TECHNISCHE
UNIVERSITÄT
DRESDEN

Comprehensive Lustre I/O Tracing with Vampir

Lustre User Group 2010

Zellescher Weg 12

WIL A 208

Tel. +49 351 - 463 – 34217

Michael Kluge (michael.kluge@tu-dresden.de)

ZIH
Zentrum für Informationsdienste
und Hochleistungsrechnen

Content

- Vampir Introduction
- VampirTrace introduction
- Enhancing I/O analysis
- Creation of a comprehensive system view
- Illustrating Examples

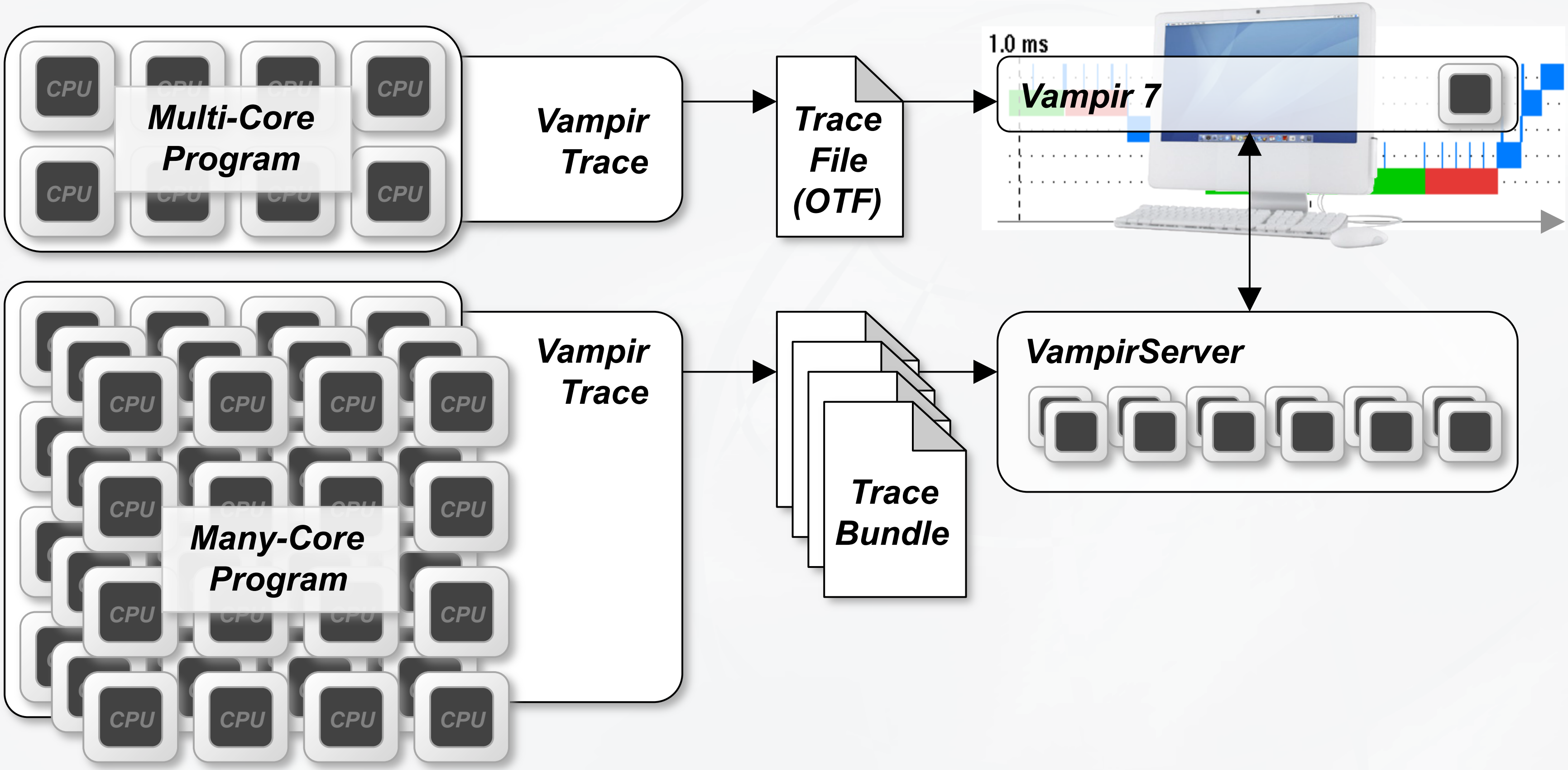
Vampir Introduction

- Visualization of dynamics of complex parallel processes
- Requires two components
  - Monitor/Collector (VampirTrace)
  - Charts/Browser (Vampir)
- Available for major platforms
- Open Source (partially)

