



SOFTWARE

+ ☐ 19.56 updatex
+ ☐ 399.70 updateien
+ ☐ 0.00 gene
+ ☐ 0.00 <<iteration loop>>
+ ☐ 447.52 genbc



FAST SOLUTIONS

- ☒ PAPI_L1_ICM
- ☐ PAPI_L2_DCM
- ☒ PAPI_L2_ICM
- ☐ PAPI_L1_TCM

VAMPIR & VAMPIRTRACE INTRODUCTION AND OVERVIEW

Jens Doleschal, Andreas Knüpfer

ZIH, TU Dresden

jens.doleschal@tu-dresden.de

September 2009

- Introduction
- Event Trace Visualization
- Vampir & VampirServer
- The Vampir Displays
 - Timeline
 - Process Timeline with Performance Counters
 - Summary Display
 - Message Statistics
- VampirTrace
 - Instrumentation & Run-Time Measurement
- Conclusions

Why bother with performance analysis?

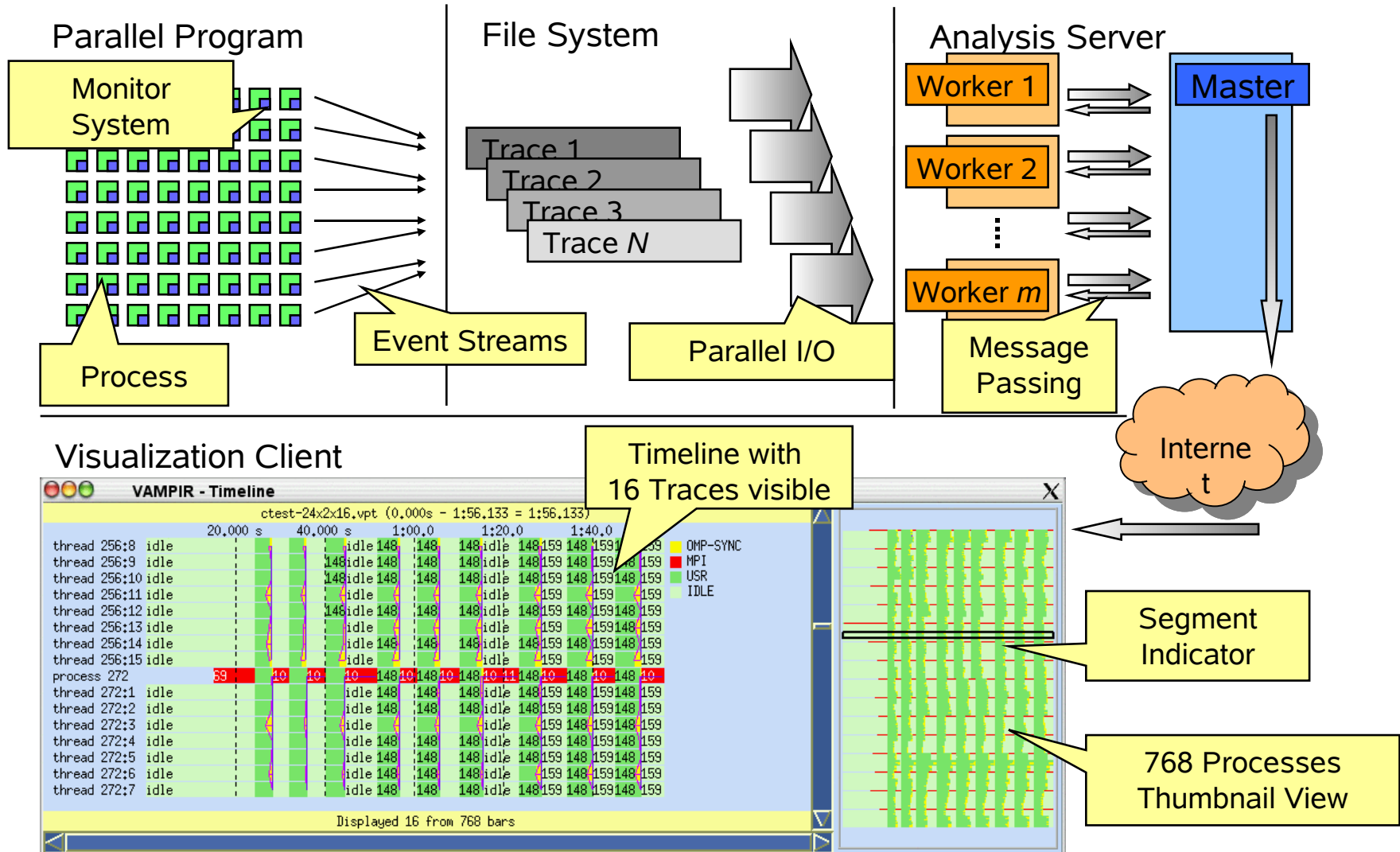
- well, why are you here after all?
- efficient usage of expensive and limited resources
- scalability to achieve next bigger simulation

Profiling and Tracing

- have an optimization phase
 - just like testing and debugging phase
- use tools!
- avoid *do-it-yourself-with-printf* solutions, really!

