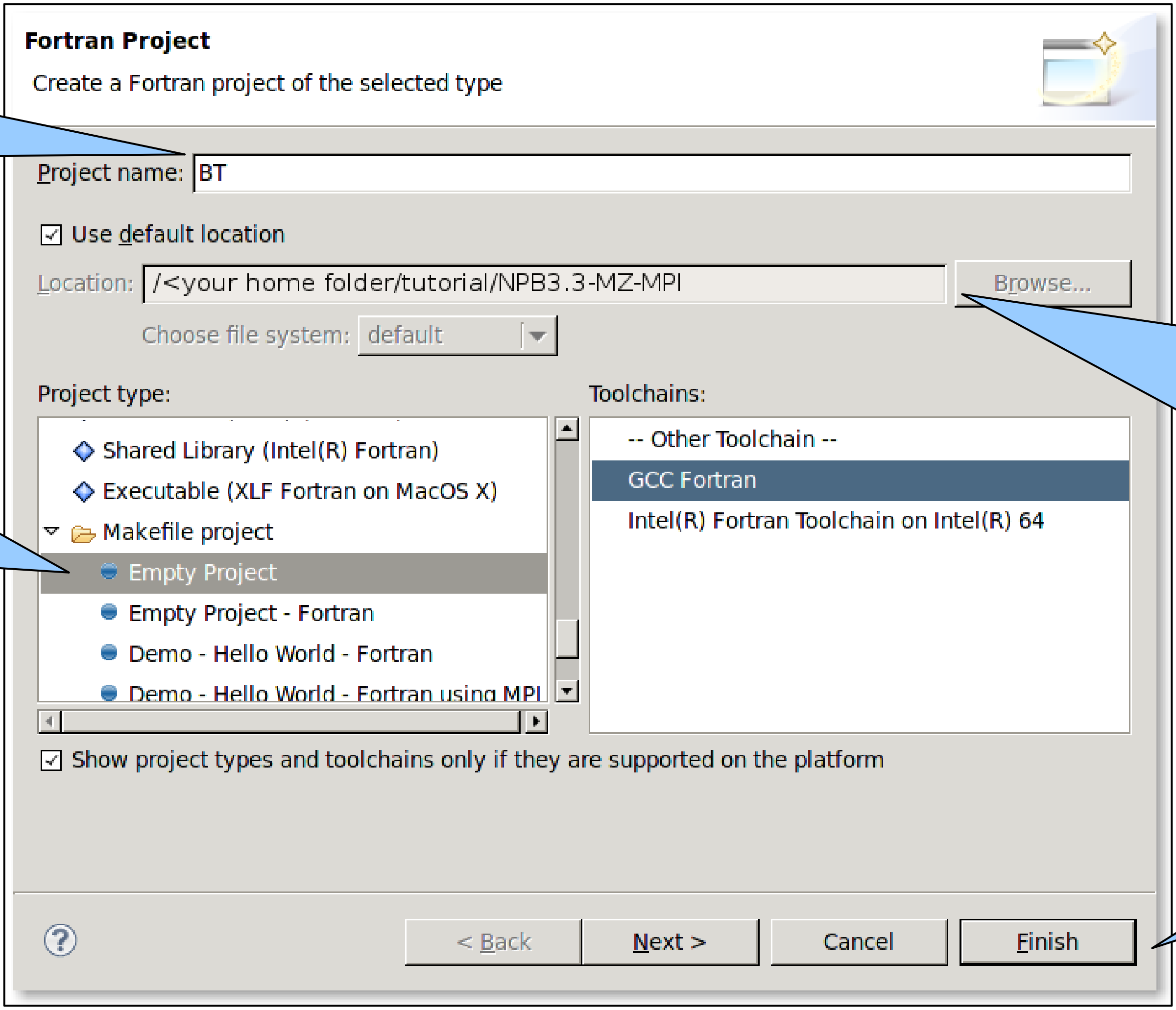
```
% eclipse
```

- 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: /<your home folder/tutorial/NPB3.3-MZ-MPI

Choose file system: default

Project type:

- Shared Library (Intel(R) Fortran)
- Executable (XLF Fortran on MacOS X)
- Makefile project
  - Empty Project
  - Empty Project - Fortran
  - Demo - Hello World - Fortran
  - Demo - Hello World - Fortran using MPI

Toolchains:

- -- Other Toolchain --
- GCC Fortran
- Intel(R) Fortran Toolchain on Intel(R) 64