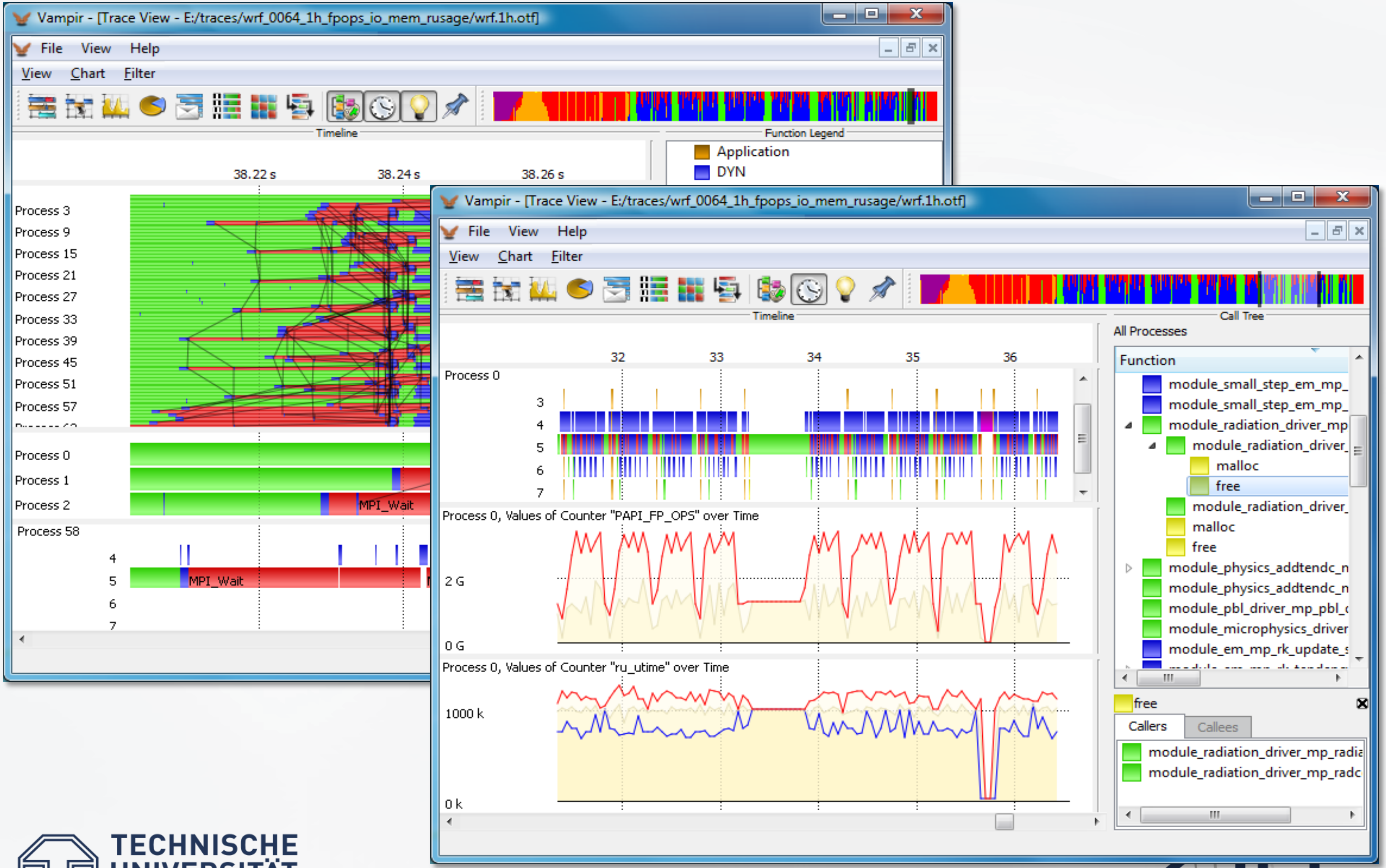
http://www.vampir.eu

http://www.tu-dresden.de/zih/vampirtrace

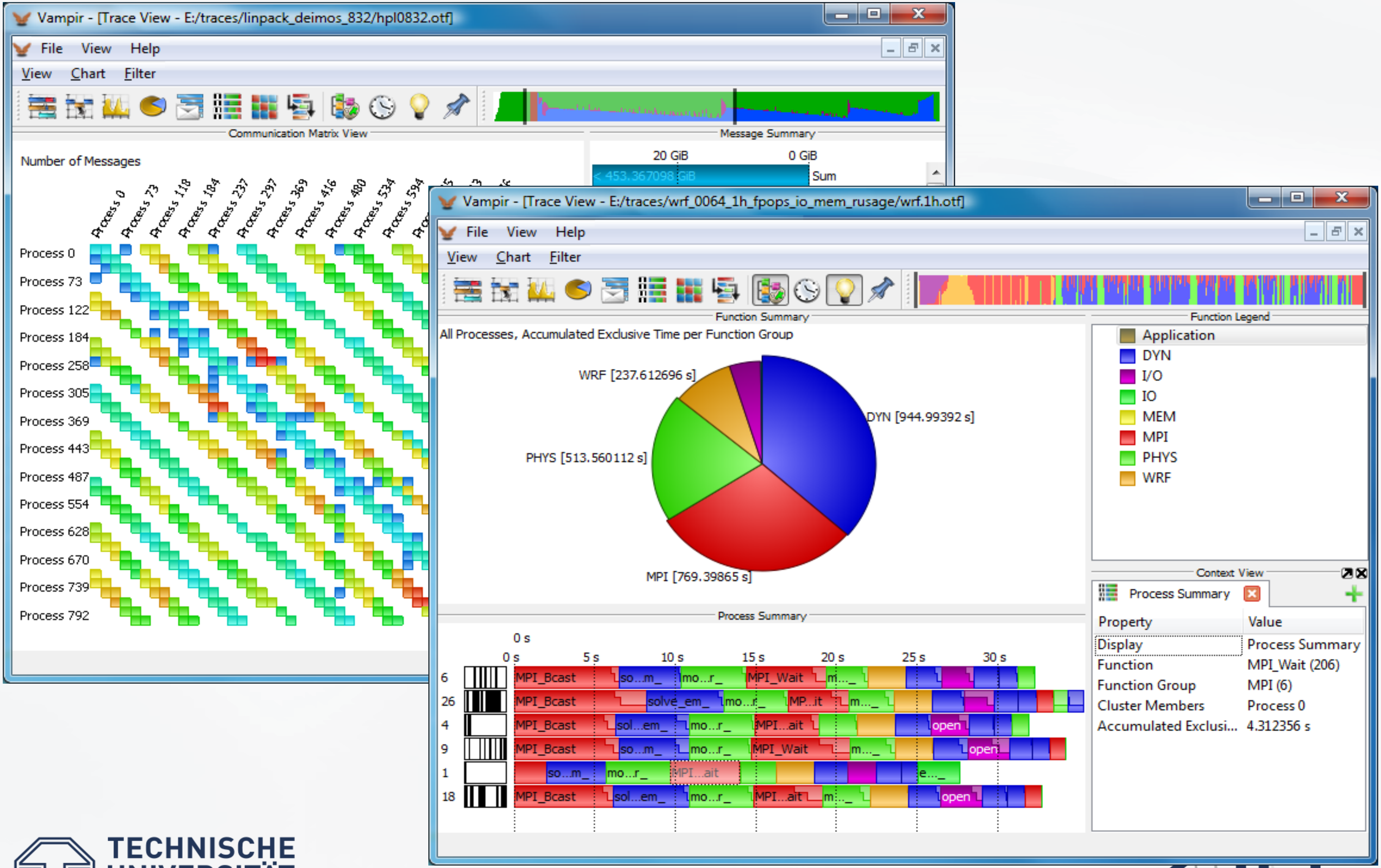
Vampir Toolset Architecture



Vampir 7: Timelines



Vampir 7: Summaries



Property	Value
Display	Process Summary
Function	MPI_Wait (206)
Function Group	MPI (6)
Cluster Members	Process 0
Accumulated Exclusi...	4.312356 s

I/O in HPC Environments

- Thousands of clients connected to a file server farm
- Dedicated infrastructure, large number of supporting components
- File systems often accessed through high level libraries
- File systems typically tuned for large I/O requests
- No backup

- Beside the 'large&fast' file system HPC systems have
