pallas

Competence in High Performance Computing

Portable MPI Tools at Work
- Cracking Performance Problems

Werner Krotz-Vogel, Pallas GmbH

<krotz@pallas.com>

42nd CUG Conference, May 02000

Pallas GmbH
Hermülheimer Straße 10
D-50321 Brühl, Germany

info@pallas.com
www.pallas.com

Pallas Tools Software

State-of-the-art program development tools ...

... in detail:

- Vampir-2.5 (online-Demo), Vampirtrace-2.0, Dimemas

... briefly:

- Etnus TotalView 4.0 Multi-process Debugger
- PGI 3.1 x86 Compilers, Cluster Development Kit (CDK)
- KAP/Pro Toolset 3.8, OpenMP
- KAI C++ 3.4, ISO standard
- FORESYS - Fortran Restructuring Tool

... free open source:

- PMB - Pallas MPI Benchmark Suite (incl. “effective Bandwith”)

PGI Cluster Development Kit (CDK)

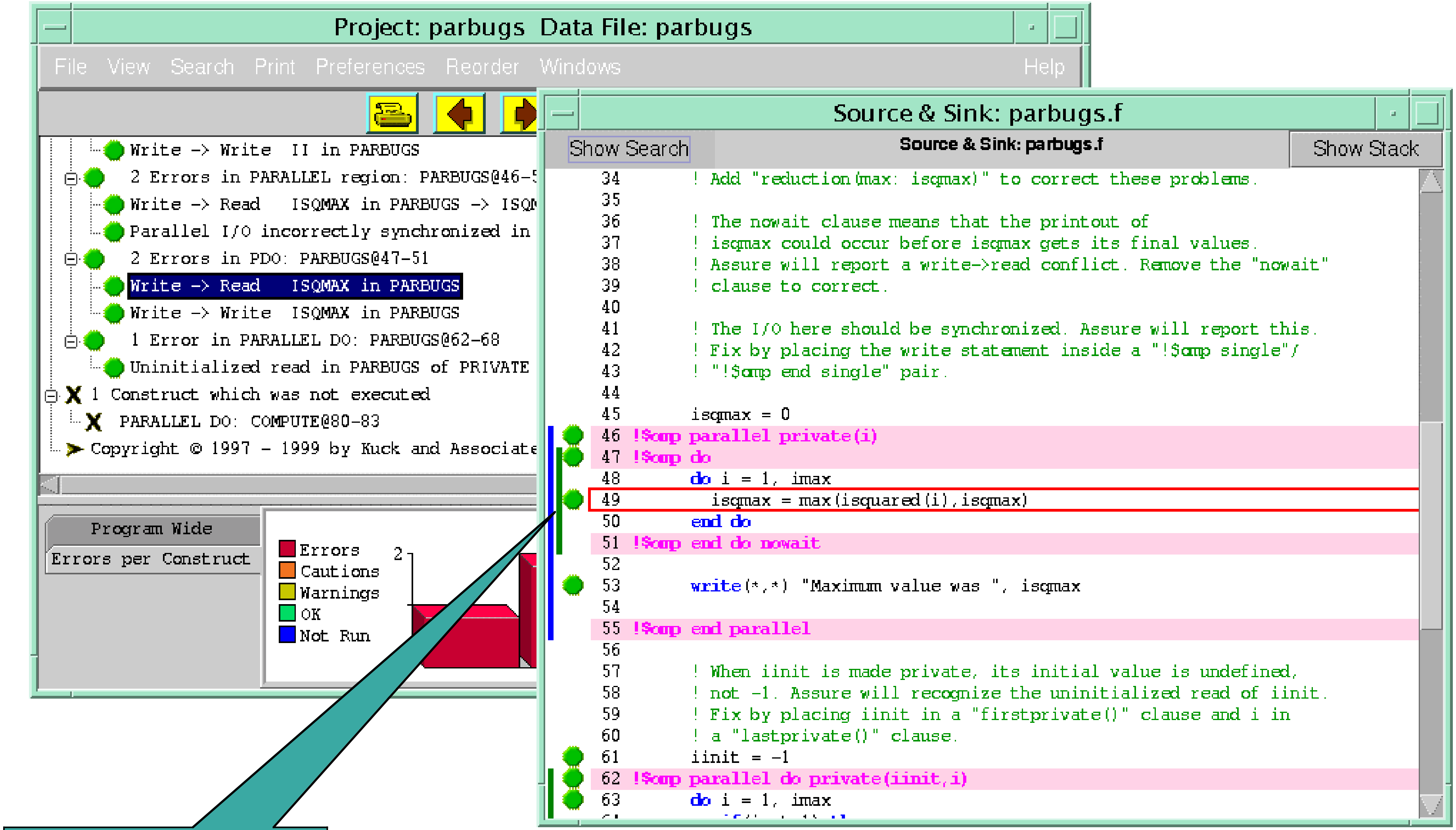
Compilers & Tools ...

- PGI 3.1 x86 compilers , C, C++, F77, F90, HPF, pgrof, pgdbg
- SMP/OpenMP support for C, C++, F77, F90

... plus convenient add-on´s:

- parallel ScaLAPACK
- optimized BLAS, LAPACK
- MPI/mpich
- PVM
- PBS - Portable Batch System
- Tutorial, examples
- Cluster management utilities

KAP/Pro Toolset - Assure Example

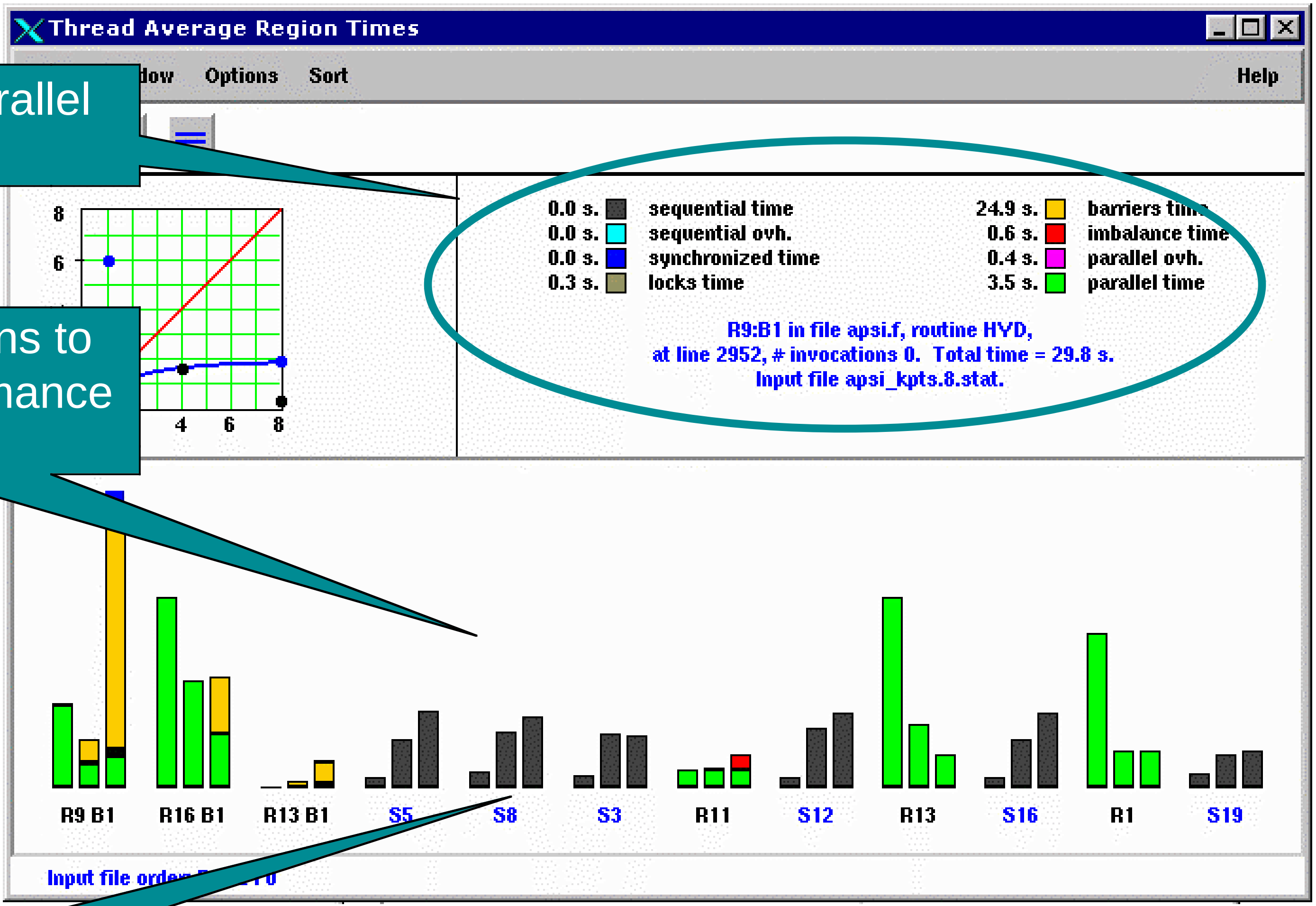


- ● Write -> Write II in PARBUGS
- ● 2 Errors in PARALLEL region: PARBUGS@46-5
  - ● Write -> Read ISQMAX in PARBUGS -> ISQM
  - ● Parallel I/O incorrectly synchronized in
- ● 2 Errors in PDO: PARBUGS@47-51
  - ● Write -> Read ISQMAX in PARBUGS
  - ● Write -> Write ISQMAX in PARBUGS
- ● 1 Error in PARALLEL DO: PARBUGS@62-68
  - ● Uninitialized read in PARBUGS of PRIVATE
- X 1 Construct which was not executed
  - X PARALLEL DO: COMPUTE@80-83
- ➤ Copyright © 1997 - 1999 by Kuck and Associate

```
34        ! Add "reduction(max: isqmax)" to correct these problems.
35
36        ! The nowait clause means that the printout of
37        ! isqmax could occur before isqmax gets its final values.
38        ! Assure will report a write->read conflict. Remove the "nowait"
39        ! clause to correct.
40
41        ! The I/O here should be synchronized. Assure will report this.
42        ! Fix by placing the write statement inside a "!$omp single"/
43        ! "!$omp end single" pair.
44
45        isqmax = 0
46  !$omp parallel private(i)
47  !$omp do
48        do i = 1, imax
49          isqmax = max(isquared(i),isqmax)
50        end do
51  !$omp end do nowait
52
53        write(*,*) "Maximum value was ", isqmax
54
55  !$omp end parallel
56
57        ! When iinit is made private, its initial value is undefined,
58        ! not -1. Assure will recognize the uninitialized read of iinit.
59        ! Fix by placing iinit in a "firstprivate()" clause and i in
60        ! a "lastprivate()" clause.
61        iinit = -1
62  !$omp parallel do private(iinit,i)
63        do i = 1, imax
```

Source location

KAP/Pro Toolset - GuideView Example



0.0 s.	⬛	sequential time	24.9 s.	🟨	barriers time
0.0 s.	🟦	sequential ovh.	0.6 s.	🟥	imbalance time
0.0 s.	🟦	synchronized time	0.4 s.	🟪	parallel ovh.
0.3 s.	🟫	locks time	3.5 s.	🟩	parallel time

R9:B1 in file apsi.f, routine HYD,
at line 2952, # invocations 0. Total time = 29.8 s.
Input file apsi_kpts.8.stat.

