

VI-HPS



MALP HANDS-ON

Andres S. CHARIF-RUBIAL
Jean-Baptiste BESNARD

Load maqao

```
> module use gpfs_projects/UNITE/tutorial/mf/UNITE/  
> module load UNITE  
> module load maqao  
  
> maqao -v
```

maqao 2.2.0 - 72a1de037244e84212ff1f3b6a972d8fa87c6d18::[20140208-230640](#)

Copyright (C) Commissariat à l'énergie atomique et aux énergies alternatives (CEA)

Copyright (C) Université de Versailles Saint-Quentin-en-Yvelines (UVSQ)

This version of MAQAO embeds the MALP module which has a special CEA/UVSQ license.

The use of the MALP module outside the 15th VI-HPS Workshop is prohibited.

It is only meant for testing purposes.

For more information about the MALP module: contact@maqao.org

=====

Copyright (C) 2004 - 2013 Université de Versailles Saint-Quentin-en-Yvelines (UVSQ)

This program is distributed in the hope that it will be useful, but WITHOUT ANY

WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

Written by The MAQAO Team.

MAQAO PerfEval

Locating function and loop hotspots

~/gpfs_projects/friday_material/maqao/

bt-mz_B.4

Due to ressource limitation we have used a different MPI bt version:

bt.C.36

bt.C.900

Generating a profile

~/gpfs_projects/friday_material/maqao/PerfEval/

> 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



Performance Evaluation - Profiling results

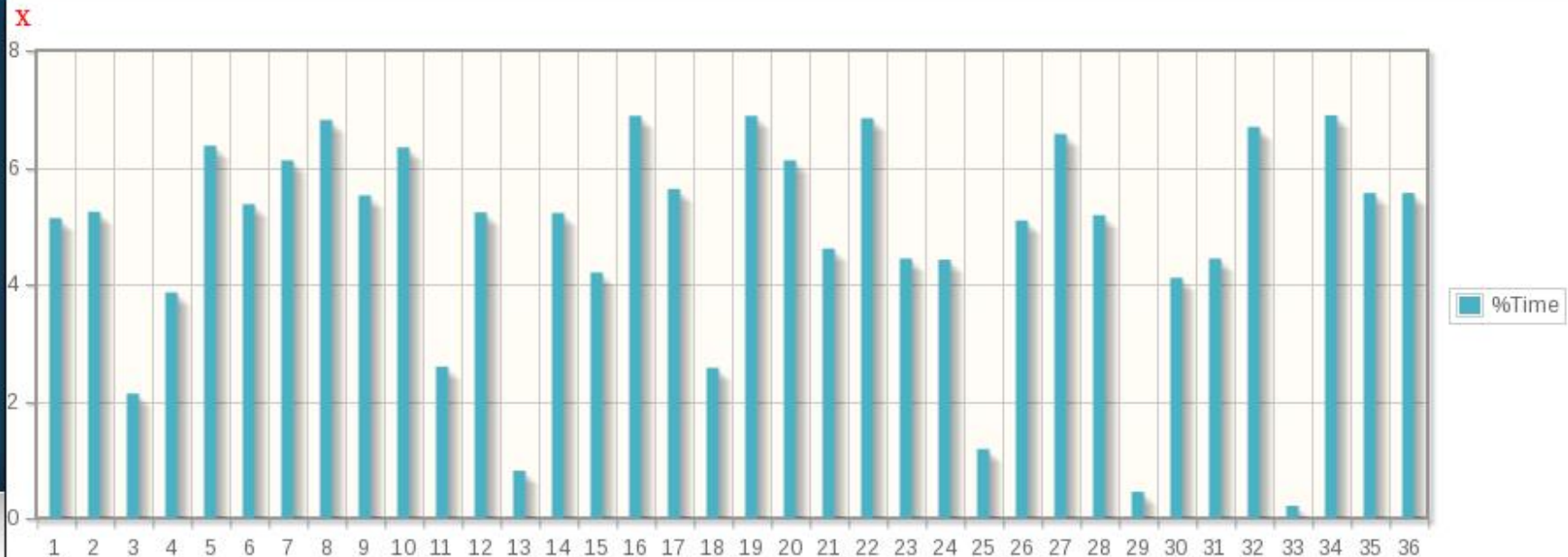
Hotspots - Functions

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_CH3L_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