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

Input project name

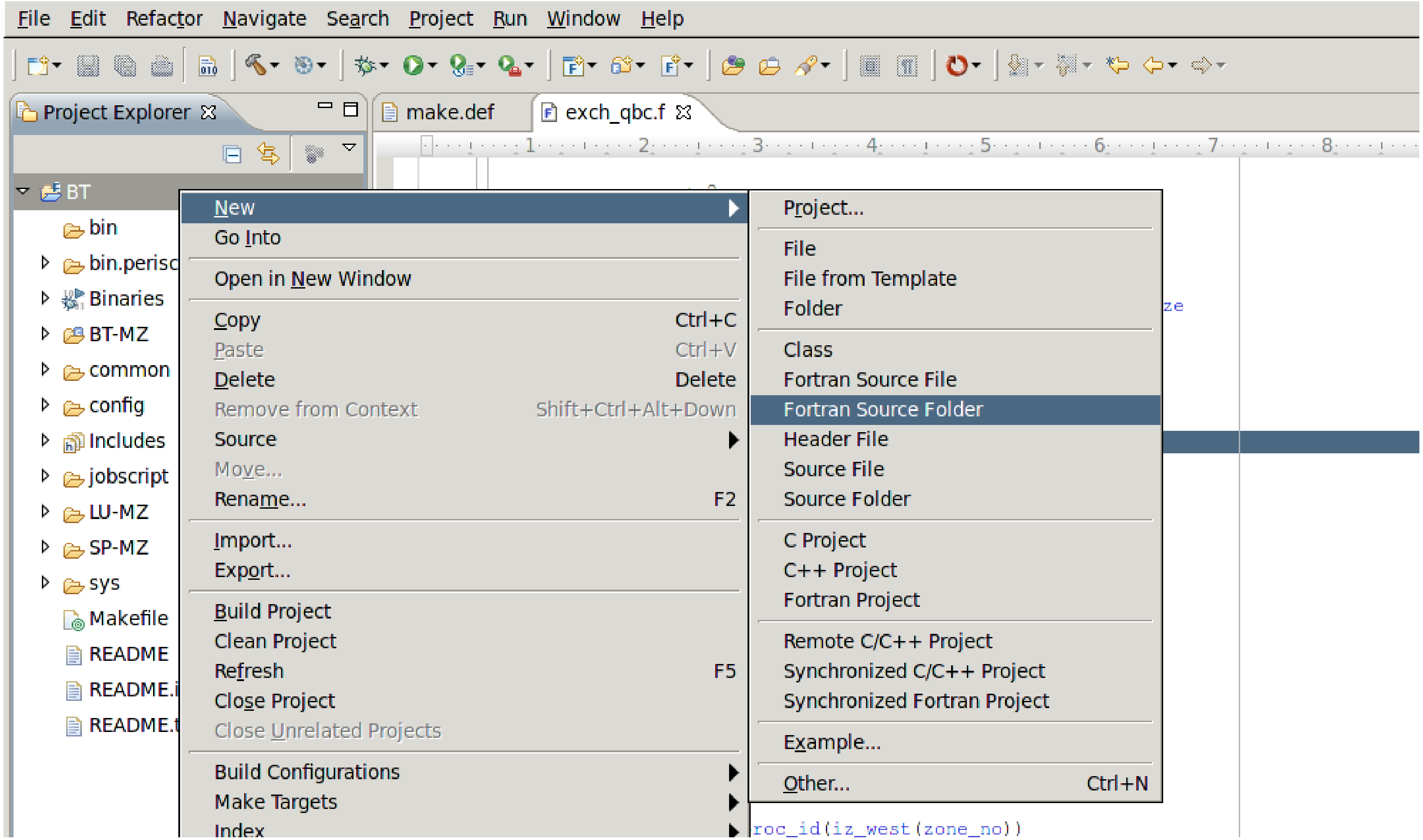
Project type

Unmark "Use default location" and provide path to *BT* folder

Press Finish

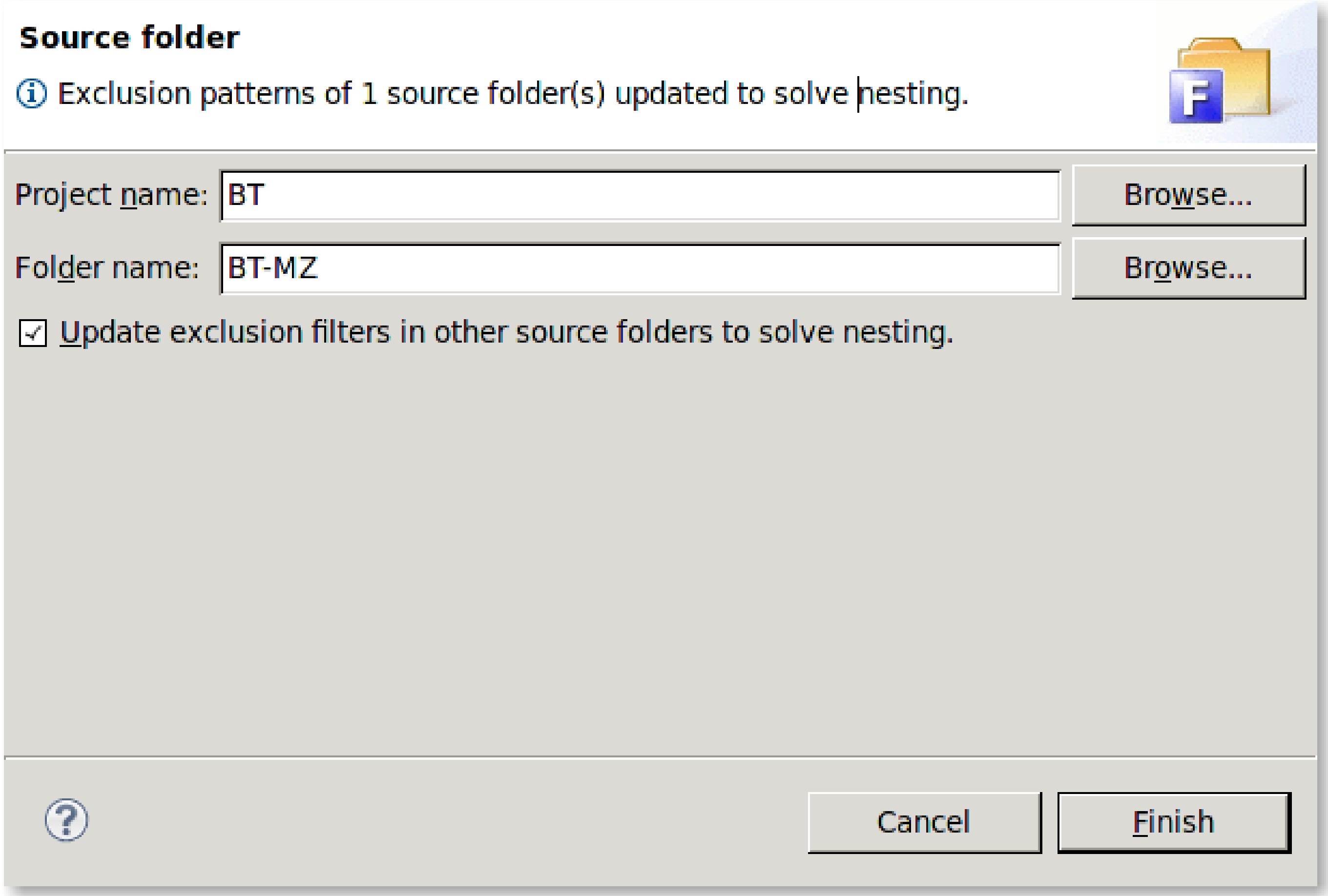
Add BT-MZ as a Source Folder