Analyse each parallel region

Sort or filter regions to navigate to performance hotspots

Identify serial regions that hurt scalability

KAI C++

The most modern, best performing, platform independant C++

- ISO C++ standard syntax, including exeptions and member templates
- ISO C++ standard class library
- multi-platform support
- meet C performance requirements
- thread safety (on most platforms)

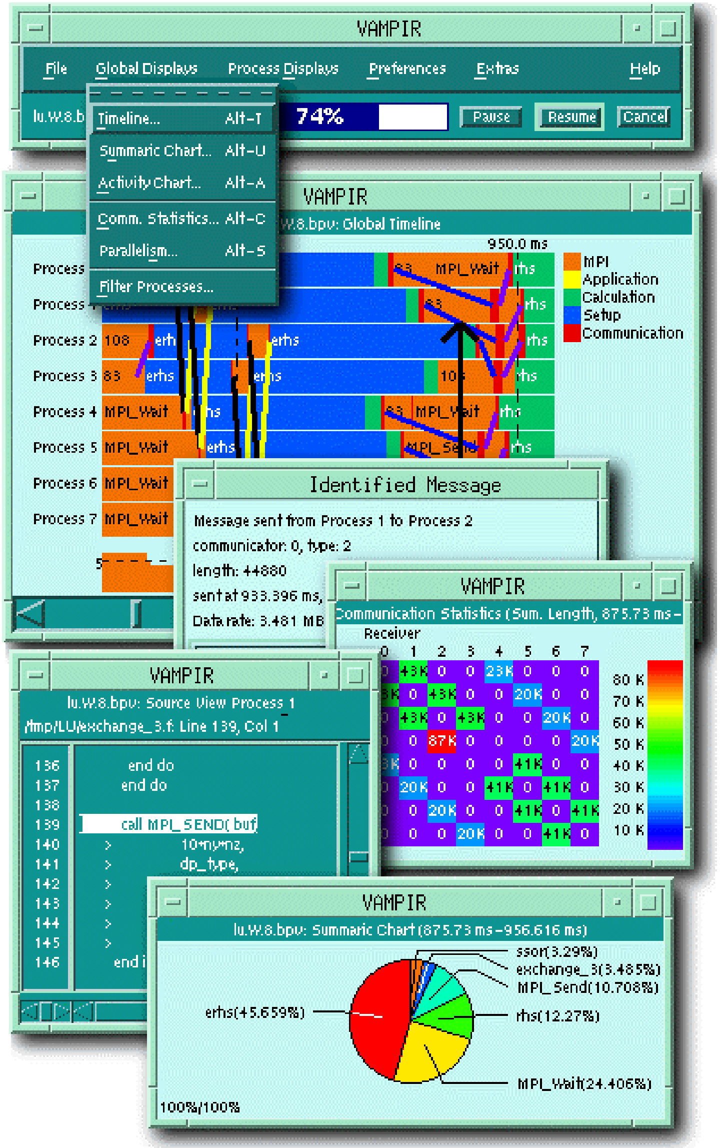
FORESYS

- Translates FORTRAN code (F77 - F95) into abstract syntax tree (ForLib)
- FORTRAN code consistency checks (definitions of functions, common blocks etc.)
- Interactive visualization & analysis of inconsistencies
- Upgrading from FORTRAN 77 to FORTRAN 90
- Interactive/batch analysis of parallelization possibilities
- Automatic code quality analysis/improvements

pallas

Vampir 2.5

Visualization and Analysis of MPI Programs



Message sent from Process 1 to Process 2
communicator: 0, type: 2
length: 44880
sent at 933.396 ms,
Data rate: 3.481 MB

Communication Statistics (Sum. Length, 875.73 ms –

Receiver

	1	2	3	4	5	6	7
0	43K	0	0	23K	0	0	0
3K	0	43K	0	0	20K	0	0
0	43K	0	43K	0	0	20K	0
0	0	87K	0	0	0	0	20K
3K	0	0	0	0	41K	0	0
0	20K	0	0	41K	0	41K	0
0	0	20K	0	0	41K	0	41K
0	0	0	20K	0	0	41K	0

lu.W.8.bpv: Source View Process 1
/tmp/LU/exchange_3.f: Line 139, Col 1

```
       end do
     end do

     call MPI_SEND( buf
>              10*ny*nz,
>              dp_type,
>
>
>
>
     end i
```

lu.W.8.bpv: Summaric Chart (875.73 ms–956.616 ms)

Vampir

- New version: Vampir 2.5

- Significant new features
  - support for collective MPI operations
  - trace comparison
  - tracefile re–write
  - message–length histogram
  - local and global calling trees
  - source–code reference
  - support for MPI–2 I/O operations

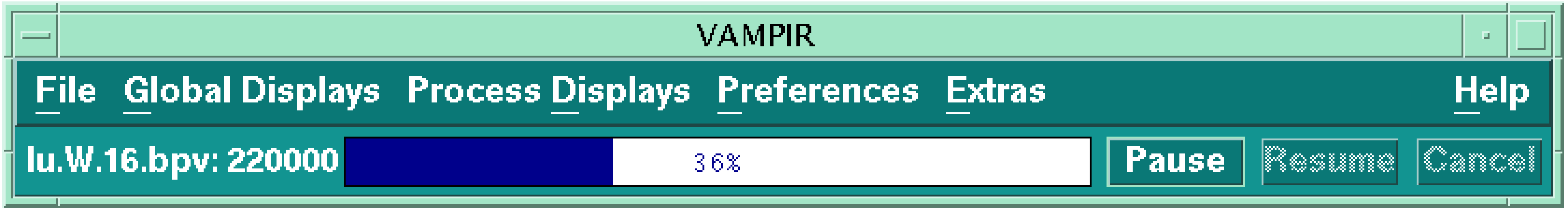
Vampir Features

- Offline trace analysis for MPI (and others ...)
- Traces generated by Vampirtrace tool (`ld ... -lVT -lpmpi -lmpi`)
- Convenient user–interface
- Scalability in time and processor–space
- Excellent zooming and filtering
- High–performance graphics
- Display and analysis of MPI and application events:
  - execution of MPI routines
  - point–to–point and collective communication
  - MPI–2 I/O operations
  - execution of application subroutines (optional)

- Easy customization

Vampir Main Window

Vampir 2.5 main window



- Tracefile loading can be interrupted at any time
- Tracefile loading can be resumed
- Tracefile can be loaded starting at a specified time offset

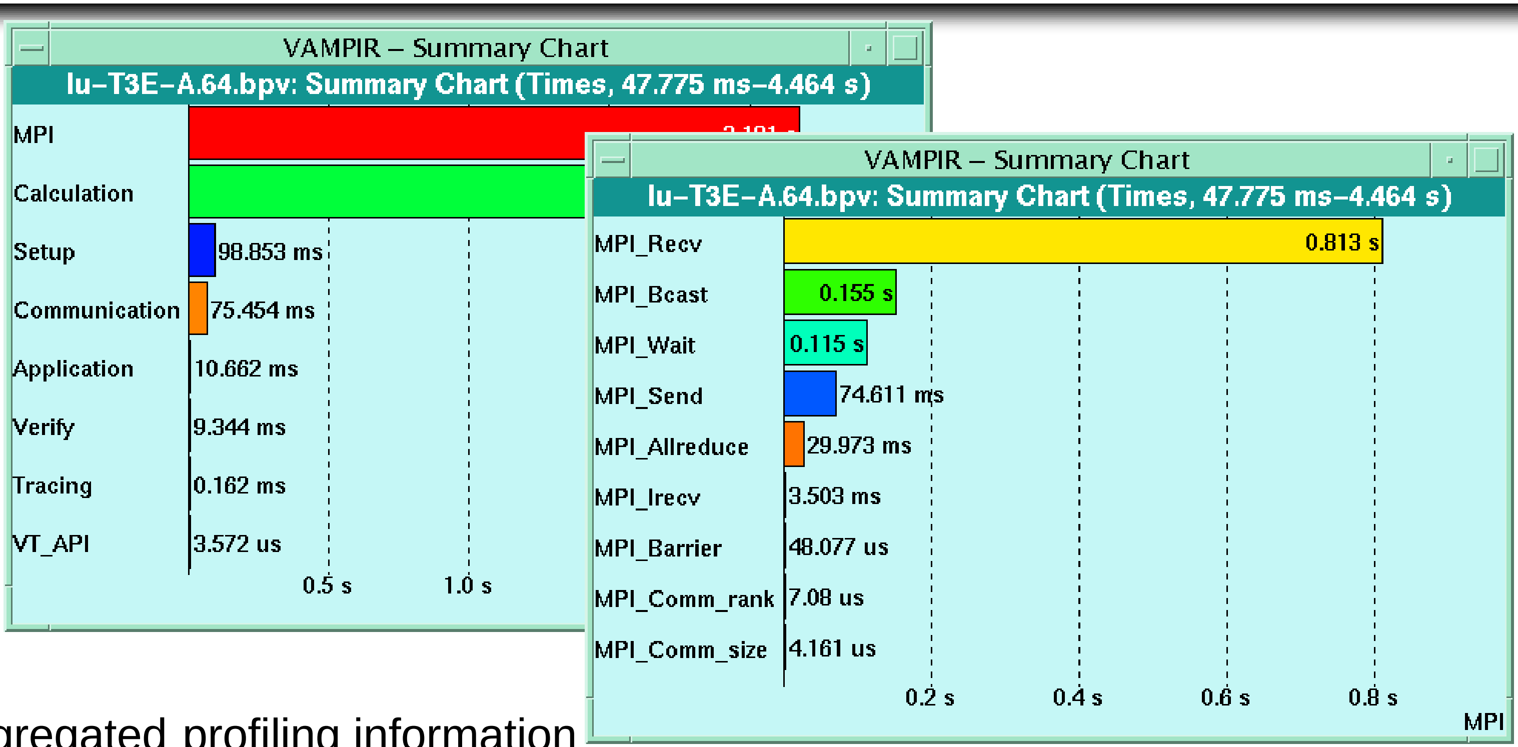
- Tracefile can be re–written (re–grouped symbols)

