TECHNISCHE UNIVERSITÄT DRESDEN

Zentrum für Informationsdienste und Hochleistungsrechnen

HRSK Overview
Phobos

Zellescher Weg 12

Tel. +49 351 - 463 - 35450

Dresden, 4 Juli 2006

Wolfgang E. Nagel, Matthias S. Müller

ZIH
Center for Information Services & High Performance Computing

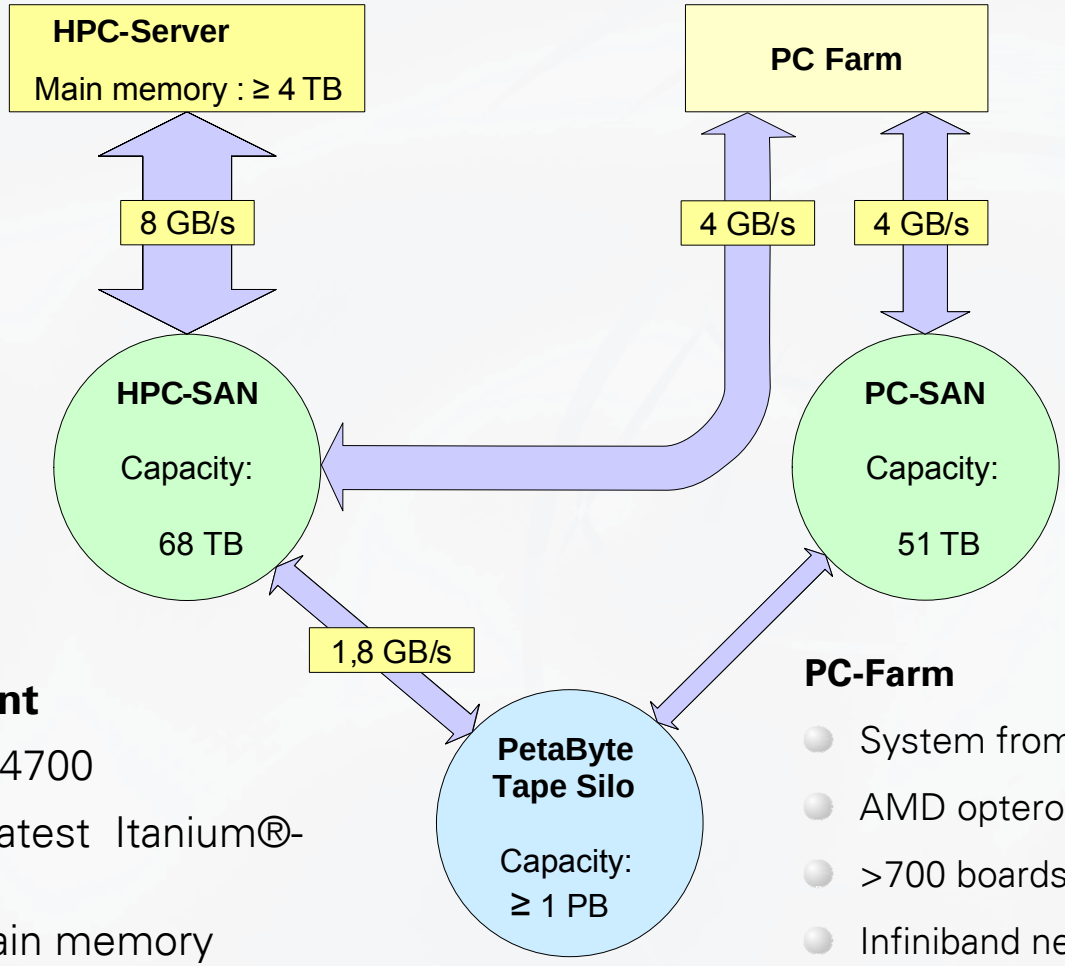
Center for Information Services and HPC (ZIH)

- Central Scientific Unit at TU Dresden
- Merged institution: TUD Computing Center (URZ) and Center for High Performance Computing (ZHR)
- Competence Center for „Parallel Computing and Software Tools“
- Strong commitment to support *real users*
- Development of algorithms and methods: Cooperation with users from all departments

Responsibilities of ZIH

- Providing infrastructure and qualified service for TU Dresden and Saxony
- Research topics
  - Architecture and performance analysis of High Performance Computers
  - Programming methods and techniques for HPC systems
  - Software tools to support programming and optimization
  - Modeling algorithms of biological processes
  - Mathematical models, algorithms, and efficient implementations
- Role of mediator between vendors, developers, and users
- Pick up and preparation of new concepts, methods, and techniques
- Teaching and Education

Procurement: Overall Infrastructure / Future Directions



HPC-Component

- SGI® Altix® 4700
- 2048 of the latest Itanium®-Cores
- 6.5 TByte main memory

PC-Farm

- System from Linux Networx
- AMD opteron CPUs
- >700 boards with >2500 cores
- Infiniband networks between the nodes

Timeline

	2005						2006								
	Jul	Aug	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep
Machine Room Upgrade															
Installation Stage 1a (Test operation)															
Installation Stage 1b															
Installation Stage 2															

Contract: SGI

HPC-Component

- Next generation Shared-Memory-Plattform SGI® Altix® (Codename ´Tollhouse´)
- more than 1500 of the latest Intel®-Itanium®-Cores
- 6 TByte main memory

- "Capability Computing" system
- Focus: „In-Memory"-Computing
- New research instrument to extract knowledge from data
- "Accelerator of the Theory"

Contract: SGI

PC-Farm

- System from Linux Networx
- CPUs: Always latest most powerful chips from AMD
- Overall more than 700 boards with more than 2000 cores
- Infiniband networks between the nodes

- “Capacity Computing“ system
- Focus: Throughput with excellent performance
- Heterogeneous diversity is part of the design

TECHNISCHE UNIVERSITÄT DRESDEN

Center for Information Services and High Performance Computing (ZIH)

Stage1a

Zellescher Weg 12

Tel. +49 351 - 463 - 35450

Wolfgang E. Nagel, Matthias S. Müller

ZIH
Center for Information Services & High Performance Computing

HRSK Stage 1a – HPC-Server

- SGI Altix 3700 Bx2
- 192x 1.5GHz/4MB L3 Cache Itanium2 CPUs
- 1152 GFlops/s Peak Performance
- 768 GB Shared Memory (4 GB/CPU), NUMA
- 1 TB lokal discs + 34 TB SAN
- SuSE SLES 9 inkl. SGI ProPack 4
- Intel Compiler and Tools:
  - C/C++ , Fortran Compiler
  - MKL
  - Vtune, Trace Collector, Trace Analyzer
- Alinea DDT Debugger
- Batchsystem LSF

merkur.hrsk.tu-dresden.de

HPC-SAN Stage 1a

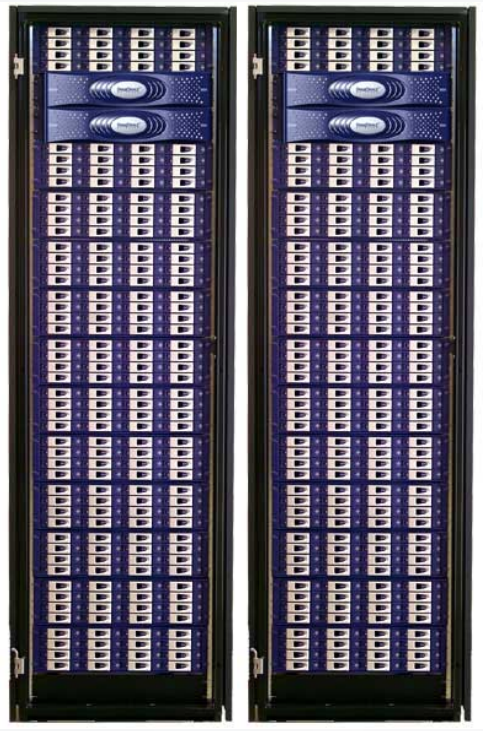
- 1x DDN RAID System S2A9500:
  - 2x 38/42U Racks (air cooled system)
  - 2x S2A9500 Couplet (5GB Cache, 8x FC4 Host Ports
  - 16x Optical SFP Interface Module
  - 20x Blue 16-Slot Dual-Loop FC 2Gb JBOD,
  - 20x Rack Mount Kits
  - 292x 146GB 10k RPM FC Disks (4 hot spare)
  - 34 TB net capacity
  - DIRECT-GUI

PC-Farm Stage 1a

phobos.hrsk.tu-dresden.de

- 64 dual CPU nodes
- 128 AMD Opteron DP248 2.2 GHz (Single-Core) CPUs
- 563,2 GFlops/s peak performance
- 256 GB main memory (4GB per node)
- SUSE operating system
- Infiniband 4x Interconnect
- 80 GB local disc per node
- 21,2 TB shared disc space:
  - 2x DDN RAID System S2A9500

PC-SAN Stage 1a



- DDN RAID System S2A9500:
  - 2x 38/42U Racks (air cooled),
  - 1x S2A9500 Couplet (5GB Cache, 6x FC4 Host Ports),
  - 8x Optical SFP Interface Module,
  - 19x Blue 16-Slot Dual-Loop FC 2Gb JBOD,
  - 19x Rack Mount Kits,
  - 184x 146GB 10k RPM FC discs (4 Hot-Spare),
- 21,2 TB netto capacity

TECHNISCHE UNIVERSITÄT DRESDEN

Center for Information Services and High Performance Computing (ZIH)

Stage1b,2

Zellescher Weg 12

Tel. +49 351 - 463 - 35450

Wolfgang E. Nagel, Matthias S. Müller

ZIH
Center for Information Services & High Performance Computing

Comparison of systems in 2005

	HPC Server		PC Farm	
Peak Performance	1152	+	563	+
#CPUs	192		128	
Nodes	1	+	64	
Total Memory	768 GB		256 GB	
Total Disc	34 TB		21 TB	
MPI		+		+
OpenMP		++		
Usable Shared Memory	768 GB	++	4 GB	-
MPI Bandwidth	~ 3.2 GB/s	++	~ 1GB/s	+
MPI Latency	~ 1.5-3 microsecond	++	~ 4-5 microseconds	+

PC-Farm Stage 1b

- Compute Partition:
  - 128 single CPU, Opteron dual core
  - 104 dual CPU, Opteron dual core
  - 24 quad CPU, Opteron dual core
- AMD Opteron dual core 2.6 GHz
- 3801,6 GFlops/s peak performance
- 864GB main memory (2GB per CPU chip)
- SUSE operating system
- Infiniband 4x Interconnect
- 21,2 TB shared disc space:
  - DDN RAID System S2A9500

PC-Farm Stage 2

- Compute Partition:
  - 256 single CPU, Opteron dual core
  - 128 dual CPU, Opteron dual core
  - 88 quad CPU, Opteron dual core
- AMD Opteron dual core 2.6 GHz
- 8294,4 GFlops/s peak performance
- 3456 GB main memory (2GB per core)
- SUSE operating system
- Infiniband 4x Interconnect
- 51 TB total shared disc space:
  - DDN RAID System S2A9500

PC-Farm final config

- Compute Partition:
  - 128+256 single CPU, Opteron dual core
  - 104+128 dual CPU, Opteron dual core
  - 24+88 quad CPU, Opteron dual core
- AMD Opteron dual core 2,6 GHz
- 13478 GFlops/s peak performance
- 4320GB main memory
  - 1-2GB per core
  - 2-32GB per node
- SUSE operating system
- Infiniband 4x Interconnect
- 51 TB total shared disc space:
  - DDN RAID System S2A9500

HPC-Server Stage 2

- SGI Tollhouse
- 1024 MonteCito CPUs (2048 Cores)
- 12288 GFlops/s peak performance
- 6 TB Shared Memory, NUMA
- 68 TB Disk
- SUSE SLES 9
- Intel Compiler and Tools:
  - C/C++ , Fortran Compiler
  - MKL
  - Vtune, Trace Collector, Trace Analyzer
- Alinea DDT debugger
- LSF batch system

Comparison of systems in 2006

	HPC Server		PC Farm	
Peak Performance	12288	+	13478	+
#Cores	2048		2592	
Nodes	4	+	728	
Total Memory	6 TB		4 TB	
Total Disc	68 TB		50 TB	
MPI		+		+
OpenMP		++		
Usable Shared Memory	2048 GB	++	<=32 GB	-
MPI Bandwidth	~ 1-2 GB/s	++	~ 1GB/s	+
MPI Latency	~ 1.5-3 microsecond	++	~ 4-5 microseconds	+

Main Contractor: SGI

HPC-component

- 2048 cores (Intel Itanium 2, Montecito)
- 6 TB Main memory with high bandwidth (major decision criteria)

SAN for HPC / PC

- 68 / 51 TB Capacity

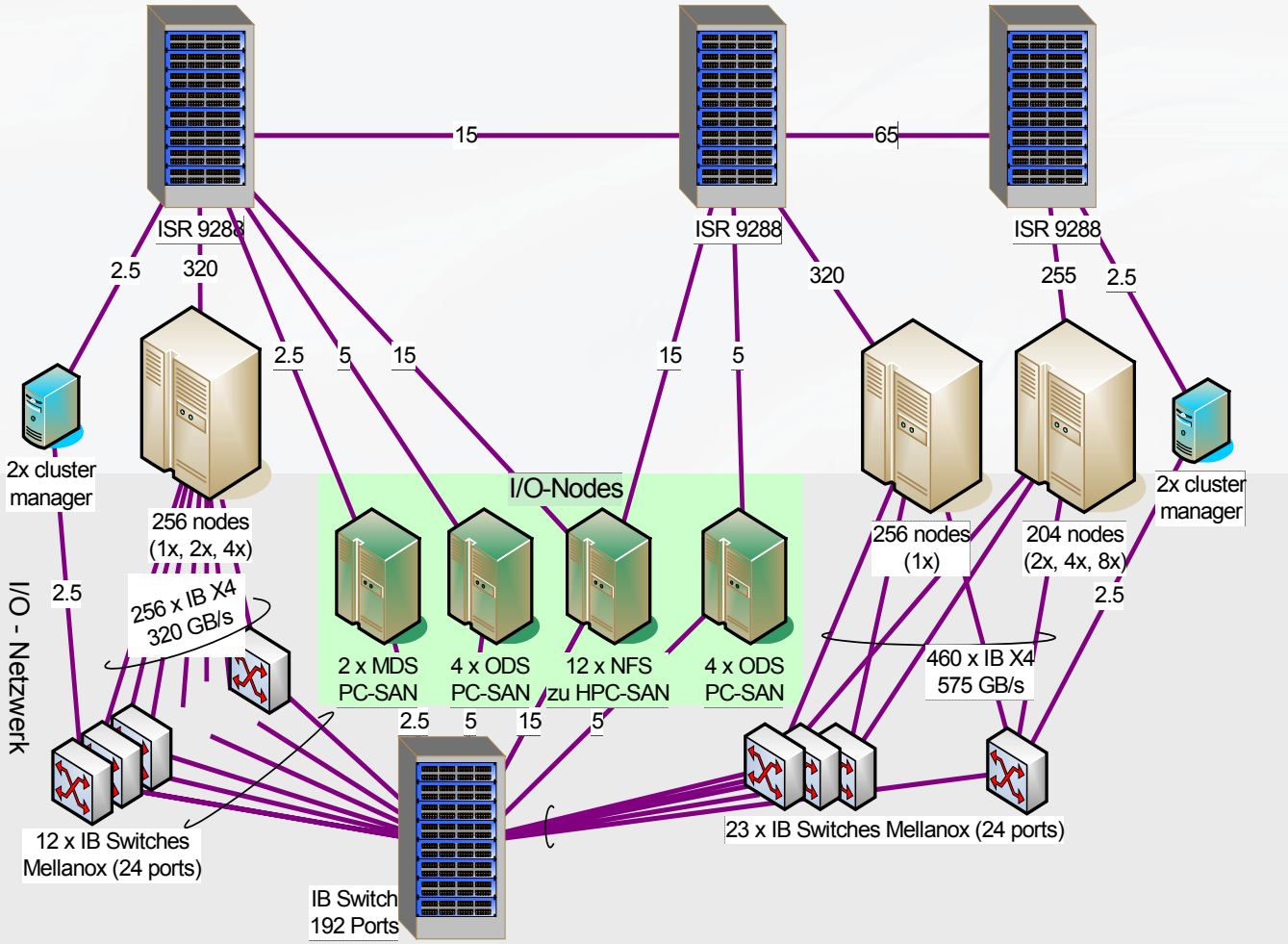
Tape Silo

- 1 PB Capacity

PC-Farm

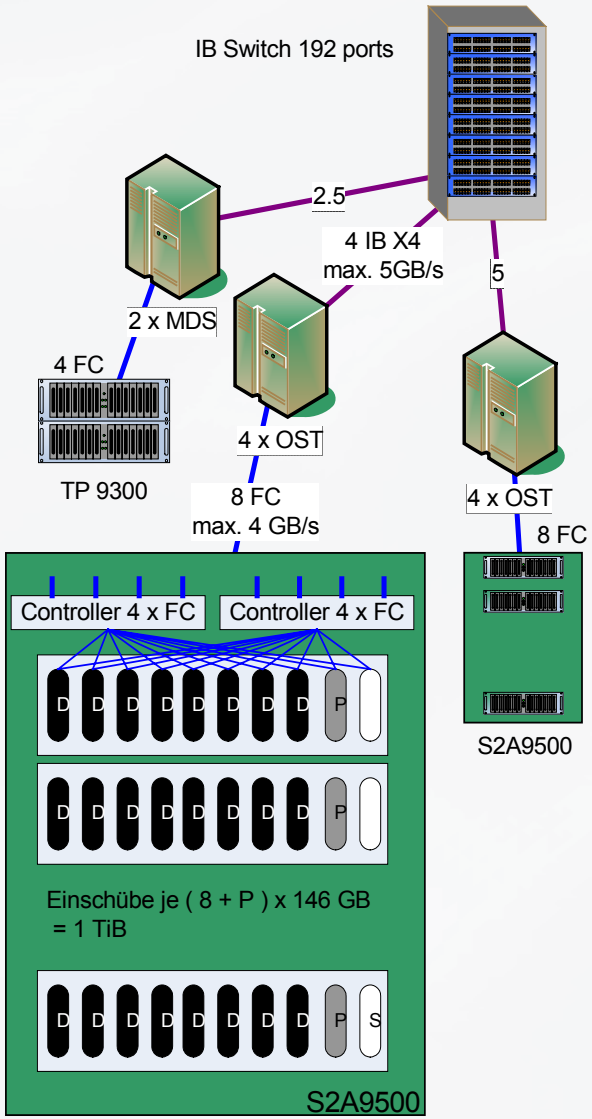
- > 700 System boards (AMD Opteron)
- Infiniband network build from 288-port switches

PC-Farm - Details



- Interconnect: IB X4
- Compute-Knoten connected via three switches (288 ports)
- connection between switches
- 2 x 2 Cluster Manager
- Connection to HPC-SAN with 12 NFS-Server (CXFS-Clients)
- cascaded I/O-net:
  - 35 IB X 4 Switches (24 ports)
  - each with 4 x IB-Uplink to…
  - IB-Switch (192 ports)

PC-San - Details



- Lustre FS
- IB Switch (192 ports)
- 2 x DDN S2A 9500
- 4 Controller with 4 x FC 4Gb/s each
- capacity: 50,9 TB
- 4 Racks
- 440 discs with 146 GB each

- 2 x MDS je 2 dual-core Opteron 275, 4 GB RAM, dual-port FC
- 4 x ODS (OSS,OST) je 2 dual-core Opteron 275, 4 GB RAM, 2 x dual-port FC
- 4 x ODS (OSS,OST) je 2 dual-core Opteron 280, 4 GB RAM, 2 x dual-port FC (Stufe 2)
- TP 9300 (MDS Storage Subsystem) 16 x 146 GB für Metadaten

TECHNISCHE UNIVERSITÄT DRESDEN

User Support at ZIH

A few examples

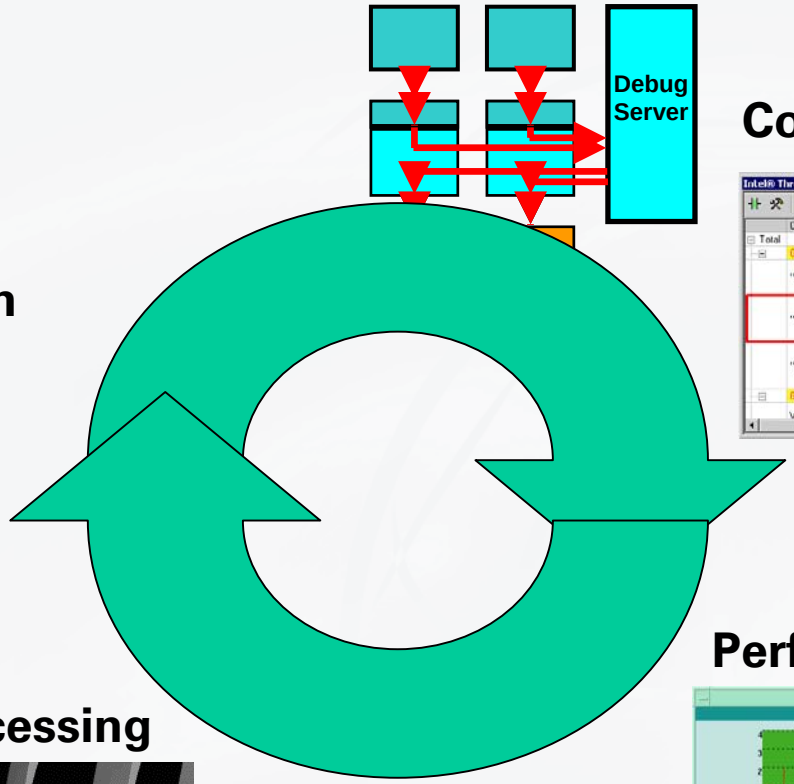
Zellescher Weg 12

Tel. +49 351 - 463 - 35450

Wolfgang E. Nagel, Matthias S. Müller

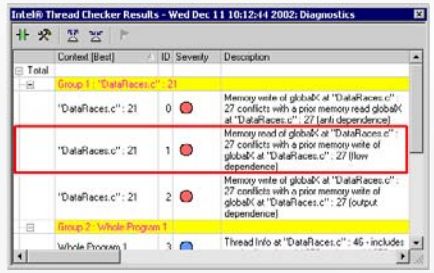
ZIH
Center for Information Services & High Performance Computing

Evolution of a parallel application

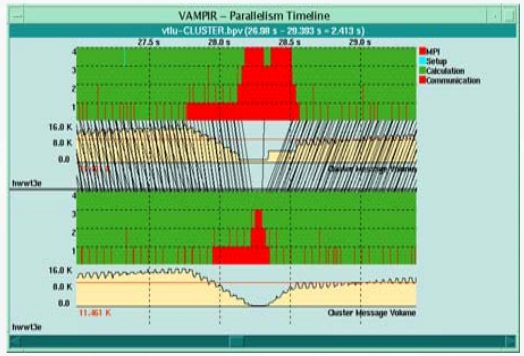


Code Modification

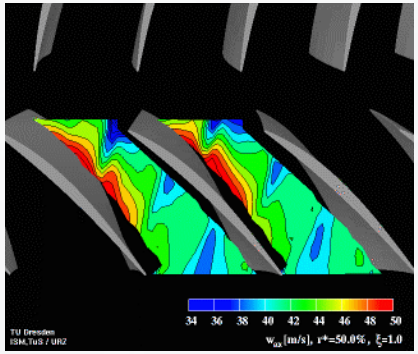
Correctness



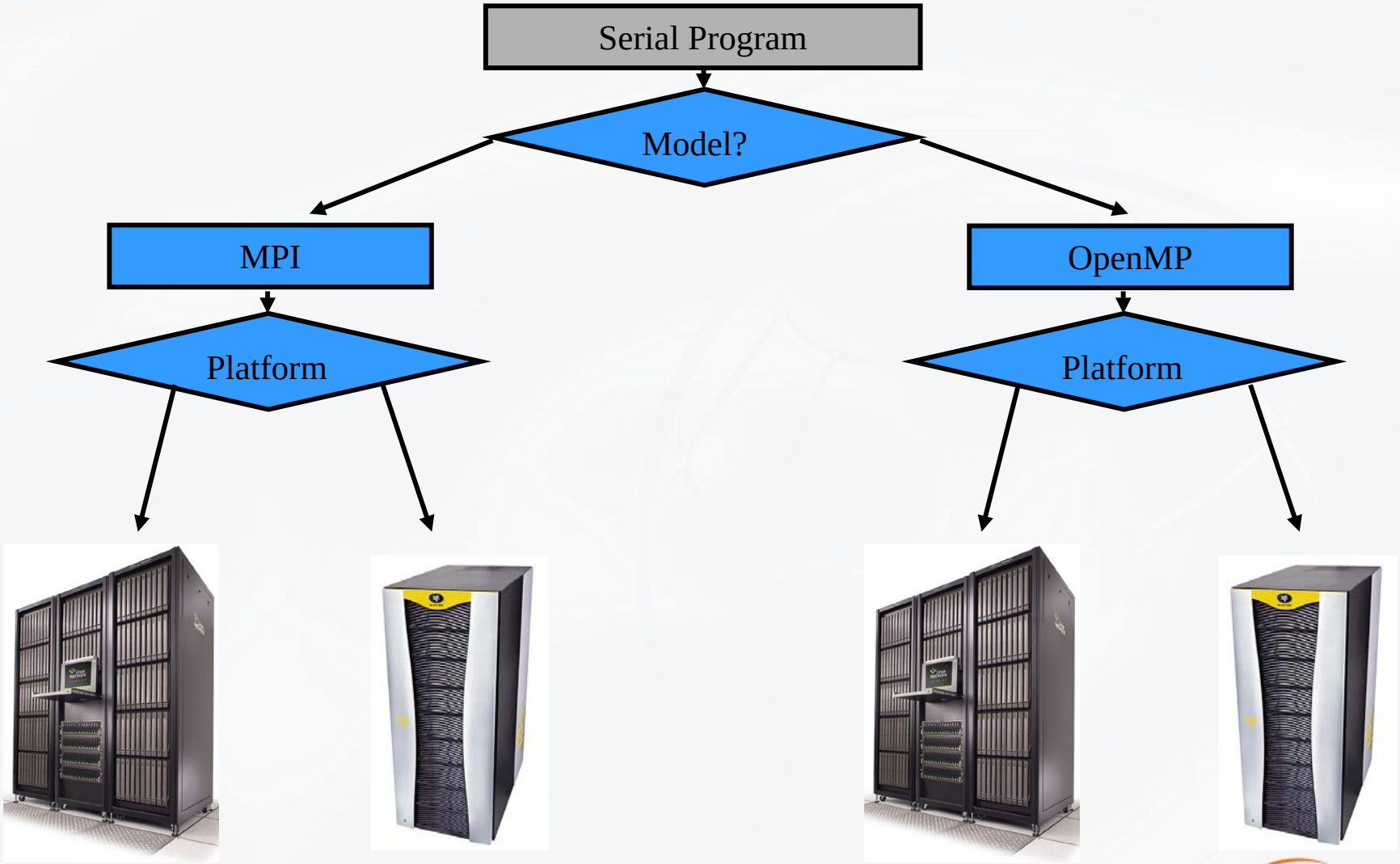
Performance



Post processing



HPC Consulting



MPI code checking with MARMOT

Tool for the development of MPI applications

Automatic runtime analysis of the application:

- Detect incorrect use of MPI
- Detect non-portable constructs
- Detect race conditions and deadlocks

MARMOT does not require source code modifications, just relinking

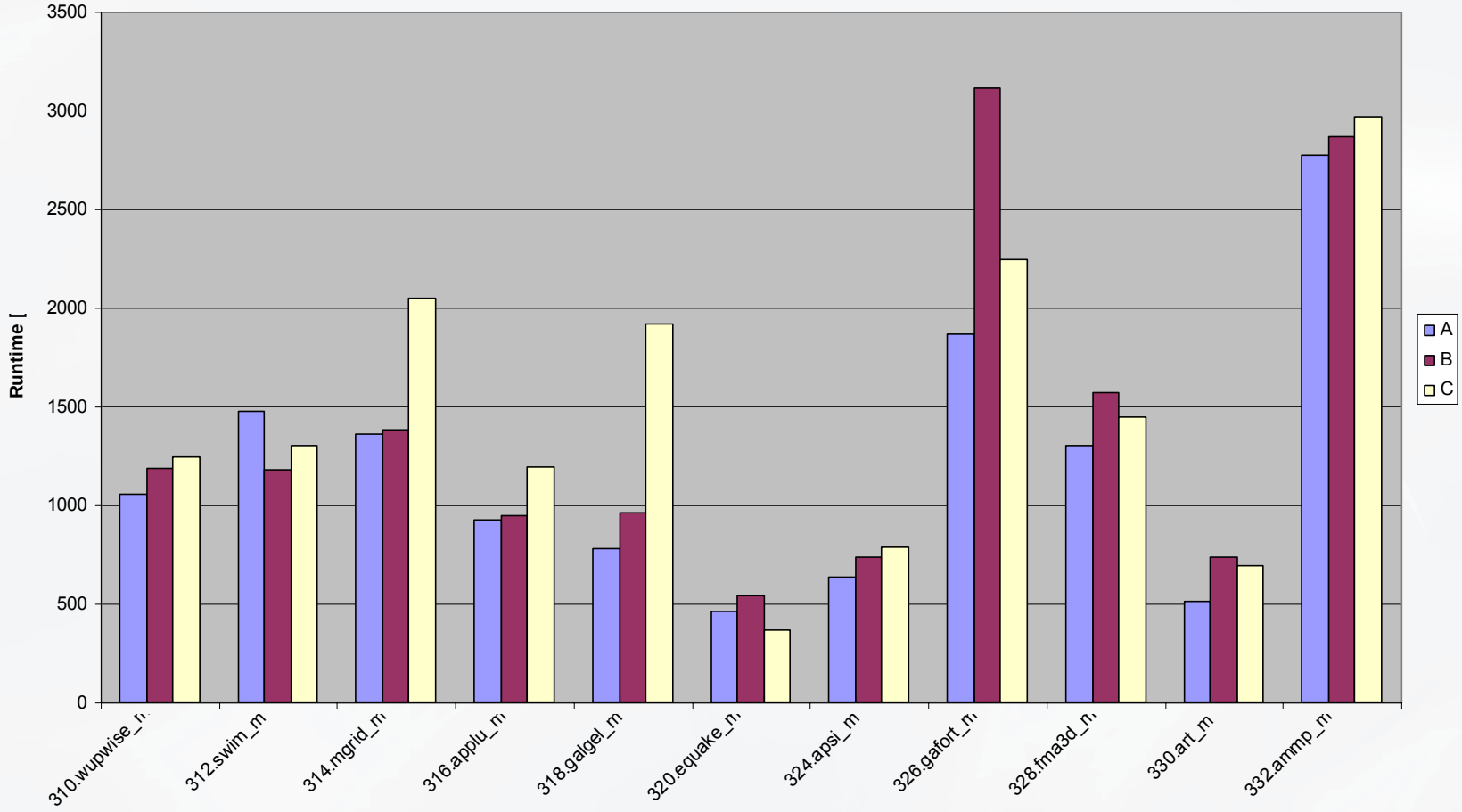
C and Fortran binding of MPI -1.2 is supported, also C++ and mixed C/Fortran code

Development is still ongoing (not every possible functionality is implemented yet…)

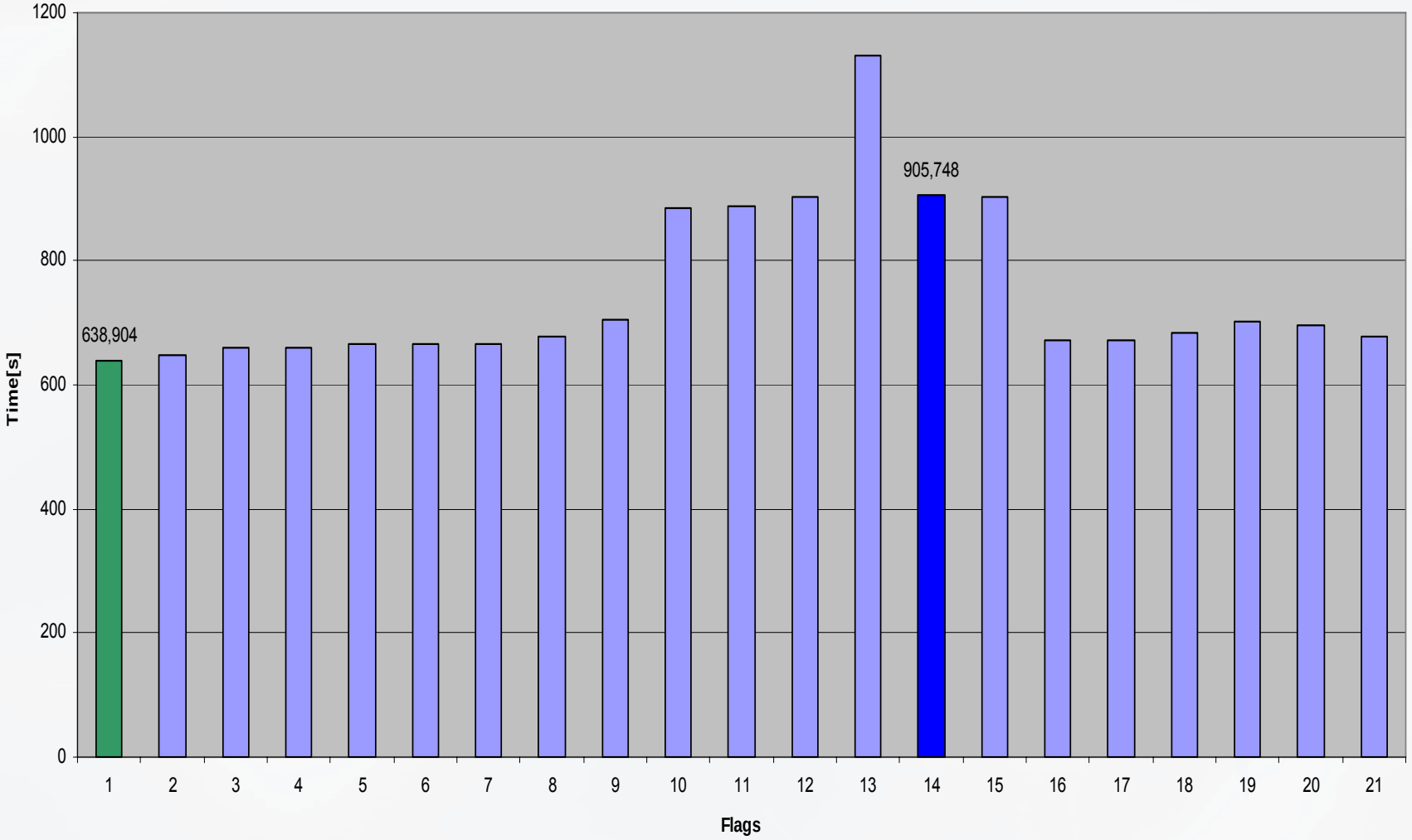
Tool makes use of the so-called *profiling interface*

MARMOT will continue to be developed jointly by ZIH and HLRS

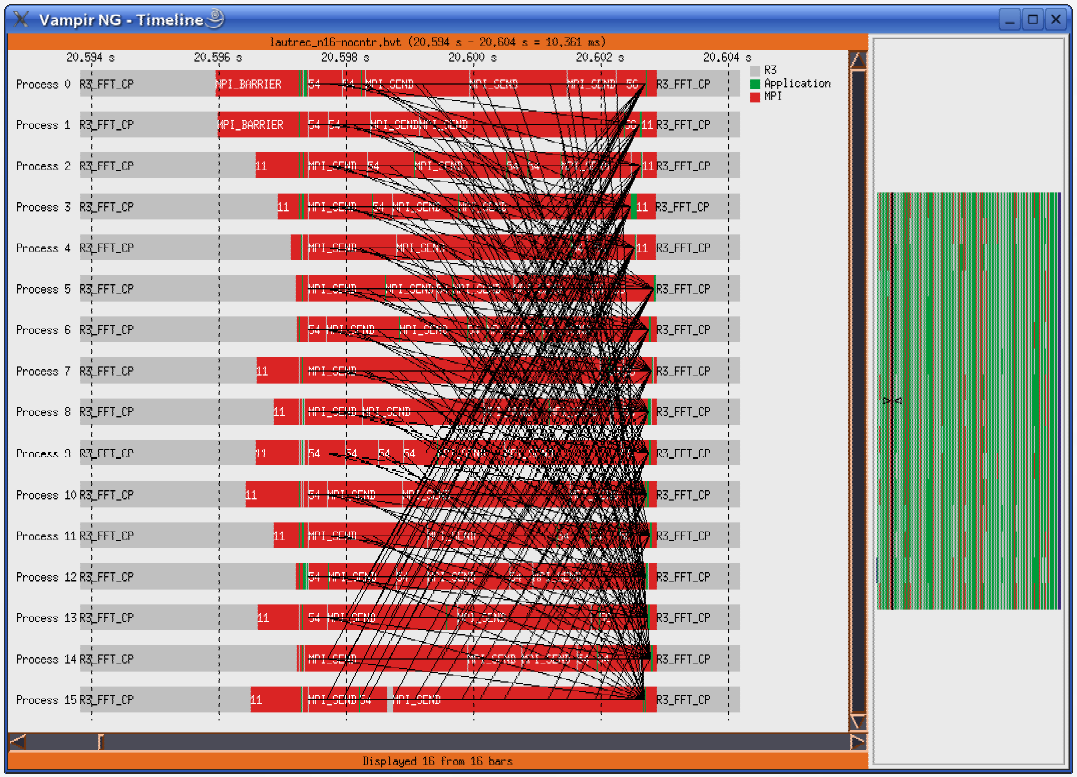
Compare different Compilers with SPEC OMPM2001

