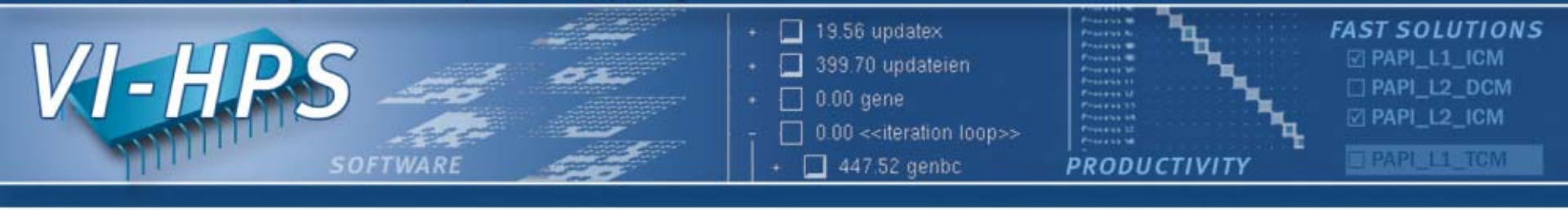


Marmot an MPI Correctness Tool: Overview

6th VI-HPS Tuning Workshop at SARA, Amsterdam
May 26th – May 28th, 2010

Andreas Knüpfer, Tobias Hilbrich

andreas.knuepfer@tu-dresden.de,
tobias.hilbrich@zih.tu-dresden.de

- Tool to check for correct MPI usage at runtime
  - Checks parameters passed to MPI
  - Checks conformance to MPI Standard
  - Monitors MPI resource usage
- A library written in C++, is linked to the application
- Does not require source code modifications
- Requires an additional process used as *DebugServer*
- Support for Fortran and C bindings of MPI-1.2
- Results are shown in a log file (different types supported)

Linking and Compilation with Marmot

- Use the Marmot compiler wrappers to compile and link:
  - Replace compiler calls by appropriate wrapper
  - For C/C++ use marmotcc or marmotcxx
  - For Fortran use marmotf77 or marmotf90
  - Source code instrumentation added automatically (usually)

Execution with Marmot

- Execution with Marmot requires one additional process (used for global view)
  - Instead of “mpirun -np n“ call “mpirun -np n+1“
  - Marmot's checks cause overhead, thus, performance may decrease

Output

- MARMOT supports 3 different log file types
  - ASCII
  - HTML
  - CUBE
- Desired log file type is selected with an environmental variable (export MARMOT_LOGFILE_TYPE=...)
- ASCII and HTML logs may become very large and are unable to give an overview
- CUBE log offers an overview along with all the detailed information

datatype.c:

```
(47) MPI_Init (&argc, &argv);
(48) MPI_Comm_rank (MPI_COMM_WORLD, &rank);
(49) MPI_Comm_size (MPI_COMM_WORLD, &size);
(50) MPI_Type_contiguous (2, MPI_INT, &cont2Int);
(51)
(52)  if (rank == 0){
(53)        MPI_Send (s_buf, 1, cont2Int, 1, MSG_TAG_1 ,MPI_COMM_WORLD);
(54)
(55) if (rank == 1){
(56)        MPI_Recv (r_buf, 1, cont2Int, 0, MSG_TAG_1,MPI_COMM_WORLD,&status);
(57)
(58) MPI_Type_free (&cont2Int);
(59) MPI_Finalize();
```

Example – Compiling, Linking, and Running

Linking and Compiling

- marmotcc datatype.c -o datatype.exe *#Compile and link*
- (On some systems you may have to load a module first)

Configuring Marmot (optional)