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 CQA

Code Quality Analysis

Analyzing loops

```
~/gpfs_projects/friday_material/maqao/CQA/
```

```
> 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



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



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

MAQAO MALP

MPI Characterization at scale

Instrument an MPI application

~/gpfs_projects/friday_material/maqao/malp/

Launch your application as usual (mpirun) but prefix it with the maqao command as follows :

> maqao malp -t -- [APP] [ARGS] [...]

This will produce a MALP_trace.json file.

You can configure some environment variables :

- Enable POSIX instrumentation :

TA_CONF_INSTRUMENT_POSIX = (0 or 1) , default 0

- Change the analyzer ratio :

VMPI_RATIO = (0.1 .. 0.5) , default 0.5

You can copy maqao to your local machine in order to facilitate the viewing.

Optionnal : graphviz with GTS is required for graph viewing. We installed it on MN

Start MALP analysis server

maqao malp -a

By default the server picks a random port and displays it :

Server Is now listening on port 8081...

You can use a defined port with :

MALP_PORT=8080 maqao malp -a

Start MALP analysis server on a trace

maqao malp -a -i=MALP_trace.json

Connect to a remote server

You can connect to a server running on MN :

ssh [mnXXX@mn1.bsc.es](#) -L8080:127.0.0.1:[SERVER PORT]

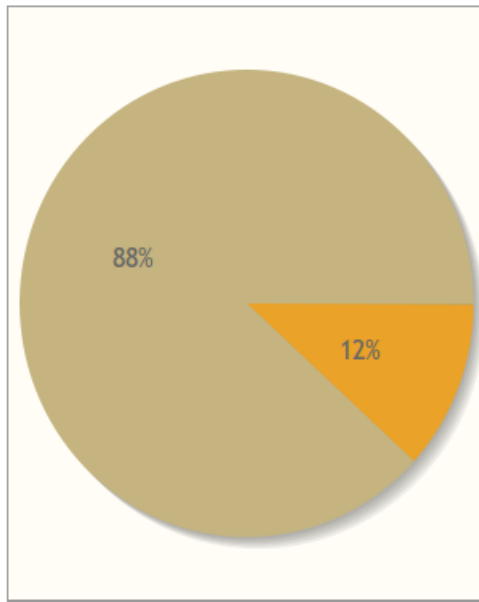