  - slow home directory
  - even more slow archive space

- Multi cluster file systems

I/O Subsystem Sizes

- TU Dresden HRSK System (installed 2006):
  - 2*68 TB: 12 GB/s, >1300 disks, 12 file servers, 48 FC cables
  - 700+ nodes

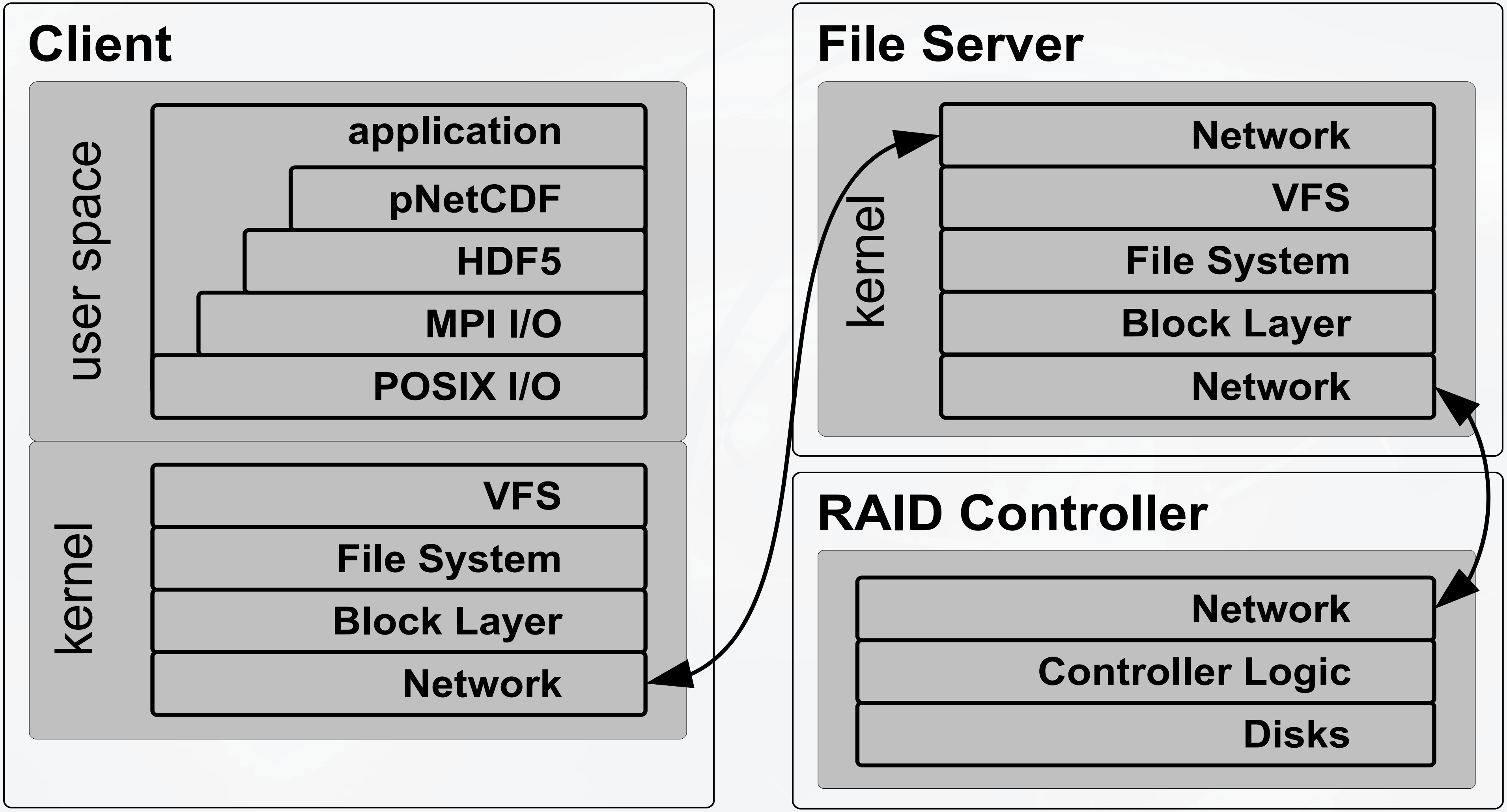
- DARPA HPCS Project (targeted 2010):
  - High Productivity Computing with millions of components
  - 50.000 spinning disks, 30.000 nodes, 100 PB file systems
  - millions of cores, 10 billion files per directory , 1 PB single file size
  - 40,000 file creates/sec from a single client node
  - 30 GB/sec single client, 1.5 TB/sec total bandwidth
  - 1 PB single file size
  - end-to-end data integrity

How to optimize I/O?

