HANDS-ON Exercices

Andres S. CHARIF-RUBIAL
Jean-Baptiste BESNARD
Emmanuel Oseret

MAQAO PerfEval

Locating function and loop hotspots

Environment Setup

Load required modules :

```
module use /gpfslocal/pub/vihps/UNITE/local/
module load intel openmpi UNITE
module load maqao
```

Get the material in : /gpfslocal/pub/vihps/materials/MAQAO/

```
cp -r /gpfslocal/pub/vihps/materials/MAQAO/ .
```

CD to this directory :

```
cd ./MAQAO/SX/
```

Generating a profile

/gpfslocal/pub/vihps/materials/MAQAO/SX

> maqao perf – – [APP] [ARGS] […]
This will generate a default maqao_... Folder
OR
> maqao perf xp=experiment_path – – [APP] [ARGS]

When using MPI, prefix the maqao command with mpirun

Display a profile's results

> maqao perf d=SX xp=experiment_path oformat=html

This will generate an html folder in your experiment path
Then you can copy the experiment_path/html/ to your laptop/workstation
Open html/index.html in your favorite browser

Sample Usage (1/2)

Poincare submission script :

```
#!/bin/bash
#@ class           = clint
#@ job_name        = BT-C-MZ
#@ total_tasks     = 4
#@ node            = 1
#@ wall_clock_limit = 00:05:00
#@ output          = $(job_name).$(jobid)
#@ error           = $(job_name).$(jobid)
#@ environment     = COPY_ALL
#@ job_type        = mpich
#@ queue
module load intel
module load openmpi
mpirun maqao perf -t=SX - xp=bt -- ../NPB3.3-MZ-MPI/bin/bt-mz.A.4
```

llsubmit sub_btmz.sh

Locating hotspots with MAQAO perfeval

Display – function hotspots

MAQAO

Performance Evaluation - Profiling results

Hotspots - Functions

Name	Median Excl %Time	Deviation
matmul_sub__ - 56@solve_subs.f	17.16	0.26
compute_rhs__ - 4@rhs.f	10	0.03
y_solve_cell__ - 385@y_solve.f	9.32	0.54
z_solve_cell__ - 385@z_solve.f	8.96	0.14
x_solve_cell__ - 391@x_solve.f	8.68	0.17
MPIDI_CH3I_Progress	5.22	3.66
matvec_sub__ - 5@solve_subs.f	3.92	0.11
x_backsubstitute__ - 330@x_solve.f	3.09	0.14
y_backsubstitute__ - 329@y_solve.f	2.05	0.03
z_backsubstitute__ - 329@z_solve.f	1.98	0.06
copy_faces__ - 4@copy_faces.f	0.88	0.06
MPID_nem_dapl_rc_poll_dyn_opt_	0.74	0.62
MPID_nem_lmt_shm_start_send	0.68	0.06

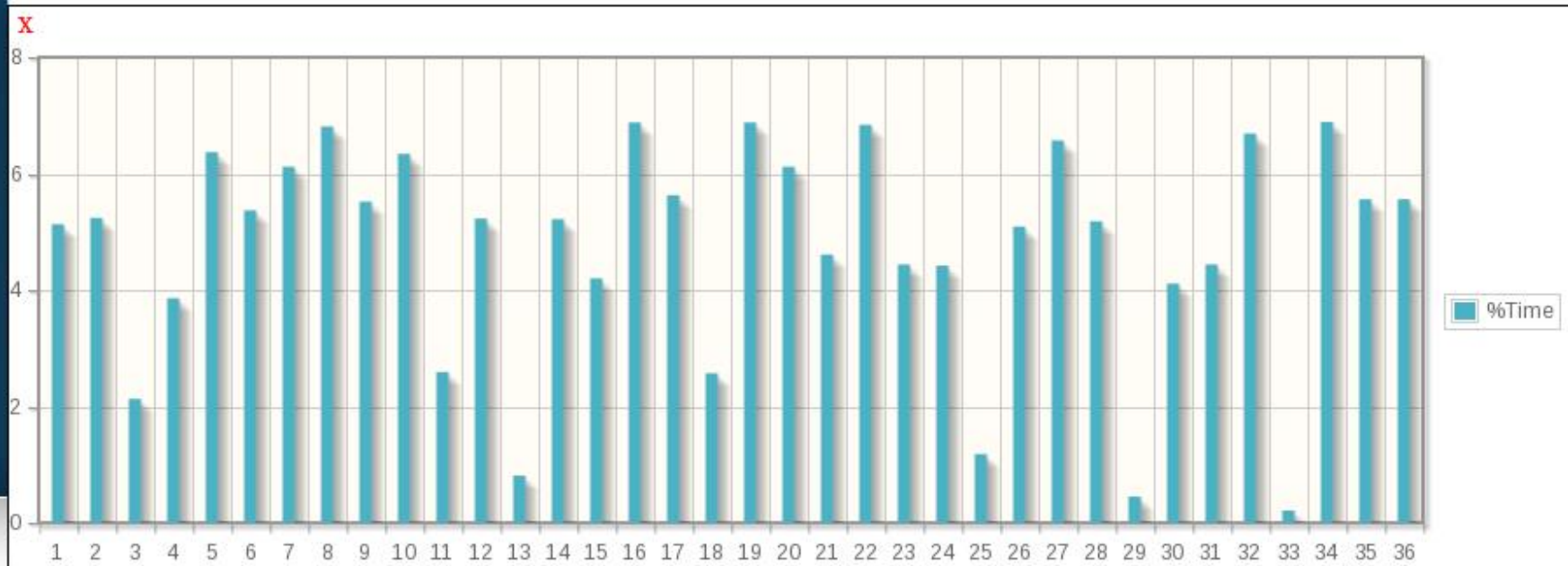
Locating hotspots with MAQAO perfeval

Display – load balancing

MAQAO

Performance Evaluation - Profiling results

Hotspots - Functions



exact_solution__ - 4@exact_solution.f	0.21	0.03
x_unpack_solve_info__ - 114@x_solve.f	0.14	0.03

Locating hotspots with MAQAO perfeval

Display – node hotspots

cirrus5003 - Process #53572 - Thread #1

Name	Excl %Time	Excl Time (s)
matmul_sub__ - 56@solve_subs.f	16.92	16.48
compute_rhs__ - 4@rhs.f	9.92	9.66
y_solve_cell__ - 385@y_solve.f	9.08	8.84
loops	9.08	
Loop 267 - y_solve.f@415	0	
Loop 268 - y_solve.f@425	0	
Loop 272 - y_solve.f@426	0.25	
Loop 270 - y_solve.f@524	6.57	
Loop 271 - y_solve.f@436	2.22	
Loop 269 - y_solve.f@716	0.04	
x_solve_cell__ - 391@x_solve.f	9.01	8.78
loops	9.01	
Loop 235 - x_solve.f@420	0	
Loop 236 - x_solve.f@429	0	
Loop 237 - x_solve.f@709	0.06	
Loop 239 - x_solve.f@431	2.71	
Loop 238 - x_solve.f@519	6.24	