Then open your browser on <http://127.0.0.1:8080>

Work locally (maybe preferable)

Copy Maqao from MN, run the server

Open [http://127.0.0.1:\[SERVER PORT\]](http://127.0.0.1:[SERVER PORT])

Lets have a look at
BT.C.36...

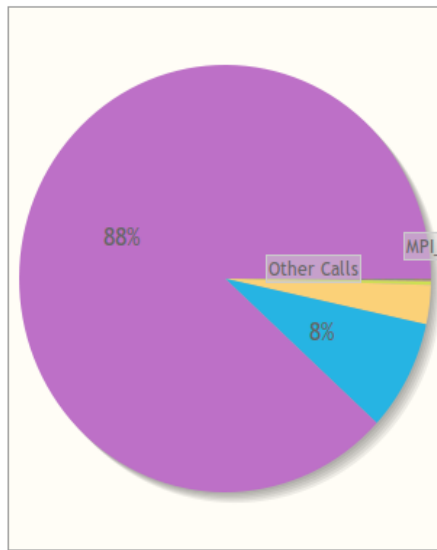


Event Type	Total time	%
Other Calls time	21 m 45.876 s	88.184
MPI time	2 m 54.282 s	11.769
POSIX time	687.287 ms	0.046

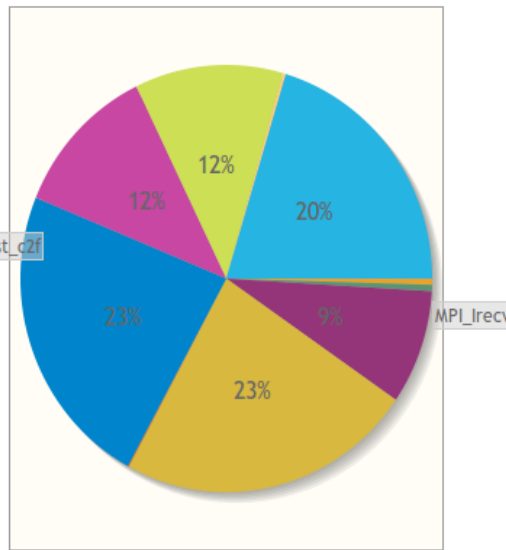
The first metric when opening a trace file is the ratio of POSIX, MPI and « OTHER » time.

