HLRN 04.11.2020

Intel® Trace Analyzer and Collector

Dr. Heinrich Bockhorst

Notices & Disclaimers

Intel technologies may require enabled hardware, software or service activation. Learn more at intel.com or from the OEM or retailer.

Your costs and results may vary.

Intel does not control or audit third-party data. You should consult other sources to evaluate accuracy.

Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice Revision #20110804. https://software.intel.com/en-us/articles/optimization-notice

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors.

Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. See backup for configuration details. For more complete information about performance and benchmark results, visit www.intel.com/benchmarks.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See configuration disclosure for details. No product or component can be absolutely secure.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

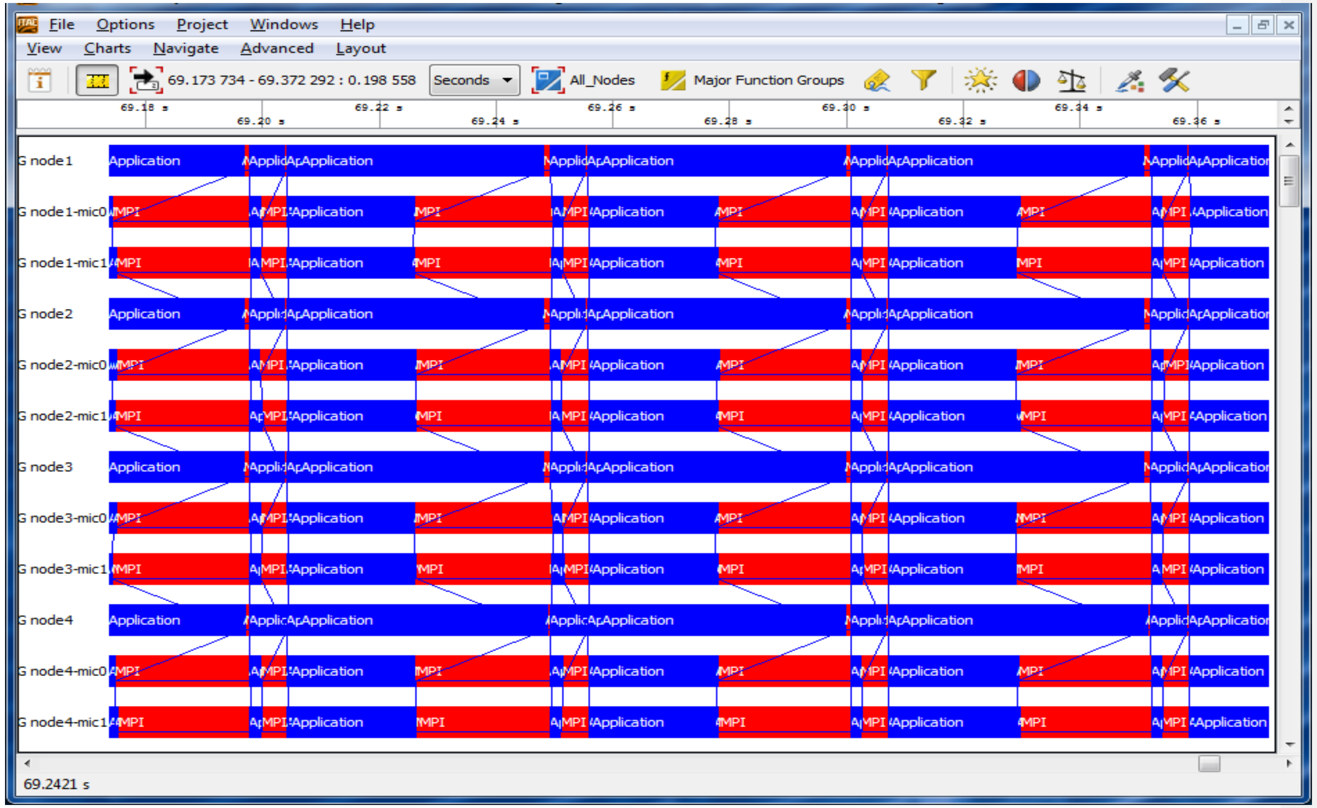
© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

Agenda

- Overview and Usage
- Flat profile
- Event timeline
- Message Profile
- Idealizer

Intel® Trace Analyzer & Collector Overview

- ## Helps Developers
  - Visualize & understand parallel application behavior
  - Evaluate profiling statistics & load balancing
  - Identify communication hotspots



- ## Features
  - Event-based approach
  - Low overhead
  - Excellent scalability
  - Powerful aggregation & filtering functions
  - Idealizer
  - Scalable

How to Use Intel® Trace Analyzer and Collector

It’s Easy…

Step 1

Run your binary and create a tracefile:

```
$ mpirun -trace -n 2 ./test
```

Alternative for slurm and mpirun:

```
$ export LD_PRELOAD=libVT.so
```

```
Run application as usual
```