Locating hotspots with MAQAO perfeval

Display – loop hotspots

cirrus5003 - Process #53572 - Thread #1

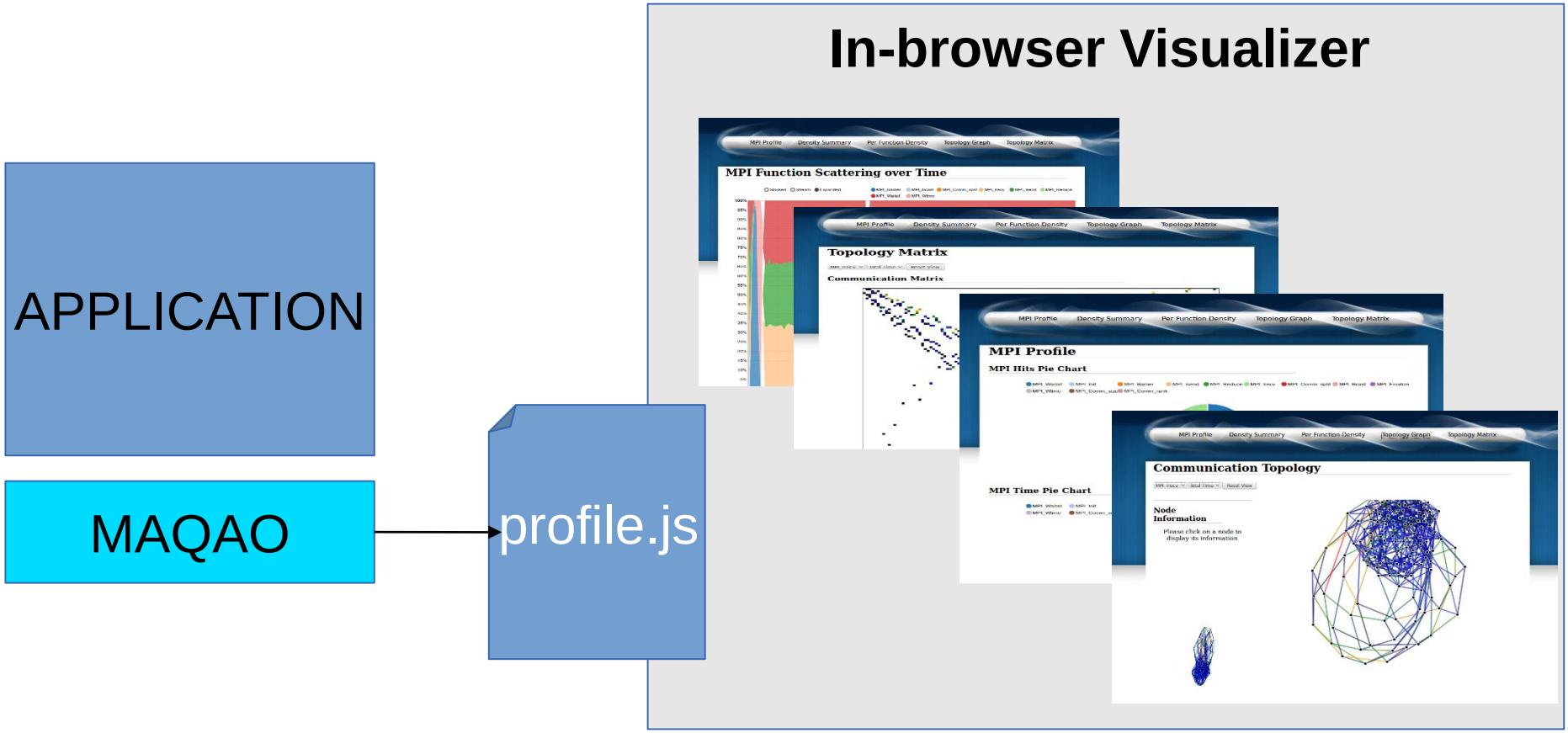
Name	Excl %Time	Excl Time (s)
matmul_sub__ - 56@solve_subs.f	16.92	16.48
compute_rhs__ - 4@rhs.f	9.92	9.66
y_solve_cell__ - 385@y_solve.f	9.08	8.84
loops	9.08	
Loop 267 - y_solve.f@415	0	
Loop 268 - y_solve.f@425	0	
Loop 272 - y_solve.f@426	0.25	
Loop 270 - y_solve.f@524	6.57	
Loop 271 - y_solve.f@436	2.22	
Loop 269 - y_solve.f@716	0.04	
x_solve_cell__ - 391@x_solve.f	9.01	8.78
loops	9.01	
Loop 235 - x_solve.f@420	0	
Loop 236 - x_solve.f@429	0	
Loop 237 - x_solve.f@709	0.06	
Loop 239 - x_solve.f@431	2.71	
Loop 238 - x_solve.f@519	6.24	

MAQAO PerfEval

MPI characterization

Workflow Overview

Perf/MPI is a simple profiling tool targetting lightweight metrics which can be reduced online (no trace required).



Environment Setup

Load required modules :

```
module use /gpfslocal/pub/vihps/UNITE/local/
module unload intelmpi
module load openmpi
module load maqao/2.1.2-openmpi
```

Get the material in : /gpfslocal/pub/vihps/materials/MAQAO/

```
cp -r /gpfslocal/pub/vihps/materials/MAQAO/ .
```

CD to this directory :

```
cd ./MAQAO/MPI/
```

Sample Usage (1/2)

Poincare submission script for openmpi:

```
#!/bin/bash
#@ class            = clallmds
#@ job_name         = BT-C-MZ
#@ total_tasks      = 128
#@ node             = 16
#@ wall_clock_limit = 00:05:00
#@ output           = $(job_name).$(jobid)
#@ error            = $(job_name).$(jobid)
#@ environment      = COPY_ALL
#@ job_type         = mpich
#@ queue
module load intel
module unload intelmpi
module load openmpi
module load maqao

mpirun maqao perf -t=MPI - xp=bt -- ../NPB3.3-MZ-MPI/bin/bt-mz.C.128
```

llsubmit sub_btmz.sh

Sample Usage (2/2)

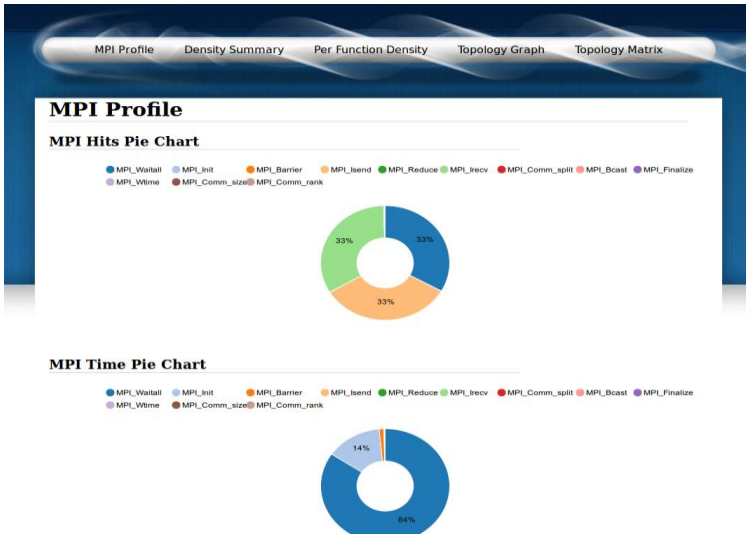
After application ends you can generate the web-viewer with :

maqao perf -d=MPI -xp=**bt**

1 – Copy the result direcory on your station with scp :

scp -r poincare:$HOME/result_dir/ .

2 – Open *index.html* and browse profile results



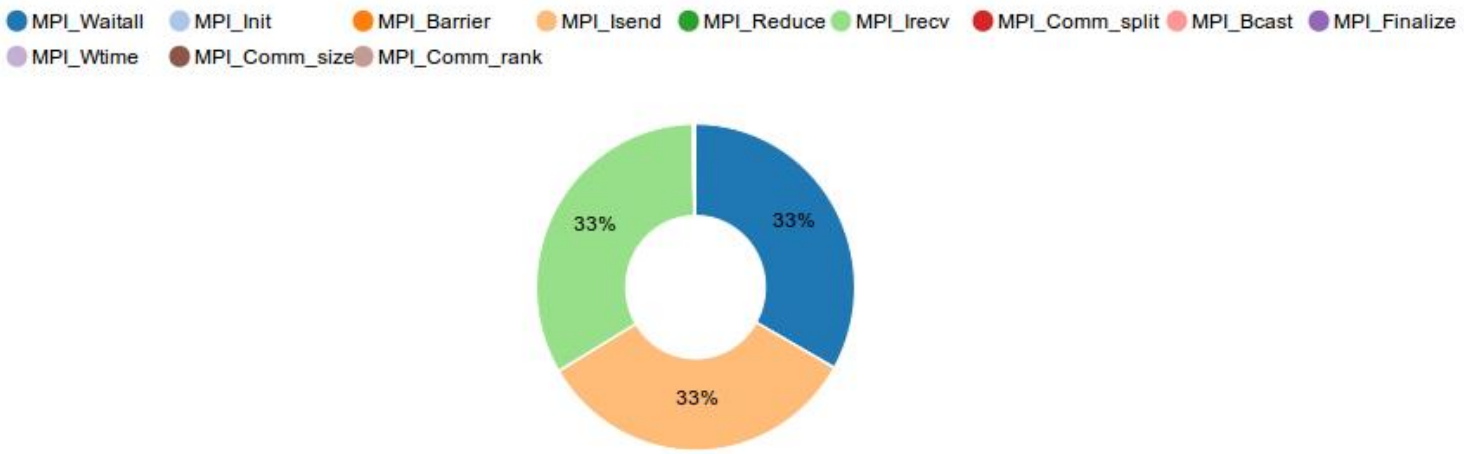
Later on you can retrieve solely the *profile.js* file if you already have the rest of the browser application

Example BT-MZ.C.128
MPI Profile

MPI Profile | Density Summary | Per Function Density | Topology Graph | Topology Matrix

