OTF Profile
Documentation and Annotated Example

Authors: Andreas Knüpfer, Matthias Jurenz, Robert Dietrich, Guido Juckeland

The `otfprofile[-mpi]` tool is a non-interactive command line command to generate a concise overview of a parallel program run from an OTF event trace.

Using MPI parallelism and OTF's parallel reading mode, it is able to process very extensive event trace data sets with highest I/O performance. The result of `otfprofile[-mpi]` is a LaTeX file using the pgfplots[1] package. If both are available, `otfprofile[-mpi]` will automatically produce a PDF file which gives a static overview about a number of essential aspects of the parallel execution covered by the original OTF trace. Otherwise, the completely self-contained LaTeX output file can be compiled on another machine where LaTeX and pgfplots are available.

The resulting PDF covers:

- general information about the event trace including monitoring settings,
- function (subroutine) statistics,
- hardware performance counter statistics (if recorded in the event trace),
- statistics about several aspects of MPI point-to-point communication, and
- statistics about four classes of MPI collective communication.

The layout is designed for a static representation in a PDF file or a print-out and is not supposed to replace interactive means of event trace analysis, of course. All output tables and graphs are designed to fit on a single A4 page, subsuming information if necessary.

For all process related information, a grouping scheme is enabled if there are more than 16 processes. Instead of per-process data it shows the respective data per group. Even if this is a limitation for the degree of detail it is inevitable to support event traces with arbitrarily many processes or threads. In that case, the charts show the average over the individual sums as well as range bars for the minimum and maximum of the processes in the group. See below for examples with grouping.

The `otfprofile[-mpi]` command supports the following options:

```
otfprofile[-mpi] - Generate a profile of an OTF trace in LaTeX format.

Syntax: otfprofile[-mpi] [options] <input file name>

  options:
     -h, --help           show this help message
     -V                   show OTF version
     -v                   increase output verbosity
                          (can be used more than once)
     -p                   show progress
     -f <n>               max. number of filehandles available per rank
                          (default: 50)
     -b <size>            set buffersize of the reader
                          (default: 1048576)
     -o <prefix>          specify the prefix of output file(s)
                          (default: result)
     -g <n>               max. number of process groups in LaTeX output
                          (range: 1-16, default: 16)
     --stat               read only summarized information, no events
     --[no]csv            enable/disable producing CSV output
                          (default: disabled)
     --[no]marker         enable/disable producing marker output for irregularity hints
                          (default: disabled)
     --[no]tex            enable/disable producing LaTeX output
                          (default: enabled)
     --[no]pdf            enable/disable producing PDF output
                          (implies --tex if enabled, default: enabled)
```

[1] See `http://sourceforge.net/projects/pgfplots/`.

Additional options are available to perform a K-means clustering based on feature vectors from every process trace:

```
-c, --cluster[ <alg>]
                      do additional clustering of processes/threads using
                      comparison algorithm <alg> (KMEANS or CLINKAGE)
                      (default comparison algorithm: KMEANS)
-m <mapfile>          write cluster mapping to <mapfile>
                      (implies -c, default: result.map)
-s <prefix>           call otfshrink to apply the cluster mapping to
                      input trace and produce a new trace named <prefix>
                      with symbolic links to the original (implies -c)
-H                    use hard groups for CLINKAGE clustering
                      (implies --cluster CLINKAGE)
-q <0-1>              quality threshold for CLINKAGE clustering
                      (implies --cluster CLINKAGE, default: 0.1)
```

The tool is licensed under BSD Open Source license and comes as regular part of the OTF package version 1.9 and later. For questions, comments, bug reports, feature requests, patches, etc. please contact the VampirTrace mailing list `vampirsupport@zih.tu-dresden.de` or Andreas Knüpfer via `andreas.knuepfer@tu-dresden.de`.

On the next pages follows an annotated example output of `otfprofile[-mpi]` to familiarize the reader with the overview representation.

OTF Profile

Trace Properties

OTF Version: 1.9sawfish
Creator: VampirTrace 5.11
File: fd4trace.otf

Number of Processes: 64
Timer Resolution: 2.40004 GHz

Comments:

```
Trace Times:
 Start: Wed May 25 10:25:07 2011 (1306311907168708)
 Stop: Wed May 25 10:25:57 2011 (1306311957367854)
 Elapsed: 00:00:50 (50199146)
VampirTrace Environment:
 VT_MODE: TRACE
 VT_BUFFER_SIZE: 100M
 VT_SYNC_FLUSH: no
 VT_SYNC_FLUSH_LEVEL: 80
 VT_MAX_FLUSHES: 1
 VT_METRICS: PAPI_FP_OPS
 VT_RUSAGE: <not set>
 VT_MPITRACE: yes
 VT_MEMTRACE: no
 VT_CPUIDTRACE: no
 VT_IOTRACE: no
 VT_FILTER_SPEC: <not set>
 VT_GROUPS_SPEC: ./groups.txt
```

The trace properties give information about the input file. These include the trace creator tool, the OTF version and the file name of the input trace. Furthermore the number of processes, the time resolution and several comments, generated by the creator tool, are shown, including essential monitoring settings.

Top 50 of 53 Functions

Function	invocations[#]	excl. time[sec] ▽	incl. time[sec]
MPI_Waitall	25856	1405.71	1405.71
advection_mp_advection_set_vorte	6464	805.238	805.238
MPI_Barrier	12800	415.039	415.039
advection_mp_advection_compute_	51200	298.926	298.926
MPI_Init	64	132.157	135.657
MPI_Allreduce	31588	56.6929	56.6929
FD4_COUPLE_MOD::FD4_COUPLE_PUT	6528	10.2587	1403.77
MPI_Bcast	8096	10.0868	10.0868
MPI_Isend	186932	7.93241	7.93241
FD4_BALANCE_MOD::FD4_BALANCE_BLO	6464	4.74949	4.74949
sync time	128	3.69511	3.69511
advection_mp_advection_init_	64	2.96611	19.3321
advection_mp_advection_init_coup	64	1.73647	1.76705
MPI_Comm_split	6480	1.66516	1.66516
MPI_Type_commit	223944	1.00994	1.00994
MAIN__	64	0.997348	3164.05
MPI_Comm_dup	872	0.813821	0.813821
MPI_Type_free	244040	0.775153	0.775153
FD4_GHOSTCOMM_MOD::FD4_GHOSTCOMM	6400	0.723271	23.3739
FD4_PART_METIS_MOD::FD4_PART_MET	6464	0.620746	13.1635
MPI_Irecv	186932	0.598095	0.598095
FD4_DOMAIN_MOD::FD4_DOMAIN_CLEAR	64	0.454692	0.458224
MPI_Type_size	170772	0.38737	0.38737
MPI_Alltoall	808	0.204013	0.204013
FD4_BALANCE_MOD::FD4_BALANCE_REA	6464	0.155236	74.6322
FD4_BALANCE_MOD::FD4_BALANCE_EST	6464	0.109189	56.5246
FD4_DOMAIN_MOD::FD4_DOMAIN_CREAT	64	0.0893182	0.875386
FD4_COUPLE_MOD::FD4_COUPLE_DELET	128	0.0427842	0.085552
FD4_COUPLE_MOD::FD4_COUPLE_COMMI	128	0.0424892	0.0424892
FD4_BALANCE_MOD::FD4_BALANCE_UPD	6464	0.0380109	0.0380109
MPI_Get_count	12928	0.0284654	0.0284654
MPI_Allgatherv	16	0.0235577	0.0235577
MPI_Wait	6464	0.0198073	0.0198073
FD4_GHOSTCOMM_MOD::FD4_GHOSTCOMM	128	0.0103993	0.0207974
MPI_Comm_free	1692	0.00999424	0.00999424
MPI_Comm_size	2628	0.00768388	0.00768388
FD4_DOMAIN_MOD::FD4_DOMAIN_DELET	64	0.00649255	0.477294
MPI_Gatherv	808	0.00625685	0.00625685
MPI_Comm_rank	1884	0.00545652	0.00545652
FD4_GHOSTCOMM_MOD::FD4_GHOSTCOMM	128	0.00511179	0.00511179
MPI_Reduce	192	0.0028241	0.0028241
MPI_Finalize	64	0.00258425	0.197571
FD4_DOMAIN_MOD::FD4_DOMAIN_CREAT	64	0.00206774	0.877454
FD4_DOMAIN_MOD::FD4_DOMAIN_DUMP_	64	0.0019427	0.0047668
FD4_BALANCE_MOD::FD4_BALANCE_REQ	64	0.00137562	0.00137562
FD4_UTIL_MOD::FD4_UTIL_ALLOCATE_	64	0.000794241	8.62907
FD4_VARTAB_MOD::FD4_VARTAB_CHECK	64	0.000579258	0.000579258
FD4_VARTAB_MOD::FD4_VARTAB_CREAT	64	0.000560677	0.000560677
FD4_BALANCE_MOD::FD4_BALANCE_PAR	64	0.000461656	0.000461656
FD4_DOMAIN_MOD::FD4_DOMAIN_MAX_B	64	0.000318863	0.000318863

The function table shows the 50 most important functions (subroutines) with their number of invocations as well as the exclusive and inclusive total run times (over all processes/threads). The selection and the sorting is according to the exclusive run time. All other functions (subroutines) are ignored. This table easily shows where the majority of the parallel run time was spent.