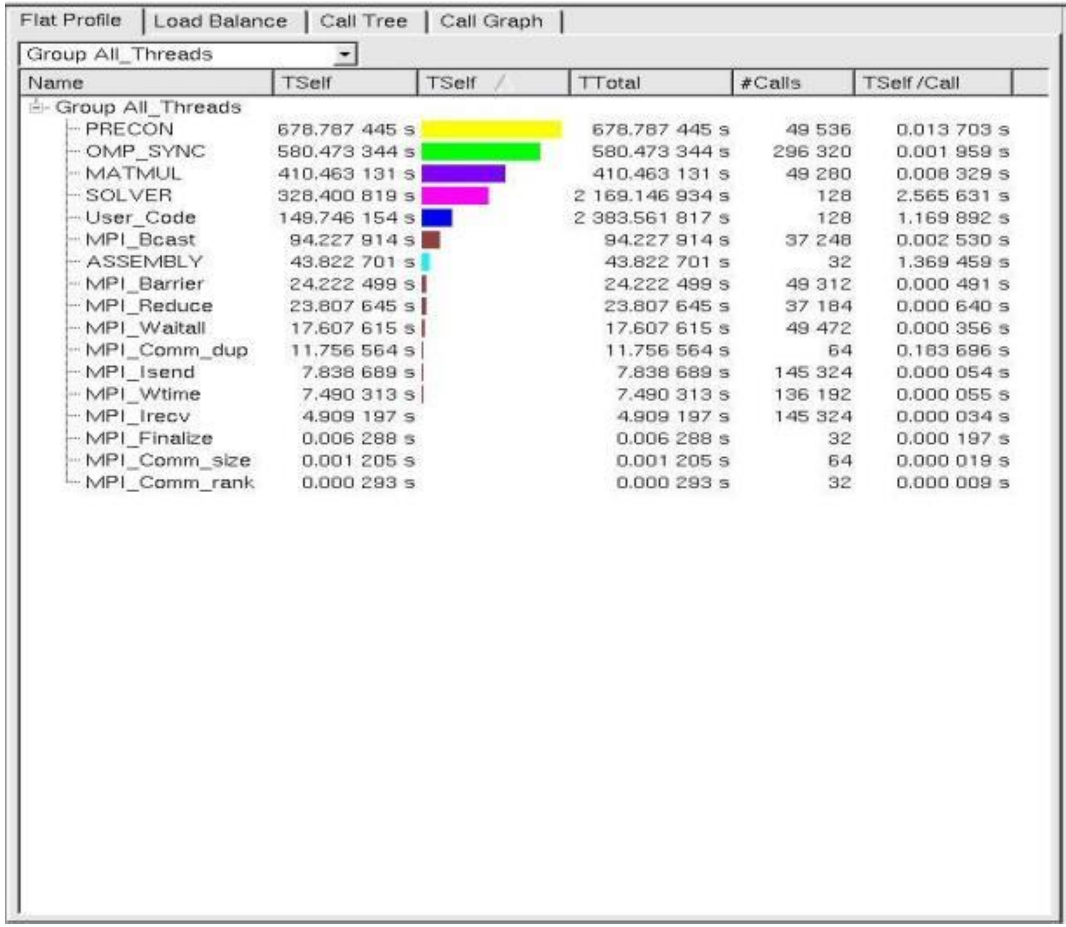
Step 2

View the results:

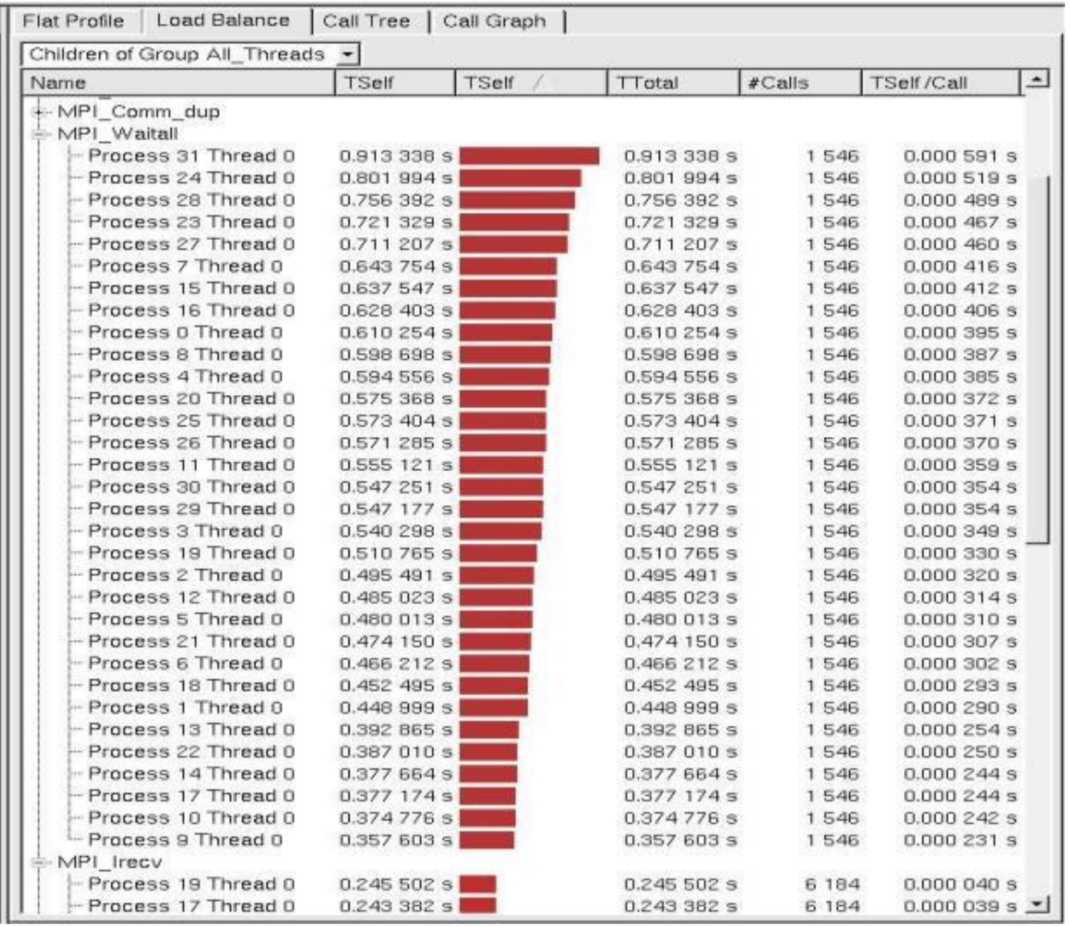
```
$ traceanalyzer &
```

Flat Function Profile

Statistics About Functions

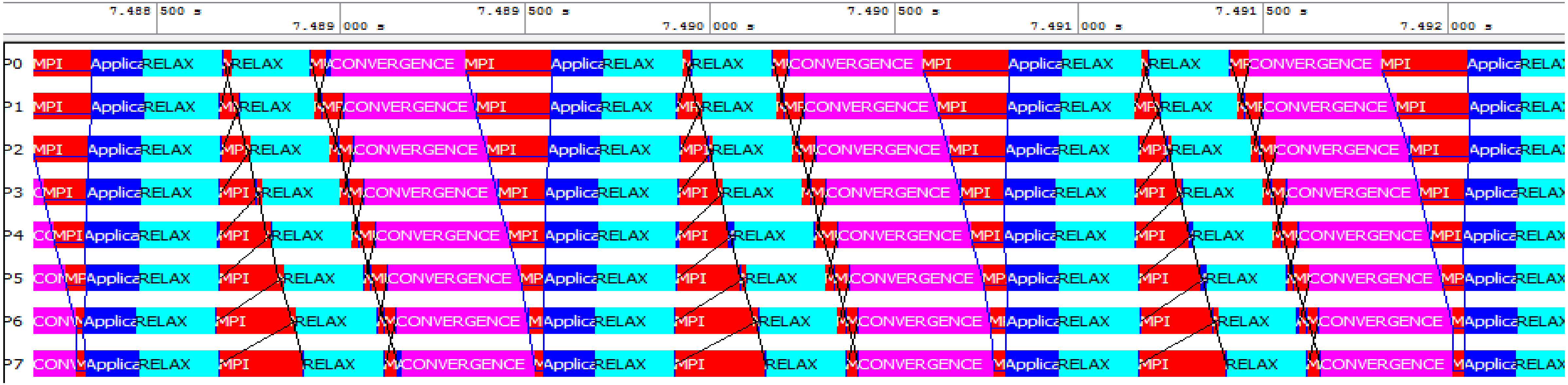


Name	TSelf	TSelf /	TTotal	#Calls	TSelf /Call
Group All_Threads					
PRECON	678.787 445 s		678.787 445 s	49 536	0.013 703 s
OMP_SYNC	580.473 344 s		580.473 344 s	296 320	0.001 959 s
MATMUL	410.463 131 s		410.463 131 s	49 280	0.008 329 s
SOLVER	328.400 819 s		2 169.146 934 s	128	2.565 631 s
User_Code	149.746 154 s		2 383.561 817 s	128	1.169 892 s
MPI_Bcast	94.227 914 s		94.227 914 s	37 248	0.002 530 s
ASSEMBLY	43.822 701 s		43.822 701 s	32	1.369 459 s
MPI_Barrier	24.222 499 s		24.222 499 s	49 312	0.000 491 s
MPI_Reduce	23.807 645 s		23.807 645 s	37 184	0.000 640 s
MPI_Waitall	17.607 615 s		17.607 615 s	49 472	0.000 356 s
MPI_Comm_dup	11.756 564 s		11.756 564 s	64	0.183 696 s
MPI_Isend	7.838 689 s		7.838 689 s	145 324	0.000 054 s
MPI_Wtime	7.490 313 s		7.490 313 s	136 192	0.000 055 s
MPI_Irecv	4.909 197 s		4.909 197 s	145 324	0.000 034 s
MPI_Finalize	0.006 288 s		0.006 288 s	32	0.000 197 s
MPI_Comm_size	0.001 205 s		0.001 205 s	64	0.000 019 s
MPI_Comm_rank	0.000 293 s		0.000 293 s	32	0.000 009 s



Name	TSelf	TSelf /	TTotal	#Calls	TSelf /Call
MPI_Comm_dup					
MPI_Waitall					
Process 31 Thread 0	0.913 338 s		0.913 338 s	1 546	0.000 591 s
Process 24 Thread 0	0.801 994 s		0.801 994 s	1 546	0.000 519 s
Process 28 Thread 0	0.756 392 s		0.756 392 s	1 546	0.000 489 s
Process 23 Thread 0	0.721 329 s		0.721 329 s	1 546	0.000 467 s
Process 27 Thread 0	0.711 207 s		0.711 207 s	1 546	0.000 460 s
Process 7 Thread 0	0.643 754 s		0.643 754 s	1 546	0.000 416 s
Process 15 Thread 0	0.637 547 s		0.637 547 s	1 546	0.000 412 s
Process 16 Thread 0	0.628 403 s		0.628 403 s	1 546	0.000 406 s
Process 0 Thread 0	0.610 254 s		0.610 254 s	1 546	0.000 395 s
Process 8 Thread 0	0.598 698 s		0.598 698 s	1 546	0.000 387 s
Process 4 Thread 0	0.594 556 s		0.594 556 s	1 546	0.000 385 s
Process 20 Thread 0	0.575 368 s		0.575 368 s	1 546	0.000 372 s
Process 25 Thread 0	0.573 404 s		0.573 404 s	1 546	0.000 371 s
Process 26 Thread 0	0.571 285 s		0.571 285 s	1 546	0.000 370 s
Process 11 Thread 0	0.555 121 s		0.555 121 s	1 546	0.000 359 s
Process 30 Thread 0	0.547 251 s		0.547 251 s	1 546	0.000 354 s
Process 29 Thread 0	0.547 177 s		0.547 177 s	1 546	0.000 354 s
Process 3 Thread 0	0.540 298 s		0.540 298 s	1 546	0.000 349 s
Process 19 Thread 0	0.510 765 s		0.510 765 s	1 546	0.000 330 s
Process 2 Thread 0	0.495 491 s		0.495 491 s	1 546	0.000 320 s
Process 12 Thread 0	0.485 023 s		0.485 023 s	1 546	0.000 314 s
Process 5 Thread 0	0.480 013 s		0.480 013 s	1 546	0.000 310 s
Process 21 Thread 0	0.474 150 s		0.474 150 s	1 546	0.000 307 s
Process 6 Thread 0	0.466 212 s		0.466 212 s	1 546	0.000 302 s
Process 18 Thread 0	0.452 495 s		0.452 495 s	1 546	0.000 293 s
Process 1 Thread 0	0.448 999 s		0.448 999 s	1 546	0.000 290 s
Process 13 Thread 0	0.392 865 s		0.392 865 s	1 546	0.000 254 s
Process 22 Thread 0	0.387 010 s		0.387 010 s	1 546	0.000 250 s
Process 14 Thread 0	0.377 664 s		0.377 664 s	1 546	0.000 244 s
Process 17 Thread 0	0.377 174 s		0.377 174 s	1 546	0.000 244 s
Process 10 Thread 0	0.374 776 s		0.374 776 s	1 546	0.000 242 s
Process 9 Thread 0	0.357 603 s		0.357 603 s	1 546	0.000 231 s
MPI_Irecv					
Process 19 Thread 0	0.245 502 s		0.245 502 s	6 184	0.000 040 s
Process 17 Thread 0	0.243 382 s		0.243 382 s	6 184	0.000 039 s

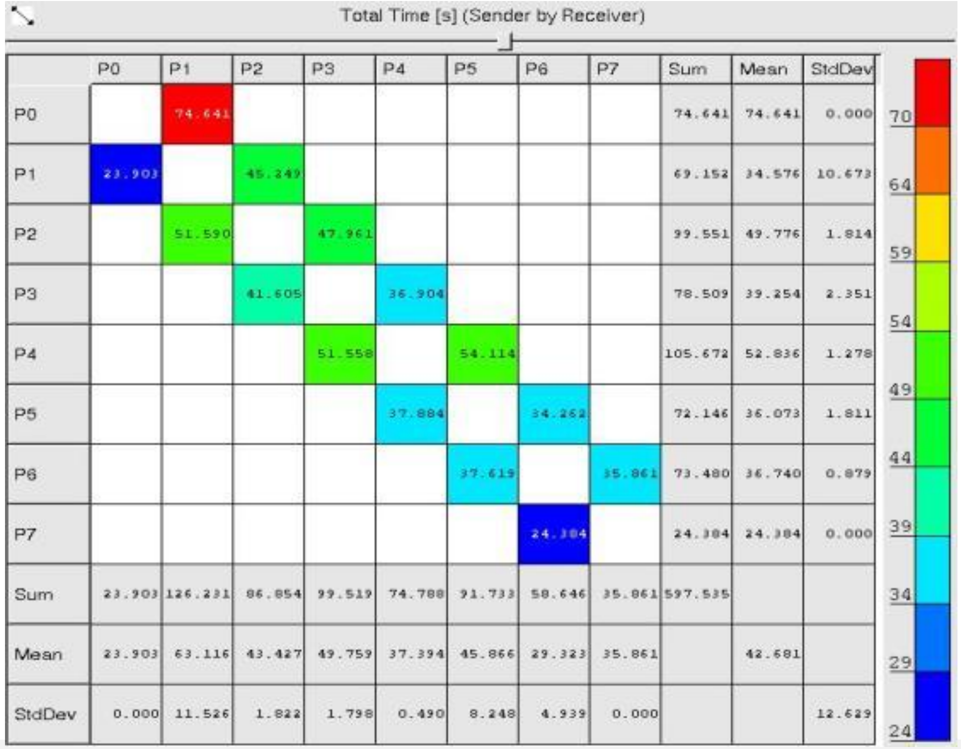
Event Timeline



- Get detailed impression of program structure.
- Display functions, messages, and collective operations for each process/thread along time axis.
- Retrieve detailed event information.