MPI Profile

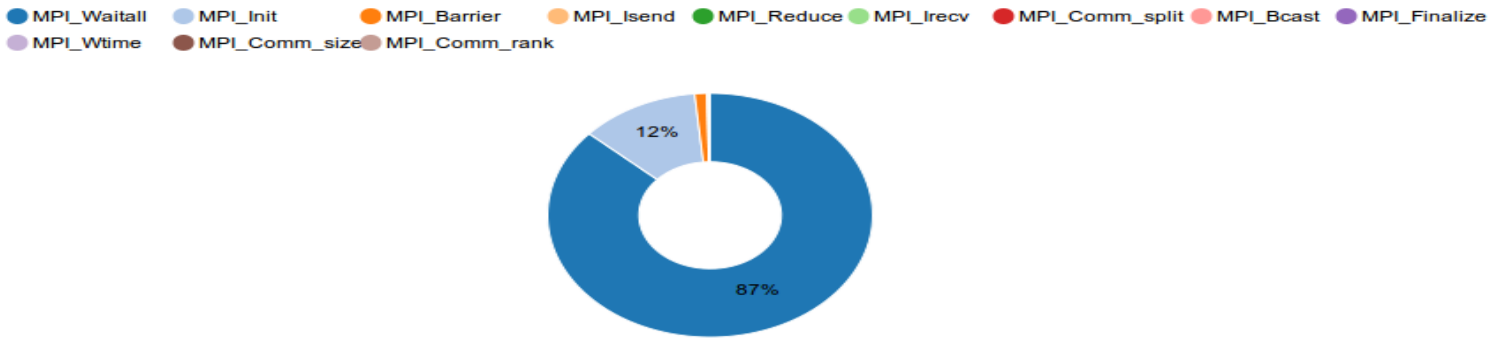
MPI Hits Pie Chart



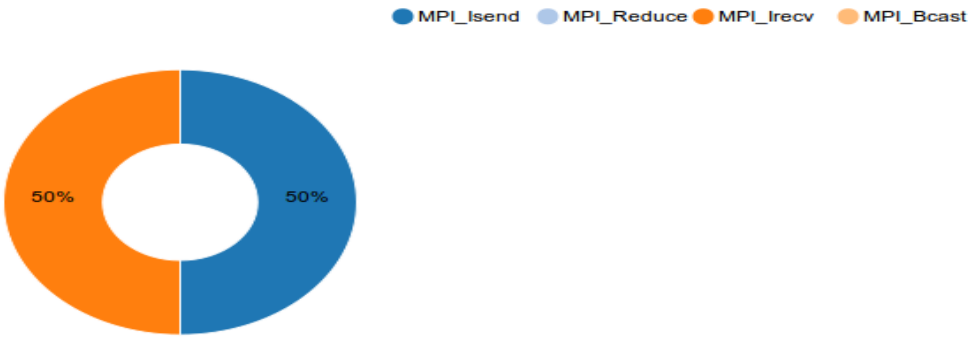
On the first page, you see the pie chart of the dominating MPI calls. Here we have MPI_Irecv, MPI_Isend and MPI_Waitall

Example BT-MZ.C.128
MPI Profile

MPI Time Pie Chart



MPI Size Pie Chart



On the same page, you can see that MPI_Waitall is the most time consuming. Dealing with sizes, MPI_Isend and MPI_Irecv are symetric with 4,568 GB.

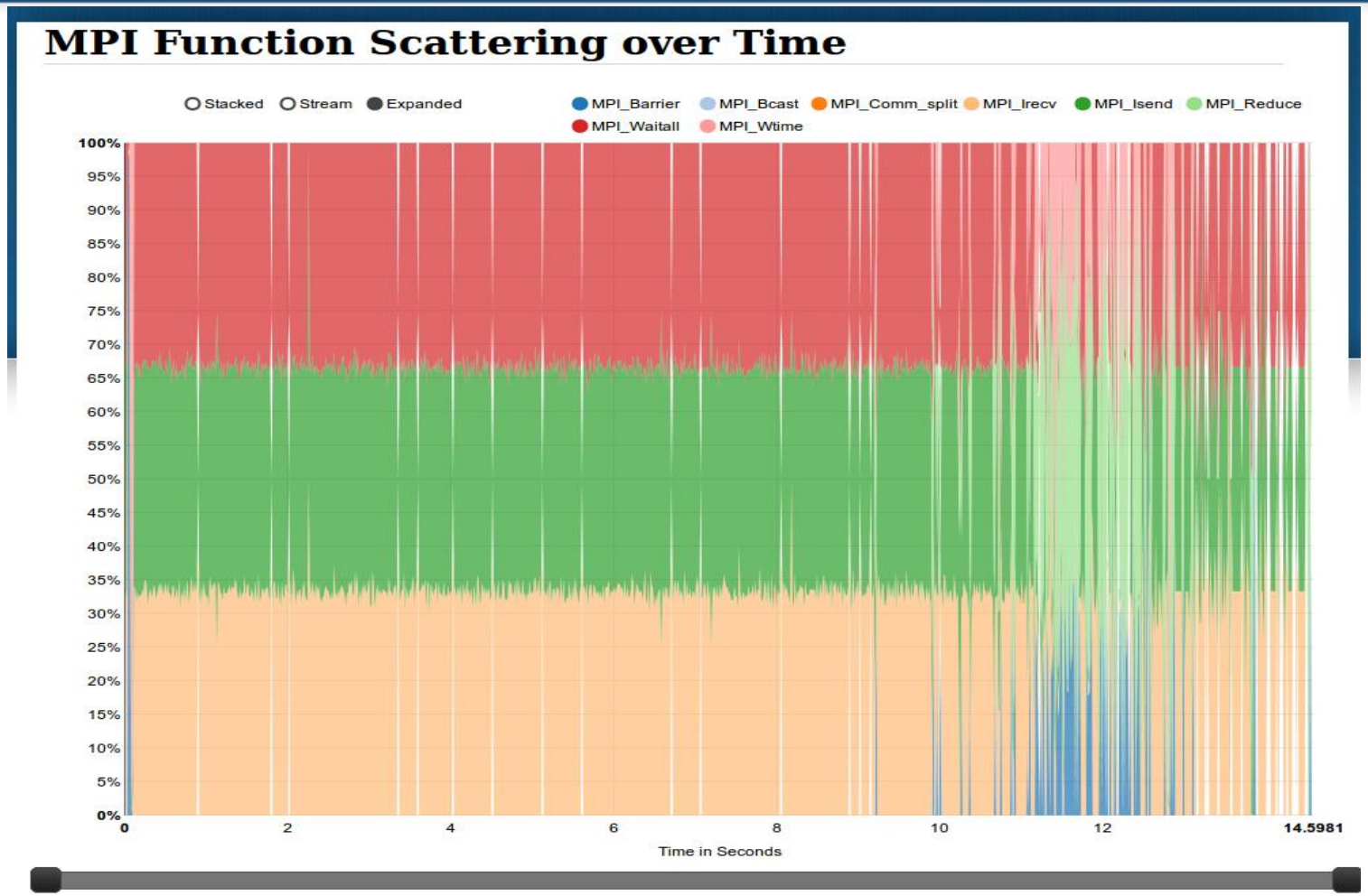
Example BT-MZ.C.128
MPI Profile

MPI Profile

Function	Hits	Time	Size	Walltime %
MPI_Waitall	192960	13 m 1.51 s	0 B	52.333%
MPI_Init	128	1 m 46.60 s	0 B	7.138%
MPI_Barrier	256	10.88 s	0 B	0.729%
MPI_Isend	192960	1.47 s	4.568 GB	0.098%
MPI_Reduce	384	5.36e-1 s	11.000 KB	0.036%
MPI_Irecv	192960	4.62e-1 s	4.568 GB	0.031%
MPI_Comm_split	128	4.05e-1 s	0 B	0.027%
MPI_Bcast	1152	3.12e-2 s	132.000 KB	0.002%
MPI_Finalize	128	2.07e-3 s	0 B	0.000%
MPI_Wtime	256	3.53e-4 s	0 B	0.000%
MPI_Comm_size	128	1.30e-4 s	0 B	0.000%
MPI_Comm_rank	256	4.28e-5 s	0 B	0.000%

At the bottom of the page you get a classical MPI profile in hits, total time, total size and walltime %.
As expected MPI_Waitall dominates for 52,33 % of total exec time.

