Hands-on:
NPB-MZ-MPI / BT

Performance Analysis Steps

- 0.0 Reference preparation for validation

- 1.0 Program instrumentation
- 1.1 Summary measurement collection
- 1.2 Summary analysis report examination

- 2.0 Summary experiment scoring
- 2.1 Summary measurement collection with filtering
- 2.2 Filtered summary analysis report examination

- 3.0 Event trace collection
- 3.1 Event trace examination & analysis

Scalasca command – One command for (almost) everything

```
% scalasca
Scalasca 2.2
Toolset for scalable performance analysis of large-scale parallel applications
usage: scalasca [OPTION]... ACTION <argument>...
    1. prepare application objects and executable for measurement:
       scalasca -instrument <compile-or-link-command> # skin (using scorep)
    2. run application under control of measurement system:
       scalasca -analyze <application-launch-command> # scan
    3. interactively explore measurement analysis report:
       scalasca -examine <experiment-archive|report>  # square

Options:
   -c, --show-config  show configuration summary and exit
   -h, --help         show this help and exit
   -n, --dry-run      show actions without taking them
       --quickref     show quick reference guide and exit
   -v, --verbose      enable verbose commentary
   -V, --version      show version information and exit
```

- The ‘`scalasca -instrument`’ command is deprecated and only provided for backwards compatibility with Scalasca 1.x., recommended: use Score-P instrumenter directly

Scalasca compatibility command: skin

```
% skin
Scalasca 2.2: application instrumenter (using Score-P instrumenter)
usage: skin [-v] [-comp] [-pdt] [-pomp] [-user] [--*] <compile-or-link-command>
  -comp={all|none|...}: routines to be instrumented by compiler [default: all]
                        (... custom instrumentation specification depends on compiler)
  -pdt:  process source files with PDT/TAU instrumenter
  -pomp: process source files for POMP directives
  -user: enable EPIK user instrumentation API macros in source code
  -v:    enable verbose commentary when instrumenting

  --*:   options to pass to Score-P instrumenter
```

- Scalasca application instrumenter
  - Provides compatibility with Scalasca 1.x
  - Recommended: use Score-P instrumenter directly

Scalasca convenience command: scan

```
% scan
Scalasca 2.2: measurement collection & analysis nexus
usage: scan {options} [launchcmd [launchargs]] target [targetargs]
       where {options} may include:
  -h    Help: show this brief usage message and exit.
  -v    Verbose: increase verbosity.
  -n    Preview: show command(s) to be launched but don't execute.
  -q    Quiescent: execution with neither summarization nor tracing.
  -s    Summary: enable runtime summarization. [Default]
  -t    Tracing: enable trace collection and analysis.
  -a    Analyze: skip measurement to (re-)analyze an existing trace.
  -e exptdir   : Experiment archive to generate and/or analyze.
                 (overrides default experiment archive title)
  -f filtfile  : File specifying measurement filter.
  -l lockfile  : File that blocks start of measurement.
  -m metrics   : Metric specification for measurement.
```

- Scalasca measurement collection & analysis nexus

Scalasca advanced command: scout - Scalasca automatic trace analyzer

```
% scout.hyb --help
SCOUT   Copyright (c) 1998-2015 Forschungszentrum Juelich GmbH
        Copyright (c) 2009-2014 German Research School for Simulation
                                Sciences GmbH

Usage: <launchcmd> scout.hyb [OPTION]... <ANCHORFILE | EPIK DIRECTORY>
Options:
  --statistics       Enables instance tracking and statistics [default]
  --no-statistics    Disables instance tracking and statistics
  --critical-path    Enables critical-path analysis [default]
  --no-critical-path Disables critical-path analysis
  --rootcause        Enables root-cause analysis [default]
  --no-rootcause     Disables root-cause analysis
  --single-pass      Single-pass forward analysis only
  --time-correct     Enables enhanced timestamp correction
  --no-time-correct  Disables enhanced timestamp correction [default]
  --verbose, -v      Increase verbosity
  --help             Display this information and exit
```

- Provided in serial (.ser), OpenMP (.omp), MPI (.mpi) and MPI+OpenMP (.hyb) variants

Scalasca advanced command: clc_synchronize

- Scalasca trace event timestamp consistency correction

```
Usage: <launchcmd> clc_synchronize.hyb <ANCHORFILE | EPIK_DIRECTORY>
```

  - Provided in MPI (.mpi) and MPI+OpenMP (.hyb) variants
  - Takes as input a trace experiment archive where the events may have timestamp inconsistencies
    - e.g., multi-node measurements on systems without adequately synchronized clocks on each compute node
  - Generates a new experiment archive (always called ./clc_sync) containing a trace with event timestamp inconsistencies resolved
    - e.g., suitable for detailed examination with a time-line visualizer

Scalasca convenience command: square

```
% square
Scalasca 2.2: analysis report explorer
usage: square [-v] [-s] [-f filtfile] [-F] <experiment archive | cube file>
   -c <none | quick | full> : Level of sanity checks for newly created reports
   -F                       : Force remapping of already existing reports
   -f filtfile              : Use specified filter file when doing scoring
   -s                       : Skip display and output textual score report
   -v                       : Enable verbose mode
   -n                       : Do not include idle thread metric
```

- Scalasca analysis report explorer

Automatic measurement configuration

- scan configures Score-P measurement by automatically setting some environment variables and exporting them
  - e.g., experiment title, profiling/tracing mode, filter file, …
  - Precedence order:
    - Command-line arguments
    - Environment variables already set
    - Automatically determined values
- Also, scan includes consistency checks and prevents corrupting existing experiment directories
- For tracing experiments, after trace collection completes then automatic parallel trace analysis is initiated
  - uses identical launch configuration to that used for measurement (i.e., the same allocated compute resources)

Setup environment

- Load module

```
% module load scalasca
```

- Change to directory containing NPB3.3-MZ-MPI sources
- Existing instrumented executable in bin.scorep/ directory can be reused

BT-MZ summary measurement collection...

```
% cd bin.scorep
% cp ../jobscript/{hamilton,cosma,archer}/scalasca2.* ./
% vi scalasca2.*
 [...]
export OMP_NUM_THREADS=4
CLASS=C
NPROCS=8
EXE=./bt-mz_$CLASS.$NPROCS

#export SCOREP_FILTERING_FILE=../config/scorep.filt
#export SCOREP_TOTAL_MEMORY=78M

scalasca -analyze -s  mpiexec -np $NPROCS  $EXE
```

- Change to directory with the executable and edit the job script

`% bsub -q bench1 -P durham < scalasca2.lsf`	COSMA
`% sbatch -p bench2 -A bench scalasca2.sbatch`	Hamilton
`% qsub scalasca2.pbs`	Archer

- Submit the job

BT-MZ summary measurement

```
S=C=A=N: Scalasca 2.2 runtime summarization
S=C=A=N: ./scorep_bt-mz_C_8x4_sum experiment archive
S=C=A=N: Thu Sep 13 18:05:17 2012: Collect start
mpiexec -np 8 ./bt-mz_C.8

 NAS Parallel Benchmarks (NPB3.3-MZ-MPI) -
    BT-MZ MPI+OpenMP Benchmark

 Number of zones:   8 x   8
 Iterations: 200    dt:   0.000300
 Number of active processes:     8

          [... More application output ...]

S=C=A=N: Thu Sep 13 18:05:39 2012: Collect done (status=0) 22s
S=C=A=N: ./scorep_bt-mz_C_8x4_sum complete.
```

- Run the application using the Scalasca measurement collection & analysis nexus prefixed to launch command
- Creates experiment directory: ./scorep_bt-mz_C_8x4_sum

BT-MZ summary analysis report examination

- Score summary analysis report

```
% square -s  scorep_bt-mz_C_8x4_sum
INFO: Post-processing runtime summarization result...
INFO: Score report written to ./scorep_bt-mz_C_8x4_sum/scorep.score
```