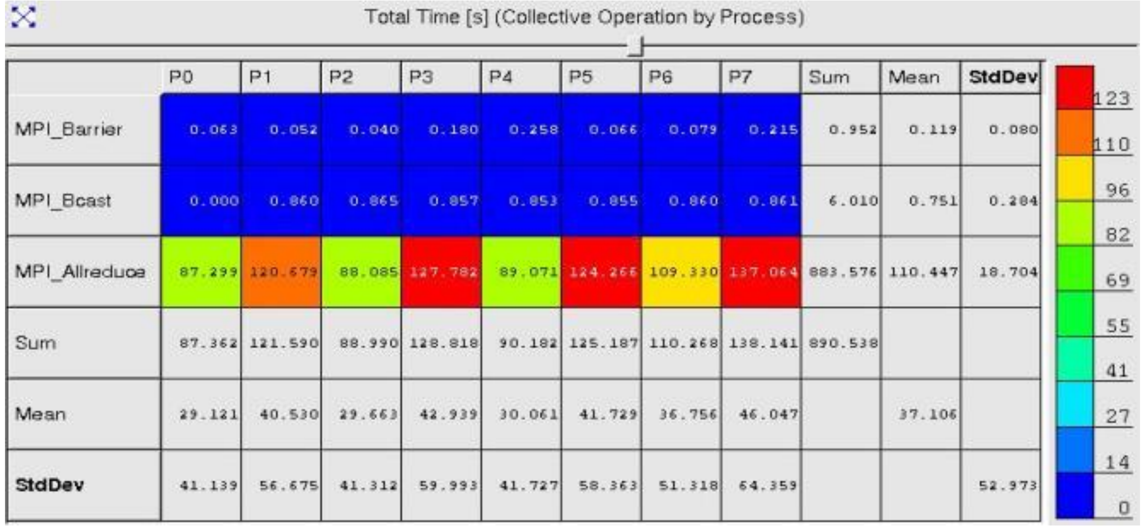
Message Profile

- Statistics about point-to-point or collective communication
- Generic matrix supports grouping by several attributes in each dimension: Sender, receiver, data volume per message, tag, communicator, type
- Available attributes: Count, bytes transferred, time, transfer rate



Total Time [s] (Sender by Receiver)

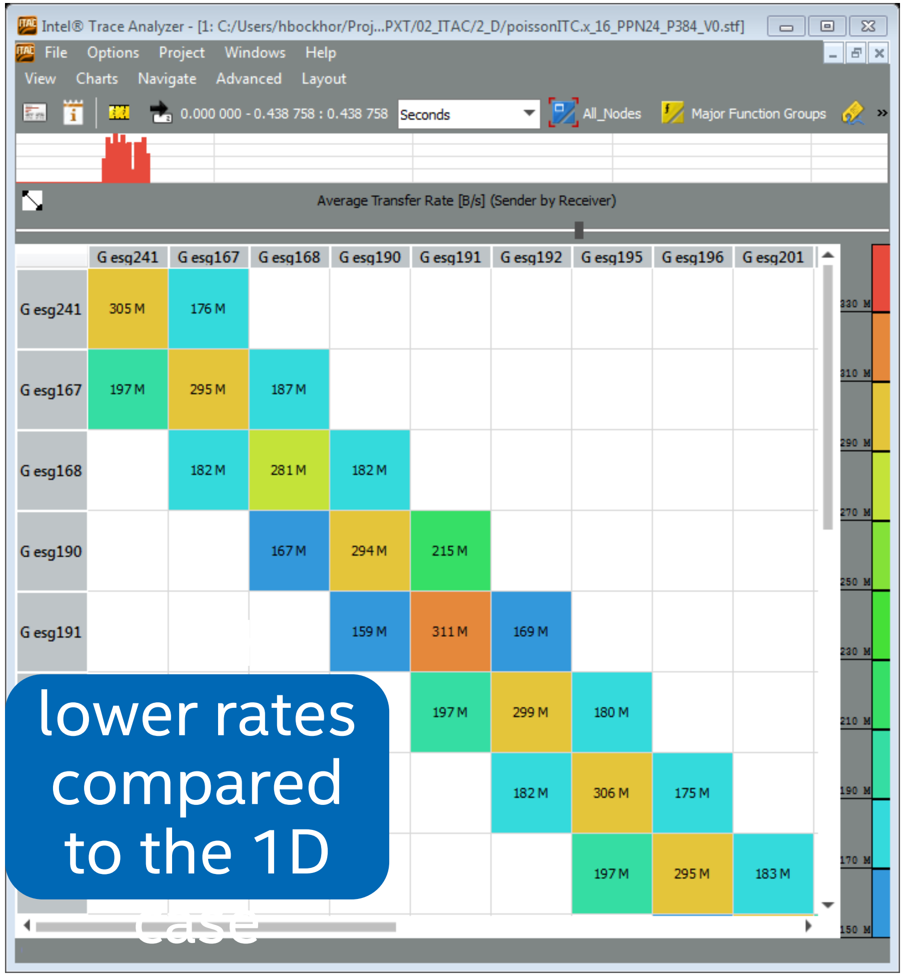
	P0	P1	P2	P3	P4	P5	P6	P7	Sum	Mean	StdDev
P0		74.641							74.641	74.641	0.000
P1	23.903		45.249						69.152	34.576	10.673
P2		51.590		47.961					99.551	49.776	1.814
P3			41.605		36.904				78.509	39.254	2.351
P4				51.558		54.114			105.672	52.836	1.278
P5					37.884		34.262		72.146	36.073	1.811
P6						37.619		35.861	73.480	36.740	0.879
P7							24.384		24.384	24.384	0.000
Sum	23.903	126.231	86.854	99.519	74.788	91.733	58.646	35.861	597.535		
Mean	23.903	63.116	43.427	49.759	37.394	45.866	29.323	35.861		42.681	
StdDev	0.000	11.526	1.822	1.798	0.490	8.248	4.939	0.000			12.629



Total Time [s] (Collective Operation by Process)