Example BT-MZ.C.128
Density Summary

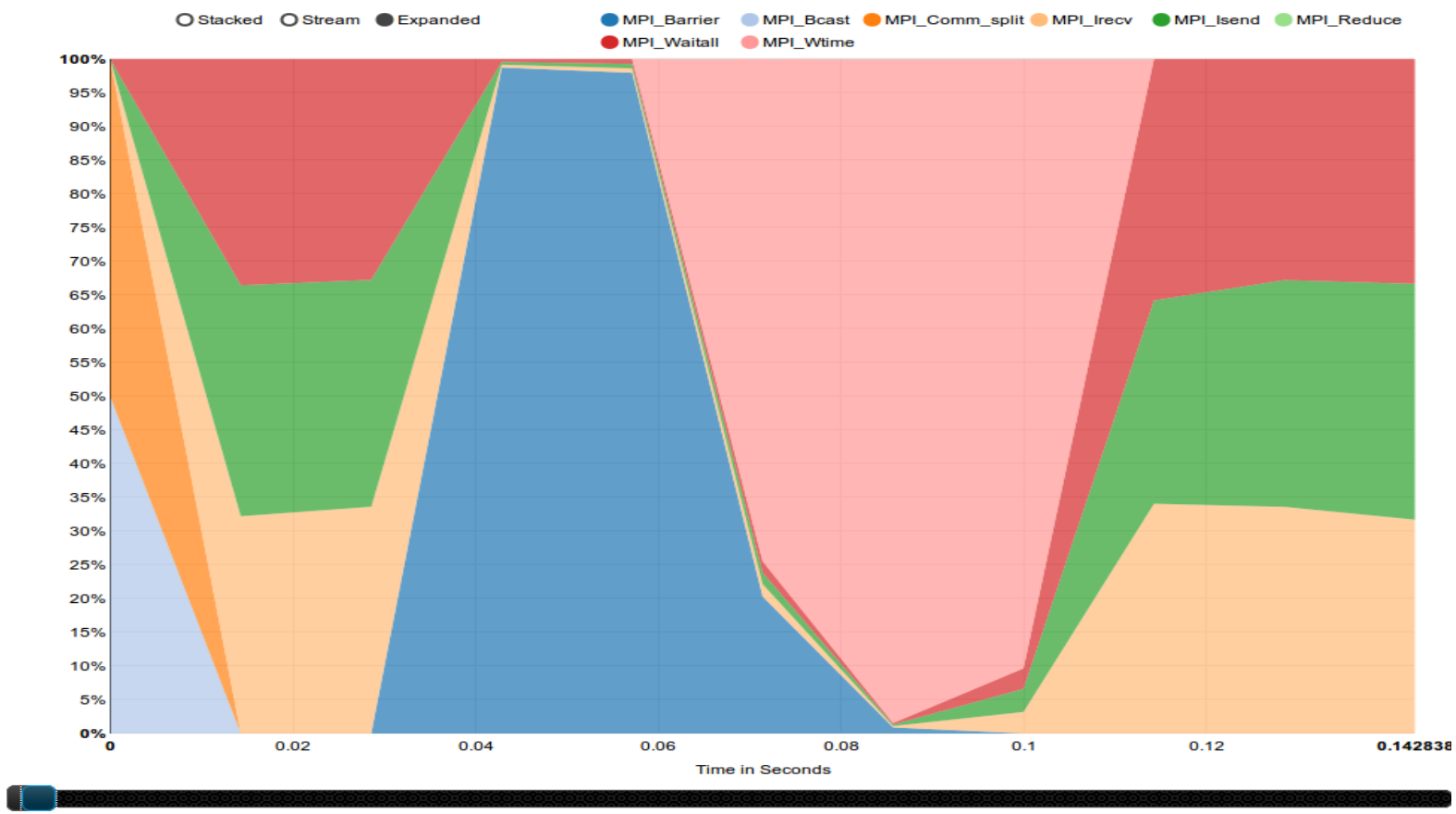


See when MPI calls are occurring during the execution.

Here we have MPI_Waitall, MPI_Isend & MPI_Irecv dominating.

But what if we zoom on the beginning for example ?

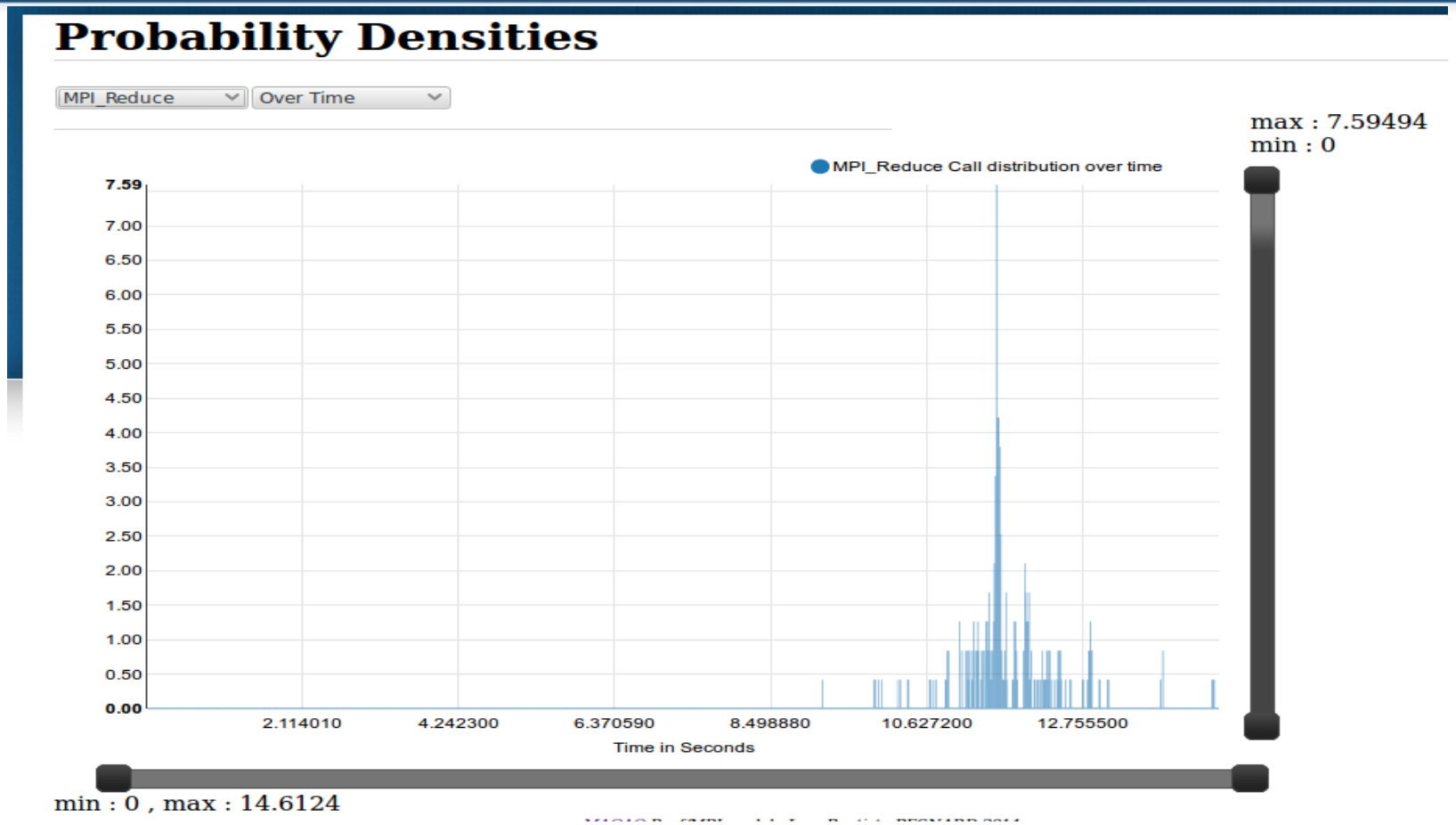
Example BT-MZ.C.128
Density Summary



min : 0 , max : 0.14612

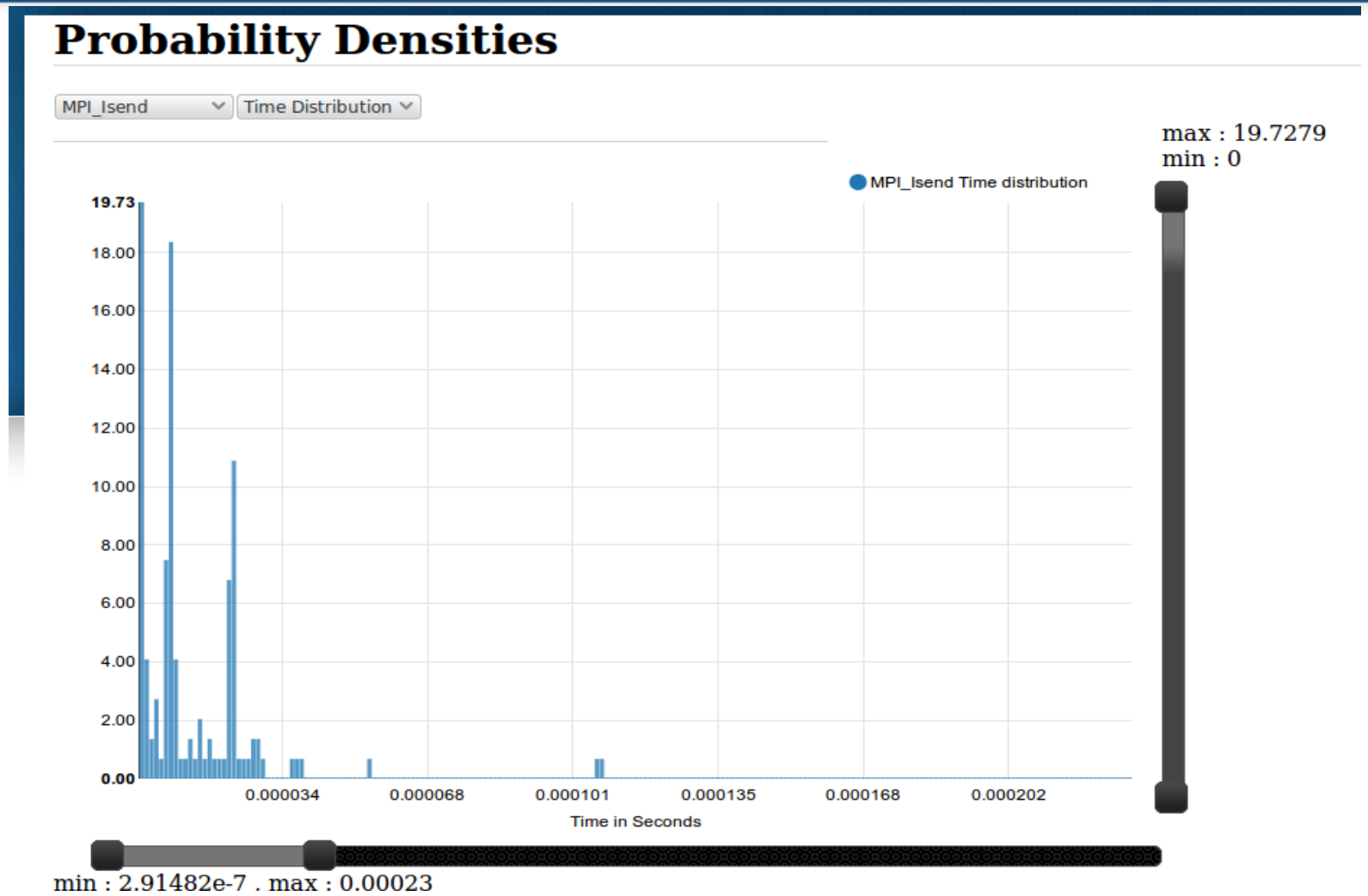
In the first 0,14 seconds we have : MPI_Bcast followed by MPI_Comm_split. A Small communication phase and an MPI_Barrier. Eventuallly, processes call MPI_Wtime.