Trace Visualization

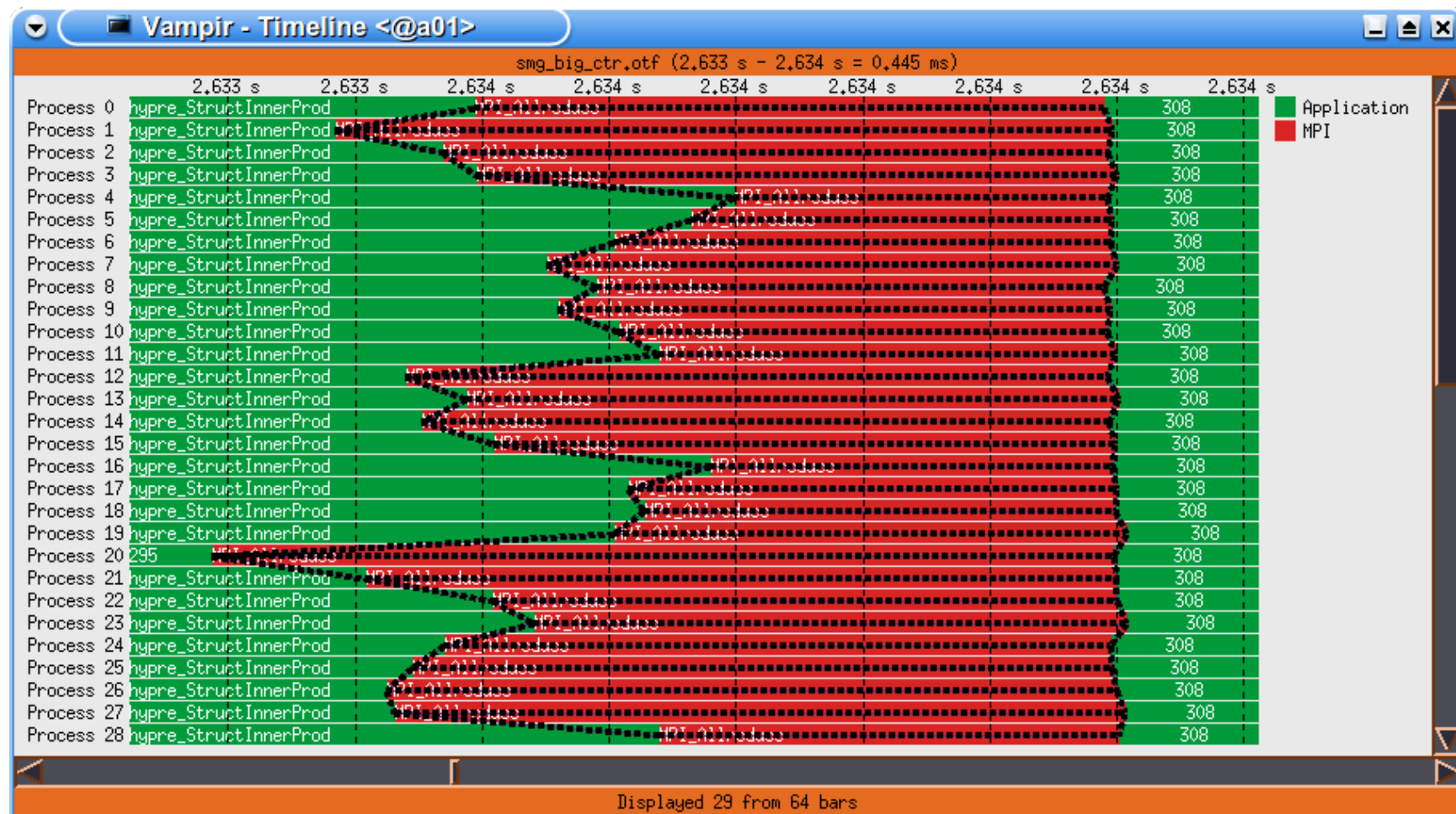
- alternative and supplement to automatic analysis
- show dynamic run-time behavior graphically
- provide statistics and performance metrics
 - Global timeline for parallel processes/threads
 - Process timeline plus performance counters
 - Statistics summary display
 - Message statistics
 - more
- interactive browsing, zooming, selecting
 - adapt statistics to zoom level (time interval)
 - also for very large and highly parallel traces