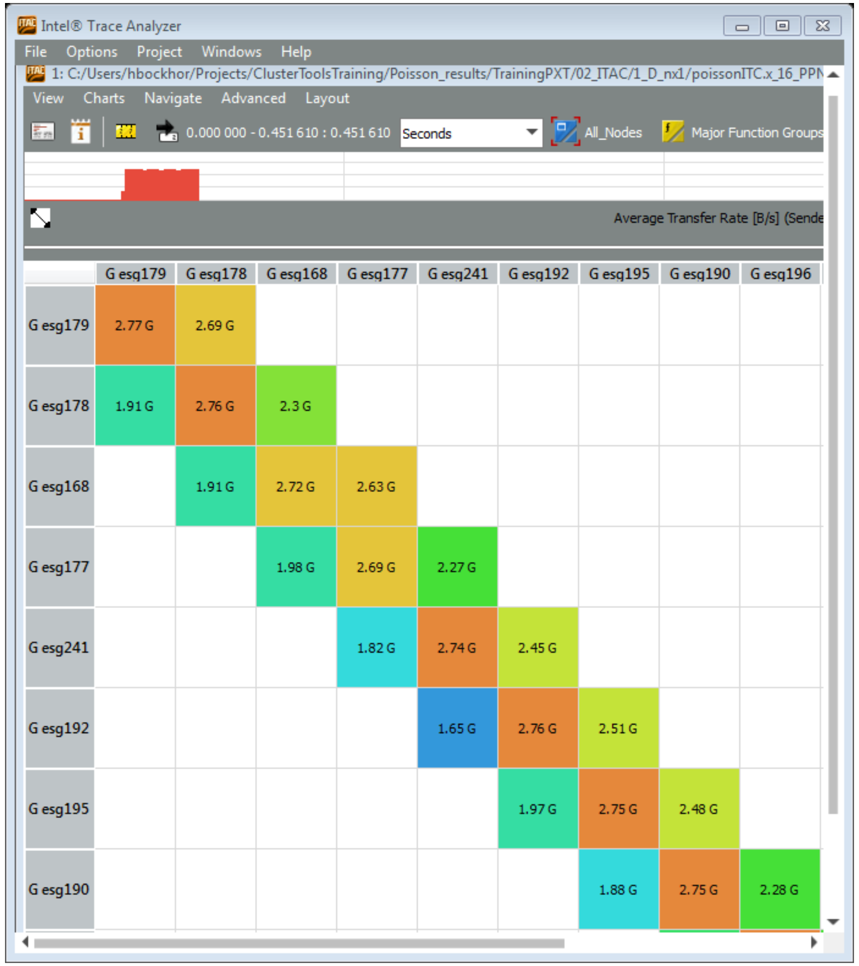
	P0	P1	P2	P3	P4	P5	P6	P7	Sum	Mean	StdDev
MPI_Barrier	0.063	0.052	0.040	0.180	0.258	0.066	0.079	0.215	0.952	0.119	0.080
MPI_Bcast	0.000	0.860	0.865	0.857	0.853	0.855	0.860	0.861	6.010	0.751	0.284
MPI_Allreduce	87.299	120.679	88.085	127.782	89.071	124.266	109.330	137.064	883.576	110.447	18.704
Sum	87.362	121.590	88.990	128.818	90.182	125.187	110.268	138.141	890.538		
Mean	29.121	40.530	29.663	42.939	30.061	41.729	36.756	46.047		37.106	
StdDev	41.139	56.675	41.312	59.993	41.727	58.363	51.318	64.359			52.973

Average Rate: 2D vs. 1D distribution


Intel® Trace Analyzer - [1: C:/Users/hbockhor/Proj...PXT/02_ITAC/2_D/poissonITC.x_16_PPN24_P384_V0.stf]
File Options Project Windows Help
View Charts Navigate Advanced Layout
0.000 000 - 0.438 758 : 0.438 758
Seconds
All_Nodes
Major Function Groups
Average Transfer Rate [B/s] (Sender by Receiver)
330 M
310 M
290 M
270 M
250 M
230 M
210 M
190 M
170 M
150 M

	G esg241	G esg167	G esg168	G esg190	G esg191	G esg192	G esg195	G esg196	G esg201
G esg241	305 M	176 M							
G esg167	197 M	295 M	187 M						
G esg168		182 M	281 M	182 M					
G esg190			167 M	294 M	215 M				
G esg191				159 M	311 M	169 M			
					197 M	299 M	180 M		
						182 M	306 M	175 M	
							197 M	295 M	183 M

lower rates compared to the 1D case


Intel® Trace Analyzer
File Options Project Windows Help
1: C:/Users/hbockhor/Projects/ClusterToolsTraining/Poisson_results/TrainingPXT/02_ITAC/1_D_nx1/poissonITC.x_16_PPN
View Charts Navigate Advanced Layout
0.000 000 - 0.451 610 : 0.451 610
Seconds
All_Nodes
Major Function Groups
Average Transfer Rate [B/s] (Sende

	G esg179	G esg178	G esg168	G esg177	G esg241	G esg192	G esg195	G esg190	G esg196
G esg179	2.77 G	2.69 G							
G esg178	1.91 G	2.76 G	2.3 G						
G esg168		1.91 G	2.72 G	2.63 G					
G esg177			1.98 G	2.69 G	2.27 G				
G esg241				1.82 G	2.74 G	2.45 G			
G esg192					1.65 G	2.76 G	2.51 G		
G esg195						1.97 G	2.75 G	2.48 G	
G esg190							1.88 G	2.75 G	2.28 G