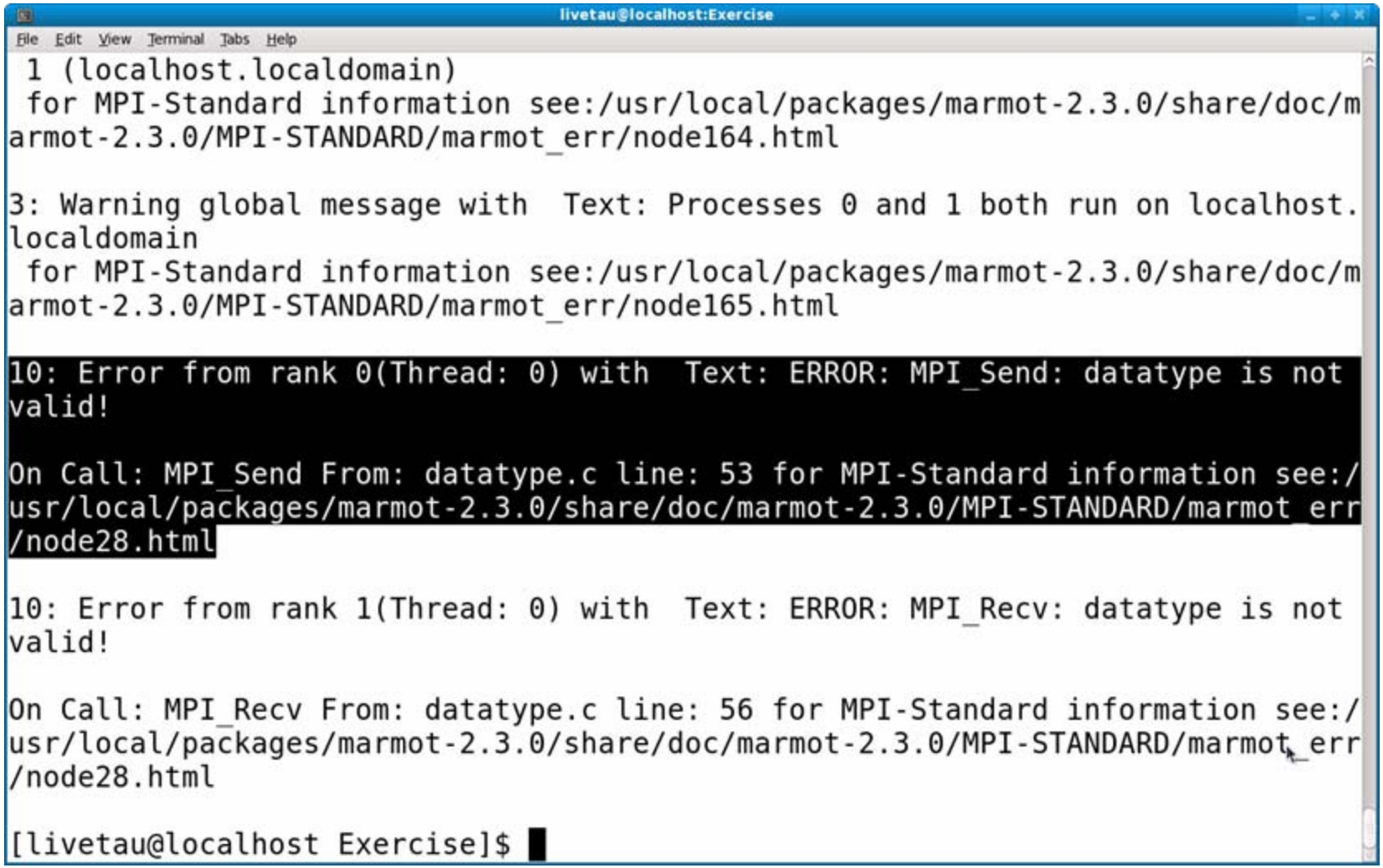
- export MARMOT_LOGFILE_TYPE=1 *#Use HTML logging*