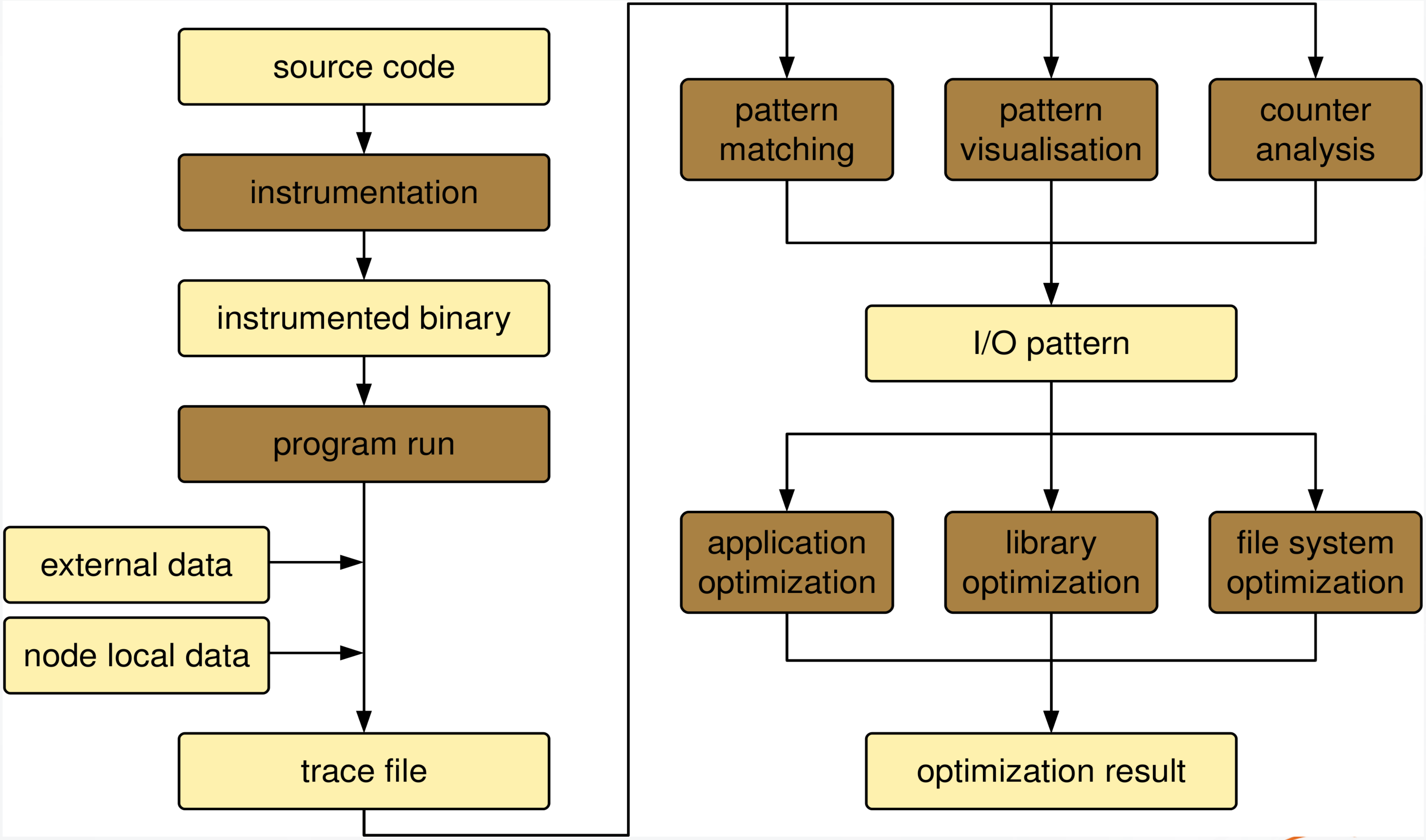
- Instrument all available data sources
- Make tools usable for production mode
- Create a global view onto the system
- Instrument the users application
- Run the user code
- Correlate and analyze the collected data
- Find an optimization strategy
- Decide about the layer to implement it
- Implement and test again