Improving Load Balance: Real-World Case

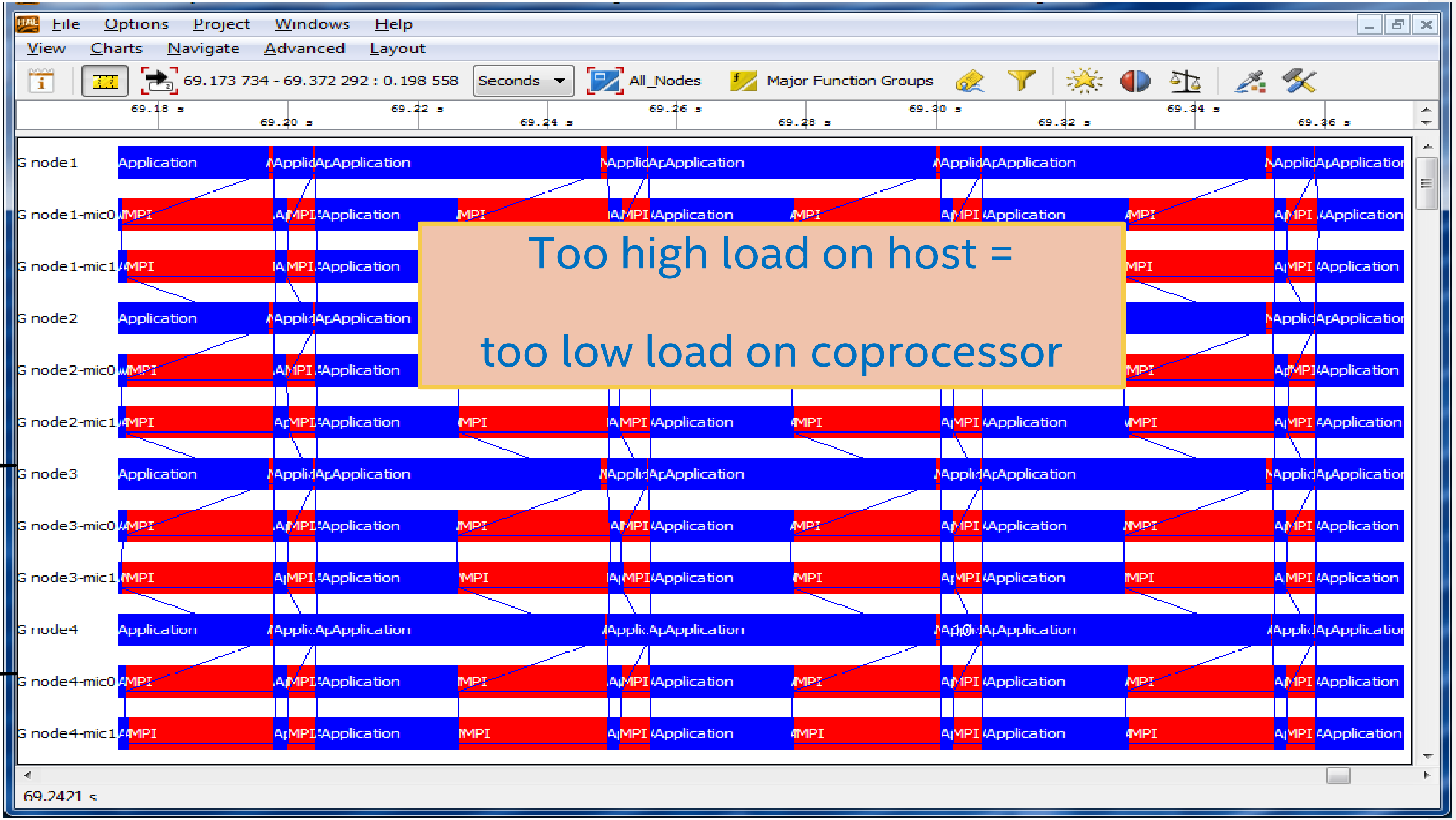
Collapsed data per node and coprocessor card

Host
16 MPI procs x

1 OpenMP* thread

Coprocessor
8 MPI procs x

28 OpenMP threads



Too high load on host =

too low load on coprocessor

Improving Load Balance: Real-World Case

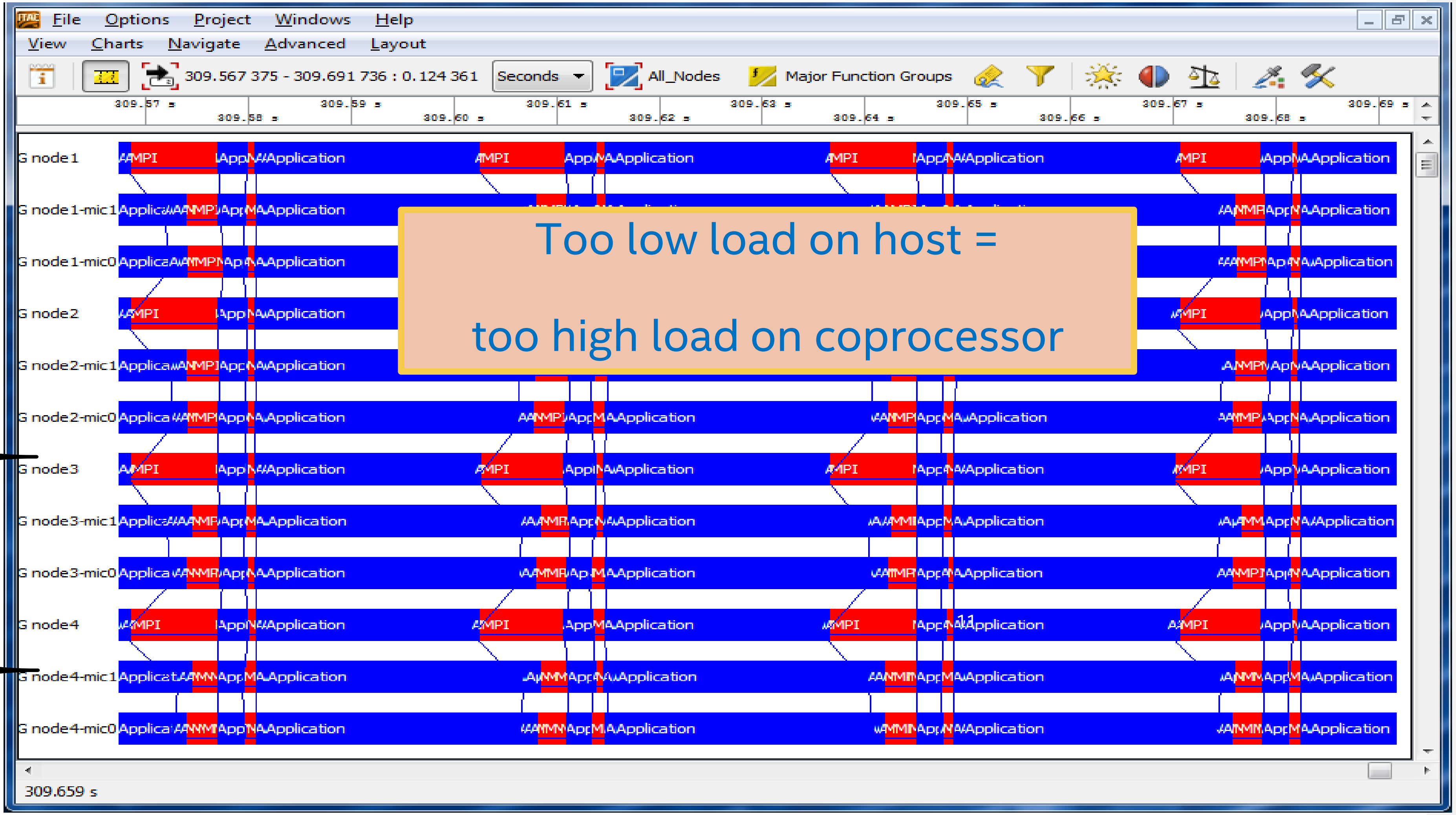
Collapsed data per node and coprocessor card