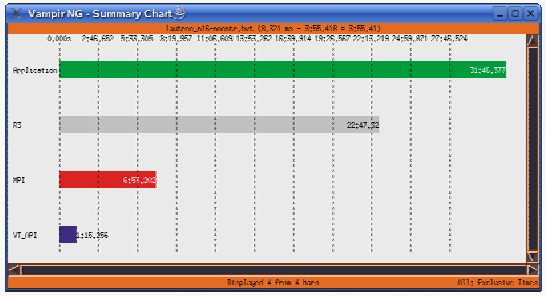


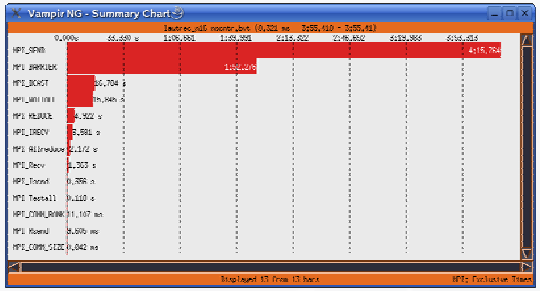
Code Tuning: different compiler flags

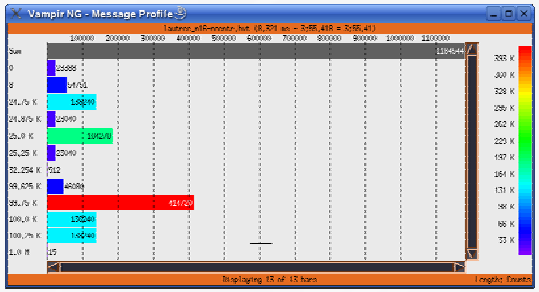


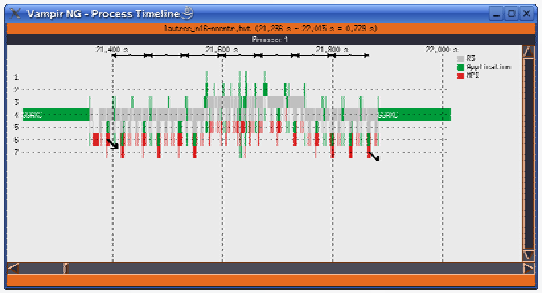
Vampir Analysis of Lautrec











TECHNISCHE UNIVERSITÄT DRESDEN

ZIH
Center for Information Services & High Performance Computing

TECHNISCHE UNIVERSITÄT DRESDEN

A few success stories

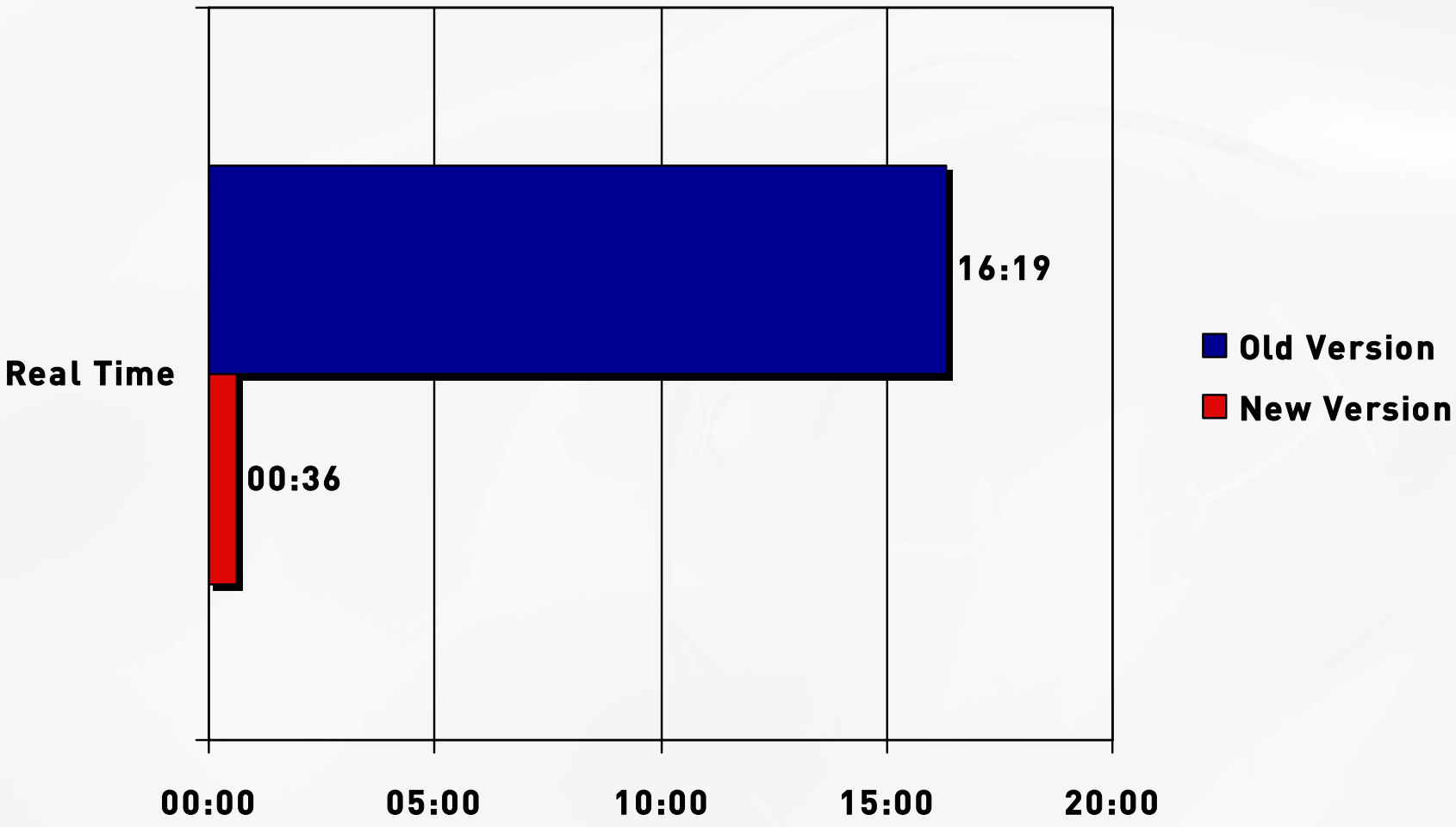
Zellescher Weg 12

Tel. +49 351 - 463 - 35450

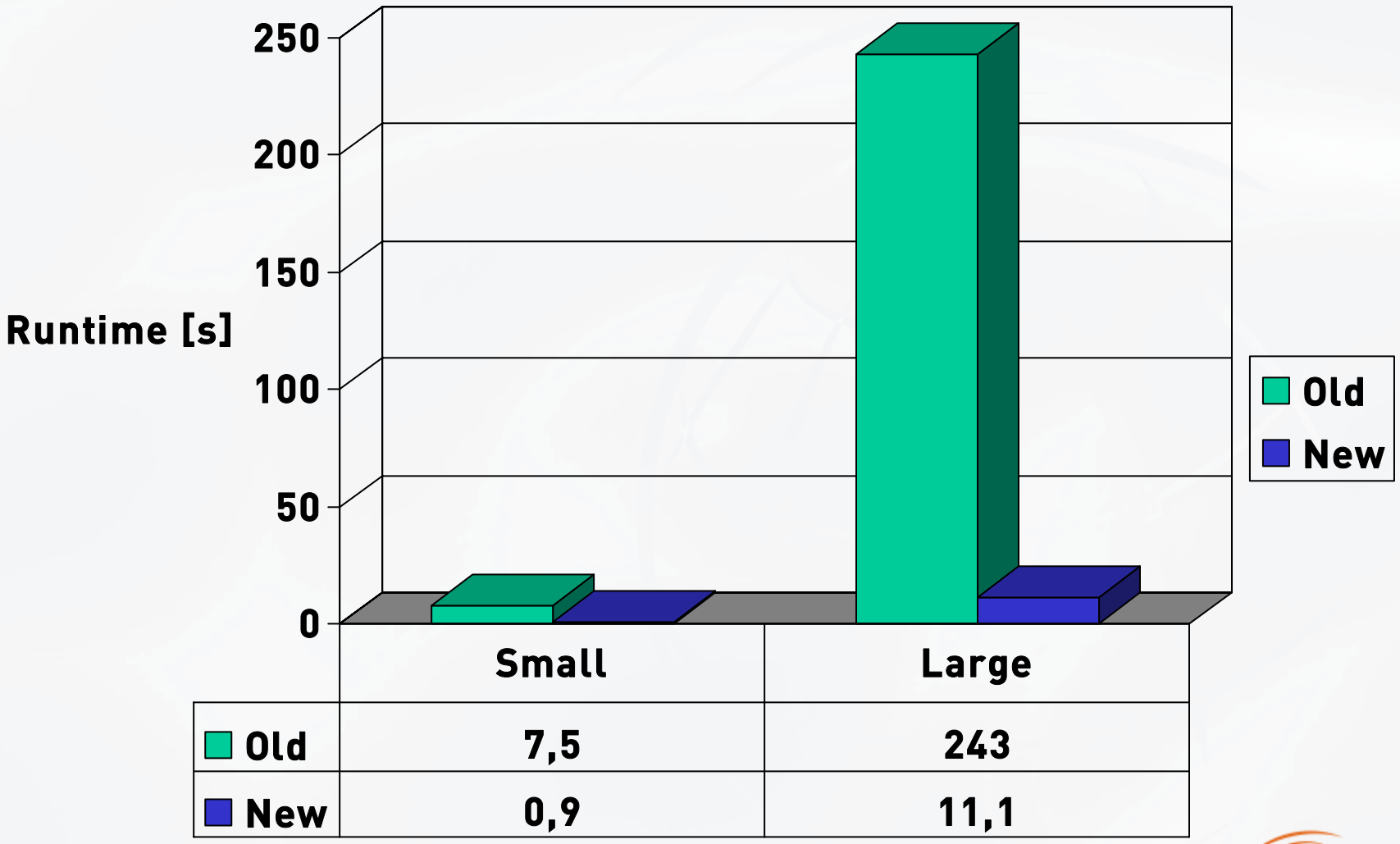
Wolfgang E. Nagel, Matthias S. Müller

ZIH
Center for Information Services & High Performance Computing

Example: Serial Optimization

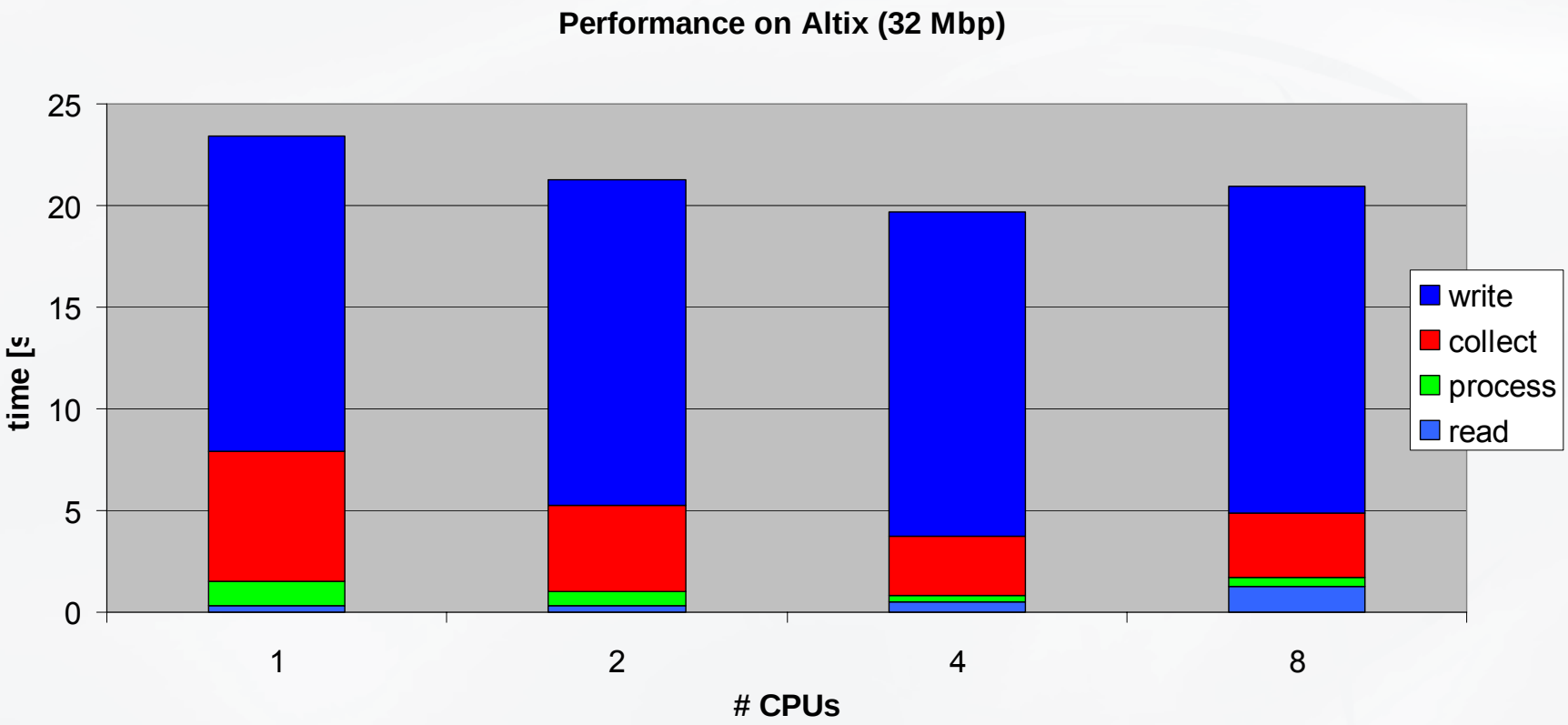


Example: Matlab Calculation

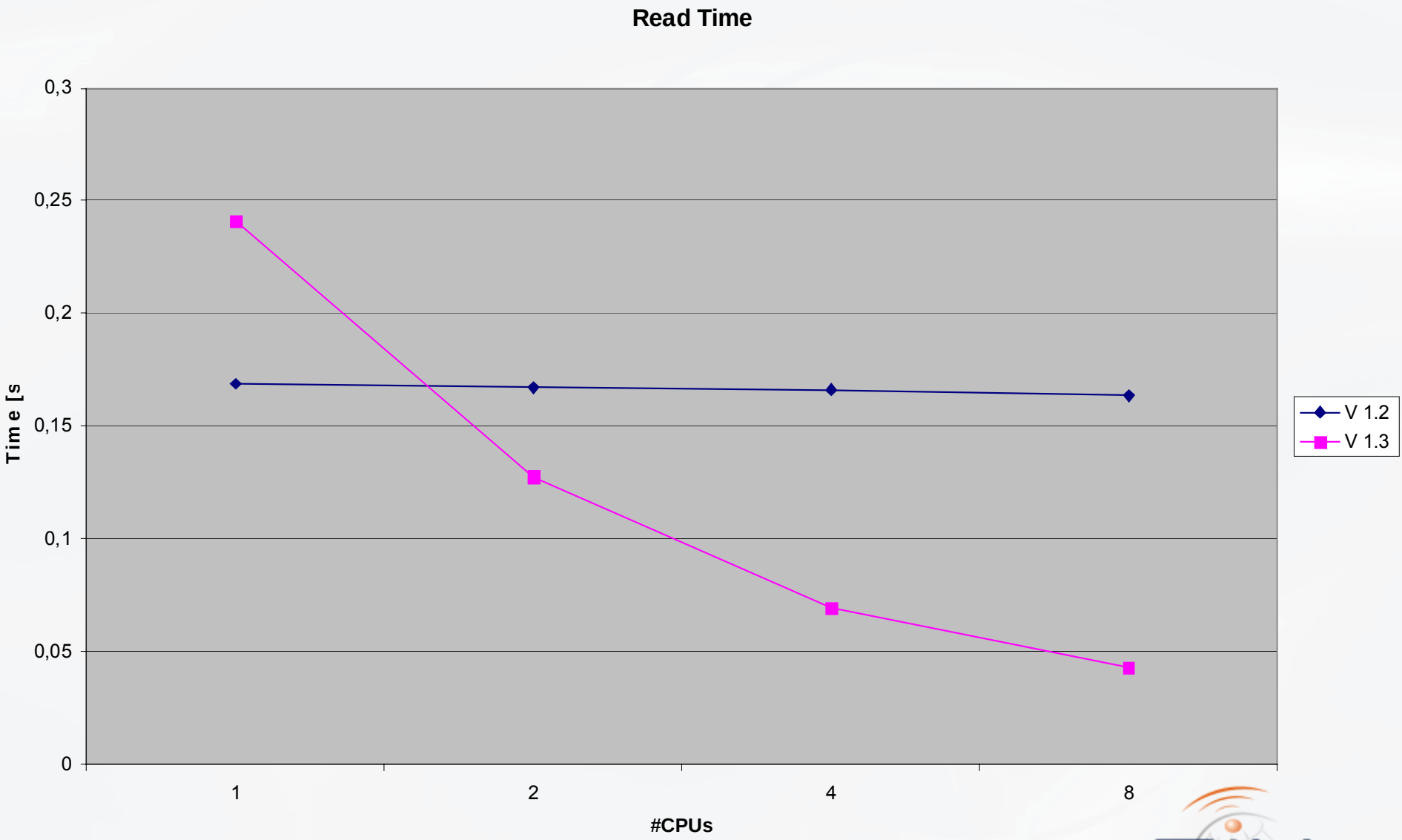


	Small	Large
Old	7,5	243
New	0,9	11,1

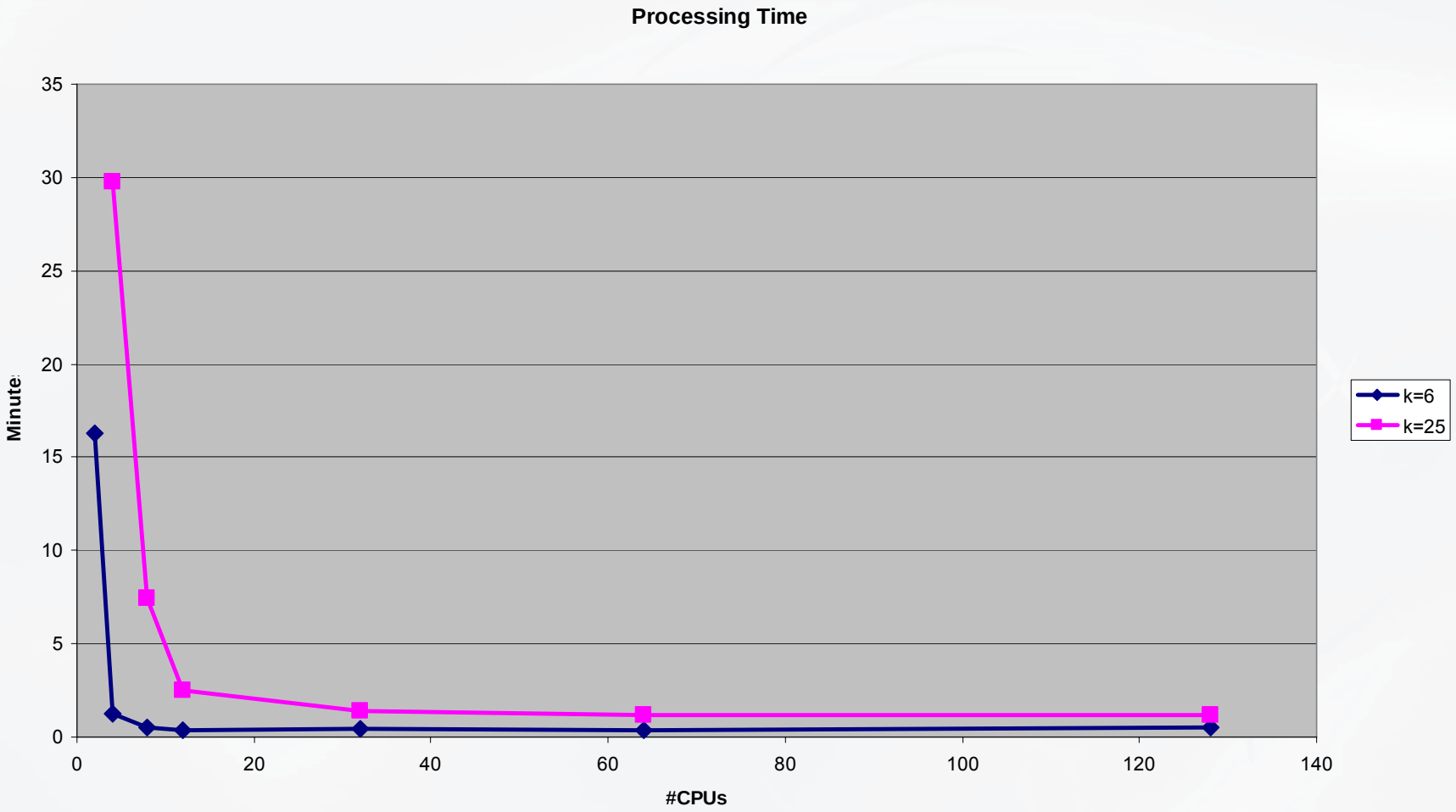
Overall performance of Geneindex 1.2



Read Performance on GPFS (davidd-o) with 32 Mbps



Analysis of Fruit Fly Genome (165 Mbps)



Recent Improvements

- Printed Documentation:
  Every new project receives one copy. Also available for download
- Mailing list for information of users:
  zih-hpcnews
- Mailing list for discussions between users:
  zih-hpcforum
- Trouble Ticket System for support (currently in beta):
  HPCsupport@zih.tu-dresden.de

Summary and Conclusion

- The new installation offers a large opportunity
- It is still a long way to go
- We hope we can be of some help
- Enjoy the user training and make the best out of it

TECHNISCHE UNIVERSITÄT DRESDEN

Zentrum für Informationsdienste und Hochleistungsrechnen

Tools on Phobos

Zellescher Weg 12

Tel. +49 351 - 463 - 33640

Dresden, June 21 2006

Ulf Markwardt

ZIH
Center for Information Services & High Performance Computing

Overview

Compilers

- Installed compilers
- Selected compiler options
- Comparison between compilers

Libraries

- Basic libraries
- Advanced mathematical libraries
- Comparison of FFT libraries (Daniel Hackenberg)

Debuggers

Profiling tools

Misc

- LSF
- Applications

Compilers

Installed Compilers

- GCC, g77 version 3.3
  on the way to get mature – but not yet competitor in performance

Commercial Compilers – Fortran 95/90/77, C 90, C++

- PathScale Compiler version 2.4
- PGI Compiler (Portland Group) version 6.0
- Intel Compiler version 9.0

  Intel is trying to compete
  currently: EM64T–code not optimized for Opteron CPU

Compilers

Selected Compiler Options

Intel	PGI	Pathscale	Description
`-O0 … -O3`	`-O0 … -O3`	`-O0 … -O3`	Optimization level. (`-O2` is a good start)
`-ipo`	`-Mipa`	`-ipa`	inter procedure optimization (across files)
`-openmp`	`-mp`	`-mp`	OpenMP support
`-parallel`	`-Mconcur`	`-apo`	Auto-parallelizer (SMP)
`-mp`	`-Kieee`	`-fno-unsafe-math-optimizations`	use this flag to limit floating-point optimizations and maintain declared precision
`-mp1`	`-Knoieee`	`-ffast-math`	some floating-point optimizations are allowed, less performance impact than IEEE conformity

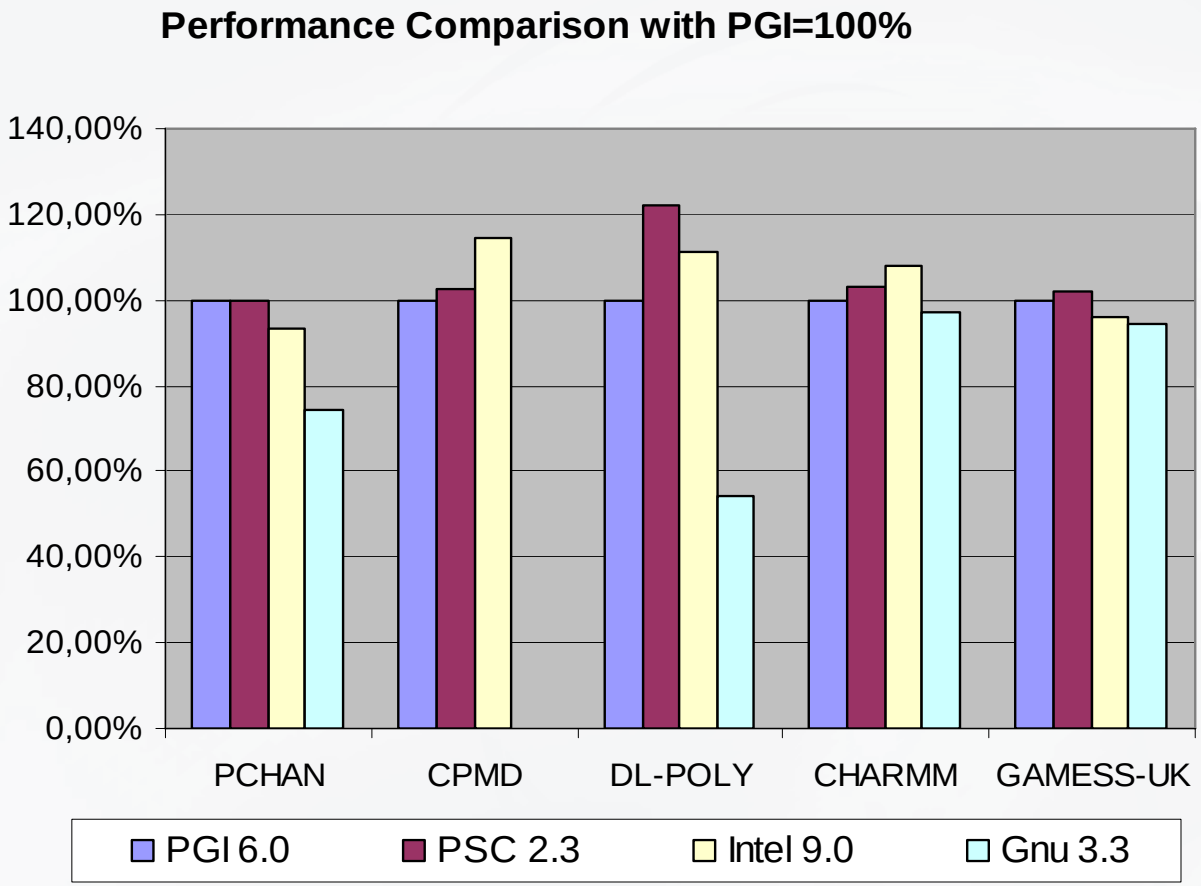
Compilers

Selected Compiler Options (continued)

Intel	PGI	Pathscale	Description
`-xW` (em64t-sup)	`-tp amd64`	`-mcpu=opteron`	optimize for Opteron processor
	`-fastsse`	`-msse2`	Use SSE instructions (increased performance for sincle precision)
`-prof-gen`	`-Mpfi`		Generate profiling information during the run
`-prof-use`	`-Mpfo`		Use profiling information for compiling

Compilers

Comparisons



Data from „Performance comparison of compilers for AMD Opteron. Kozin, I.N., Deegan, M.J. et al. Computational Science an Engineering Dep. CCLRC Daresbury Laboratory, Warrington UK. 12/17/05

Libraries

Basic Libraries

- MPICH over Infiniband: MVAPICH version 1.2.6
  built for Intel, PGI, PathScale, Gnu – correct version for loaded module will be chosen:
  - module load pgi
  - mpicc, mpicxx, mpif77, mpif90
- OpenMPI
  http://www.open-mpi.org/
  built for Intel and Gnu
- Performance Application Programming Interface – PAPI 3.2.1
  http://icl.cs.utk.edu/papi/
  usage of performance counters
- Tracing Libraries
  vampirtrace, …

Libraries

Mathematical Libraries

- Automatically Tuned Linear Algebra Software - ATLAS 3.6.0
  http://www.netlib.org/atlas/
  BLAS, LAPACK
- Intel Math Kernel Library - MKL 8.0
  http://www.intel.com/cd/software/products/asmo-na/eng/perflib/mkl/index.htm
  BLAS, LAPACK, FFT, scalar and vector functions, sparse and dense matrices
- AMD Core Math Library – ACML 2.7.0
  http://developer.amd.com/acml.aspx
  BLAS, LAPACK, FFT, scalar and vector functions , sparse and dense matrices
- Gnu Scientific Library - GSL 1.8
  http://www.gnu.org/software/gsl
  functions above …. + differential eq., numeric differentiation, random distributions, Eigensystems, interpolation, etc.
- Numeric Algorithms Group – NAG Library Mark 8
  http://nag.co.uk/numeric
  functions optimization, integration, random distributions
- Fastest Fourier Tranfor in the West FFTW 3.1
  http://www.fftw.org
  FFT – might become obsolete with fftw2mkl – wrappers MKL 8.1

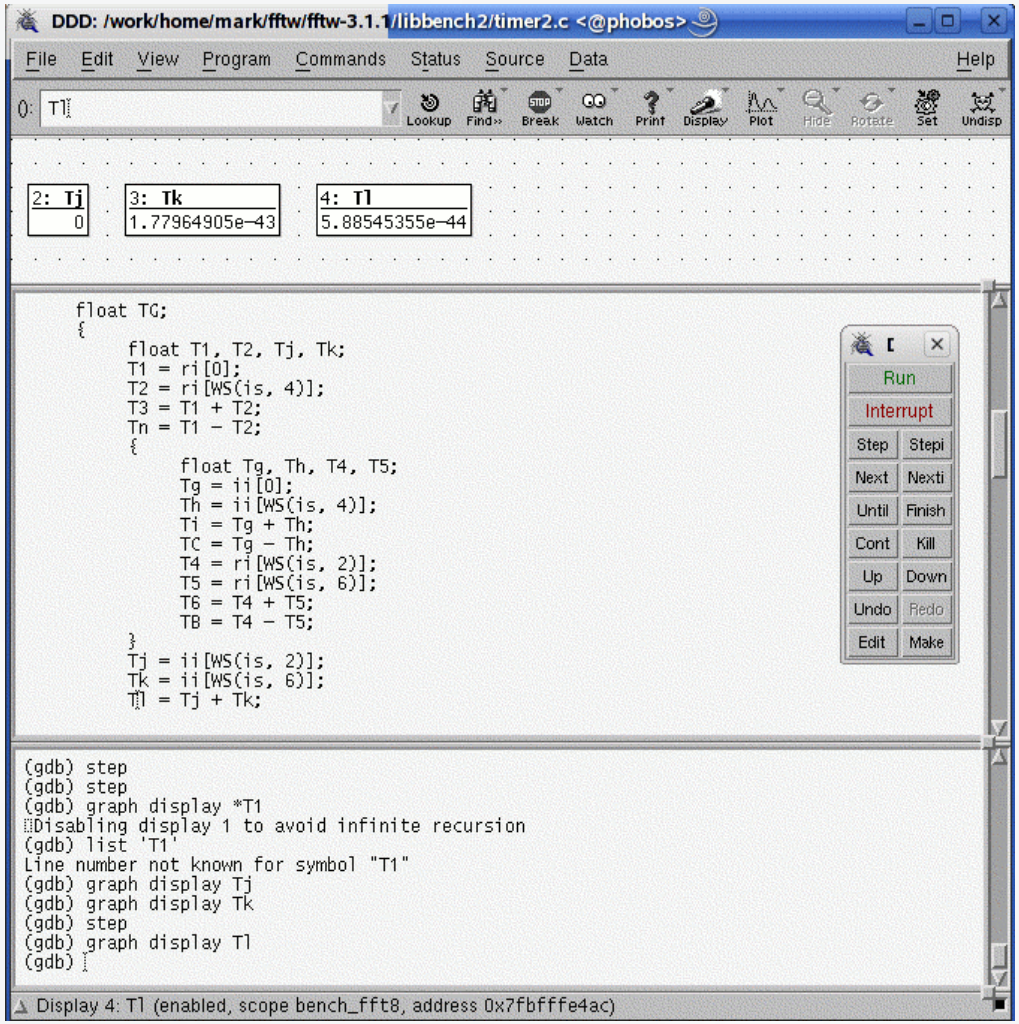
Libraries

Comparison of Libraries

- Performance depends on problem size
- Users are strongly recommended to test the best

Debuggers

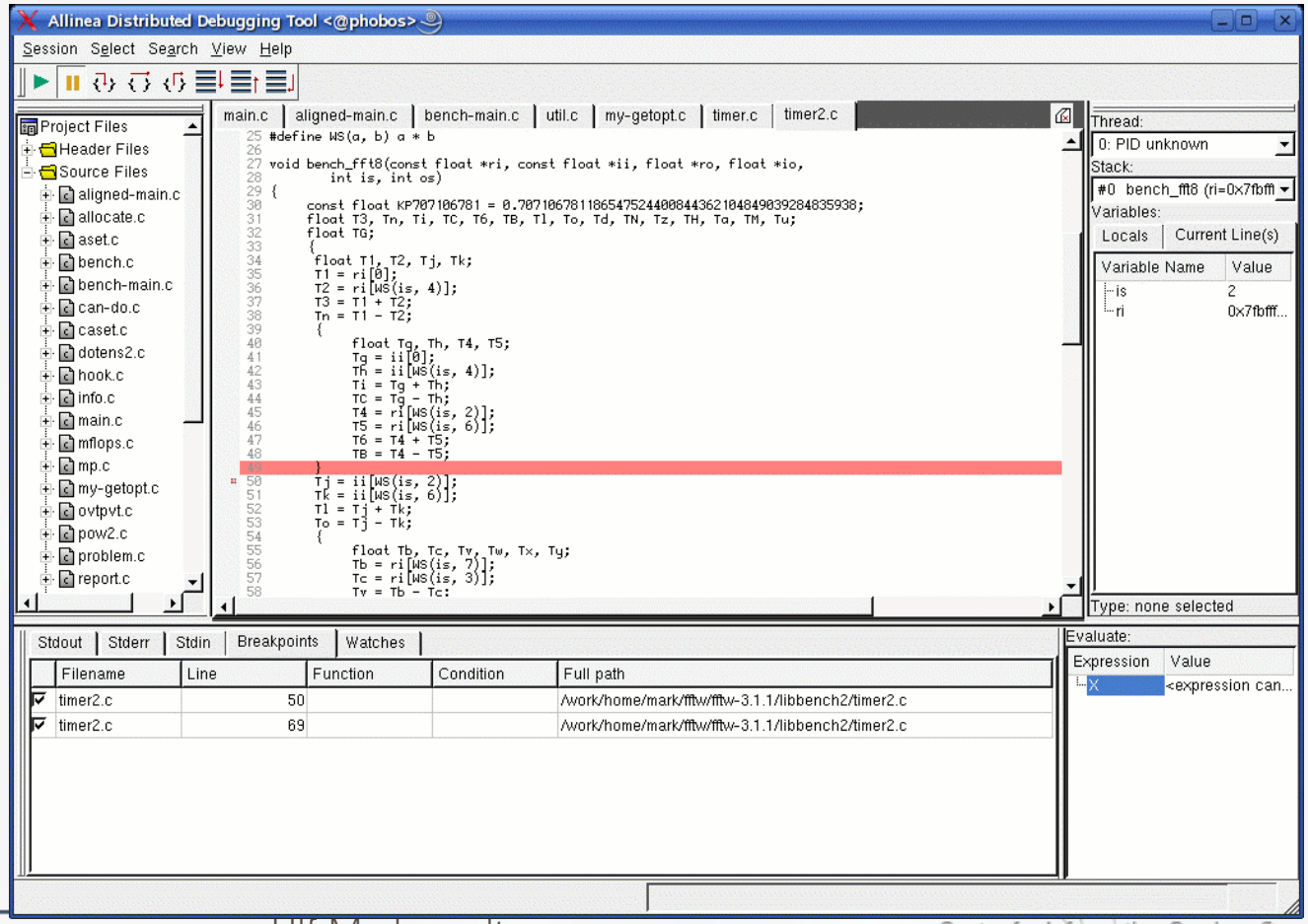
- gdb – GNU Debugger
  GUI: /usr/bin/ddd
  no threading support
- Intel: idb
- Pathscale: pathdb



Debuggers

Allinea DDT

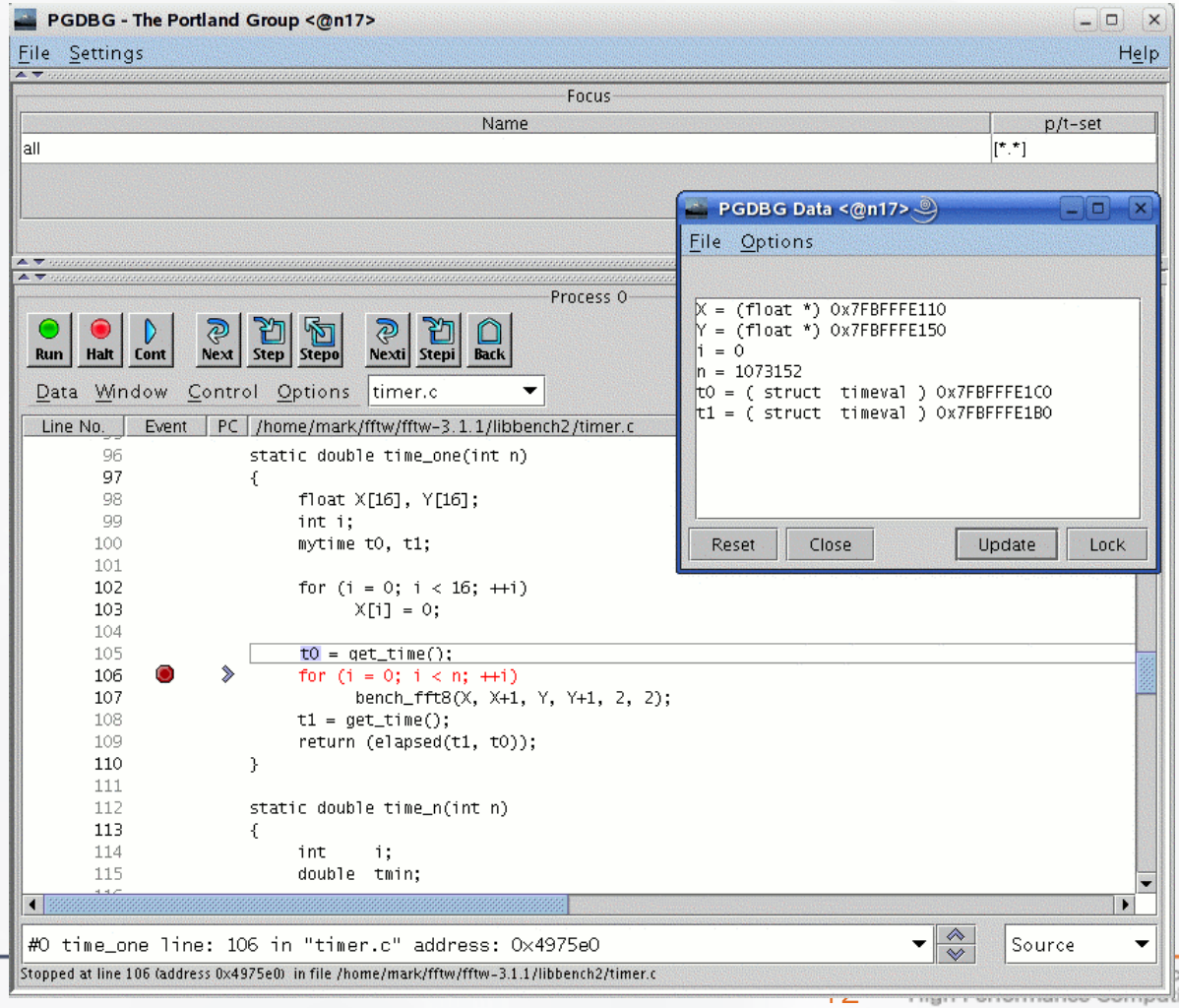
- `module load ddt`
- Multi-threaded and OpenMP applications
- MPI applications **NOT YET** debuggable



Debuggers

pgdgb

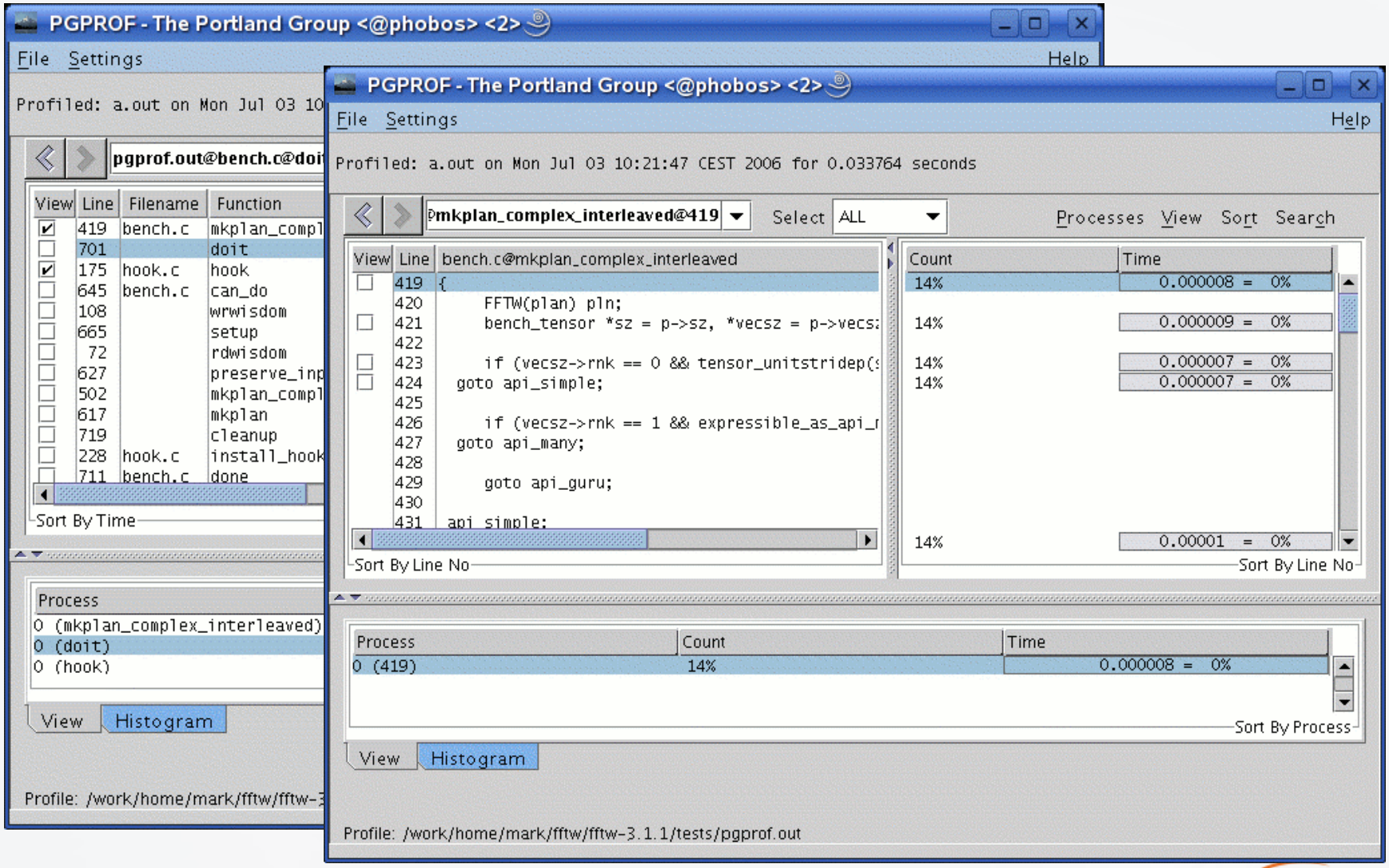
- `module load pgi`
- Multi-threaded and OpenMP applications
- MPI applications **NOT YET** debuggable



Profiling

- PathScale:
  - Compile code with `-pg`
  - Run the program
  - Start `pathprof`
- Gnu:
  - Compile code with `-pg`
  - Run the program
  - Start `gprof`

Profiling with pgprof



Batch System LSF

Commands

`bsub`	job submission
`bjobs`	display status of (default: running or pending) jobs (use **`-l <id>`** for more detailed information)
`/usr/local/bin/lsf_info`	display all running and pending jobs
`bkill <id>`	cancel job with given ID
`bqueues`	display queue information (use **`-l <queue>`** for more detailed information)
`bhosts`	display status of compute nodes

Submission examples

- Single CPU job (<10 min) submit: **`bsub -W 10 -o output.txt -e error.txt ./a.out`**
- MPI-parallel job submit (using default queue): **`bsub -n 16 -a mvapich mpirun.lsf ./a.out`**
- OpenMP job submit: **`bsub -q small -n 2 -x ./a.out`** (exclusively usage of a node)
- MPI-parallel, use only 1 CPU per node: **`bsub -q single_cpu -n 2 -x ./a.out`**
- MPI-parallel, use only 1 CPU per node: **`bsub -q single_cpu -n 2 -x ./a.out`**
- Job dependencies

```
bsub -J "Job1" run1
bsub -w 'done(Job1)' run2
```

Applications

- Abaqus 6.5.4
- Ansys
- LS-DYNA

- CFX 10.0
- Fluent 6.2

- Gaussian 3.0
- Gromacs
- NAMD 2.6

- MatLab 7.2

TECHNISCHE UNIVERSITÄT DRESDEN

Zentrum für Informationsdienste und Hochleistungsrechnen (ZIH)

FFT-Berechnungen auf phobos

Daniel Hackenberg – daniel.hackenberg@zih.tu-dresden.de

ZIH
Zentrum für Informationsdienste und Hochleistungsrechnen

FFT-Bibliotheken auf phobos

- Intel MKL:
  - include files in /opt/cluster/intel/mkl/mkl80/include
  - libraries in /opt/cluster/intel/mkl/mkl80/lib/em64t
  - alternativ: 'module load mkl/default' und dann Umgebungsvariablen $MKL_INC und $MKL_LIB nutzen
- AMD ACML:
  - include files in /opt/cluster/acml/2.7.0/include
  - libraries in /opt/cluster/acml/2.7.0/lib
  - alternativ: 'module load acml/default' und dann Umgebungsvariablen $ACML_INC und $ACML_LIB nutzen
- FFTW (Fastest Fourier Transform in the West):
  - freie Software (GPL), erhältlich unter fftw.org
  - auf phobos nicht verfügbar, muss manuell installiert werden

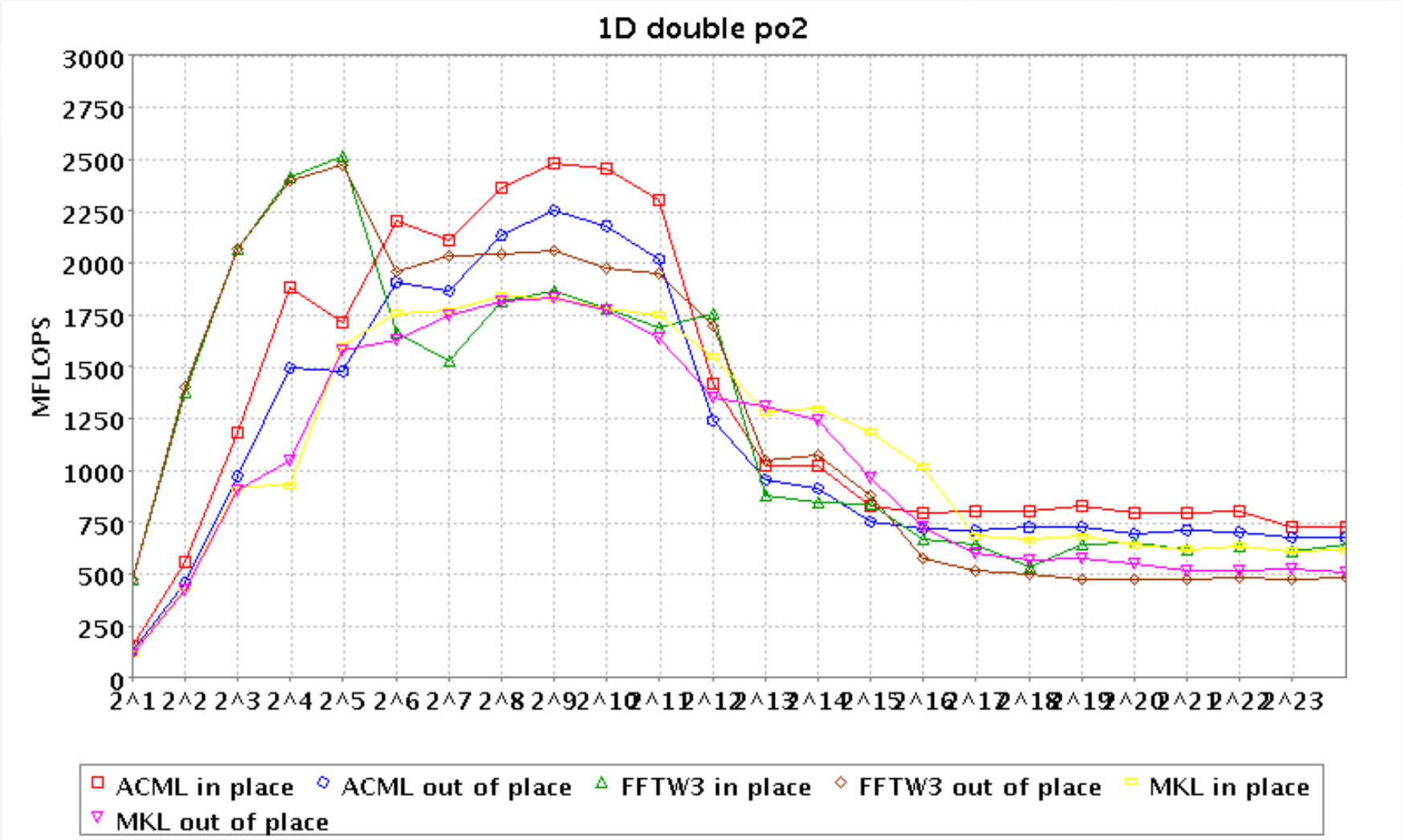
FFTW Installation

- Download unter http://www.fftw.org/download.html (derzeit Version 3.1.1)
- ./configure CC="icc" –prefix="/home/username/fftw3.1.1"
  - für single precision: “--enable-float”
  - für long-double precision: “--enable-long-double”
- make, make install