- Right-click -> File-> New -> Fortran Source Folder



Add BT-MZ as a Source Folder

- Choose BT-MZ as a source folder



Source folder

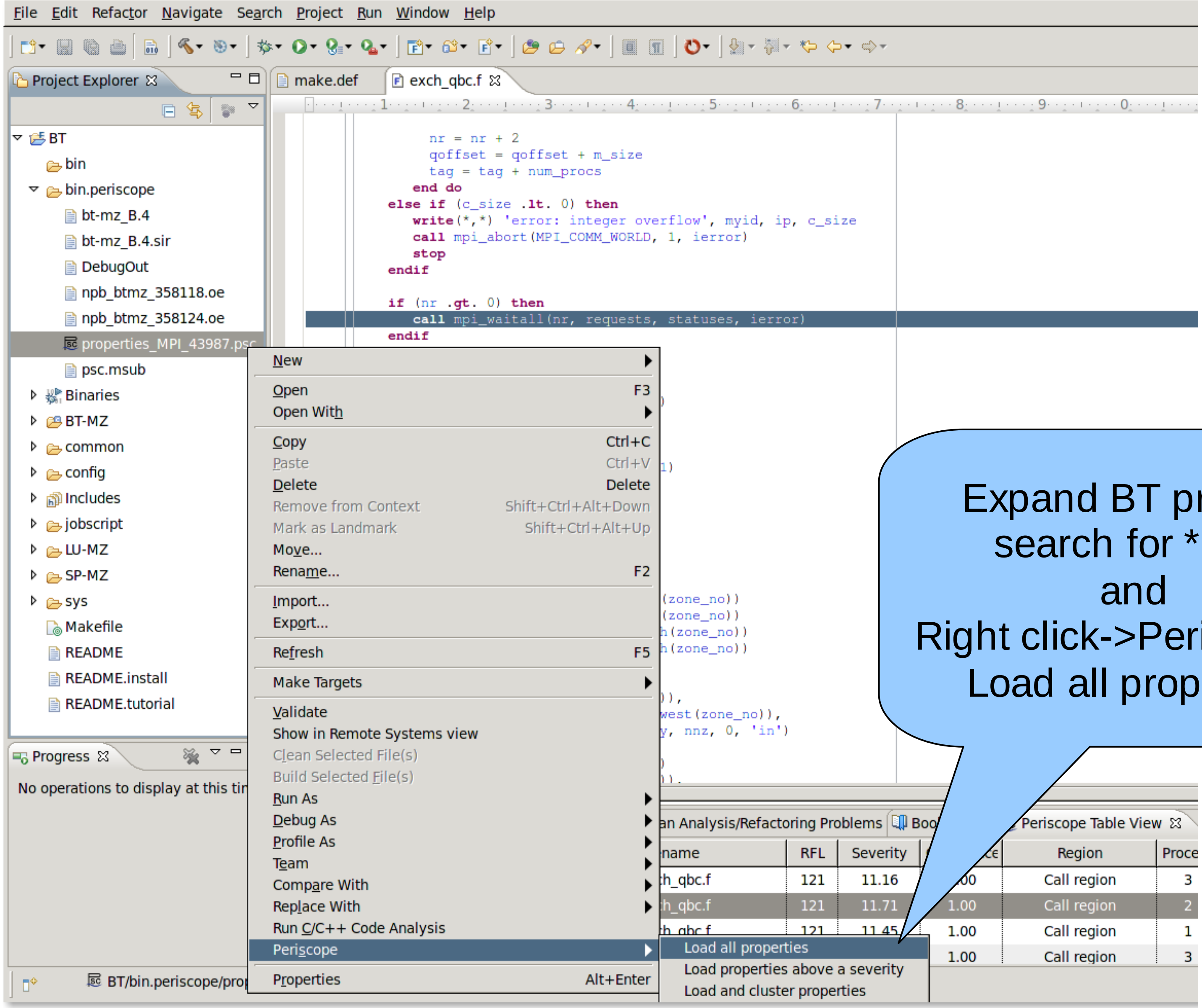
ⓘ Exclusion patterns of 1 source folder(s) updated to solve nesting.