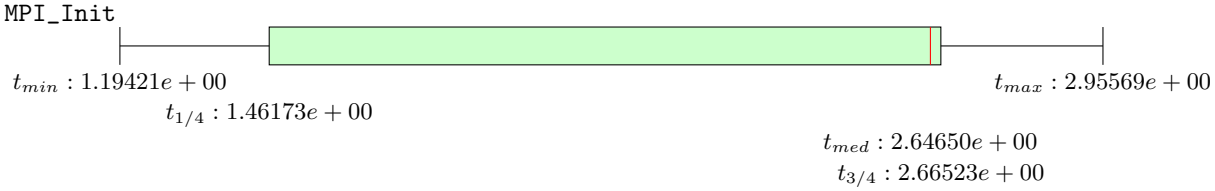
PAPI_FP_OPS [Mega] (Top 50)

Function	excl. time[sec]	PAPI_FP_OPS/sec
MPI_Waitall	1405.71	0.0505527
advection_mp_advection_set_vorte	805.238	879.29
MPI_Barrier	415.039	3.2527e-07
advection_mp_advection_compute_	298.926	332.41
MPI_Init	132.157	0.008061
MPI_Allreduce	56.6929	0.0294365
FD4_COUPLE_MOD::FD4_COUPLE_PUT	10.2587	0.00551209
MPI_Bcast	10.0868	0.016133
MPI_Isend	7.93241	0.162627
FD4_BALANCE_MOD::FD4_BALANCE_BLO	4.74949	0.0027424
sync time	3.69511	0.0027807
advection_mp_advection_init_	2.96611	373.422
advection_mp_advection_init_coup	1.73647	0.0145635
MPI_Comm_split	1.66516	0.00690383
MPI_Type_commit	1.00994	0.00063667
MAIN__	0.997348	0.179971
MPI_Comm_dup	0.813821	0.00387063
MPI_Type_free	0.775153	0.00552794
FD4_GHOSTCOMM_MOD::FD4_GHOSTCOMM	0.723271	0.000647061
FD4_PART_METIS_MOD::FD4_PART_MET	0.620746	25.4162
MPI_Irecv	0.598095	0.00862238
FD4_DOMAIN_MOD::FD4_DOMAIN_CLEAR	0.454692	0.00131957
MPI_Type_size	0.38737	0
MPI_Alltoall	0.204013	0.21007
FD4_BALANCE_MOD::FD4_BALANCE_REA	0.155236	0.00265402
FD4_BALANCE_MOD::FD4_BALANCE_EST	0.109189	0.660828
FD4_DOMAIN_MOD::FD4_DOMAIN_CREAT	0.0893182	0.153227
FD4_COUPLE_MOD::FD4_COUPLE_DELET	0.0427842	0
FD4_COUPLE_MOD::FD4_COUPLE_COMMI	0.0424892	4.74
FD4_BALANCE_MOD::FD4_BALANCE_UPD	0.0380109	0.0413829
MPI_Get_count	0.0284654	1.63289
MPI_Allgatherv	0.0235577	0.0898646
MPI_Wait	0.0198073	1.08919
FD4_GHOSTCOMM_MOD::FD4_GHOSTCOMM	0.0103993	0
MPI_Comm_free	0.00999424	0.207219
MPI_Comm_size	0.00768388	0
FD4_DOMAIN_MOD::FD4_DOMAIN_DELET	0.00649255	0.168193
MPI_Gatherv	0.00625685	0.0083109
MPI_Comm_rank	0.00545652	0
FD4_GHOSTCOMM_MOD::FD4_GHOSTCOMM	0.00511179	2.75598
MPI_Reduce	0.0028241	0.894444
MPI_Finalize	0.00258425	0.369547
FD4_DOMAIN_MOD::FD4_DOMAIN_CREAT	0.00206774	2.28171
FD4_DOMAIN_MOD::FD4_DOMAIN_DUMP_	0.0019427	0.256344
FD4_BALANCE_MOD::FD4_BALANCE_REQ	0.00137562	0
FD4_UTIL_MOD::FD4_UTIL_ALLOCATE_	0.000794241	0
FD4_VARTAB_MOD::FD4_VARTAB_CHECK	0.000579258	1.18773
FD4_VARTAB_MOD::FD4_VARTAB_CREAT	0.000560677	0.825788
FD4_BALANCE_MOD::FD4_BALANCE_PAR	0.000461656	0
FD4_DOMAIN_MOD::FD4_DOMAIN_MAX_B	0.000318863	0

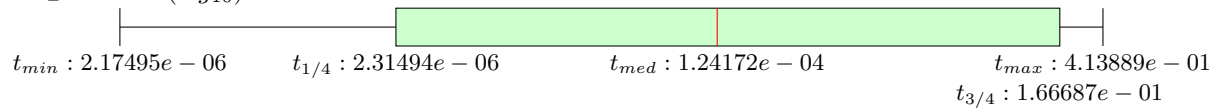
If available, this table shows all available performance counter rates (counter difference over time) for the 50 functions from the previous table. Again, it is sorted by the total exclusive run time, which is repeated as a reference to assess critical or noncritical function calls. Only accumulated counters (like typical PAPI counters), which can be directly mapped to the function call intervals, are supported here. This table allows to see hot-spots of counter activities spread over the essential functions (subroutines).

Top 50 Dispersion of Functions (in seconds)

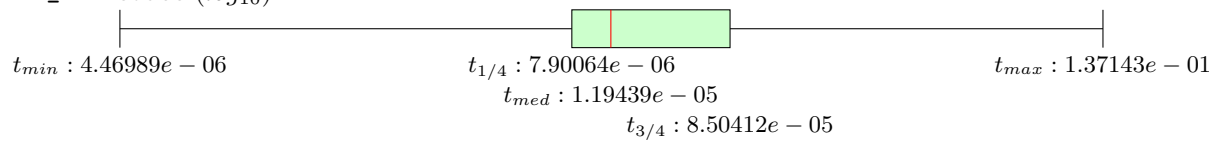
`MPI_Init`



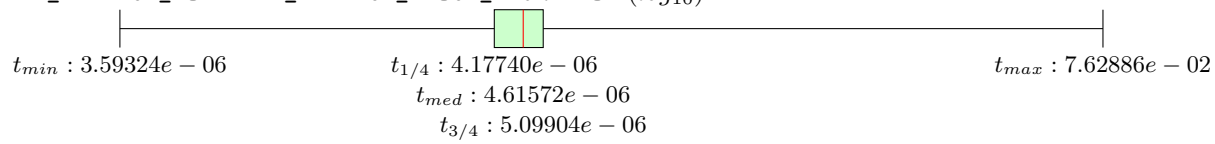
`MPI_Waitall` (log_{10})



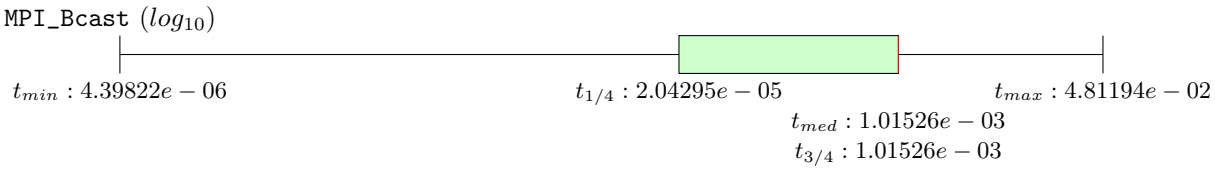
`MPI_Allreduce` (log_{10})



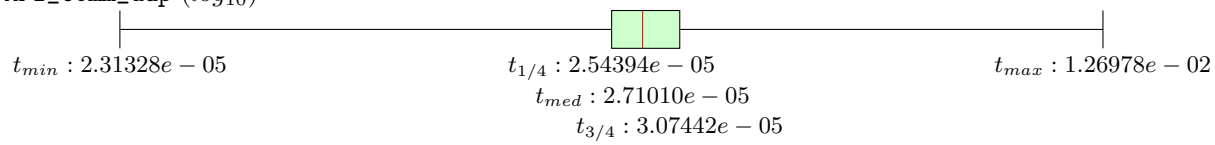
`FD4_BALANCE_MOD::FD4_BALANCE_BLOCK_MIGRATION` (log_{10})



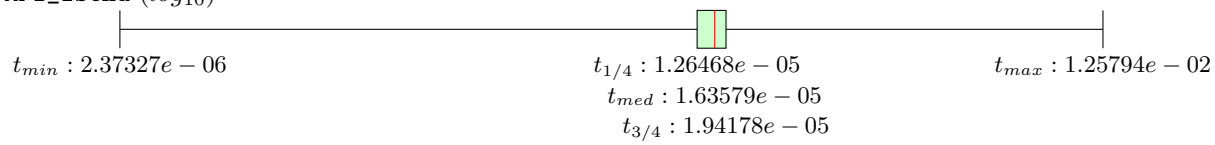
`MPI_Bcast` (log_{10})



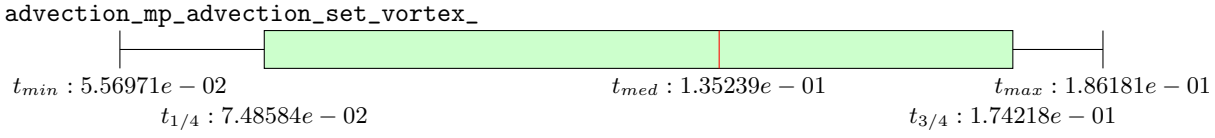
`MPI_Comm_dup` (log_{10})



`MPI_Isend` (log_{10})