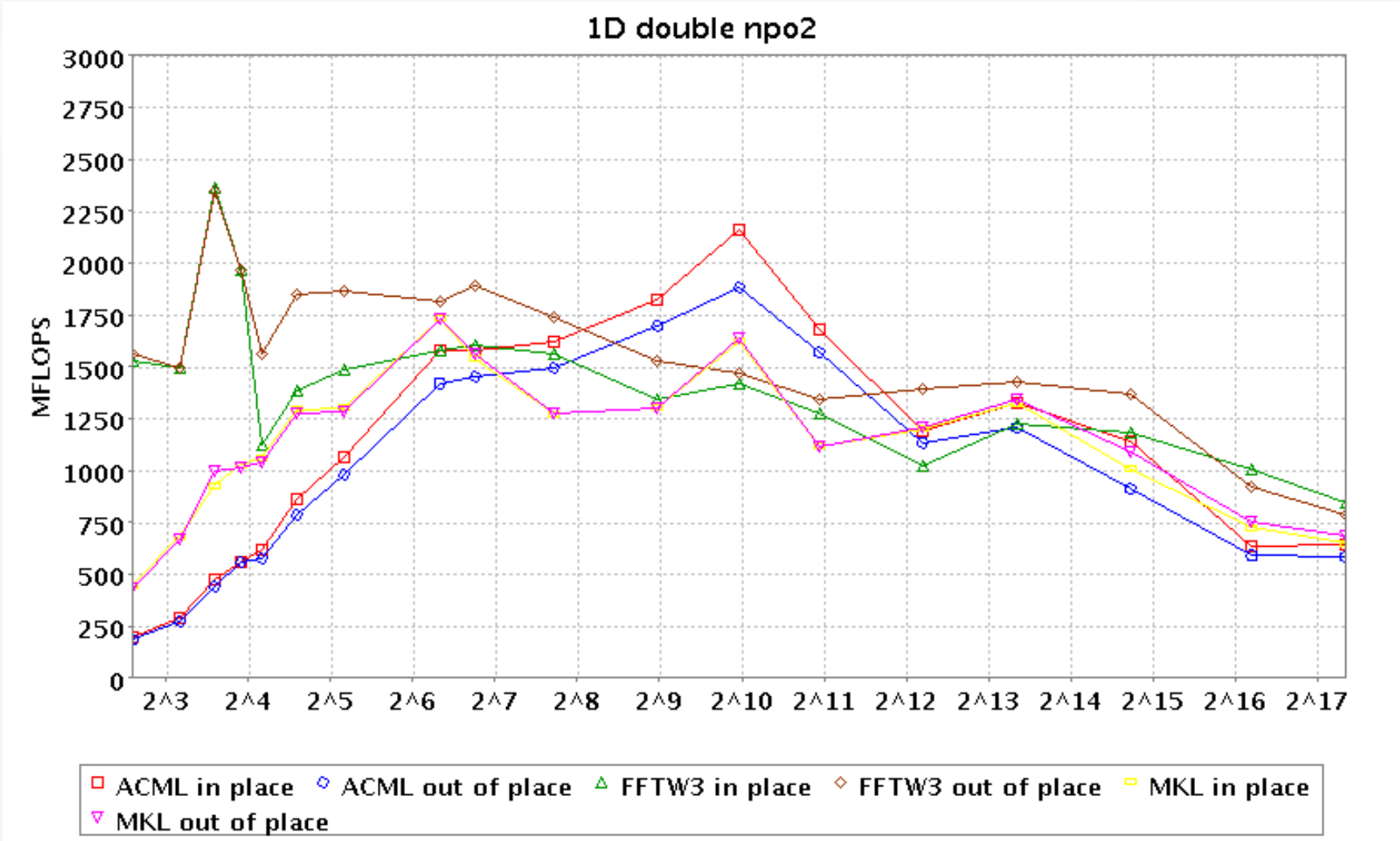
Dokumentation

- ACML: /opt/cluster/acml/2.7.0/Doc/acml.pdf (S. 19 ff.)
- MKL: /opt/cluster/intel/mkl/mkl80/doc/mklman.pdf (S. 11-1 ff.)
- FFTW: http://www.fftw.org/fftw3_doc/

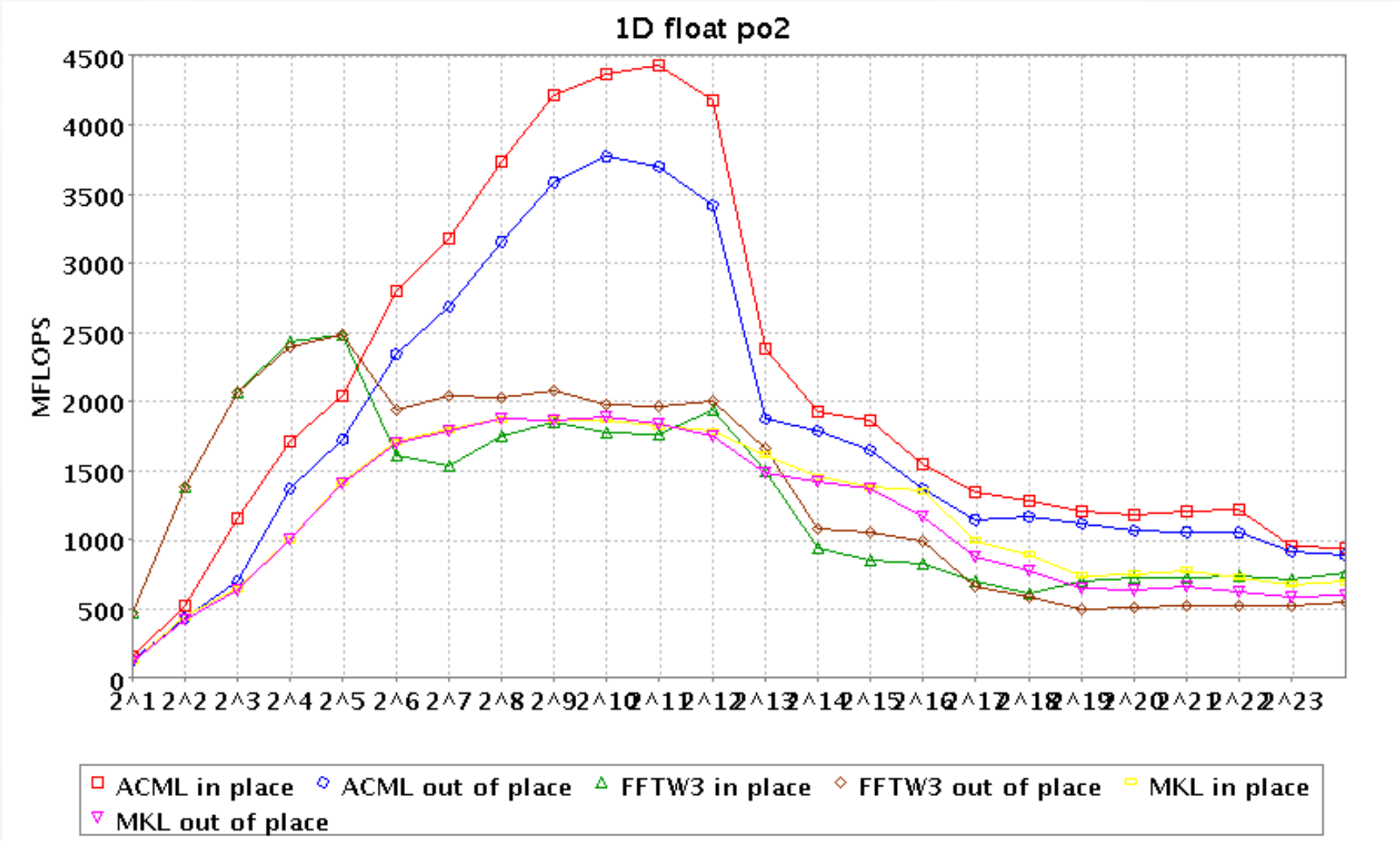
FFT: 1D, double precision, powers of two



FFT: 1D, double precision, non-powers of two



FFT: 1D, single precision, powers of two



Hinweise

- bei sehr vielen gleichartigen FFTs:
  - FFTW: Performance bei Plangenerierung mit “FFTW_MEASURE” statt “FFTW_ESTIMATE” testen (sucht besten Algorithmus durch Messung, Plangenerierung dauert entsprechend länger)
  - ACML: Initialisierung mit MODE=100, Effekt wie oben
- bei 2D/3D-FFTs: OpenMP-Parallelisierung mit MKL auf 2 CPUs möglich (OMP_NUM_THREADS=2)
- FFT ist für Zweierpotenzen am schnellsten, ggf. möglichst kleine Primfaktioren nutzen
- ACML bei Primzahlen zum Teil um mehr als Faktor 100 langsamer als MKL; bei nicht kontrollierbarer Problemgröße somit unbrauchbar!

TECHNISCHE UNIVERSITÄT DRESDEN

Center for Information Services and High Performance Computing (ZIH)

VAMPIR Performance Visualization

Software-Tuning on the Phobos PC-Farm
July 4th, 2006

Zellescher Weg 12

Willers-Bau A25 rechts

Tel. +49 351 - 463 - 35048

Holger Brunst (holger.brunst@tu-dresden.de)

Vampir History

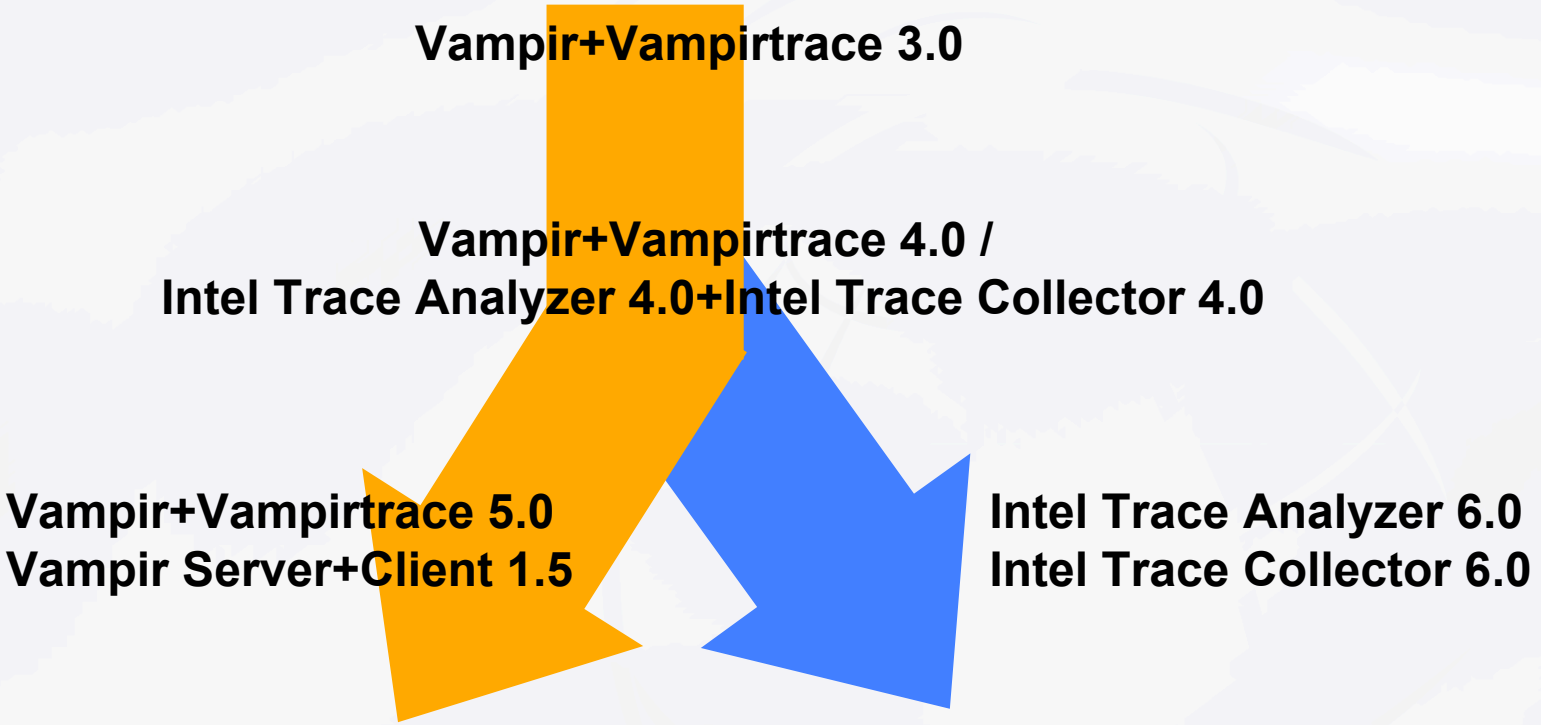
PARvis at Research Center Jülich

- Vampir at Research Center Jülich

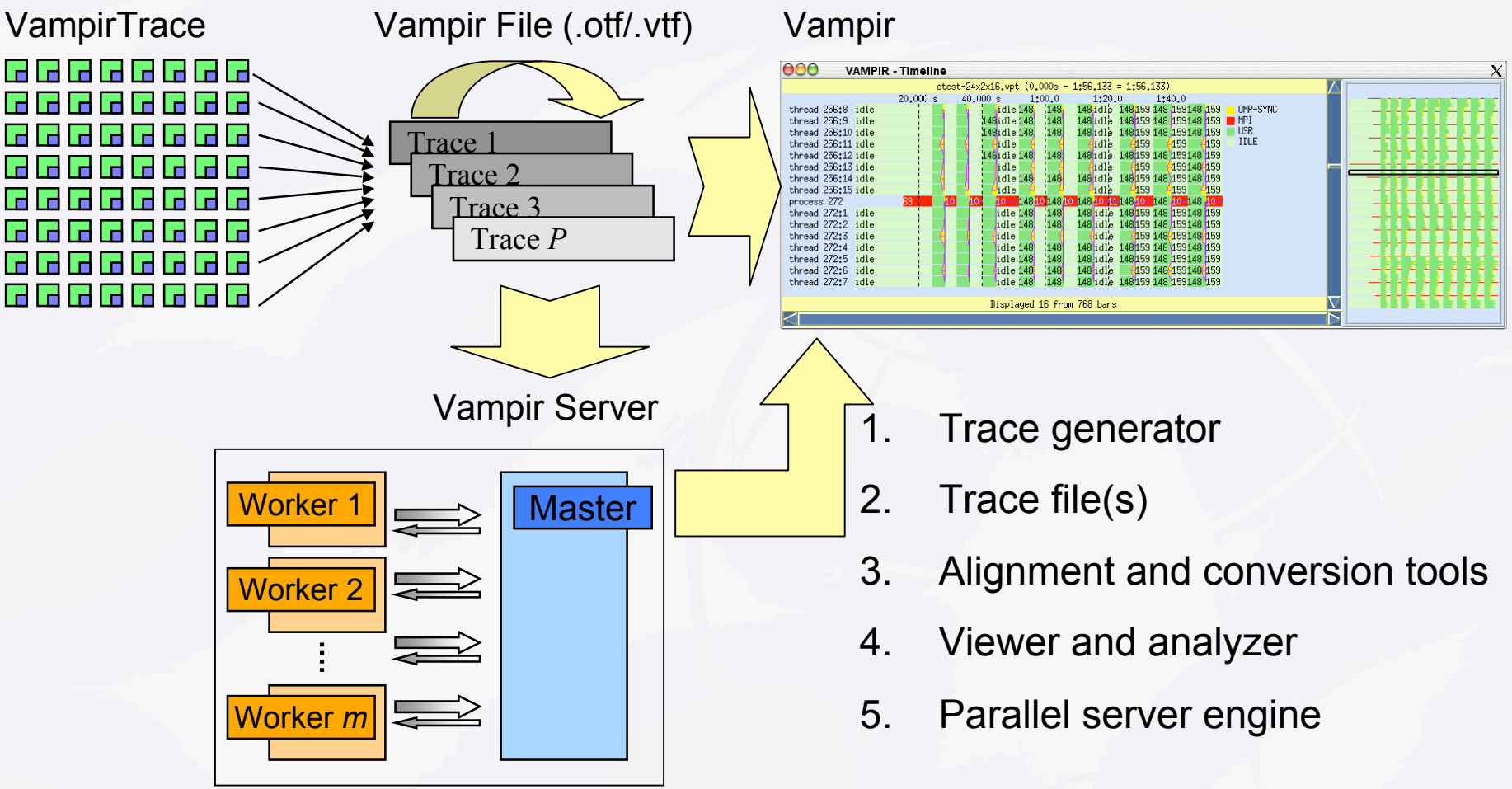
  http://www.top500.org/reports/1995/vampir/vampir.html

- Vampir at ZIH, TU Dresden
  - was commercially available via Pallas GmbH, later via Intel
  - Vampir (GUI) was developed by ZIH, Dresden
  - Vampirtrace was developed by Pallas/Intel
- Successor: Vampir NG (next Generation)
  - client/server architecture
  - distributed storage, more scalable

Vampir Versions

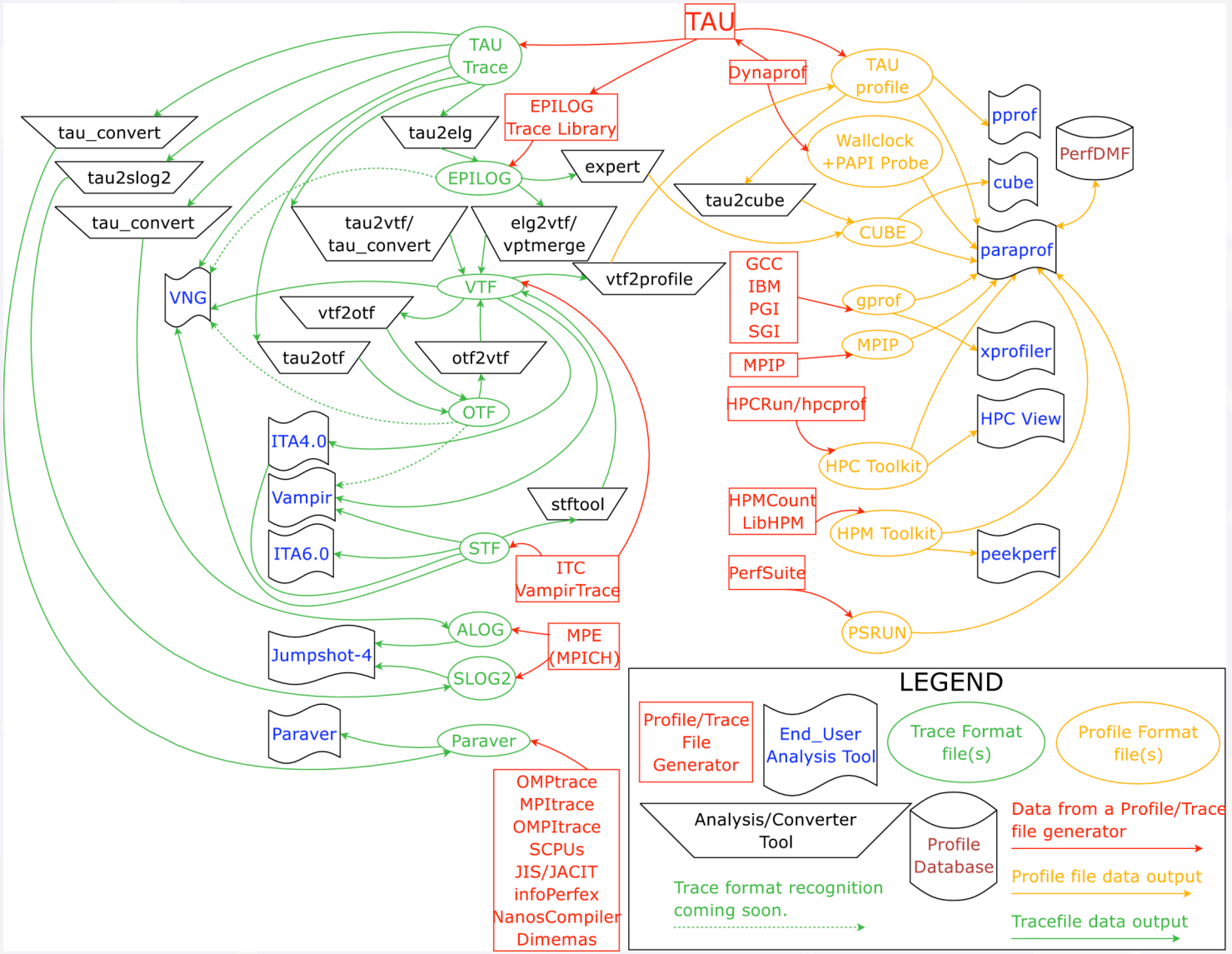


Vampir: technical components



1. Trace generator
2. Trace file(s)
3. Alignment and conversion tools
4. Viewer and analyzer
5. Parallel server engine

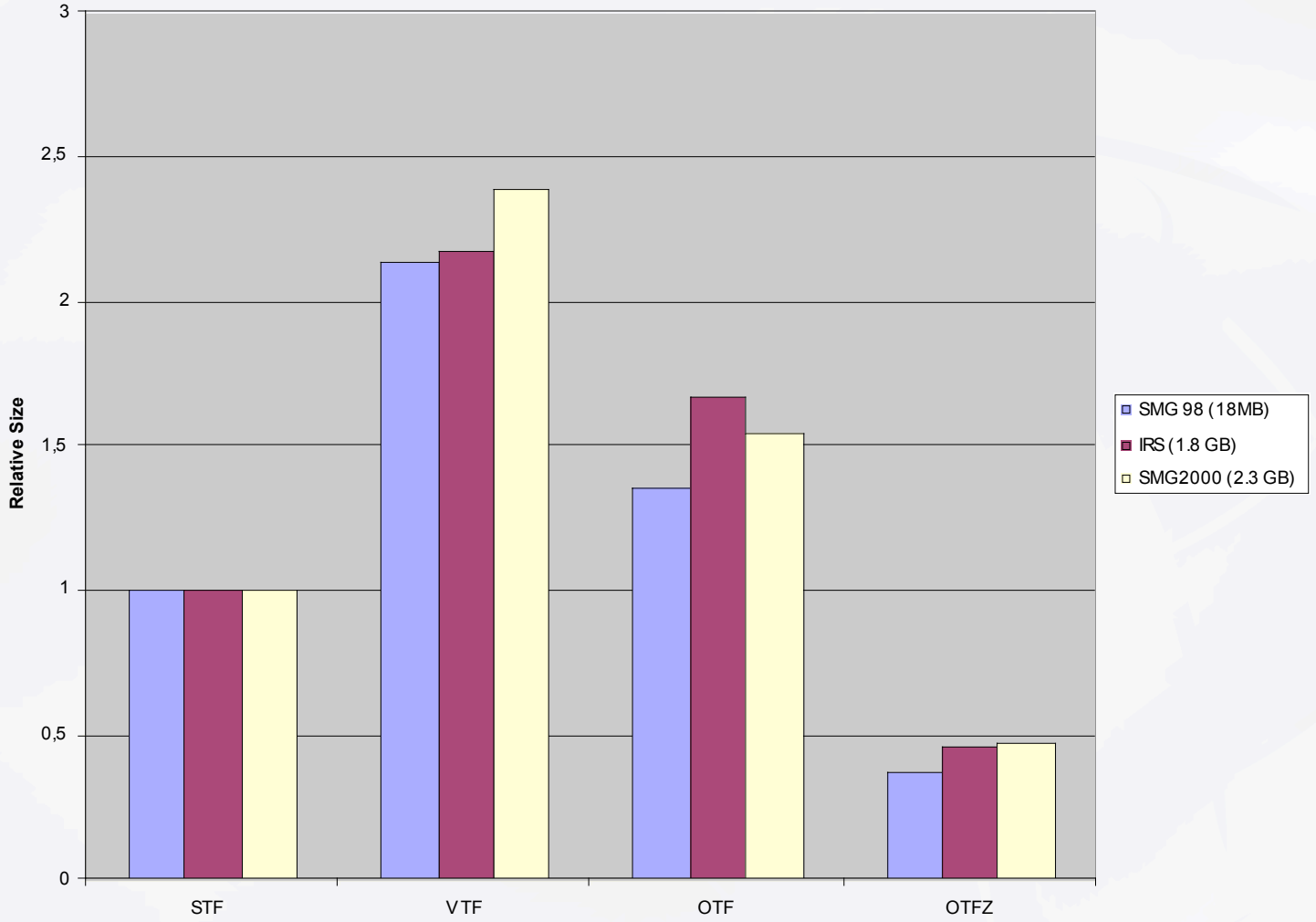
Many trace formats to choose from …



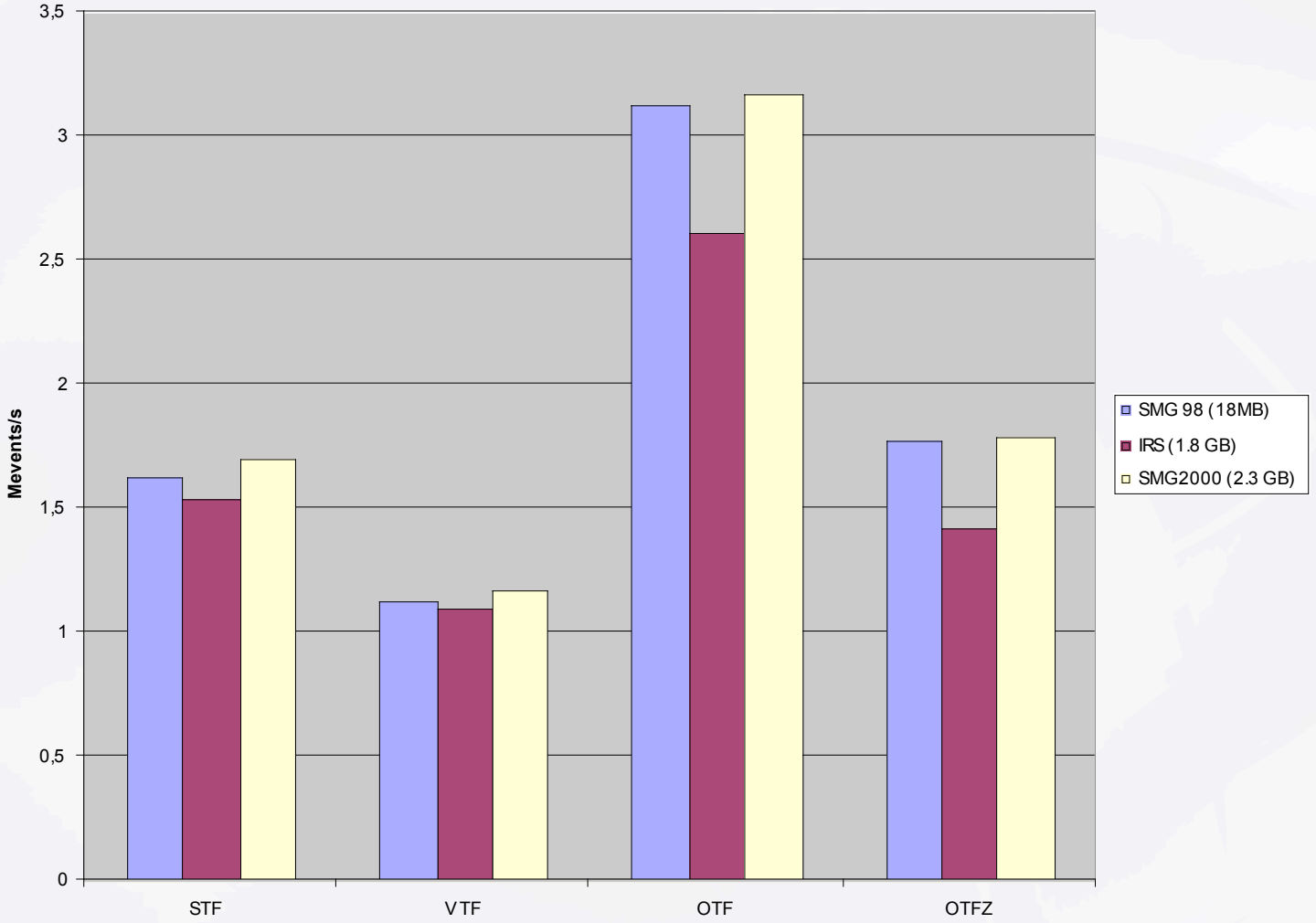
OTF Features

- Fast and efficient sequential and parallel access
- Platform independent
- Selective access to
  - Processes
  - Time intervals
- API / Interfaces
  - High level interface for analysis tools
    - Read/write complete traces with multiple files
    - Supports filtering and parallel I/O
  - Low level interface for trace libraries

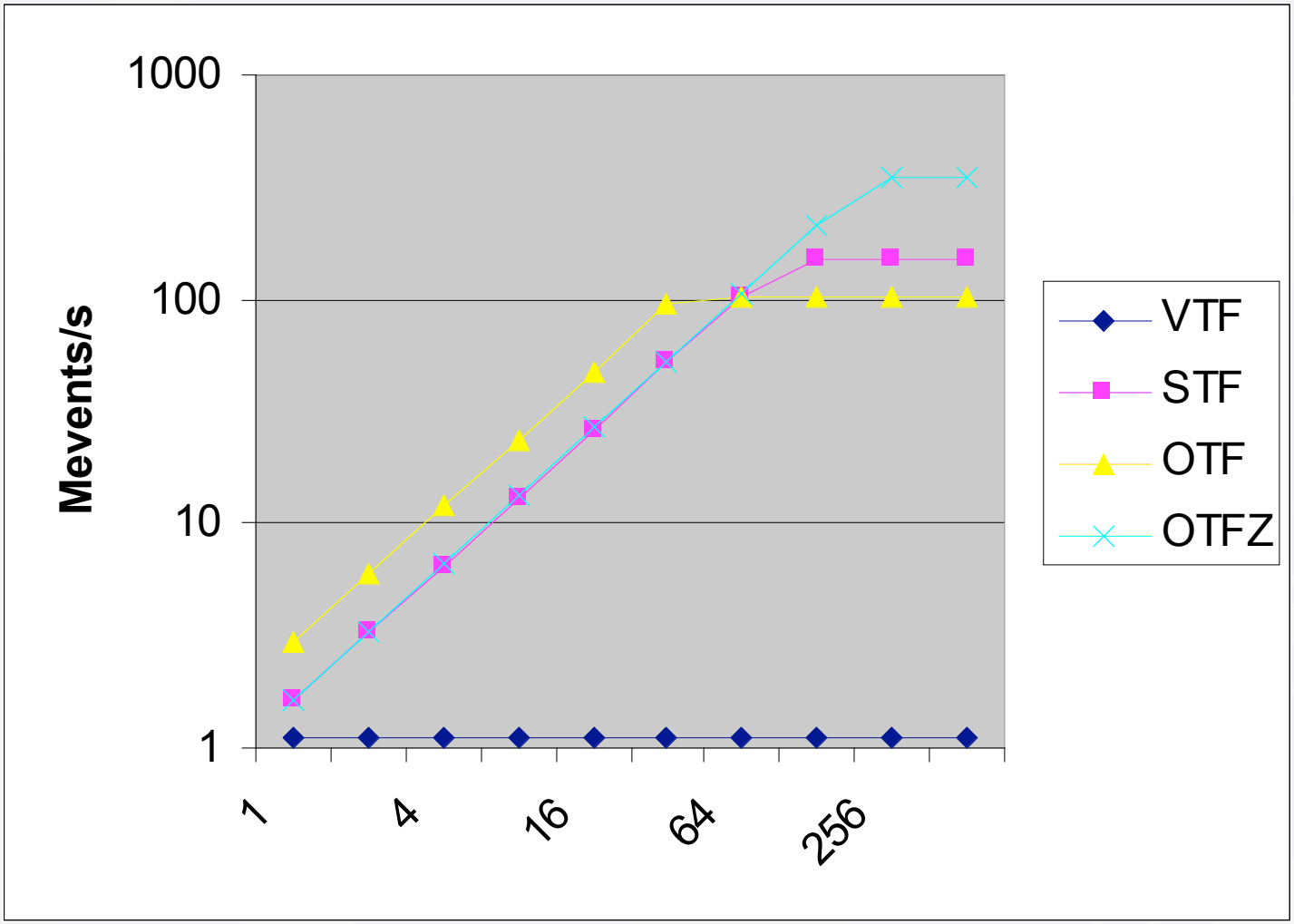
Relative File Size



Read Performance



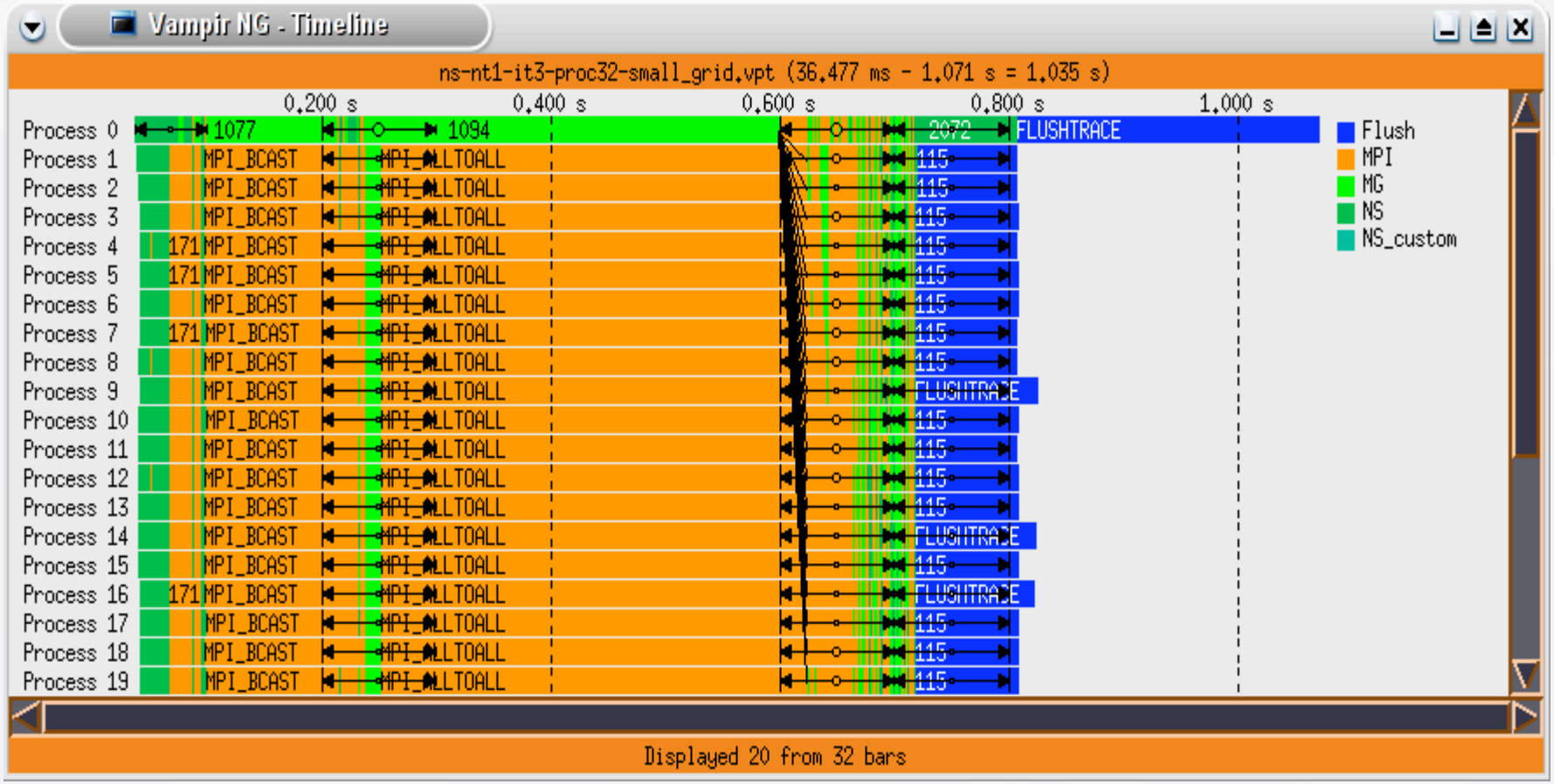
Performance Scalability



Vampir Displays

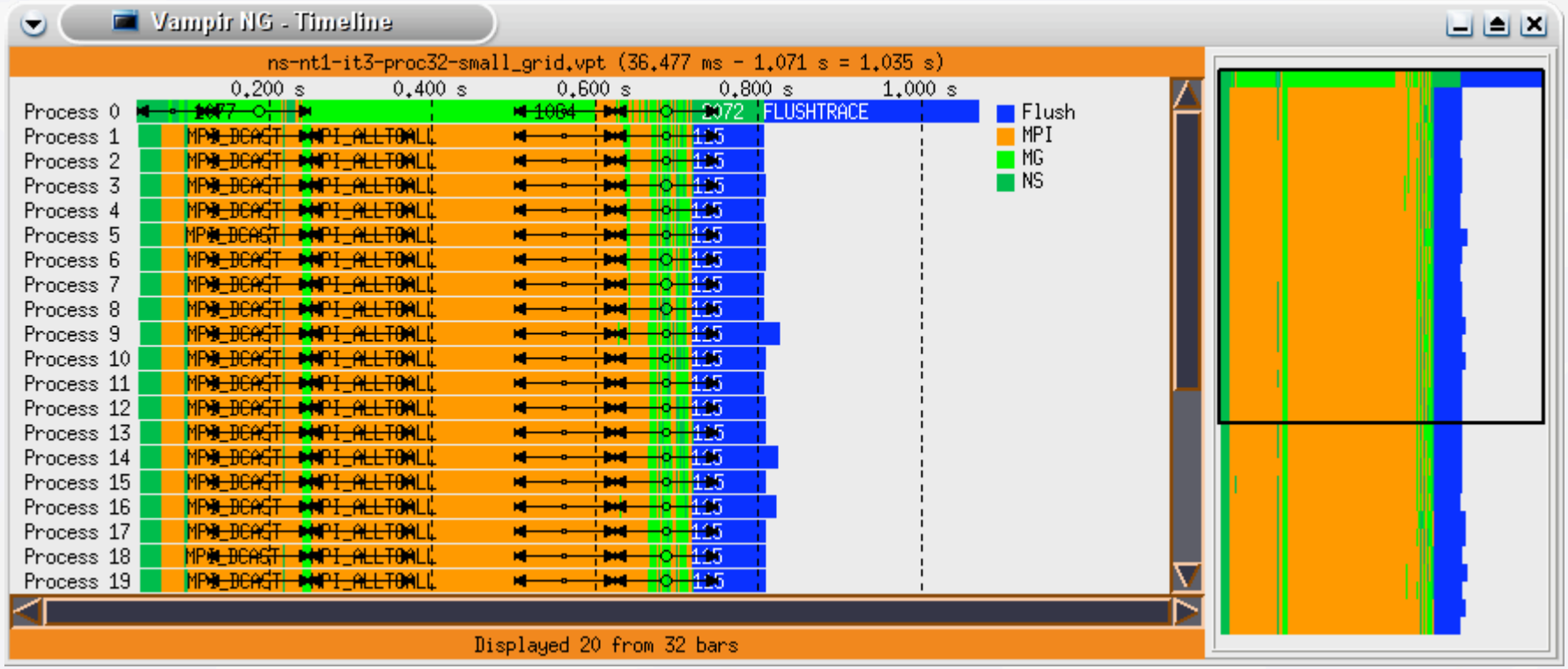
- Global Timeline
- Process Timeline
- Summary Chart
- Summary Timeline
- Counter Timeline
- Process Profile
- Call Tree
- Message Statistics
- Navigator
- Global Filters
- OpenMP