Running

- mpirun –np 3 datatype.exe *#Execute with 2(!) app. processes*

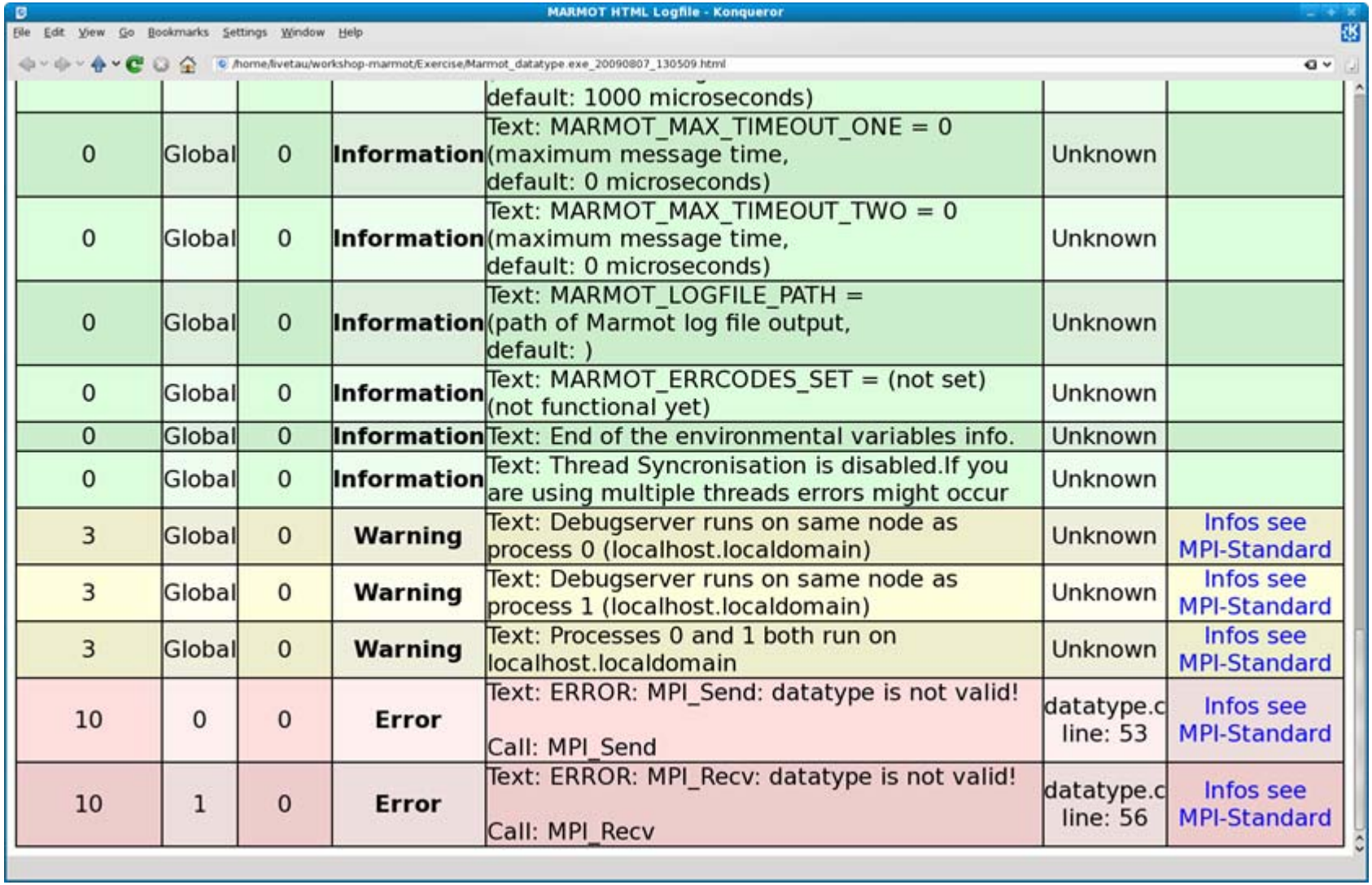
Example – Text Output

export MARMOT_LOGFILE_TYPE=0



Example – HTML Output

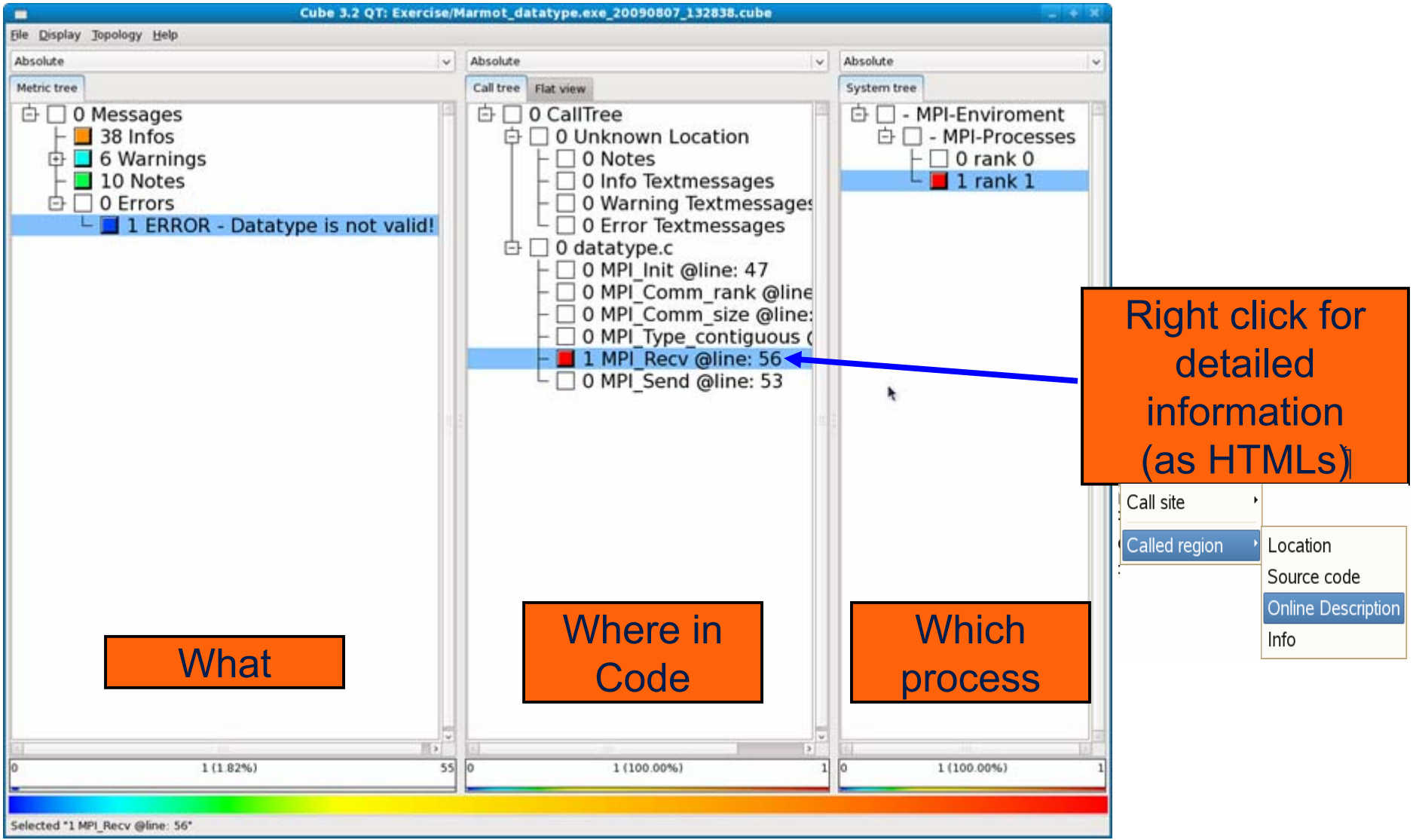
export MARMOT_LOGFILE_TYPE=1



				default: 1000 microseconds)		
0	Global	0	**Information**	Text: MARMOT_MAX_TIMEOUT_ONE = 0 (maximum message time, default: 0 microseconds)	Unknown	
0	Global	0	**Information**	Text: MARMOT_MAX_TIMEOUT_TWO = 0 (maximum message time, default: 0 microseconds)	Unknown	
0	Global	0	**Information**	Text: MARMOT_LOGFILE_PATH = (path of Marmot log file output, default:)	Unknown	
0	Global	0	**Information**	Text: MARMOT_ERRCODES_SET = (not set) (not functional yet)	Unknown	
0	Global	0	**Information**	Text: End of the environmental variables info.	Unknown	
0	Global	0	**Information**	Text: Thread Syncronisation is disabled.If you are using multiple threads errors might occur	Unknown	
3	Global	0	**Warning**	Text: Debugserver runs on same node as process 0 (localhost.localdomain)	Unknown	Infos see MPI-Standard
3	Global	0	**Warning**	Text: Debugserver runs on same node as process 1 (localhost.localdomain)	Unknown	Infos see MPI-Standard
3	Global	0	**Warning**	Text: Processes 0 and 1 both run on localhost.localdomain	Unknown	Infos see MPI-Standard
10	0	0	**Error**	Text: ERROR: MPI_Send: datatype is not valid! Call: MPI_Send	datatype.c line: 53	Infos see MPI-Standard
10	1	0	**Error**	Text: ERROR: MPI_Recv: datatype is not valid! Call: MPI_Recv	datatype.c line: 56	Infos see MPI-Standard