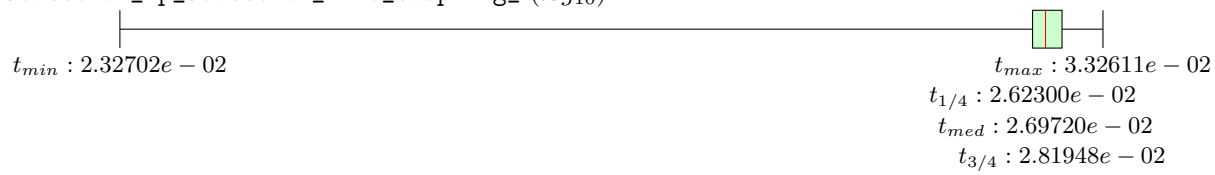
`advection_mp_advection_set_vortex_`



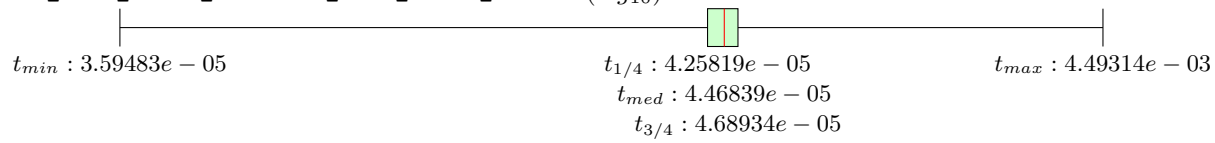
advection_mp_advection_compute_ (log_{10})

FD4_COUPLE_MOD::FD4_COUPLE_PUT (log_{10})

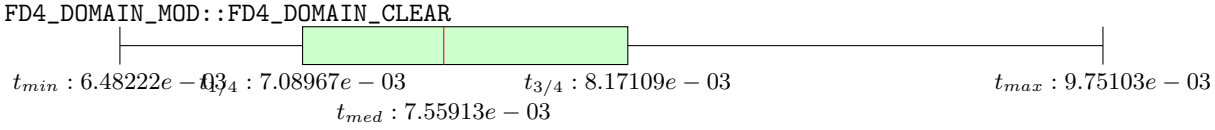
advection_mp_advection_init_coupling_ (log_{10})



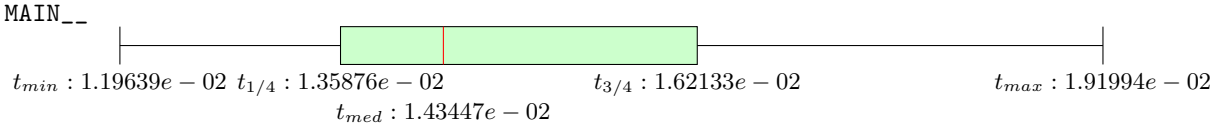
FD4_PART_METIS_MOD::FD4_PART_METIS_SUBSET (log_{10})



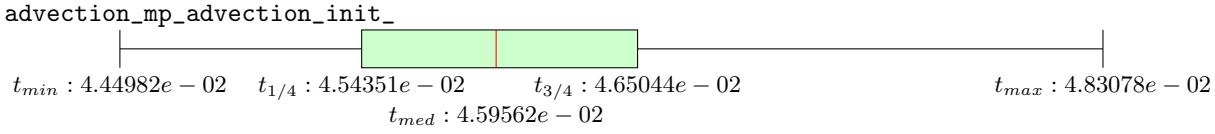
FD4_DOMAIN_MOD::FD4_DOMAIN_CLEAR



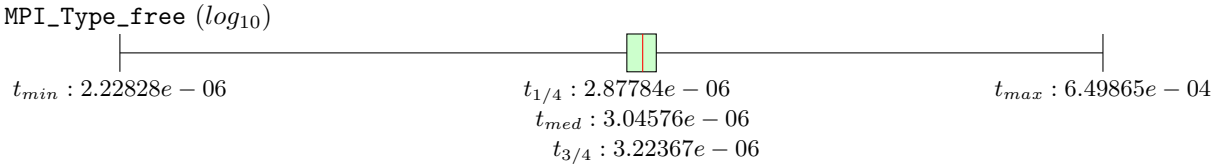
MAIN__



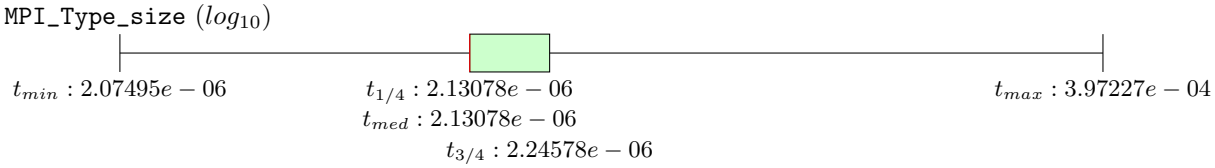
advection_mp_advection_init_



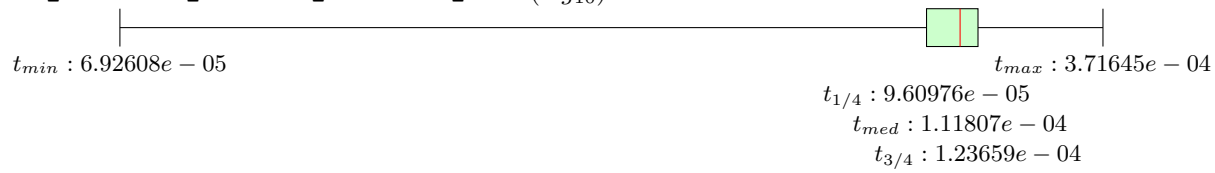
MPI_Type_free (log_{10})



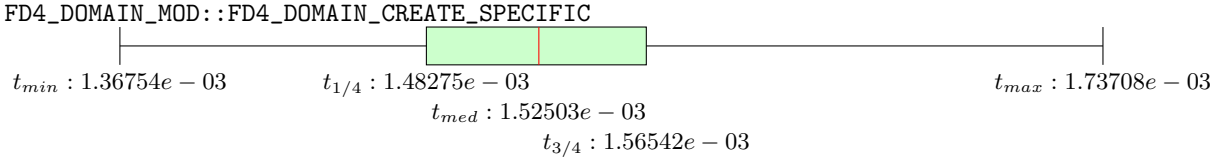
MPI_Type_size (log_{10})



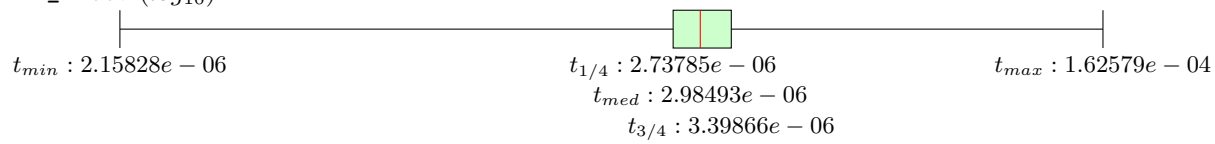
FD4_GHOSTCOMM_MOD::FD4_GHOSTCOMM_EXCH (log_{10})



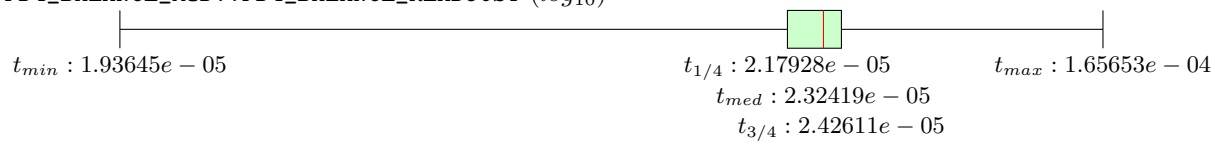
FD4_DOMAIN_MOD::FD4_DOMAIN_CREATE_SPECIFIC



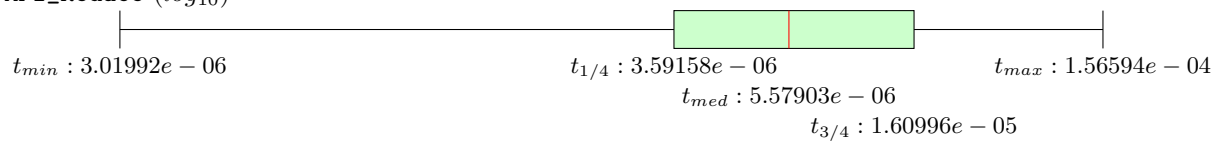
MPI_Irecv (log_{10})



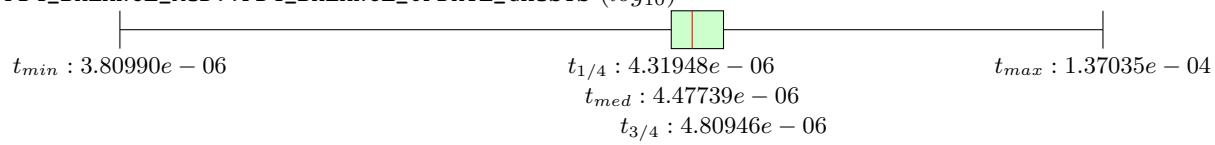
FD4_BALANCE_MOD::FD4_BALANCE_READJUST (log_{10})