Vampir Displays - Global Timeline



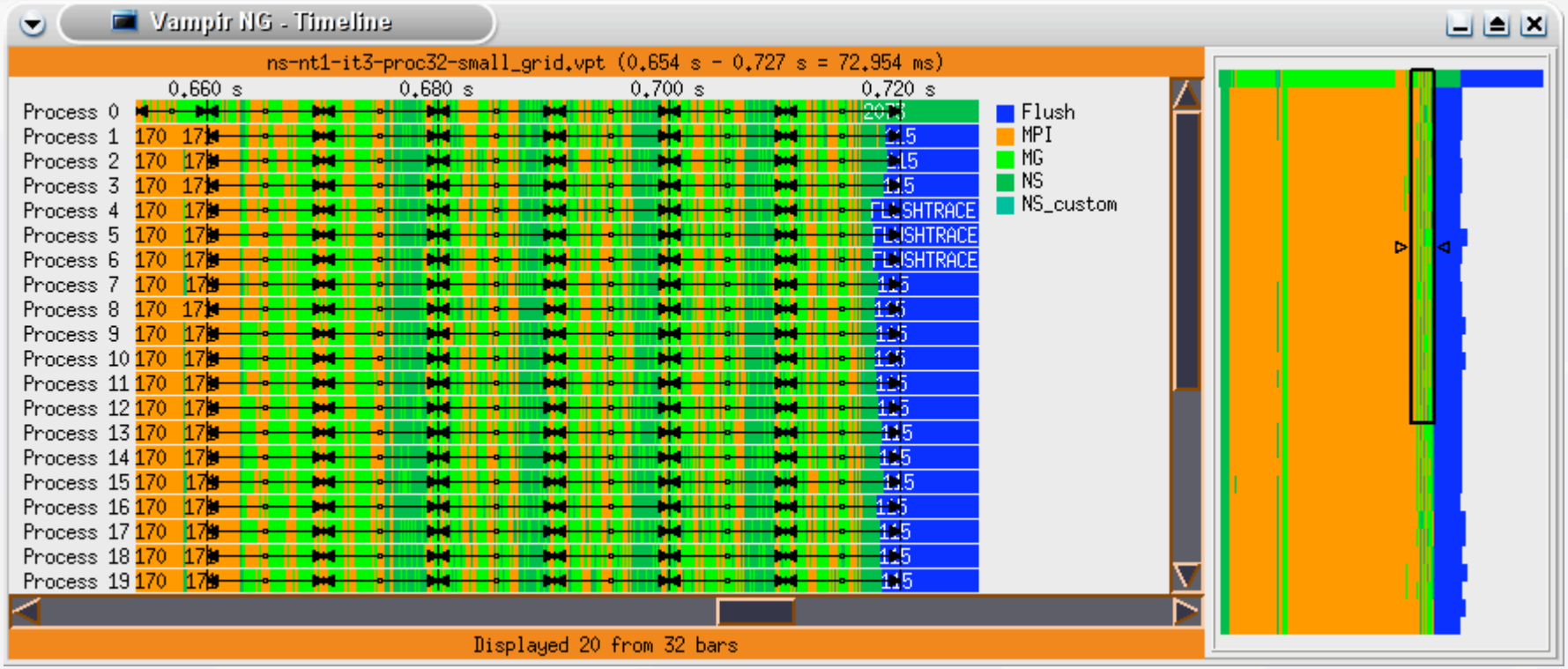
Global Timeline

Vampir Displays - Global Timeline with Thumbnail



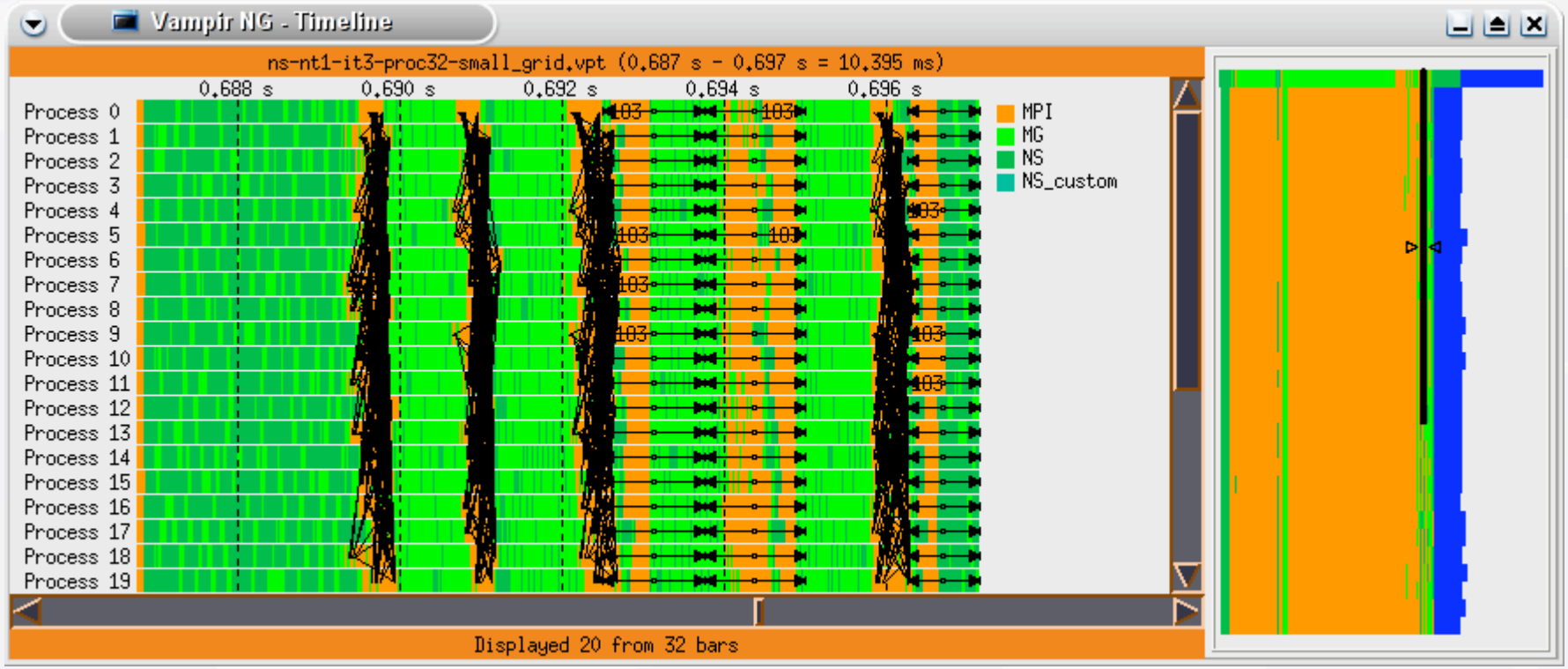
Global Timeline with Thumbnail View

Vampir Displays - Global Timeline with Thumbnail



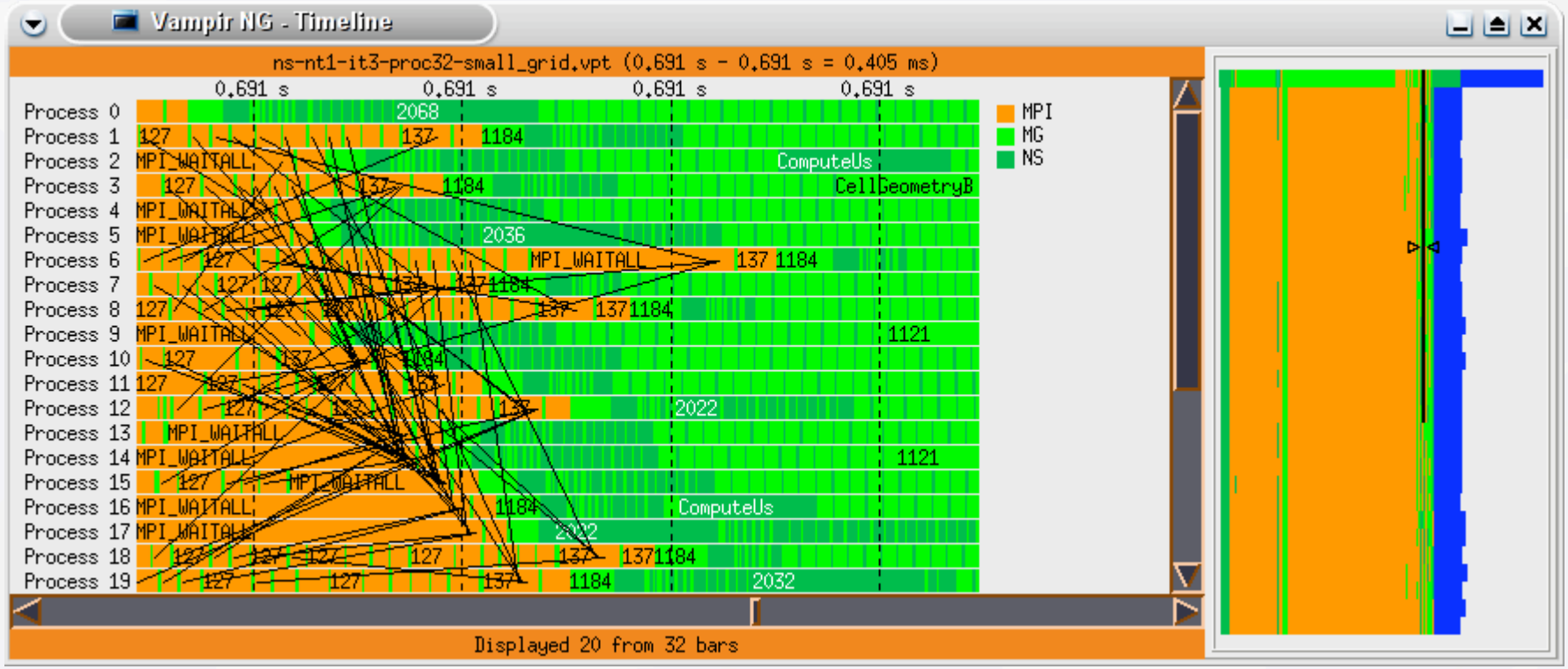
Zoomed

Vampir Displays - Global Timeline with Thumbnail



Further Zoomed

Vampir Displays - Global Timeline with Thumbnail

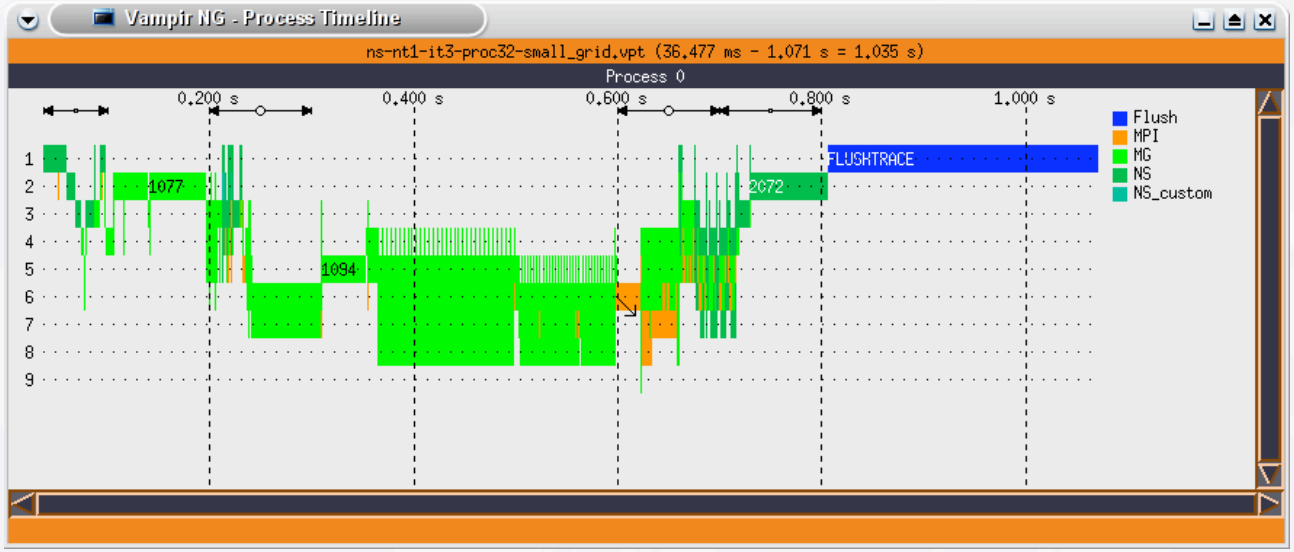


Even More Zoomed

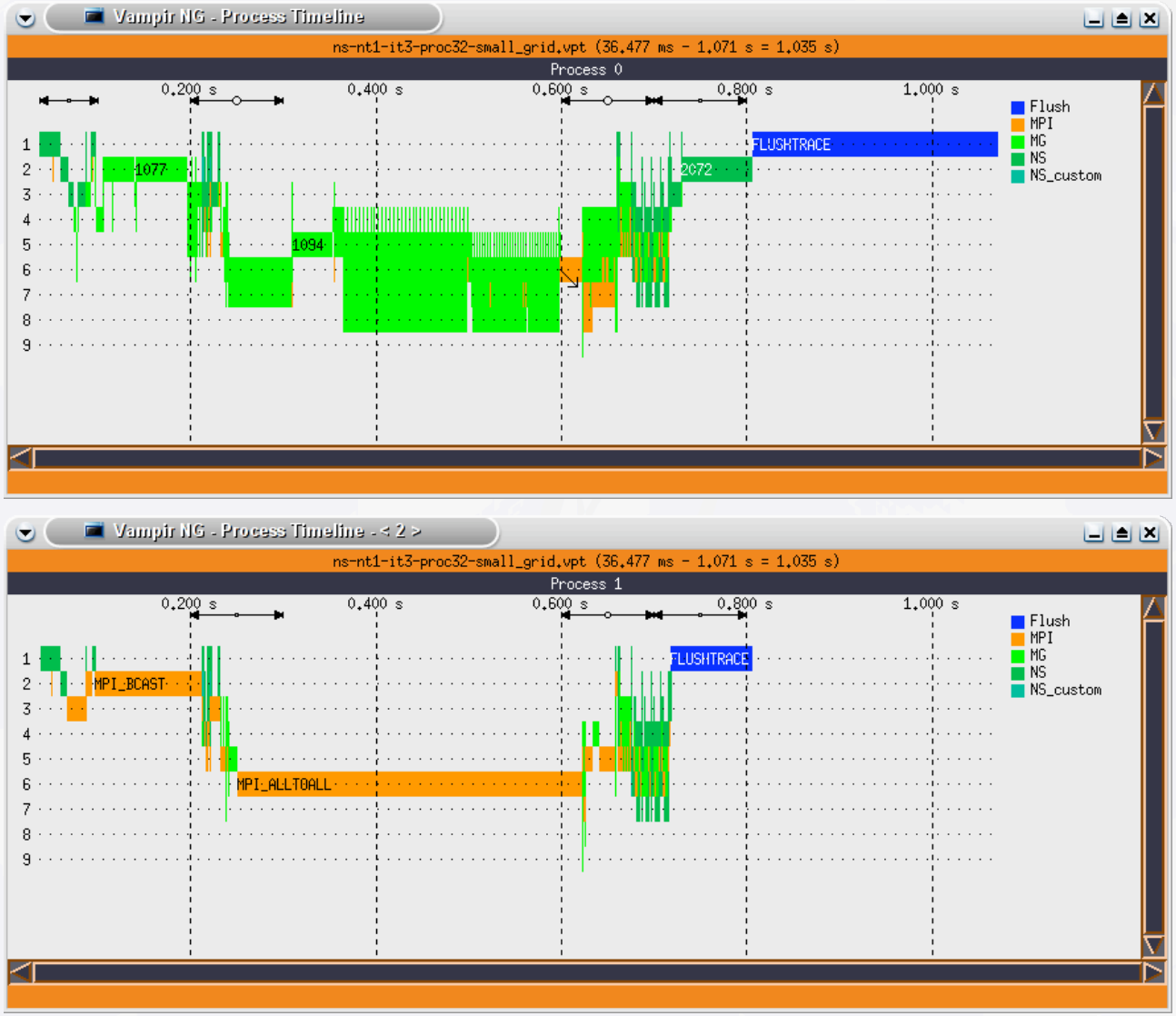
Vampir Displays - Global Timeline

- show temporal behavior for
  - processes
  - functions resp. function groups
  - messages
  - collective ops
  - I/O activities
- context information on click
- zoomable
- control actual time interval for other displays

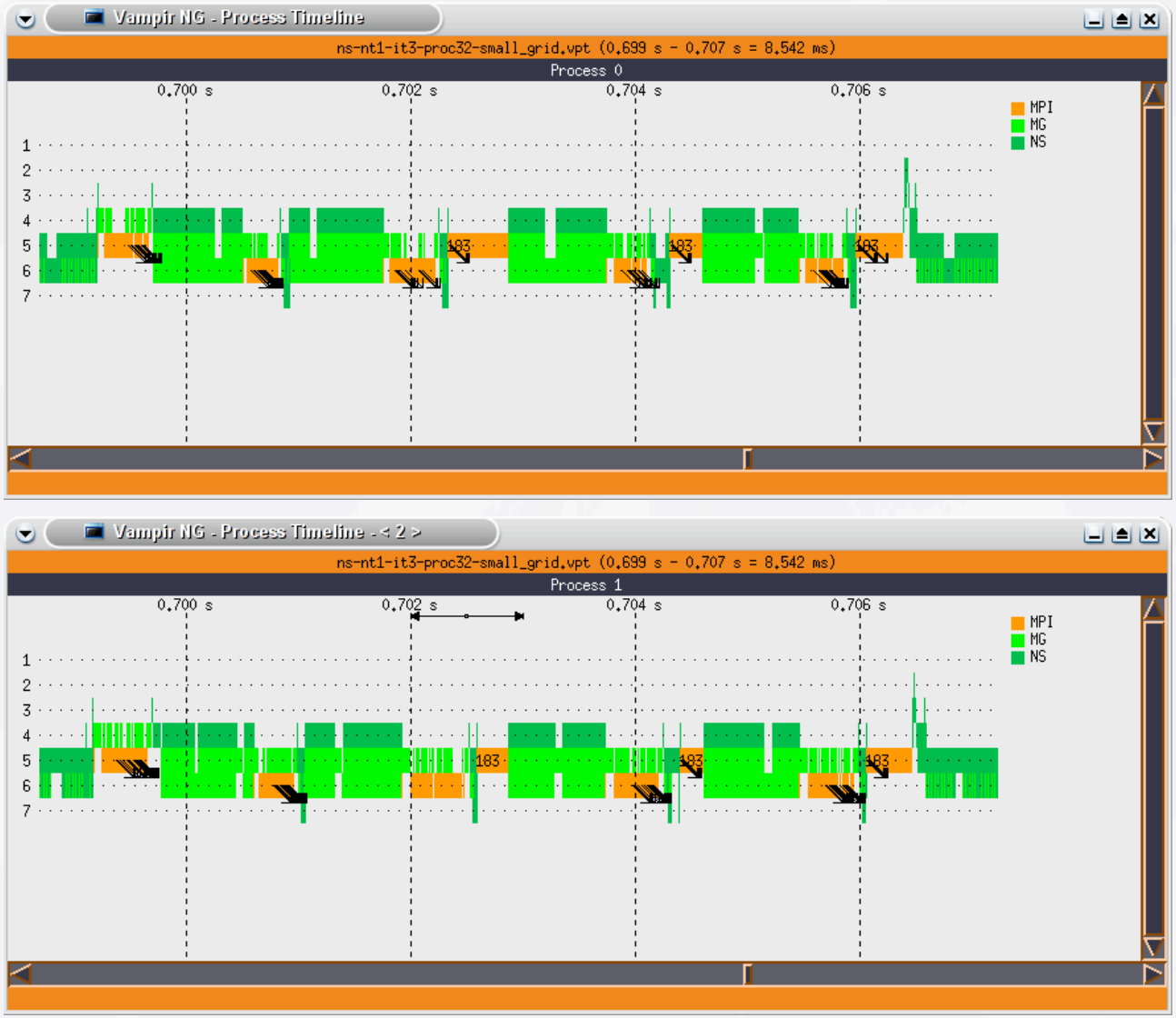
Vampir Displays - Process Timeline



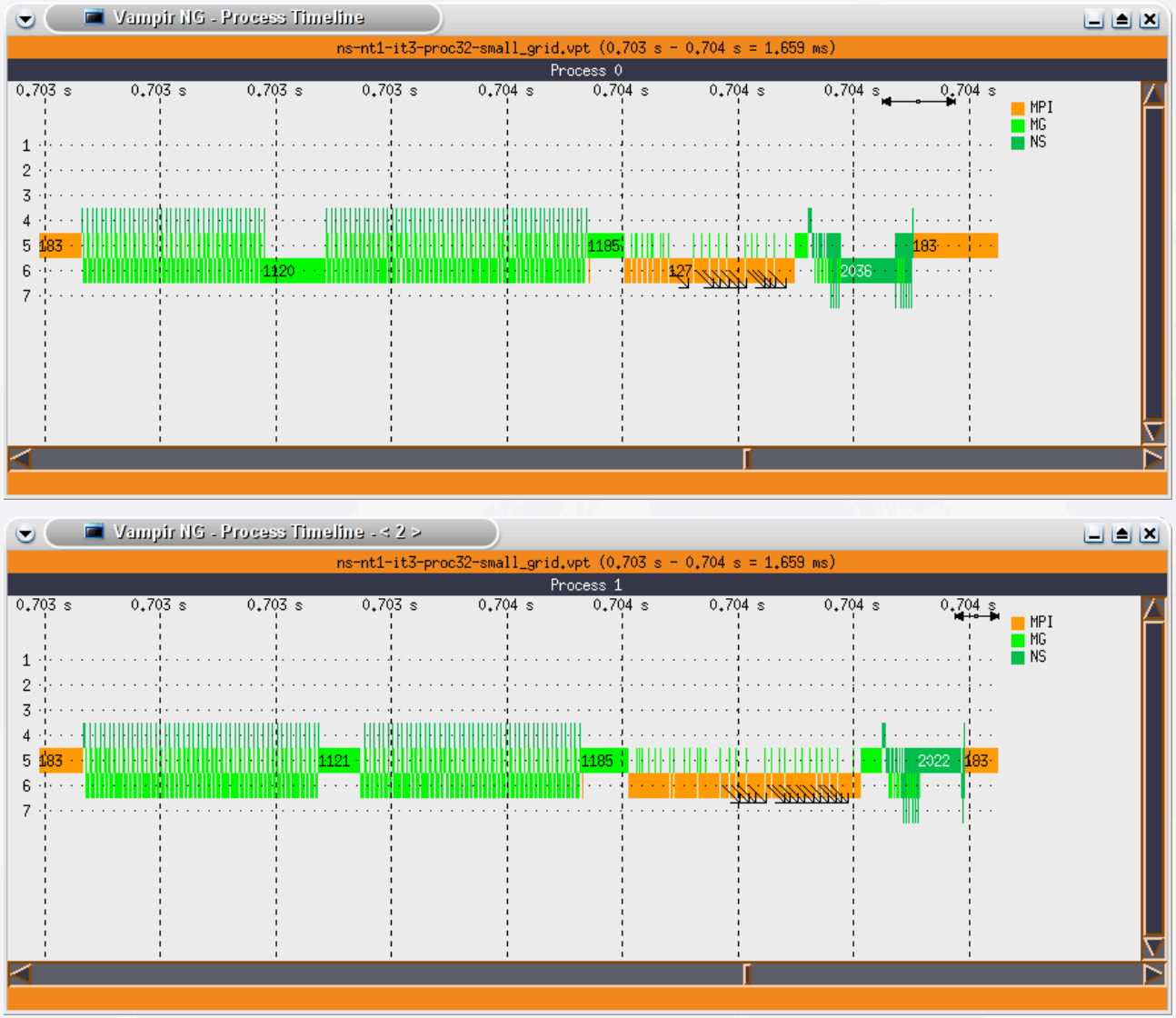
Vampir Displays – Aligned Process Timelines



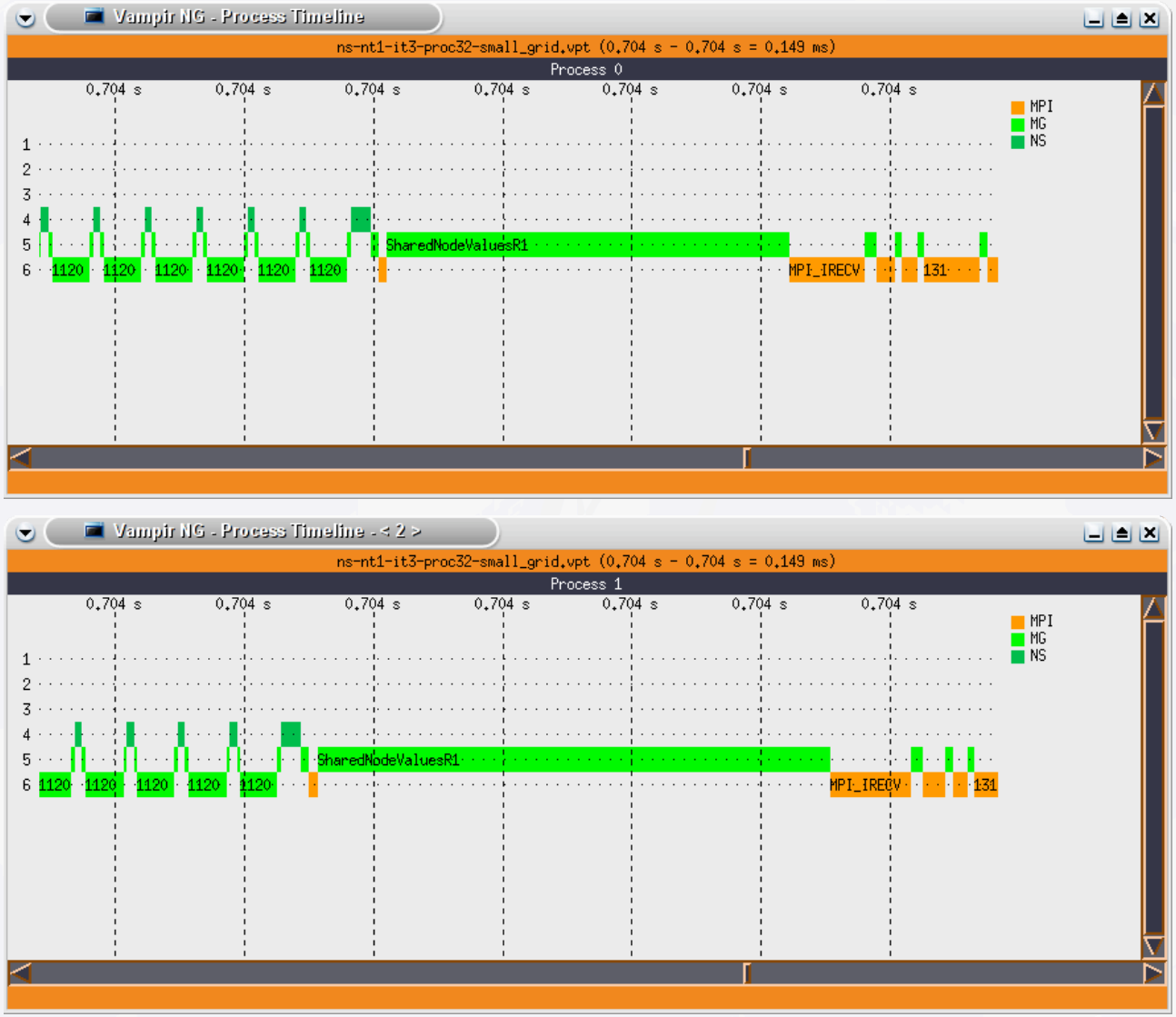
Vampir Displays - Process Timeline zoomed



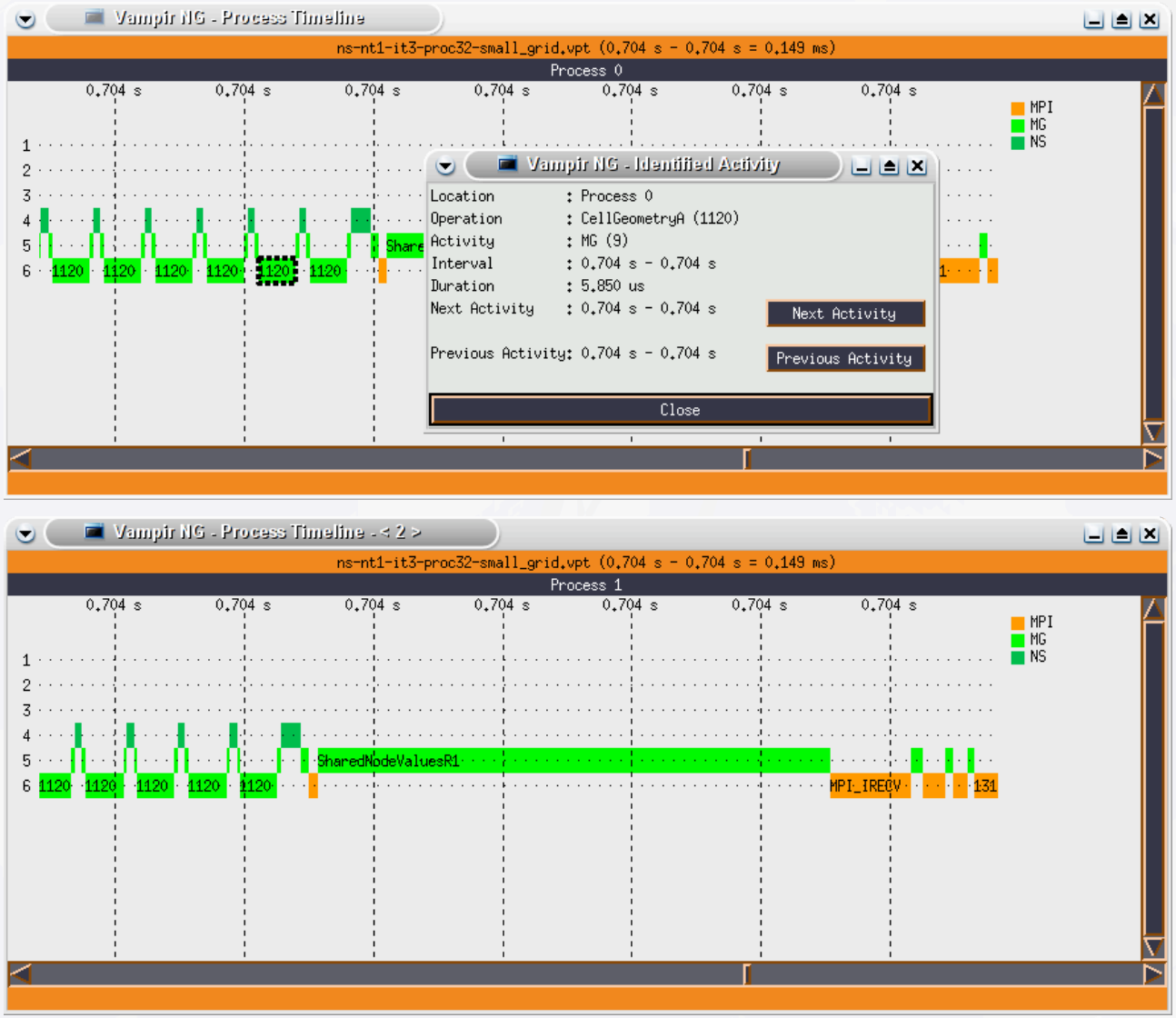
Vampir Displays - Process Timeline further zoomed



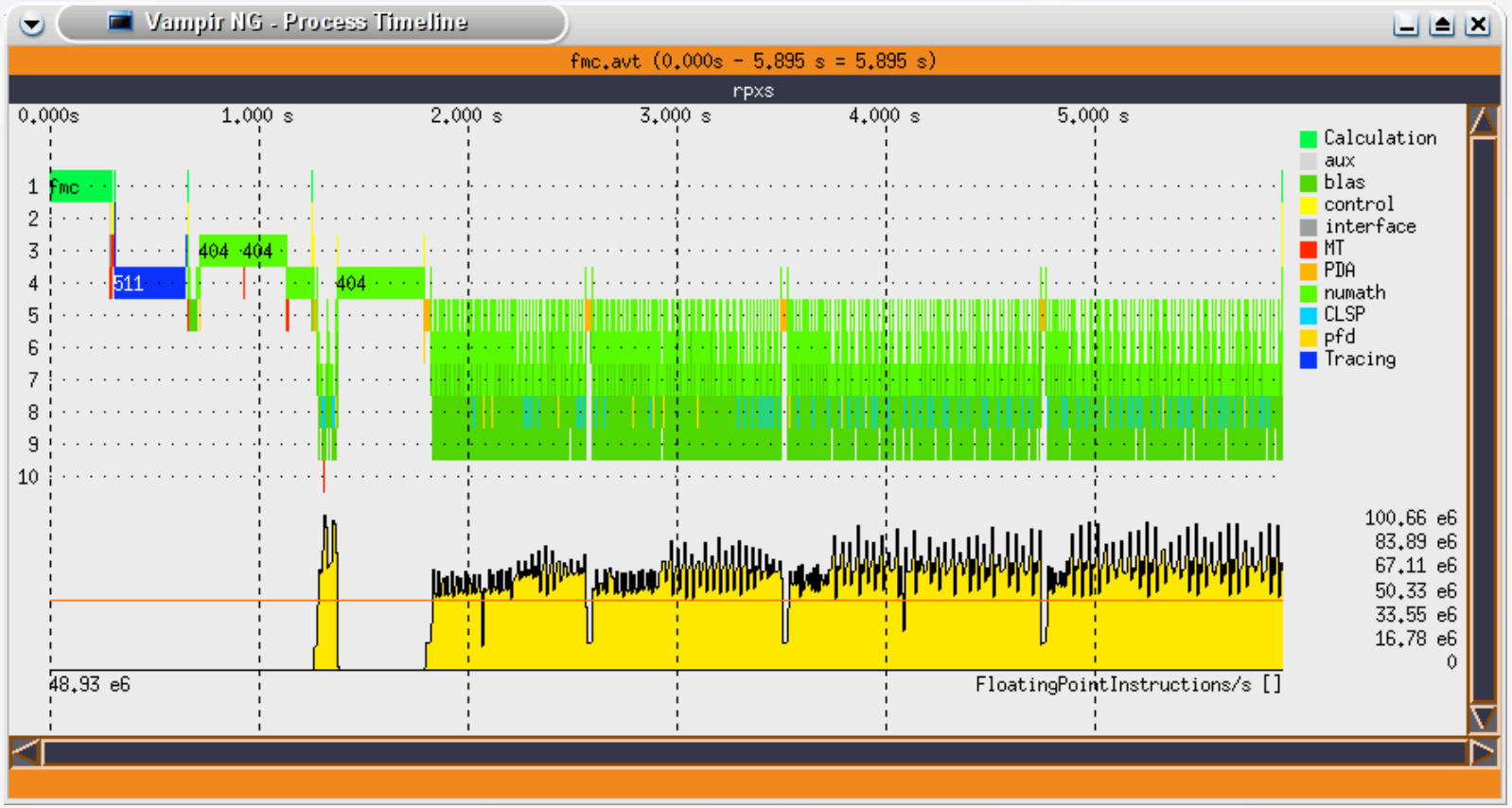
Vampir Displays - Process Timeline further zoomed II



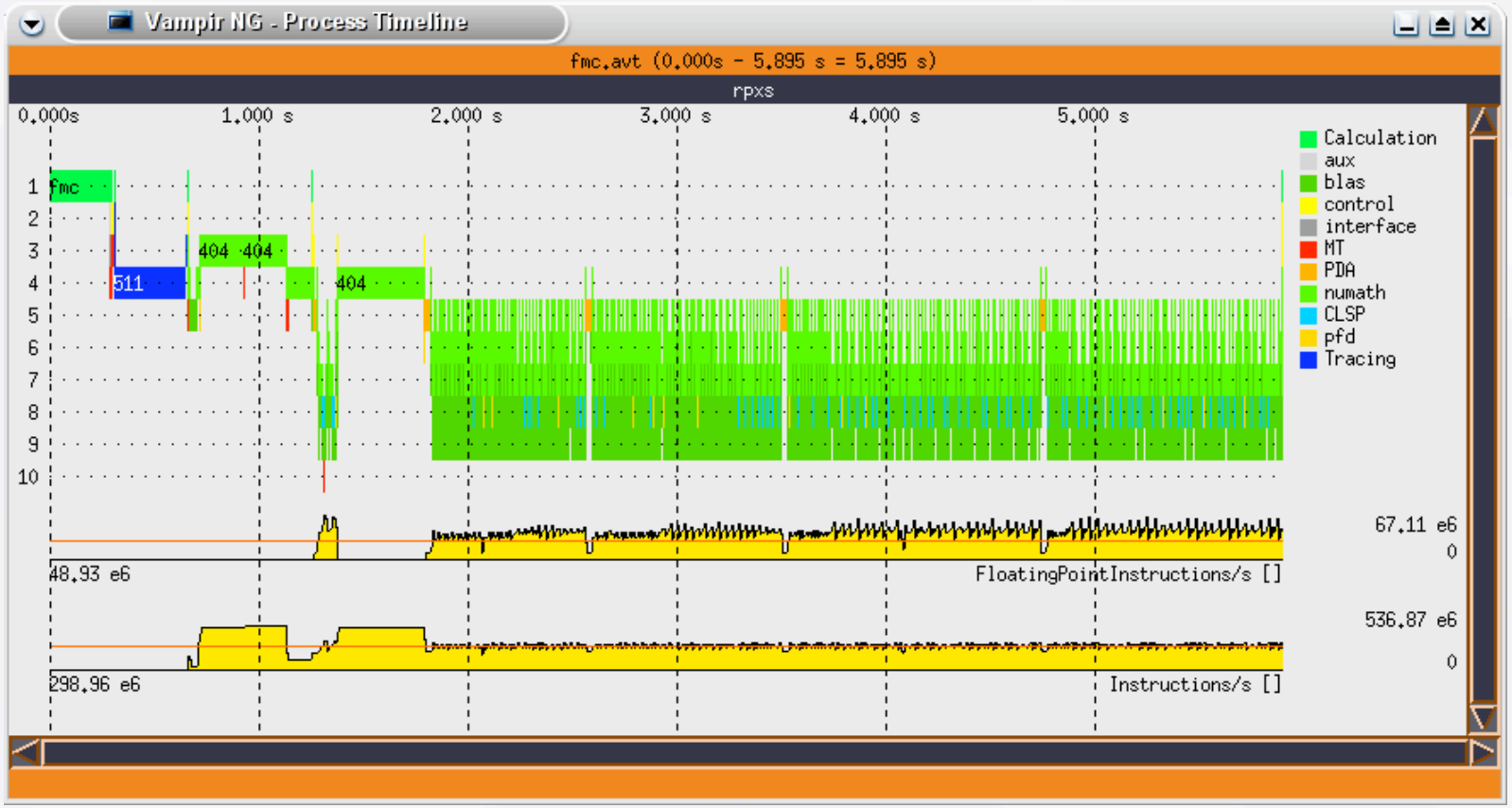
Vampir Displays - Process Timeline with context



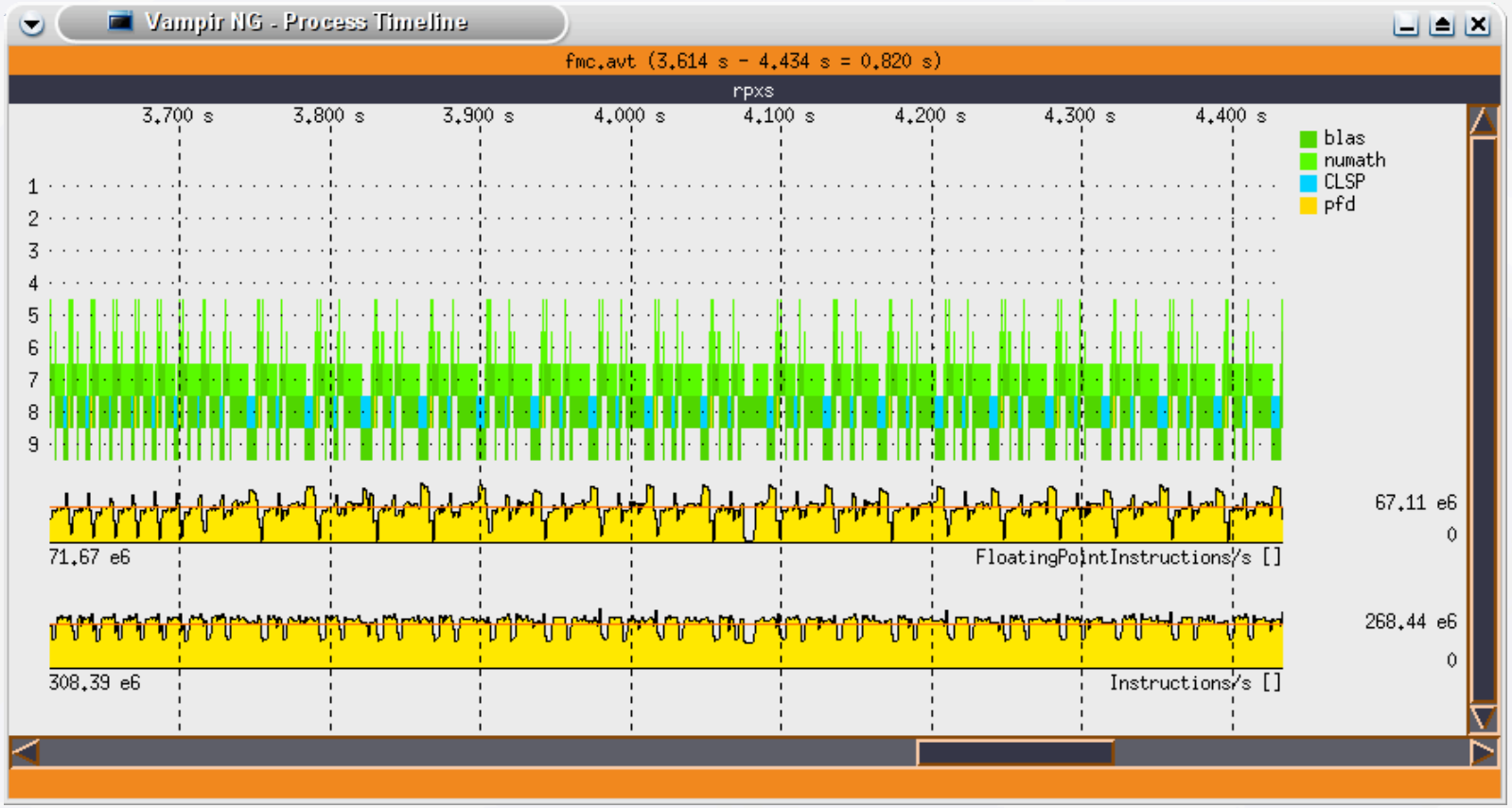
Vampir Displays - Process Timeline with Flop/s



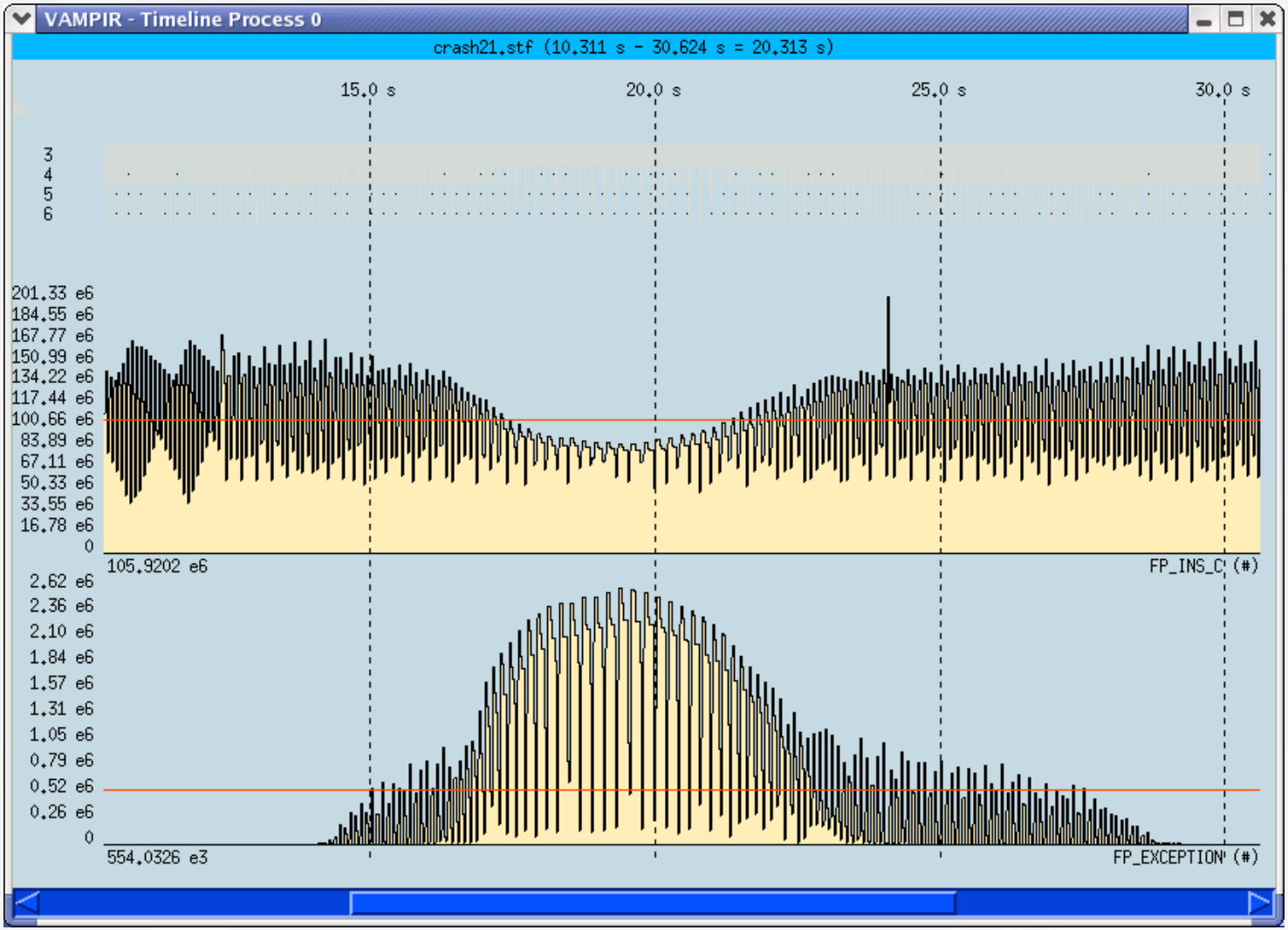
Vampir Displays - Process Timeline with two counters