Project name: BT

Folder name: BT-MZ

☑ Update exclusion filters in other source folders to solve nesting.

Loading properties



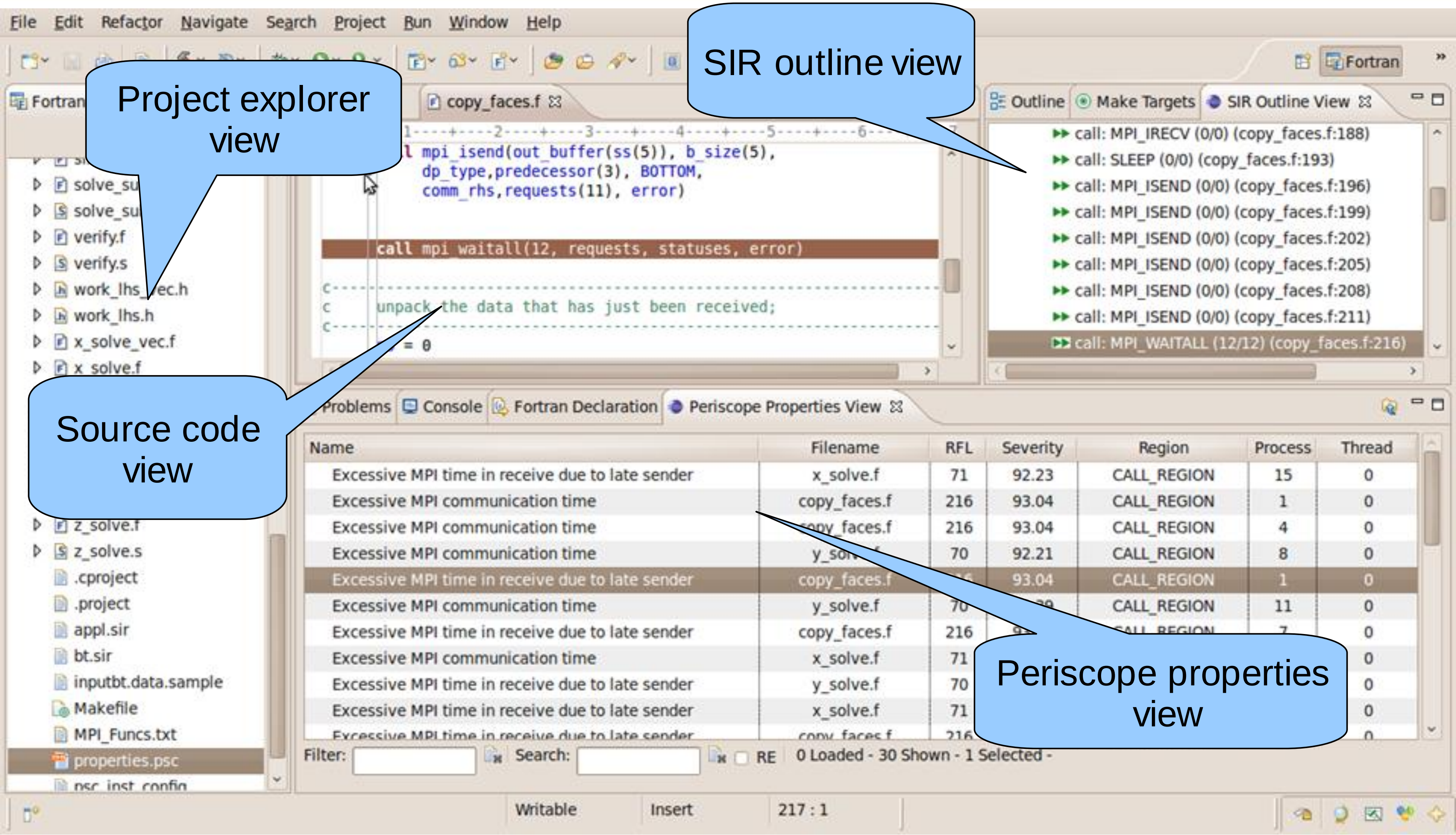
```
          nr = nr + 2
          qoffset = qoffset + m_size
          tag = tag + num_procs
       end do
    else if (c_size .lt. 0) then
       write(*,*) 'error: integer overflow', myid, ip, c_size
       call mpi_abort(MPI_COMM_WORLD, 1, ierror)
       stop
    endif

    if (nr .gt. 0) then
       call mpi_waitall(nr, requests, statuses, ierror)
    endif
```

```
)
1)
(zone_no))
(zone_no))
h(zone_no))
h(zone_no))
)),
west(zone_no)),
y, nnz, 0, 'in')
)
)).
```

name	RFL	Severity	ce	Region	Proce
th_qbc.f	121	11.16	00	Call region	3
h_qbc.f	121	11.71	1.00	Call region	2
h_qbc.f	121	11.45	1.00	Call region	1
			1.00	Call region	3