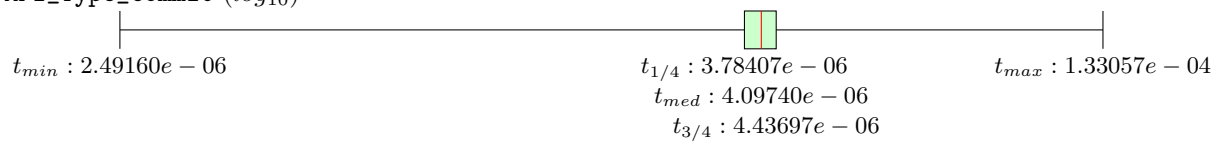
MPI_Reduce (log_{10})



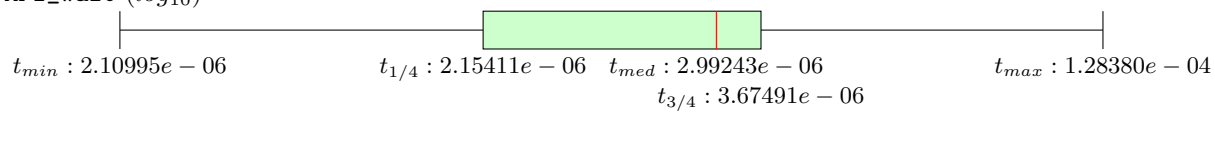
FD4_BALANCE_MOD::FD4_BALANCE_UPDATE_GHOSTS (log_{10})



MPI_Type_commit (log_{10})

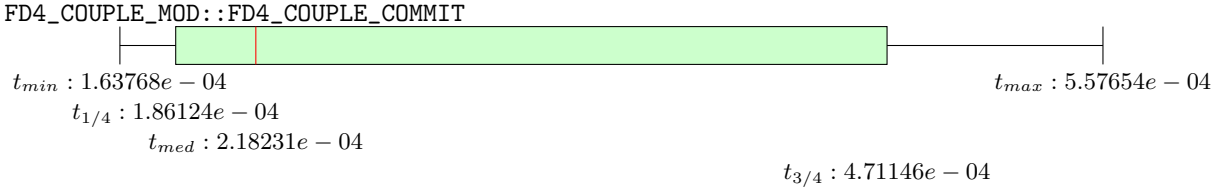


MPI_Wait (log_{10})



MPI_Alltoall (log_{10})

FD4_COUPLE_MOD::FD4_COUPLE_COMMIT

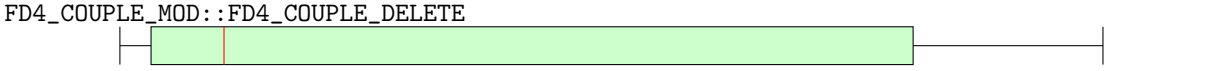


t_{min} : 1.63768$e-04$ $t_{1/4}$: 1.86124$e-04$ t_{med} : 2.18231$e-04$ $t_{3/4}$: 4.71146$e-04$ t_{max} : 5.57654$e-04$

MPI_Comm_split

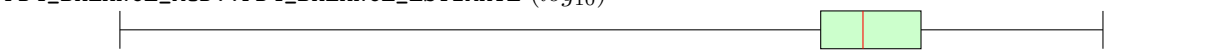
t_{min} : 1.60096$e-05$ $t_{1/4}$: 2.11848$e-04$ t_{med} : 2.24727$e-04$ $t_{3/4}$: 2.31460$e-04$ t_{max} : 3.06341$e-04$

FD4_COUPLE_MOD::FD4_COUPLE_DELETE



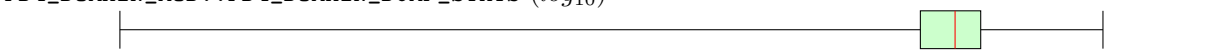
t_{min} : 1.78987$e-04$ $t_{1/4}$: 1.90659$e-04$ t_{med} : 2.18010$e-04$ $t_{3/4}$: 4.76304$e-04$ t_{max} : 5.47458$e-04$

FD4_BALANCE_MOD::FD4_BALANCE_ESTIMATE (log_{10})



t_{min} : 1.17547$e-05$ $t_{1/4}$: 1.36442$e-05$ t_{med} : 1.48946$e-05$ $t_{3/4}$: 1.80604$e-05$ t_{max} : 6.78766$e-05$

FD4_DOMAIN_MOD::FD4_DOMAIN_DUMP_STATS (log_{10})

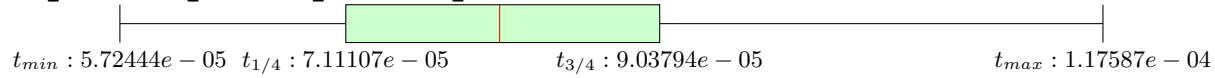


t_{min} : 2.19961$e-05$ $t_{1/4}$: 2.74643$e-05$ t_{med} : 3.02459$e-05$ $t_{3/4}$: 3.31729$e-05$ t_{max} : 6.95583$e-05$

MPI_Comm_free (log_{10})

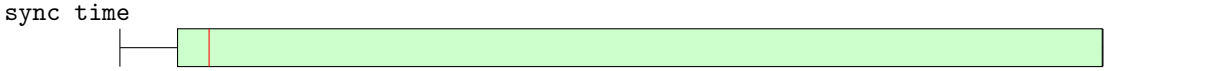
t_{min} : 3.33825$e-06$ $t_{1/4}$: 3.63241$e-06$ t_{med} : 5.47403$e-06$ $t_{3/4}$: 6.31901$e-06$ t_{max} : 3.71757$e-05$

FD4_GHOSTCOMM_MOD::FD4_GHOSTCOMM_DELETE



t_{min} : 5.72444$e-05$ $t_{1/4}$: 7.11107$e-05$ t_{med} : 8.05530$e-05$ $t_{3/4}$: 9.03794$e-05$ t_{max} : 1.17587$e-04$

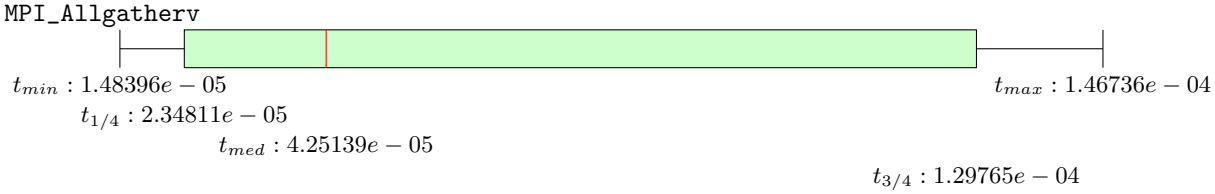
sync time



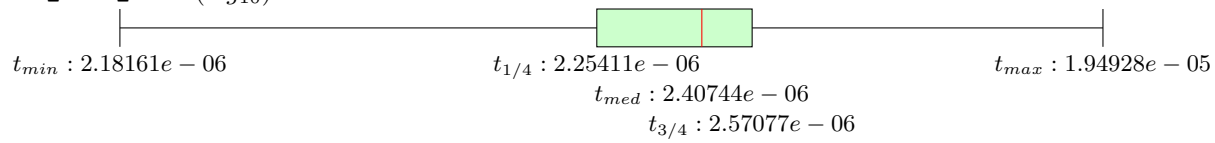
t_{min} : 7.40593$e-04$ $t_{1/4}$: 2.87220$e-03$ t_{med} : 4.03859$e-03$ $t_{3/4}$: 3.70612$e-02$ t_{max} : 3.70876$e-02$

MPI_Gatherv (log_{10})

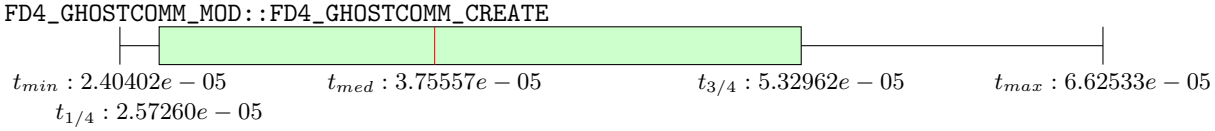
MPI_Allgatherv



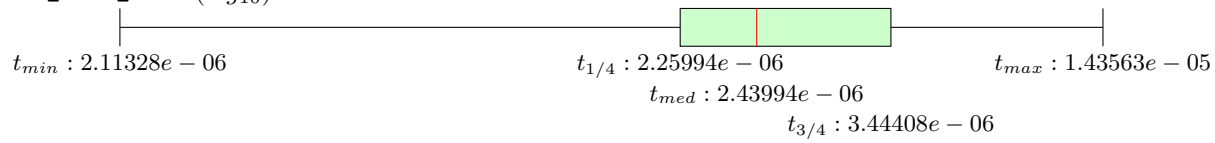
MPI_Comm_rank (log_{10})



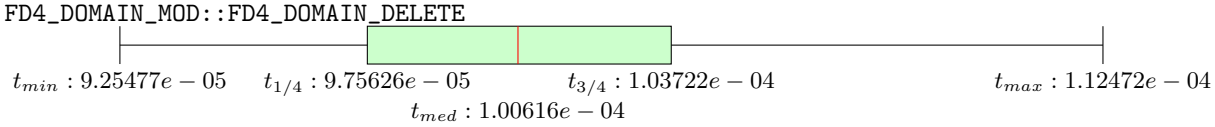
FD4_GHOSTCOMM_MOD::FD4_GHOSTCOMM_CREATE



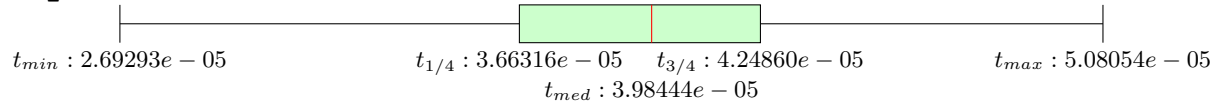
MPI_Comm_size (log_{10})