Vampir Displays

- Global displays show all selected processes
  - Summary Chart: aggregated profiling information
  - Activity Chart: presents per–process profiling information
  - Timeline: detailed application execution over time axis
  - Communication statistics: message statistics for each process pair
  - Global Comm. Statistics: collective operations statistics
  - I/O Statistics: MPI I/O operation statistics
  - Calling Tree: draws global or local dynamic calling trees

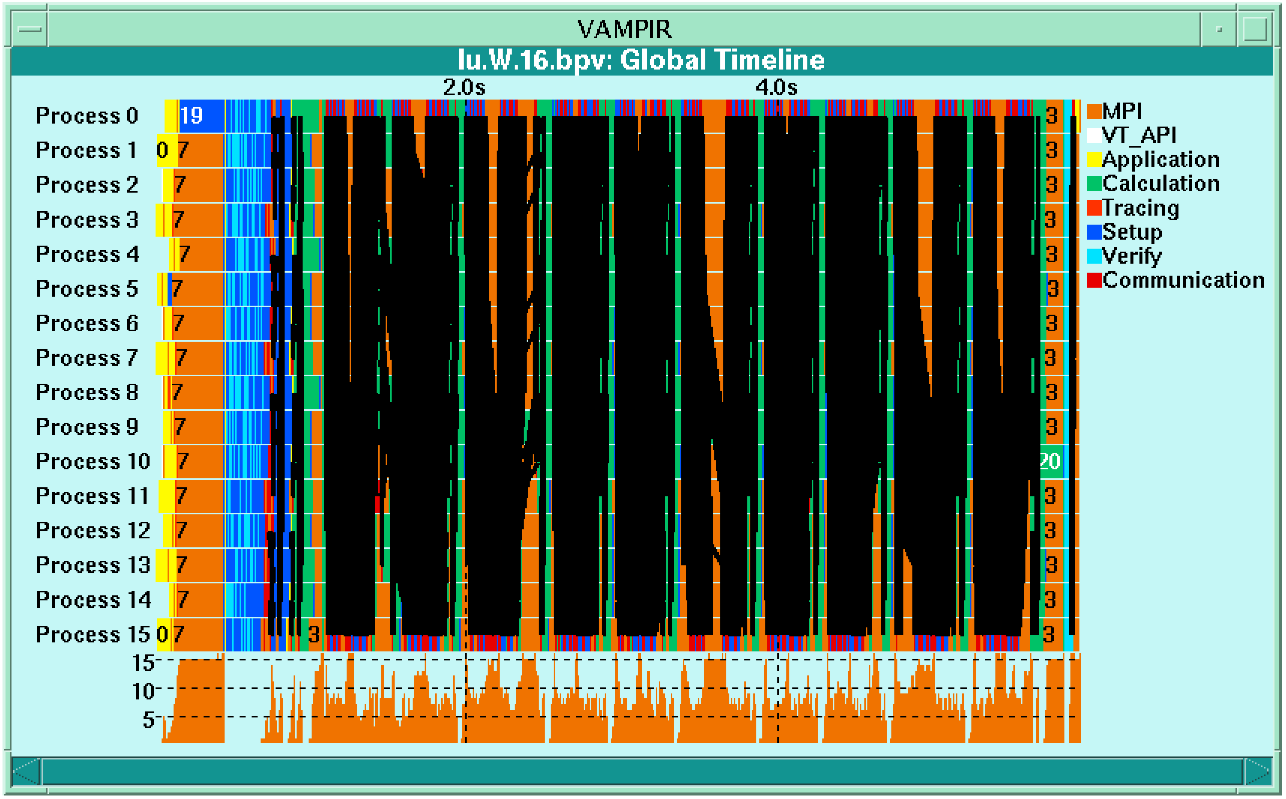
- Process displays show a single process per window
  - Activity Chart
  - Timeline
  - Calling Tree

Summary Chart



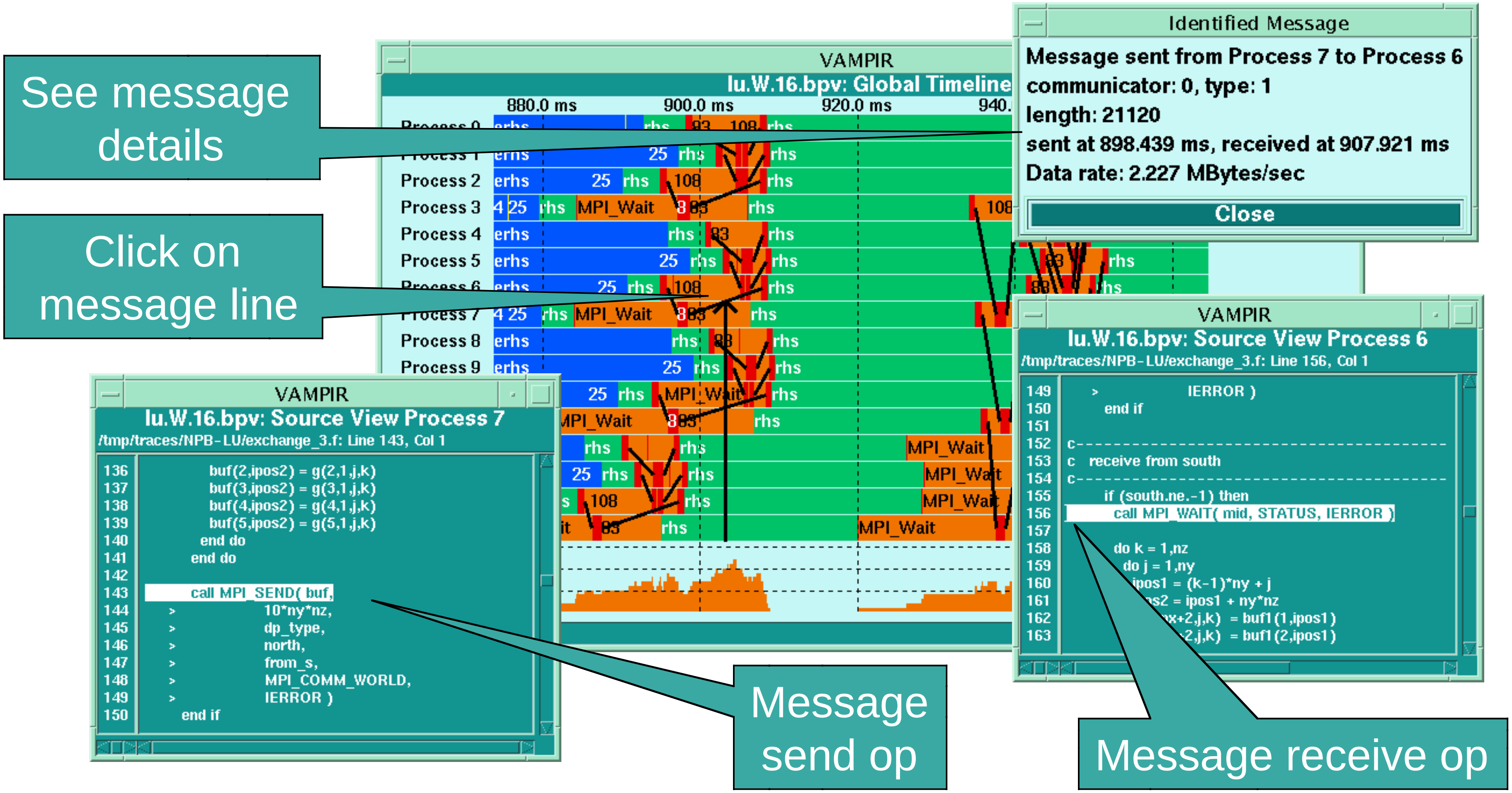
- Aggregated profiling information
  - execution time
  - number of calls
- Inclusive or exclusive of called routines
- Look at all/any category or all states
- Values can be exported/imported
- Tracefiles can be compared1

Timeline Display



- Now displays MPI collective and I/O operations
- To zoom, draw rectangle with the mouse
- Also used to select sub–intervals for statistics

Timeline Display (Message Info)



See message details

Click on message line

Message send op

Message receive op

Identified Message

Message sent from Process 7 to Process 6
communicator: 0, type: 1
length: 21120
sent at 898.439 ms, received at 907.921 ms
Data rate: 2.227 MBytes/sec

VAMPIR

lu.W.16.bpv: Source View Process 7

/tmp/traces/NPB-LU/exchange_3.f: Line 143, Col 1

```
136          buf(2,ipos2) = g(2,1,j,k)
137          buf(3,ipos2) = g(3,1,j,k)
138          buf(4,ipos2) = g(4,1,j,k)
139          buf(5,ipos2) = g(5,1,j,k)
140        end do
141      end do
142
143      call MPI_SEND( buf,
144   >                  10*ny*nz,
145   >                  dp_type,
146   >                  north,
147   >                  from_s,
148   >                  MPI_COMM_WORLD,
149   >                  IERROR )
150    end if
```