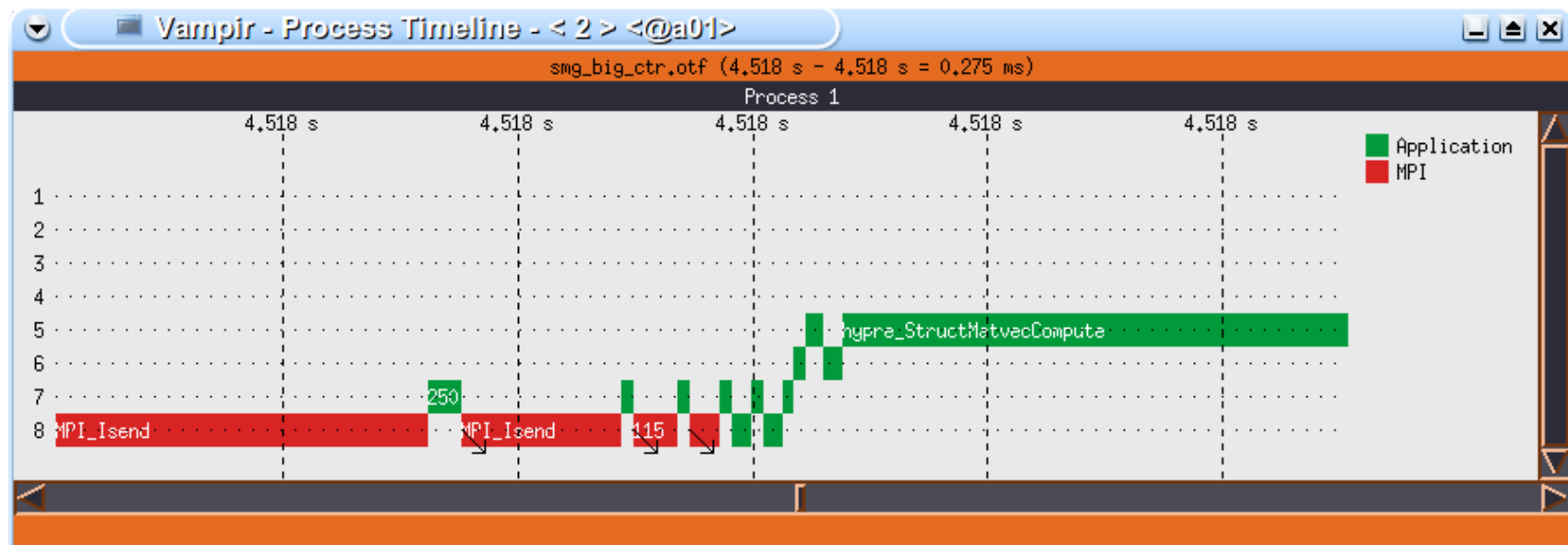
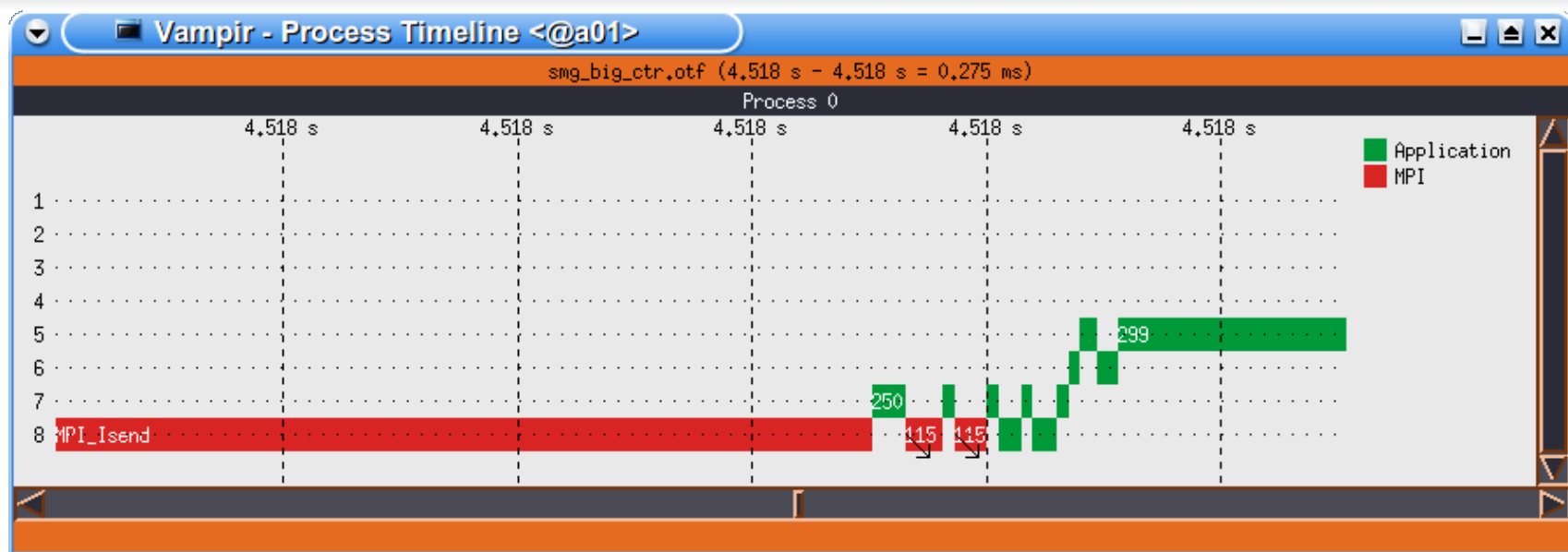
The main displays of Vampir:

- Global Timeline
- Process Timeline w/o Counters
- Statistic Summary
- Summary Timeline
- Message Statistics
- Collective Operation Statistics
- Counter Timeline
- Call Tree