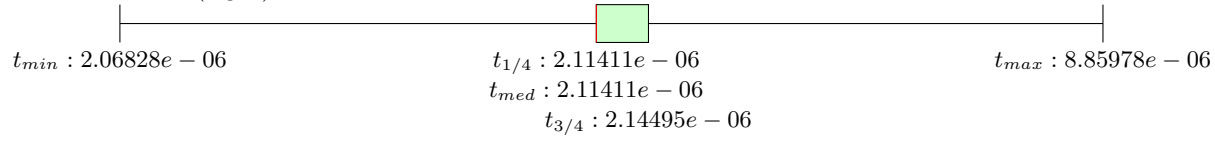
FD4_DOMAIN_MOD::FD4_DOMAIN_DELETE



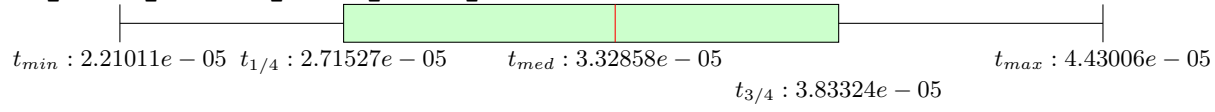
MPI_Finalize



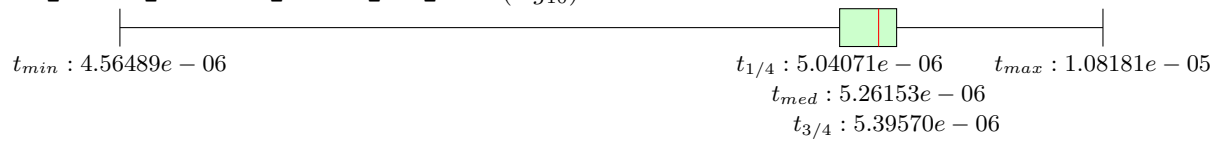
MPI_Get_count (log_{10})



FD4_DOMAIN_MOD::FD4_DOMAIN_CREATE_FIXED

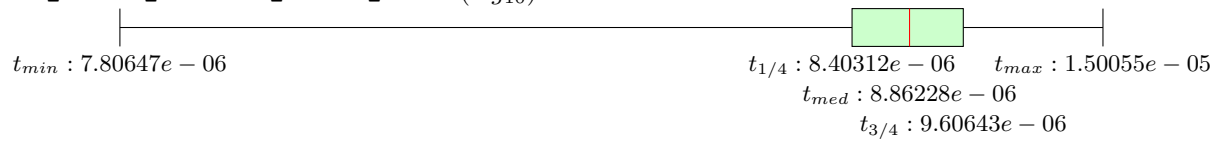


FD4_DOMAIN_MOD::FD4_DOMAIN_MAX_BEXT (log_{10})



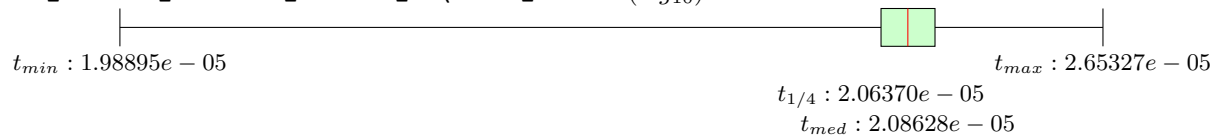
t_{min} : 4.56489e − 06
$t_{1/4}$: 5.04071e − 06
t_{med} : 5.26153e − 06
$t_{3/4}$: 5.39570e − 06
t_{max} : 1.08181e − 05

FD4_VARTAB_MOD::FD4_VARTAB_CHECK (log_{10})



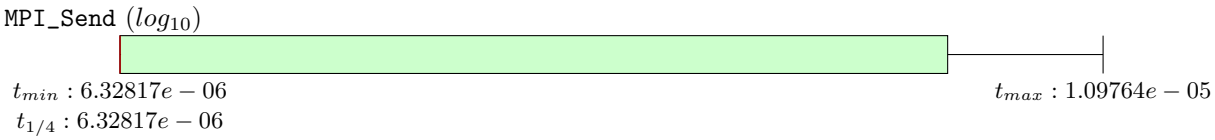
t_{min} : 7.80647e − 06
$t_{1/4}$: 8.40312e − 06
t_{med} : 8.86228e − 06
$t_{3/4}$: 9.60643e − 06
t_{max} : 1.50055e − 05

FD4_BALANCE_MOD::FD4_BALANCE_REQUIRED_BLOCKS (log_{10})



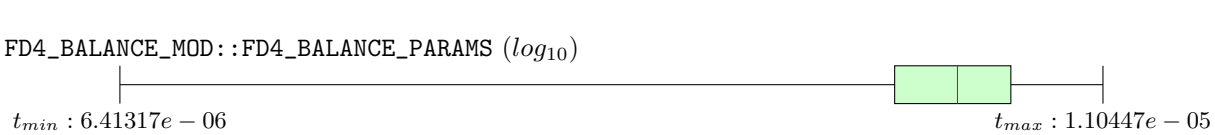
t_{min} : 1.98895e − 05
$t_{1/4}$: 2.06370e − 05
t_{med} : 2.08628e − 05
$t_{3/4}$: 2.11611e − 05
t_{max} : 2.65327e − 05

MPI_Send (log_{10})



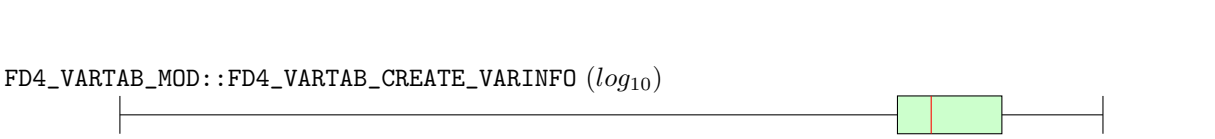
t_{min} : 6.32817e − 06
$t_{1/4}$: 6.32817e − 06
t_{med} : 6.32817e − 06
$t_{3/4}$: 7.39315e − 06
t_{max} : 1.09764e − 05

FD4_BALANCE_MOD::FD4_BALANCE_PARAMS (log_{10})



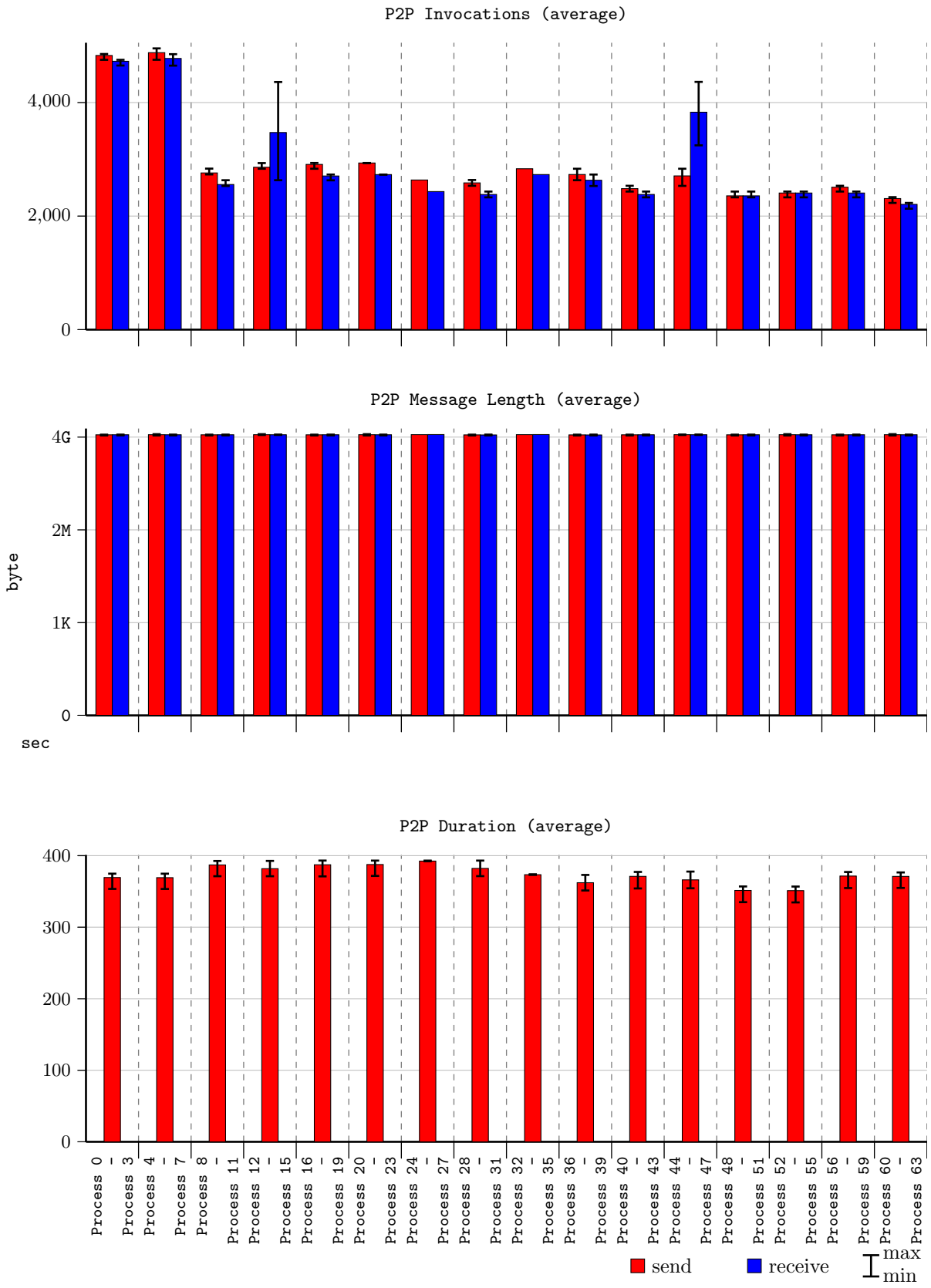
t_{min} : 6.41317e − 06
$t_{1/4}$: 7.05649e − 06
t_{med} : 7.57981e − 06
$t_{3/4}$: 8.34812e − 06
t_{max} : 1.10447e − 05