VAMPIR

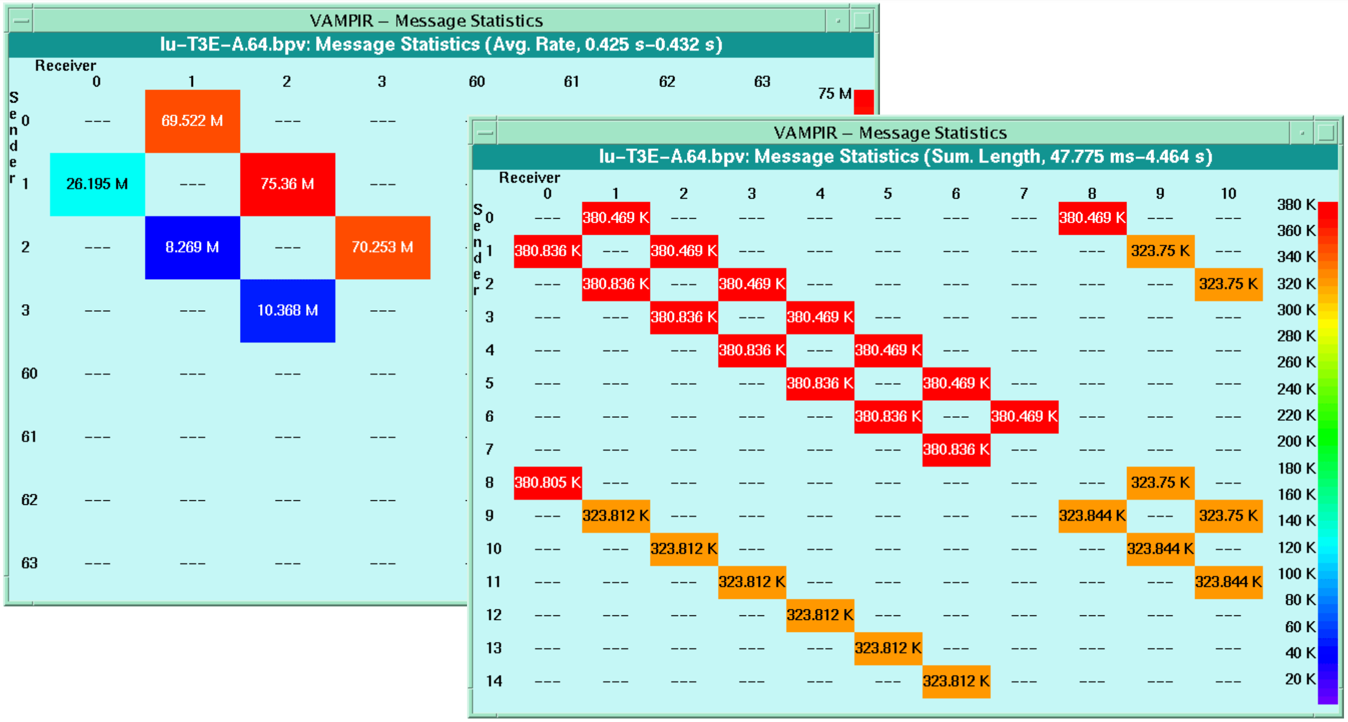
lu.W.16.bpv: Source View Process 6

/tmp/traces/NPB-LU/exchange_3.f: Line 156, Col 1

```
149   >              IERROR )
150     end if
151
152 c-------------------------------------------------------
153 c  receive from south
154 c-------------------------------------------------------
155     if (south.ne.-1) then
156       call MPI_WAIT( mid, STATUS, IERROR )
157
158       do k = 1,nz
159         do j = 1,ny
160           ipos1 = (k-1)*ny + j
161           os2 = ipos1 + ny*nz
162              nx+2,j,k)  = buf1(1,ipos1)
163              2,j,k)  = buf1(2,ipos1)
```

Source–code references are displayed if recorded by

Communication Statistics



VAMPIR – Message Statistics

lu-T3E-A.64.bpv: Message Statistics (Avg. Rate, 0.425 s-0.432 s)

Sender \ Receiver	0	1	2	3	60	61	62	63
0	---	69.522 M	---	---				
1	26.195 M	---	75.36 M	---				
2	---	8.269 M	---	70.253 M				
3	---	---	10.368 M	---				
60	---	---	---	---				
61	---	---	---	---				
62	---	---	---	---				
63	---	---	---	---				

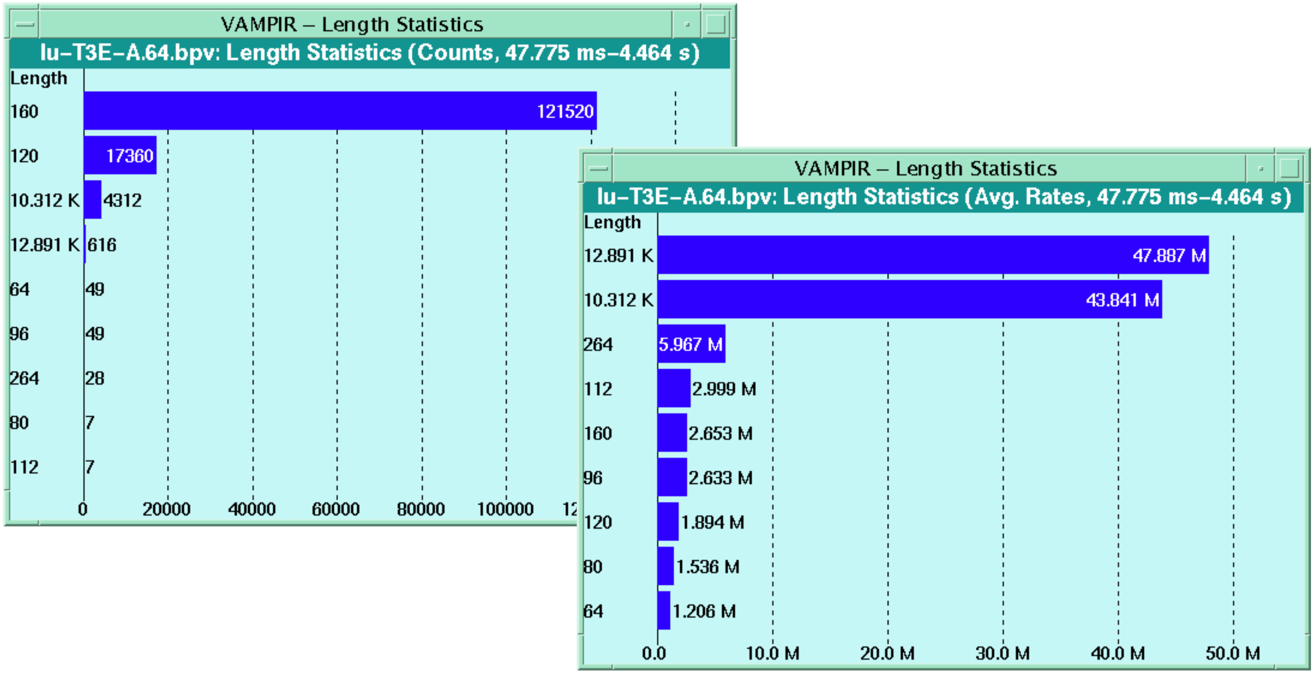
VAMPIR – Message Statistics

lu-T3E-A.64.bpv: Message Statistics (Sum. Length, 47.775 ms-4.464 s)

Sender \ Receiver	0	1	2	3	4	5	6	7	8	9	10
0	---	380.469 K	---	---	---	---	---	---	380.469 K	---	---
1	380.836 K	---	380.469 K	---	---	---	---	---	---	323.75 K	---
2	---	380.836 K	---	380.469 K	---	---	---	---	---	---	323.75 K
3	---	---	380.836 K	---	380.469 K	---	---	---	---	---	---
4	---	---	---	380.836 K	---	380.469 K	---	---	---	---	---
5	---	---	---	---	380.836 K	---	380.469 K	---	---	---	---
6	---	---	---	---	---	380.836 K	---	380.469 K	---	---	---
7	---	---	---	---	---	---	380.836 K	---	---	---	---
8	380.805 K	---	---	---	---	---	---	---	---	323.75 K	---
9	---	323.812 K	---	---	---	---	---	---	323.844 K	---	323.75 K
10	---	---	323.812 K	---	---	---	---	---	---	323.844 K	---
11	---	---	---	323.812 K	---	---	---	---	---	---	323.844 K
12	---	---	---	---	323.812 K	---	---	---	---	---	---
13	---	---	---	---	---	323.812 K	---	---	---	---	---
14	---	---	---	---	---	---	323.812 K	---	---	---	---

- Message statistics for each process pair:
  - Byte and message count
  - min/max/avg message length
  - min/max/avg bandwidth
- Filter for message tags or communicators

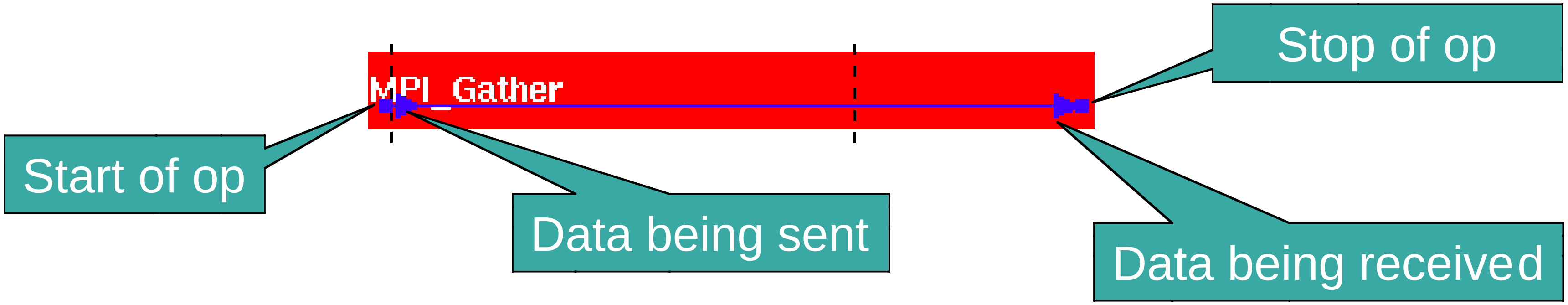
Message Histograms



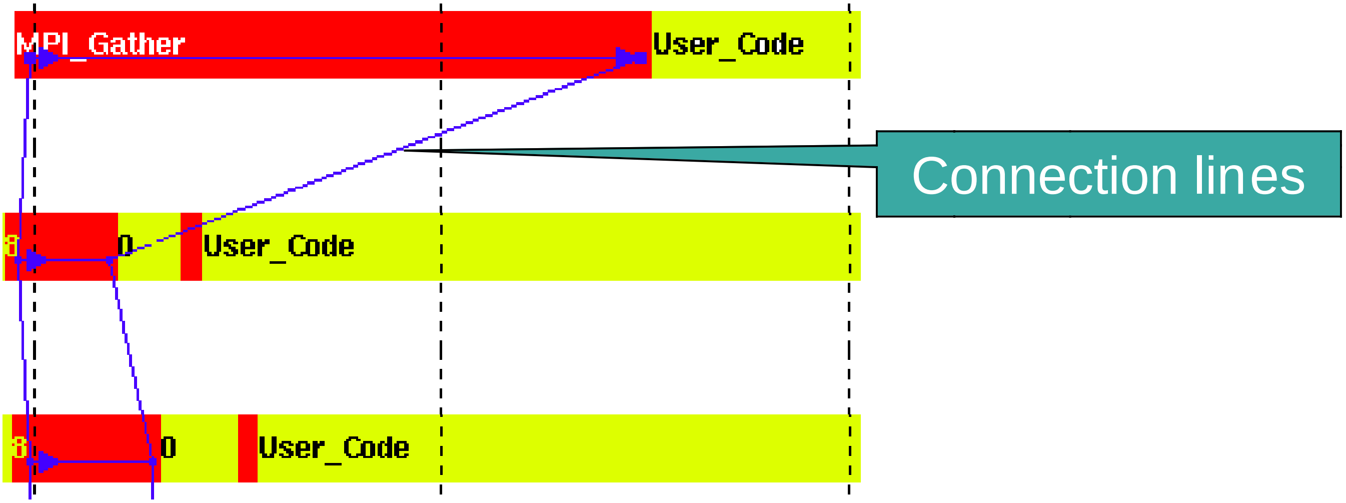
- Message statistics by length, tag or communicator
  - Byte and message count
  - min/max/avg bandwidth
- Filter for message tags or communicators