Vampir Displays - Process Timeline zoomed



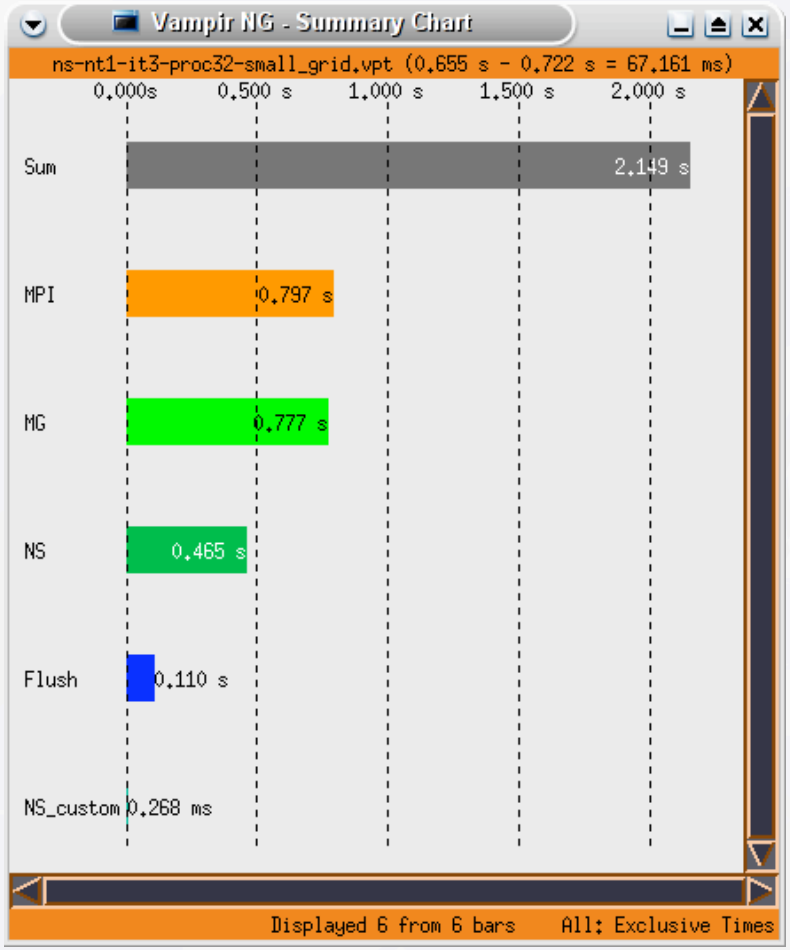
Vampir Displays - Counter Timeline

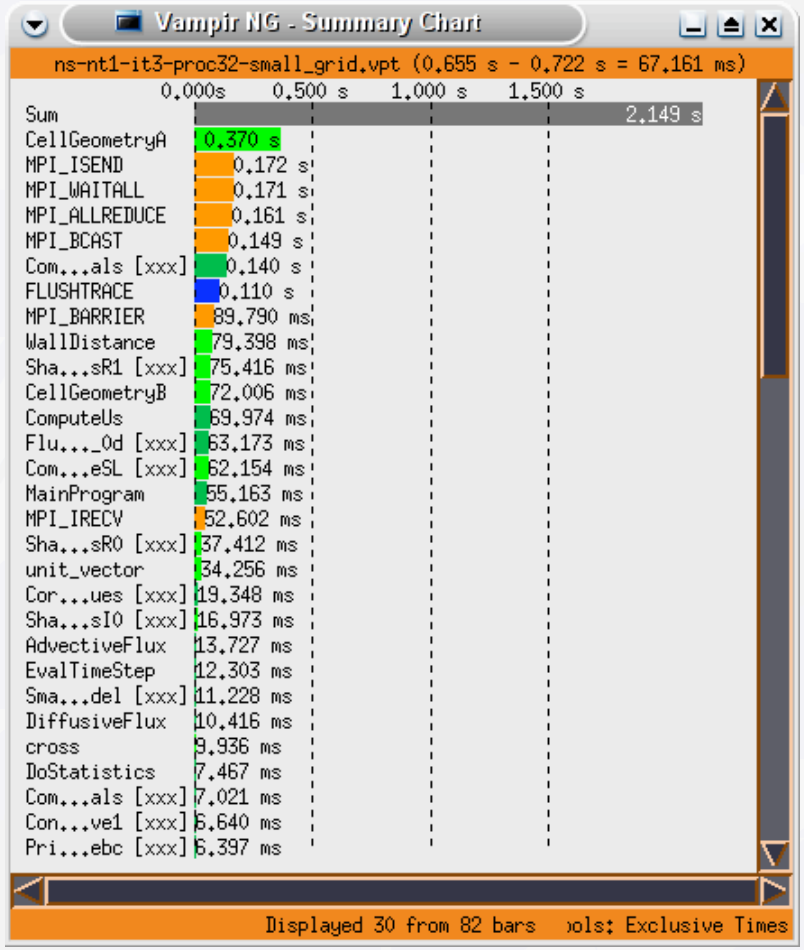


Vampir Displays - Process Timeline

- timeline for single processes
- unfolded call stack in vertical dimension
- shows:
  - functions/function groups
  - messages, collective communication
  - I/O activities
  - performance counter values
- zoomable in time
- context display

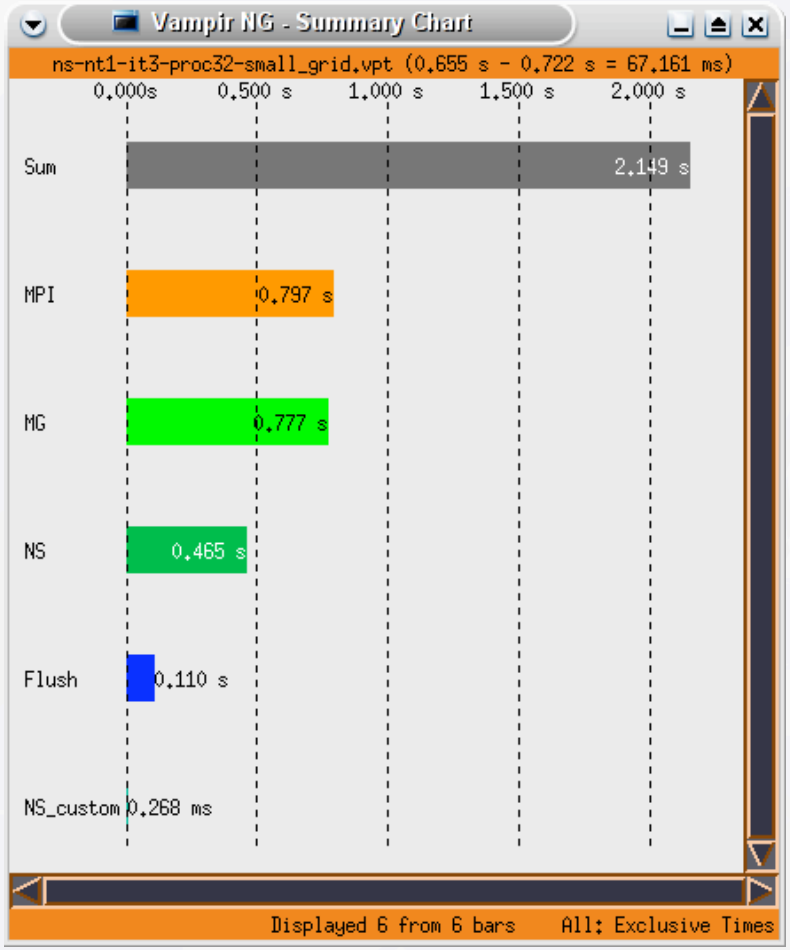
Vampir Displays - Summary Chart

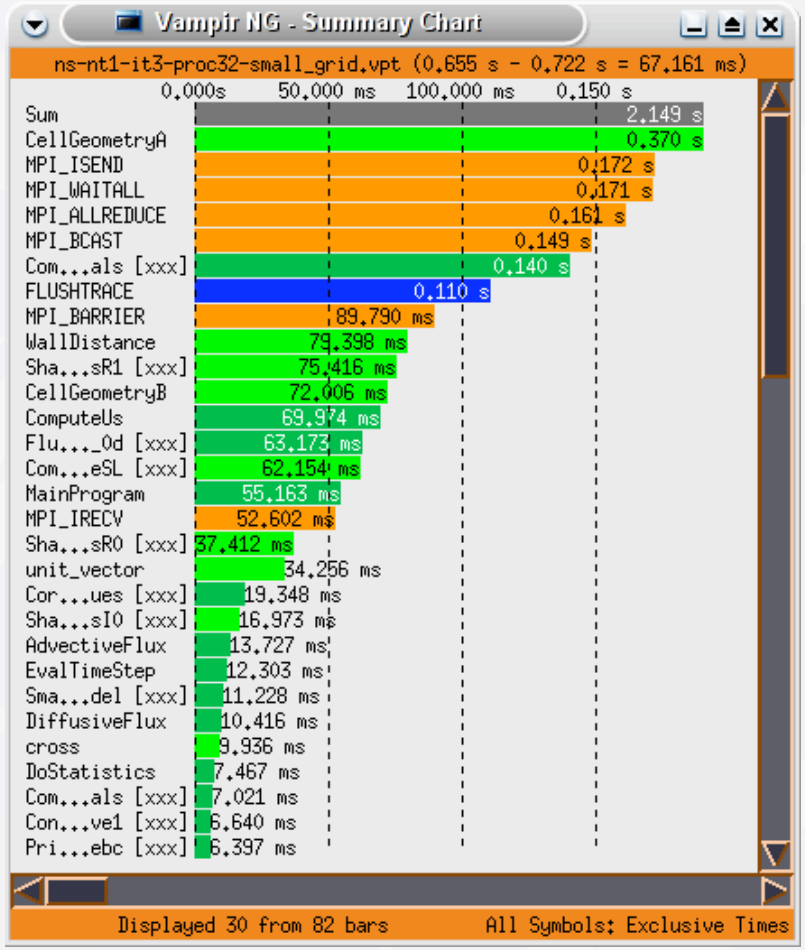




Grouped / Comprehensive Function Statistics

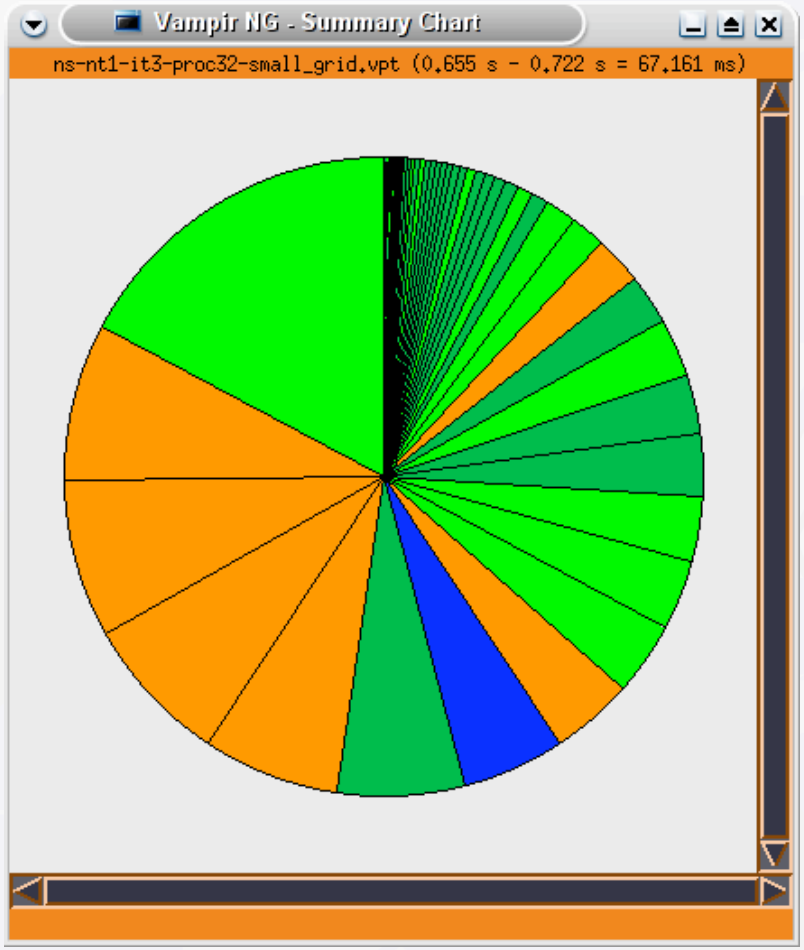
Vampir Displays - Summary Chart

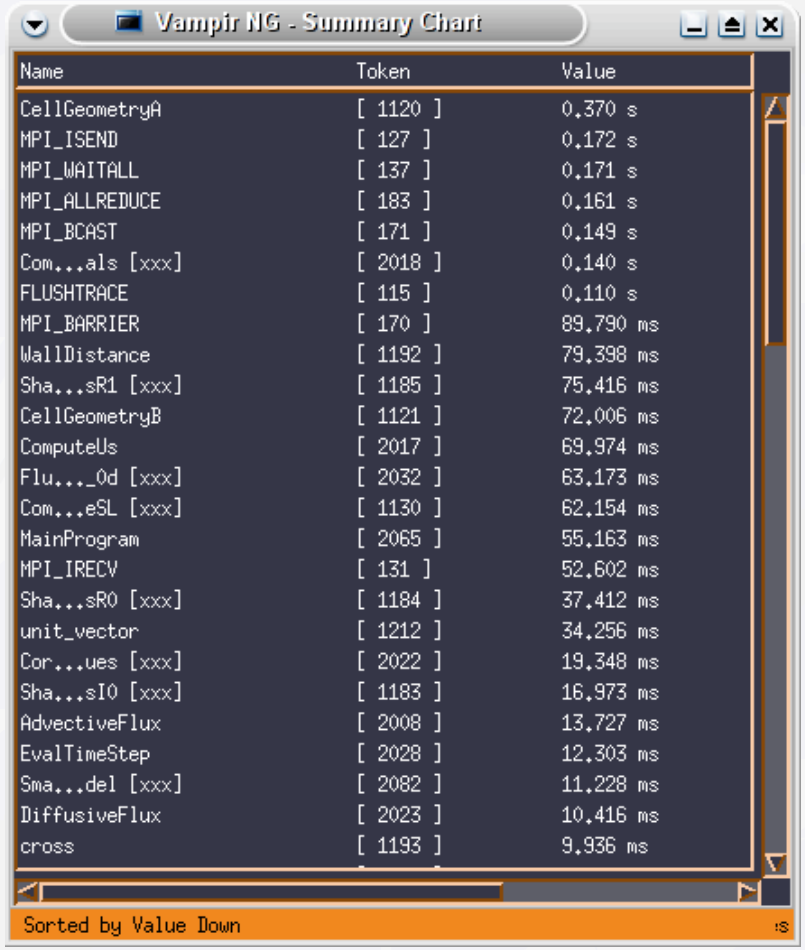




Grouped / Comprehensive Function Statistics

Vampir Displays - Summary Chart





Alternative Representation

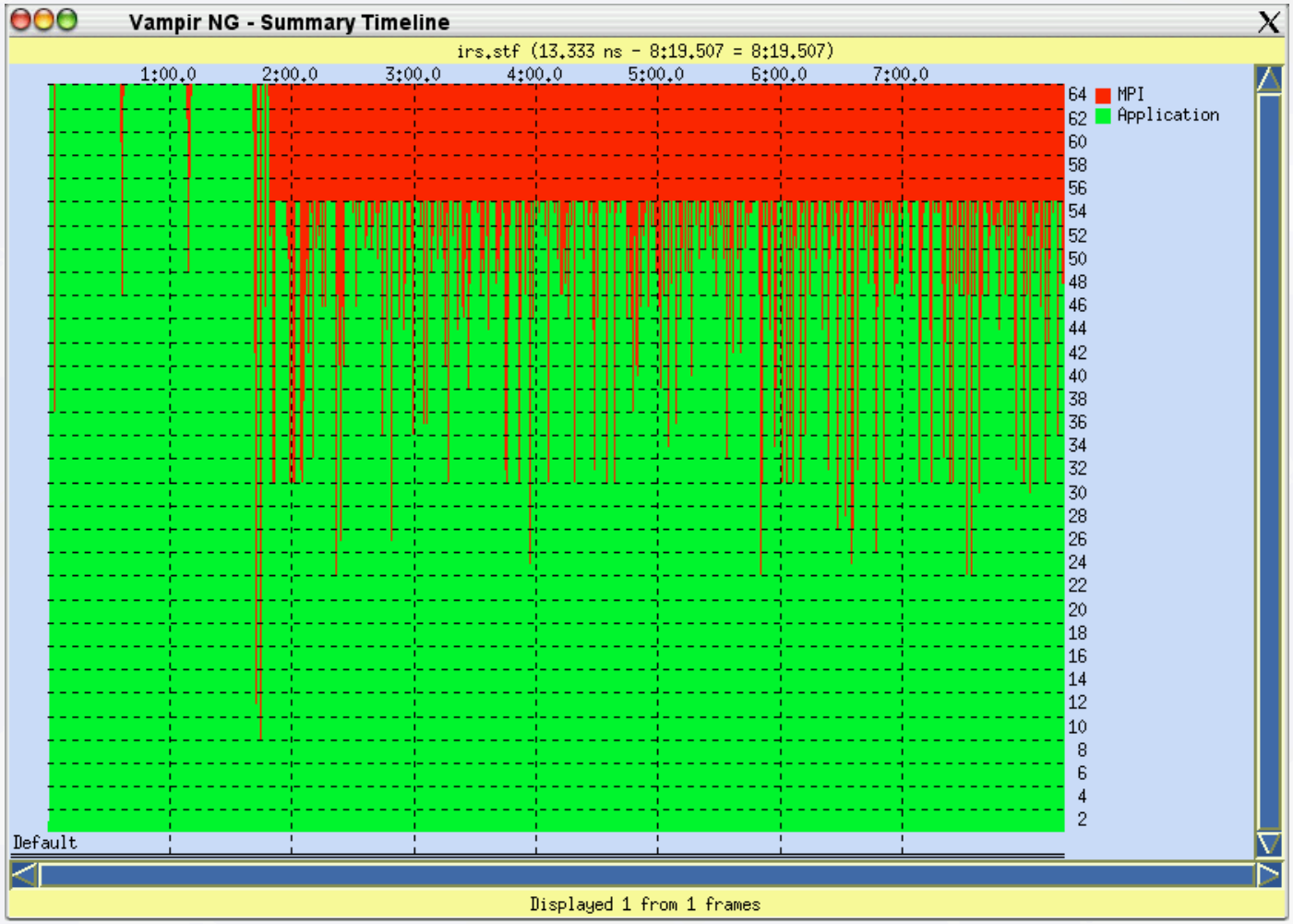
Vampir Displays - Summary Chart

- statistical overview over functions

  resp. groups of functions
- as bar chart, pie chart or table
- global (all processes) or local (single process)
- exclusive/inclusive time, occurrences
- absolute or logarithmic scale
- absolute values or percentages
- zoomable
- respect the current time interval

  (according to timeline)

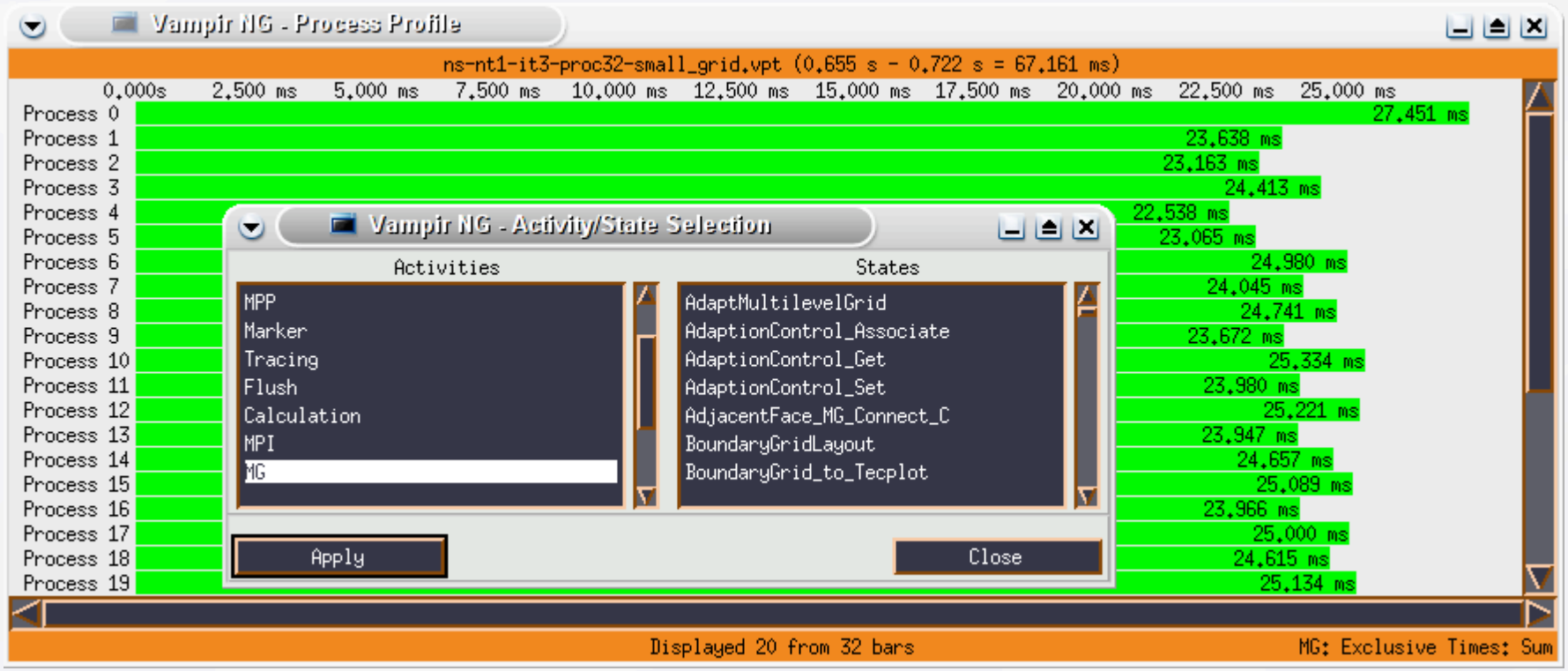
Vampir Displays: Summary Timeline



Vampir Displays - Summary Timeline

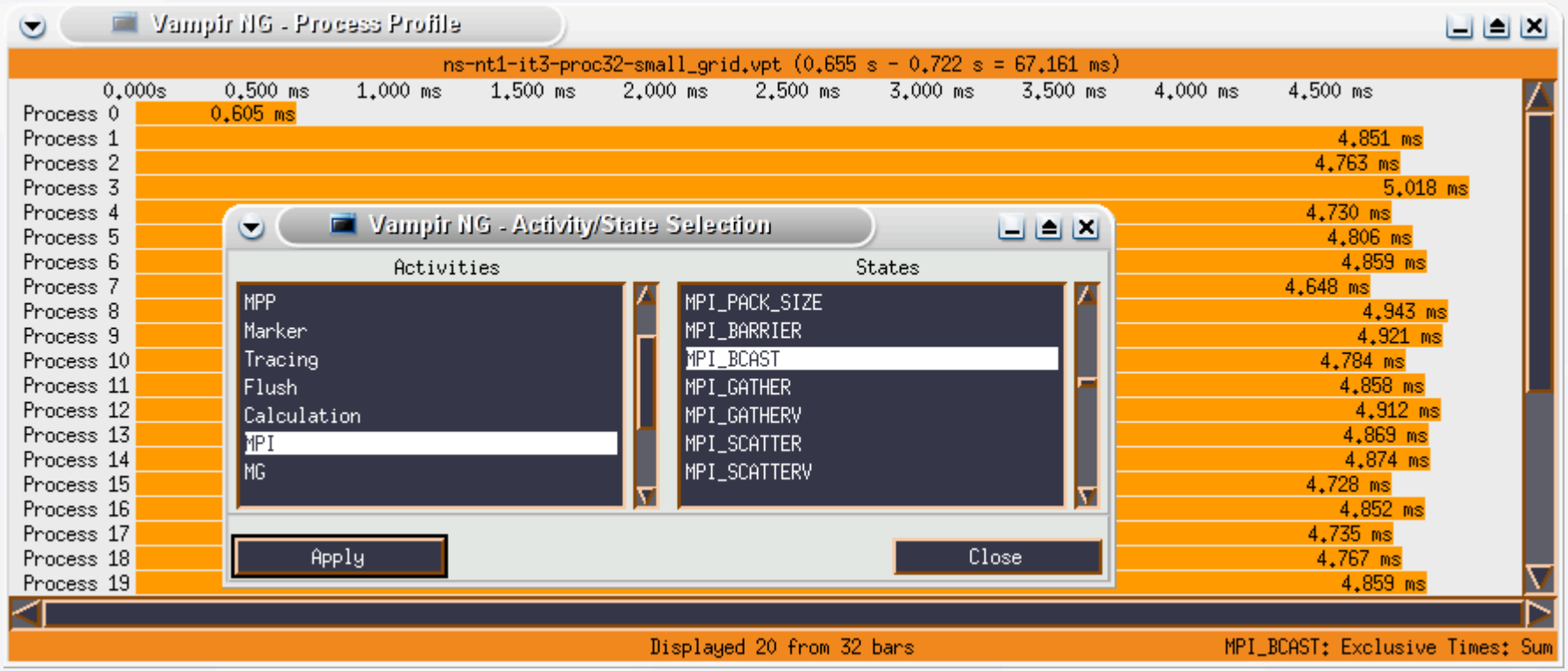
- Summary Chart over time
- same functionality
- zoomable

Vampir Displays - Process Profile



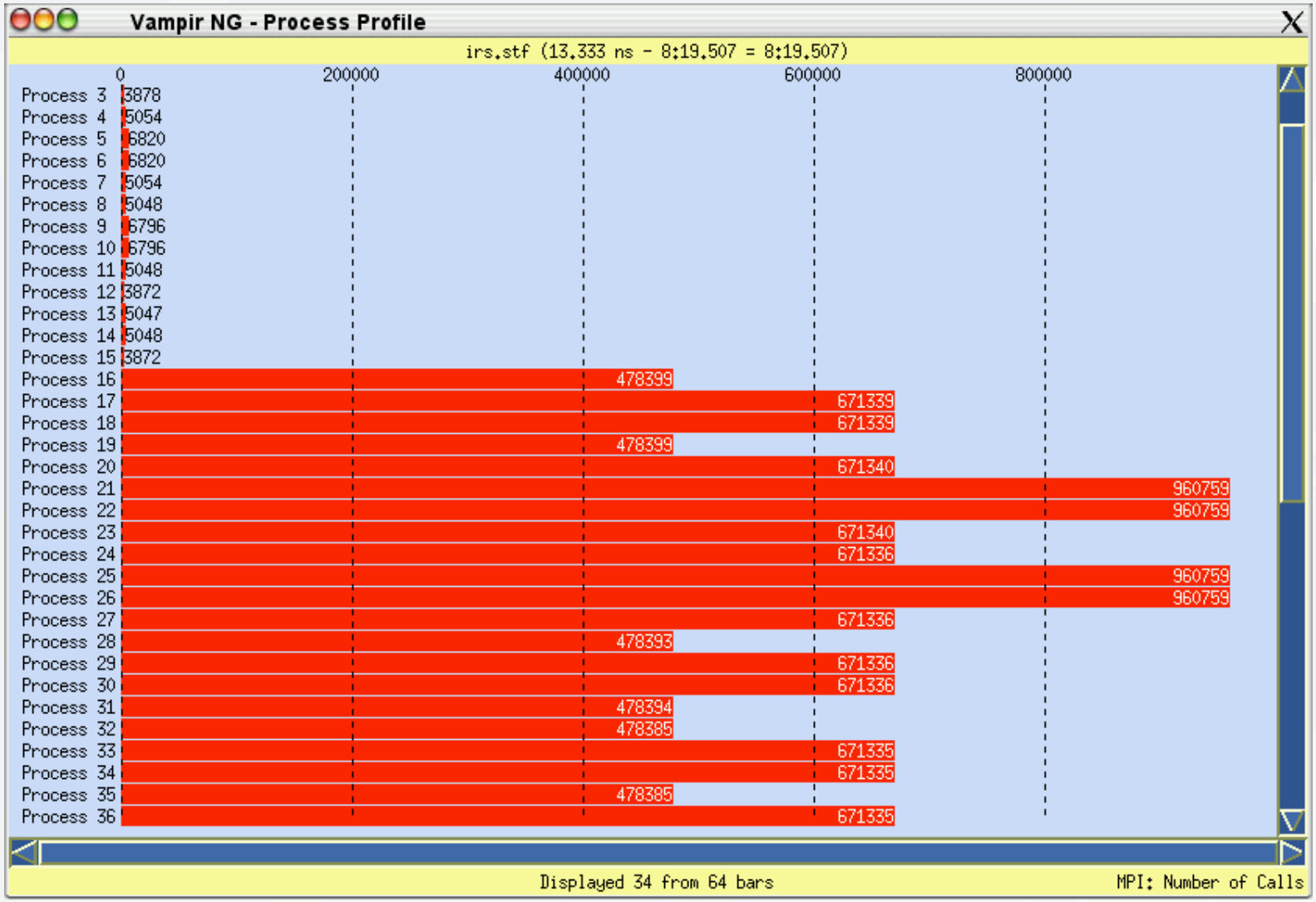
Process Profile

Vampir Displays - Counter Timeline



Process Profile

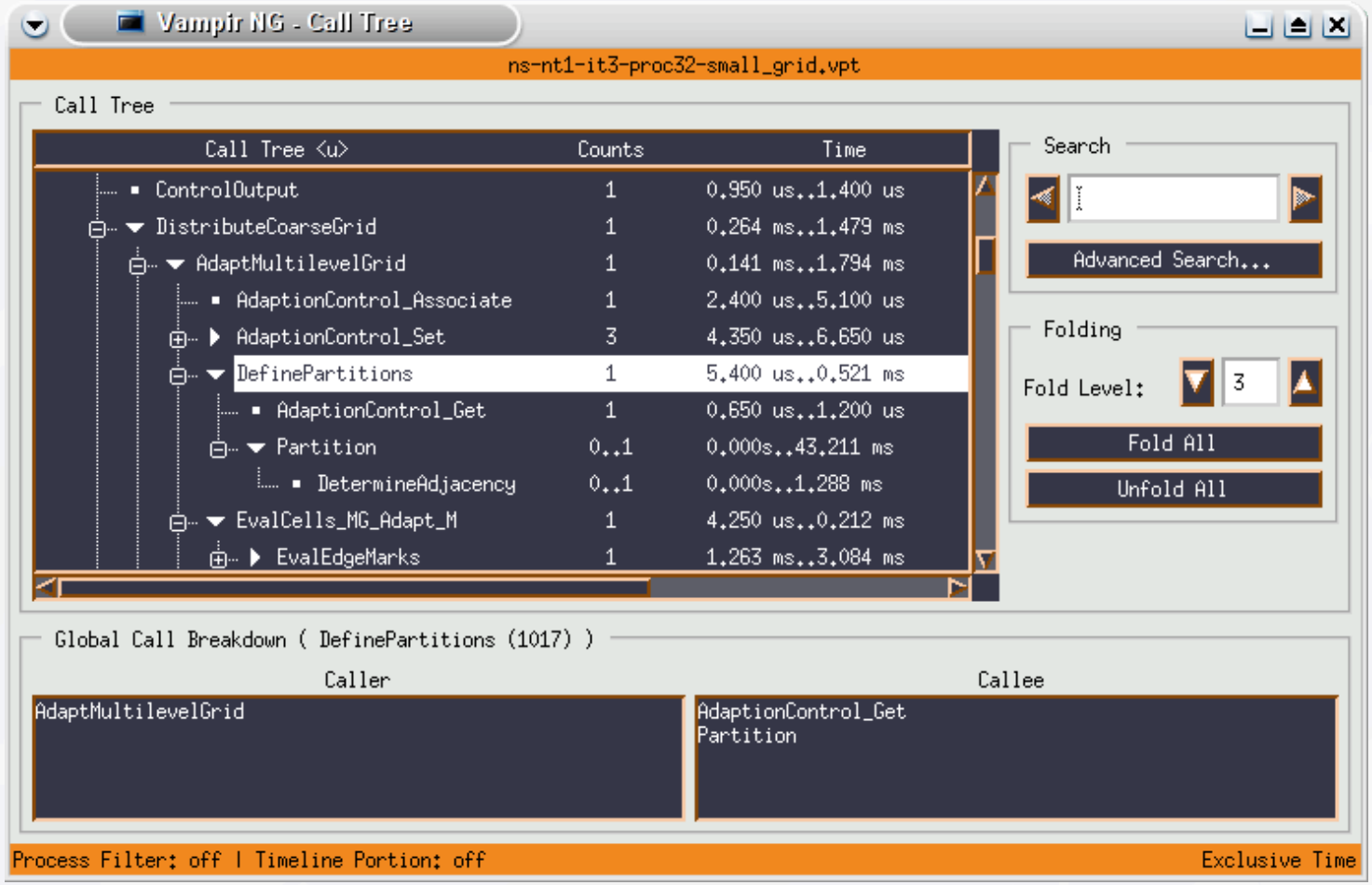
Vampir Display – Process Profile (MPI)



Vampir Displays - Process Profile

- profile for function/group of functions per process
- similar to results of regular profiling
- but now available for arbitrary time intervals

Vampir Displays - Call Tree

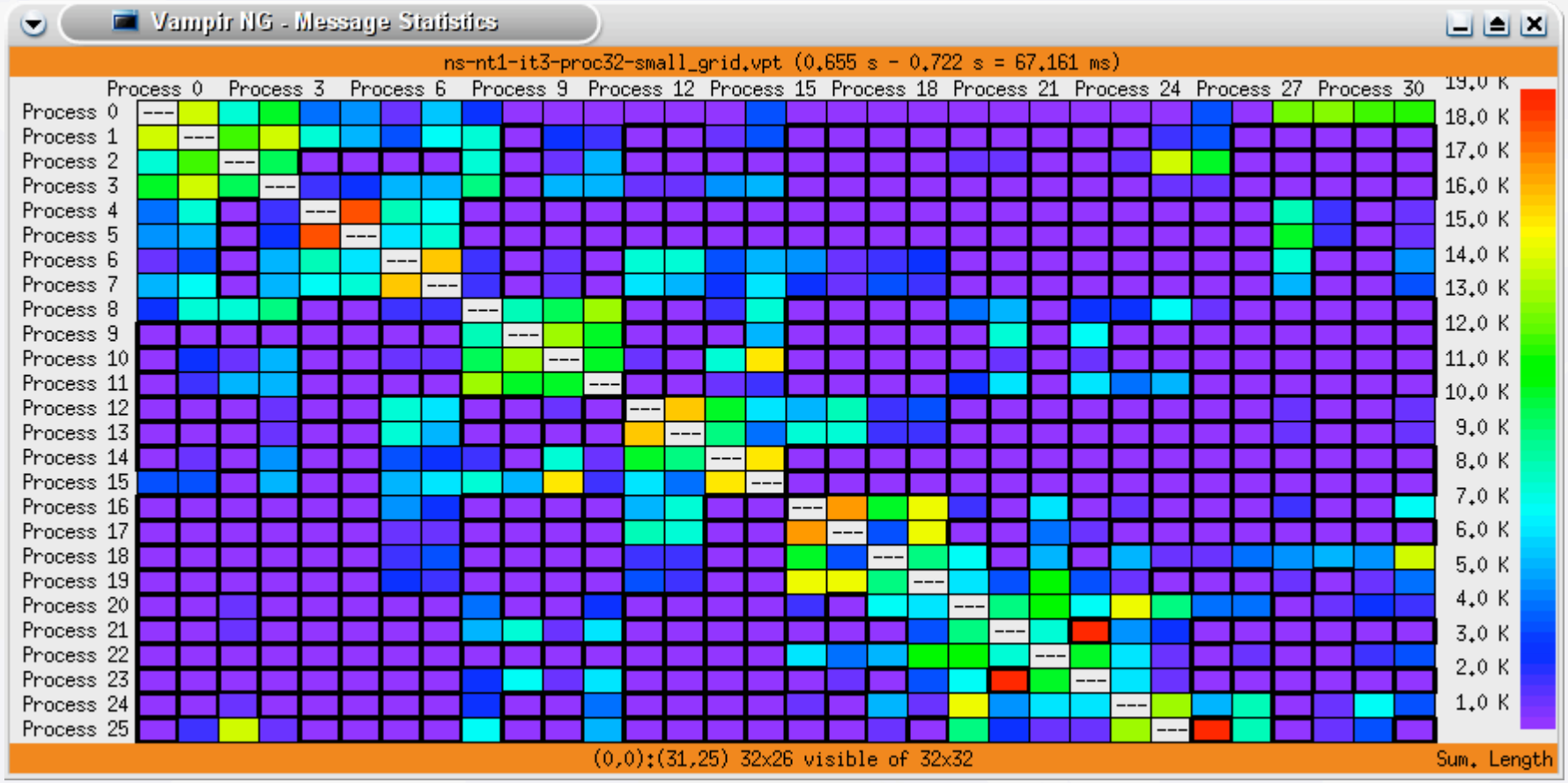


Call Tree

Vampir Displays - Call Tree

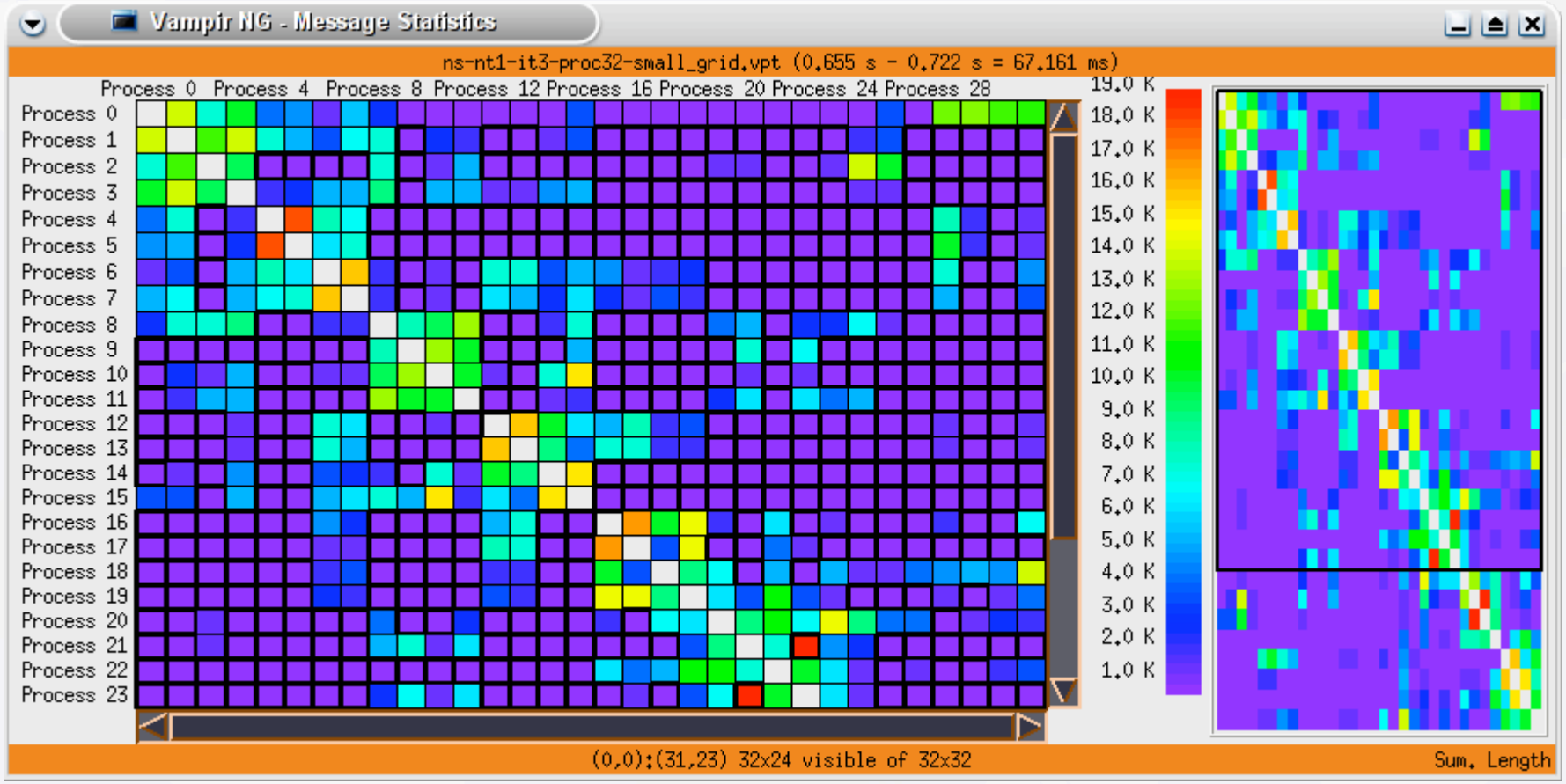
- shows function call hierarchy
- provide callers & callees
- call count
- min/max run time
- fold/unfold
- search