FD4_VARTAB_MOD::FD4_VARTAB_CREATE_VARINFO (log_{10})



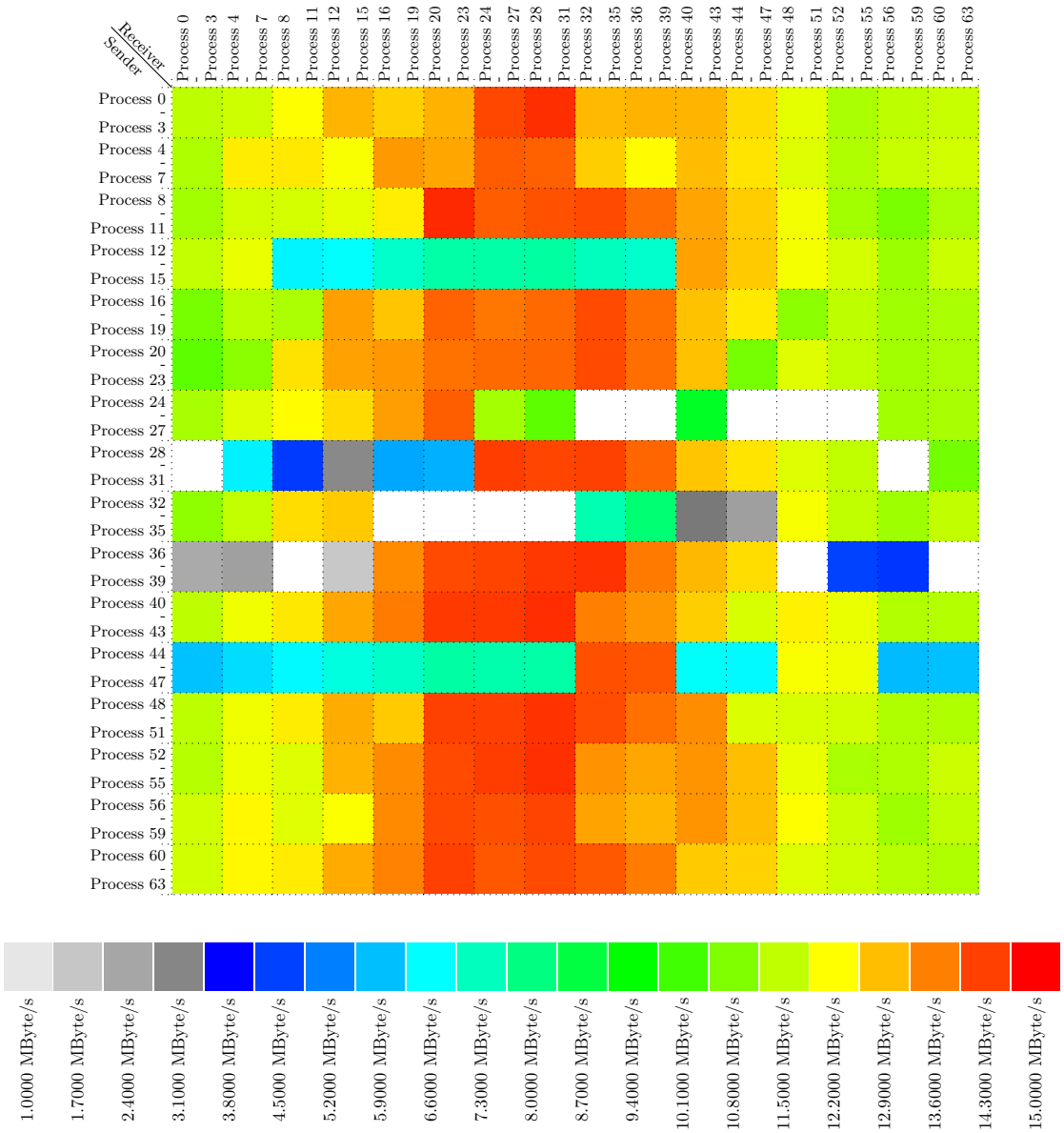
t_{min} : 6.50484e − 06
$t_{1/4}$: 7.09482e − 06
t_{med} : 7.31315e − 06
$t_{3/4}$: 8.06813e − 06
t_{max} : 1.05197e − 05

If available, this list of box plots of the 50 most important functions (subroutines) shows the dispersion and distribution of each function with information about the minimum (left end of the whisker), 25th percentile, median, 75th percentile, and maximum (right end of the whisker). The list is sorted by the distance of the maximum and 75th percentile. Some plots uses a logarithmic scale for visualization indicated by the log_{10} after the function name.



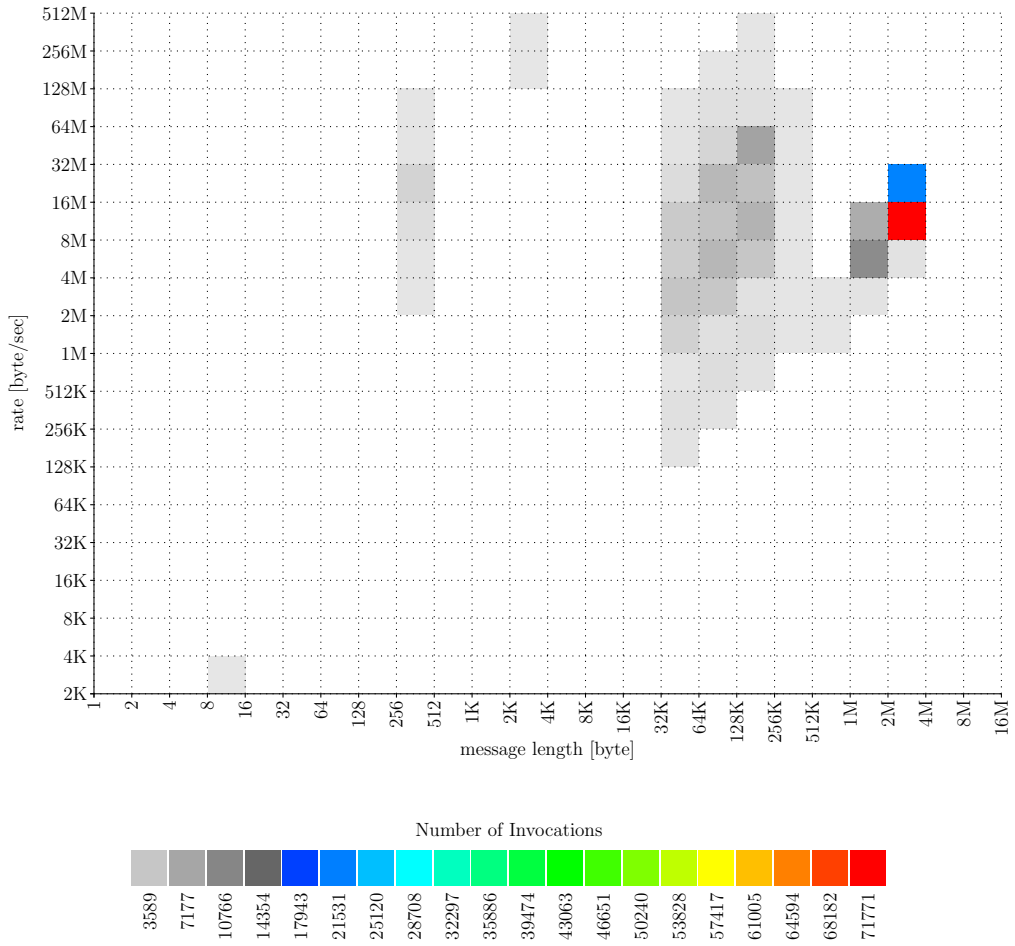
If the trace file contains point-to-point (P2P) communication, statistics about invocations (top), message volumes (middle) and durations (bottom) are visualized as bar charts. The first two charts distinguish send (red) and receive (blue) values. This chart shows the uniform or irregular distribution of message transfers. Here, an example with grouping is shown: 64 processes are subsumed in 16 groups with 4 members each.