Example BT-MZ.C.128
Density



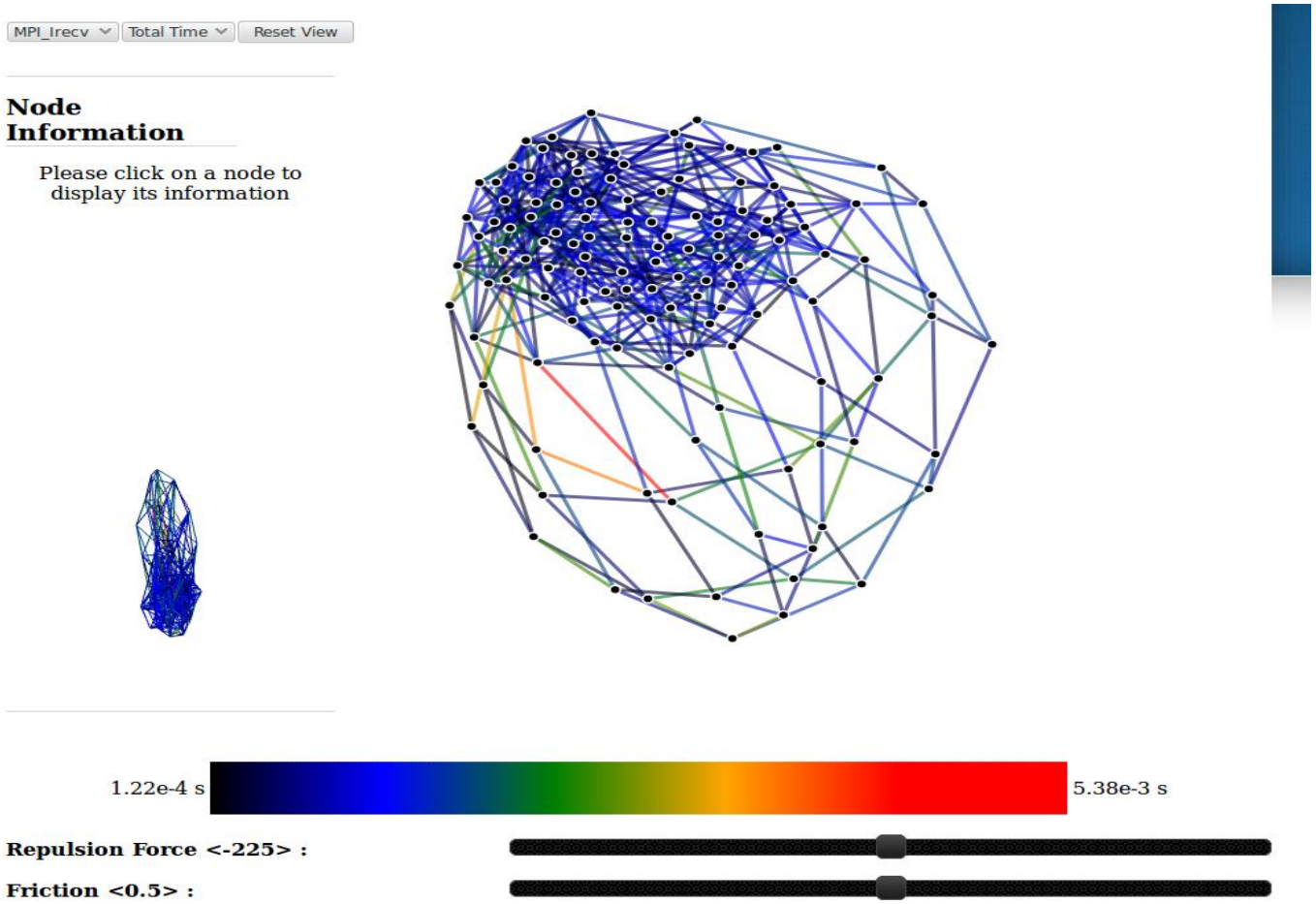
MPI_Reduce calls are located mostly at the end of the execution

Example BT-MZ.C.128
Density



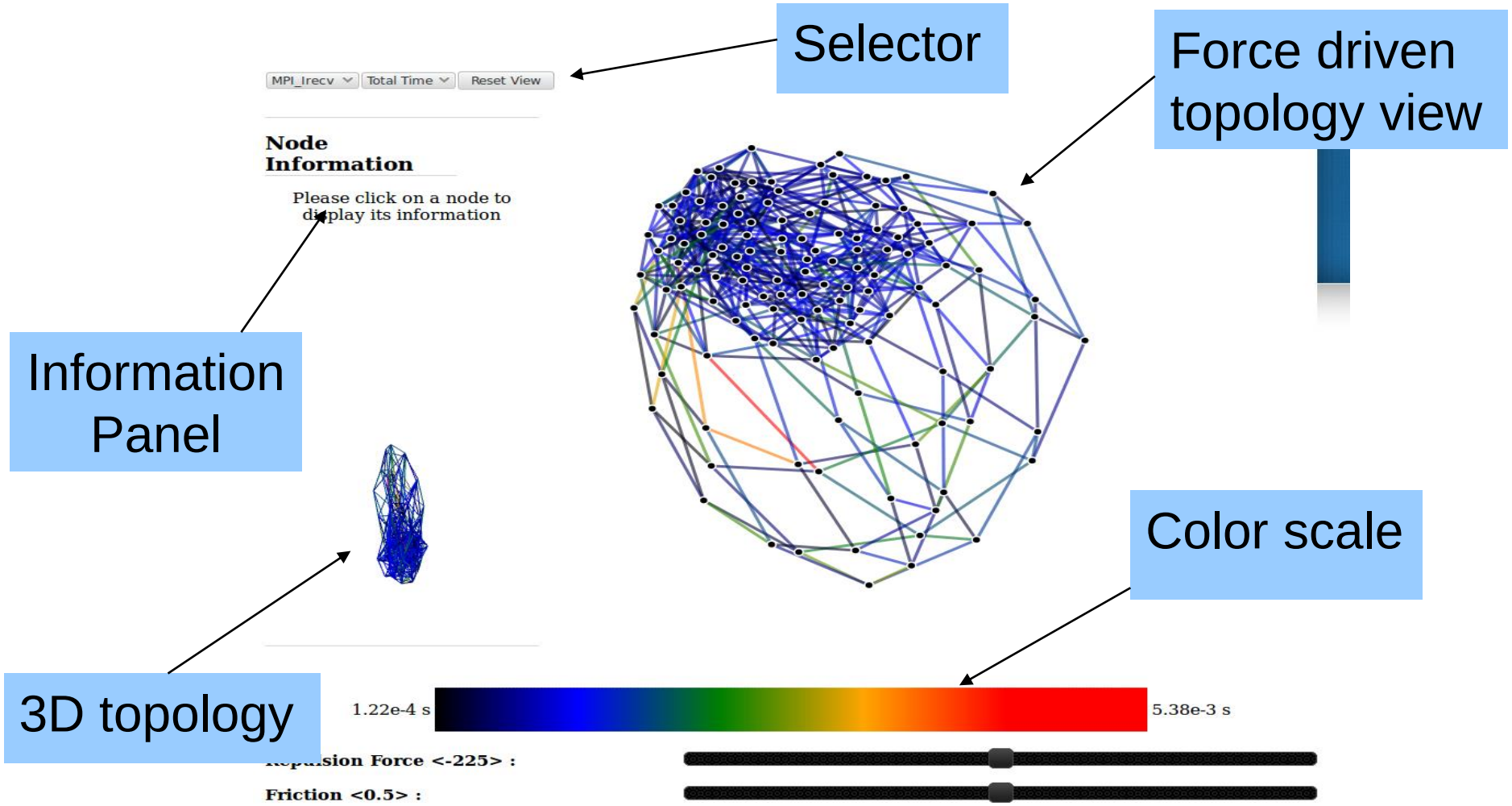
Distribution for MPI_Isend duration (19,73 % are in the 1e-6 range)