Vampir Displays - Message Statistics



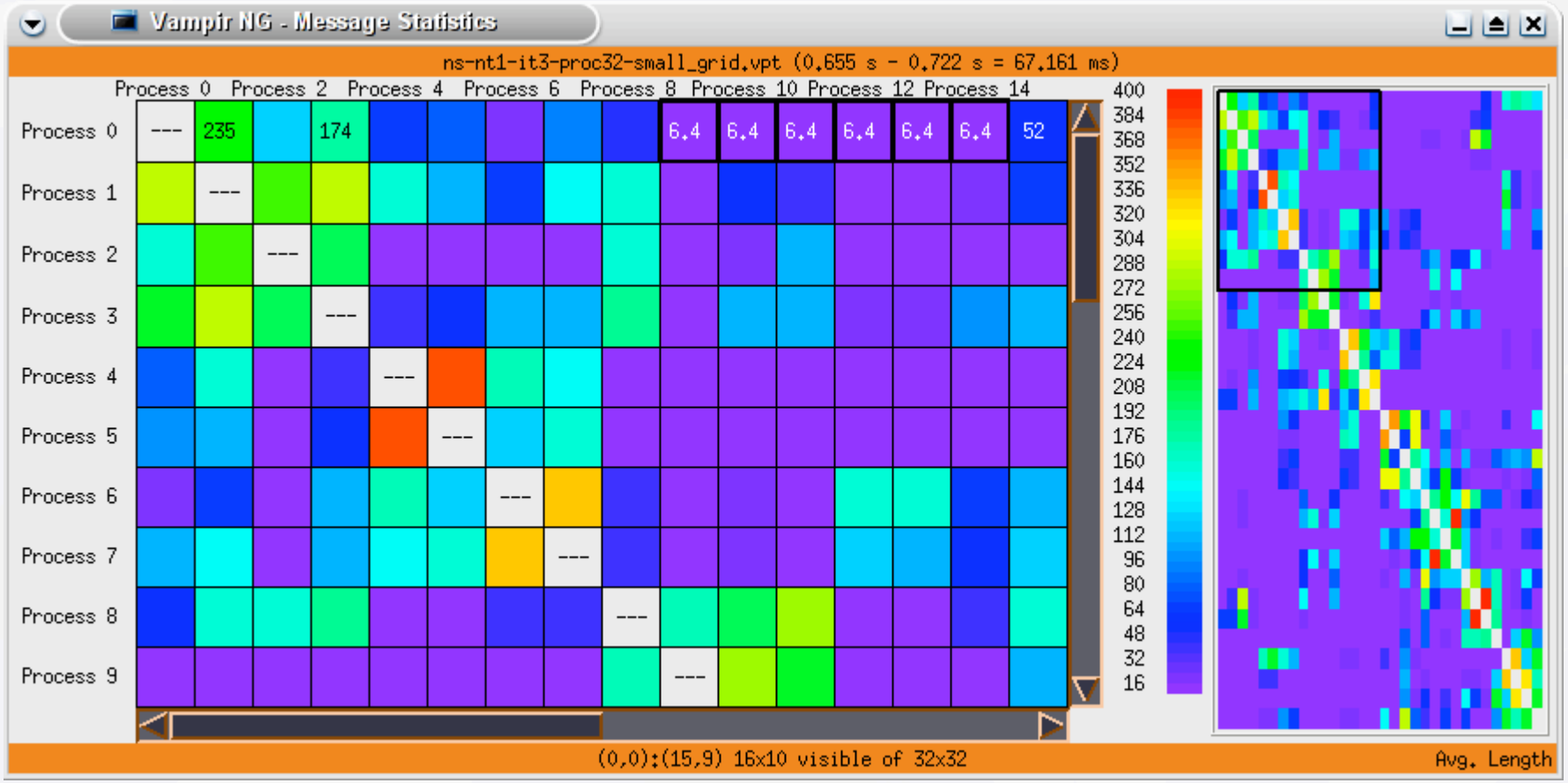
Message Statistics

Vampir Displays - Message Statistics



Message Statistics with Thumbnails

Vampir Displays - Message Statistics

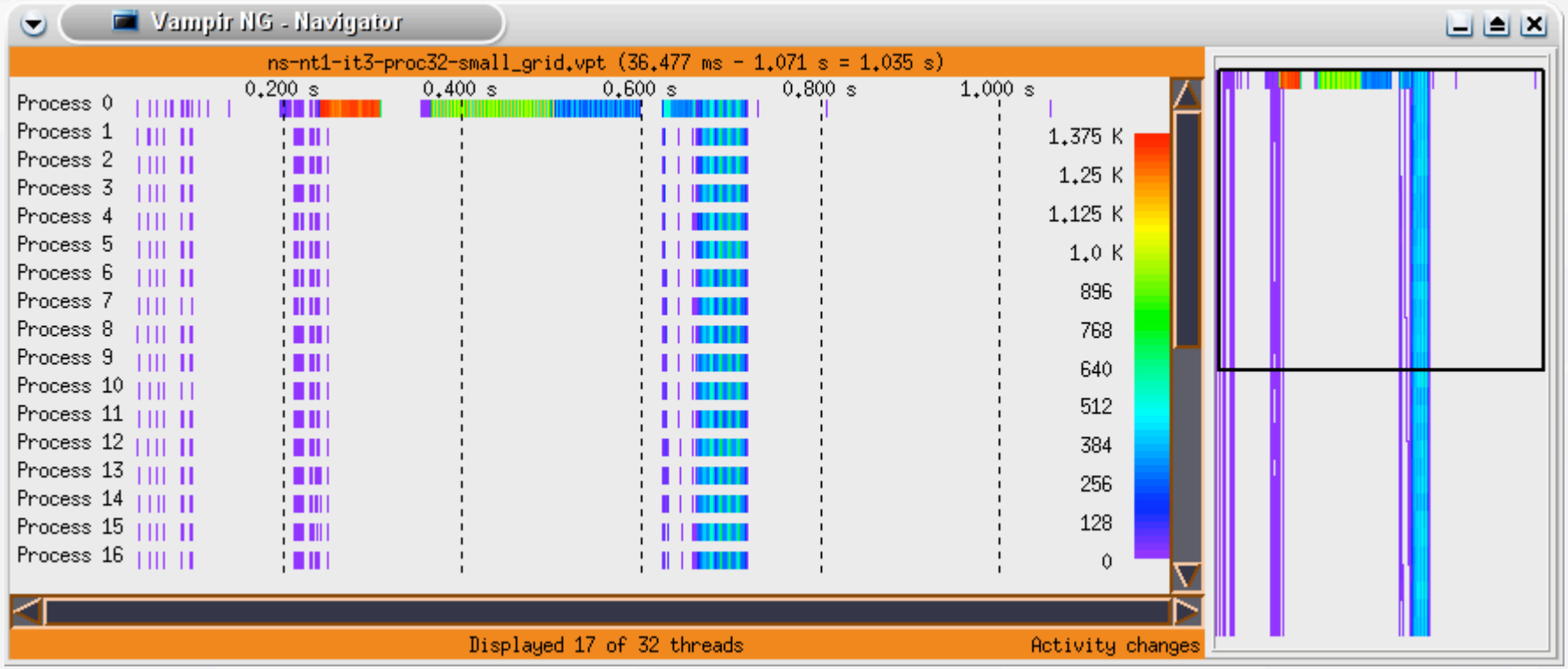


Zoomed Message Statistics

Vampir Displays - Message Statistics

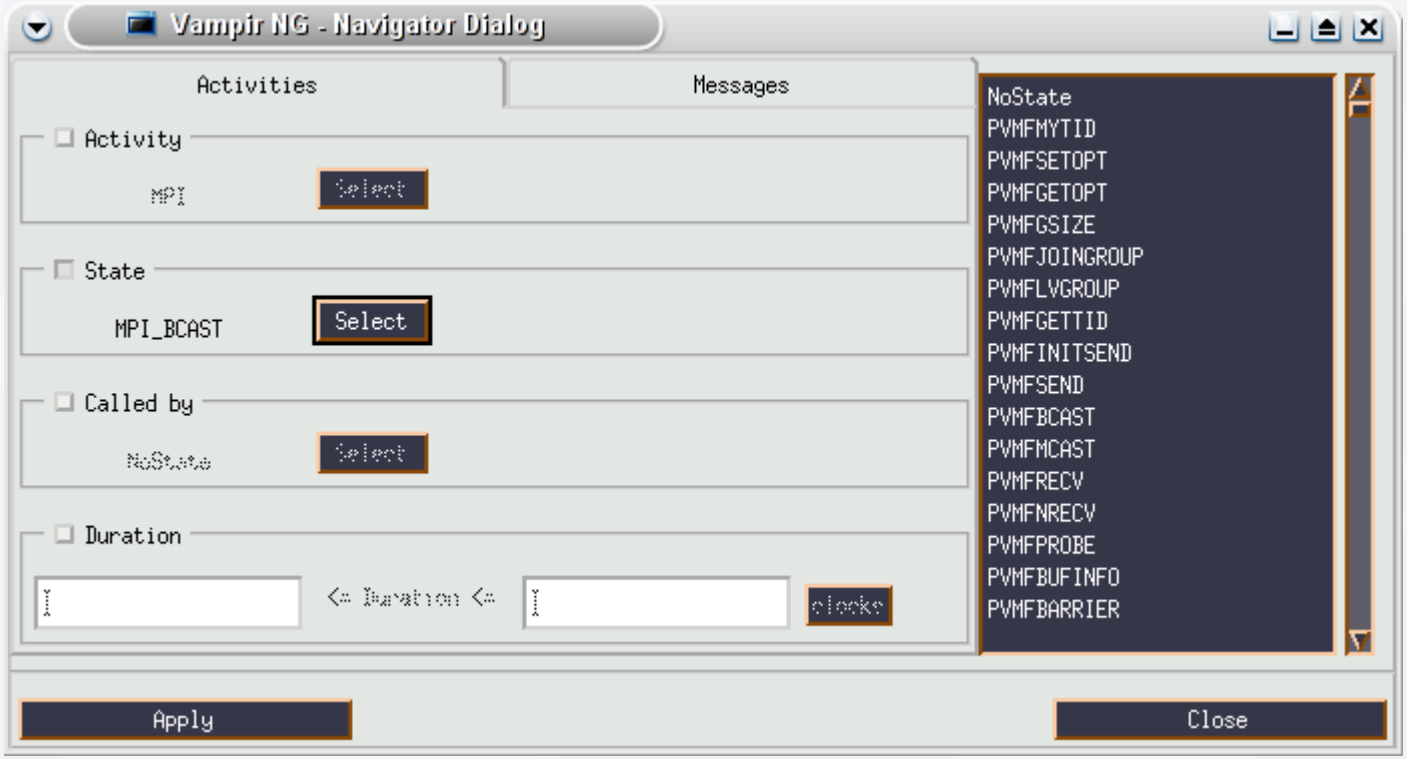
- sender-receiver matrix
- zoomable, for > 1000 processes
- show message properties:
  - Length, rate
  - duration
  - count
- optionally as:
  - min, max, avg, sum
- (message length histogram)

Vampir Displays - Navigator



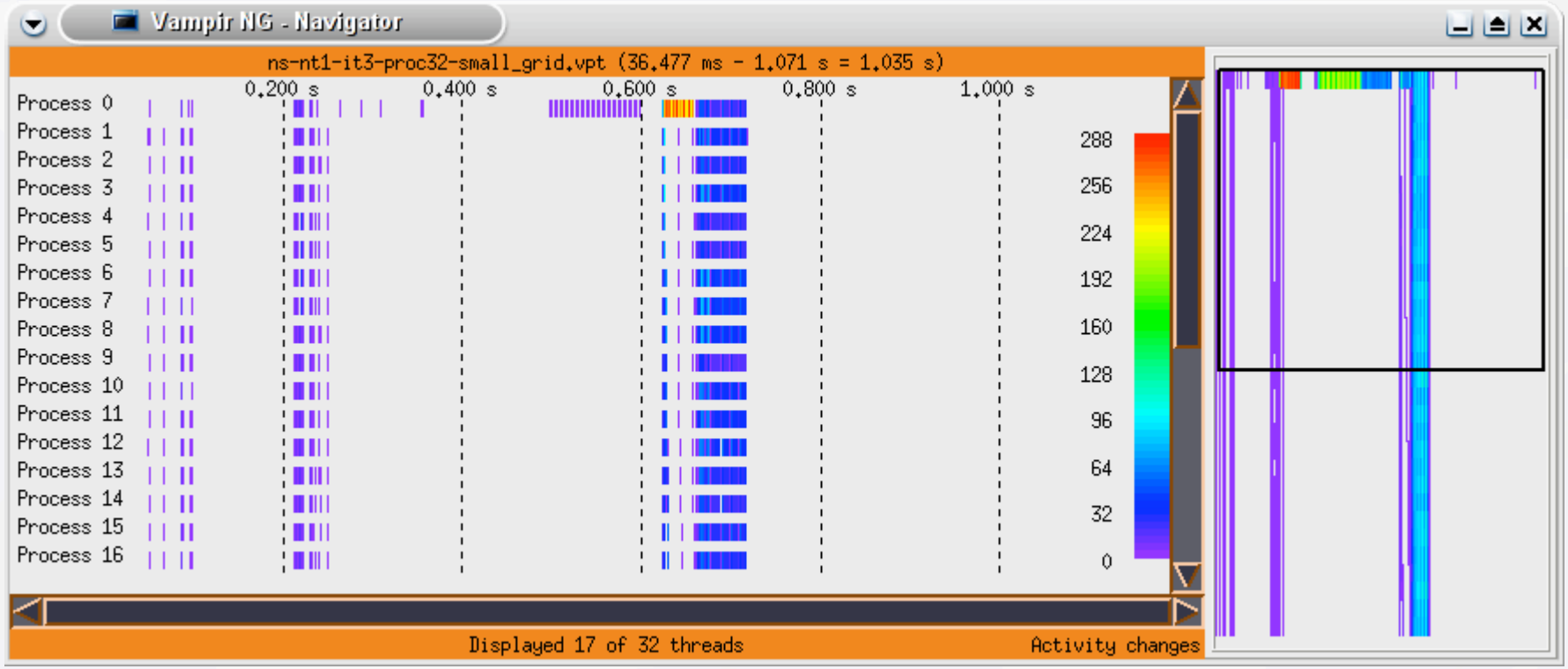
Event Navigator

Vampir Displays - Navigator



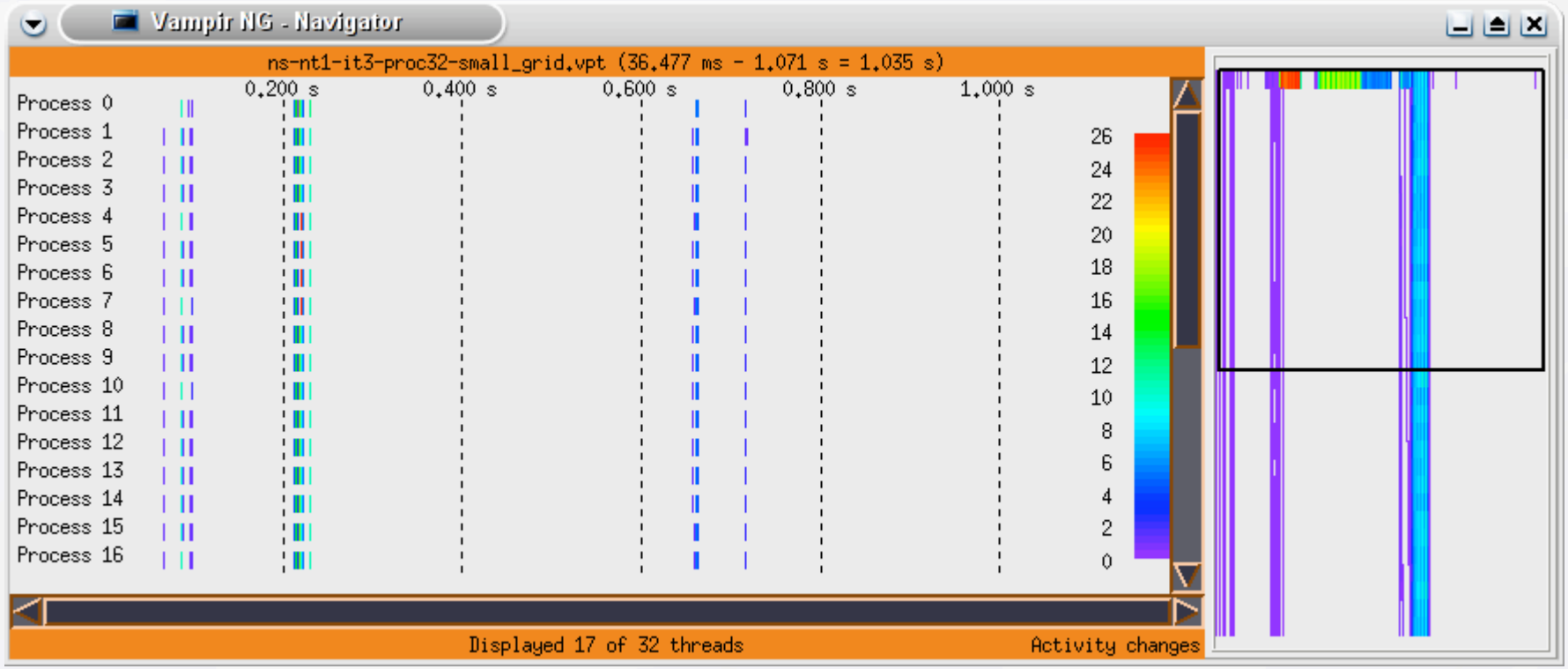
Navigator's Function Selection Dialog

Vampir Displays - Navigator



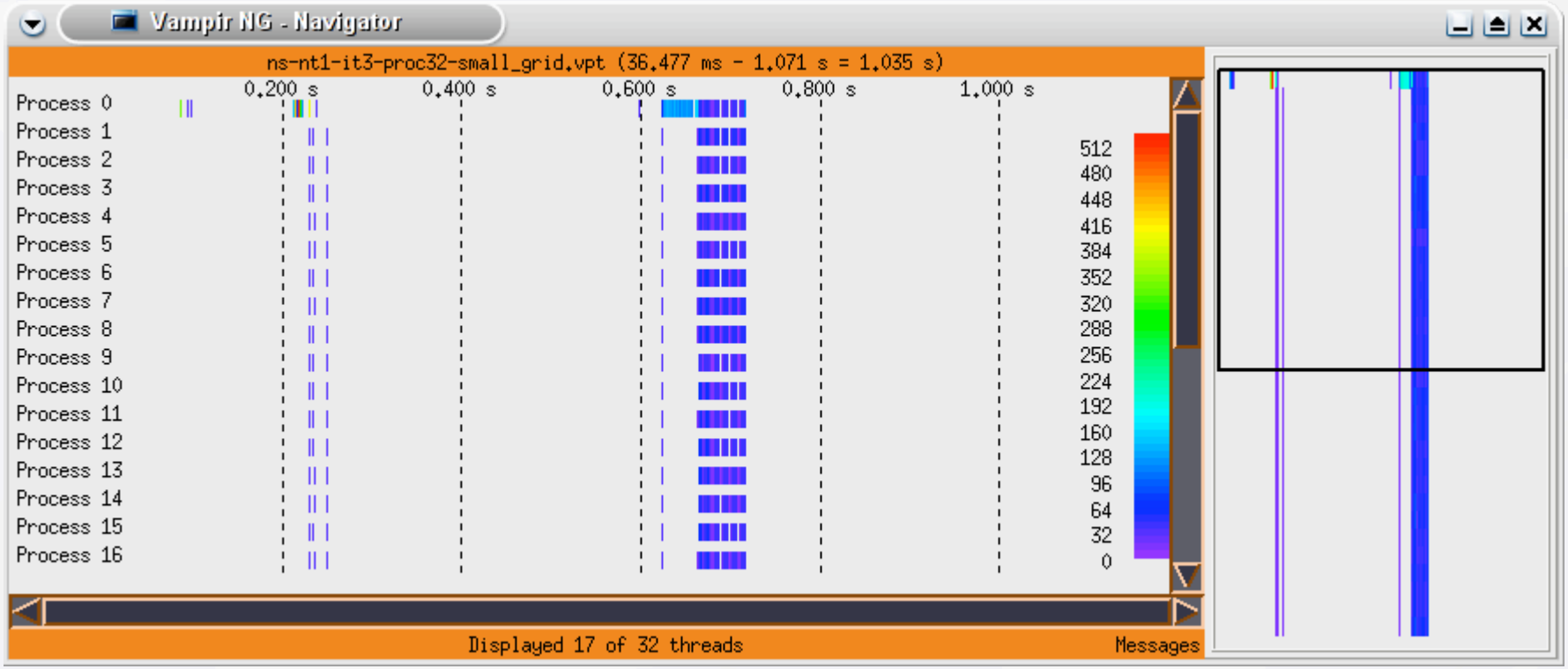
Event Navigator for Function Group (MPI)

Vampir Displays - Navigator



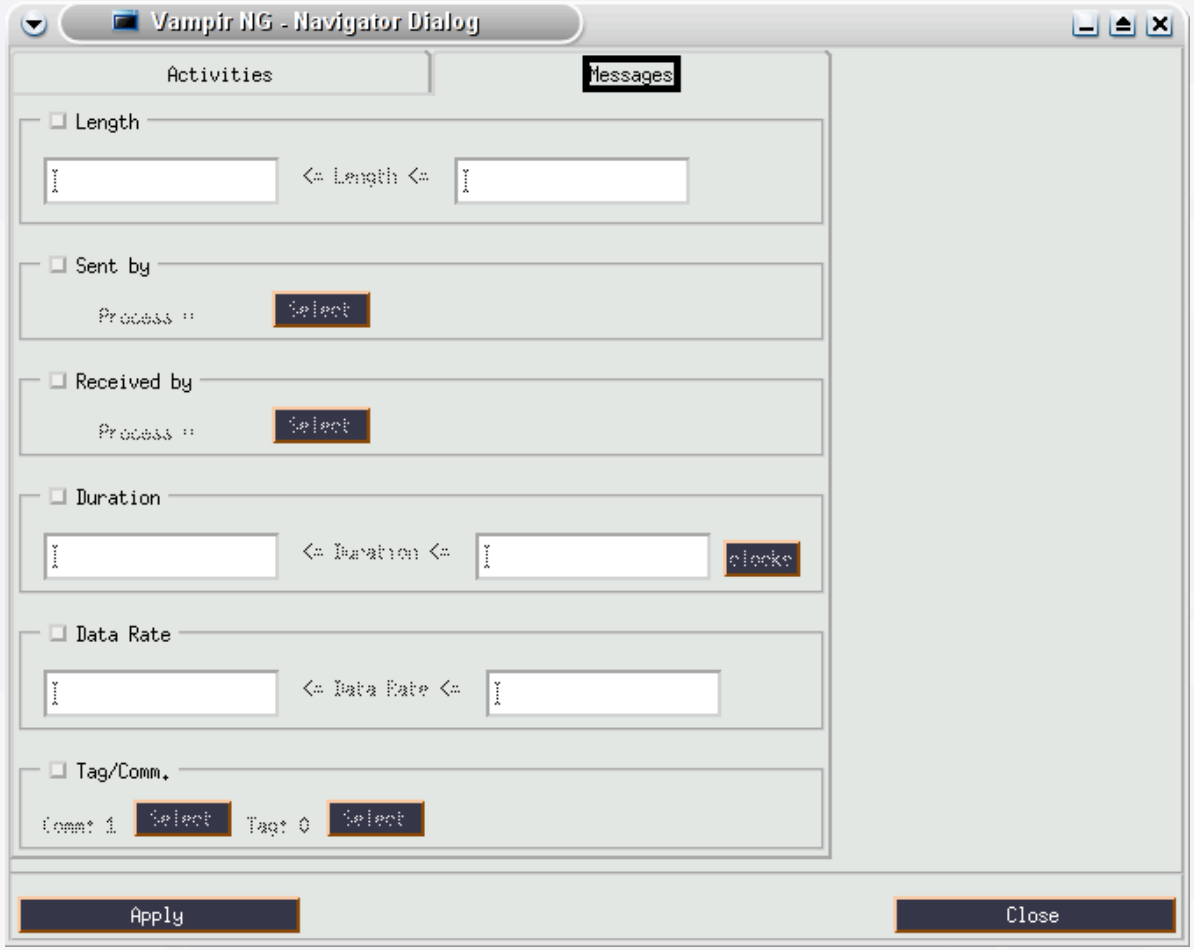
Event Navigator for Function (MPI_Bcast)

Vampir Displays - Navigator



Event Navigator for Messages

Vampir Displays - Navigator

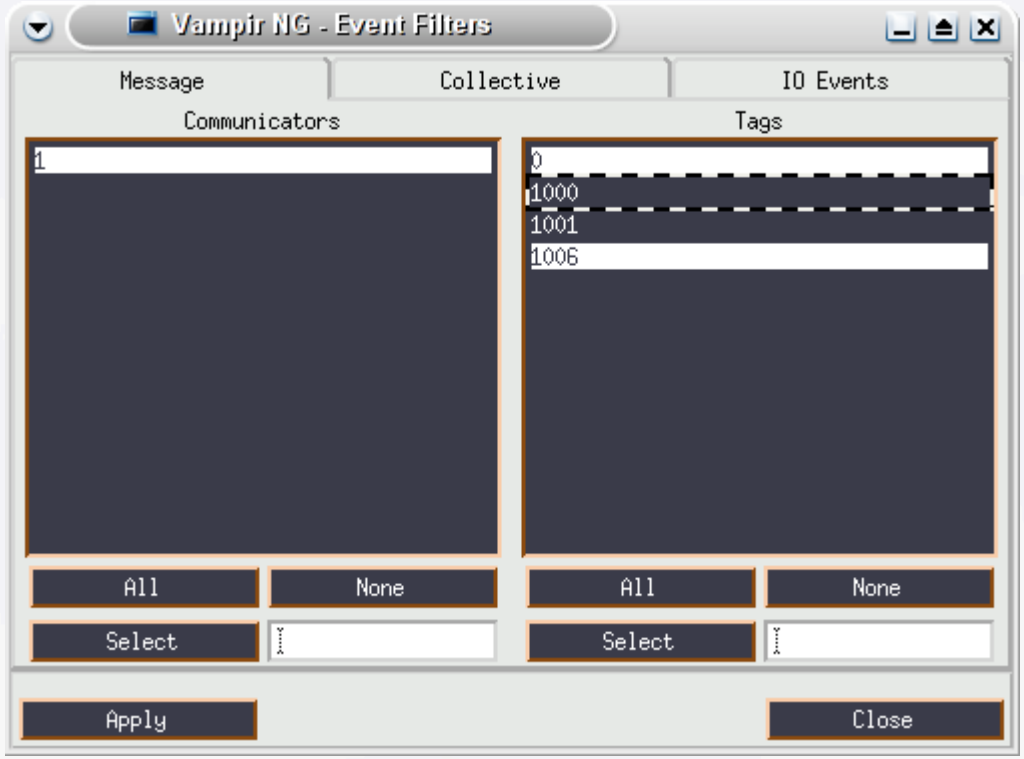


Navigator's Message Selection Dialog

Vampir Displays - Navigator

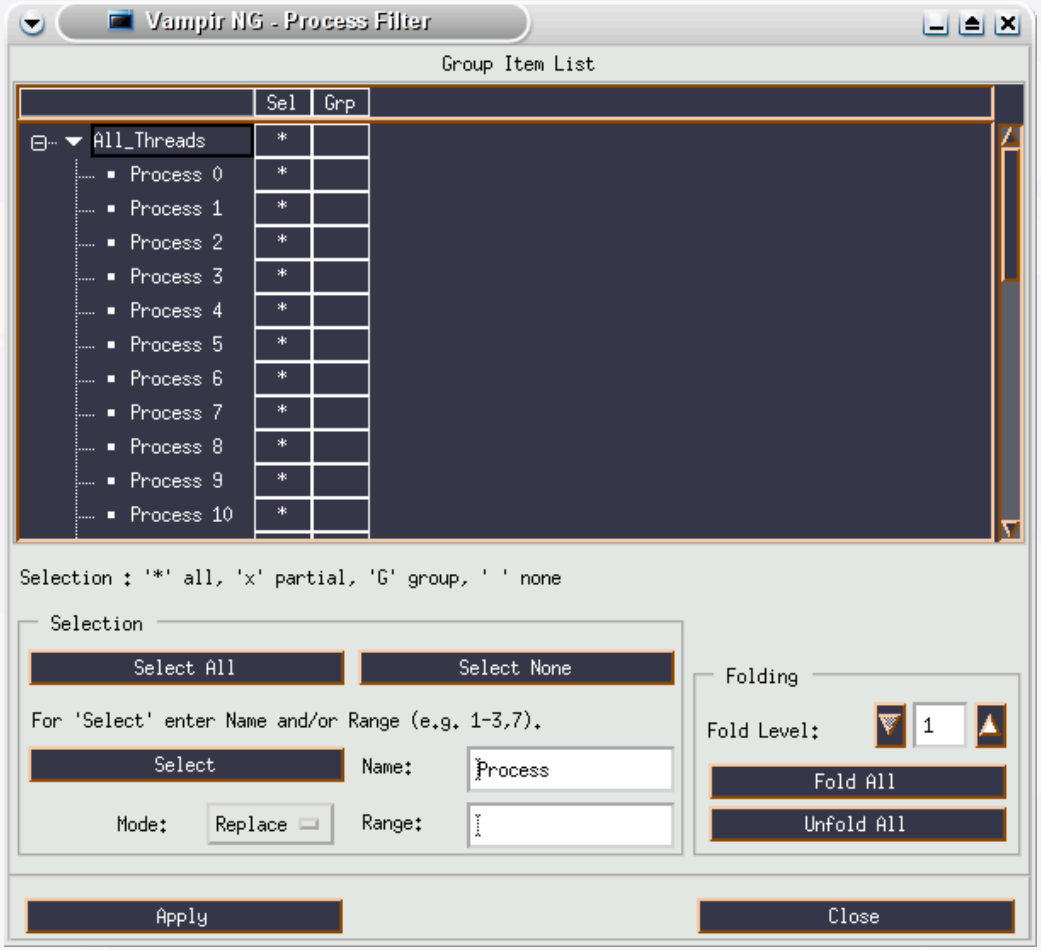
- find occurrences of certain kinds of events
- show frequency by color scale
- filter functions by
  - functions, function groups, caller
  - duration interval
- filter messages by
  - sender, receiver processes
  - message length, duration or speed
  - MPI communicator or tag
- zoomable in time

Vampir Displays - Global Filters



Event Filter Dialog

Vampir Displays - Global Filters

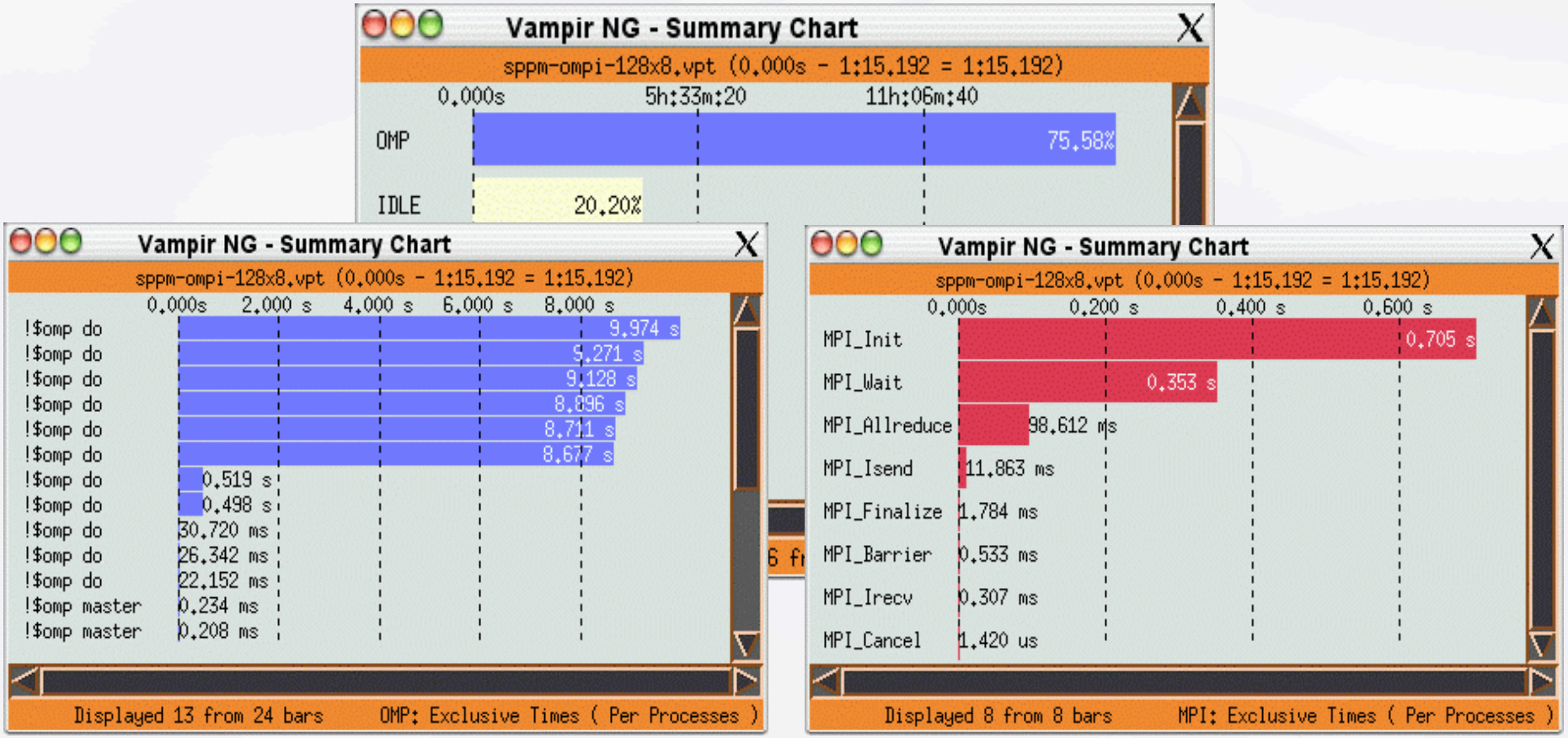


Process Filter Dialog

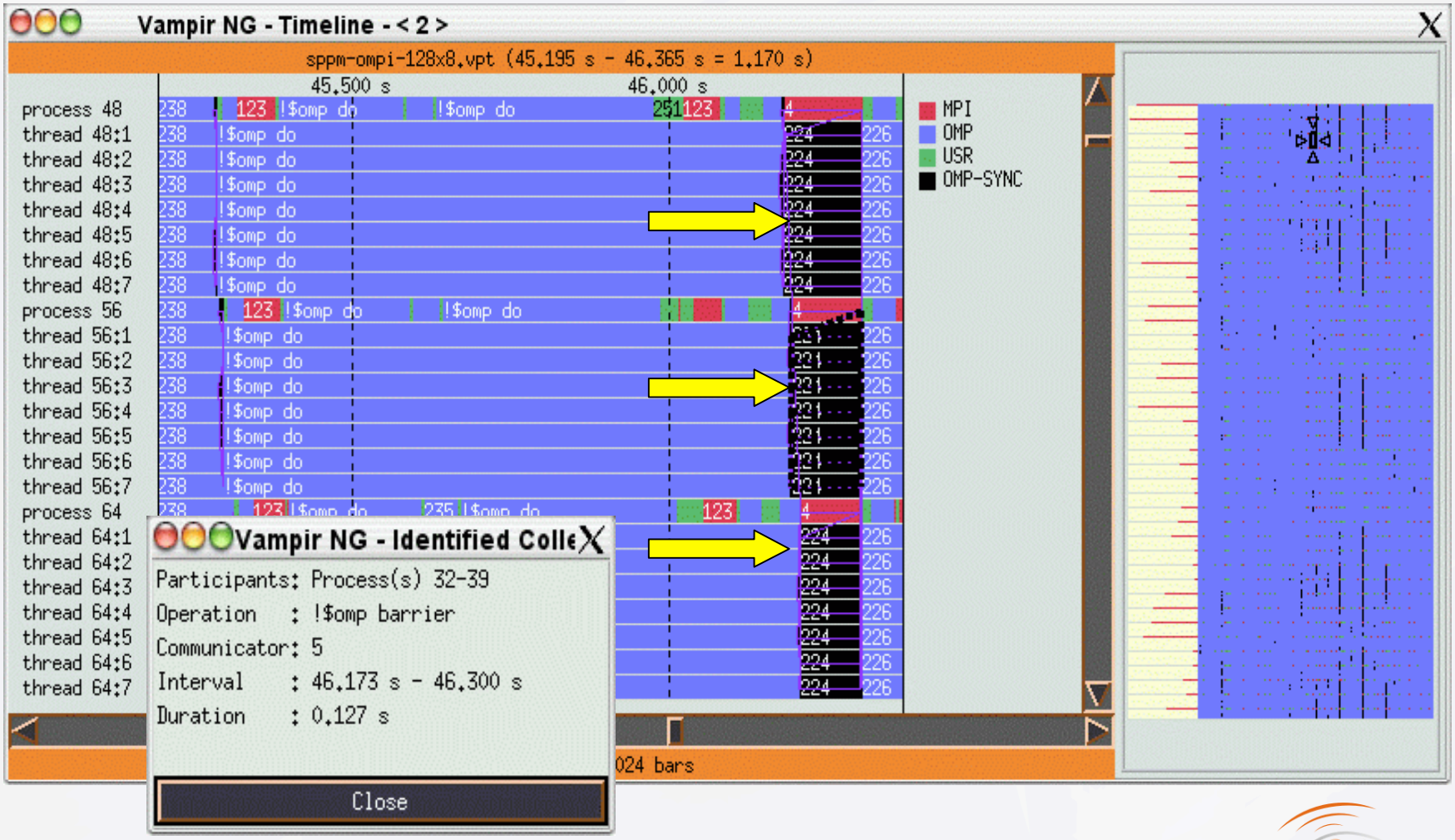
Vampir Displays - Global Filters

- ignore certain items globally
- processes/threads or groups of them
- messages
  - by communicator or by tag
- collective operations
  - by communicator or by type
- I/O events
  - by communicator
  - by file
  - by read/write access

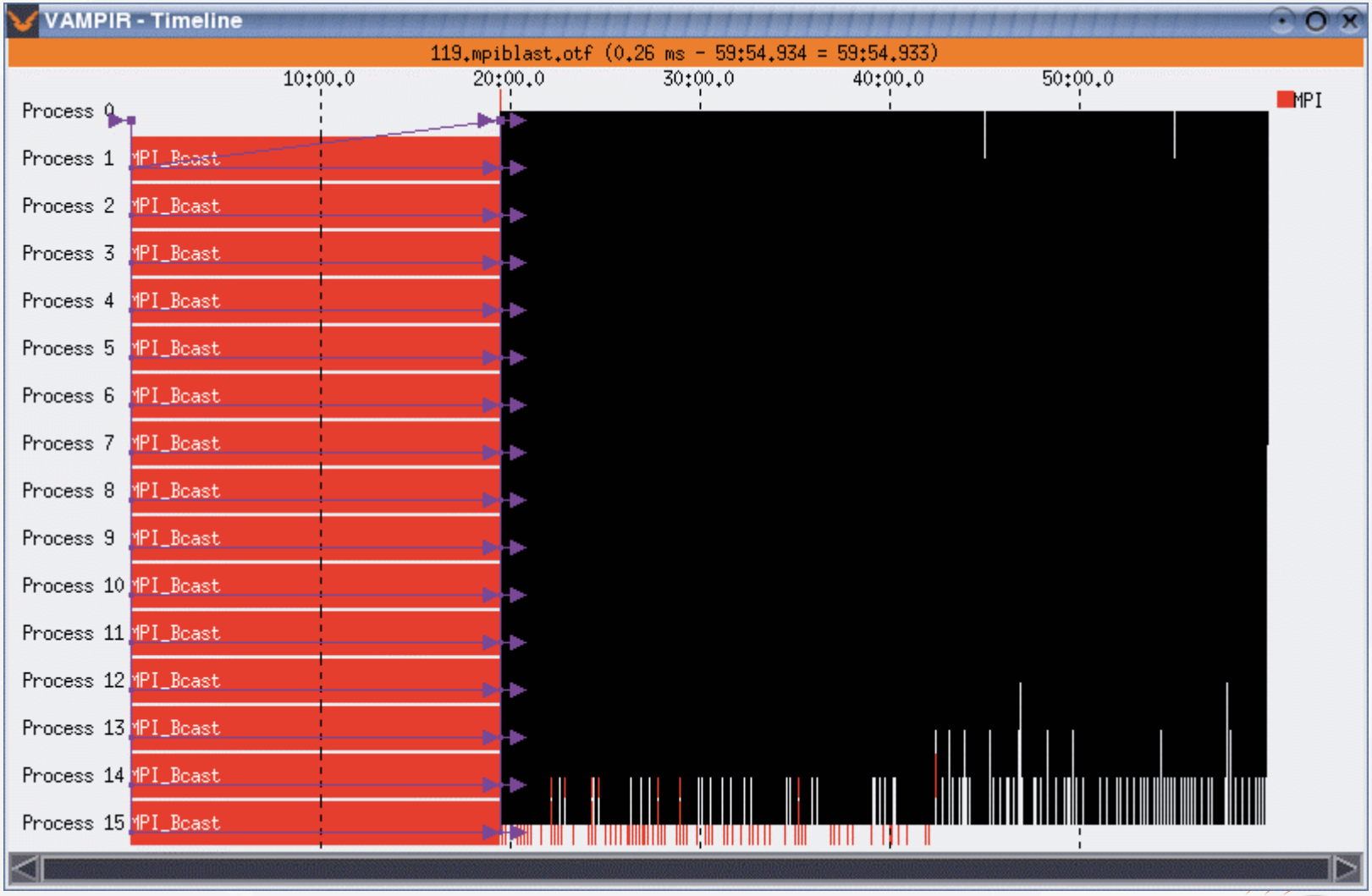
Vampir Displays - OpenMP and MPI Profiles



Vampir Displays - OpenMP Barrier Synchronization



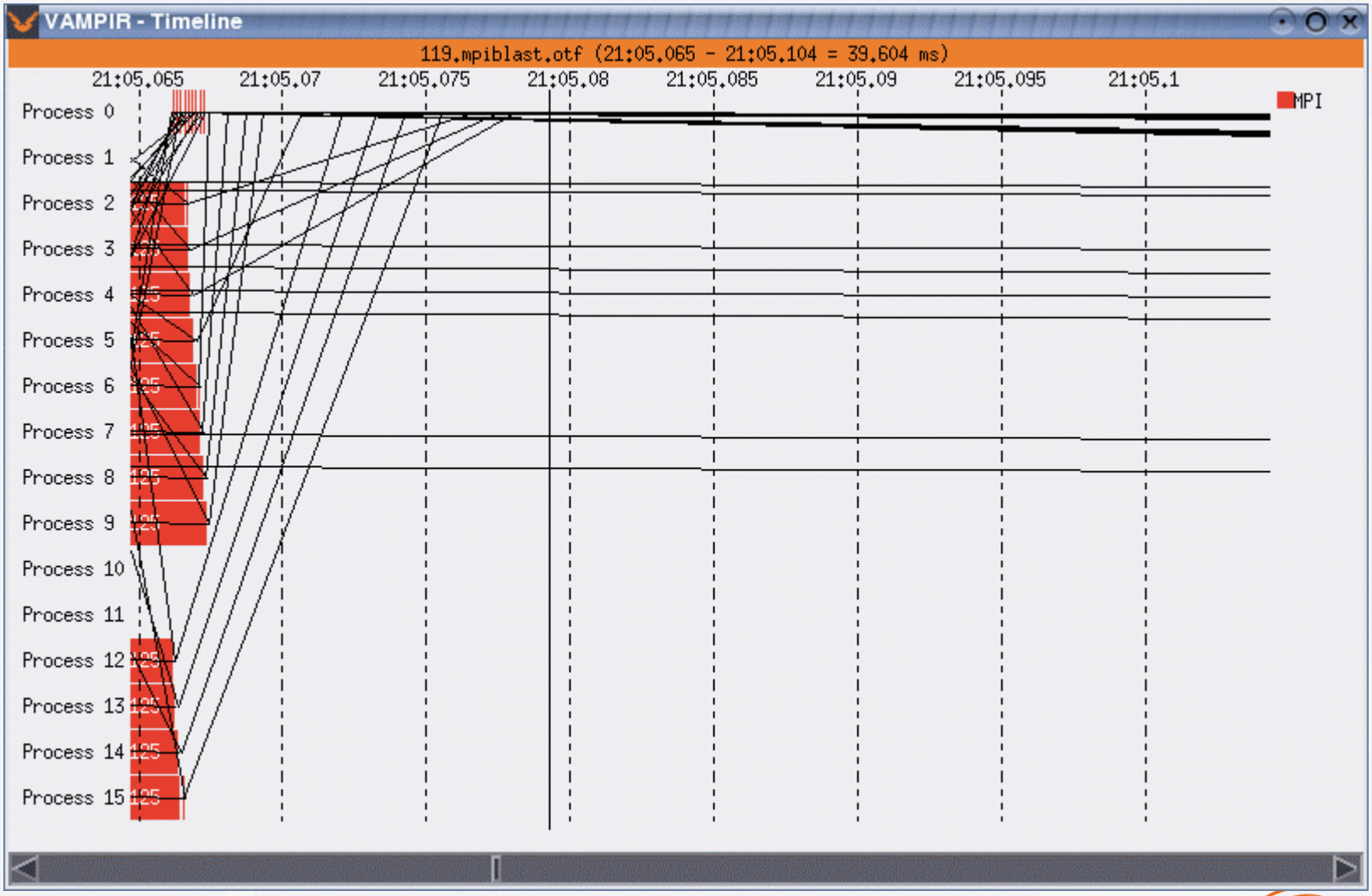
Vampir (old)


VAMPIR - Timeline
119.mpiblast.otf (0.26 ms - 59:54.934 = 59:54.933)
10:00.0
20:00.0
30:00.0
40:00.0
50:00.0
MPI
Process 0
Process 1 MPI_Bcast
Process 2 MPI_Bcast
Process 3 MPI_Bcast
Process 4 MPI_Bcast
Process 5 MPI_Bcast
Process 6 MPI_Bcast
Process 7 MPI_Bcast
Process 8 MPI_Bcast
Process 9 MPI_Bcast
Process 10 MPI_Bcast
Process 11 MPI_Bcast
Process 12 MPI_Bcast
Process 13 MPI_Bcast
Process 14 MPI_Bcast
Process 15 MPI_Bcast

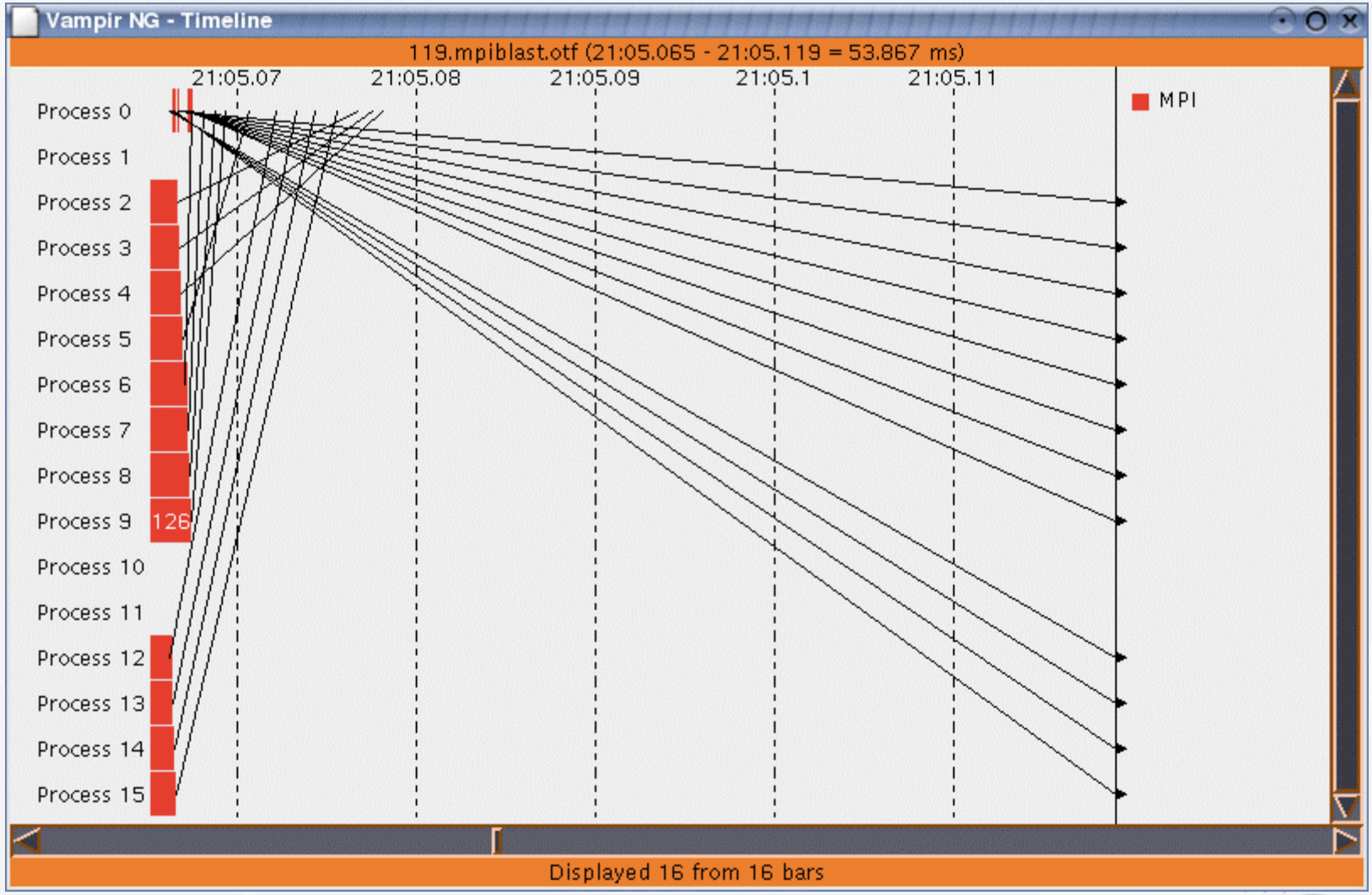
Vampir (new)