Example – CUBE Output

export MARMOT_LOGFILE_TYPE=2



Marmot an MPI Correctness Tool: Hands-on Exercise

6th VI-HPS Tuning Workshop at SARA, Amsterdam
May 26th – May 28th, 2010

Andreas Knüpfer, Tobias Hilbrich

andreas.knuepfer@tu-dresden.de,
tobias.hilbrich@zih.tu-dresden.de

BT Tutorial exercise

- The following tutorial steps will be similar for all the tools
- Use the provided NPB-3.3-MPI tutorial directory
  - Previously used to build and run normal version of BT

```
% cd workshop-vihps/NPB3.3-MPI; ls
BT/       CG/     DT/      EP/      FT/      IS/      LU/      MG/      SP/
Bin/      common/ config/  Makefile README   README.tutorial   sys/
```

- Edit config/make.def to adjust build configuration
  - Adjust MPIF77 compiler specification
- Make clean and build new tool-specific executable
-

```
% make clean
% make bt CLASS=W NPROCS=16
...
Built executable ../bin.$(TOOL)/bt_W.16
```

- Change to the directory containing the new executable and run it as directed

```
% cd bin.$(TOOL)
% mpiexec –np 16 bt_W.16
```

BT - Applying Marmot

- Keep in mind:
  - The following steps will be similar for all of the other tools
- Select the Marmot compiler wrapers

```
% gedit config/make.def
        -> comment out line 32, resulting in:
                        ...
                        32: #MPIF77 = mpif77
                        ...
        -> remove the comment from line 35, resulting in:
                        ...
                        35: MPIF77 = marmotf77
                        ...
                -> comment out line 86, resulting in:
                        ...
                        86: #MPICC = mpicc
                        ...
        -> remove the comment from line 89, resulting in:
                ...
                89: MPICC = marmotcc
                        ...
```

BT- Building with Marmot

- No further modifications needed to build with Marmot

```
% make bt NPROCS=16 CLASS=W
cd BT; make NPROCS=4 CLASS=W SUBTYPE= VERSION=
gmake: Entering directory 'BT'
cd ../sys; cc  -o setparams setparams.c
../sys/setparams bt 4 W
marmotf77 -c  -O bt.f
...
marmotf77 -O -o ../bin/bt_W.16 \
bt.o make_set.o initialize.o exact_solution.o exact_rhs.o \
set_constants.o adi.o define.o copy_faces.o rhs.o solve_subs.o \
x_solve.o y_solve.o z_solve.o add.o error.o verify.o setup_mpi.o \
../common/print_results.o ../common/timers.o btio.o
Built executable ../bin.marmot/bt_W.16
gmake: Leaving directory 'BT'
```

- Marmot-instrumented executable(s) installed in dedicated bin.marmot subdirectory
- Running the executable(s) in that directory will keep output files generated by Marmot together