Example BT-MZ.C.128
Topology



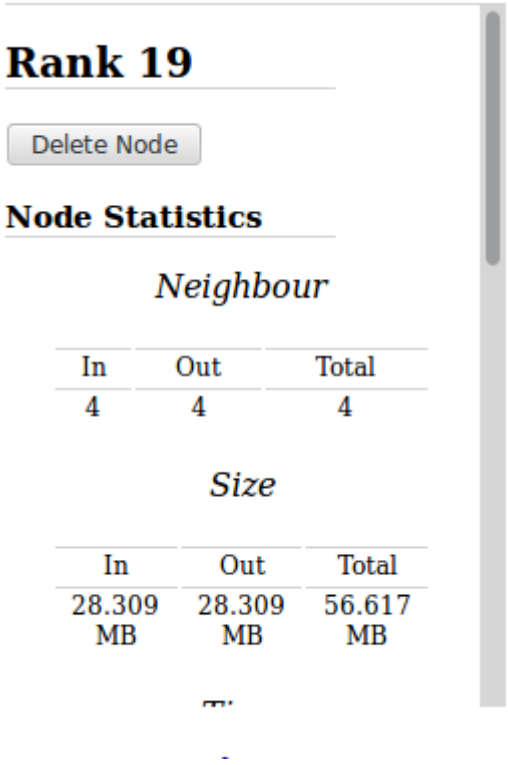
View the topology and observe its regularity. Project time, hits and size metric over it in order to observe imbalances

Example BT-MZ.C.128
Topology

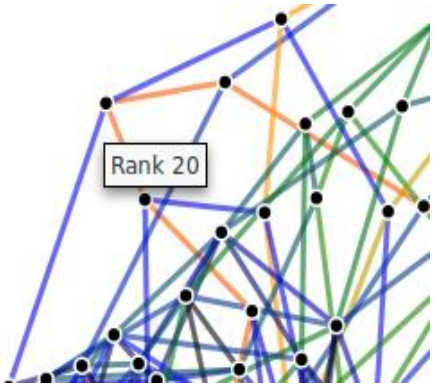


Have a look at the topology and play with it =)

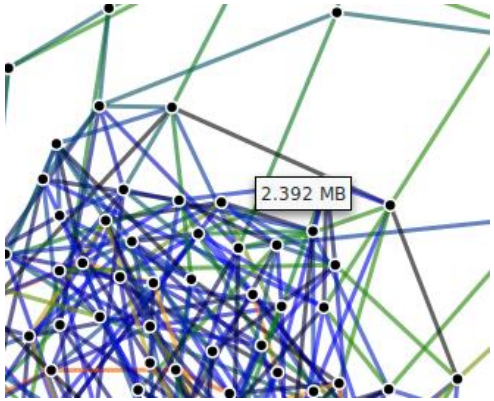
Example BT-MZ.C.128
Topology



Click on a node in the force layout to display its information



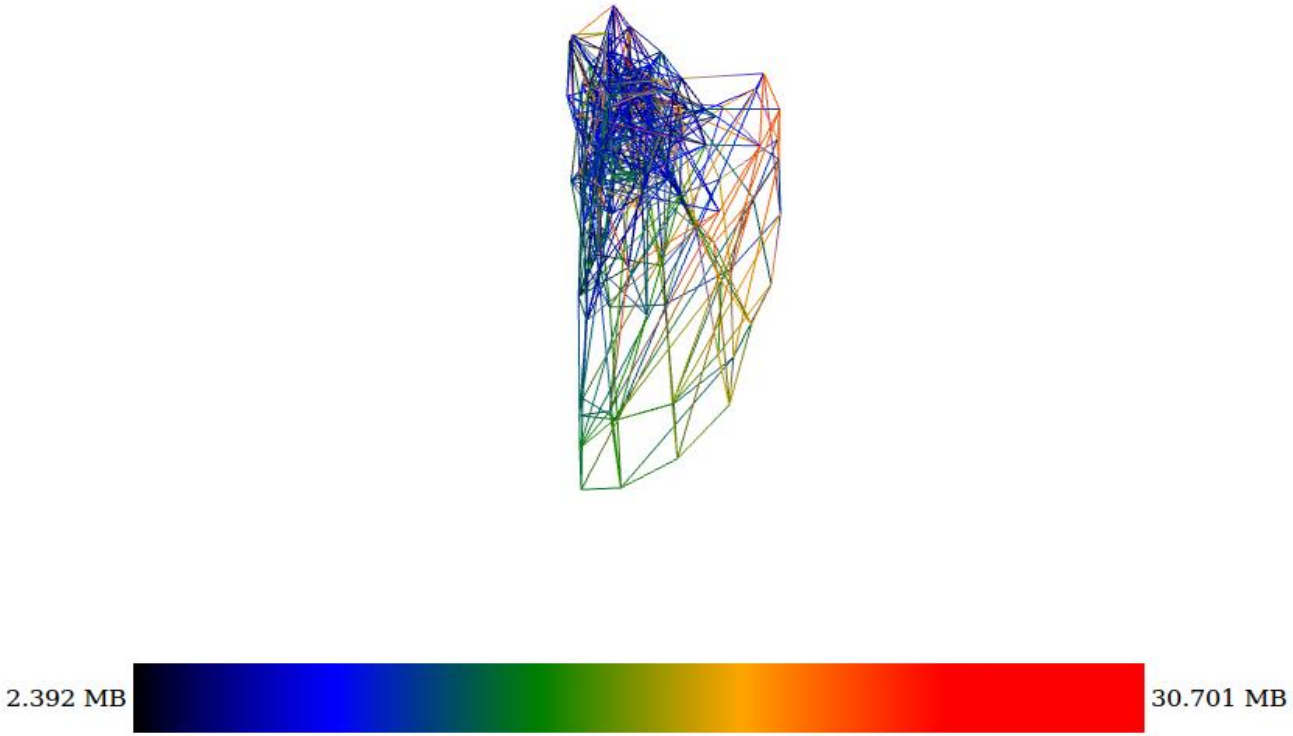
Hover a node to see its MPI rank



Hover an edge to see its value

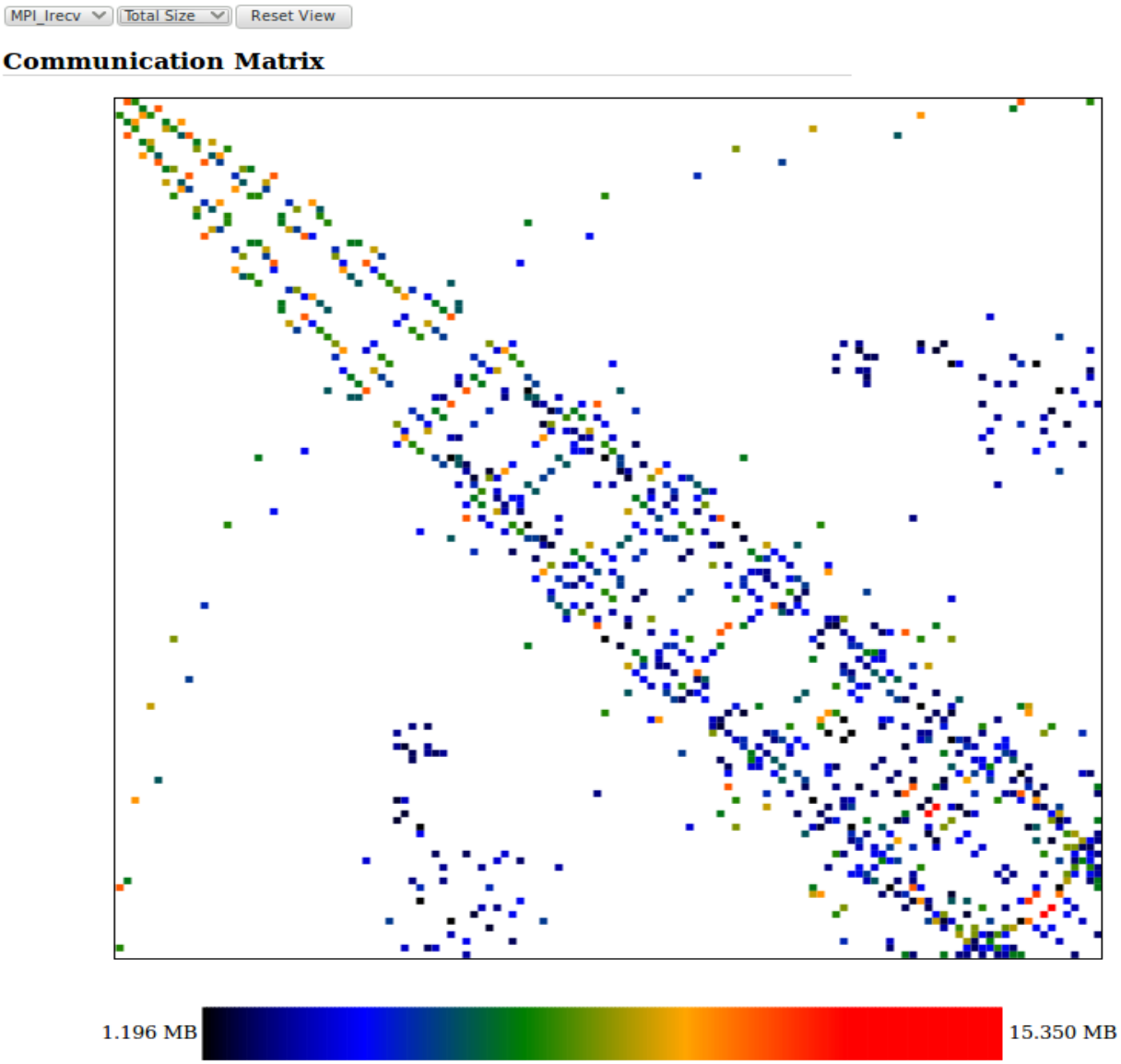
Example BT-MZ.C.128
Topology

Communication Graph



Click on the 3D topology to see it rotating at full scale.