It allows a quick understanding of which type of calls is dominating

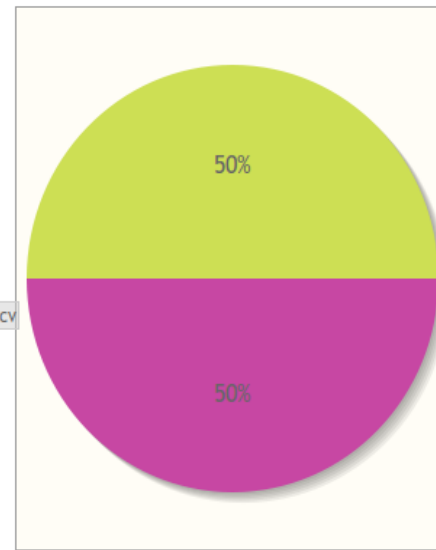
Time



Calls



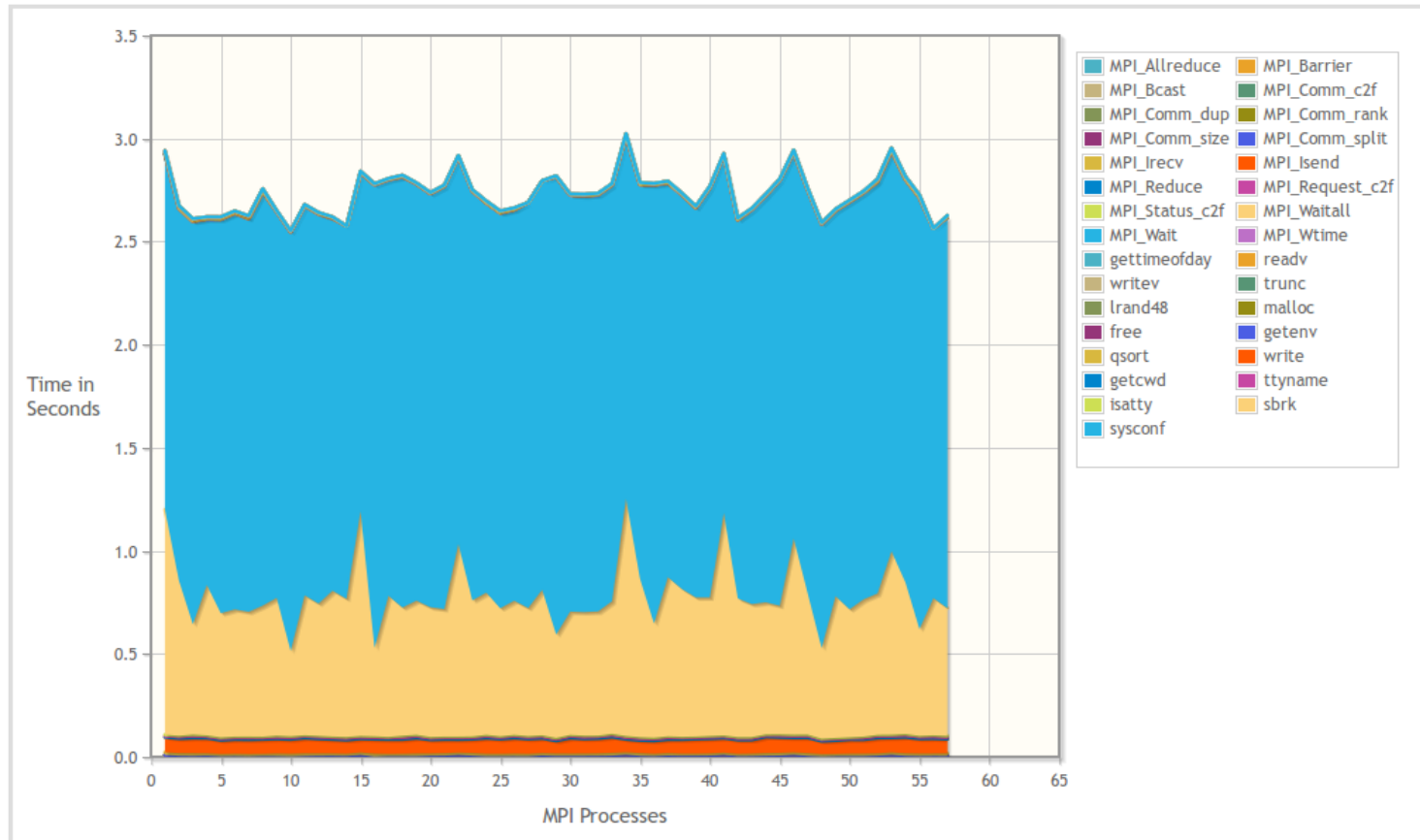
Size



Provides an overview of calls costs in terms of Time, Calls & Size

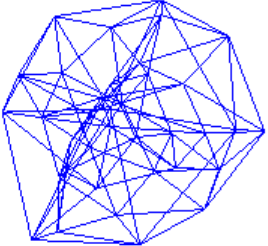
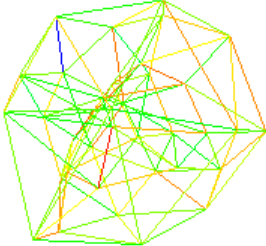
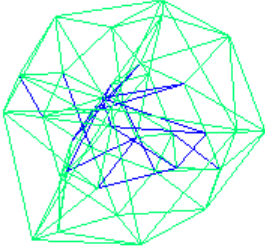
MPI Characterization

Example of BT.D @ 36



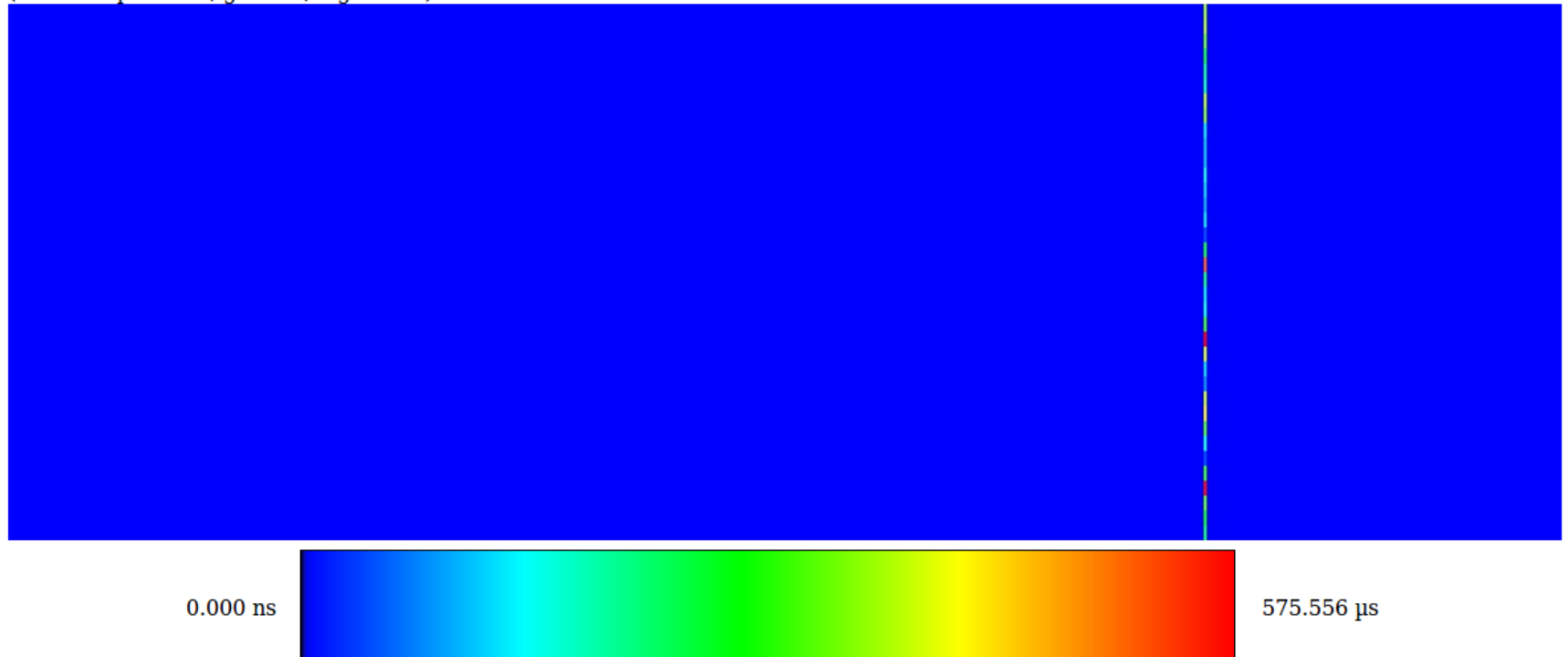