- Check if you killed someone else‘s performance

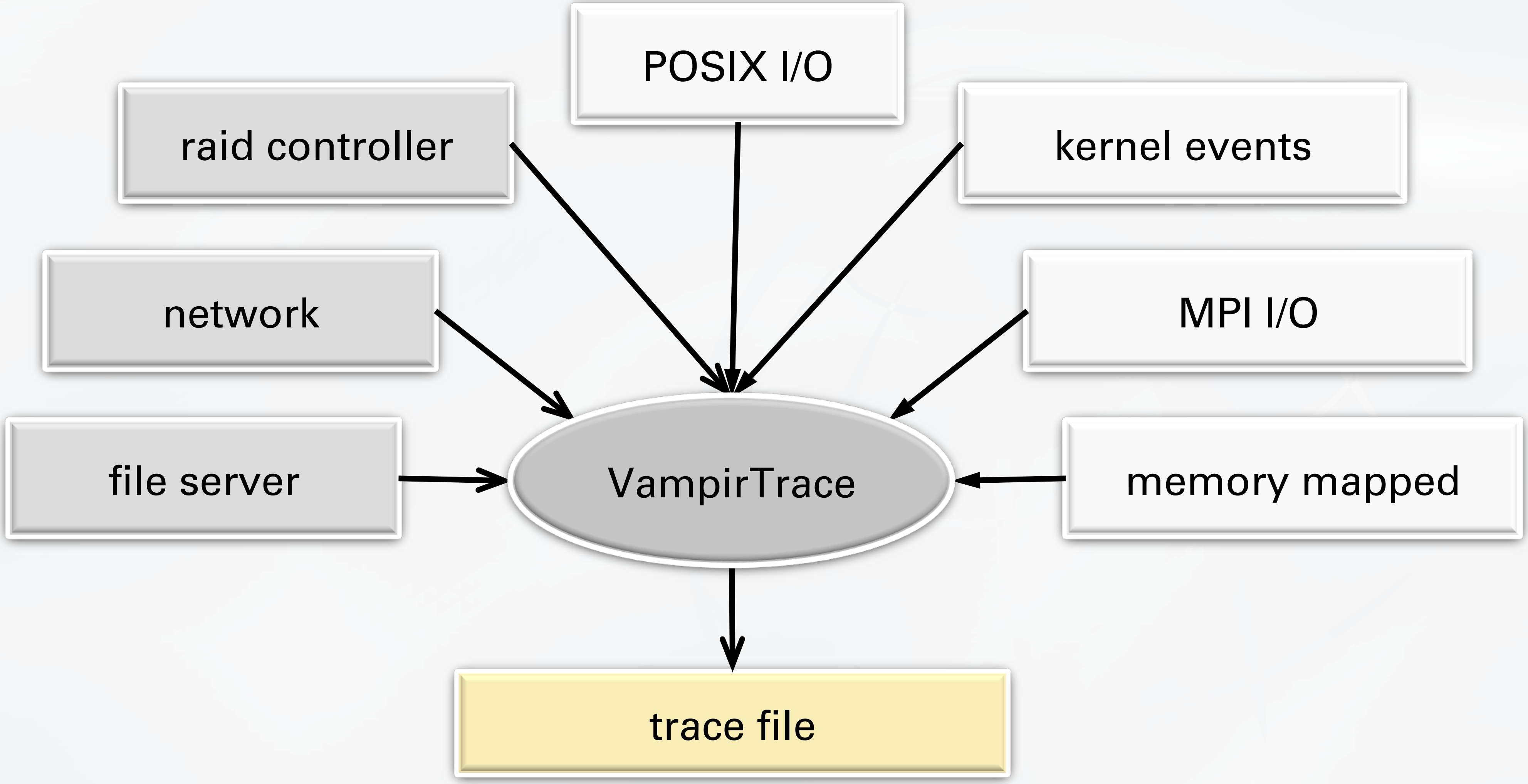
I/O Stack in HPC Environments



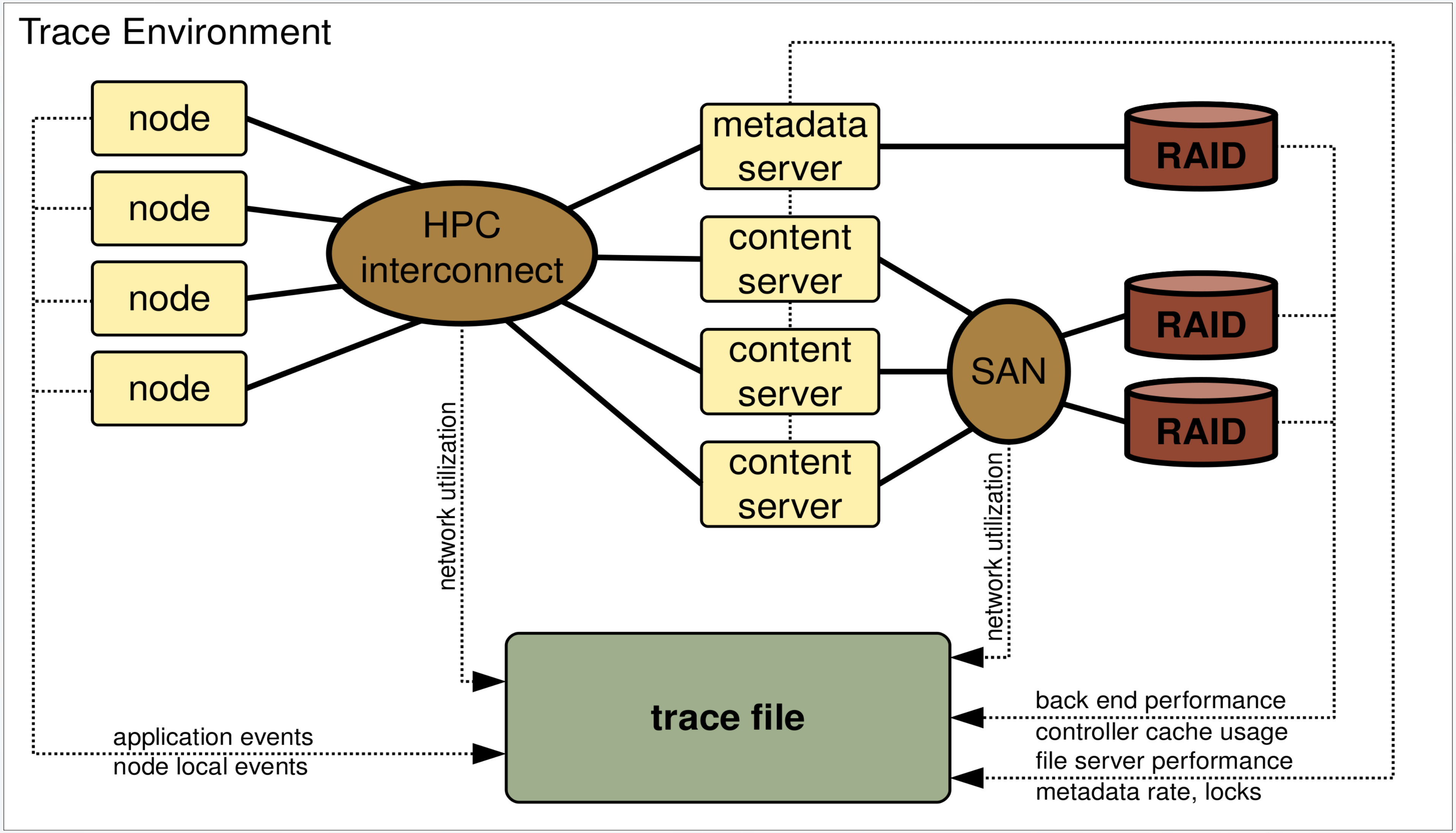
I/O Analysis Workflow



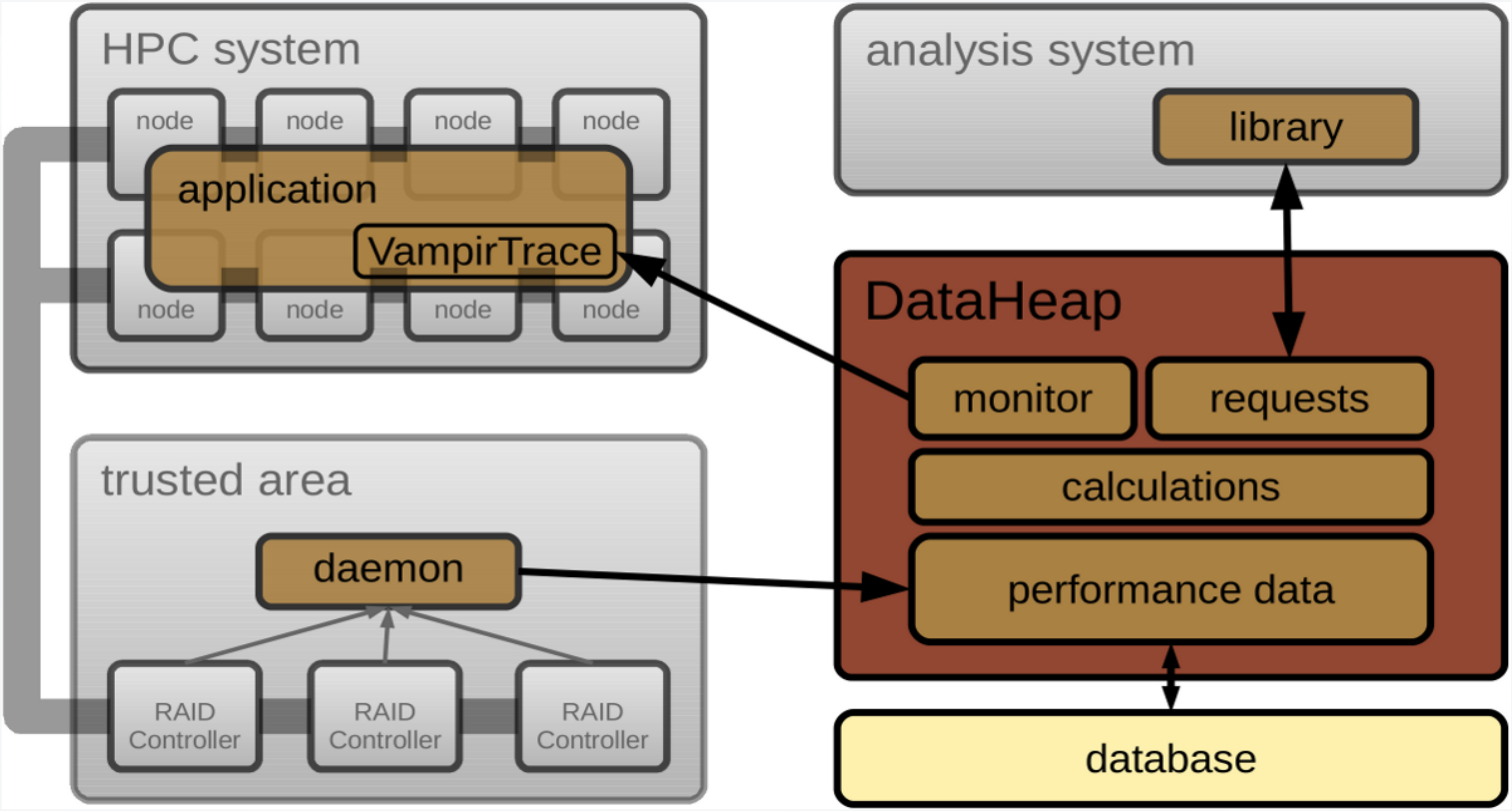
I/O Analysis Support in VampirTrace



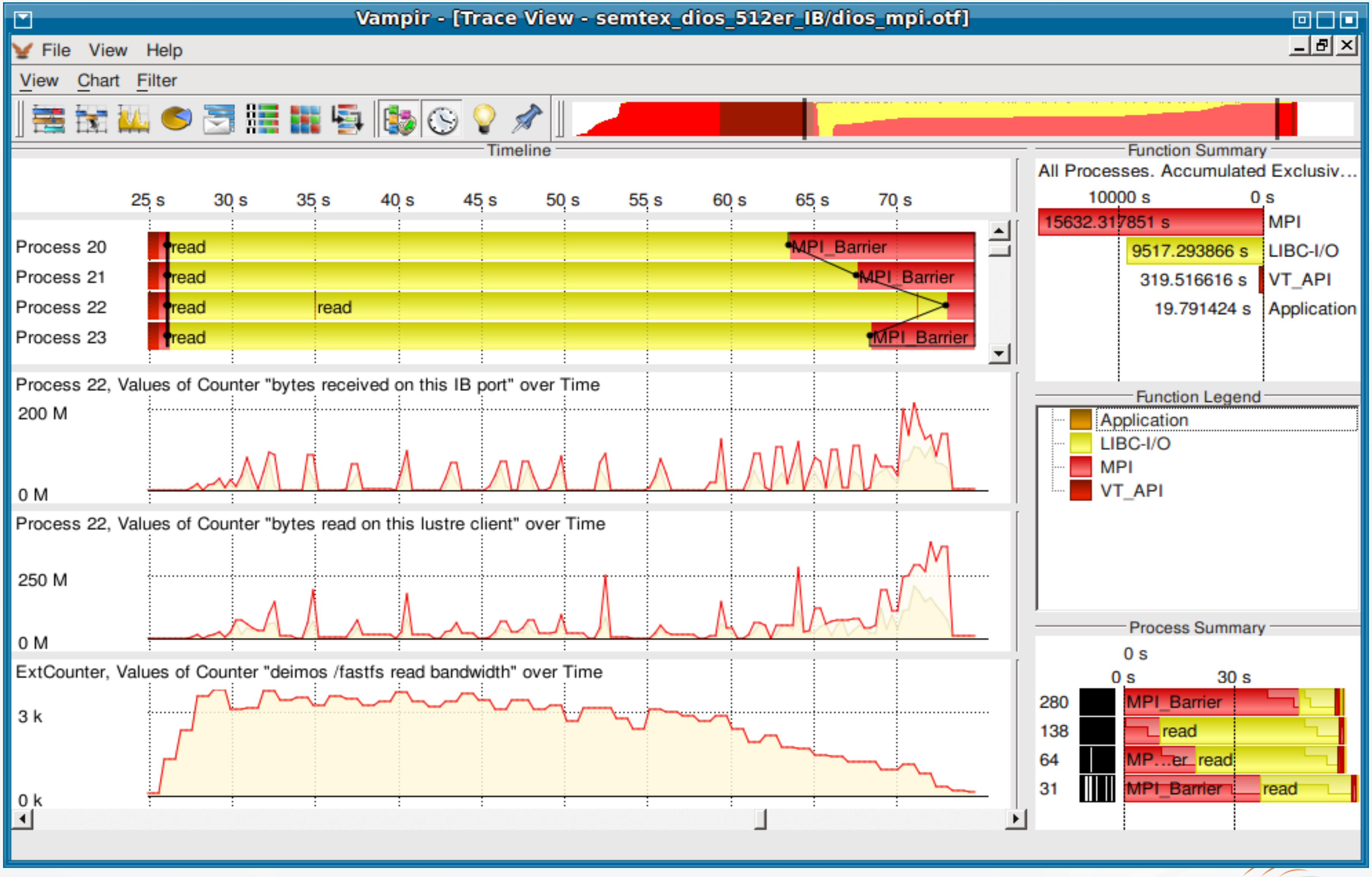
Instrumented System