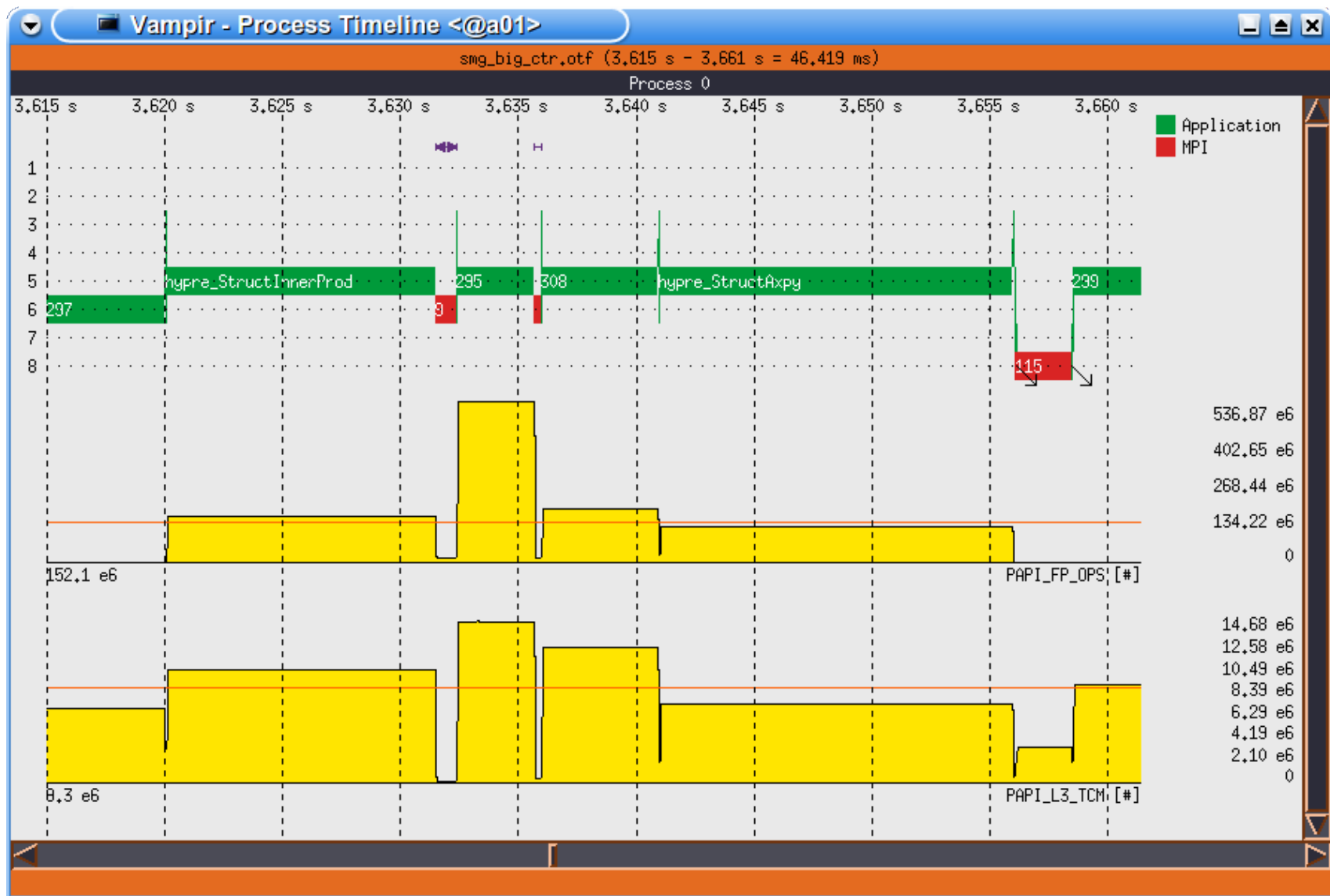
Vampir Global Timeline Display



Process Timeline Display



Process Timeline with Counters

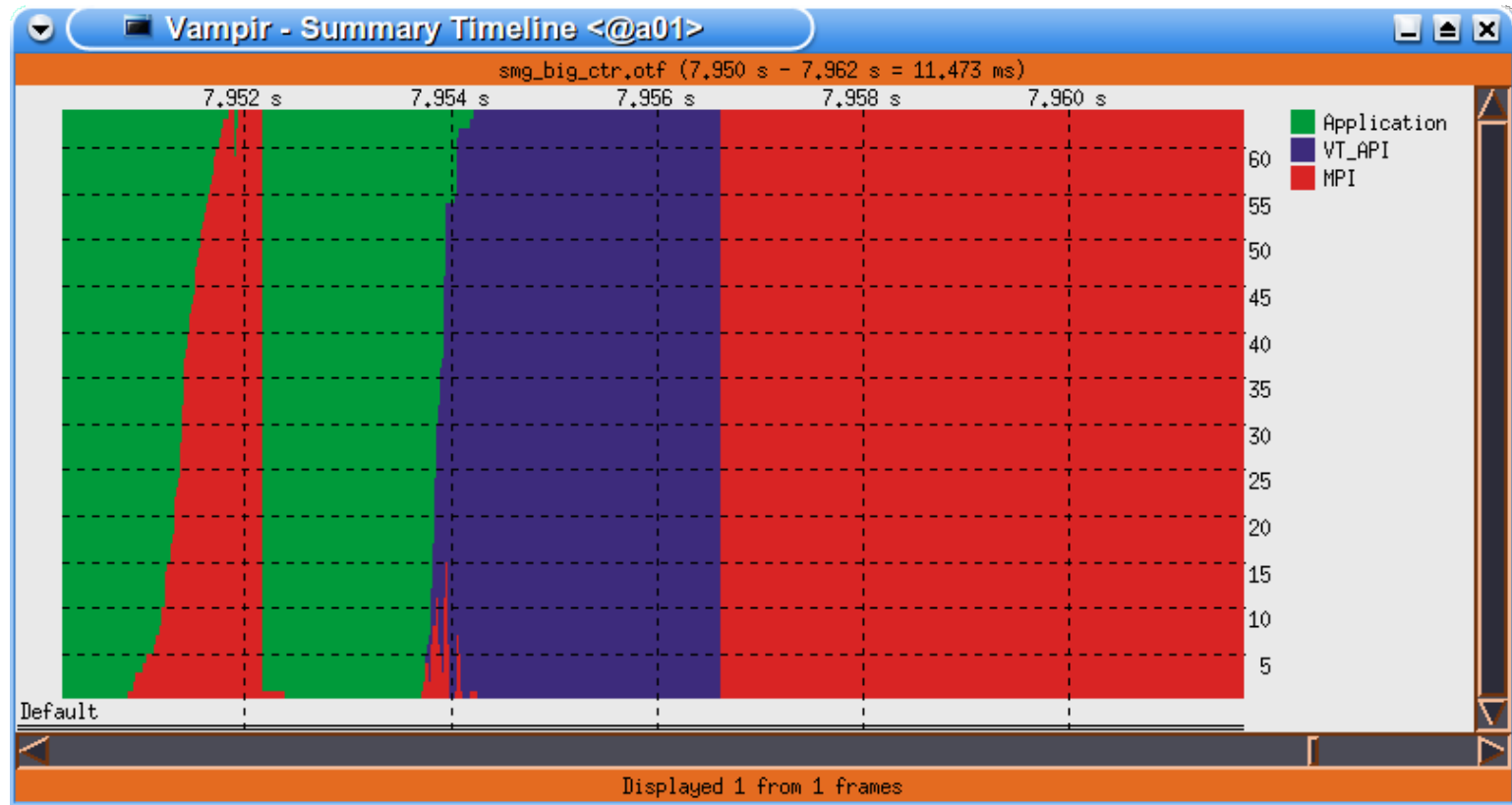


Vampir - Summary Chart - < 2 > <@a01>

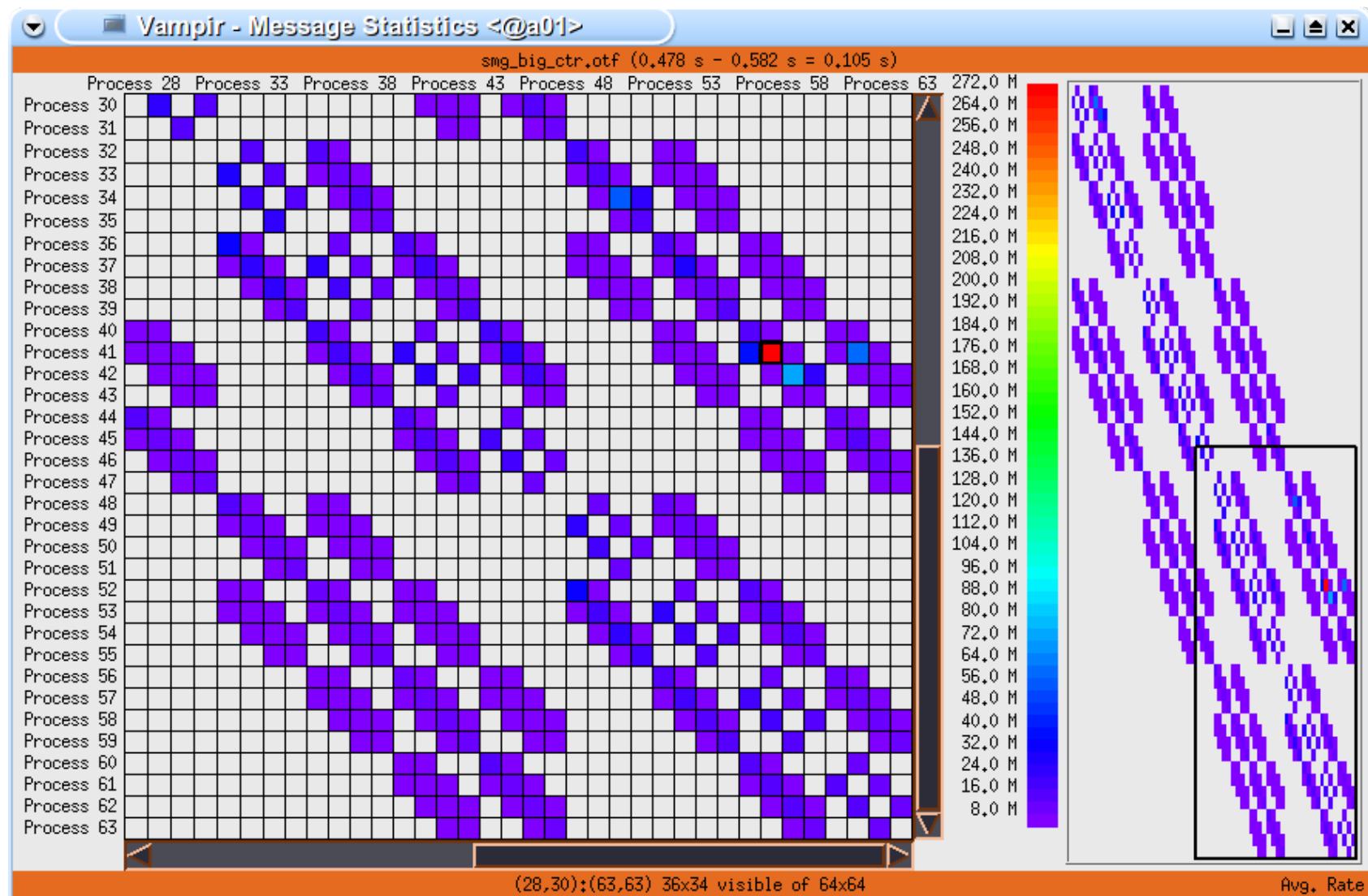
Name	Token	Value
hypre_StructMatvecCompute	[299]	2:15,199
hypre_StructAxp	[306]	2:10,744
hypre_StructInnerProd	[295]	1:14,087
MPI_Finalize	[82]	1:04,179
hypre_StructCopy	[297]	51,516 s
MPI_Waitall	[163]	20,135 s
hypre_StructVectorSetConstantVa	[303]	20,124 s
hypre_StructScale	[308]	15,580 s
MPI_Allreduce	[9]	13,283 s
MPI_Isend	[115]	9,010 s
hypre_StructMatrixSetBoxValues	[229]	8,455 s
sync	[2]	5,654 s
main	[184]	4,661 s
hypre_CAlloc	[186]	2,050 s
hypre_StructVectorSetBoxValues	[260]	1,827 s
hypre_StructMatrixInitializeDat	[224]	0,738 s
hypre_StructKrylovAxp	[305]	0,668 s
MPI_Init	[108]	0,436 s
hypre_StructKrylovCopyVector	[296]	0,221 s
hypre_StructKrylovMatvec	[298]	0,215 s
hypre_PCGSolve	[293]	0,212 s
MPI_Irecv	[113]	0,190 s
hypre_BoxGetSize	[227]	0,182 s
hypre_Free	[187]	0,169 s
hypre_InitializeCommunication	[250]	0,160 s
hypre_StructMatrixSetBoxValues	[229]	8,455 s