P2P - Message Rate (average)



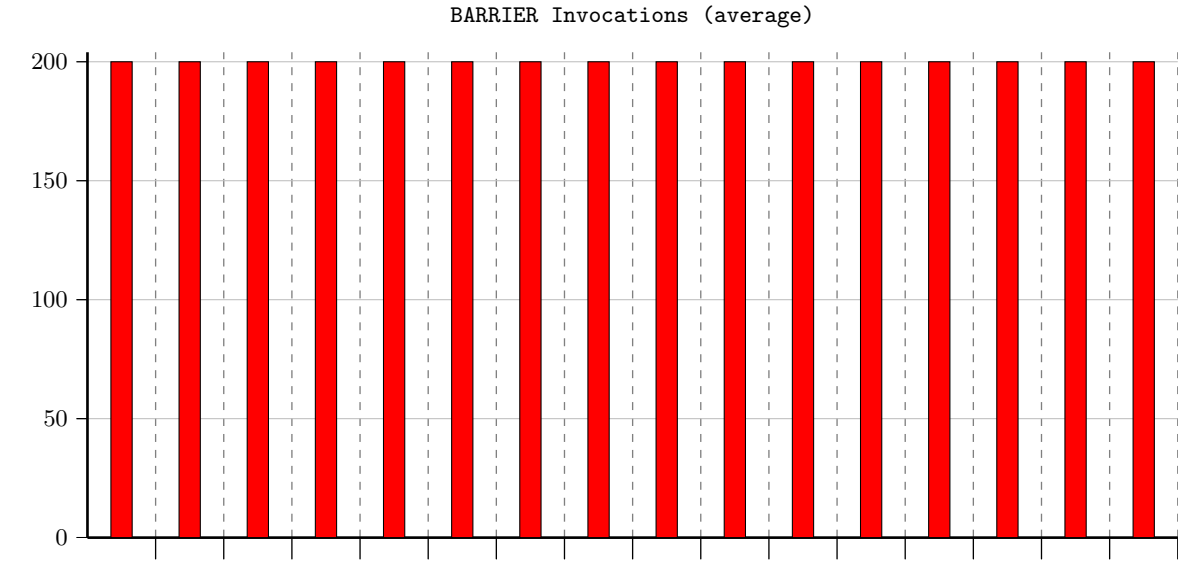
The message rate matrix shows the average message rate between each communication pair. The message rate values are color-coded according to the given scale. The senders are labeled vertically and the receivers horizontally. If processes are grouped, a colored field shows the average data rate between two groups instead of individual pairs. This chart uncovers the prevalent point-to-point communication relationships inside the parallel target application.

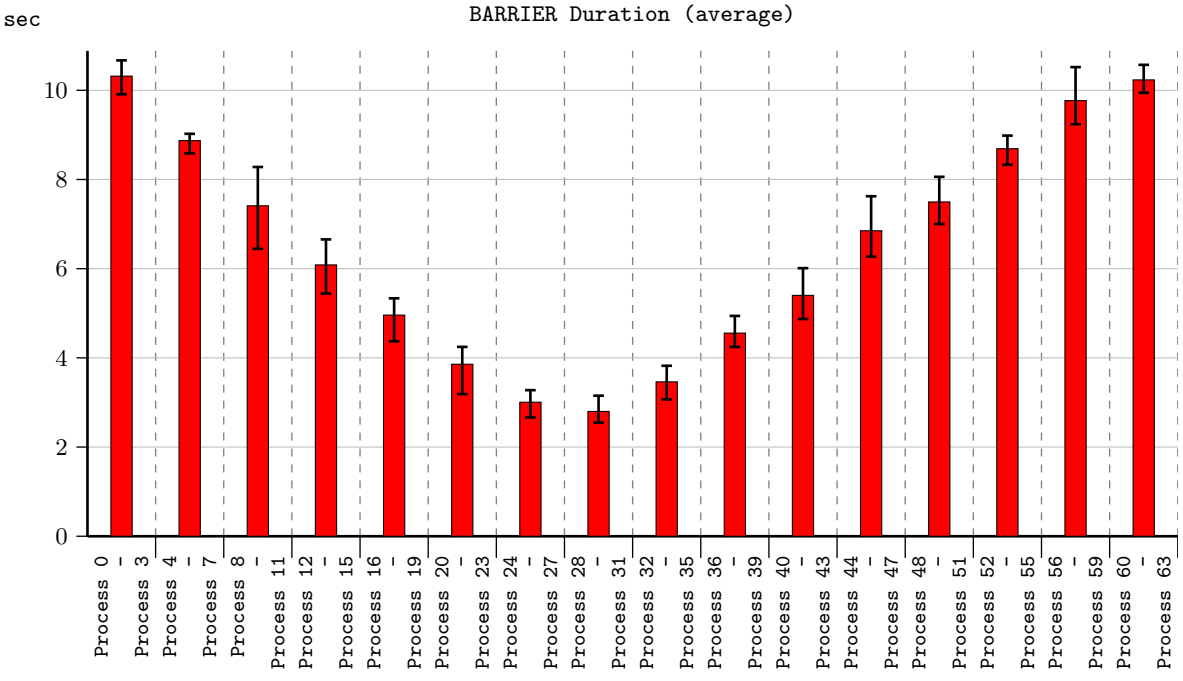
P2P - Message Rate Histogram



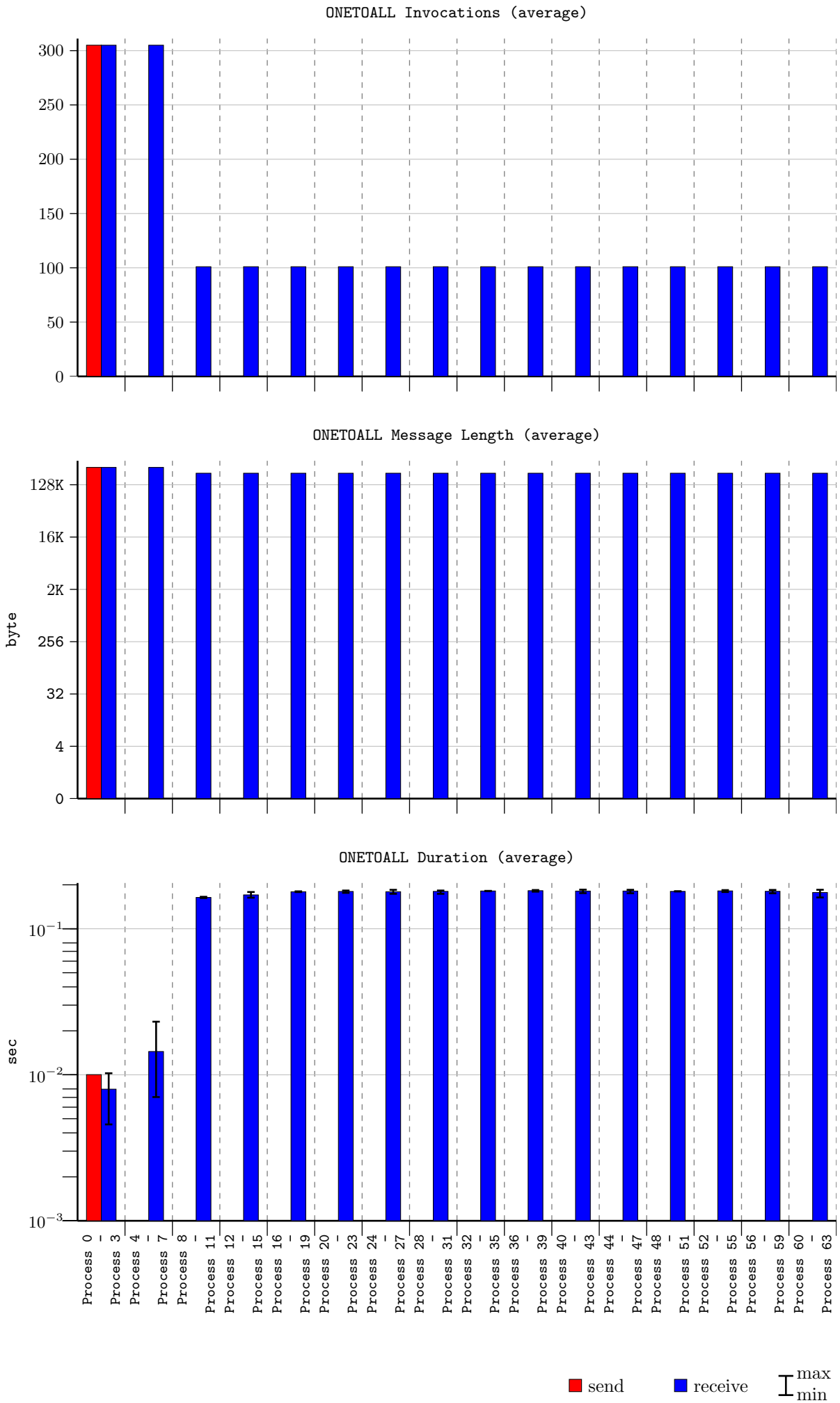
The message rate histogram shows the relation between message length (first dimension), the data transfer rate (second dimension) and the number of messages in each two- dimensional histogram bin. The number of invocations is color-coded according to the given scale. This chart allows to assess the message size (pre-determined by the target application) and their speed (observed during event trace recording).