Collective Operations

- For each process: mark operation locally

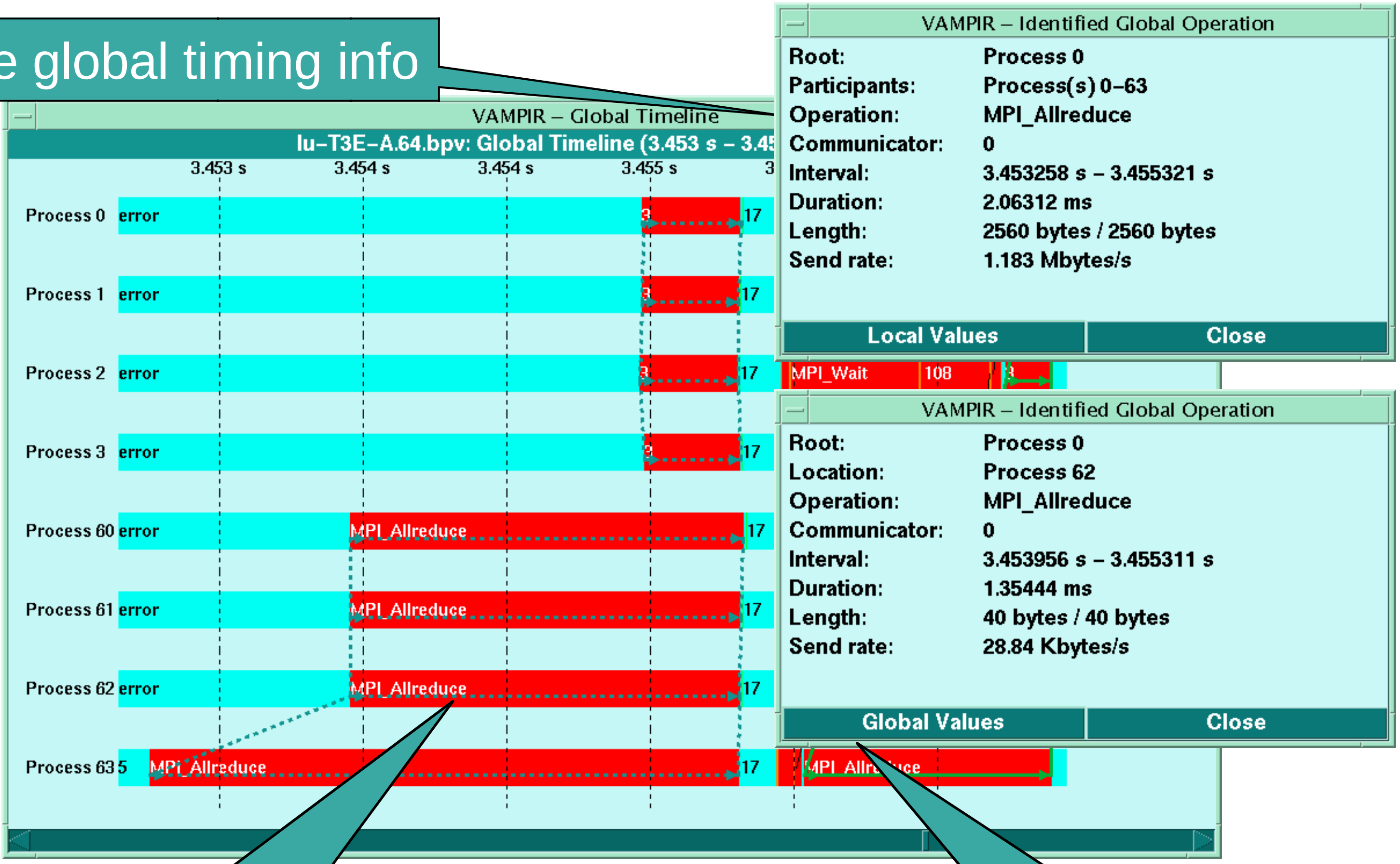


- Connect start/stop points by lines



Collective Operations

See global timing info



VAMPIR – Identified Global Operation

Root: Process 0
Participants: Process(s) 0–63
Operation: MPI_Allreduce
Communicator: 0
Interval: 3.453258 s – 3.455321 s
Duration: 2.06312 ms
Length: 2560 bytes / 2560 bytes
Send rate: 1.183 Mbytes/s

VAMPIR – Identified Global Operation

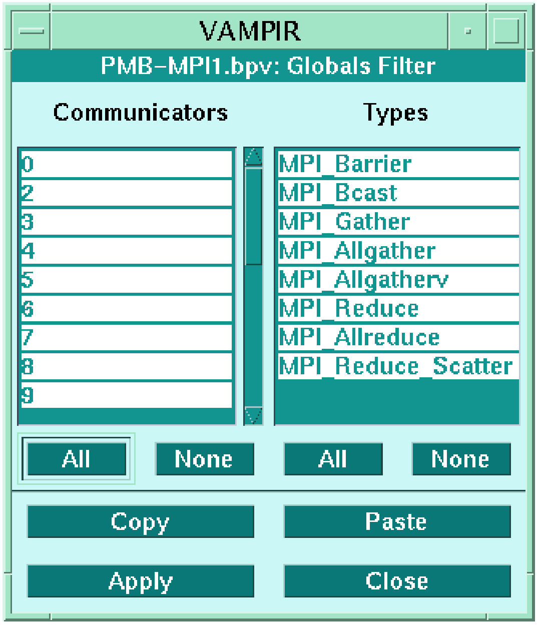
Root: Process 0
Location: Process 62
Operation: MPI_Allreduce
Communicator: 0
Interval: 3.453956 s – 3.455311 s
Duration: 1.35444 ms
Length: 40 bytes / 40 bytes
Send rate: 28.84 Kbytes/s

Click on collective operation display

See local timing info

Collective Operations

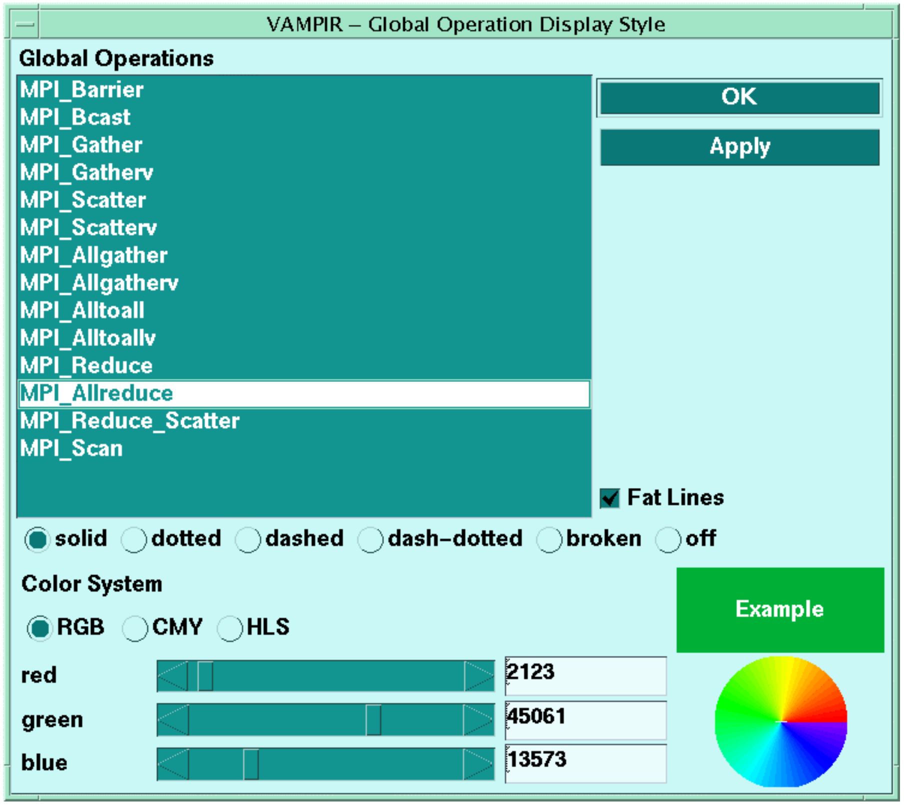
- Collective operations can be filtered



VAMPIR

PMB-MPI1.bpv: Globals Filter

Communicators	Types
0	MPI_Barrier
2	MPI_Bcast
3	MPI_Gather
4	MPI_Allgather
5	MPI_Allgatherv
6	MPI_Reduce
7	MPI_Allreduce
8	MPI_Reduce_Scatter
9	



VAMPIR – Global Operation Display Style

Global Operations

MPI_Barrier
MPI_Bcast
MPI_Gather
MPI_Gatherv
MPI_Scatter
MPI_Scatterv
MPI_Allgather
MPI_Allgatherv
MPI_Alltoall
MPI_Alltoallv
MPI_Reduce
MPI_Allreduce
MPI_Reduce_Scatter
MPI_Scan