Vampir NG - Timeline
119.mpiblast.otf (0.260 ms - 59:54.934 = 59:54.933)
20:00.0
40:00.0
MPI
Process 0
Process 1 MPI_Bcast
Process 2 MPI_Bcast
Process 3 MPI_Bcast
Process 4 MPI_Bcast
Process 5 MPI_Bcast
Process 6 MPI_Bcast
Process 7 MPI_Bcast
Process 8 MPI_Bcast
Process 9 MPI_Bcast
Process 10 MPI_Bcast
Process 11 MPI_Bcast
Process 12 MPI_Bcast
Process 13 MPI_Bcast
Process 14 MPI_Bcast
Process 15 MPI_Bcast
Displayed 16 from 16 bars

Vampir (old)


VAMPIR - Timeline
119.mpiblast.otf (21:05.065 - 21:05.104 = 39.604 ms)
21:05.065
21:05.07
21:05.075
21:05.08
21:05.085
21:05.09
21:05.095
21:05.1
MPI
Process 0
Process 1
Process 2
Process 3
Process 4
Process 5
Process 6
Process 7
Process 8
Process 9
Process 10
Process 11
Process 12
Process 13
Process 14
Process 15

Vampir (new)


Vampir NG - Timeline
119.mpiblast.otf (21:05.065 - 21:05.119 = 53.867 ms)
21:05.07
21:05.08
21:05.09
21:05.1
21:05.11
MPI
Process 0
Process 1
Process 2
Process 3
Process 4
Process 5
Process 6
Process 7
Process 8
Process 9
126
Process 10
Process 11
Process 12
Process 13
Process 14
Process 15
Displayed 16 from 16 bars

Summary

Vampir/Vampir NG offers

- Comprehensive performance analysis environment
- Demonstrated scalability to 5000 processors
- Perfect solution for large centers with remote users and a variety of platforms

Vampir Future Developments

- I/O Analysis
- One sided communication

Vampir Literature

- H. Brunst, H.-Ch. Hoppe, W. E. Nagel, and M. Winkler: **Performance Optimization for Large Scale Computing: The Scalable VAMPIR Approach.** Proc. of ICCS 2001, San Francisco, CA, USA, May 28-30 2001,Springer LNCS 2074, pp 751-760, 2001
- H. Brunst, D. Kranzlmüller, W. E. Nagel: **Tools for Scalable Parallel Program Analysis - Vampir NG and DeWiz.** Distributed and Parallel Systems, Cluster and Grid Computing, Springer, Kluwer Intl. Series in Engineering and Computer Science, Vol 777, Budapest, Hungary, 2004
- A. Knüpfer, H. Brunst, and W. E. Nagel: **High Performance Trace Visualization.** In Proc. of 13th Euromicro Conference on Parallel, Distributed and Network-based Processing, Lugano, Switzerland, February 9-11 2005
- Vampir User Guide

TECHNISCHE UNIVERSITÄT DRESDEN

Zentrum für Informationsdienste und Hochleistungsrechnen (ZIH)

Performance-Analyse mit Vampir

Am Beispiel eines komplexen Modellsystems

Zellescher Weg 12

Willers-Bau A106

Tel. +49 351 - 463 31945

Matthias Lieber

Matthias.Lieber@tu-dresden.de

ZIH
Zentrum für Informationsdienste und Hochleistungsrechnen

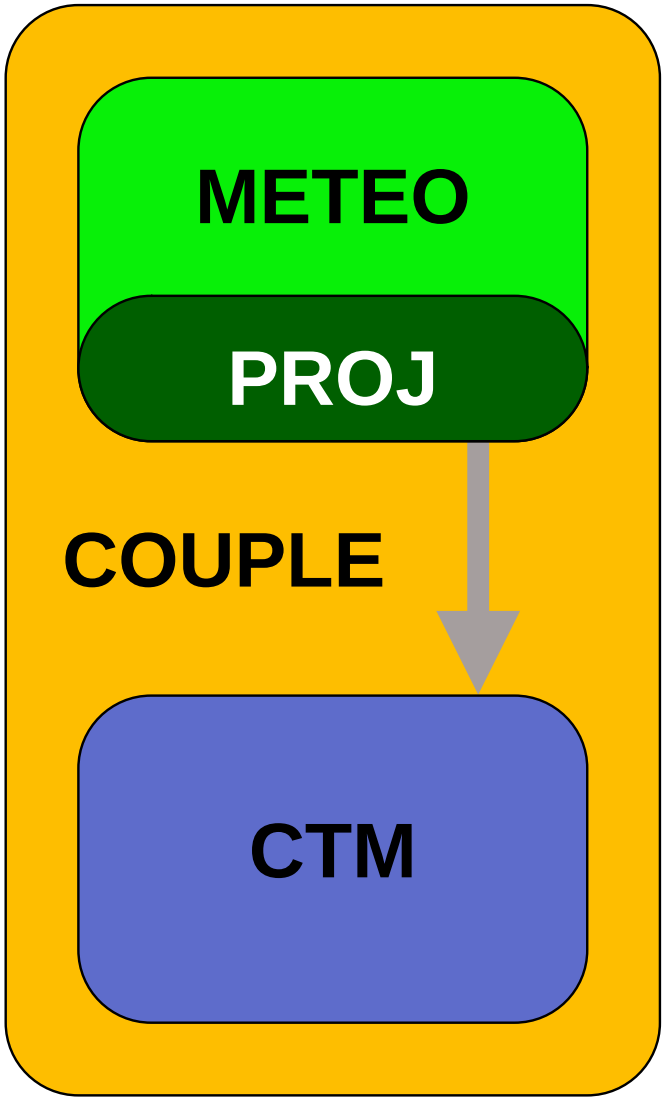
Inhalt

- kurze Einführung zum Modellsystem
- Analyse der Tracefiles mit Vampir NG

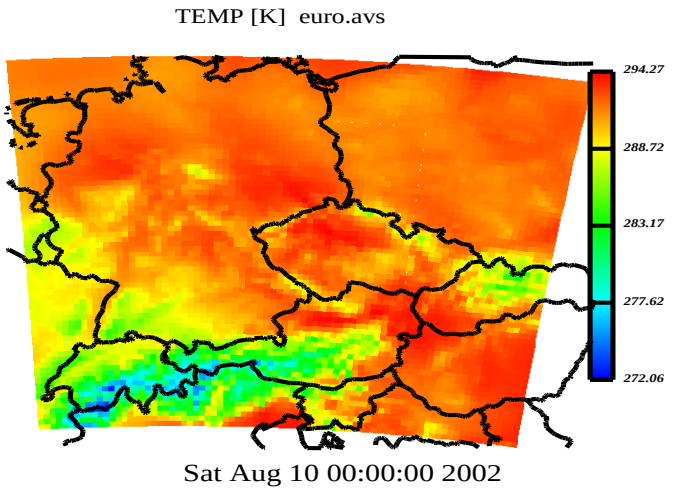
Modellsystem

- Simulation von Umwandlungs- und Transportprozessen von Schadstoffen in der Atmosphäre
- Anwendungen: Ozonbelastung, Feinstaub-Bildung, Saharastaub-Ausbreitung
- entwickelt am Leibniz-Institut für Troposphärenforschung Leipzig (IfT)
- Kopplung zweier unabhängiger Simulationsmodelle
  - Wettervorhersagemodell des DWD (**METEO**)
  - Chemie-Transport-Modell (**CTM**)

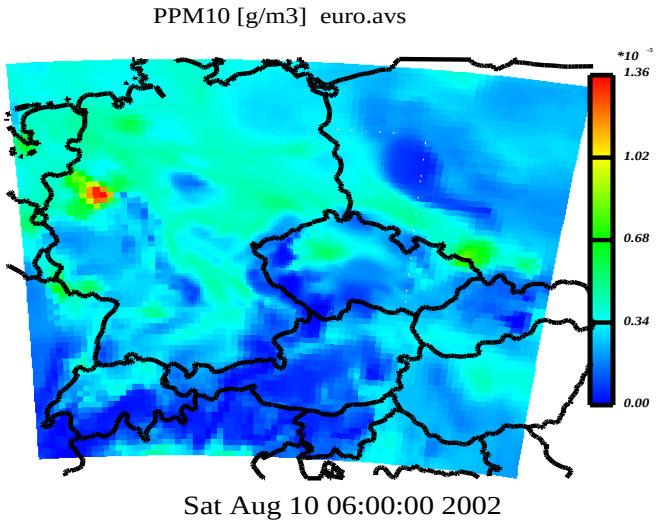
Kopplung



Temp.
Druck
Wind
...

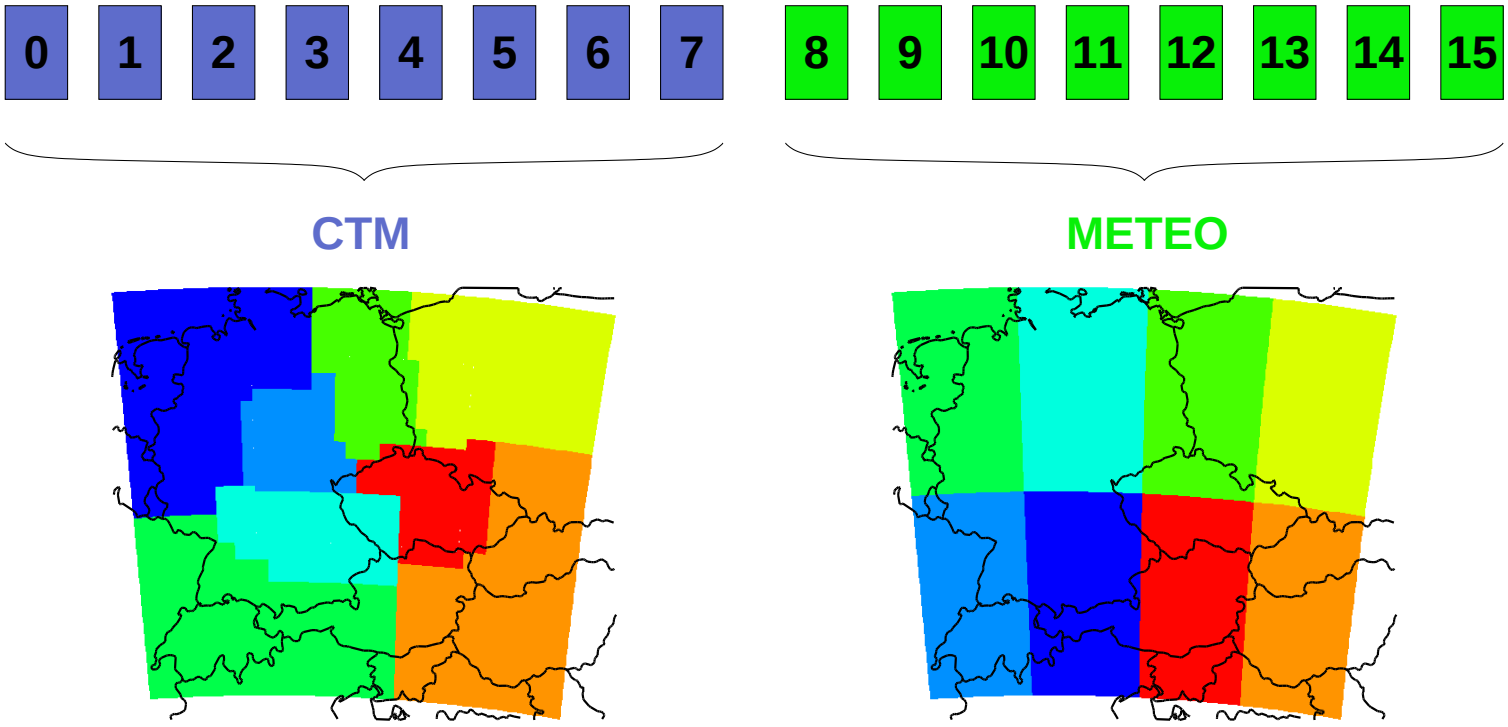


Feinstaub
Ozon
SO_2
...



Parallelisierung

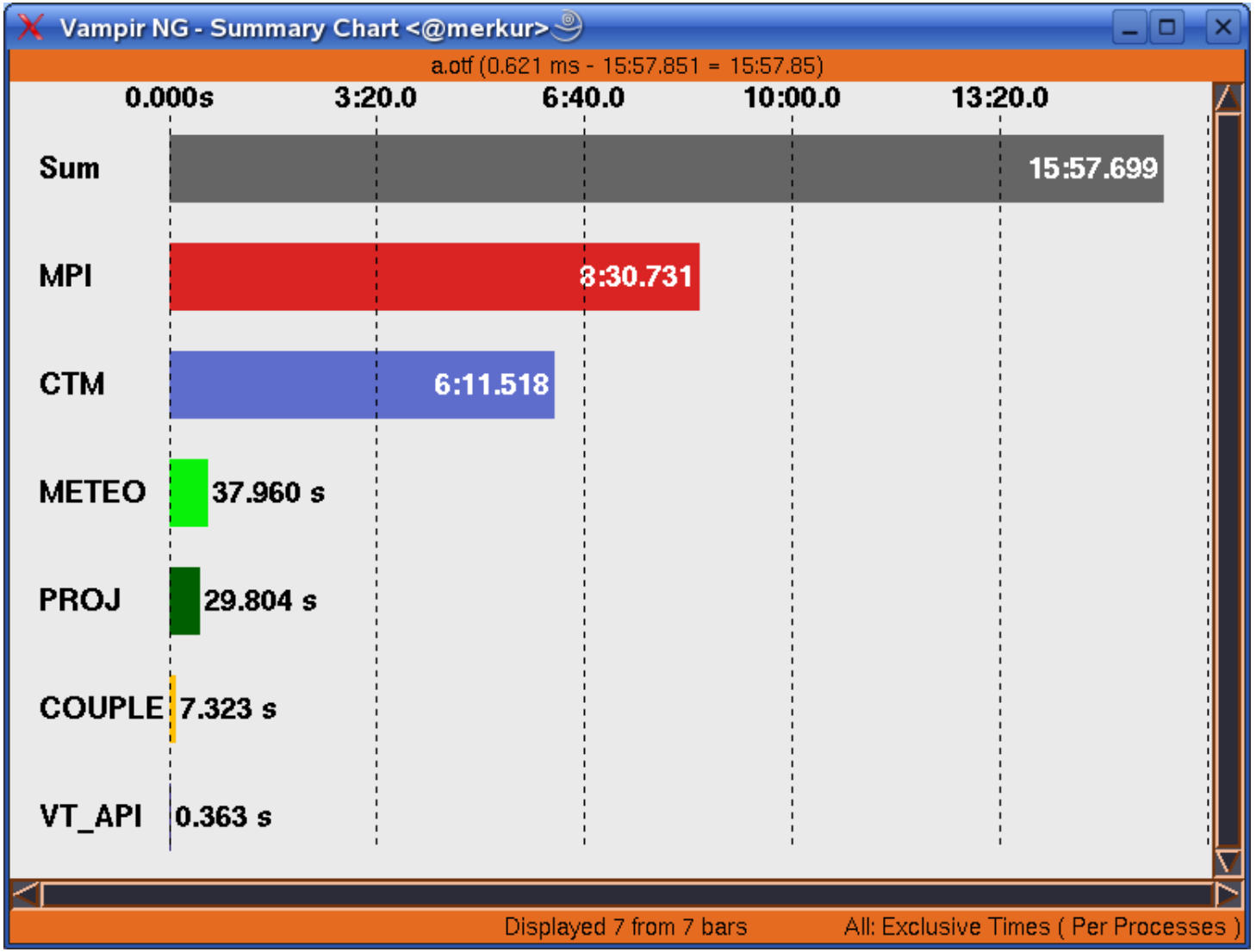
- Aufteilung der CPUs in **CTM** und **METEO**
- innerhalb der Modelle durch Aufteilung des Modell-Gebietes unter den Prozessoren



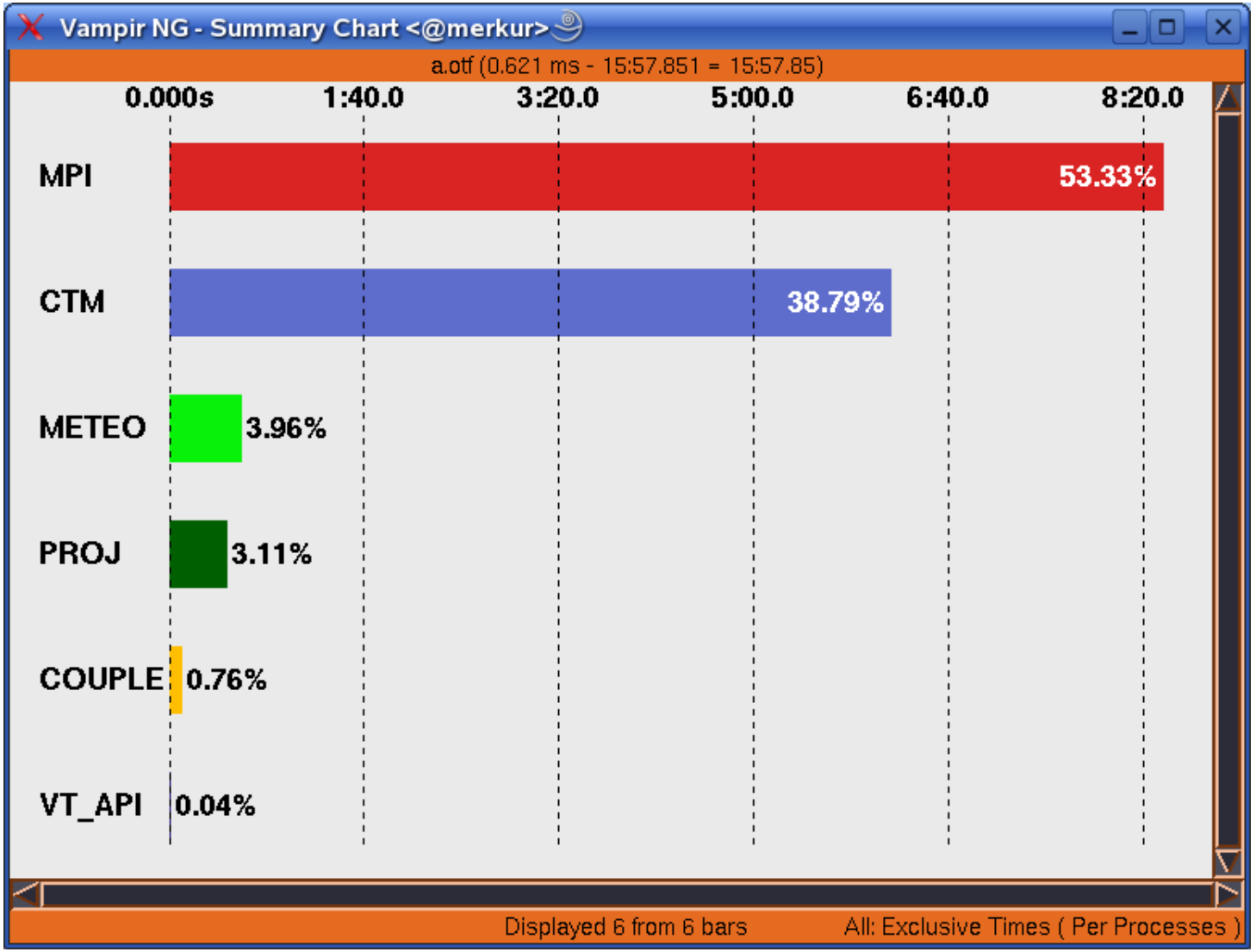
Konfiguration der Performance-Analyse

- Gruppen (*activities*):
  - Chemie-Transport-Modell (**CTM**)
  - Wettervorhersagemodell des DWD (**METEO**)
  - Projektion – notwendige Korrektur von Wind-Daten (**PROJ**)
  - Routinen zur Kopplung der beiden Modelle (**COUPLE**)
  - Kommunikation und Synchronistation via MPI (**MPI**)
- 16 CPUs auf merkur – 8 für **CTM** und 8 für **METEO**
- Vorhersagezeit: 6h
- Filter für Funktionen notwendig um Größe der Tracefiles zu reduzieren
- Größe der Tracefiles: ca. 300 MB (ohne Filter: ca. 4 GB)

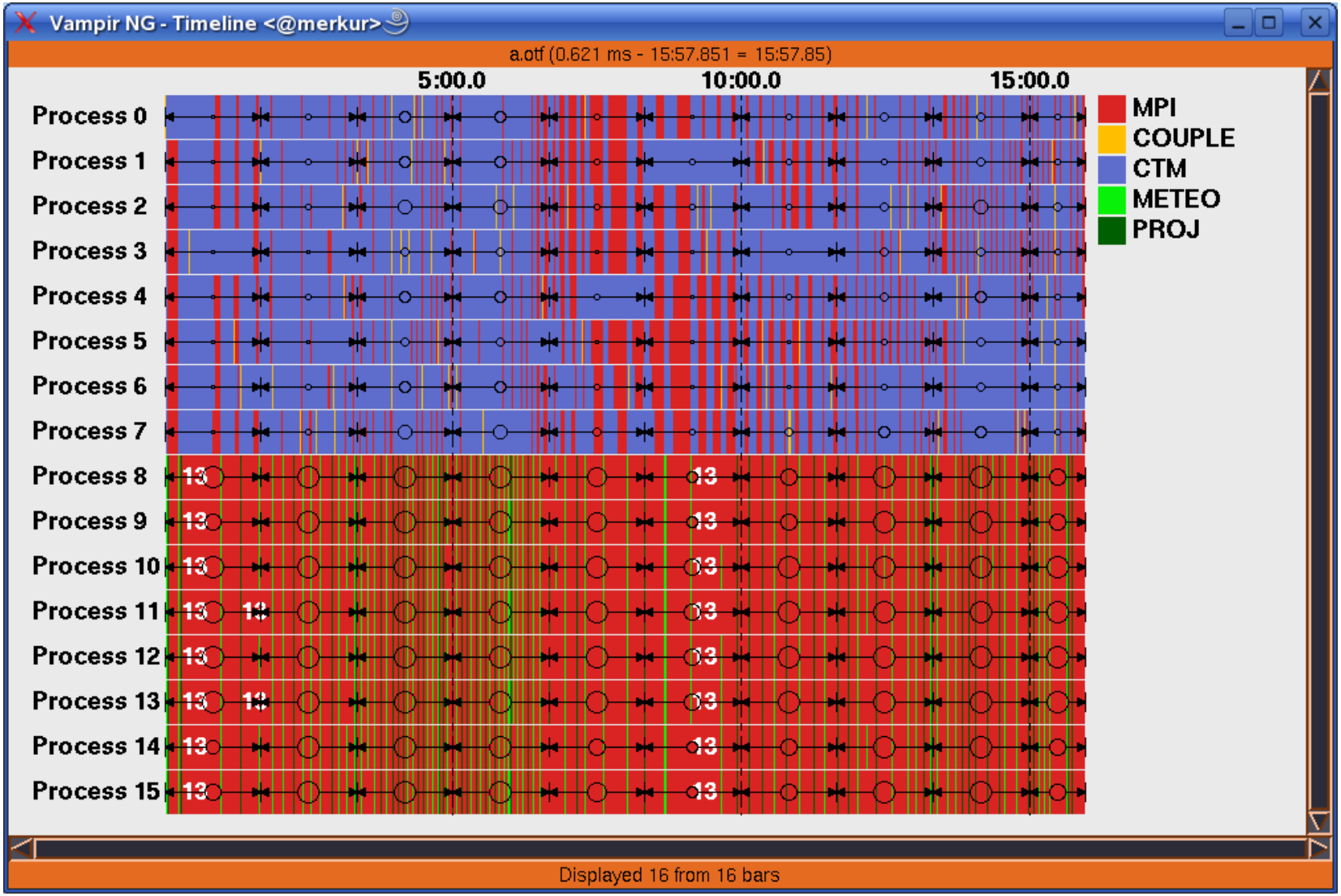
Summary Chart - Gesamtlaufzeit



Summary Chart - Anteile der Gruppen



Timeline


Vampir NG - Timeline <@merkur>
a.otf (0.621 ms - 15:57.851 = 15:57.85)
5:00.0
10:00.0
15:00.0
Process 0
Process 1
Process 2
Process 3
Process 4
Process 5
Process 6
Process 7
Process 8
Process 9
Process 10
Process 11
Process 12
Process 13
Process 14
Process 15
MPI
COUPLE
CTM
METEO
PROJ
Displayed 16 from 16 bars

Timeline - Ausschnitt

Process Activity Chart - Prozess 15 (METEO)

Diskussion

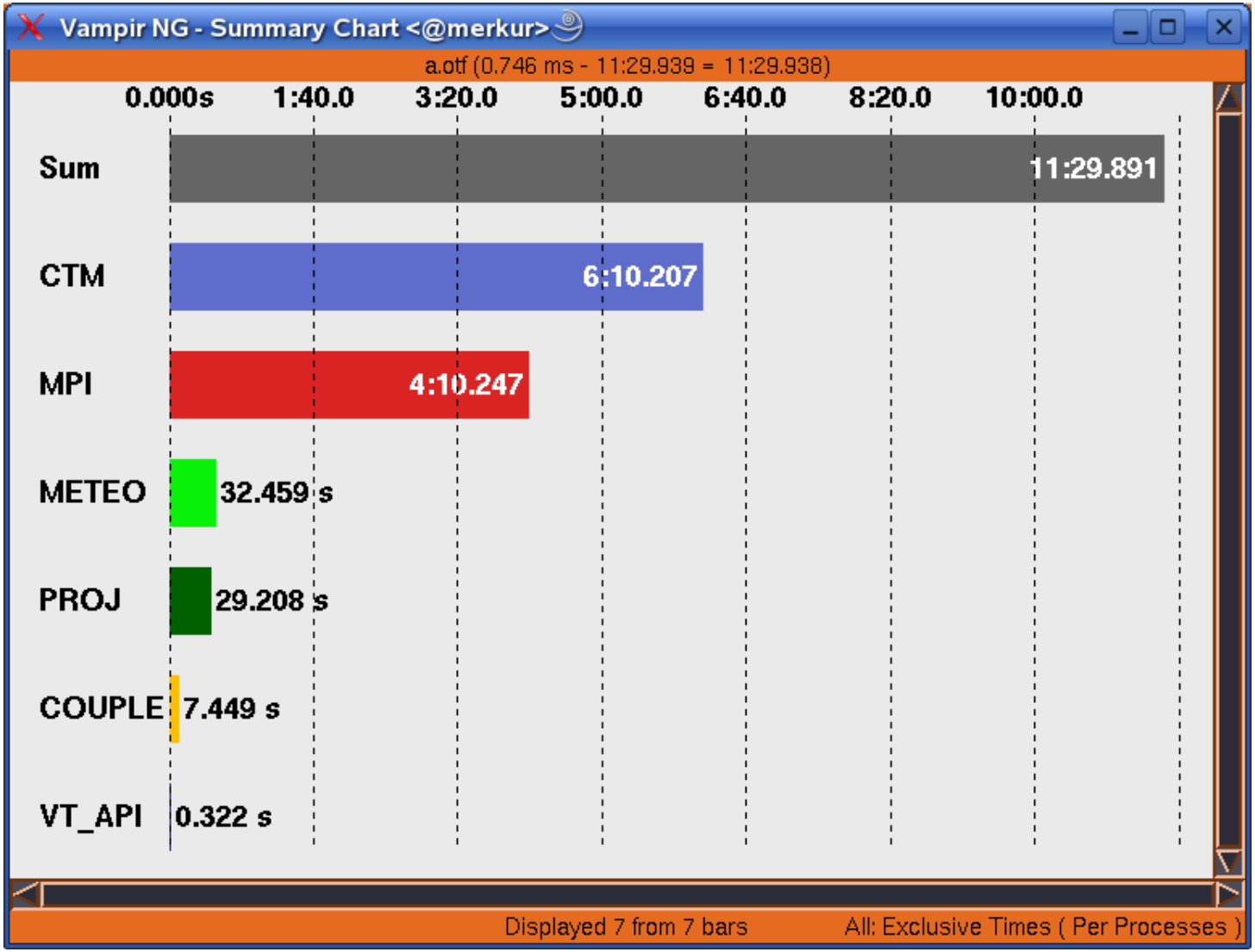
- CPUs von **METEO** haben die meiste Zeit gewartet
- Load-Imbalance durch Kopplung der Modelle
- Verbesserung: Das Verhältnis der **CTM**- zu **METEO**-CPUs optimieren!

→ weniger **METEO**-CPUs, mehr **CTM**-CPUs

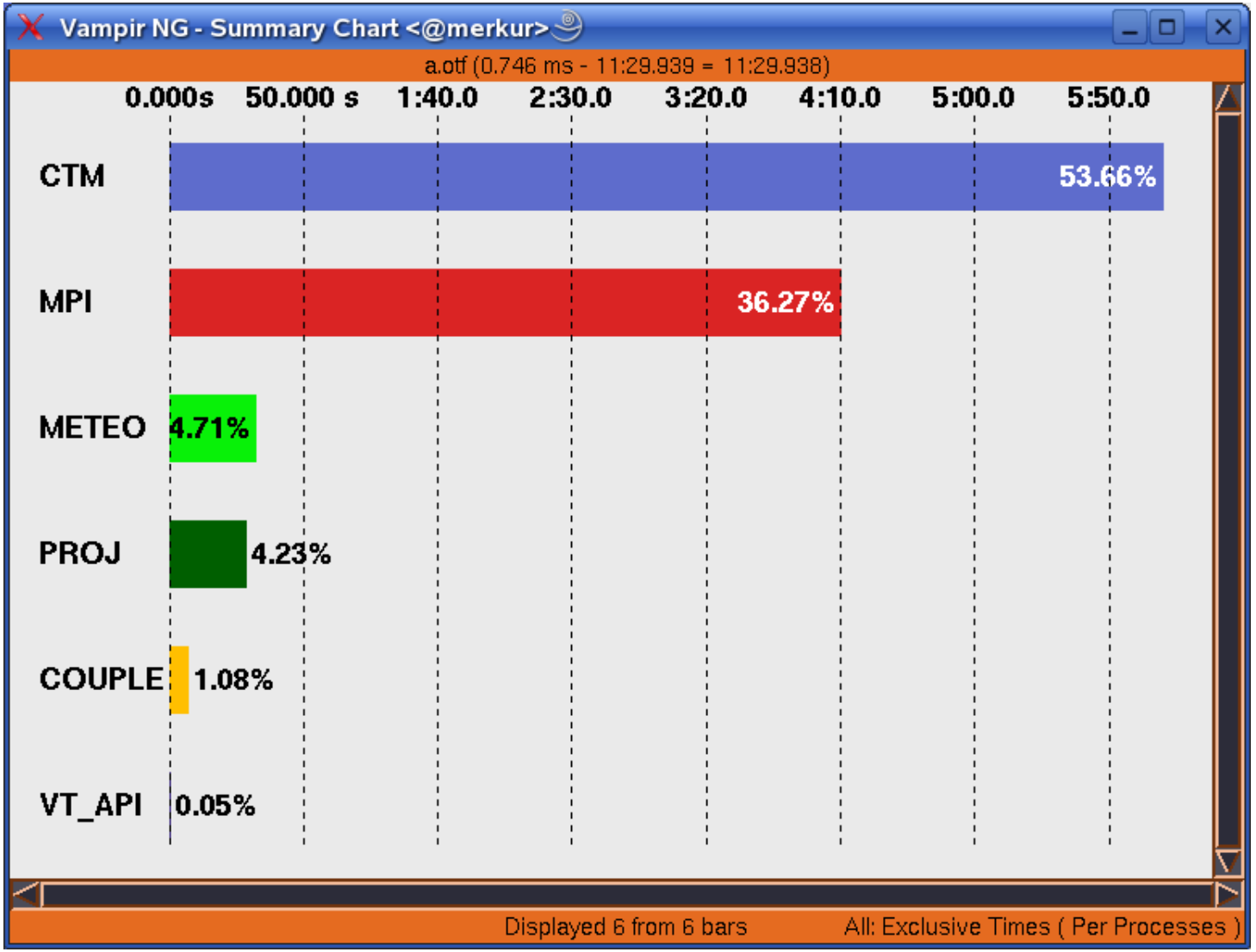
→ z. B.: 4 CPUs für **METEO** und 12 CPUs für **CTM**

→ Reduzierung der Laufzeit?

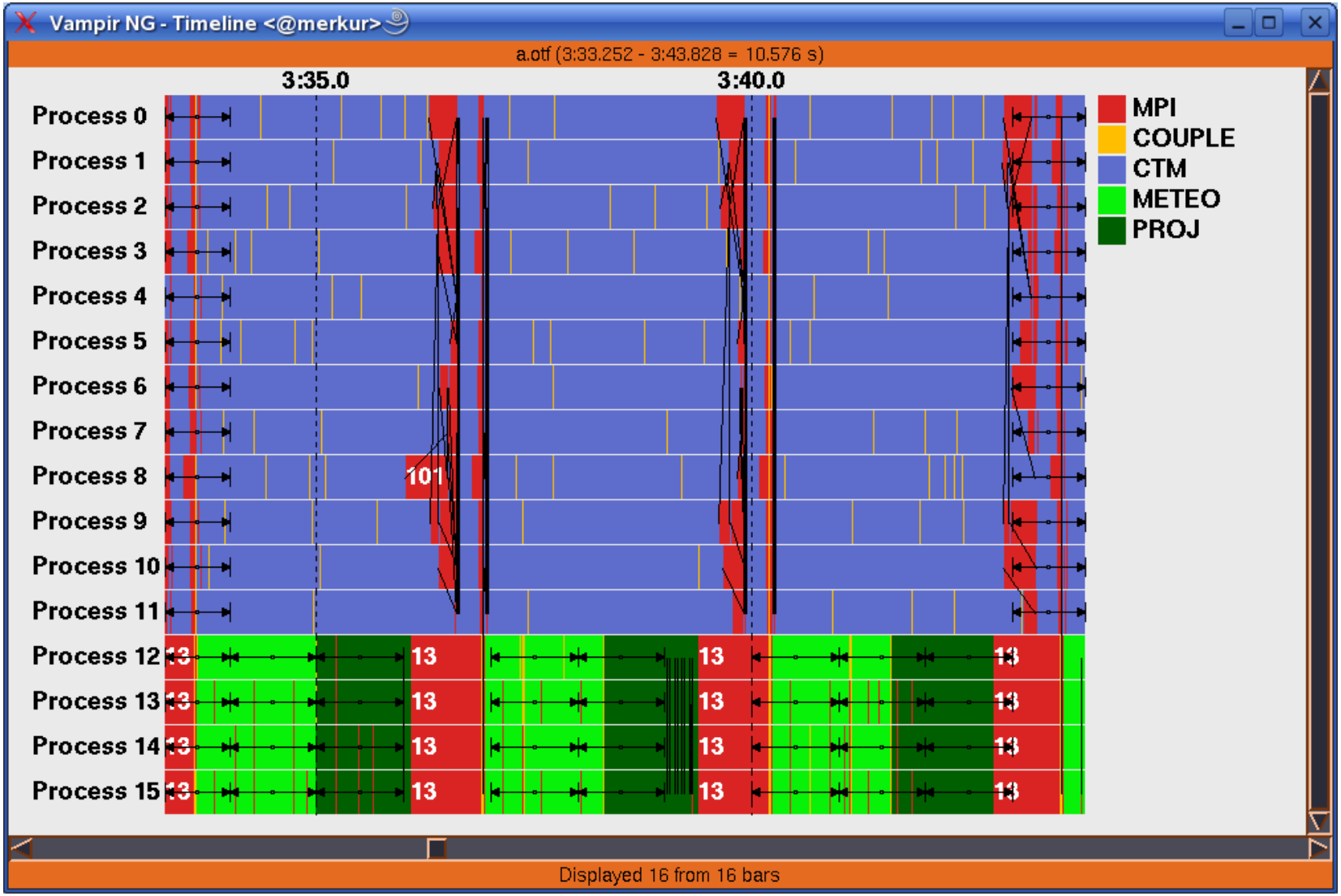
Summary Chart - Gesamtlaufzeit



Summary Chart - Anteile der Gruppen



Timeline - Ausschnitt



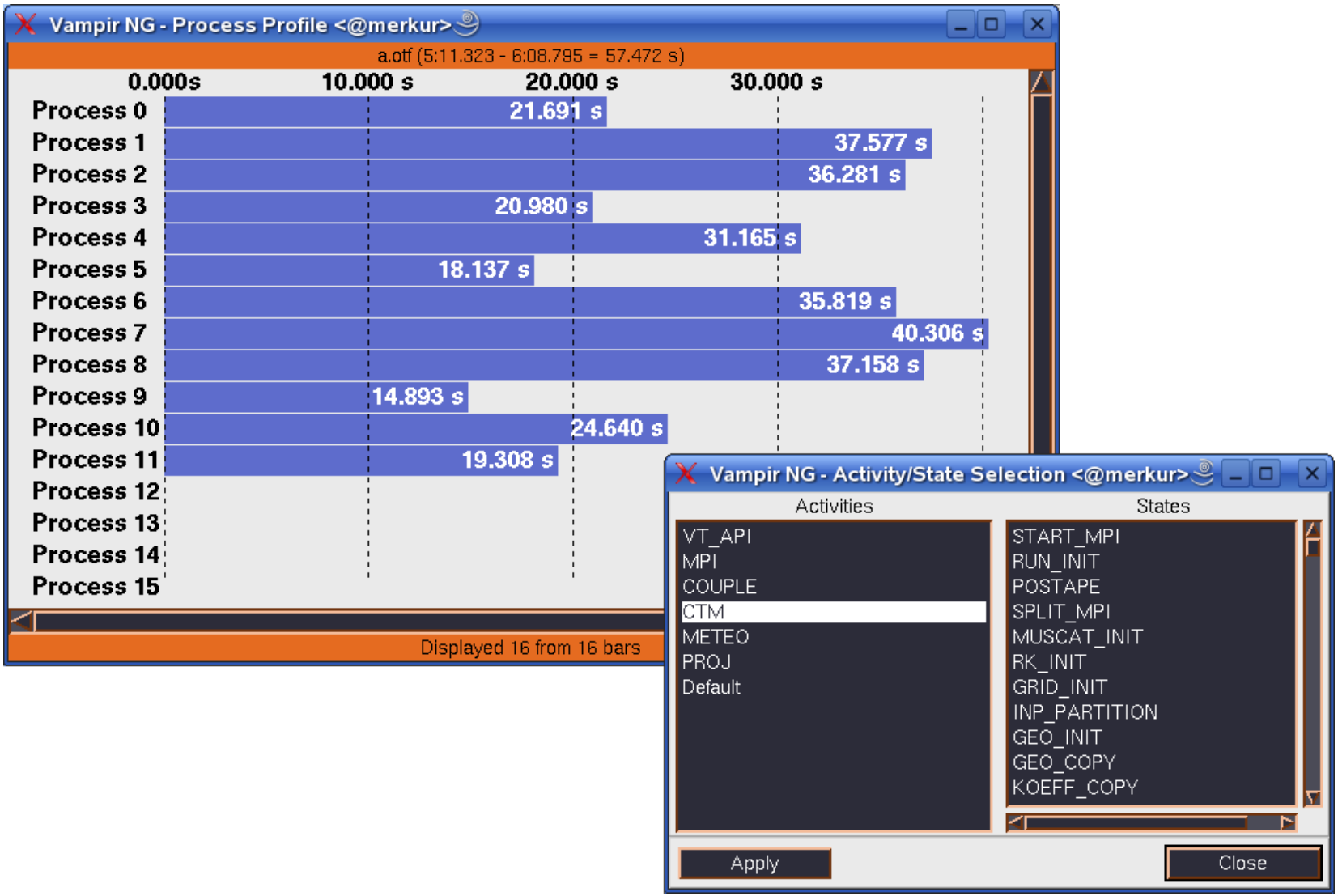
Process Activity Chart - Prozess 15 (METEO)

Timeline

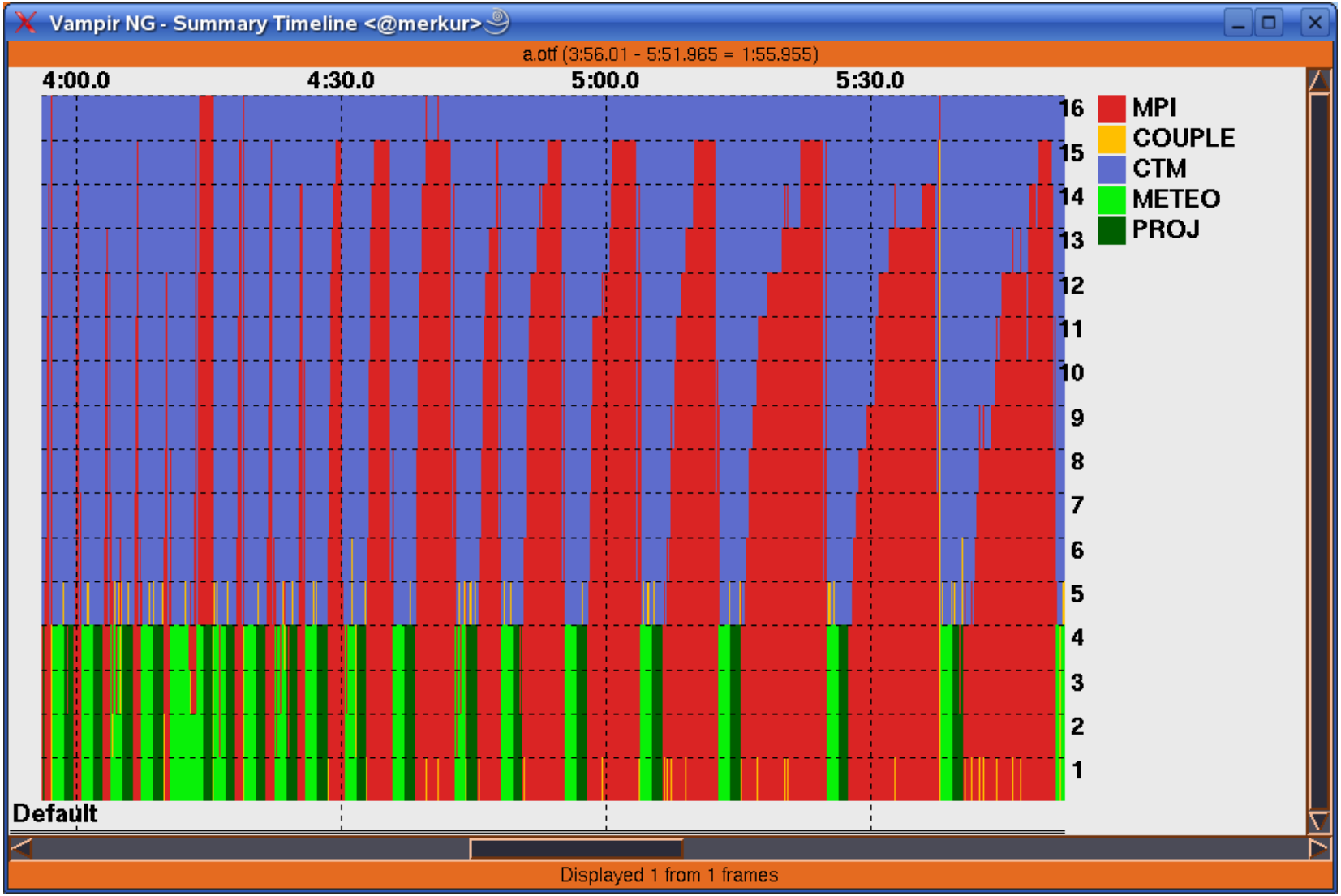

Vampir NG - Timeline <@merkur>
a.otf (0.746 ms - 11:29.939 = 11:29.938)
5:00.0
10:00.0
Process 0
Process 1
Process 2
Process 3
Process 4
Process 5
Process 6
Process 7
Process 8
Process 9
Process 10
Process 11
Process 12
Process 13
Process 14
Process 15
MPI
COUPLE
CTM
METEO
PROJ
Displayed 16 from 16 bars