- Post-processing and interactive exploration with CUBE

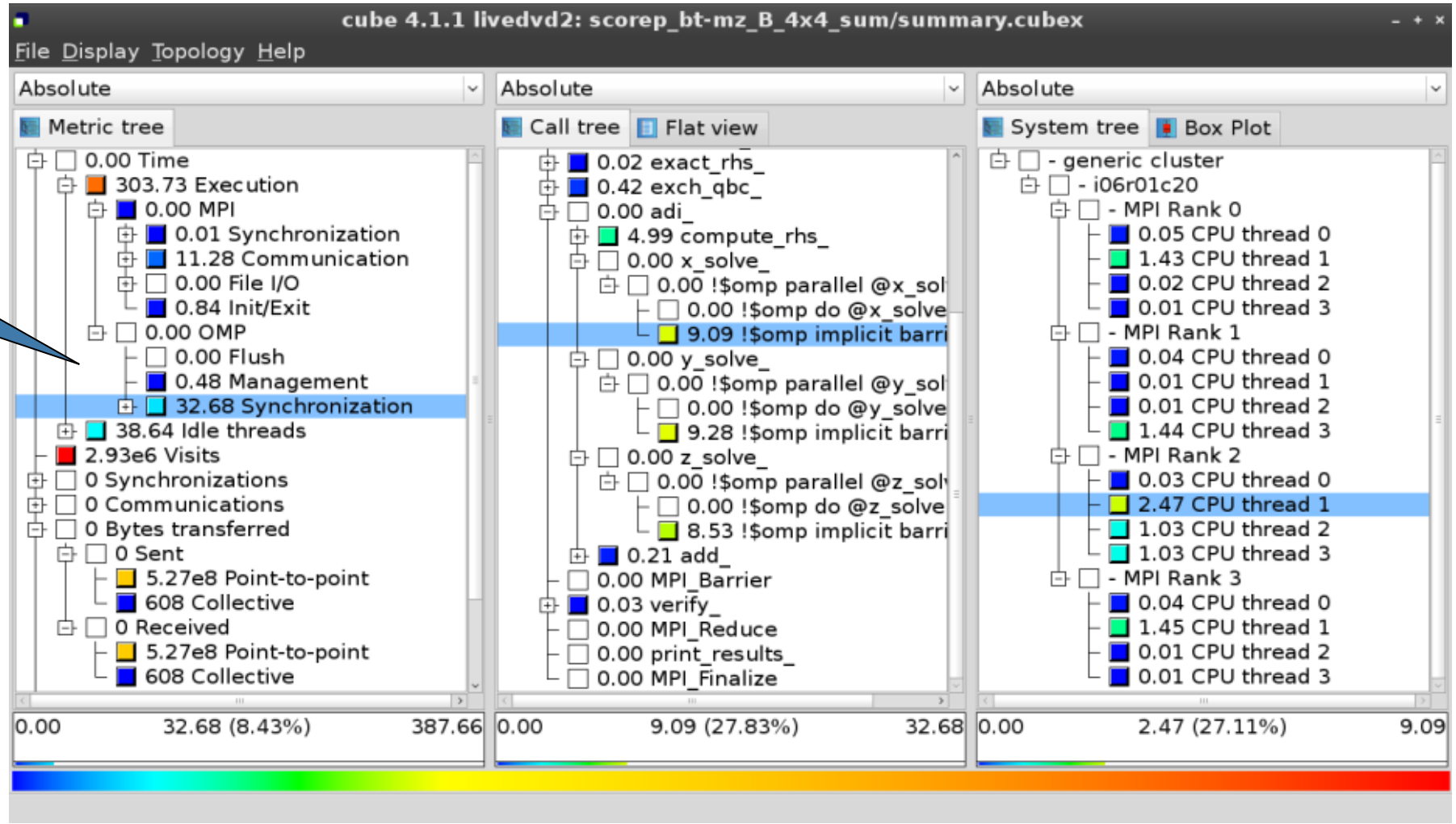
```
% square  scorep_bt-mz_C_8x4_sum
INFO: Displaying ./scorep_bt-mz_C_8x4_sum/summary.cubex...

[GUI showing summary analysis report]
```

- The post-processing derives additional metrics and generates a structured metric hierarchy

Post-processed summary analysis report

Split base metrics into more specific metrics



Performance Analysis Steps

- 0.0 Reference preparation for validation

- 1.0 Program instrumentation
- 1.1 Summary measurement collection
- 1.2 Summary analysis report examination

- 2.0 Summary experiment scoring
- 2.1 Summary measurement collection with filtering
- 2.2 Filtered summary analysis report examination

- 3.0 Event trace collection
- 3.1 Event trace examination & analysis

BT-MZ trace measurement collection...

```
% cd bin.scorep
% cp ../jobscript/{hamilton,cosma,archer}/scalasca2.* ./
% vi scalasca2.*

  [...]

export OMP_NUM_THREADS=4
CLASS=C
NPROCS=8
EXE=./bt-mz_$CLASS.$NPROCS

export SCOREP_FILTERING_FILE=../config/scorep.filt
export SCOREP_TOTAL_MEMORY=168M
export SCOREP_METRIC_PAPI=PAPI_TOT_INS,PAPI_TOT_CYC

scalasca -analyze -t  mpiexec -np $NPROCS $EXE
```

- Change to directory with executable and edit job script

`% bsub -q bench1 -P durham < scalasca2.lsf`	COSMA
`% sbatch -p bench2 -A bench scalasca2.sbatch`	Hamilton
`% qsub scalasca2.pbs`	Archer

- Submit the job

BT-MZ trace measurement … collection

```
S=C=A=N: Scalasca 2.2 trace collection and analysis
S=C=A=N: Fri Sep 20 15:09:59 2013: Collect start
mpiexec -np 8 ./bt-mz_C.8

 NAS Parallel Benchmarks (NPB3.3-MZ-MPI) - BT-MZ MPI+OpenMP \
>Benchmark

 Number of zones:   8 x   8
 Iterations: 200    dt:   0.000300
 Number of active processes:     8

          [... More application output ...]

S=C=A=N: Fri Sep 20 15:10:16 2013: Collect done (status=0) 28s
```

- Starts measurement with collection of trace files …

BT-MZ trace measurement … analysis

```
S=C=A=N: Fri Sep 20 15:09:59 2013: Analyze start
mpiexec -np 8 scout.hyb ./scorep_bt-mz_C_8x4_trace/traces.otf2

Analyzing experiment archive
./scorep_bt-mz_C_8x4_trace/traces.otf2

Opening experiment archive ... done (0.019s).
Reading definition data    ... done (0.178s).
Reading event trace data   ... done (2.068s).
Preprocessing              ... done (3.789s).
Analyzing trace data       ... done (5.413s).
Writing analysis report    ... done (1.994s).

Total processing time: 34.812s
S=C=A=N: Fri Sep 20 15:10:16 2013: Analyze done (status=0) 39s
```

- Continues with automatic (parallel) analysis of trace files

BT-MZ trace analysis report exploration

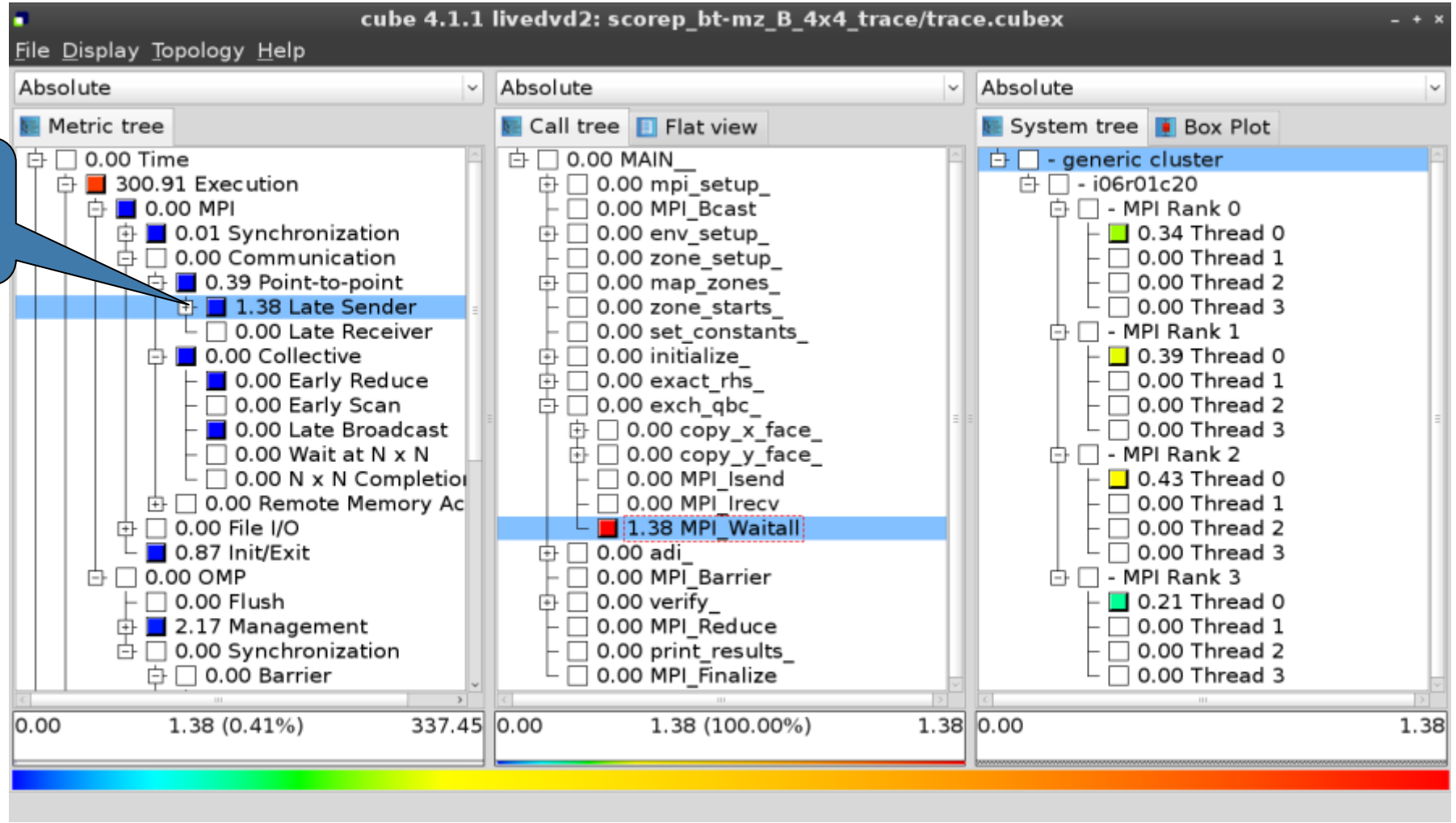
- Produces trace analysis report in the experiment directory containing trace-based wait-state metrics

```
% square  scorep_bt-mz_C_8x4_trace
INFO: Post-processing runtime summarization result...
INFO: Post-processing trace analysis report...
INFO: Displaying ./scorep_bt-mz_C_8x4_trace/trace.cubex...

[GUI showing trace analysis report]
```