Sorted by Value Down All Symbols: Exclusive Times

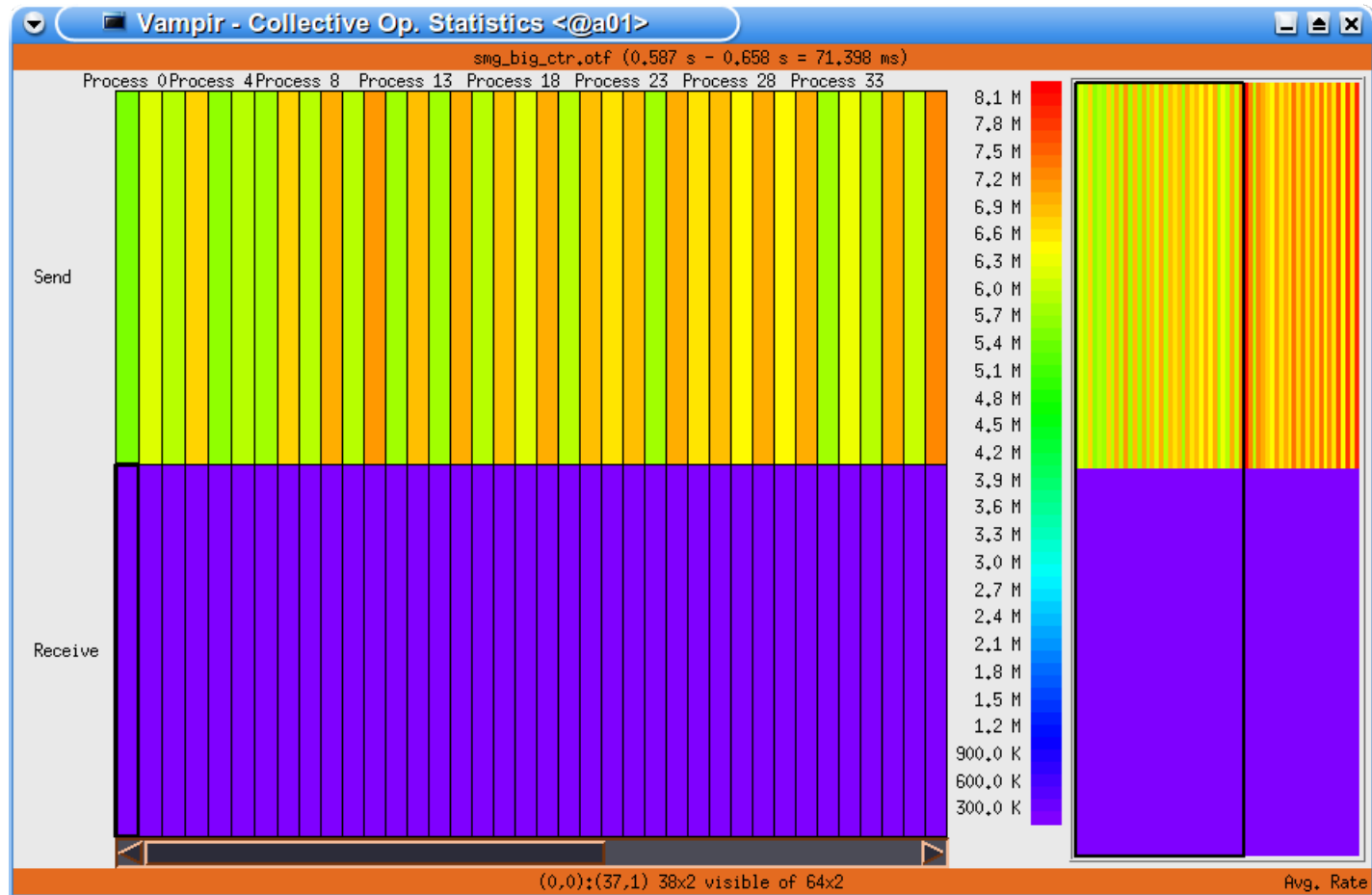
Summary Timeline Display



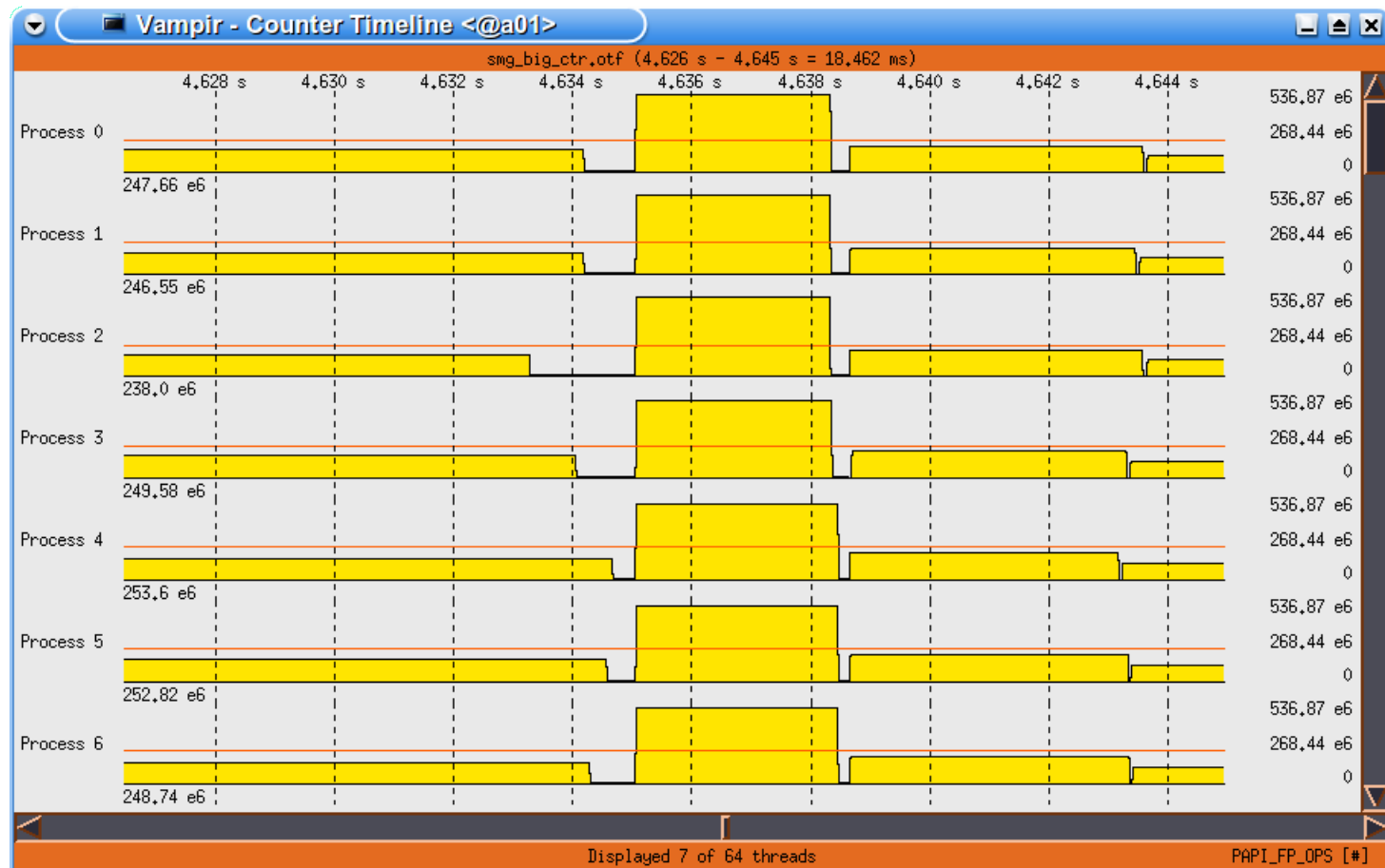
Message Statistics Display

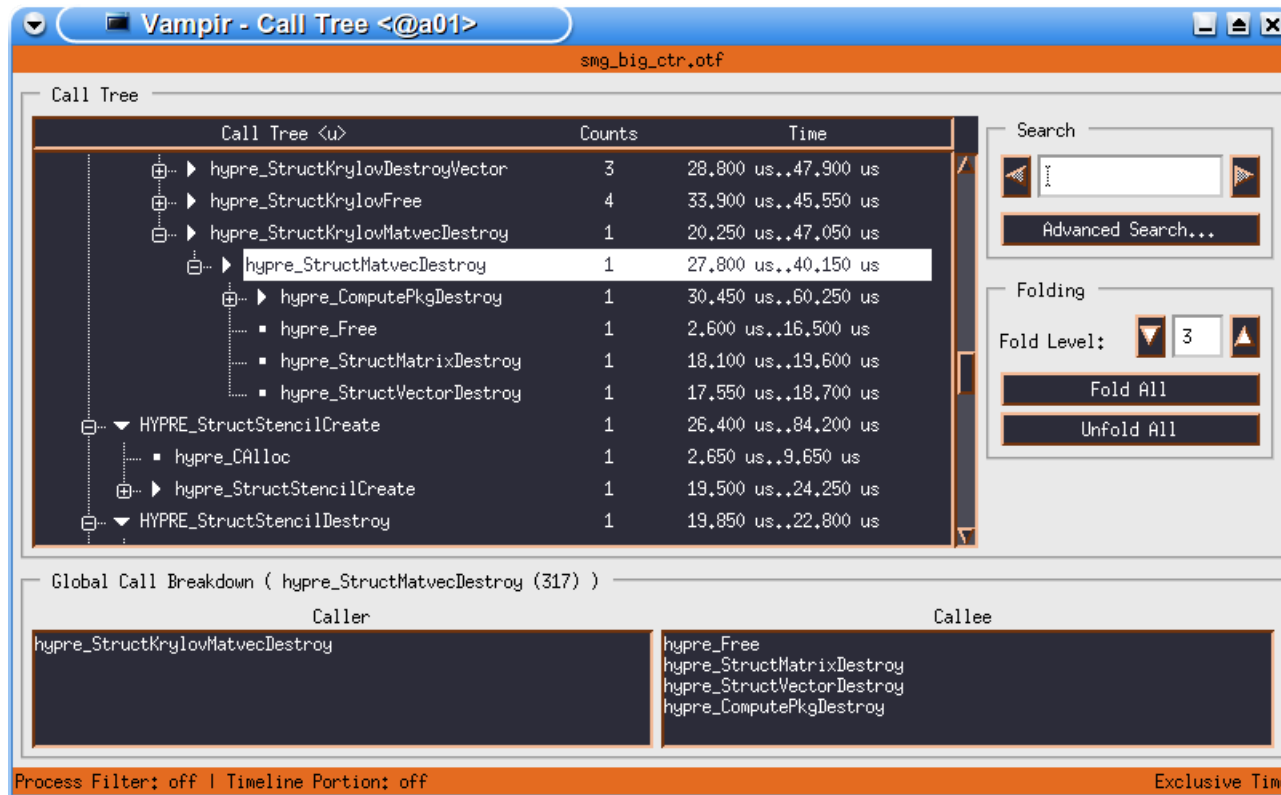


Collective Operation Statistics



Counter Timeline Display





Program Instrumentation

- detect run-time events (points of interest)
- pass information to run-time measurement library

Profile Recording

- collect aggregated information (Time, Counts, ...)
- about program and system entities
 - functions, loops, basic blocks
 - application, processes, threads, ...

Trace Recording

- save individual event records together with precise timestamp and process or thread ID
- plus event specific information

- What do you need to do for it?
 - Use VampirTrace
- Instrumentation (automatic with compiler wrappers)

```
CC=icc  
CXX=icpc  
F90=ifc  
MPICC=mpicc
```