Check the balancing of your application relatively to MPI & Posix calls.

MPI_Irecv Topology

Calls	Time	Size
		

MPI_Allreduce in time

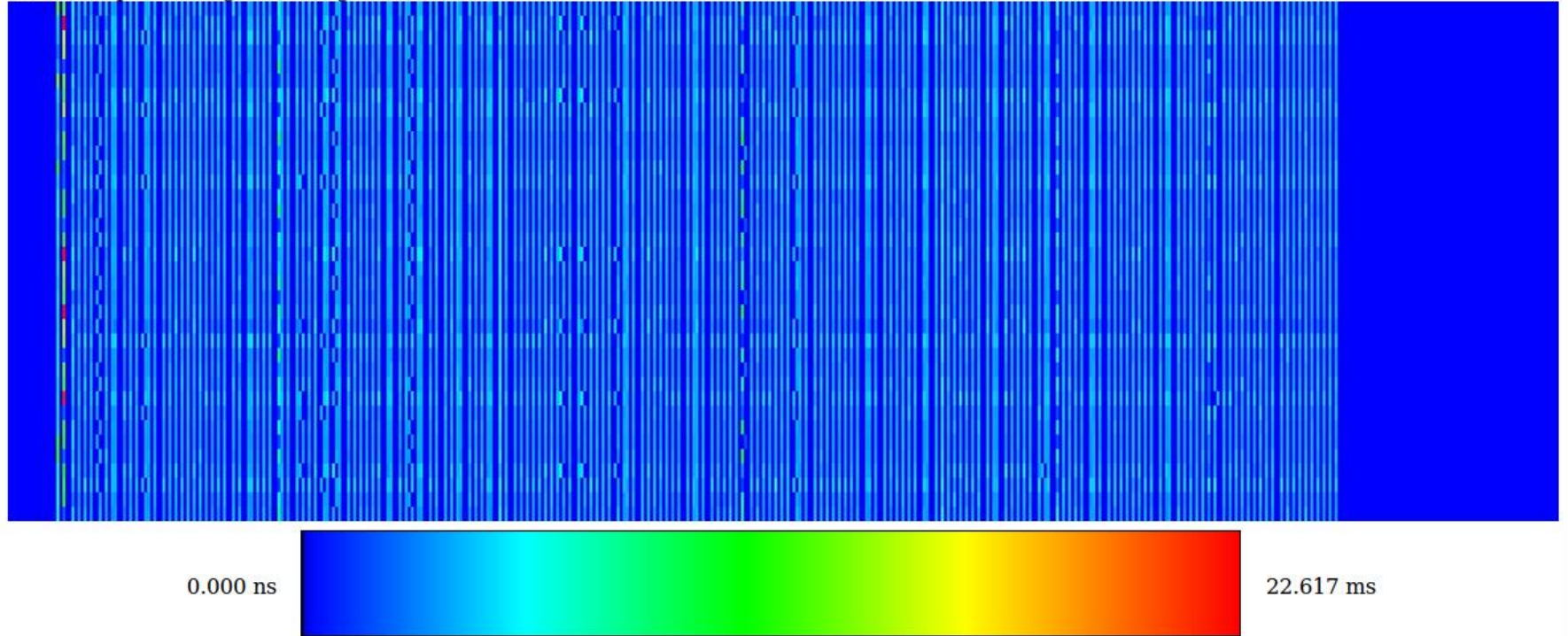
(scale temperature, gauss 0, logscale 0)



See the layout of your calls over space & time

MPI_Waitall in time

(scale temperature, gauss 0, logscale 0)



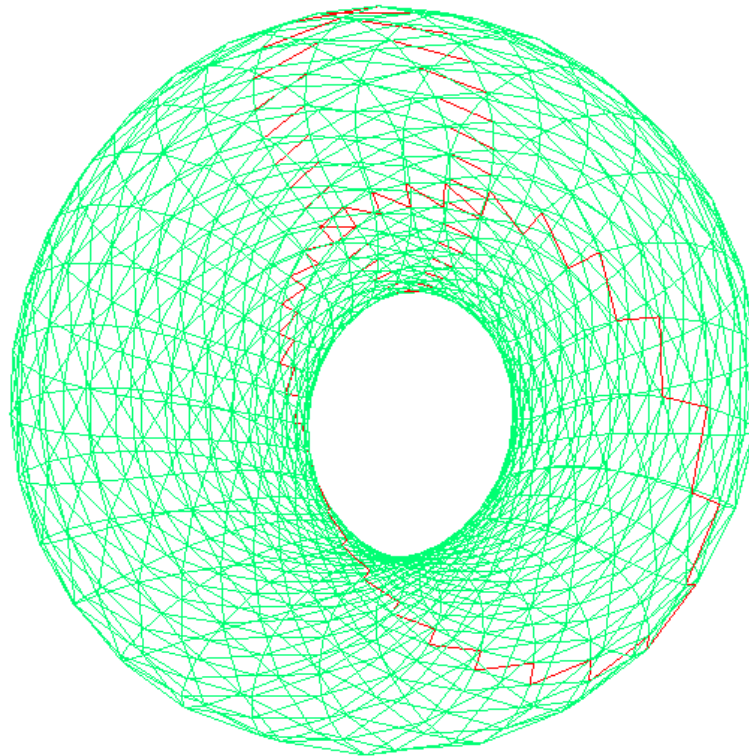
See the layout of your calls over space & time

But patterns are more
visible at scale....