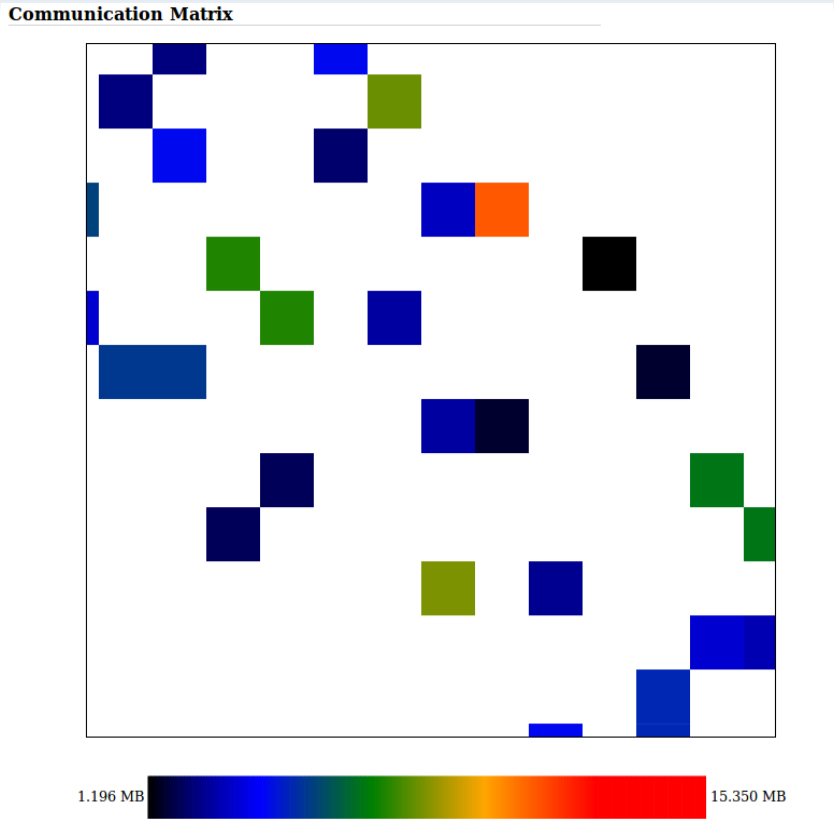
Example BT-MZ.C.128
Communication matrix



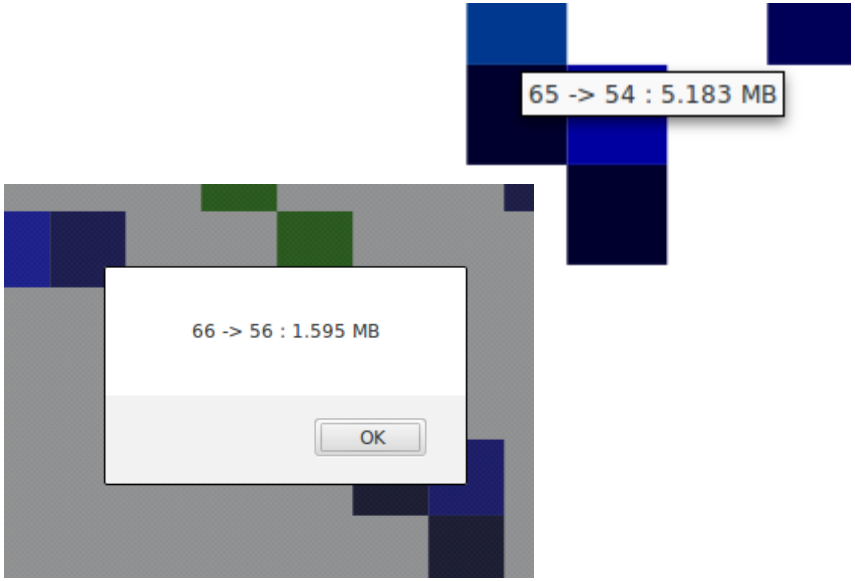
Observe communication balancing in hits, time and size.

See if there is no lines (all to one) or imbalances in the communication scheme.

Example BT-MZ.C.128
Communication matrix



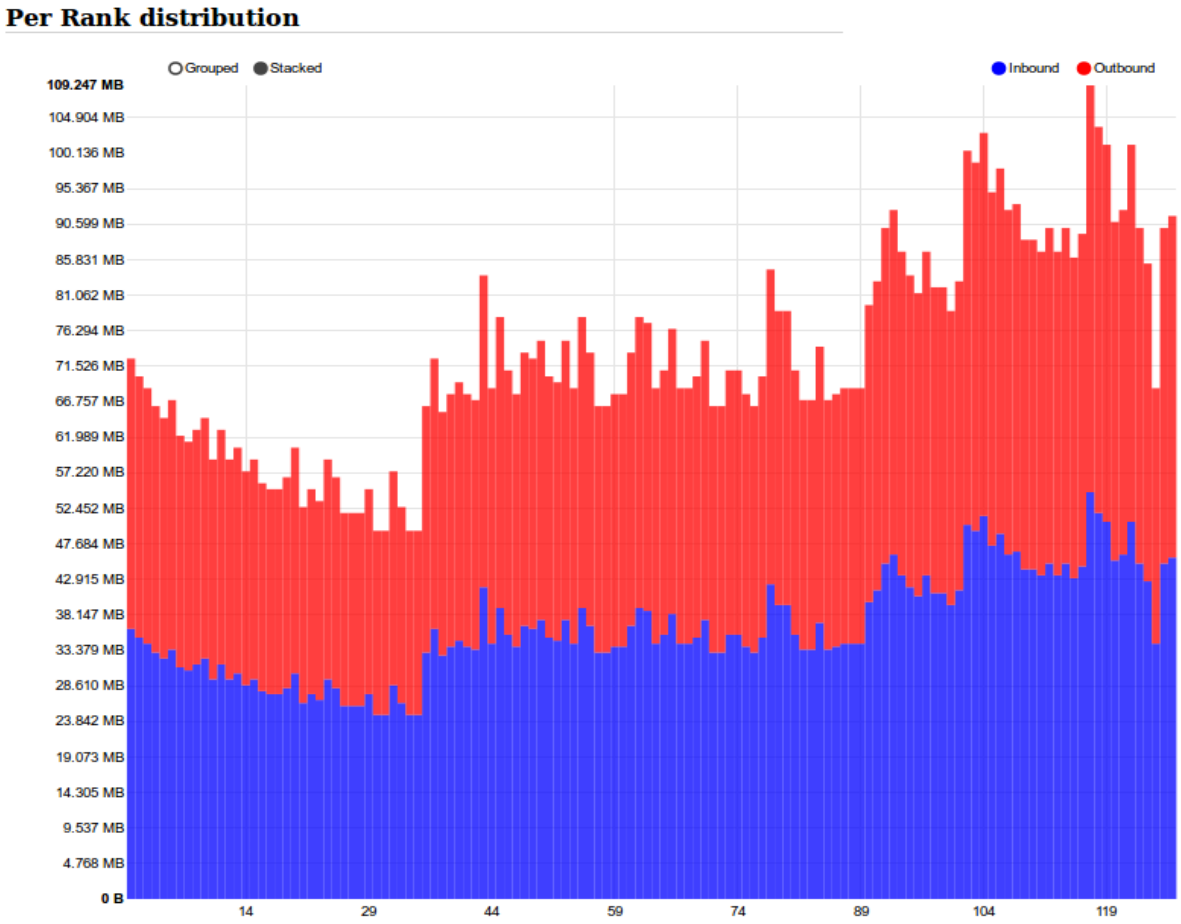
Zoom the matrix with mouse wheel, drag the view to explore it.



Hover or click on a square to display its informations :