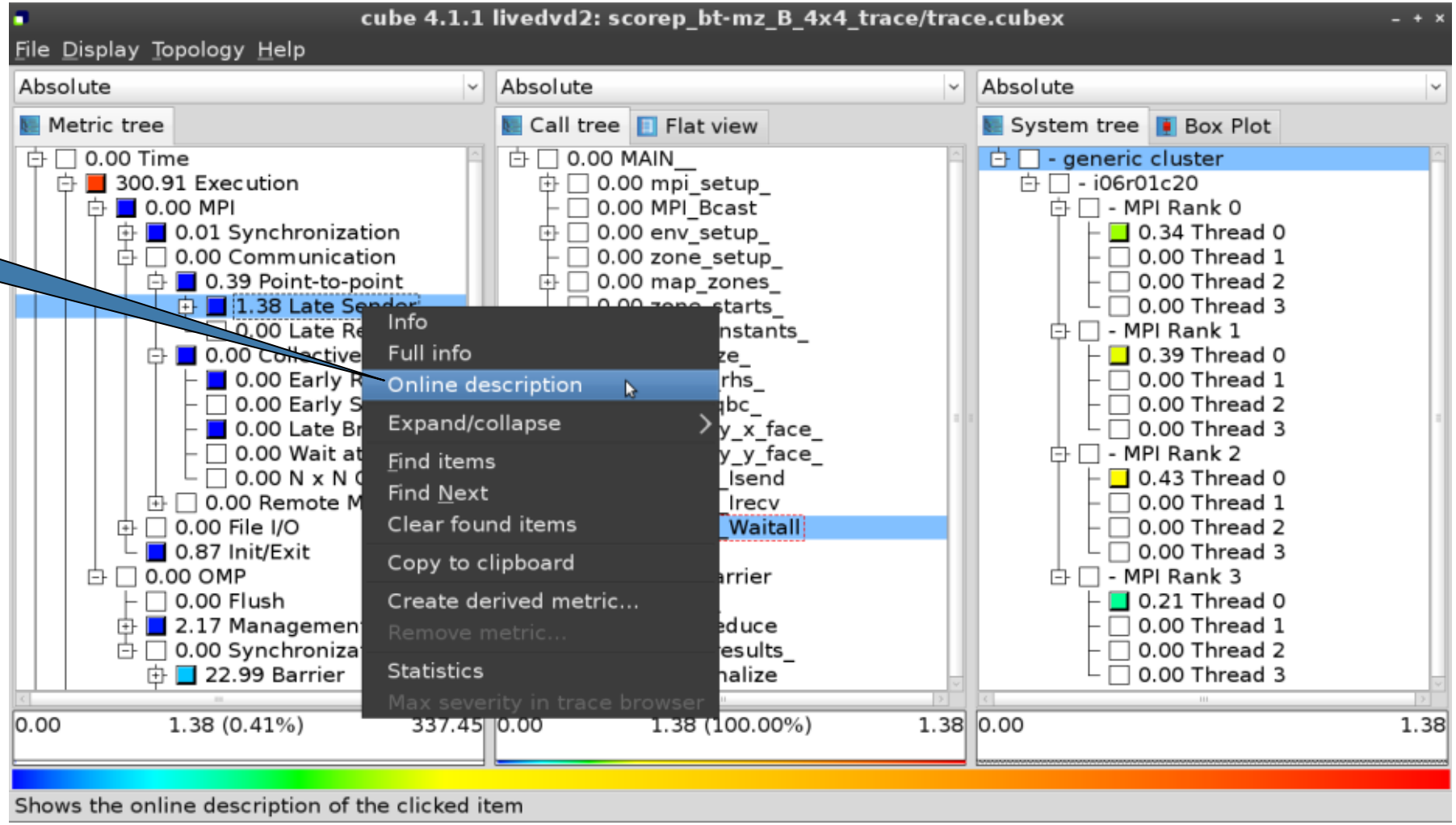
Post-processed trace analysis report

Additional trace-based metrics in metric hierarchy



Online metric description

Access online metric description via context menu



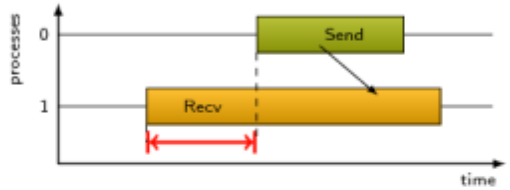
Online metric description

Performance properties

Late Sender Time

Description:

Refers to the time lost waiting caused by a blocking receive operation (e.g., `MPI_Recv` or `MPI_Wait`) that is posted earlier than the corresponding send operation.



If the receiving process is waiting for multiple messages to arrive (e.g., in an call to `MPI_Waitall`), the maximum waiting time is accounted, i.e., the waiting time due to the latest sender.

Unit:

Seconds

Diagnosis:

Try to replace `MPI_Recv` with a non-blocking receive `MPI_Irecv` that can be posted earlier, proceed concurrently with computation, and complete with a wait operation after the message is expected to have been sent. Try to post sends earlier, such that they are available when receivers need them. Note that outstanding messages (i.e., sent before the receiver is ready) will occupy internal message buffers, and that large numbers of posted receive buffers will also introduce message management overhead, therefore moderation is advisable.

Parent:

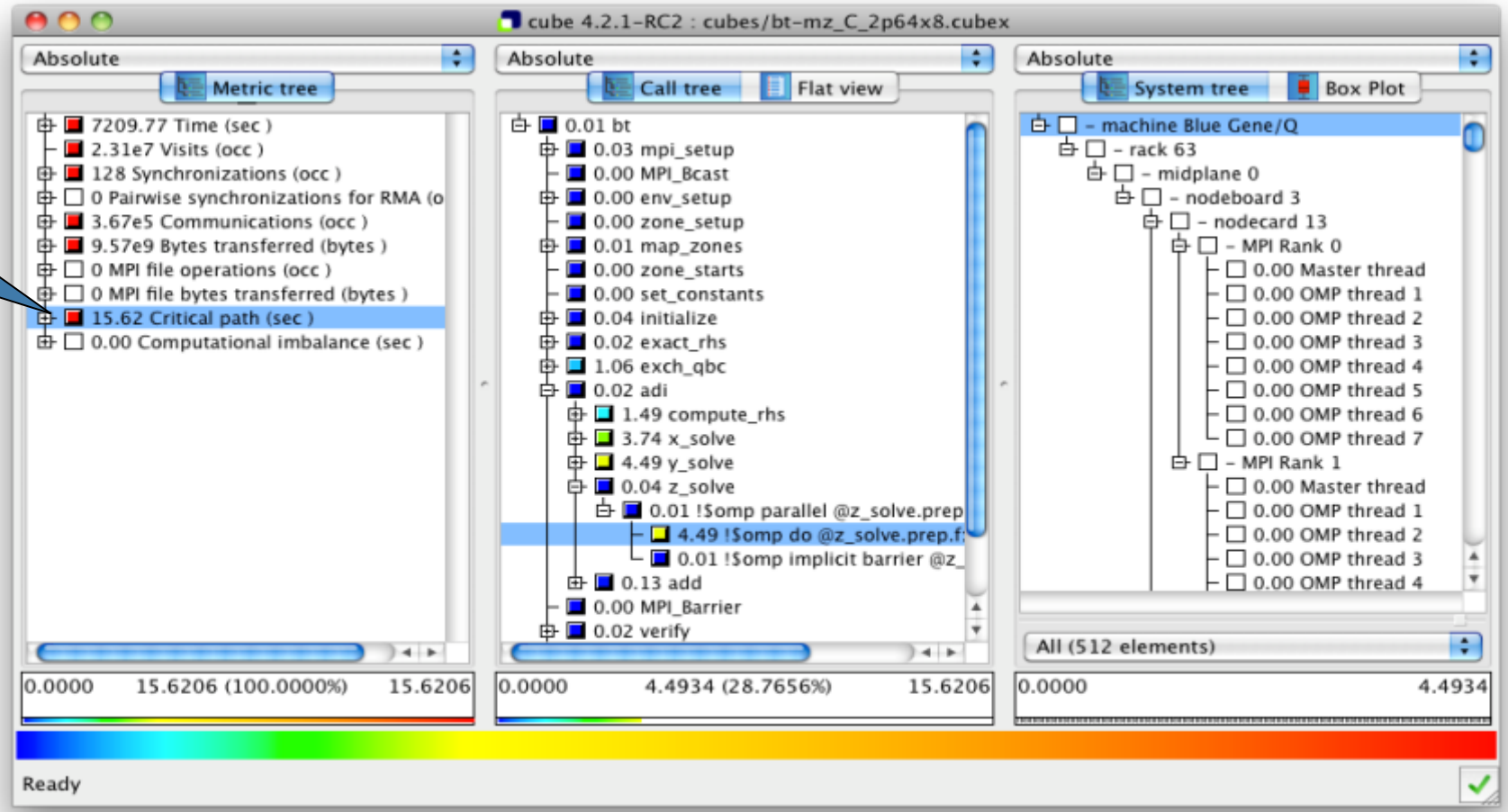
MPI Point-to-point Communication Time

Children:

Close

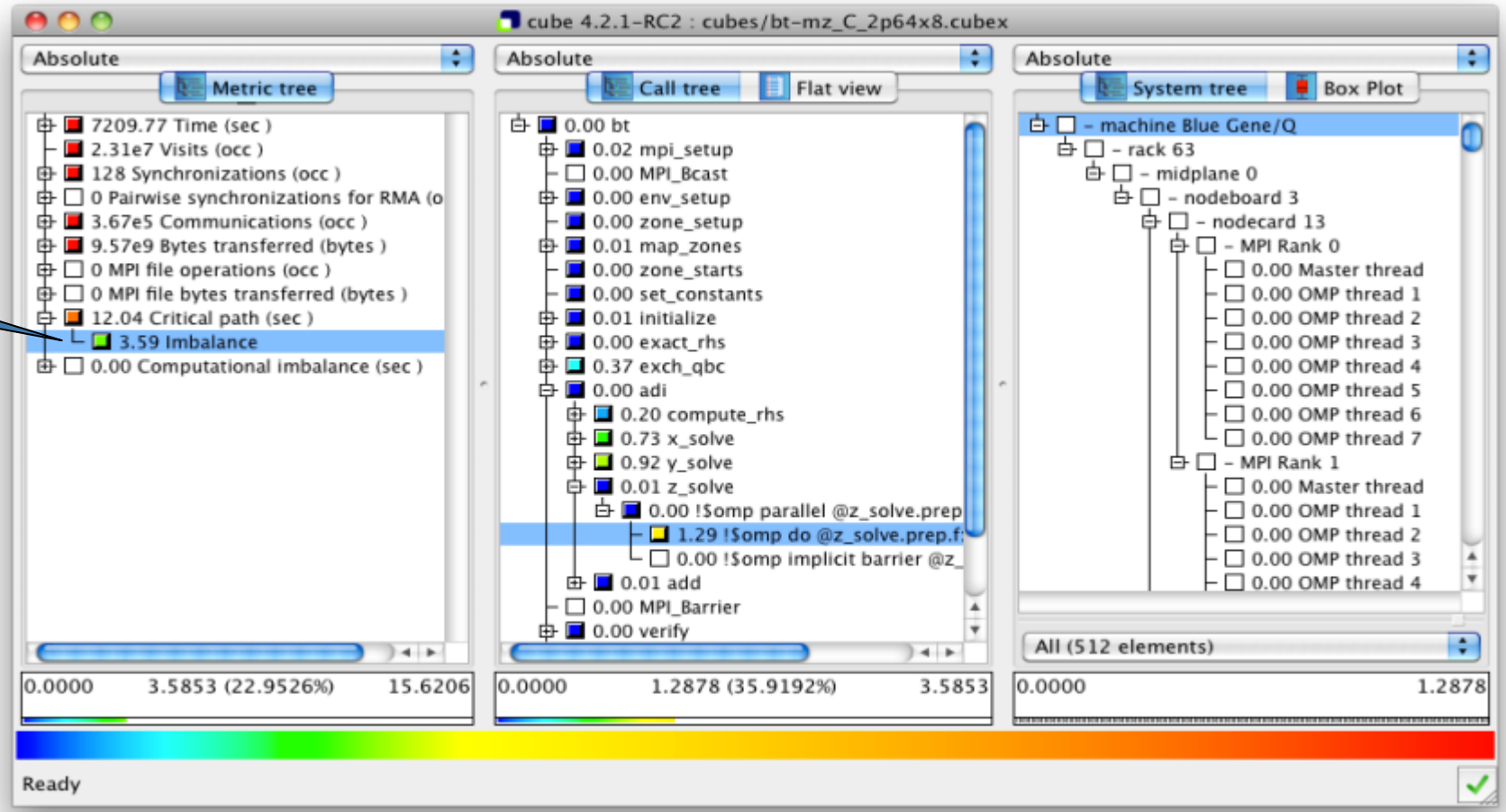
Critical-path analysis

Critical-path profile shows wall-clock time impact



Critical-path analysis

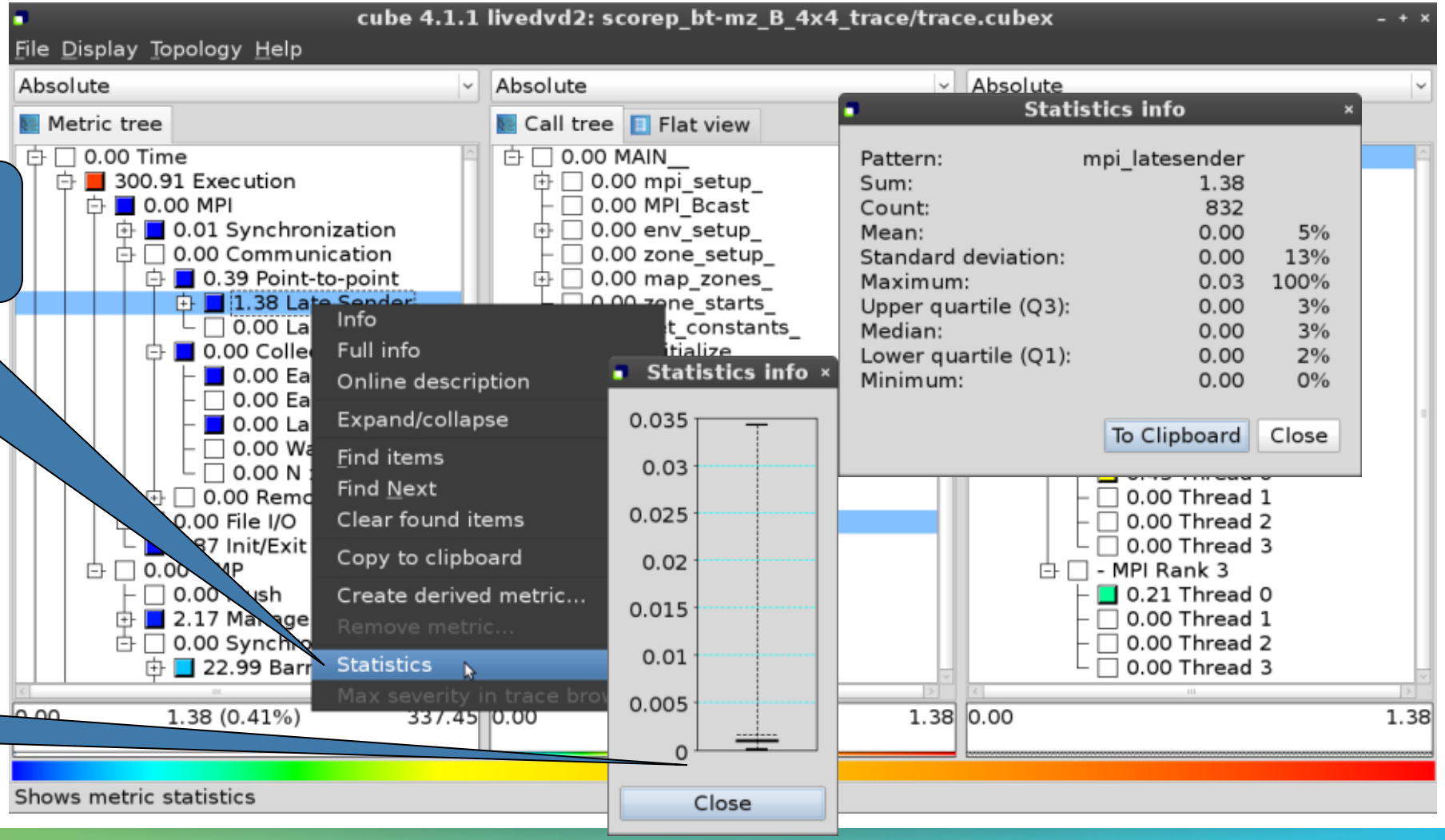
Critical-path imbalance highlights inefficient parallelism