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

Periscope GUI



Project explorer view

- si…
- solve_su…
- solve_su…
- verify.f
- verify.s
- work_lhs_vec.h
- work_lhs.h
- x_solve_vec.f
- x_solve.f
- z_solve.f
- z_solve.s
- .cproject
- .project
- appl.sir
- bt.sir
- inputbt.data.sample
- Makefile
- MPI_Funcs.txt
- properties.psc
- psc_inst_config

Source code view

```
    1----+----2----+----3----+----4----+----5----+----6---
l mpi_isend(out_buffer(ss(5)), b_size(5),
   dp_type,predecessor(3), BOTTOM,
   comm_rhs,requests(11), error)

      call mpi_waitall(12, requests, statuses, error)

c---------------------------------------------------------------------
c     unpack the data that has just been received;
c---------------------------------------------------------------------
       = 0
```

SIR outline view

```
▶▶ call: MPI_IRECV (0/0) (copy_faces.f:188)
▶▶ call: SLEEP (0/0) (copy_faces.f:193)
▶▶ call: MPI_ISEND (0/0) (copy_faces.f:196)
▶▶ call: MPI_ISEND (0/0) (copy_faces.f:199)
▶▶ call: MPI_ISEND (0/0) (copy_faces.f:202)
▶▶ call: MPI_ISEND (0/0) (copy_faces.f:205)
▶▶ call: MPI_ISEND (0/0) (copy_faces.f:208)
▶▶ call: MPI_ISEND (0/0) (copy_faces.f:211)
▶▶ call: MPI_WAITALL (12/12) (copy_faces.f:216)
```

Periscope properties view

Name	Filename	RFL	Severity	Region	Process	Thread
Excessive MPI time in receive due to late sender	x_solve.f	71	92.23	CALL_REGION	15	0
Excessive MPI communication time	copy_faces.f	216	93.04	CALL_REGION	1	0
Excessive MPI communication time	copy_faces.f	216	93.04	CALL_REGION	4	0
Excessive MPI communication time	y_solve.f	70	92.21	CALL_REGION	8	0
Excessive MPI time in receive due to late sender	copy_faces.f	216	93.04	CALL_REGION	1	0
Excessive MPI communication time	y_solve.f	70	[illegible]	CALL_REGION	11	0
Excessive MPI time in receive due to late sender	copy_faces.f	216	[illegible]	CALL_REGION	7	0
Excessive MPI communication time	x_solve.f	71	[illegible]	[illegible]	[illegible]	0
Excessive MPI time in receive due to late sender	y_solve.f	70	[illegible]	[illegible]	[illegible]	0
Excessive MPI time in receive due to late sender	x_solve.f	71	[illegible]	[illegible]	[illegible]	0
Excessive MPI time in receive due to late sender	copy_faces.f	216	[illegible]	[illegible]	[illegible]	0

Filter: Search: □ RE 0 Loaded - 30 Shown - 1 Selected -

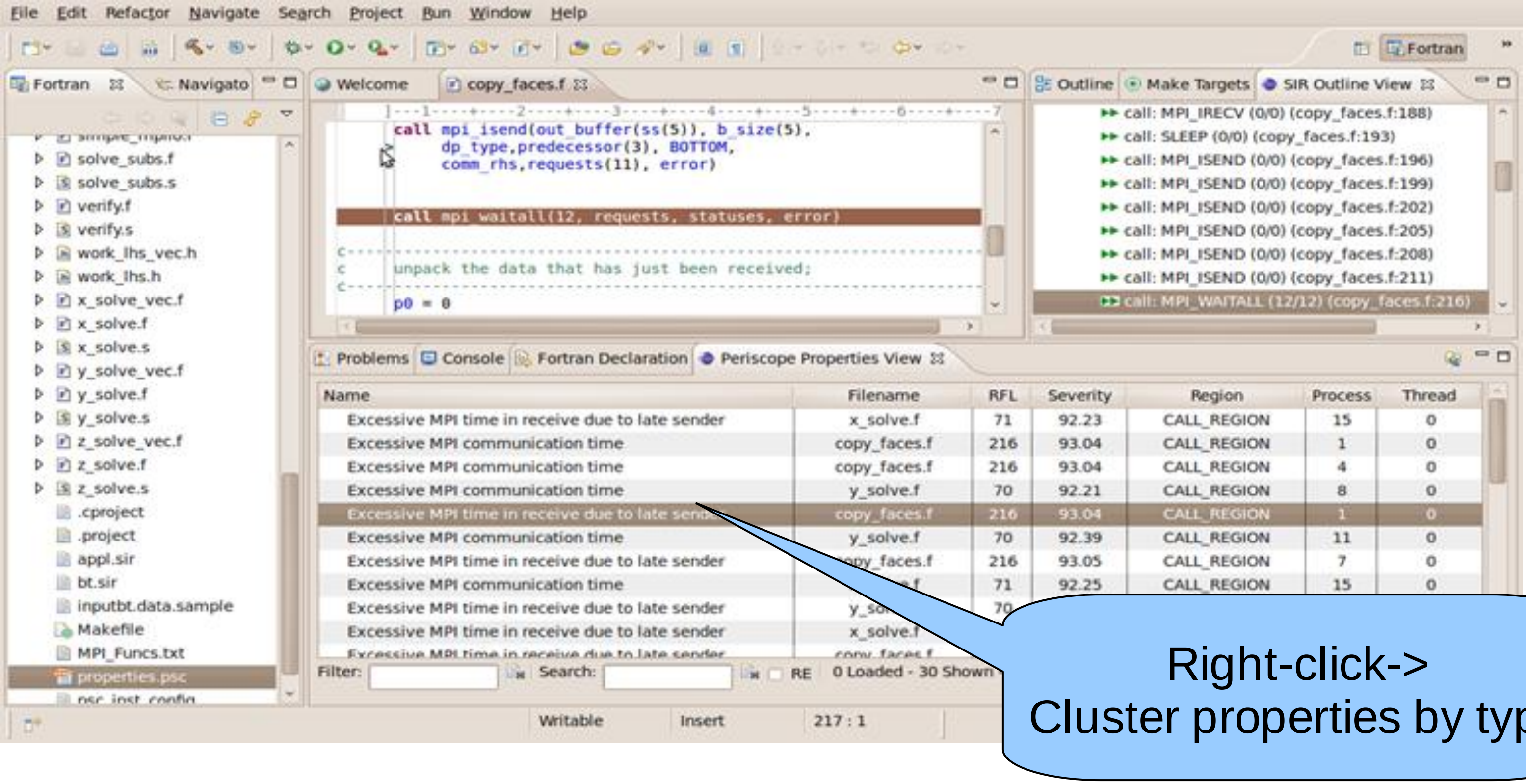
Periscope GUI report exploration features