```
CC=vtcc  
CXX=vtcxx  
F90=vtf90  
MPICC=vtcc
```

- Re-compile & re-link
- Trace Run (run with appropriate test data set)
- more details later

What does VampirTrace do in the background?

- Instrumentation:
 - via compiler wrappers
 - by underlying compiler with specific options
 - MPI instrumentation with replacement lib
 - OpenMP instrumentation with Opari
 - also binary instrumentation with Dyninst
 - partial manual instrumentation

What does VampirTrace do in the background?

- Trace Run:
 - event data collection
 - precise time measurement
 - parallel timer synchronization
 - collecting parallel process/thread traces
 - collecting performance counters (from PAPI, memory usage, POSIX I/O calls and fork/system/exec calls, and more ...)
 - filtering and grouping of function calls

- **Vampir & VampirServer**
 - interactive trace visualization and analysis
 - intuitive browsing and zooming
 - scalable to large trace data sizes (100GByte)
 - scalable to high parallelism (2000 processes)
- Vampir for Windows in progress, beta available
- **VampirTrace**
 - convenient instrumentation and measurement
 - hides away complicated details
 - provides many options and switches for experts
- VampirTrace is part of Open MPI 1.3



Forschungszentrum Jülich

- Jülich Supercomputing Centre



RWTH Aachen University

- Center for Computing and Communication



Technical University of Dresden

- Center for Information Services and High Performance Computing



University of Tennessee

- Innovative Computing Laboratory



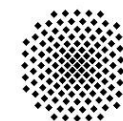
Technical University of München

- Chair for Computer Architecture

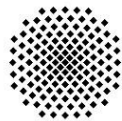


University of Stuttgart

- High Performance Computing Centre



Universität Stuttgart



Universität Stuttgart