Pattern instance statistics

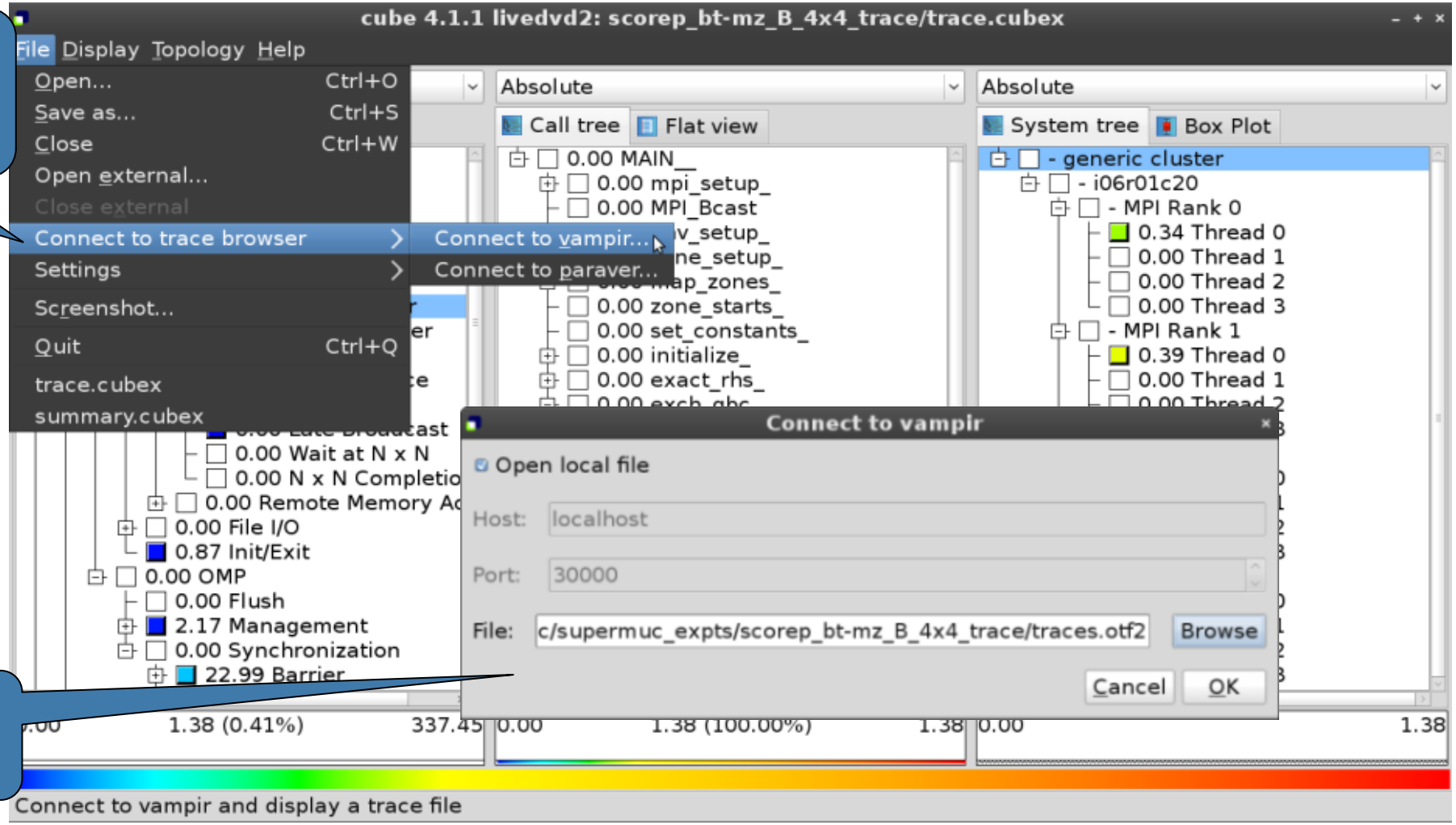
Access pattern instance statistics via context menu

Click to get statistics details



Connect to Vampir trace browser

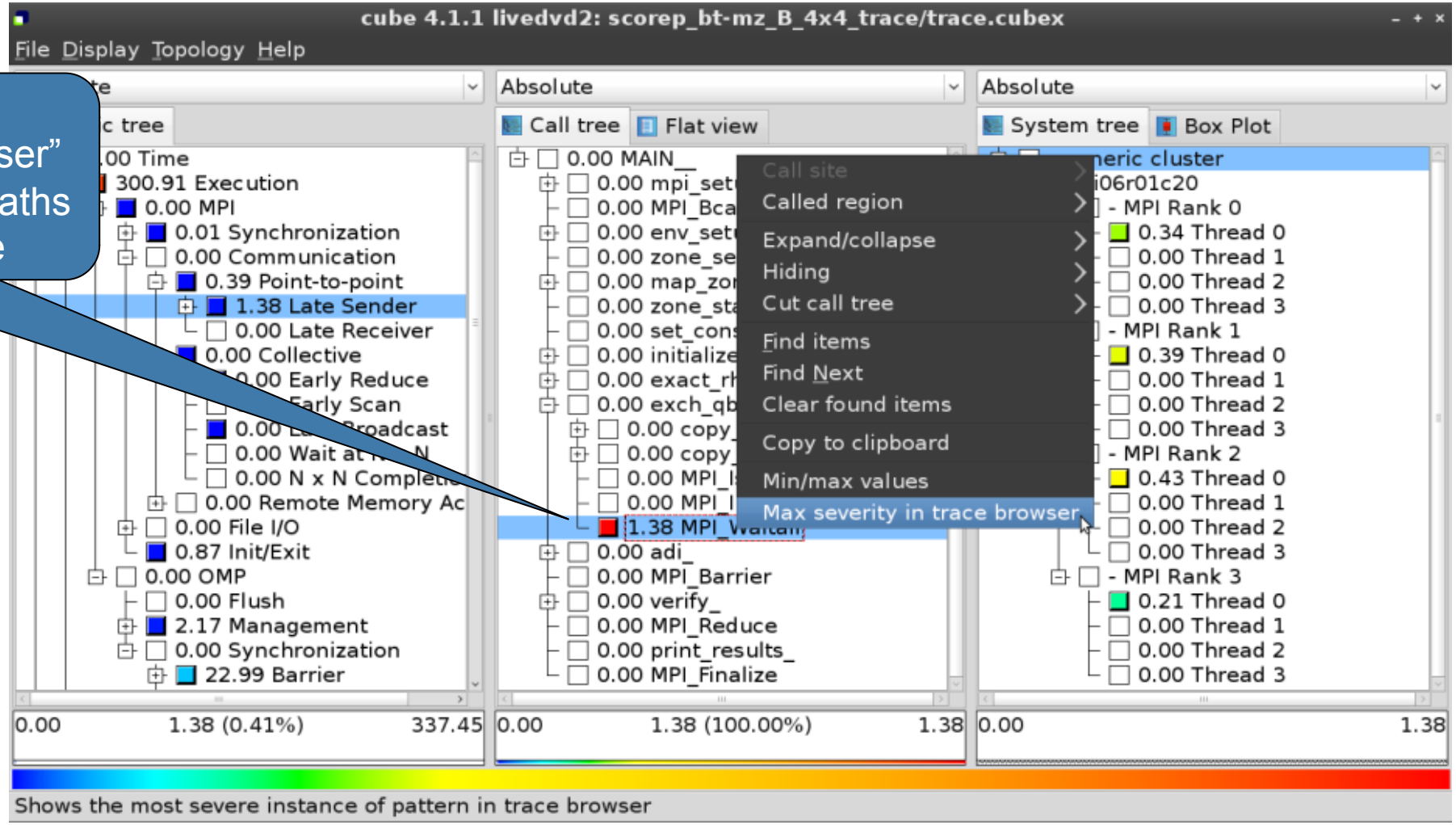
To investigate most severe pattern instances, connect to a trace browser…