Source → Dest : Metric

Example BT-MZ.C.128
Communication matrix

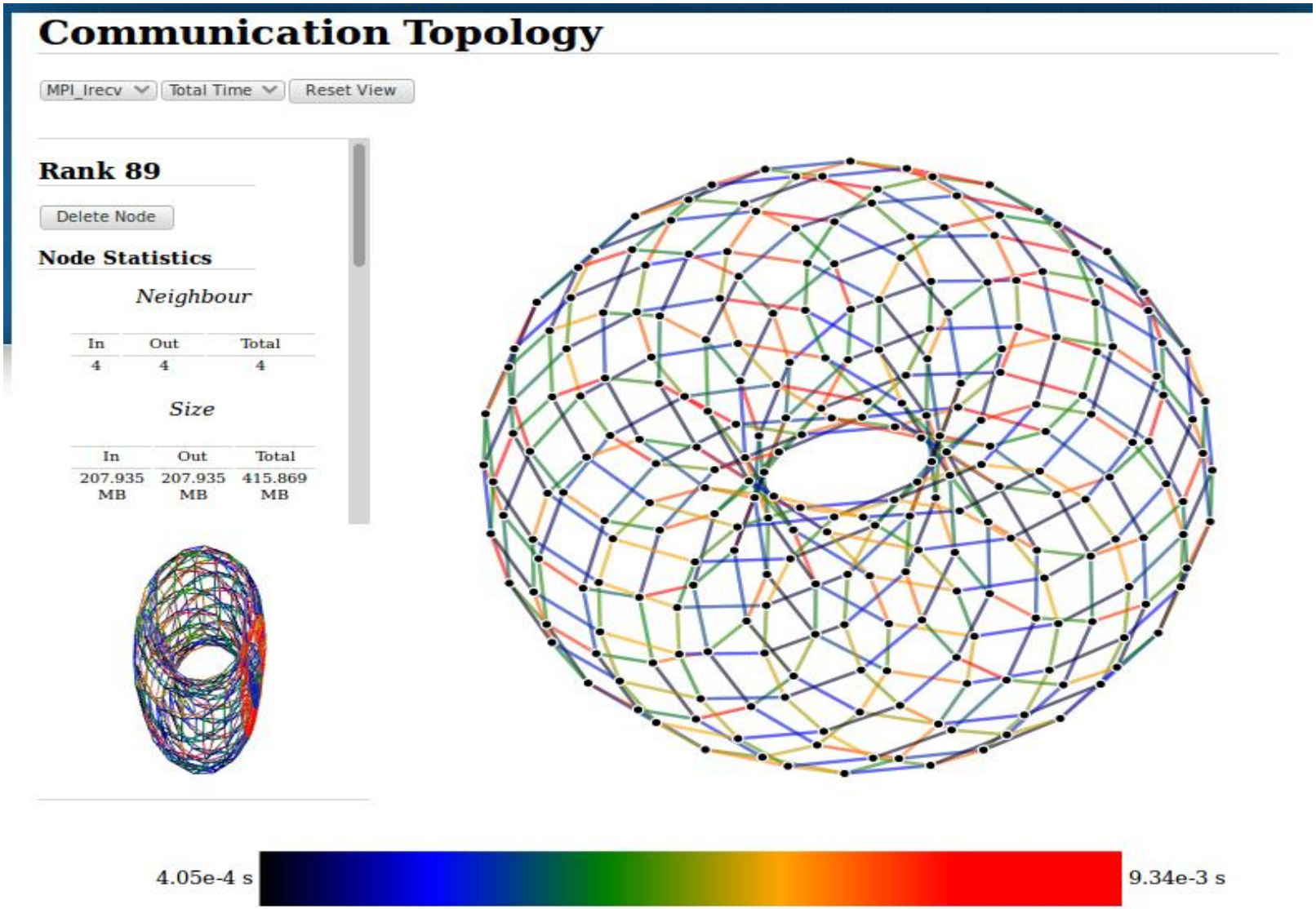


On the same page under the communication matrix, you can observe point to point imbalance over MPI ranks in terms of hits time and size.

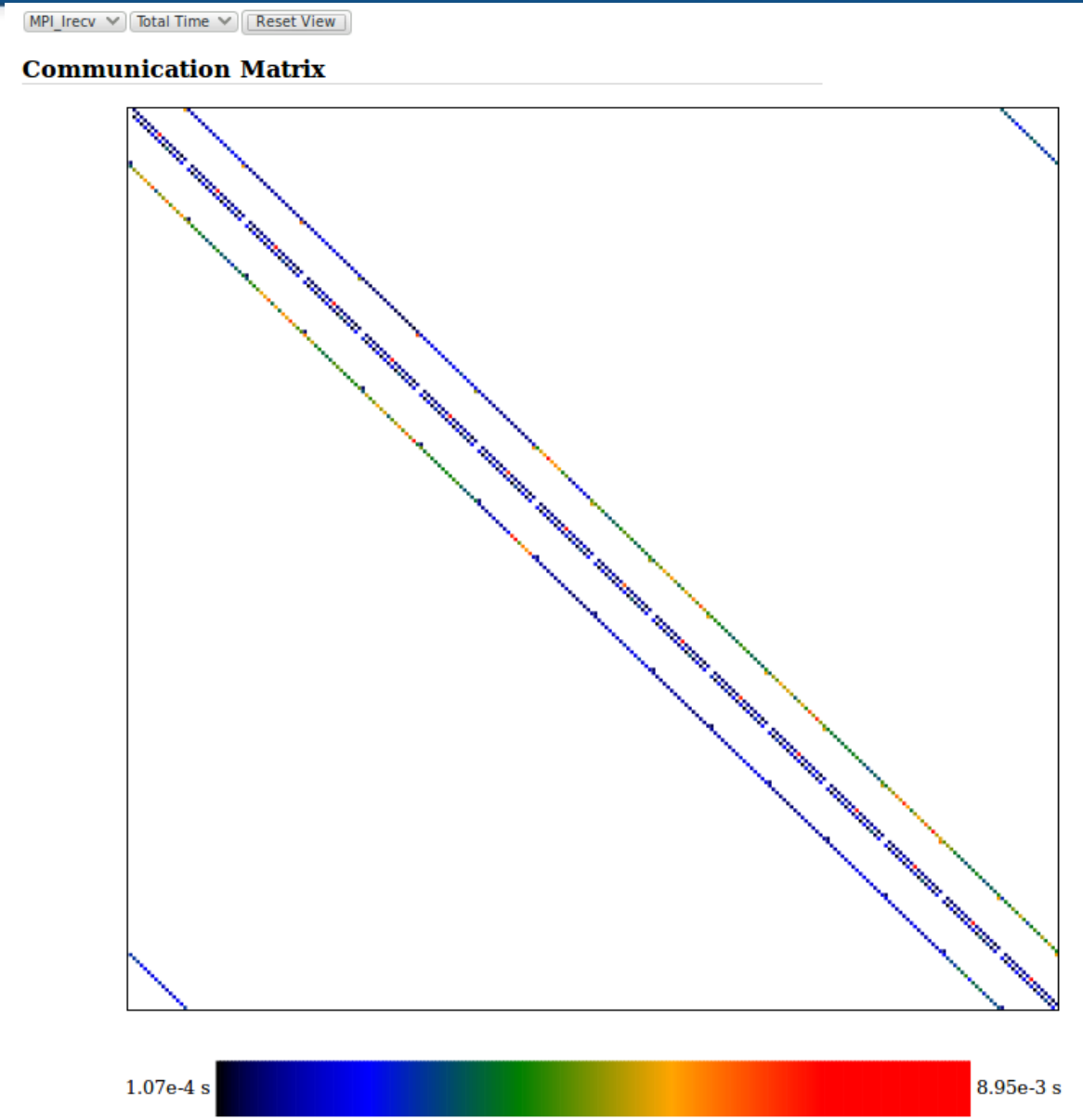
Additional Example : *SP-MZ.D.256*

If you want to look at a larger profile (SP-MZ.D.256) copy **profile.js** from**/gpfslocal/pub/vihps/materials/MAQAO/PERF/sp-d-256 and** replace your profile.js with it.

Sample Outputs
SP-MZ.D.256



Sample Outputs
SP-MZ.D.256



MAQAO CQA

Code Quality Analysis

Analyzing loops

/gpfslocal/pub/vihps/materials/MAQAO/

> maqao cqa loop=X,Y,Z of=html [BIN]
This will generate a default cqa_html folder

Display results

Then you can copy the cqa_html to your laptop/workstation
Open index.html in your favorite browser

Evaluating loops' code quality with MAQAO CQA

Display – gain confidence level reports

MAQAO

Code quality analysis

Source loop ending at line 682

MAQAO binary loop id: 238

The loop is defined in MPI/BT/x_solve.f:519-682

15% of peak computational performance is used (1.23 out of 8.00 FLOP per cycle (GFLOPS @ 1GHz))

Gain | Potential gain | Hints | Experts only

Vectorization

Your loop is processing FP elements but is NOT OR PARTIALLY VECTORIZED and could benefit from full vectorization.
By fully vectorizing your loop, you can lower the cost of an iteration from 190.00 to 60.75 cycles (3.13x speedup).

Since your execution units are vector units, only a fully vectorized loop can use their full power.

Proposed solution(s):

Two propositions:
- Try another compiler or update/tune your current one:
- Remove inter-iterations dependences from your loop and make it unit-stride.

Bottlenecks

By removing all these bottlenecks, you can lower the cost of an iteration from 190.00 to 143.00 cycles (1.33x speedup).

Source loop ending at line 734

Evaluating loops' code quality with MAQAO CQA
Display – Hints confidence level

MAQAO

Code quality analysis

Source loop ending at line 682

MAQAO binary loop id: 238

The loop is defined in MPI/BT/x_solve.f:519-682

15% of peak computational performance is used (1.23 out of 8.00 FLOP per cycle (GFLOPS @ 1GHz))

Gain | Potential gain | **Hints** | Experts only

Type of elements and instruction set

234 SSE or AVX instructions are processing arithmetic or math operations on double precision FP elements in scalar mode (one at a time).

Vectorization status

Your loop is probably not vectorized (store and arithmetical SSE/AVX instructions are used in scalar mode and, for others, at least one is in vector mode).
Only 28% of vector length is used.

Matching between your loop (in the source code) and the binary loop

The binary loop is composed of 234 FP arithmetical operations:
- 95: addition or subtraction
- 139: multiply

The binary loop is loading 1600 bytes (200 double precision FP elements).
The binary loop is storing 616 bytes (77 double precision FP elements).

Arithmetic intensity