- 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



```
|---1----+----2----+----3----+----4----+----5----+----6----+----7
      call mpi_isend(out_buffer(ss(5)), b_size(5),
     >     dp_type,predecessor(3), BOTTOM,
     >     comm_rhs,requests(11), error)


      call mpi_waitall(12, requests, statuses, error)

c---------------------------------------------------------------------
c     unpack the data that has just been received;
c---------------------------------------------------------------------
      p0 = 0
```

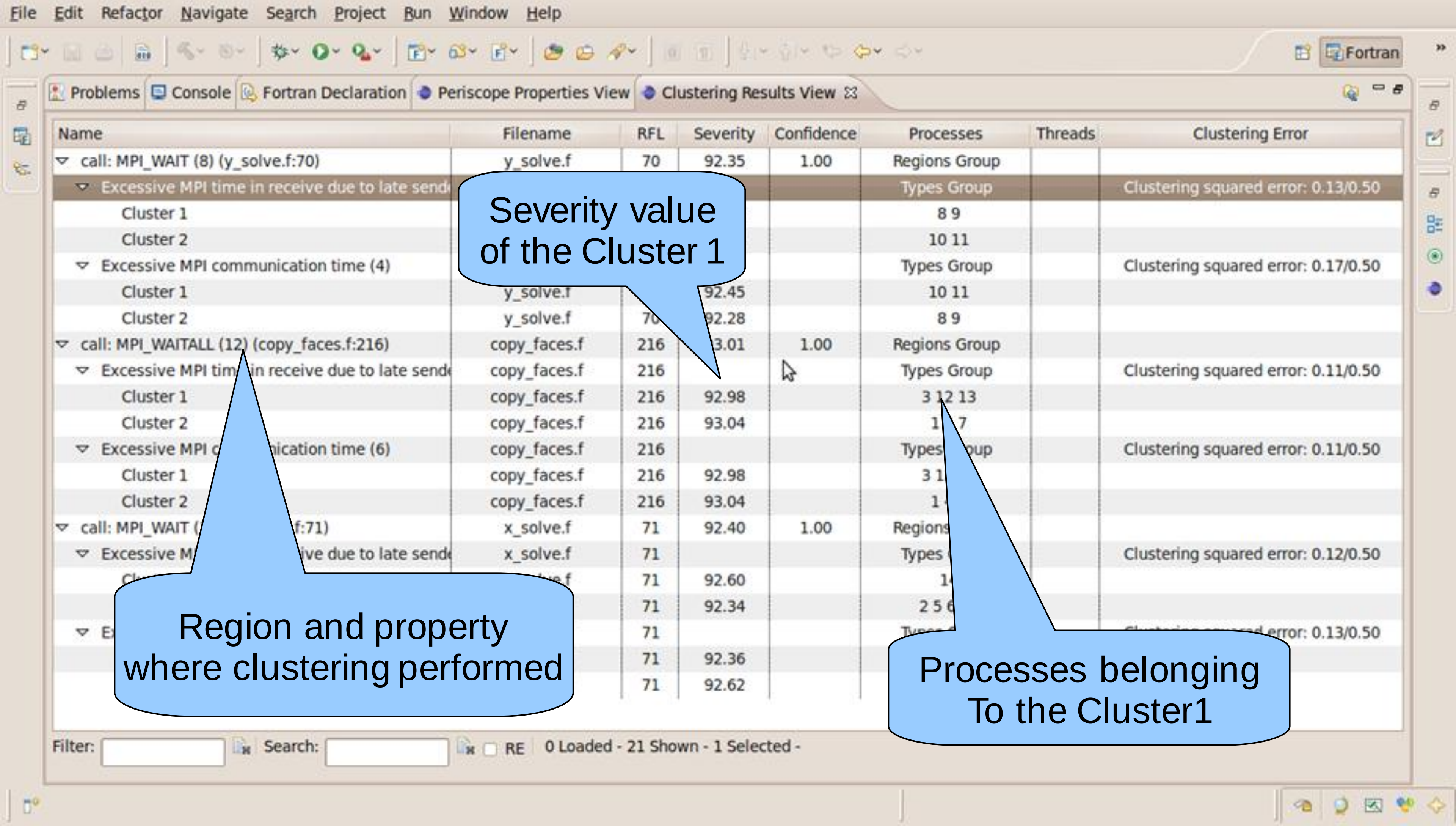
```
▶▶ call: MPI_IRECV (0/0) (copy_faces.f:188)
▶▶ call: SLEEP (0/0) (copy_faces.f:193)
▶▶ call: MPI_ISEND (0/0) (copy_faces.f:196)
▶▶ call: MPI_ISEND (0/0) (copy_faces.f:199)
▶▶ call: MPI_ISEND (0/0) (copy_faces.f:202)
▶▶ call: MPI_ISEND (0/0) (copy_faces.f:205)
▶▶ call: MPI_ISEND (0/0) (copy_faces.f:208)
▶▶ call: MPI_ISEND (0/0) (copy_faces.f:211)
▶▶ call: MPI_WAITALL (12/12) (copy_faces.f:216)
```