…and select trace file from the experiment directory

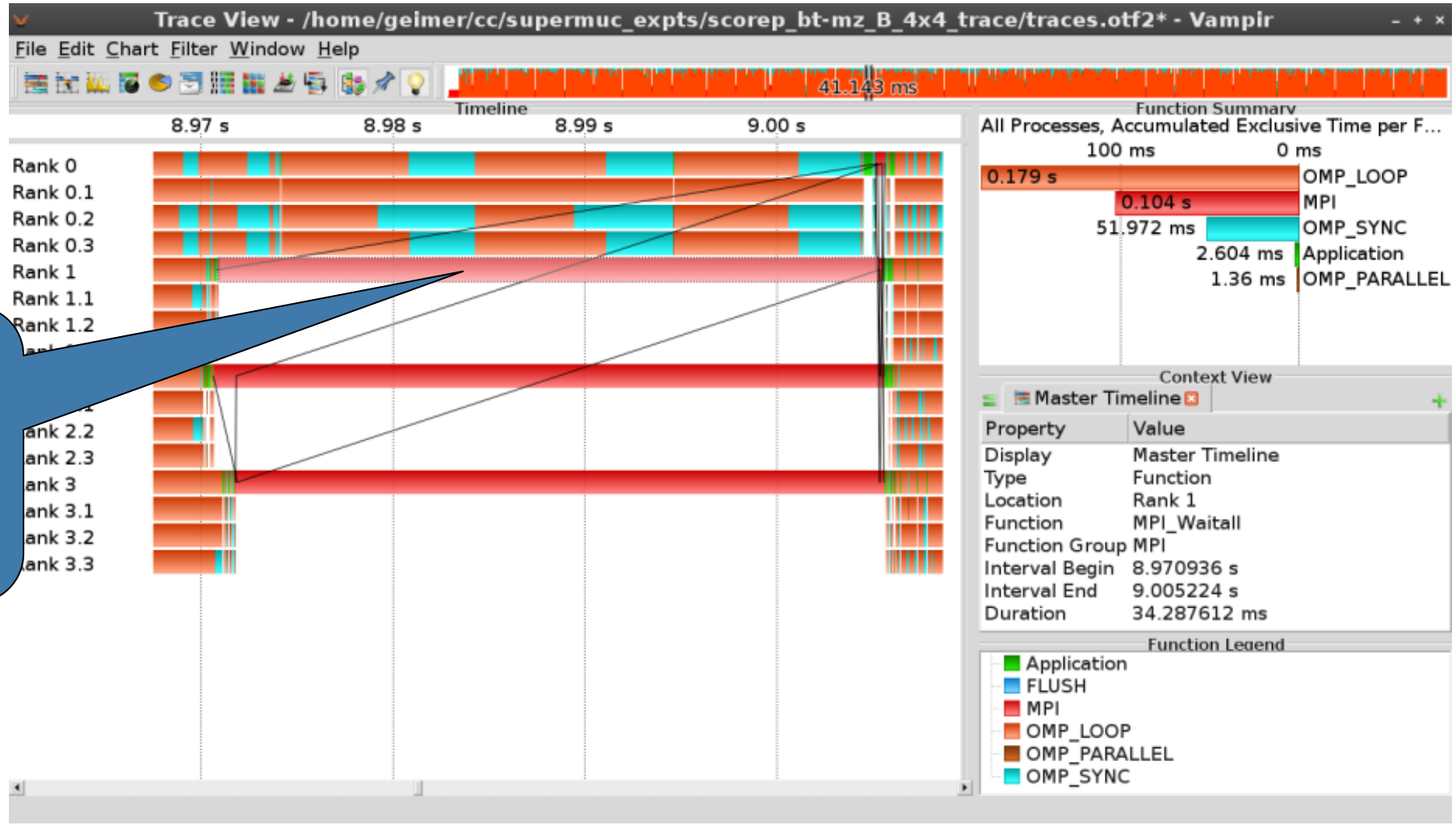
Show most severe pattern instances

Select
“Max severity in trace browser”
from context menu of call paths
marked with a red frame



Investigate most severe instance in Vampir

Vampir will automatically zoom to the worst instance in multiple steps (i.e., undo zoom provides more context)



Further information

Scalable performance **a**nalysis of **la**rge-**sc**ale parallel **a**pplications

- toolset for scalable performance measurement & analysis of MPI, OpenMP & hybrid parallel applications
- supporting most popular HPC computer systems
- available under New BSD open-source license
- sources, documentation & publications:
  - http://www.scalasca.org
  - mailto: scalasca@fz-juelich.de

scalasca

BT-MZ trace analysis

```
% OMP_NUM_THREADS=4 scan -a mpiexec -np 8 ./bt-mz_C.8
S=C=A=N: Scalasca 2.2 trace analysis
S=C=A=N: Fri Sep 20 15:09:59 2013: Analyze start
mpiexec -np 8 scout.hyb ./scorep_bt-mz_C_8x4_trace/traces.otf2

Analyzing experiment archive
            ./scorep_bt-mz_C_8x4_trace/traces.otf2

Opening experiment archive ... done (0.019s).
Reading definition data    ... done (0.178s).
Reading event trace data   ... done (2.068s).
Preprocessing              ... done (3.789s).
Analyzing trace data       ...
  Wait-state detection (fwd)      (1/4) ... done (2.889s).
  Wait-state detection (bwd)      (2/4) ... done (1.136s).
  Synchpoint exchange (fws)       (3/4) ... done (0.813s).
  Critical-path & delay analysis  (4/4) ... done (0.568s).
done (5.413s).
Writing analysis report    ... done (1.994s).

Total processing time: 34.812s
S=C=A=N: Fri Sep 20 15:10:16 2013: Analyze done (status=0) 39s
```

- Automatic trace analysis of existing experiment archives

BT-MZ trace measurement & time-corrected analysis

```
% SCAN_TRACE_ANALYZER=none  scan -t  mpiexec -np 8 ./bt-mz_C.8
S=C=A=N: Scalasca 2.2 trace collection and analysis
Info: Automatic trace analysis will be skipped!
...
S=C=A=N: Fri Mar 21: 18:00:56 2014: Collect done (status=0) 28s
S=C=A=N: ./scorep_bt-mz_C_8x4_trace complete.
% cd scorep_bt-mz_C_8x4_trace
% mpiexec -np 8 clc_synchronize.hyb ./traces.otf2

        # passes         : 1
        # violated       : 3362
        # corrected      : 1610977
        # reversed-p2p   : 233
        # reversed-coll  : 0
        # reversed-omp   : 3129
        # events         : 6287852
        max. error       : 0.000112 [s]
        error at final.  : 0.000118 [%]
        Max slope        : 0.010000000
% scan -a -e ./clc_sync  mpiexec -np 8 ../bt-mz_C.8
S=C=A=N: Scalasca 2.2 trace analysis
...
S=C=A=N: Fri Mar 21 18:29:29 2014: Analyze done (status=0) 39s
S=C=A=N: ./clc_sync complete
```

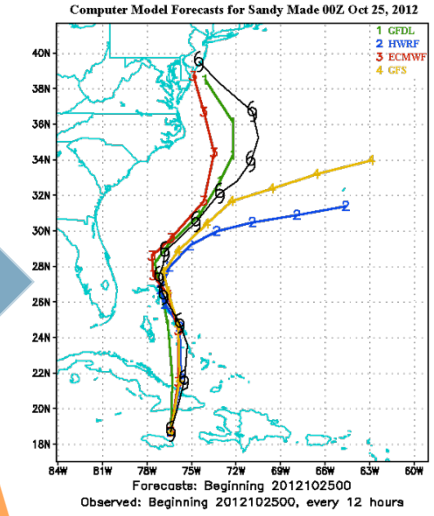
- Generating a time-corrected trace and its analysis