Host
16 MPI procs x

1 OpenMP* thread

Coprocessor
24 MPI procs x

8 OpenMP threads



Too low load on host =

too high load on coprocessor

Improving Load Balance: Real-World Case

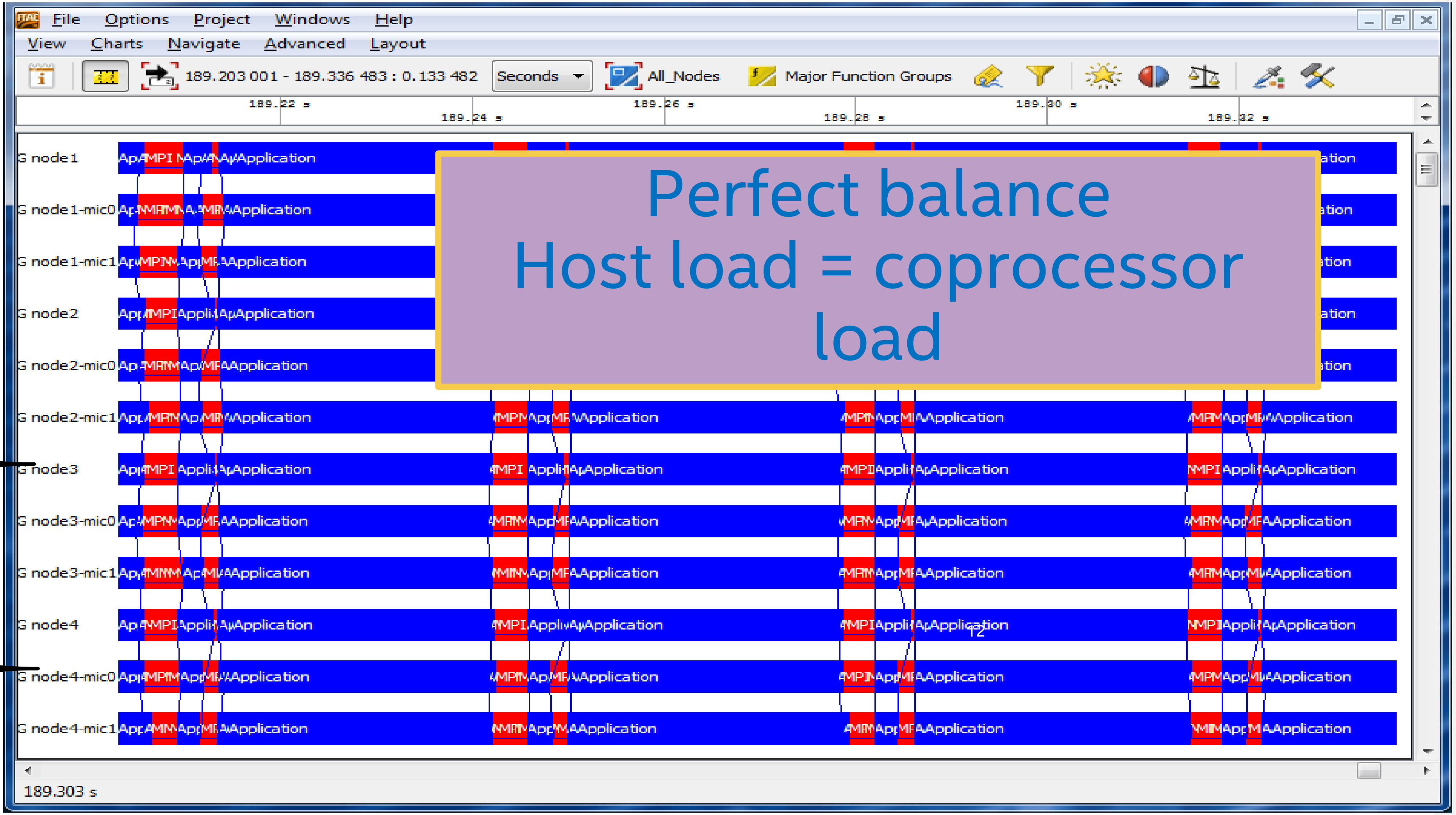
Collapsed data per node and coprocessor card

Host
16 MPI procs x

1 OpenMP* thread

Coprocessor
16 MPI procs x

12 OpenMP thrds

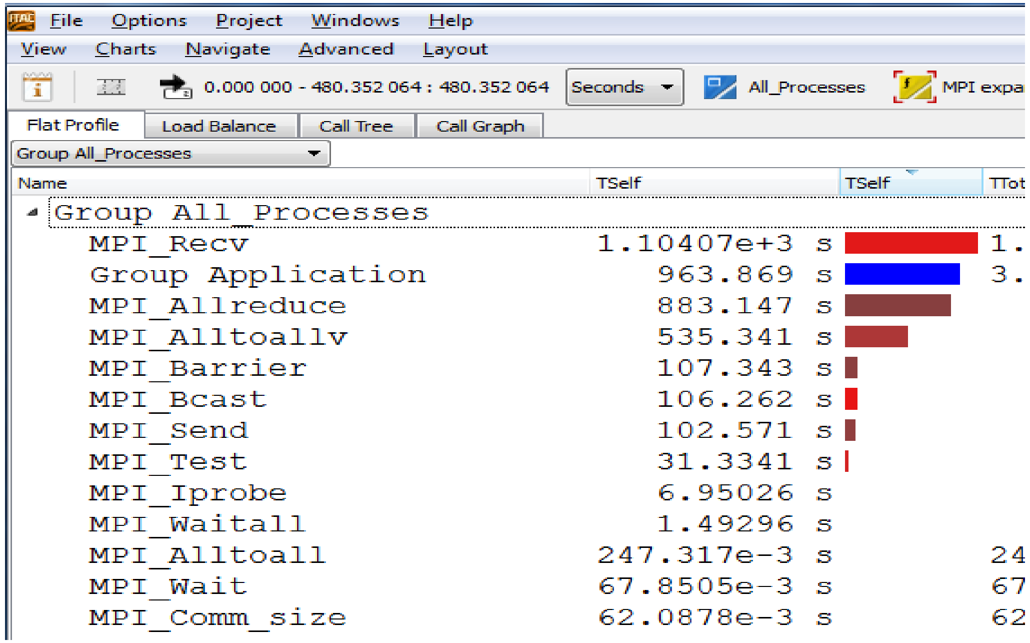


Perfect balance
Host load = coprocessor load

Ideal Interconnect Simulator (Idealizer)

Helps to Figure Out Application's Imbalance, Simulating its Behavior in the 'Ideal' Communication Environment