Name	Filename	RFL	Severity	Region	Process	Thread
Excessive MPI time in receive due to late sender	x_solve.f	71	92.23	CALL_REGION	15	0
Excessive MPI communication time	copy_faces.f	216	93.04	CALL_REGION	1	0
Excessive MPI communication time	copy_faces.f	216	93.04	CALL_REGION	4	0
Excessive MPI communication time	y_solve.f	70	92.21	CALL_REGION	8	0
Excessive MPI time in receive due to late sender	copy_faces.f	216	93.04	CALL_REGION	1	0
Excessive MPI communication time	y_solve.f	70	92.39	CALL_REGION	11	0
Excessive MPI time in receive due to late sender	copy_faces.f	216	93.05	CALL_REGION	7	0
Excessive MPI communication time	[illegible]	71	92.25	CALL_REGION	15	0
Excessive MPI time in receive due to late sender	y_so[illegible]	70	[illegible]	[illegible]	[illegible]	[illegible]
Excessive MPI time in receive due to late sender	x_solve.f	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Excessive MPI time in receive due to late sender	copy_faces.f	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Filter: Search: □ RE 0 Loaded - 30 Shown

Right-click->
Cluster properties by type

Properties clustering


File Edit Refactor Navigate Search Project Run Window Help
Fortran
Problems
Console
Fortran Declaration
Periscope Properties View
Clustering Results View

Name	Filename	RFL	Severity	Confidence	Processes	Threads	Clustering Error
call: MPI_WAIT (8) (y_solve.f:70)	y_solve.f	70	92.35	1.00	Regions Group		
Excessive MPI time in receive due to late send					Types Group		Clustering squared error: 0.13/0.50
Cluster 1					8 9		
Cluster 2					10 11		
Excessive MPI communication time (4)					Types Group		Clustering squared error: 0.17/0.50
Cluster 1	y_solve.f		92.45		10 11		
Cluster 2	y_solve.f	70	92.28		8 9		
call: MPI_WAITALL (12) (copy_faces.f:216)	copy_faces.f	216	3.01	1.00	Regions Group		
Excessive MPI tim in receive due to late send	copy_faces.f	216			Types Group		Clustering squared error: 0.11/0.50
Cluster 1	copy_faces.f	216	92.98		3 12 13		
Cluster 2	copy_faces.f	216	93.04		1 7		
Excessive MPI c nication time (6)	copy_faces.f	216			Types oup		Clustering squared error: 0.11/0.50
Cluster 1	copy_faces.f	216	92.98		3 1		
Cluster 2	copy_faces.f	216	93.04		1		
call: MPI_WAIT (f:71)	x_solve.f	71	92.40	1.00	Regions		
Excessive M ive due to late send	x_solve.f	71			Types		Clustering squared error: 0.12/0.50
		71	92.60		1		
		71	92.34		2 5 6		
		71					error: 0.13/0.50
		71	92.36				
		71	92.62				