Hands-On Exercise: Analyzing for MPI Usage Errors with MUST

VI-HPS Team

MUST

Runtime Correctness Analysis Workflow



Results

Rank	Thread	Type	Message	From	References	MPI-Standard Reference
1		Error	A send and a receive operation use datatypes that do not match! Missmatch occurs at (CONTIGUOUS)[0] (MPI_INT) in the send type and at (MPI_BYTE) in the receive type (consult the MUST manual for a detailed description of datatype positions). The send operation was started at reference 1, the receive operation was started at reference 2. (Information on communicator: MPI_COMM_WORLD) (Information on send of count 1 with type:Datatype created at reference 3 is for C, commited at reference 4, based on the following type(s): { MPI_INT}Typemap = {(MPI_INT, 0), (MPI_INT, 4)}) (Information on receive of count 8 with type:MPI_BYTE)	**MPI_Sendrecv** from: #0 main@test.c:54 #1 start_main@libc.so	reference 1 rank 0: **MPI_Sendrecv** from: #0 main@test.c:54 #1 start_main@libc.so reference 2 rank 1: **MPI_Sendrecv** from: #0 main@test.c:54 #1 start_main@libc.so reference 3 rank 0: **MPI_Type_contiguous** from: #0 main@test.c:35 #1 start_main@libc.so reference 4 rank 0: **MPI_Type_commit** from: #0 main@test.c:51 #1 start_main@libc.so	

Correctness report

NPB-MZ-MPI / BT – Make MUST available

- COSMA

```
% #Nothing to do, its in the \
   default modules
% cd <…>/NPB3.3-MZ-MPI
```

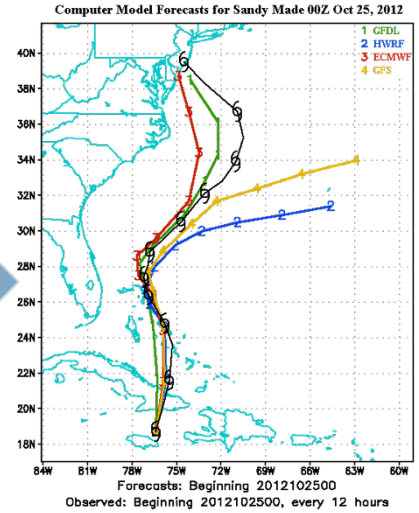
- Hamilton

```
% module load \
    must/impi/intel/1.4.0 \
    intel/xe_2015.2 \
    intelmpi/intel/5.0.3
% cd <…>/NPB3.3-MZ-MPI
```

- Archer

```
% module use \
 /home/y07/y07/scalasca/modules
% module switch \
         PrgEnv-cray PrgEnv-gnu
% module load must
% cd <…>/NPB3.3-MZ-MPI
```

Overview – Next: Attach MUST to the application



Results

Rank	Thread	Type	Message	From	References	MPI-Standard Reference
1		Error	A send and a receive operation use datatypes that do not match! Missmatch occurs at (CONTIGUOUS)[0] (MPI_INT) in the send type and at (MPI_BYTE) in the receive type (consult the MUST manual for a detailed description of datatype positions). The send operation was started at reference 1, the receive operation was started at reference 2. (Information on communicator: MPI_COMM_WORLD) (Information on send of count 1 with type:Datatype created at reference 3 is for C, commited at reference 4, based on the following type(s): { MPI_INT}Typemap = {(MPI_INT, 0), (MPI_INT, 4)}) (Information on receive of count 8 with type:MPI_BYTE)	**MPI_Sendrecv** from: #0 main@test.c:54 #1 start_main@libc.so	reference 1 rank 0: **MPI_Sendrecv** from: #0 main@test.c:54 #1 start_main@libc.so reference 2 rank 1: **MPI_Sendrecv** from: #0 main@test.c:54 #1 start_main@libc.so reference 3 rank 0: **MPI_Type_contiguous** from: #0 main@test.c:35 #1 start_main@libc.so reference 4 rank 0: **MPI_Type_commit** from: #0 main@test.c:51 #1 start_main@libc.so	

Correctness report

NPB-MZ-MPI / BT – All we need is a dynamically linked executable

- Edit config/make.def to adjust build configuration
  - Remove tools from the specifications of MPIF77 and COMPFLAGS

- COSMA and Hamilton

```
#            SITE- AND/OR PLATFORM-SPECIFIC …
#-------------------------------------------
# Items in this file may need to be changed …
#-------------------------------------------
COMPFLAGS = -openmp
...
#-------------------------------------------
# The Fortran compiler used for MPI programs
#-------------------------------------------
MPIF77 = mpiifort

...
#MPIF77 = scorep --user mpiifort
...
FFLAGS  = -O3 $(OPENMP) -g
```

- Archer

```
#            SITE- AND/OR PLATF     PECIFIC …
#-------------------------------     ----------
# Items in this file may need     e changed …
#-------------------------------     ----------
COMPFLAGS = -fopenmp
...
#-------------------------------------------
# The Fortran compiler used     r MPI programs
#-------------------------------------------
MPIF77 = ftn

...
#MPIF77 = scorep --user ftn
...
FFLAGS  = -O3 $(OPENMP) -g -dynamic
```

Comment all of them out

-dynamic is important

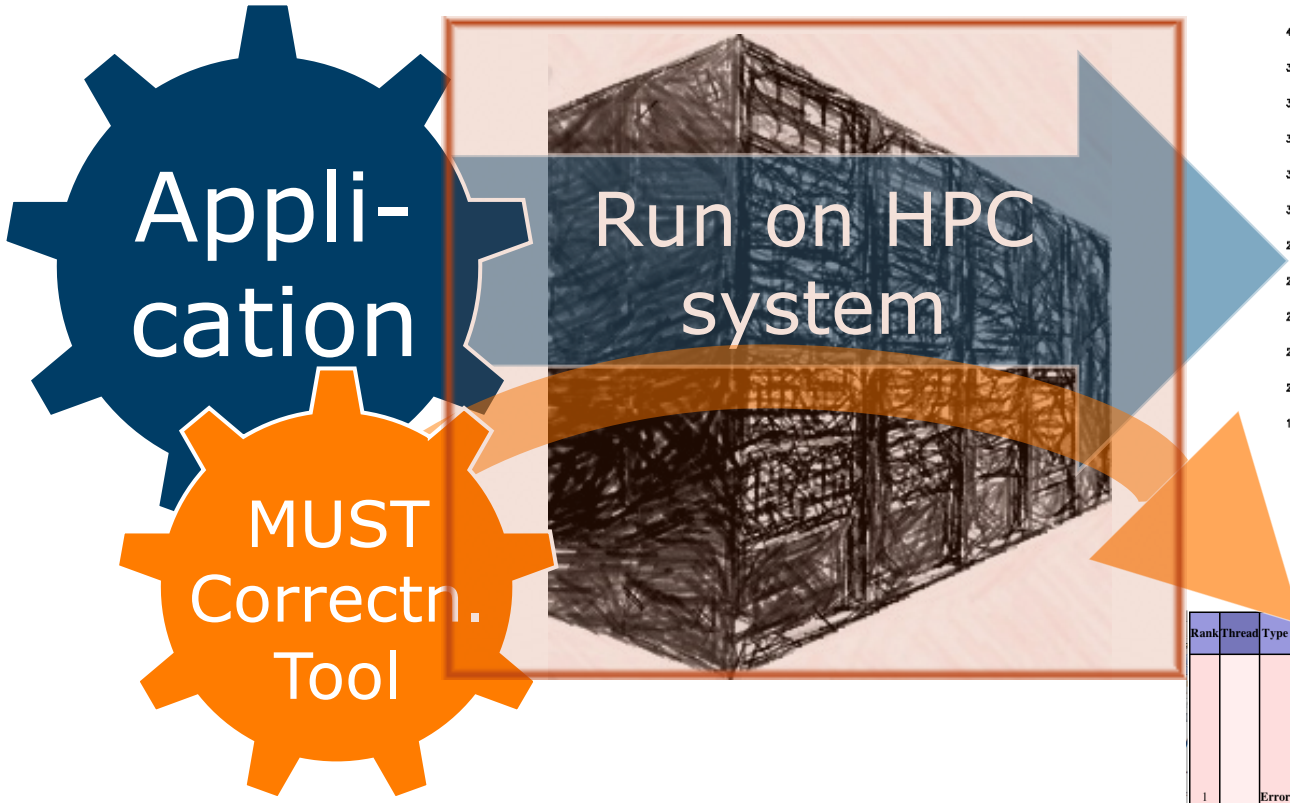
NPB-MZ-MPI / BT Instrumented – Build target executable