☑ Fat Lines

● solid ○ dotted ○ dashed ○ dash-dotted ○ broken ○ off

Color System

● RGB ○ CMY ○ HLS

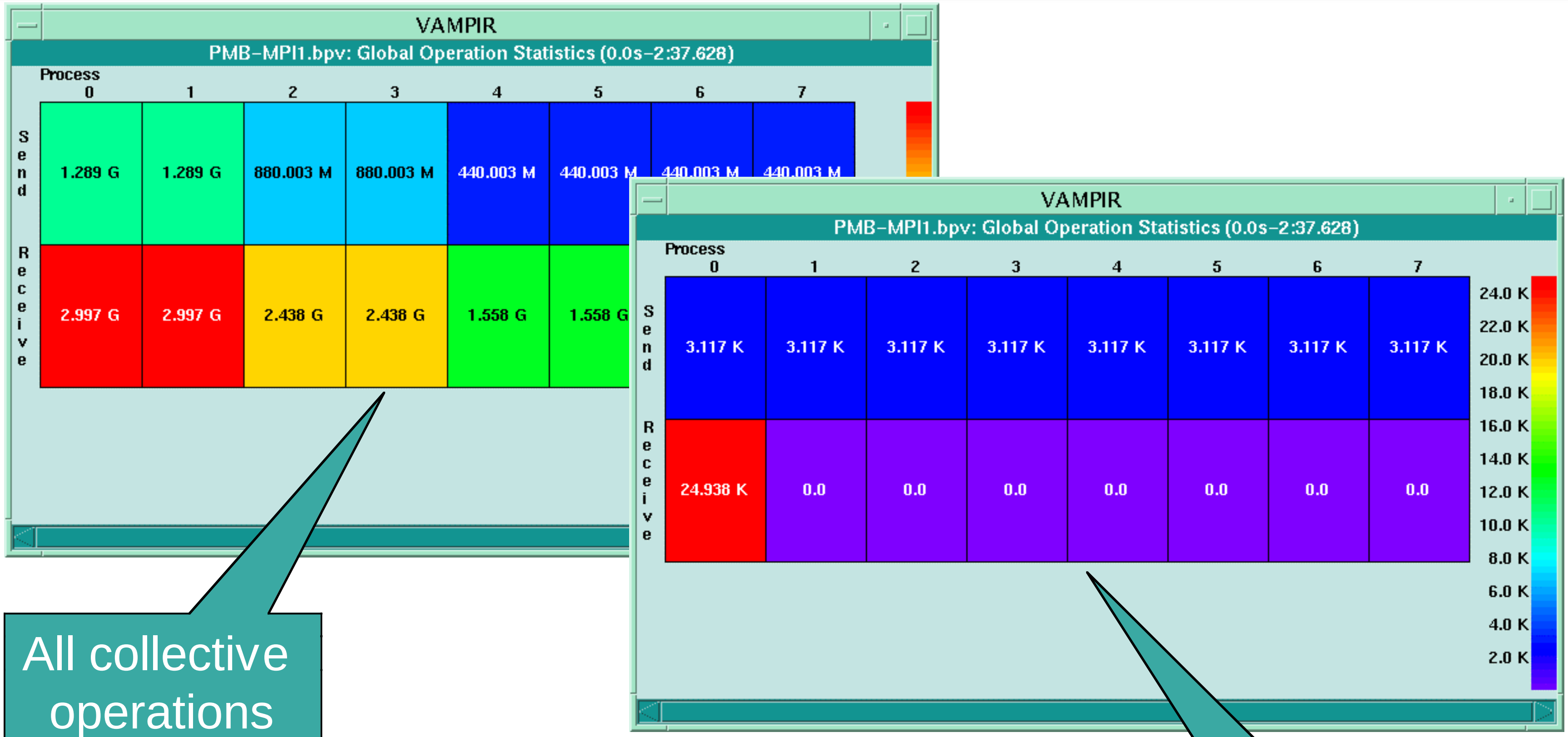
red 2123

green 45061

blue 13573

- The display style can be adapted for each collective operation

Global Communication Statistics



VAMPIR

PMB-MPI1.bpv: Global Operation Statistics (0.0s-2:37.628)

Process	0	1	2	3	4	5	6	7
Send	1.289 G	1.289 G	880.003 M	880.003 M	440.003 M	440.003 M	440.003 M	440.003 M
Receive	2.997 G	2.997 G	2.438 G	2.438 G	1.558 G	1.558 G		

VAMPIR

PMB-MPI1.bpv: Global Operation Statistics (0.0s-2:37.628)

Process	0	1	2	3	4	5	6	7
Send	3.117 K	3.117 K	3.117 K	3.117 K	3.117 K	3.117 K	3.117 K	3.117 K
Receive	24.938 K	0.0	0.0	0.0	0.0	0.0	0.0	0.0

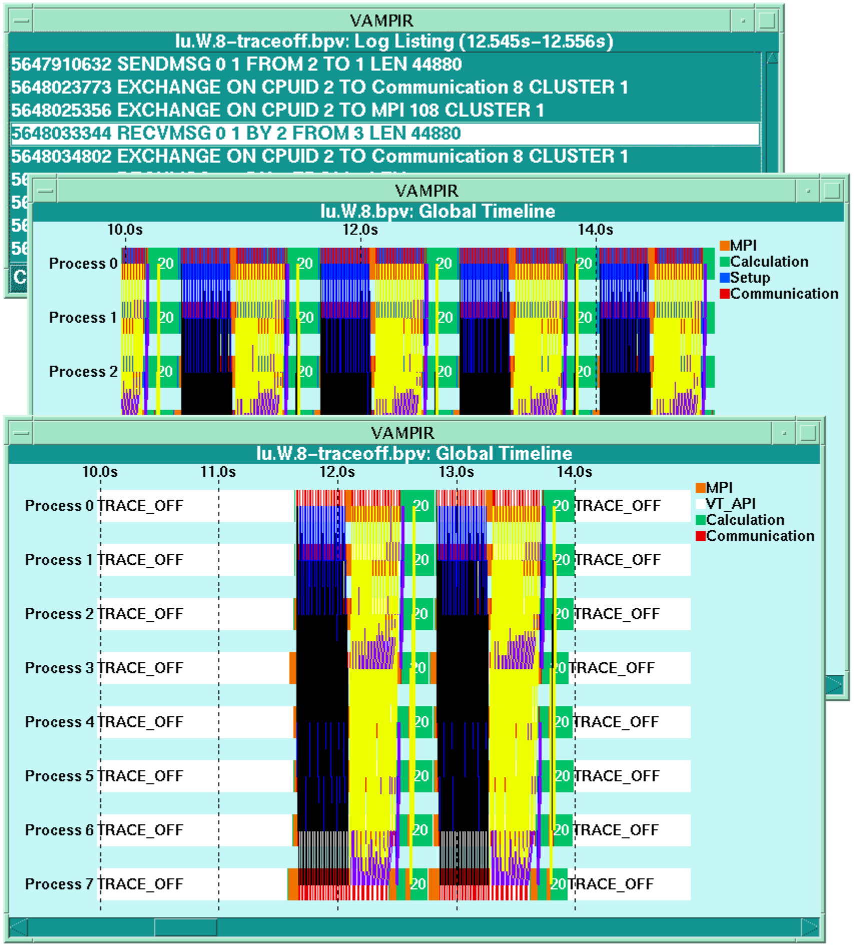
All collective operations

MPI_Gather only

- Statistics for collective operations:
  - operation counts, Bytes sent/received
  - transmission rates
- Filter for collective operation

Vampirtrace

Tracing of
MPI and
Application
Events



```
5647910632 SENDMSG 0 1 FROM 2 TO 1 LEN 44880
5648023773 EXCHANGE ON CPUID 2 TO Communication 8 CLUSTER 1
5648025356 EXCHANGE ON CPUID 2 TO MPI 108 CLUSTER 1
5648033344 RECVMSG 0 1 BY 2 FROM 3 LEN 44880
5648034802 EXCHANGE ON CPUID 2 TO Communication 8 CLUSTER 1
```

Vampirtrace

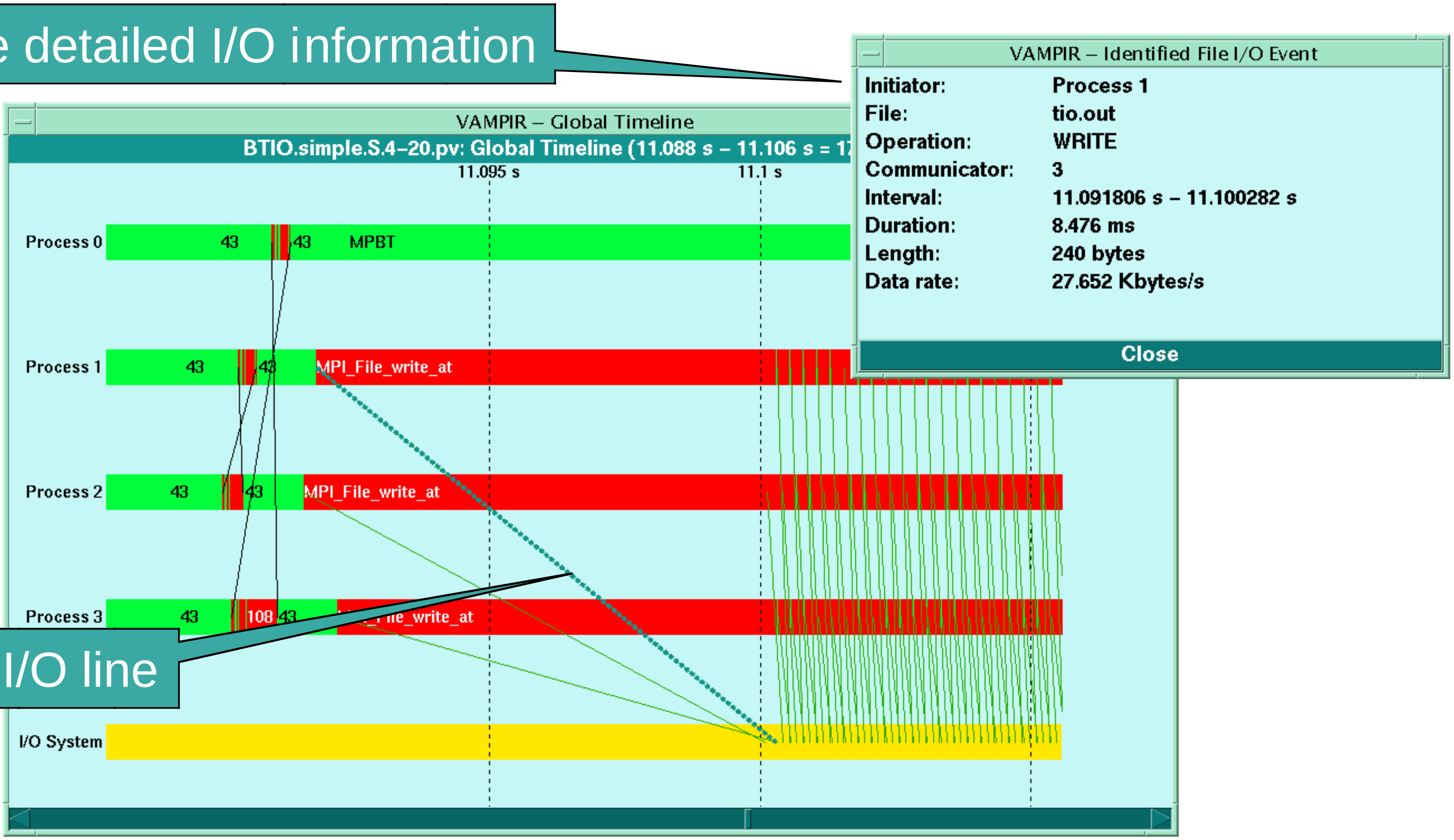
- New version: Vampirtrace 2.0

- Significant new features:
  - records collective communication
  - enhanced filter functions
  - extended API
  - records source–code information (selected platforms)
  - support for shmem (Cray T3E)
  - records MPI–2 I/O operations