Filter: Search: □ RE 0 Loaded - 21 Shown - 1 Selected -

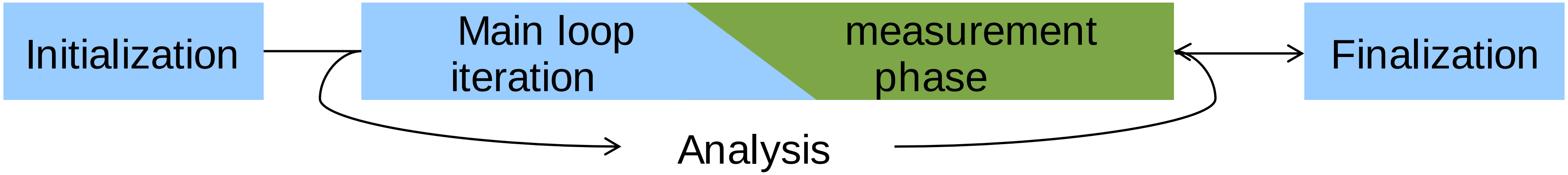
Severity value of the Cluster 1

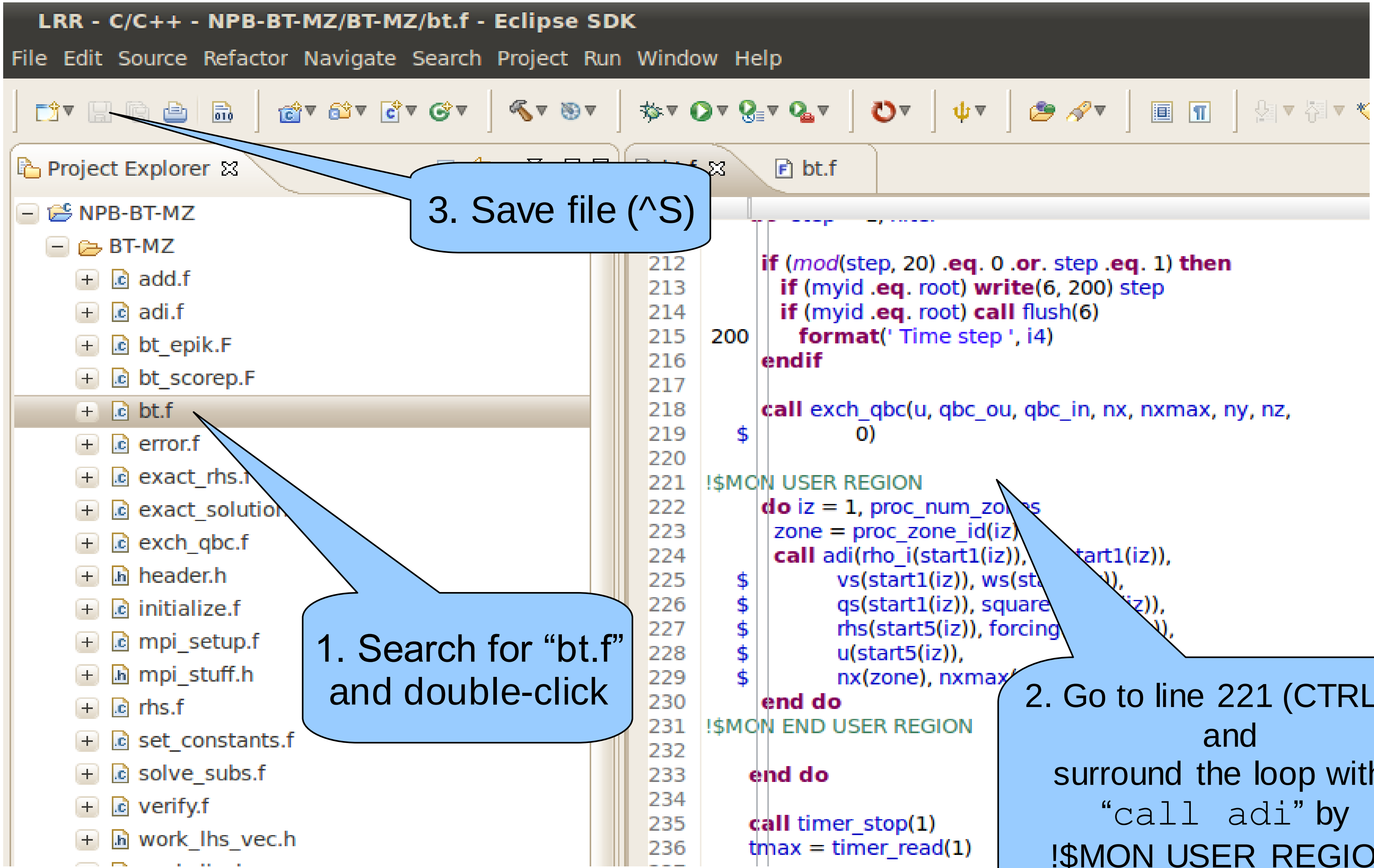
Region and property where clustering performed

Processes belonging To the Cluster1

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





```
212        if (mod(step, 20) .eq. 0 .or. step .eq. 1) then
213          if (myid .eq. root) write(6, 200) step
214          if (myid .eq. root) call flush(6)
215  200       format(' Time step ', i4)
216        endif
217
218        call exch_qbc(u, qbc_ou, qbc_in, nx, nxmax, ny, nz,
219     $              0)
220
221 !$MON USER REGION
222        do iz = 1, proc_num_zo  s
223         zone = proc_zone_id(iz)
224         call adi(rho_i(start1(iz)),      tart1(iz)),
225     $           vs(start1(iz)), ws(st         )),
226     $           qs(start1(iz)), square         iz)),
227     $           rhs(start5(iz)), forcing          )),
228     $           u(start5(iz)),
229     $           nx(zone), nxmax
230        end do
231 !$MON END USER REGION
232
233      end do
234
235      call timer_stop(1)
236      tmax = timer_read(1)
```

3. Save file (^S)

1. Search for “bt.f” and double-click

2. Go to line 221 (CTRL+L) and surround the loop with “`call adi`” by !$MON USER REGION !$MON END USER REGION

Repeating Periscope analysis

- Return to root directory and clean-up

```
% make clean
```

- Re-build BT with the original command

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

- Change directory into location of executable

```
% cd bin.periscope
```

Re-running Periscope

- Re-run Periscope analysis by executing `psc_frontend`

```
% psc_frontend --sir=bt-mz_W.4.sir --apprun=./bt-mz_W.4 --strategy=MPI
--mpinumprocs=4 --force-localhost
[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_*.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_*.psc` in Periscope GUI. Now found properties should have more precise severities values