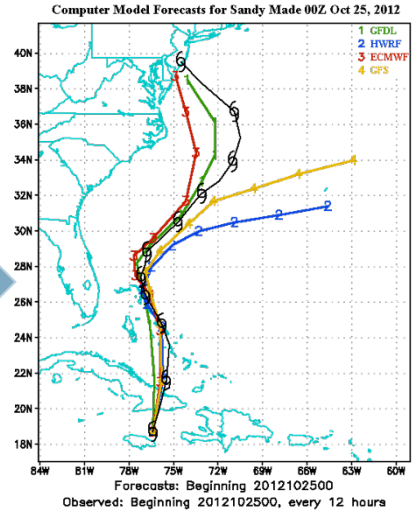
```
% make clean

% make bt-mz CLASS=C NPROCS=8
cd BT-MZ; make CLASS=C NPROCS=8 VERSION=
make: Entering directory 'BT-MZ'
cd ../sys; cc  -o setparams setparams.c -lm
../sys/setparams bt-mz 8 C
mpiifort -c  -O3 -g -openmp bt.f
 [...]
cd ../common; mpiifort -c  -O3 -g -openmp timers.f
mpiifort -O3 -fopenmp -g -o ../bin/bt-mz_C.8 \
bt.o initialize.o exact_solution.o exact_rhs.o set_constants.o \
adi.o rhs.o zone_setup.o x_solve.o y_solve.o exch_qbc.o \
solve_subs.o z_solve.o add.o error.o verify.o mpi_setup.o \
../common/print_results.o ../common/timers.o
Built executable ../bin/bt-mz_C.8
make: Leaving directory 'BT-MZ'
```

- Clean-up
- Re-build executable with NPB build system (this is unrelated to MUST and simply part of the NPB benchmarks)

Overview – Next: Run with MUST





Results

Rank	Thread	Type	Message	From	References	MPI-Standard Reference
1		Error	A send and a receive operation use datatypes that do not match! Missmatch occurs at (CONTIGUOUS)[0] (MPI_INT) in the send type and at (MPI_BYTE) in the receive type (consult the MUST manual for a detailed description of datatype positions). The send operation was started at reference 1, the receive operation was started at reference 2. (Information on communicator: MPI_COMM_WORLD) (Information on send of count 1 with type:Datatype created at reference 3 is for C, commited at reference 4, based on the following type(s): { MPI_INT}Typemap = {(MPI_INT, 0), (MPI_INT, 4)}) (Information on receive of count 8 with type:MPI_BYTE)	**MPI_Sendrecv** from: #0 main@test.c:54 #1 start_main@libc.so	reference 1 rank 0: **MPI_Sendrecv** from: #0 main@test.c:54 #1 start_main@libc.so reference 2 rank 1: **MPI_Sendrecv** from: #0 main@test.c:54 #1 start_main@libc.so reference 3 rank 0: **MPI_Type_contiguous** from: #0 main@test.c:35 #1 start_main@libc.so reference 4 rank 0: **MPI_Type_commit** from: #0 main@test.c:51 #1 start_main@libc.so	

Correctness report

MUST Configuration: Switches to "mustrun"

```
% mustrun --must:help
"mustrun" from MUST v1.4.0
Prepares and runs an application that is linked with P^nMPI for
runtime analysis with the MPI correctness tool MUST.

Replace your regular mpiexec/mpirun command with mustrun, usage:

mustrun [--help|-help|--must:help] [--must:nocrash] [--must:layout <xml>]
        [--must:mode {prepare|run|preparerun}] [--must:mpiexec <COMMAND>]
        [--must:np <NP-SWITCH>] [--must:temp <DIR>] [--must:apis <XMLS>]
        [--must:analyses <ANALYSES-XMLS>] [--must:verbose] [--must:quiet]
        <MPIRUNARGS> <COMMAND> <ARGS>

--help,-help,--must:help:
         Prints this information.
  [... More configuration variables ...]
```

- MUST is controlled via mustrun

```
% mustrun --must:info -np 4
[MUST] MUST configuration ... centralized checks with fall-back
                              application crash handling (very slow)
[MUST] Required total number of processes ... 5
[MUST] Number of application processes ... 4
[MUST] Number of tool processes ... 1
[MUST] Tool layers sizes ... 4:1
```

- Use "--must:info" to correctly allocate in batch jobs

NPB-MZ-MPI / BT – Run with MUST

- Change to the directory containing the new executable (bin)

```
% cd bin
% cp ../jobscript/{archer,cosma,hamilton}/must.* ./
% nano must.*
...
export OMP_NUM_THREADS=4
...
mustrun --must:mpiexec  aprun -n $NPROCS \
        -d $OMP_NUM_THREADS $EXE
```

4 instead of 6 makes room for the extra MUST process

- Change to directory with the executable and edit the job script

`% bsub -q bench1 -P durham < must.lsf`	COSMA
`% sbatch -p bench2 -A bench must.sbatch`	Hamilton
`% qsub must.pbs`	Archer

- Submit the job

NPB-MZ-MPI / BT – Run with MUST, Output

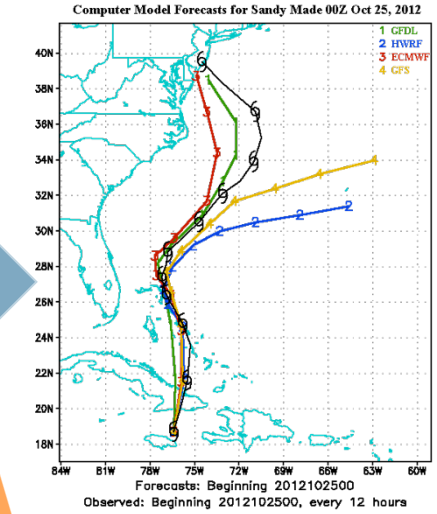
```
% less <Jobscript/Shell-Output>
[MUST] MUST configuration ... centralized checks with …
[MUST] Information: overwritting old intermediate data …
[MUST] Weaver ... success
[MUST] Code generation ... success
[MUST] Build file generation ... success
[MUST] Configuring intermediate build ... success
[MUST] Building intermediate sources ... success
[MUST] Installing intermediate modules ... success
[MUST] Generating P^nMPI configuration ... success
[MUST] Search for linked P^nMPI ... not found ... using …
[MUST] Executing application:

 NAS Parallel Benchmarks (NPB3.3-MZ-MPI) - BT-MZ MPI+OpenMP \
>Benchmark

 Number of zones:   8 x   8
 Iterations: 200    dt:   0.000300
 Number of active processes:     8
  ...
```

- Check the output of the application run
- The 1.4.0 release may report an "no DOT-NOTFOUND" issue at the end, its only a cosmetic issue

Overview – Next: Investigate the correctness report



Results

Rank	Thread	Type	Message	From	References	MPI-Standard Reference
1		Error	A send and a receive operation use datatypes that do not match! Missmatch occurs at (CONTIGUOUS)[0](MPI_INT) in the send type and at (MPI_BYTE) in the receive type (consult the MUST manual for a detailed description of datatype positions). The send operation was started at reference 1, the receive operation was started at reference 2. (Information on communicator: MPI_COMM_WORLD) (Information on send of count 1 with type:Datatype created at reference 3 is for C, commited at reference 4, based on the following type(s): { MPI_INT}Typemap = {(MPI_INT, 0), (MPI_INT, 4)}) (Information on receive of count 8 with type:MPI_BYTE)	**MPI_Sendrecv** from: #0 main@test.c:54 #1 start_main@libc.so	reference 1 rank 0: **MPI_Sendrecv** from: #0 main@test.c:54 #1 start_main@libc.so reference 2 rank 1: **MPI_Sendrecv** from: #0 main@test.c:54 #1 start_main@libc.so reference 3 rank 0: **MPI_Type_contiguous** from: #0 main@test.c:35 #1 start_main@libc.so reference 4 rank 0: **MPI_Type_commit** from: #0 main@test.c:51 #1 start_main@libc.so	

Correctness report

NPB-MZ-MPI / BT – MUST Output File

```
% ls
bt-mz_C.8  must_mzmpibt.o2973155 MUST_Output.html must_temp
% firefox MUST_Output.html

      [Browser GUI showing correctness report]
```

- "MUST_Output.html" Reports all correctness violations reported by MUST
- Level of detail depends on availability of Dyninst stack tracing utility (If MUST finds issues)