BT- Executing with Marmot

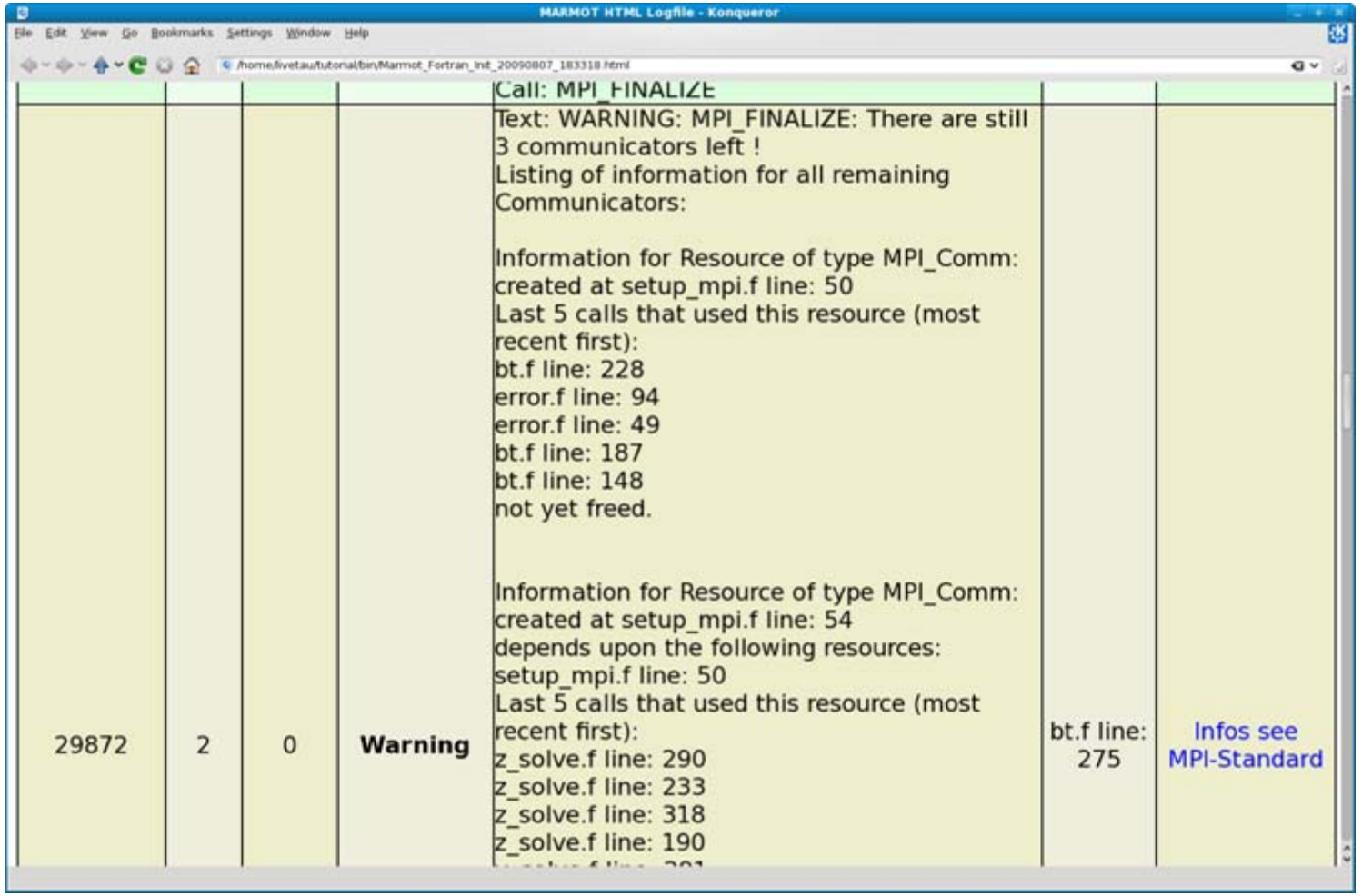
- Keep in mind to use one more process:

```
% cd bin.marmot
% mpiexec –np 17 bt_W.16
 NAS Parallel Benchmarks 3.3 -- BT Benchmark
 Size:   24x  24x  24
 Iterations:  200    dt:   0.0008000
 Number of active processes:    16

 Time step    1
 Time step   20
 Time step   40
 Time step   60
 Time step   80
 Time step  100
 Time step  120
 Time step  140
 Time step  160
 Time step  180
 Time step  200
 Verification Successful

 BT Benchmark Completed.
 Time in seconds = 30.40
```

BT – Marmot Results (1/2)

- Open the Marmot output:

```
% konqueror Marmot_Fortran_Init_<TIMESTAMP>.html
```



				Call: MPI_FINALIZE		
29872	2	0	**Warning**	Text: WARNING: MPI_FINALIZE: There are still 3 communicators left ! Listing of information for all remaining Communicators: Information for Resource of type MPI_Comm: created at setup_mpi.f line: 50 Last 5 calls that used this resource (most recent first): bt.f line: 228 error.f line: 94 error.f line: 49 bt.f line: 187 bt.f line: 148 not yet freed. Information for Resource of type MPI_Comm: created at setup_mpi.f line: 54 depends upon the following resources: setup_mpi.f line: 50 Last 5 calls that used this resource (most recent first): z_solve.f line: 290 z_solve.f line: 233 z_solve.f line: 318 z_solve.f line: 190	bt.f line: 275	Infos see MPI-Standard

BT – Marmot Results (2/2)

- The code contains no severe errors
- (Is to be expected from a widely used benchmark)
- However:
  - Marmot detects that three communicators are not freed
  - A quality application should free all MPI ressources
  - Marmot also lists details for each of these communicators:
    - Were is it created
    - Where was it used last