Timeline - Ausschnitt

Process Profile - CTM

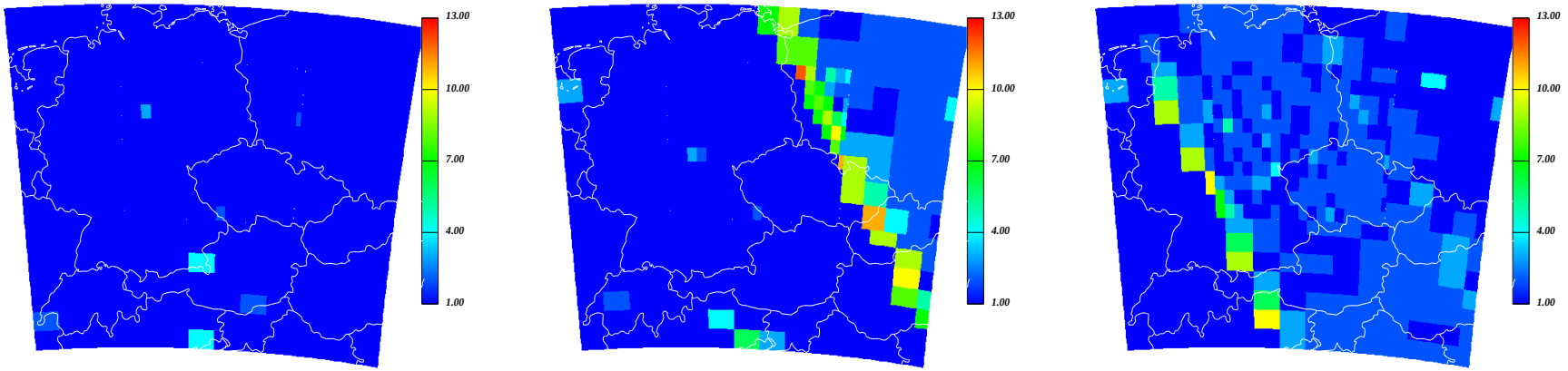


Summary Timeline



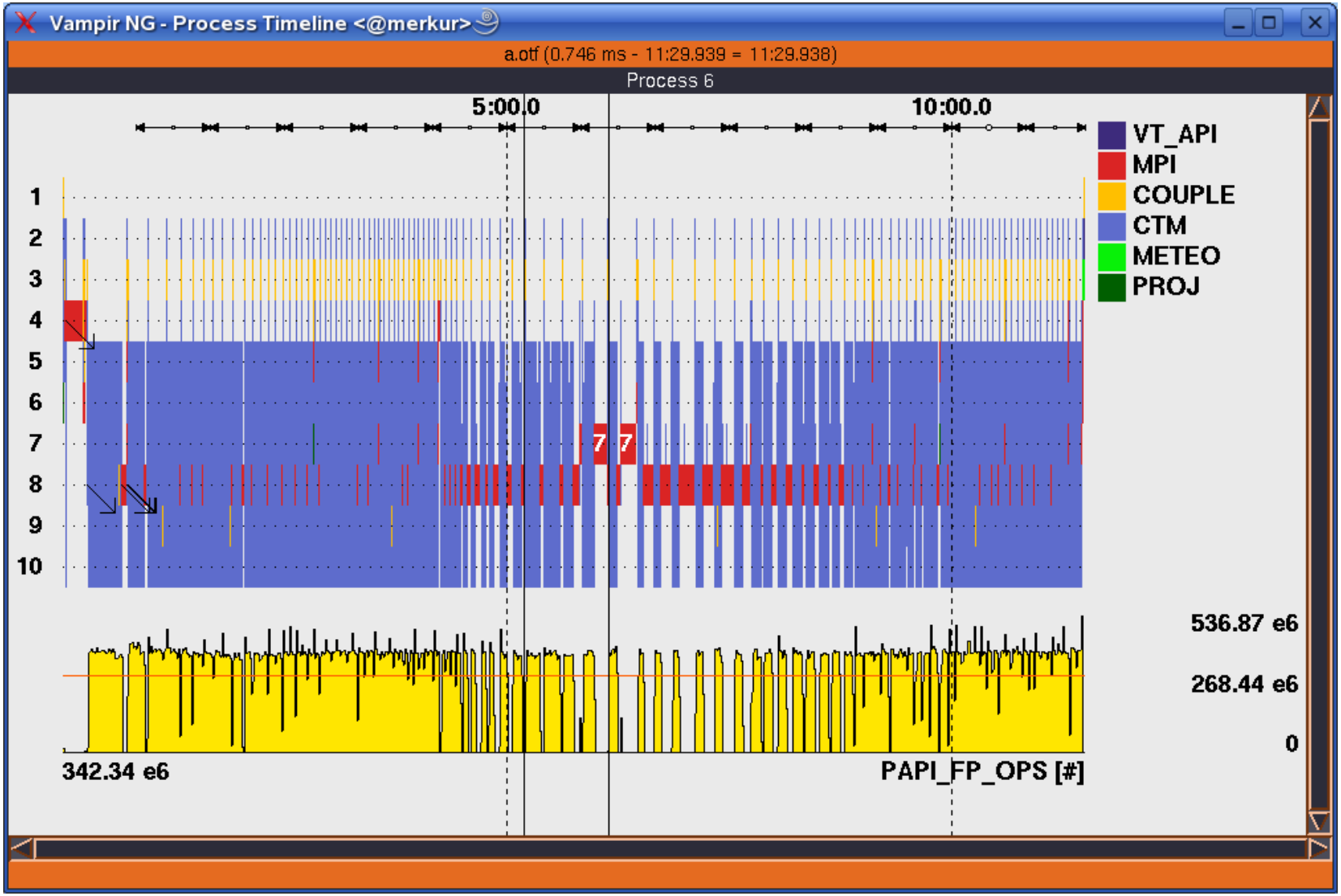
Ursachen der Load-Imbalance im CTM

Starke räumliche und zeitliche Schwankungen des Rechenaufwands, in diesem Fall zu Sonnenaufgang:

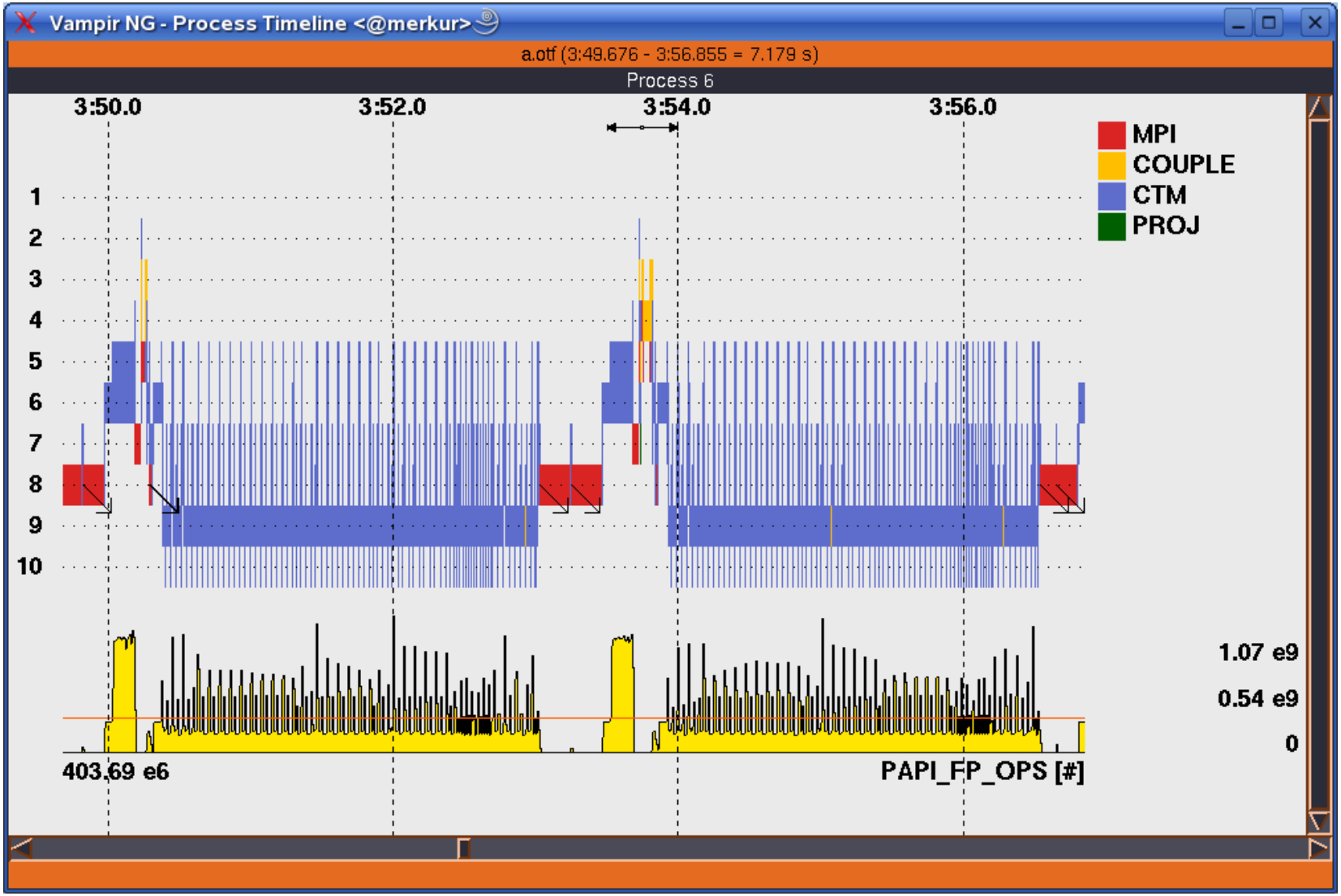


Räumliche Verteilung des Rechenaufwands zu 3 Zeitpunkten

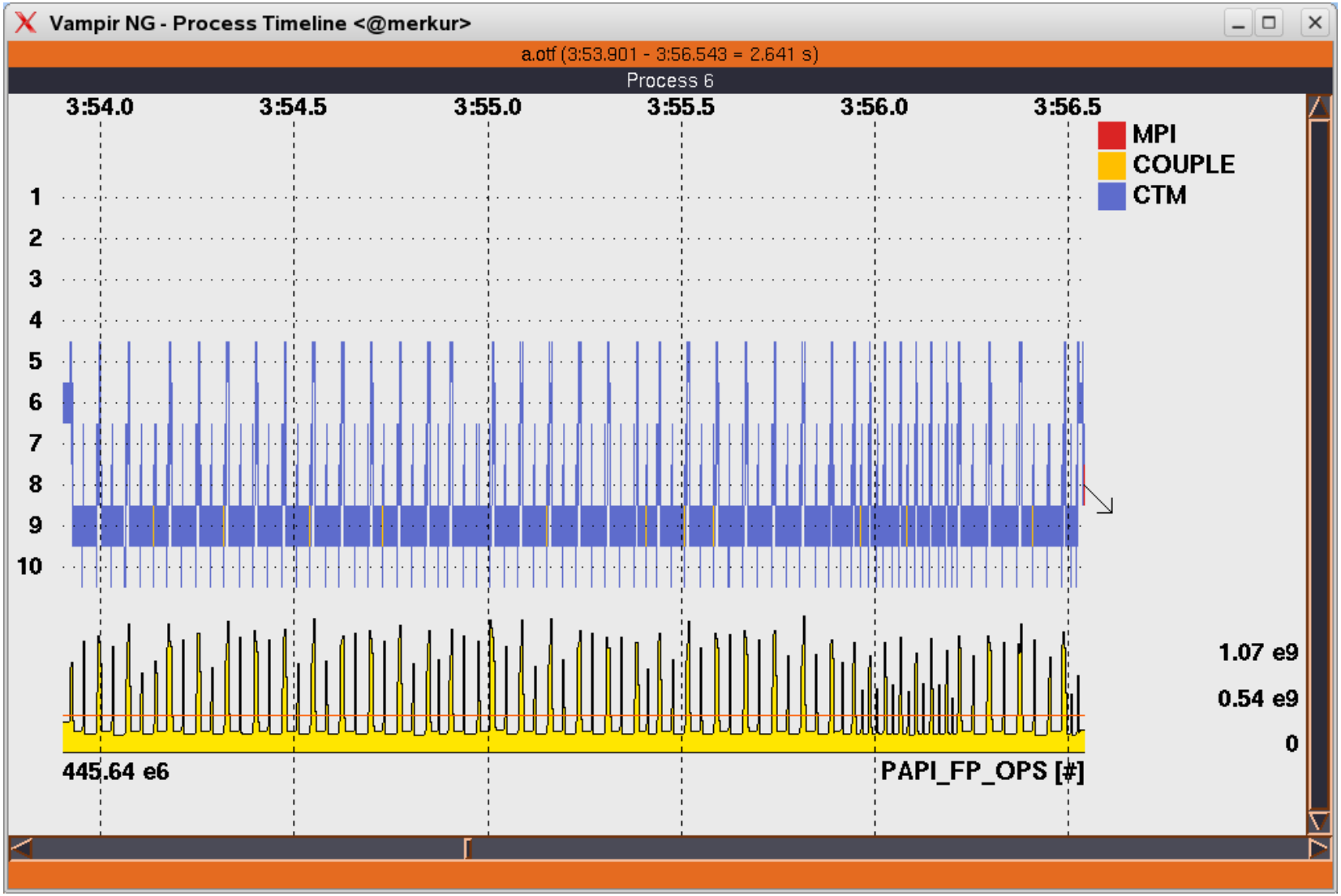
Process Timeline - Prozess 6 (CTM)



Process Timeline - Prozess 6 (CTM) - Zoom #1

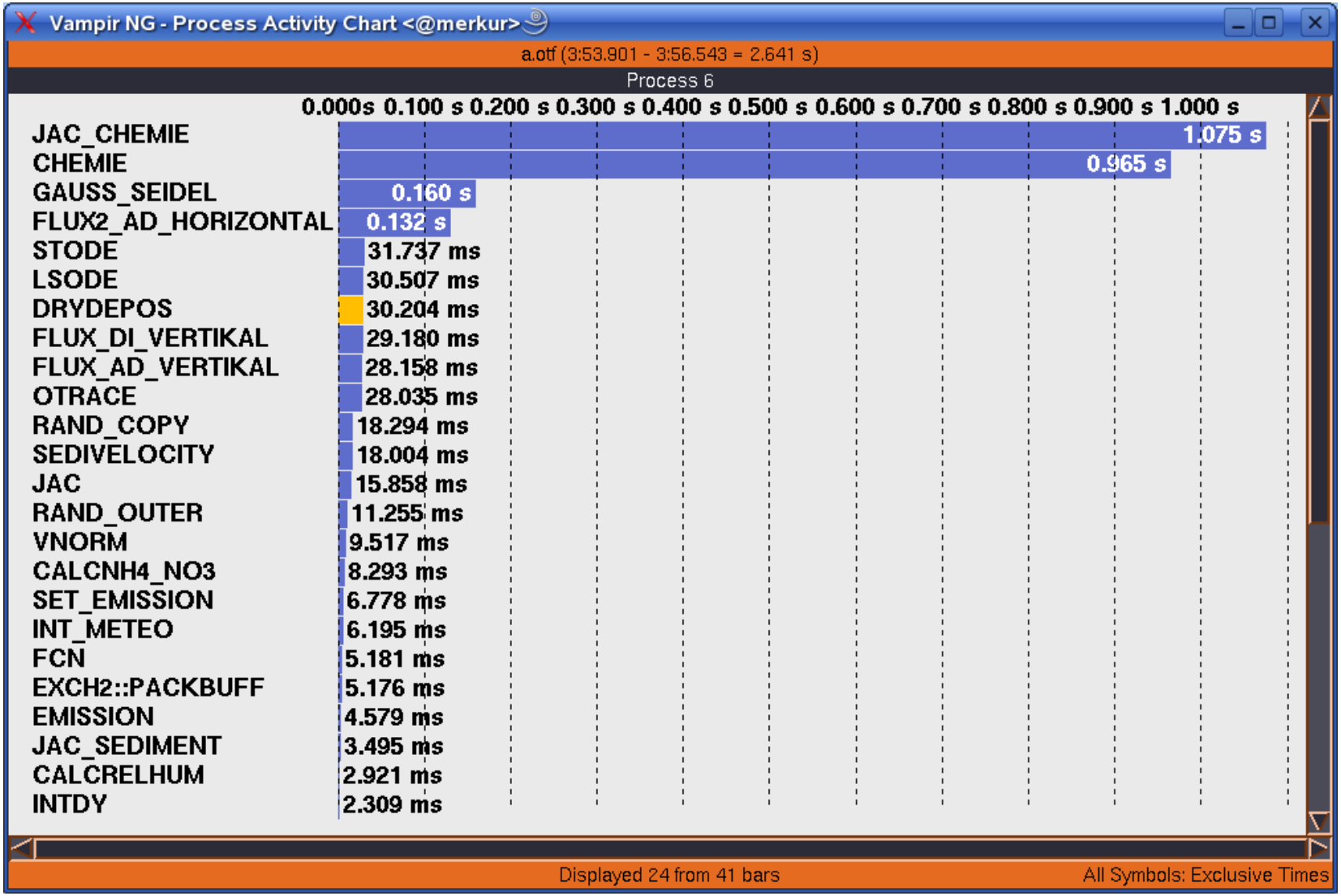


Process Timeline - Prozess 6 (CTM) - Zoom #2



Process Timeline - Prozess 6 (CTM) - Zoom #3

Process Activity Chart - Prozess 6 (CTM)

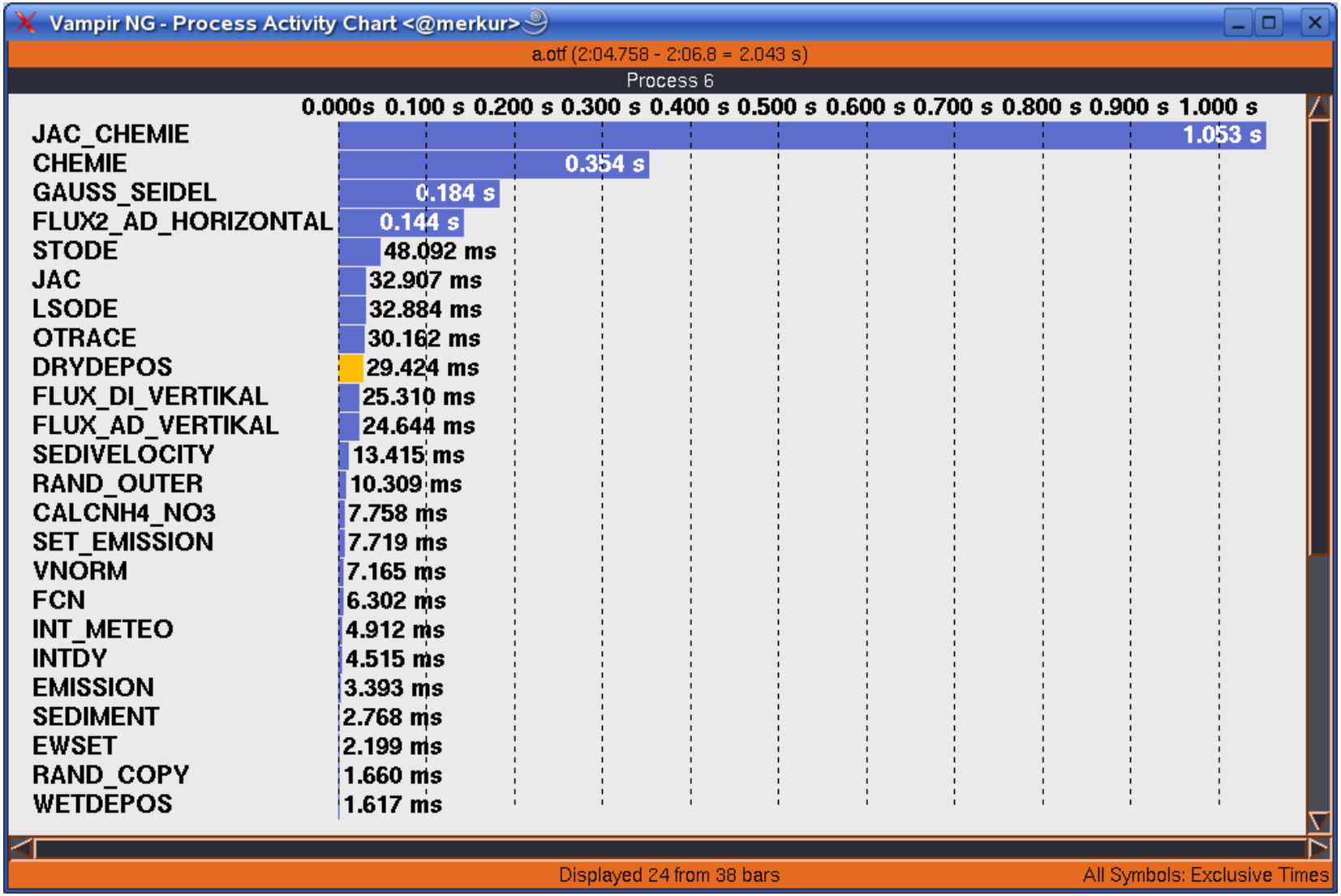


Diskussion

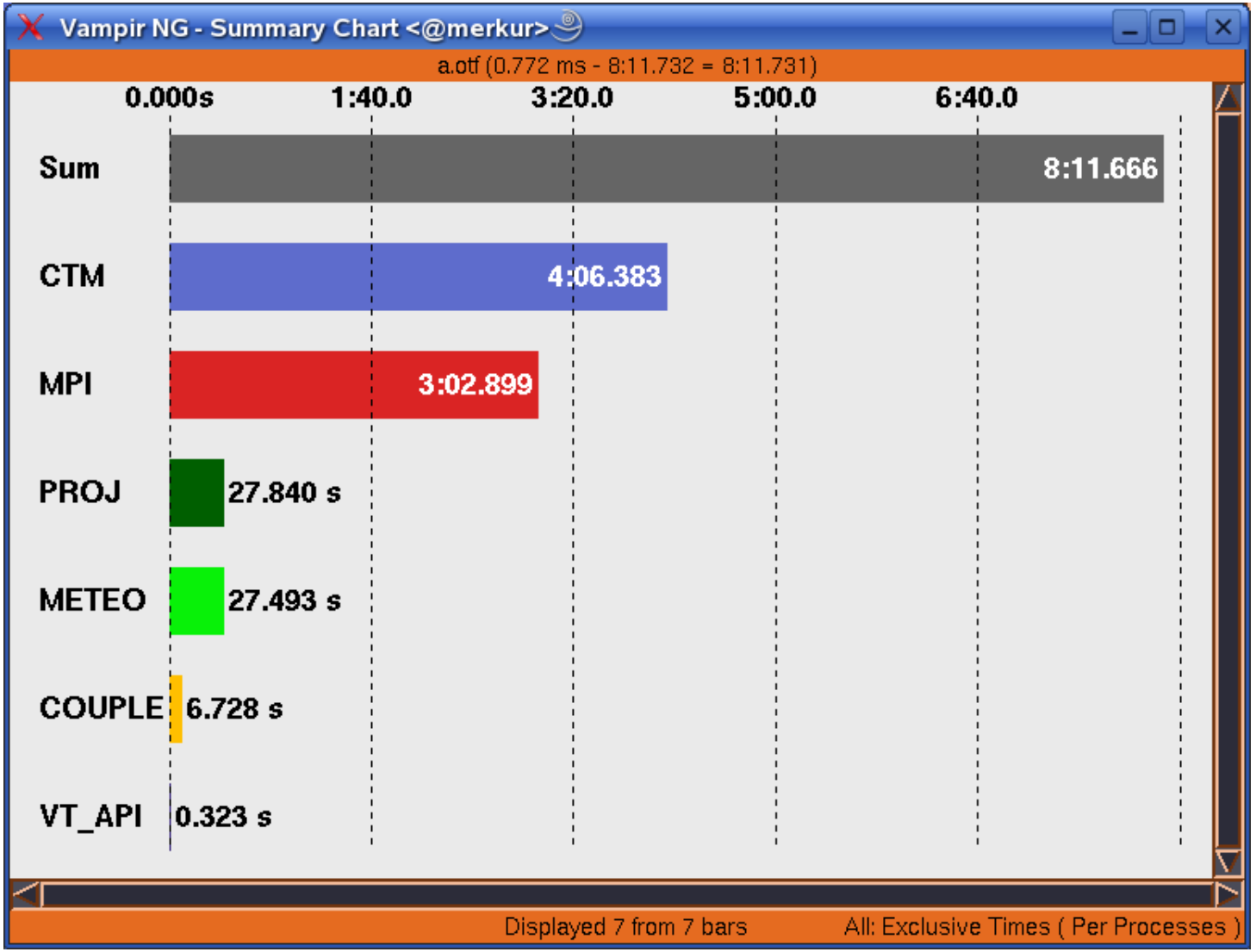
- Funktionen `chemie` und `jac_chemie` wenige FLOPS
- Jedoch sehr hohen Anteil an der Laufzeit
- Genaue Untersuchung der Ursachen notwendig!
  - Werden in diesen Funktionen viele floating point Operationen ausgeführt? $\rightarrow$ ja, d.h. Optimierung lohnenswert
  - Warum dann so wenige FLOPS? $\rightarrow$ Cache Misses einschalten, Quellcode studieren, ...
- Mit neuer Version 9.1 des Intel-Compilers...

Process Timeline - Prozess 6 (CTM) - Zoom #3

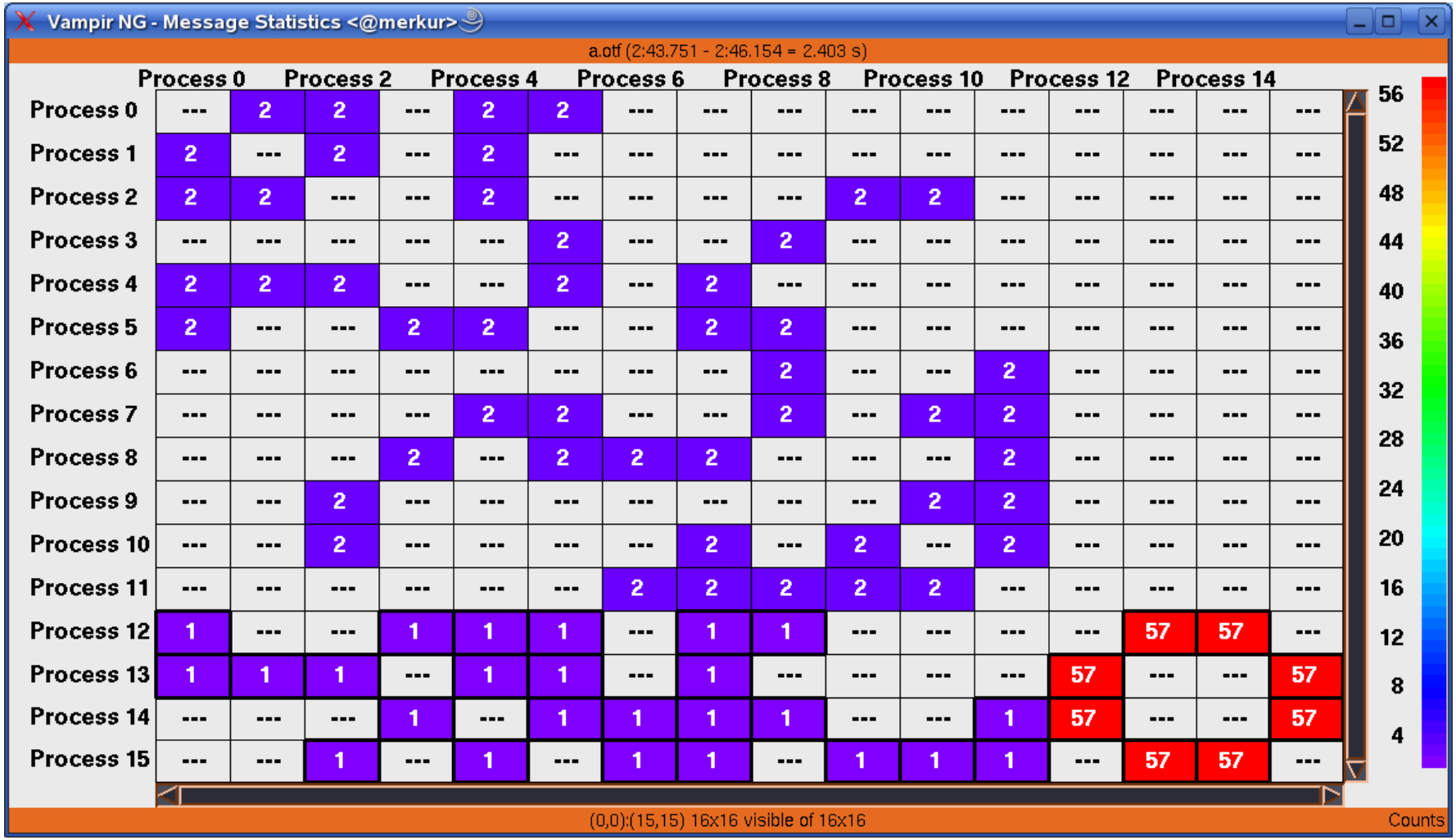
Process Activity Chart - Prozess 6 (CTM)



Summary Chart - Gesamtlaufzeit

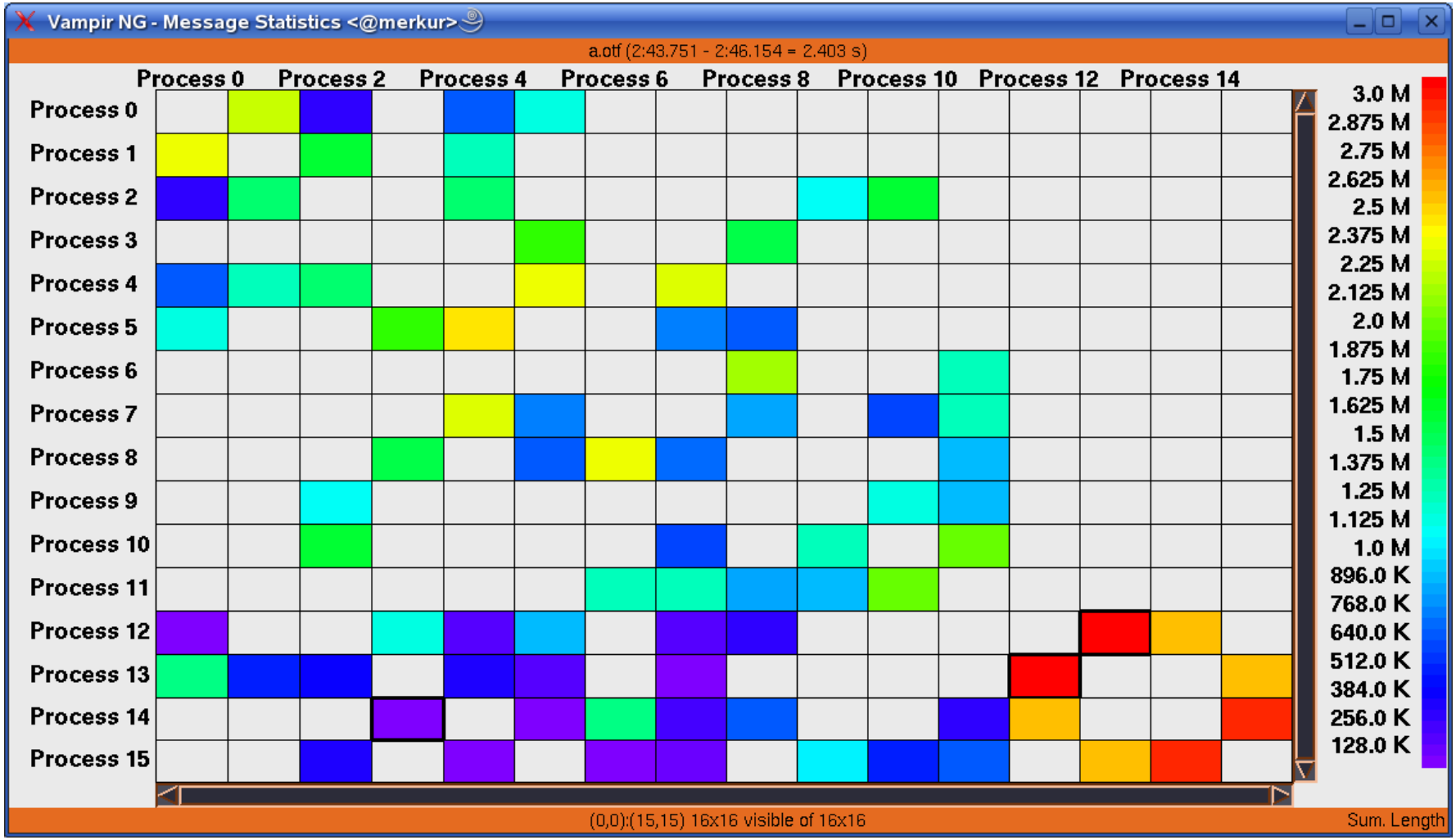


Message Statistics - Anzahl der Nachrichten



	Process 0	Process 1	Process 2	Process 3	Process 4	Process 5	Process 6	Process 7	Process 8	Process 9	Process 10	Process 11	Process 12	Process 13	Process 14	Process 15
Process 0	---	2	2	---	2	2	---	---	---	---	---	---	---	---	---	---
Process 1	2	---	2	---	2	---	---	---	---	---	---	---	---	---	---	---
Process 2	2	2	---	---	2	---	---	---	---	2	2	---	---	---	---	---
Process 3	---	---	---	---	---	2	---	---	2	---	---	---	---	---	---	---
Process 4	2	2	2	---	---	2	---	2	---	---	---	---	---	---	---	---
Process 5	2	---	---	2	2	---	---	2	2	---	---	---	---	---	---	---
Process 6	---	---	---	---	---	---	---	---	2	---	---	2	---	---	---	---
Process 7	---	---	---	---	2	2	---	---	2	---	2	2	---	---	---	---
Process 8	---	---	---	2	---	2	2	2	---	---	---	2	---	---	---	---
Process 9	---	---	2	---	---	---	---	---	---	---	2	2	---	---	---	---
Process 10	---	---	2	---	---	---	---	2	---	2	---	2	---	---	---	---
Process 11	---	---	---	---	---	---	2	2	2	2	2	---	---	---	---	---
Process 12	1	---	---	1	1	1	---	1	1	---	---	---	---	57	57	---
Process 13	1	1	1	---	1	1	---	1	---	---	---	---	57	---	---	57
Process 14	---	---	---	1	---	1	1	1	1	---	---	1	57	---	---	57
Process 15	---	---	1	---	1	---	1	1	---	1	1	1	---	57	57	---

Message Statistics - Summe der übertragenen Daten



TECHNISCHE UNIVERSITÄT DRESDEN

Center for Information Services and High Performance Computing (ZIH)

VampirTrace Event Monitor

Software-Tuning on the Phobos PC-Farm

July 4th, 2006

Zellescher Weg 12

Willers-Bau A106

Tel. +49 351 - 463 - 31945

Matthias Jurenz (matthias.jurenz@tu-dresden.de)

ZIH

Center for Information Services & High Performance Computing

Overview

- What is „VampirTrace“
- Instrumentation of user applications with VampirTrace
  - MPI function calls
  - Manually by Instrumentation API
  - Automatically by compiler
- Configuration
  - Region filter
  - Region grouping
  - Hardware counter
  - Environment variables

What is „VampirTrace“

The VampirTrace event tracing tool for serial und MPI parallelized applications produces tracefiles that can be analyzed with the Vampir performance analysis tool. It records all calls to the MPI library and all transmitted messages, and allows arbitrary user defined events to be recorded.

recordable events:

- MPI function calls
- transmitted messages

  (point to point and collective operations)
- user function calls
- hardware counter

Instrumentation of user applications with VampirTrace

- MPI calls: Automatically by PMPI Wrapper Library
- User functions
  - Manually using VampirTrace Instrumentation API
  - Automatically by Compiler (PGI, Sunf90, IBM, GNU)

Instrumentation – MPI calls (1)

- The MPI profiling interface is a part of the MPI standard specification
- Each MPI-routine call has a corresponding PMPI-call with identical syntax and functionality

```
MPI_Send(...) <=> PMPI_Send(...)
```

- This interface can be used to intercept all MPI-calls with wrapper routines:

```
int MPI_Send(...) {
      int rc;
      log_start_of_send(...);
      rc = PMPI_Send(...);
      log_end_of_send(...);
      return rc;
}
```

Instrumentation – MPI calls (2)

- Relink MPI application replacing `-lmpi` by

  C/C++:

  ```
  -L$(VT_LIB) -lelg.otf.mpi -lotf -lmpi
  ```

  Fortran:

  ```
  -L$(VT_LIB) -lfmpi -lelg.otf.mpi -lotf -lmpi
  ```

- No recompilation necessary

Instrumentation – MPI calls (3)

- Advantages
  - All MPI function calls and communication will be recorded
  - No recompilation necessary
- Disadvantages
  - **No user functions or source code regions will be recorded**

Instrumentation – Manually by Instrumentation API (1)

- Include header file for using VampirTrace instrumentation API

  C/C++: `#include “elg_user.h“`

  Fortran: `#include “elg_user.inc“`

- Place instrumentation macros/function calls at start and end of every function or region

  C:
  ```
  ELG_USER_START(“function-name“);
  ELG_USER_END(“function-name“);
  ```

  C++:
  ```
  ELG_TRACER(“function-name“);
  ```

  Fortran:
  ```
  call ELG_USER_START(‘function-name‘)
  call ELG_USER_END(‘function-name‘)
  ```

Instrumentation – Manually by Instrumentation API (2)

- Compile manually instrumented application with

  `-I$(VT_INC) -DEPILOG`

- Without `-DEPILOG` the `ELG_*` calls expand to nothing and will be ignored

- Fortran source files have to be **preprocessed**

Instrumentation – Manually by Instrumentation API (3)

- Link manually instrumented application as follows:

Sequential (no MPI):

```
-L$(VT_LIB) -lelg.otf -lotf
```

MPI:

```
-L$(VT_LIB) (-lfmpi) -lelg.otf.mpi –lotf -lmpi
```

Instrumentation – Manually by Instrumentation API (4)

- Advantages
  - Independent of used compiler
  - Also source code regions are instrumentable (e.g. loops)
  - Only functions/regions will be recorded, which are interesting for user
- Disadvantages
  - Error-prone
    (e.g. corrupt call stack due to missing return instrumentation call)
  - **Very hard instrumentation of large source codes**

Instrumentation - Automatically by Compiler (1)

- Uses the compiler profiling interface
  (e.g. PGI -> compiler flag: `-Mprof=func`)

- In the commands to build the application (e.g. in a Makefile), the user should precede all compile and link commands with „`kinst`“
  Example:

  Instead of the command

  ```
  % mpif90 myprog1.f90 myprog2.f90 –o myprog
  ```

  the command

  ```
  % kinst mpif90 myprog1.f90 myprog2.f90 –o myprog
  ```

  has to be executed

Instrumentation - Automatically by Compiler (2)

- Advantages
  - **Very simple way of instrumentation**
- Disadvantages
  - Depends on used compiler

    (supported compiler: PGI, Sunf90, IBM, GNU)
  - Tracefiles can get very large because **all** user functions will be instrumented

Configuration – Region Filter (1)

- Tracefile written by VampirTrace can easily get **very large**

  => The powerful filter management of VampirTrace is very useful to reducing size of tracefile, which will be generated

- The user can decide how often an instrumented region has to be recorded (call limit) to tracefile

- Summary of filtered regions will be written at the end of tracing

Configuration – Region Filter (2)

- Create a file which contains filter directives in following format:

```
<regions> -- <limit>
```

regions	semicolon-separated list of regions (wildcards allowed)
call limit	assigned call limit (per CPU)
	0 = region(s) denied
	-1 = unlimited
	n = set call limit to *n*

- Set path of filter configuration file by environment variable `ELG_FILTER_SPEC`

Configuration – Region Filter - Example

- Objectives:
  - Limit recording of function „`search()`“ to 100
  - Deny recording of all remaining functions except MPI calls

- Create a file „`filter.spec`“ with following content:

```
search -- 100
MPI_* -- -1
* -- 0
```

- Set environment variable `ELG_FILER_SPEC` to “`filter.spec`“

```
export ELG_FILTER_SPEC=./filter.spec
```

Configuration – Region Grouping (1)

- The region grouping management allows the user to assign functions/regions to a group
- Very useful to summarize program parts
- Vampir allows different colors for groups (activities)
- MPI functions will be assigned to group „MPI“ and remaining functions to group „USR“ by default

Configuration – Region Grouping (2)

- Create a file, which contains grouping assignments in following format:

  `<group>=<regions>`

group	group name
regions	semicolon-separated list of regions (wildcards allowed)

- Set path to group configuration file by environment variable `ELG_GROUPS_SPEC`

Configuration – Region Grouping - Example

- Objectives:
  - The functions „`add()`“, „`sub()`“, „`mul()`“ and „`div()`“ should be assigned to group „CALC“
  - All remaining functions should be assigned to group „Application“

- Create a file „`groups.spec`“ with following content:

```
CALC=add;sub;mul;div
Application=*
```

- Set environment variable `ELG_GROUPS_SPEC` to “`groups.spec`“

```
export ELG_GROUPS_SPEC=./groups.spec
```

Configuration – Hardware Counter

- If VampirTrace has been built with enabled hardware-counter support,

  VampirTrace is capable of recording hardware counter information

- Uses the PAPI hardware-counter library

- Will be recorded by each enter/leave region event

- The user must set the environment variable `ELG_METRICS`, which should contain a colon-separated list of PAPI counter names

- Example:

```
export ELG_METRICS=PAPI_FP_OPS:PAPI_TOT_INS
```

Configuration – Environment Variables (1)

Variable	Meaning
`ELG_PFORM_GDIR`	Name of global, cluster-wide directory to store final trace file
`ELG_PFORM_LDIR`	Name of node-local directory, that can be used to store temporary trace files
`ELG_FILE_PREFIX`	Prefix used for trace file names
`ELG_BUFFER_SIZE`	Size of internal event trace buffer in bytes
`ELG_VERBOSE`	Print VampirTrace related control information during measurement

Configuration – Environment Variables (2)

Variable	Meaning
`ELG_METRICS`	Specify counter metrics to be recorded with trace events as a colon-separated list of PAPI counter names
`ELG_FILTER_SPEC`	Name of file specifying region filter directives
`ELG_GROUPS_SPEC`	Name of file specifying region grouping assignments