- Available for all major MPI platforms

MPI–I/O Operations

See detailed I/O information



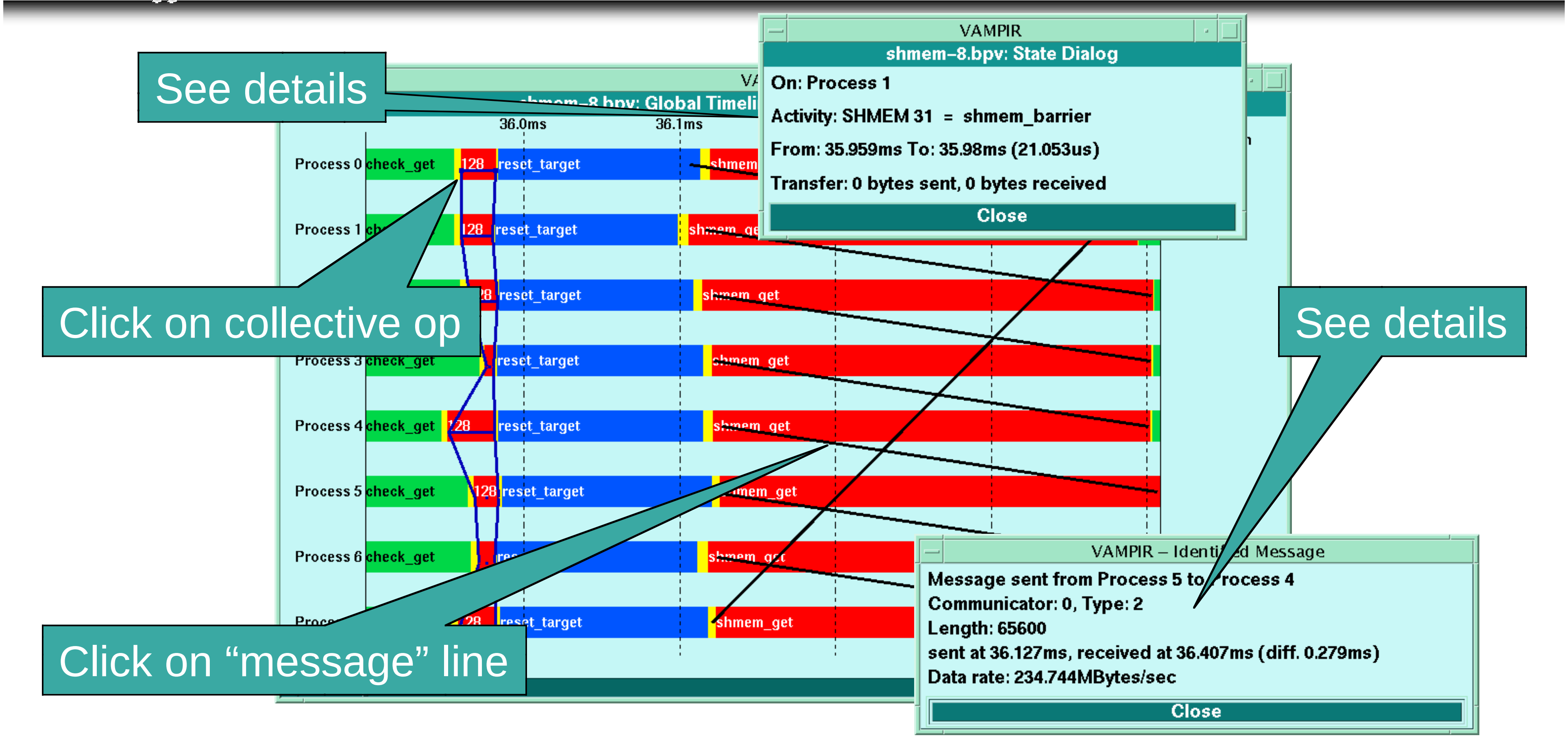
VAMPIR – Identified File I/O Event

Initiator: Process 1
File: tio.out
Operation: WRITE
Communicator: 3
Interval: 11.091806 s – 11.100282 s
Duration: 8.476 ms
Length: 240 bytes
Data rate: 27.652 Kbytes/s

Click on I/O line

- I/O transfers are shown as lines

shmem Operations



VAMPIR

shmem–8.bpv: State Dialog

On: Process 1

Activity: SHMEM 31 = shmem_barrier

From: 35.959ms To: 35.98ms (21.053us)

Transfer: 0 bytes sent, 0 bytes received

Close

VAMPIR – Identified Message

Message sent from Process 5 to Process 4

Communicator: 0, Type: 2

Length: 65600

sent at 36.127ms, received at 36.407ms (diff. 0.279ms)

Data rate: 234.744MBytes/sec

Close

- Display one–sided transfers as messages
- Display shmem global operations

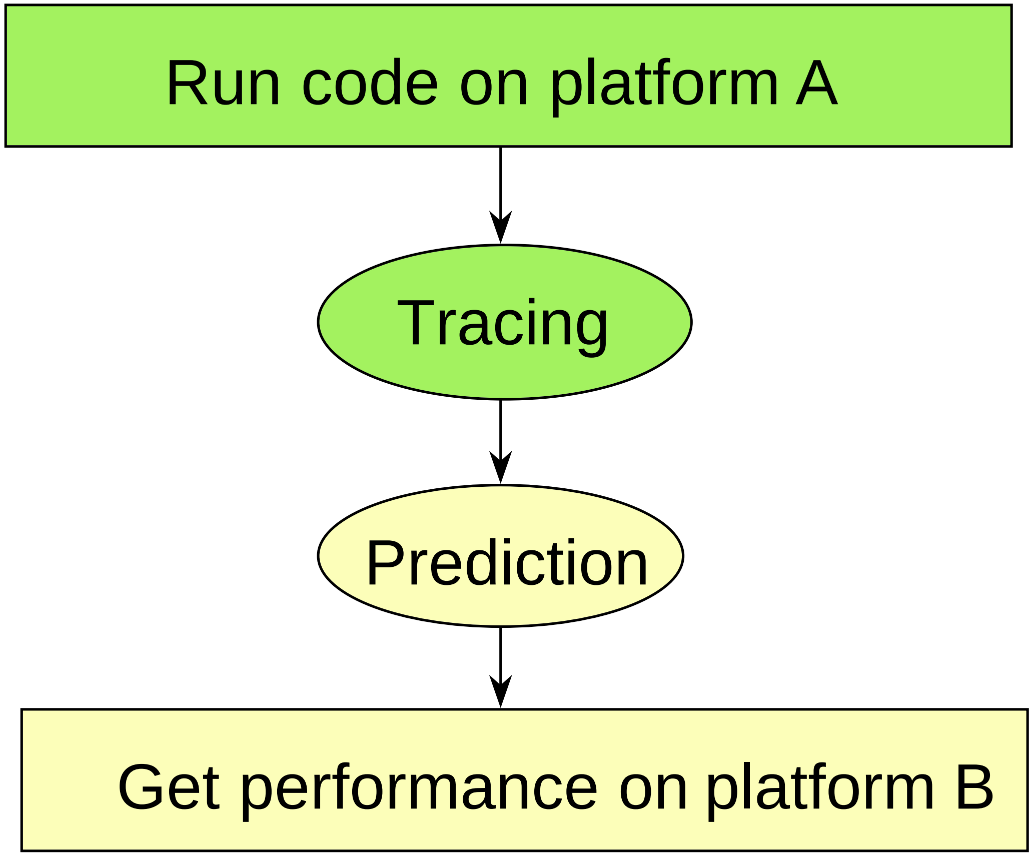
Vampir Track Record

- Reference customers: ARL, ARSC, CEWES, LANL, LLNL, MHPCC, NASA, NERSC, NSA, Cornell TC, Oregon Univ., CEA, DWD, ECMWF, GMD, HLRS, LRZ, PC^2, RUKA, ...

- URLs:
  - www.tc.cornell.edu/Edu/Tutor/Vampir
  - www.llnl.gov/sccd/lc/DEG/vampir/vampir.html
  - www.uni-karlsruhe.de/~Vampir
  - www.lrz-muenchen.de/services/software/parallel/vampir
  - www.hlrs.de/structure/support/parallel_computing/tools/performance/vampir.html

Dimemas

Performance
Prediction
Made
Easy

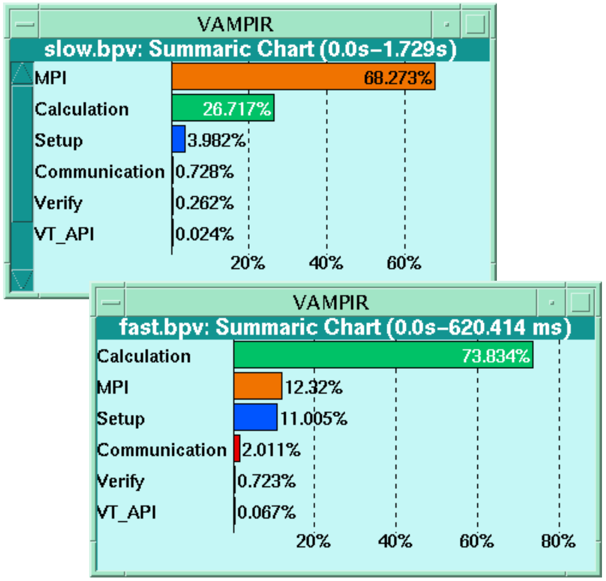


Dimemas Outputs – Vampir Tracefiles

- Actual program run

vs.