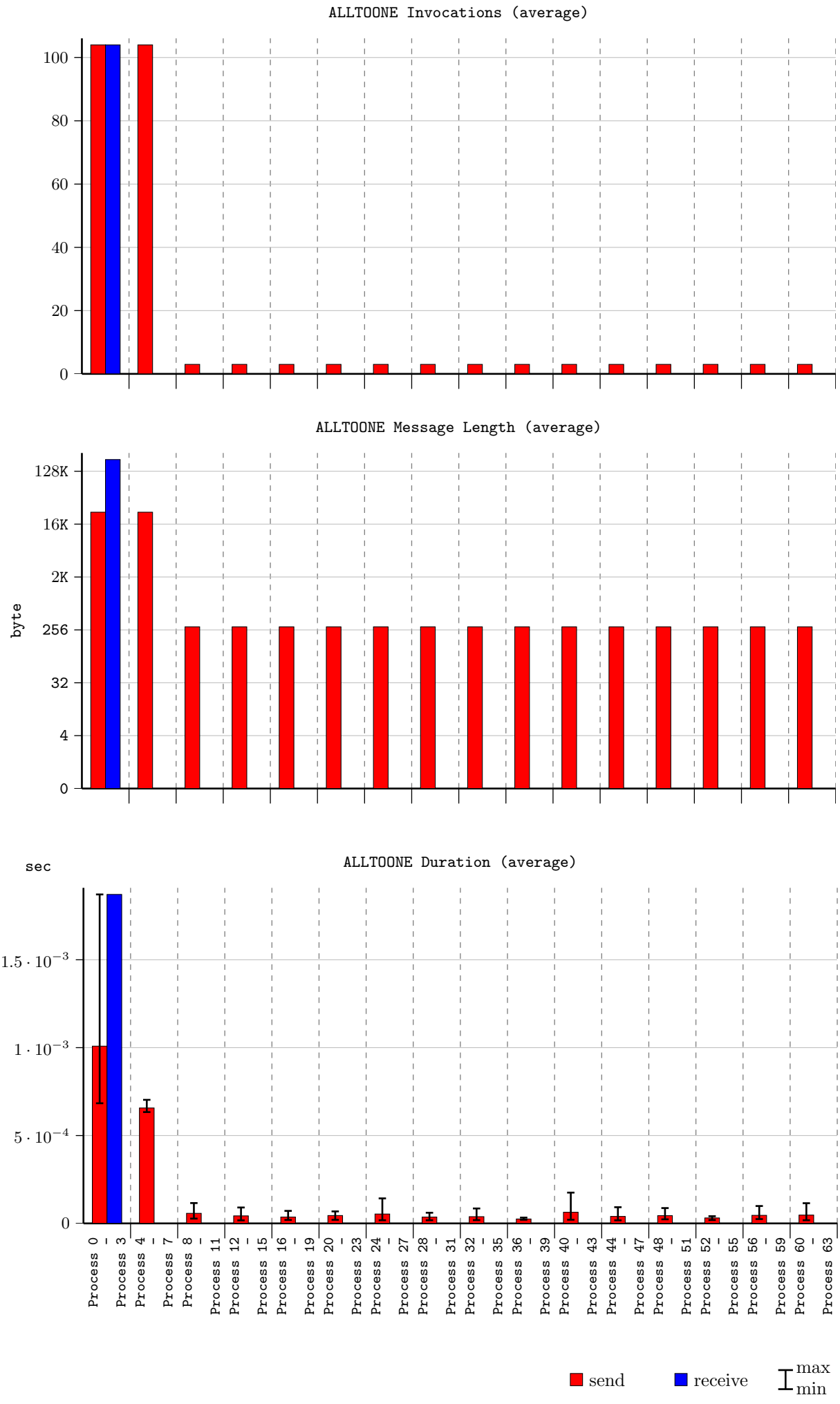
The following charts show statistics for the four kinds of collective communications: barriers, all-to-one, one-to-all and all-to-all. Like in the point-to-point communication charts, for processes or groups the sender values (red) and receiver values (blue) are distinguished where appropriate. The statistics include invocations, message lengths and durations. If the processes are grouped, average, minimum and maximum values per process are shown for every groups. The collective communication charts allow to judge the total distribution of collective communication load.

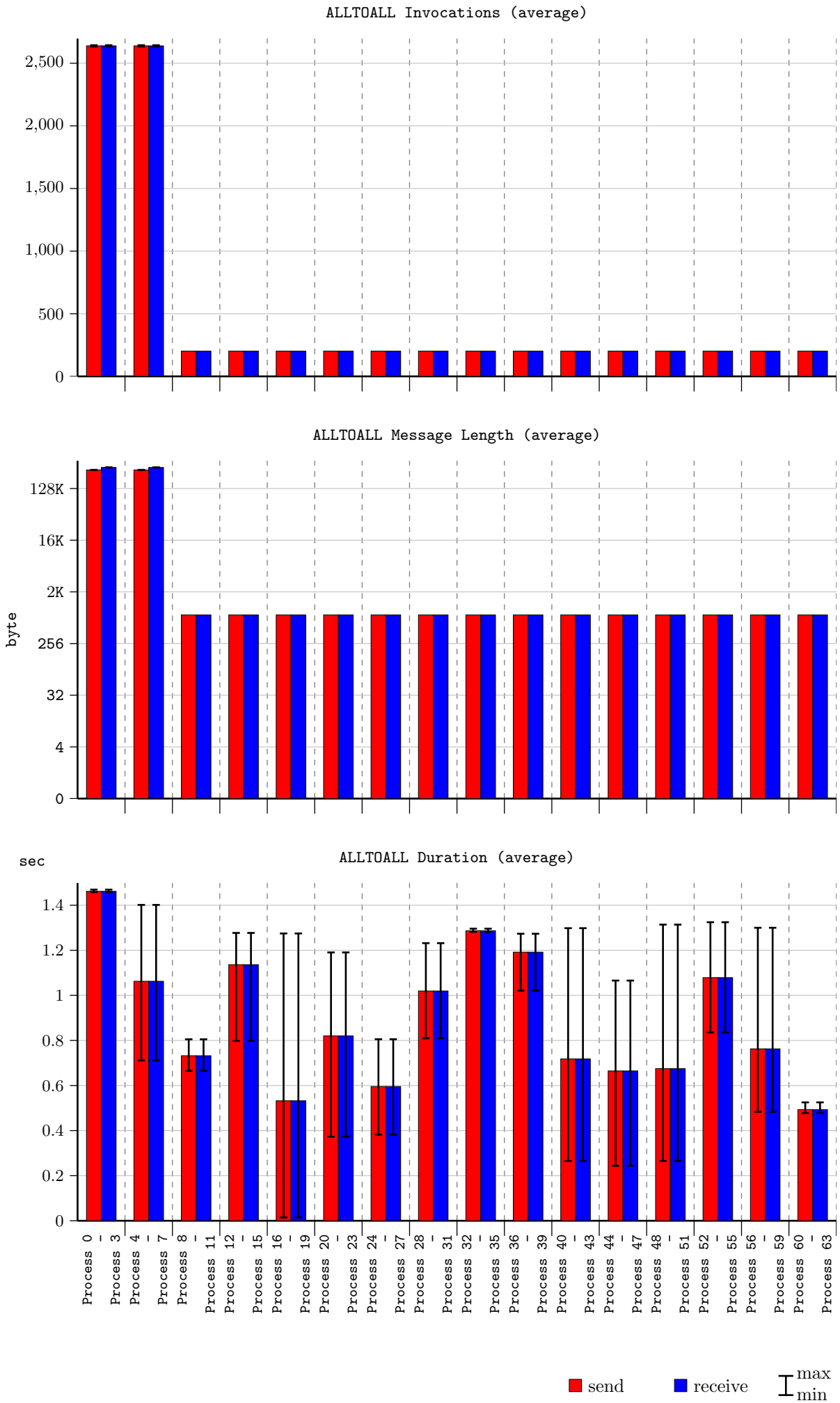




max
min


ONETOALL Invocations (average)
300
250
200
150
100
50
0
ONETOALL Message Length (average)
byte
128K
16K
2K
256
32
4
0
ONETOALL Duration (average)
sec
10^{-1}
10^{-2}
10^{-3}
Process 0 - Process 3
Process 4 - Process 7
Process 8 - Process 11
Process 12 - Process 15
Process 16 - Process 19
Process 20 - Process 23
Process 24 - Process 27
Process 28 - Process 31
Process 32 - Process 35
Process 36 - Process 39
Process 40 - Process 43
Process 44 - Process 47
Process 48 - Process 51
Process 52 - Process 55
Process 56 - Process 59
Process 60 - Process 63
send
receive
max
min


ALLTOONE Invocations (average)
100
80
60
40
20
0
ALLTOONE Message Length (average)
byte
128K
16K
2K
256
32
4
0
ALLTOONE Duration (average)
sec
1.5 · 10−3
1 · 10−3
5 · 10−4
0
Process 0 - Process 3
Process 4 - Process 7
Process 8 - Process 11
Process 12 - Process 15
Process 16 - Process 19
Process 20 - Process 23
Process 24 - Process 27
Process 28 - Process 31
Process 32 - Process 35
Process 36 - Process 39
Process 40 - Process 43
Process 44 - Process 47
Process 48 - Process 51
Process 52 - Process 55
Process 56 - Process 59
Process 60 - Process 63
send
receive
max
min


ALLTOALL Invocations (average)
2,500
2,000
1,500
1,000
500
0
ALLTOALL Message Length (average)
byte
128K
16K
2K
256
32
4
0
ALLTOALL Duration (average)
sec
1.4
1.2
1
0.8
0.6
0.4
0.2
0
Process 0 - Process 3
Process 4 - Process 7
Process 8 - Process 11
Process 12 - Process 15
Process 16 - Process 19
Process 20 - Process 23
Process 24 - Process 27
Process 28 - Process 31
Process 32 - Process 35
Process 36 - Process 39
Process 40 - Process 43
Process 44 - Process 47
Process 48 - Process 51
Process 52 - Process 55
Process 56 - Process 59
Process 60 - Process 63
send
receive
max
min