Real trace

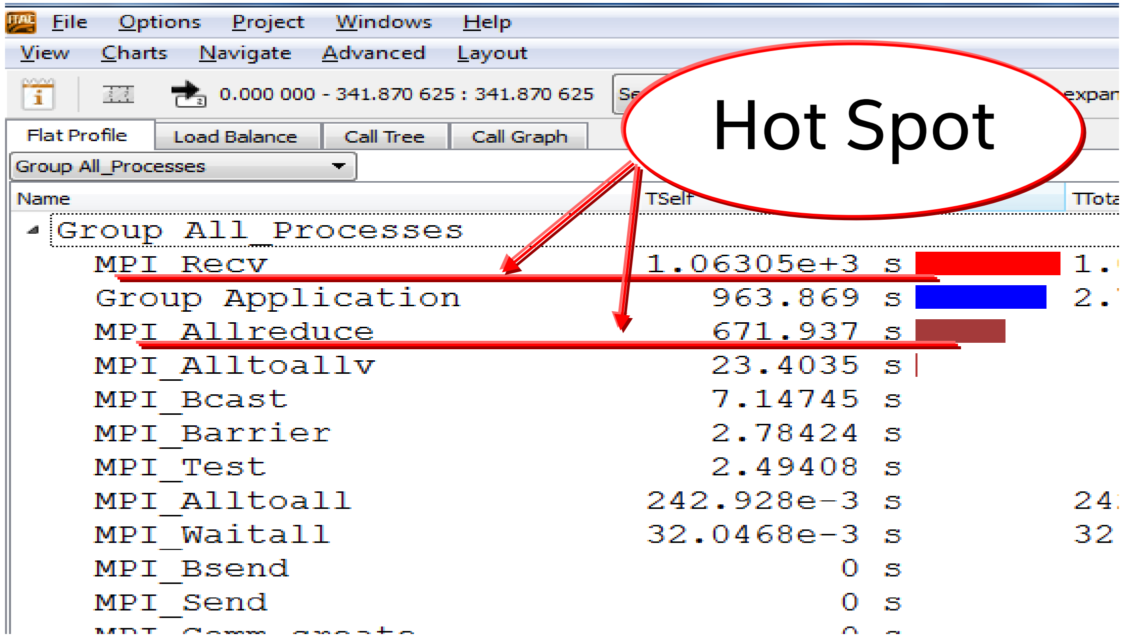

File Options Project Windows Help
View Charts Navigate Advanced Layout
Flat Profile
Load Balance
Call Tree
Call Graph
MPI expa

0.000 000 - 480.352 064 : 480.352 064 Seconds All_Processes

Group All_Processes

Name	TSelf	TTot
Group All_Processes		
MPI_Recv	1.10407e+3 s	1.
Group Application	963.869 s	3.
MPI_Allreduce	883.147 s	
MPI_Alltoallv	535.341 s	
MPI_Barrier	107.343 s	
MPI_Bcast	106.262 s	
MPI_Send	102.571 s	
MPI_Test	31.3341 s	
MPI_Iprobe	6.95026 s	
MPI_Waitall	1.49296 s	
MPI_Alltoall	247.317e-3 s	24
MPI_Wait	67.8505e-3 s	67
MPI_Comm_size	62.0878e-3 s	62

Ideal trace


File Options Project Windows Help
View Charts Navigate Advanced Layout
Flat Profile
Load Balance
Call Tree
Call Graph
Hot Spot

0.000 000 - 341.870 625 : 341.870 625

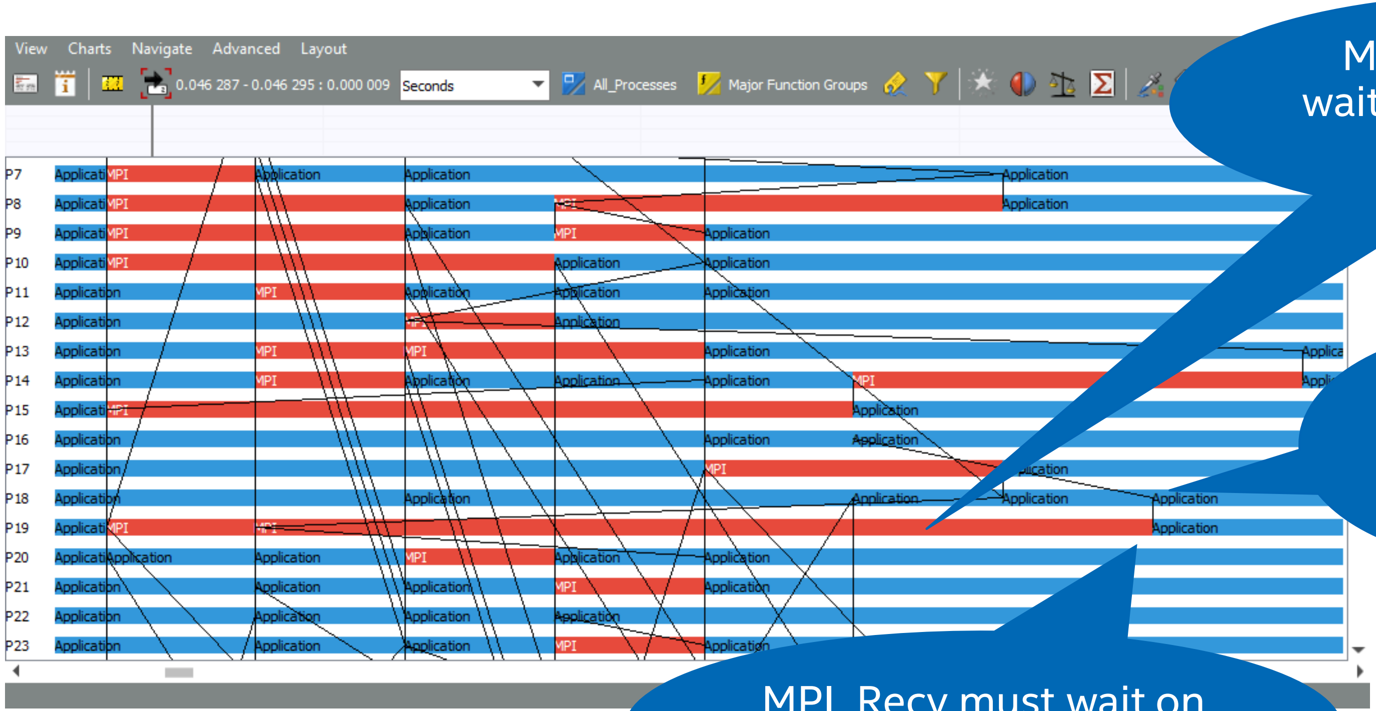
Group All_Processes

Name	TSelf	TTota
Group All_Processes		
MPI_Recv	1.06305e+3 s	1.
Group Application	963.869 s	2.
MPI_Allreduce	671.937 s	
MPI_Alltoallv	23.4035 s	
MPI_Bcast	7.14745 s	
MPI_Barrier	2.78424 s	
MPI_Test	2.49408 s	
MPI_Alltoall	242.928e-3 s	24
MPI_Waitall	32.0468e-3 s	32
MPI_Bsend	0 s	
MPI_Send	0 s	

Easy Way to Identify Application Bottlenecks

Waiting time due to dependencies

“Ideal” trace file



MPI _Recv is pure waiting time inside an ideal trace file

MPI_Send time shrinks to 0

MPI_Recv must wait on MPI_Send call

Summary

- ITAC generate traces with wallclock timing
- Shows how MPI algorithm works
- Idealizer generates ideal tracefile reveals algorithmic issues
- Trace files can become very large
- Application Performance Snapshot (APS) can scale higher

intel®