- Ideal communication



VAMPIR

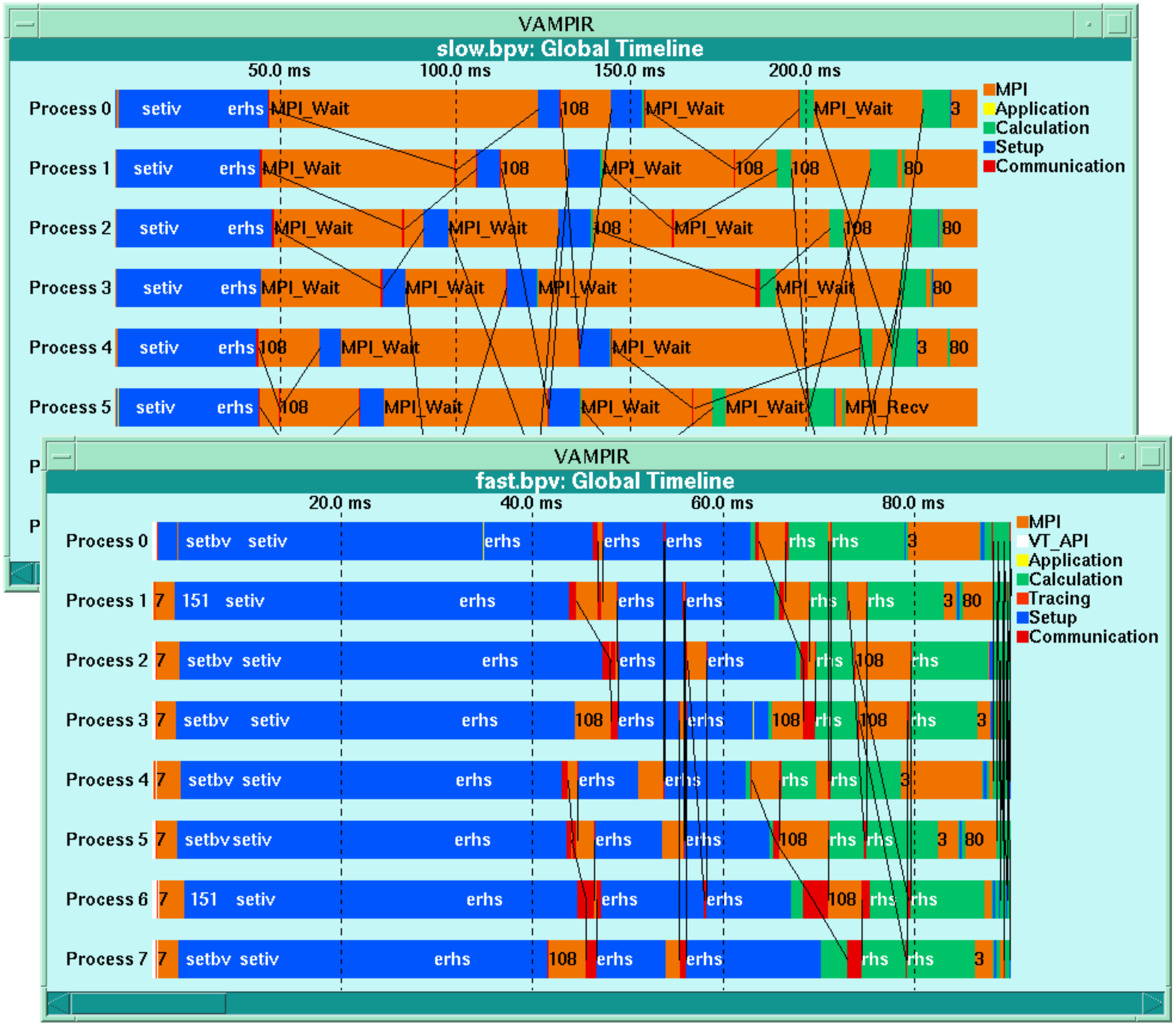
slow.bpv: Summaric Chart (0.0s–1.729s)

MPI	68.273%
Calculation	26.717%
Setup	3.982%
Communication	0.728%
Verify	0.262%
VT_API	0.024%

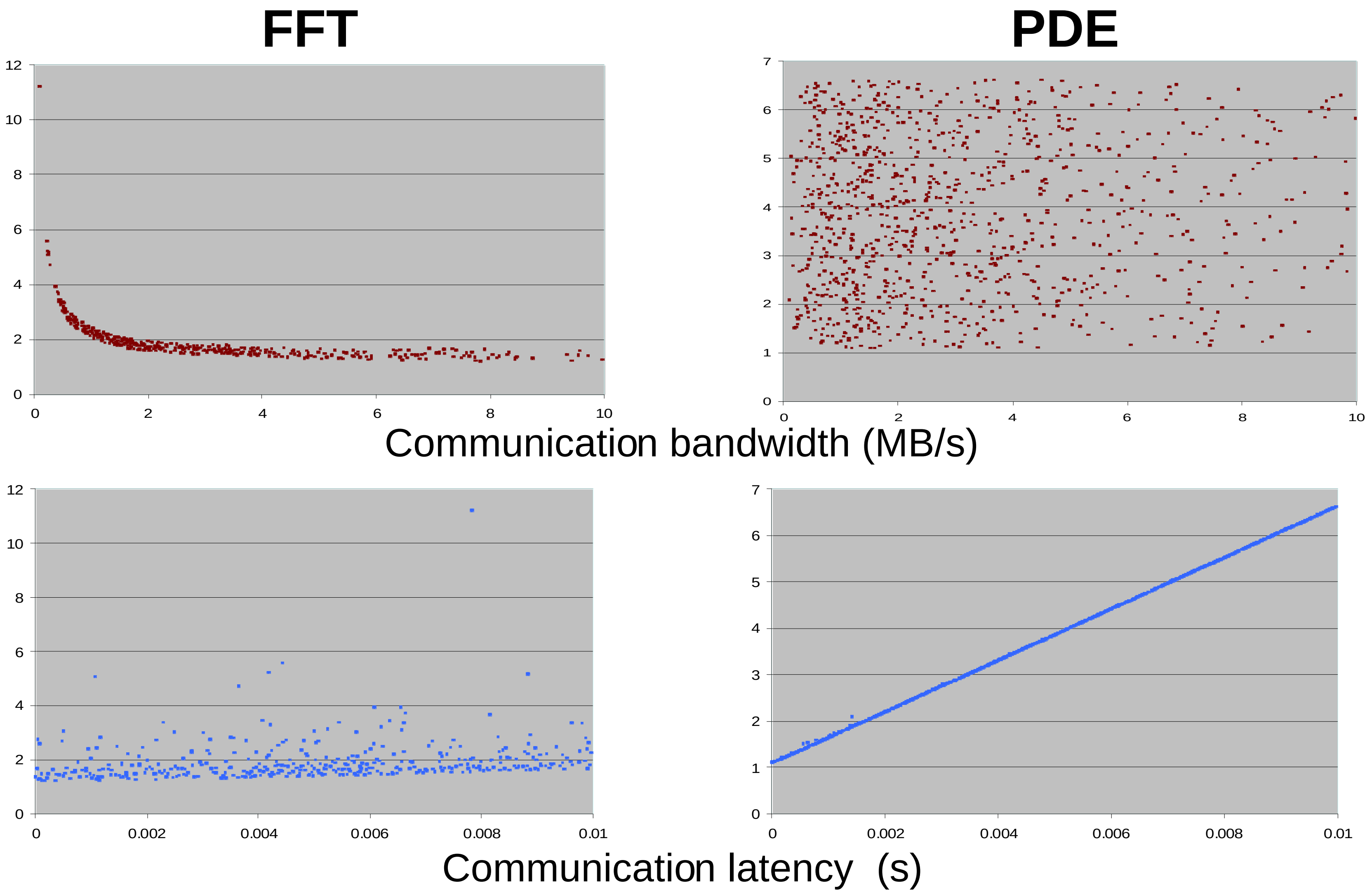
VAMPIR

fast.bpv: Summaric Chart (0.0s–620.414 ms)

Calculation	73.834%
MPI	12.32%
Setup	11.005%
Communication	2.011%
Verify	0.723%
VT_API	0.067%



Dimemas - Prediction studies



Example Results – Performance Prediction

- NAS LU benchmarks

Benchmark	Class	Time	Simulated	
BT	*A*	316.580000	319.858708	1.04%
BT	*W*	6.410000	5.548973	13.43%
CG	*A*	4.480000	4.210193	6.02%
CG	*B*	342.860000	335.641787	2.11%
CG	*W*	1.130000	1.210509	7.12%
EP	*A*	18.970000	18.843001	0.67%
EP	*B*	75.990000	77.122498	1.49%
EP	*W*	2.380000	2.391074	0.47%
LU	*A*	131.660000	129.139735	1.91%
LU	*B*	793.070000	785.966349	0.90%
LU	*W*	16.550000	17.415359	5.23%
MG	*A*	16.850000	16.290057	3.32%
MG	*B*	59.480000	64.974789	9.24%
MG	*W*	1.052000	1.112022	5.71%
SP	*A*	152.020000	151.821937	0.13%
SP	*W*	16.950000	15.237959	10.10%

Future plans - Vampir, Dimemas

- Towards automatic performance analysis
  - improve user guidance in Vampir and Dimemas
  - add “assistant” module for inexperienced users
- Support for clustered shared–memory systems
  - support shared–memory programming models (threads, OpenMP)
  - expose cluster structure
  - aggregate information on SMP nodes
- Support for (very) large systems
  - new structured tracefile format
  - fine–grain interactive control over tracing
  - scalable displays
  - new Vampir structure (can exploit parallelism)

Access to Pallas Tools

Download free evaluation copies

http://www.pallas.com

Thanks for your attention!

Pallas GmbH
Hermülheimer Straße 10
D–50321 Brühl,
Germany

info@pallas.com
www.pallas.com