MPI Characterization at scale

Example of BT.D @ 900

VI-HPS



156.110 MB



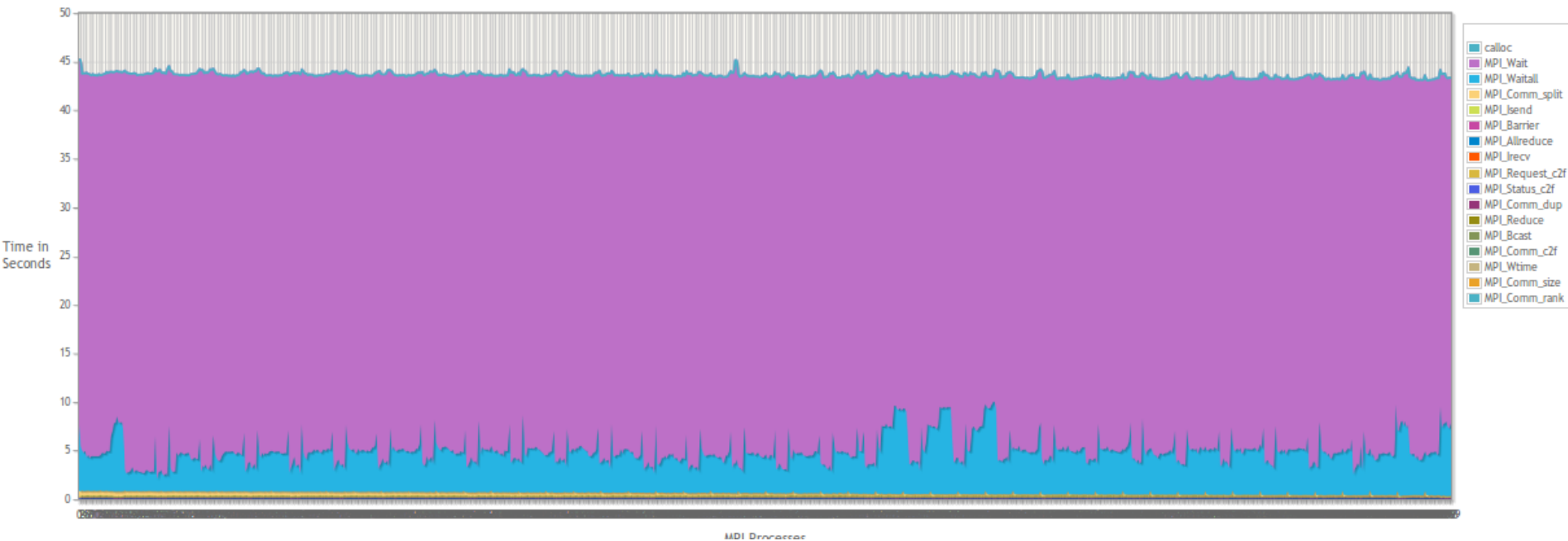
430.113 MB

MPI Characterization at scale

Example of BT.D @ 900

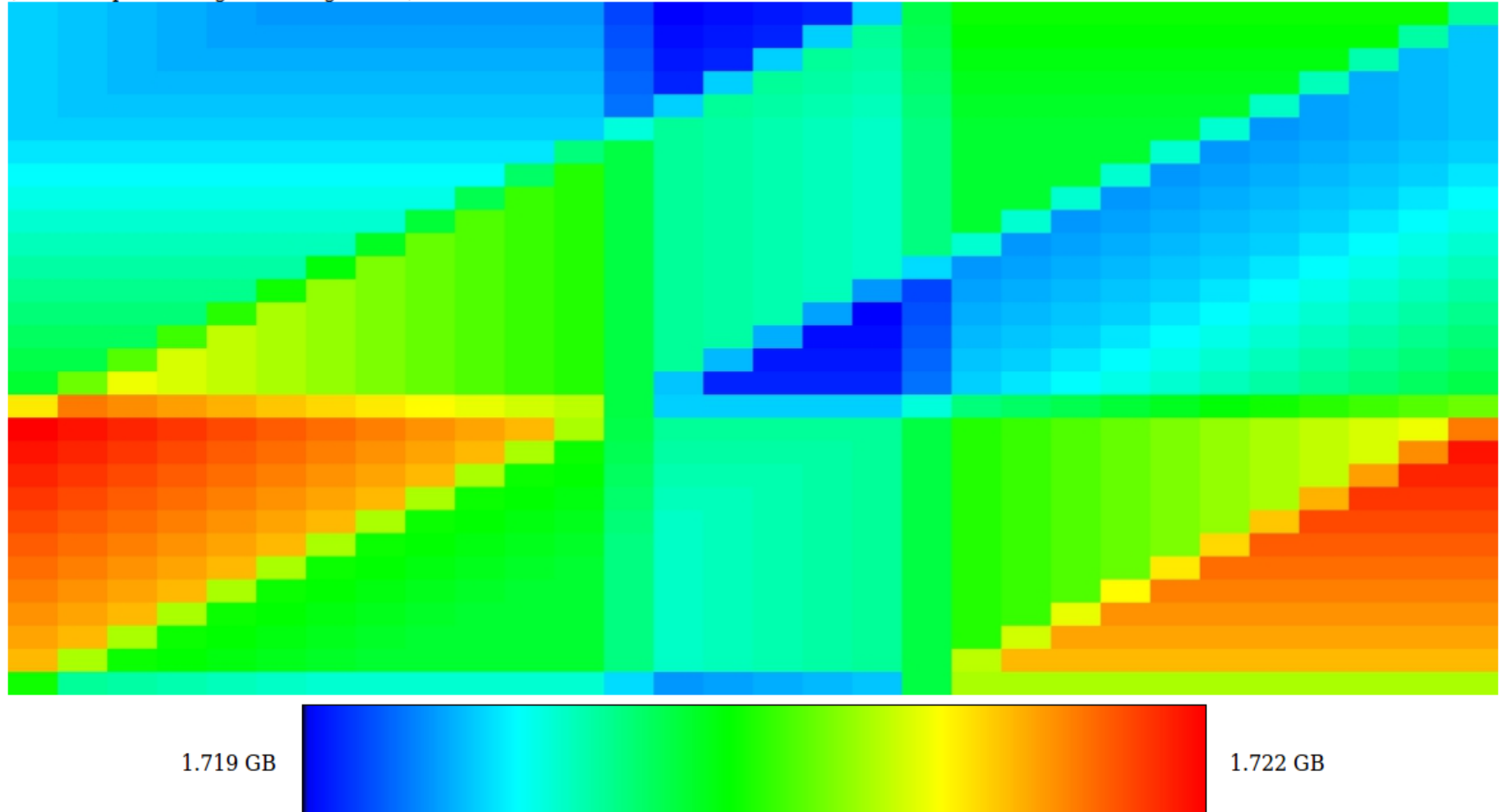
VI-HPS

Time Breakdown over Functions



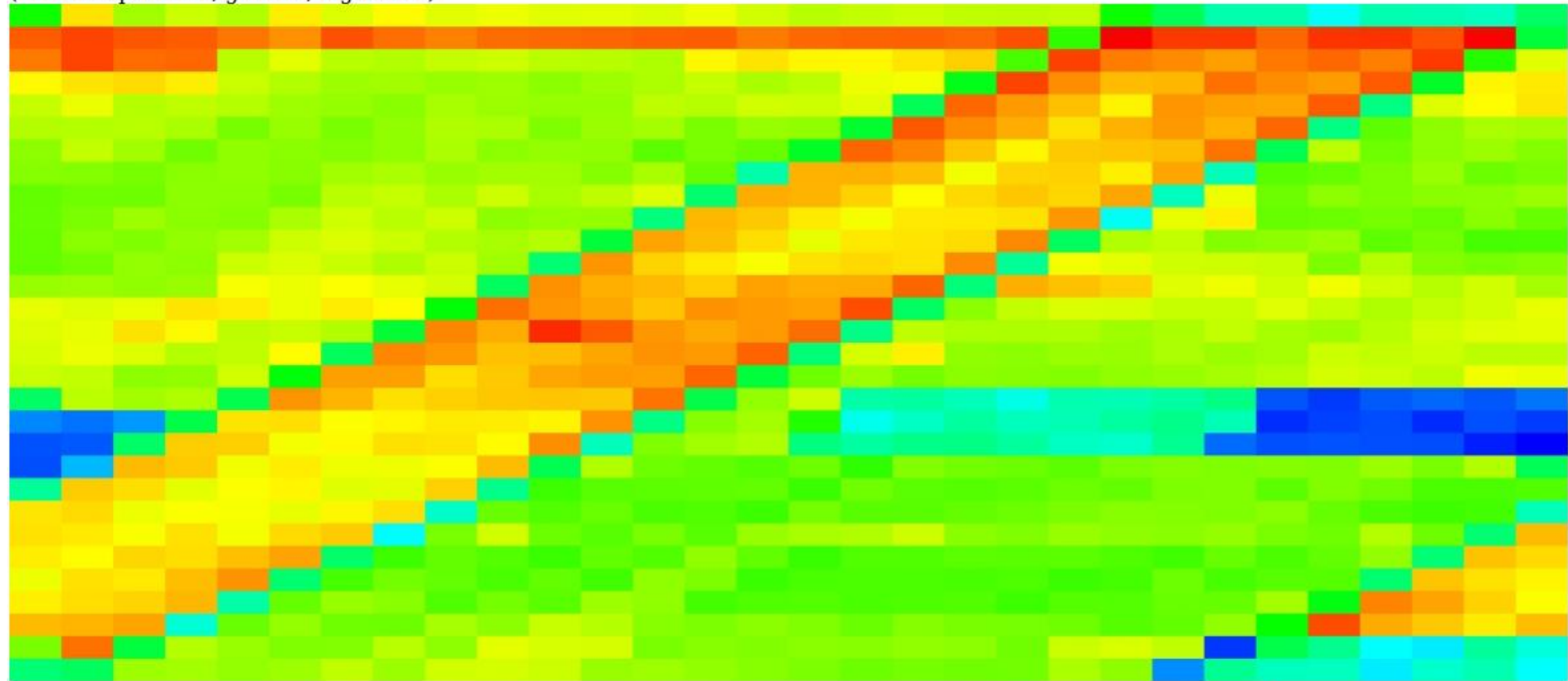
MPI_Irecv in size

(scale temperature, gauss 0, logscale 0)

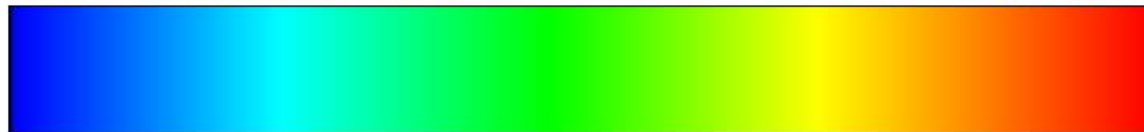


MPI_Wait in time

(scale temperature, gauss 0, logscale 0)



33.533 s



41.884 s