External Tool Architecture

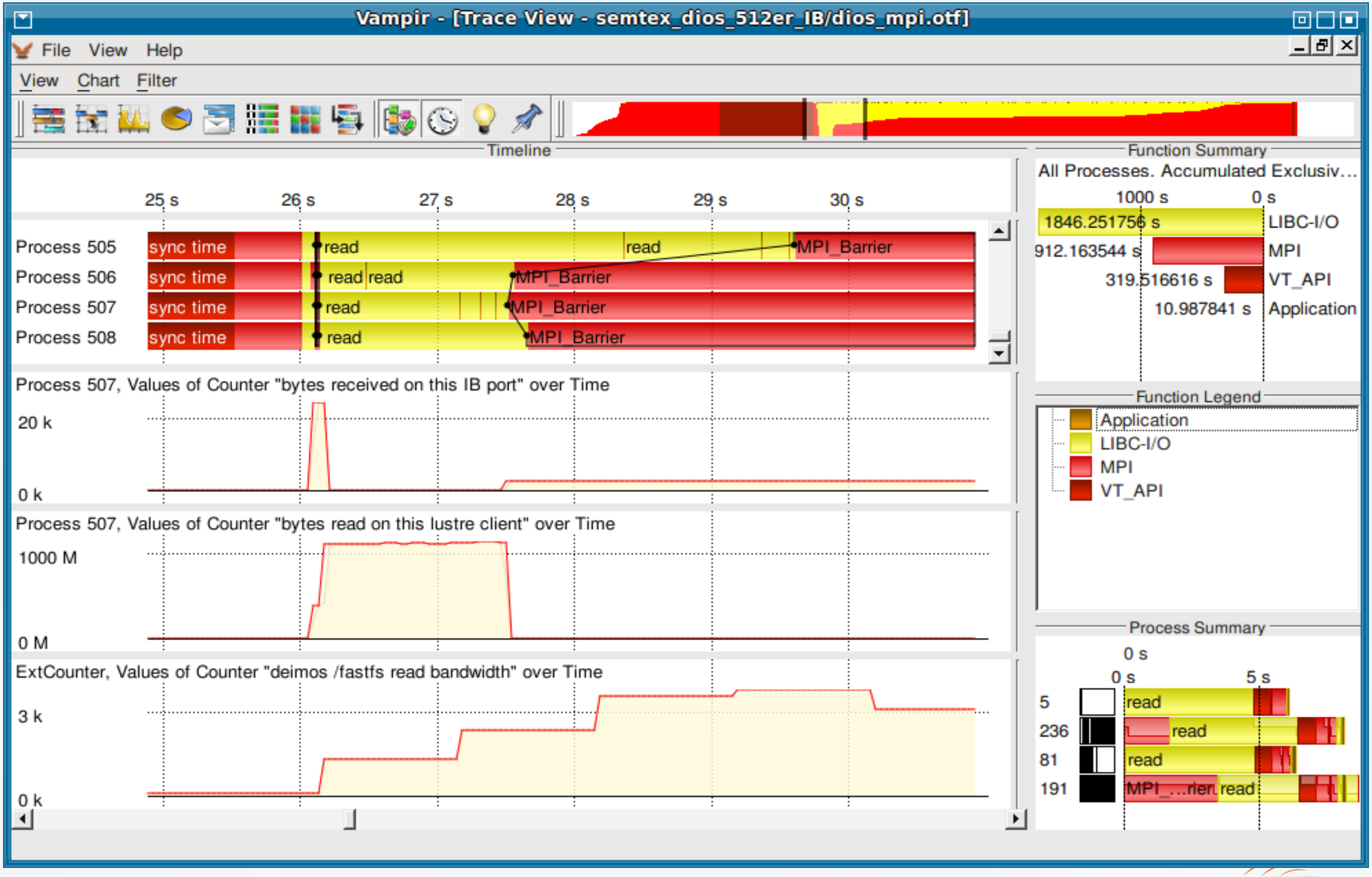


Example – Local and External Counter Values

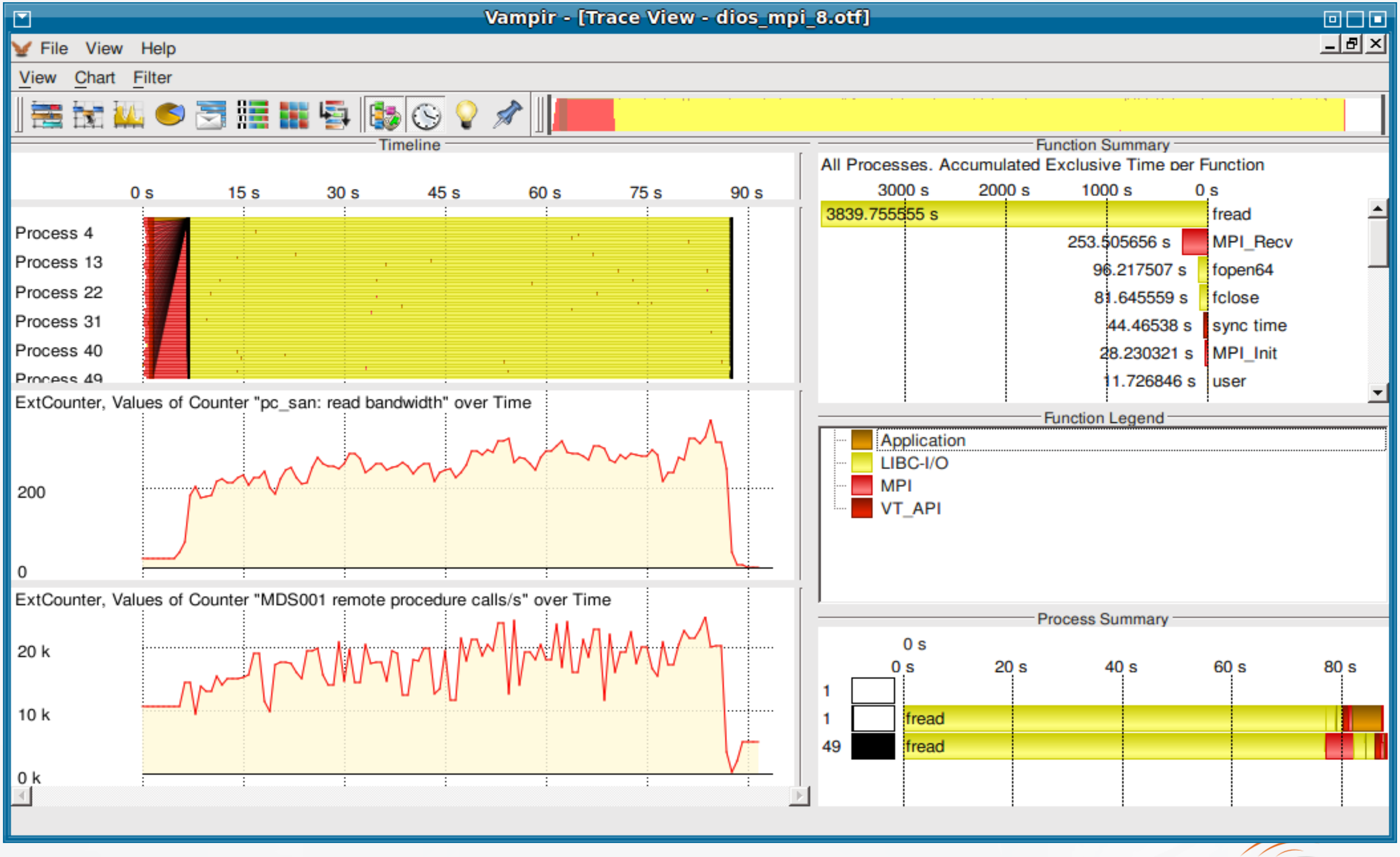


Vampir - [Trace View - semtex_dios_512er_IB/dios_mpi.otf]
File View Help
View Chart Filter
Timeline
25 s 30 s 35 s 40 s 45 s 50 s 55 s 60 s 65 s 70 s
Process 20
Process 21
Process 22
Process 23
read
MPI_Barrier
Process 22, Values of Counter "bytes received on this IB port" over Time
200 M
0 M
Process 22, Values of Counter "bytes read on this lustre client" over Time
250 M
0 M
ExtCounter, Values of Counter "deimos /fastfs read bandwidth" over Time
3 k
0 k
Function Summary
All Processes. Accumulated Exclusiv...
10000 s
0 s
15632.317851 s MPI
9517.293866 s LIBC-I/O
319.516616 s VT_API
19.791424 s Application
Function Legend
Application
LIBC-I/O
MPI
VT_API
Process Summary
0 s
0 s
30 s
280
138
64
31
MPI_Barrier
read
MP...er

Example – Local and External Counter Values



Example – threader simulation with 50 processes



ExtCounter, Values of Counter "pc_san: read bandwidth" over Time

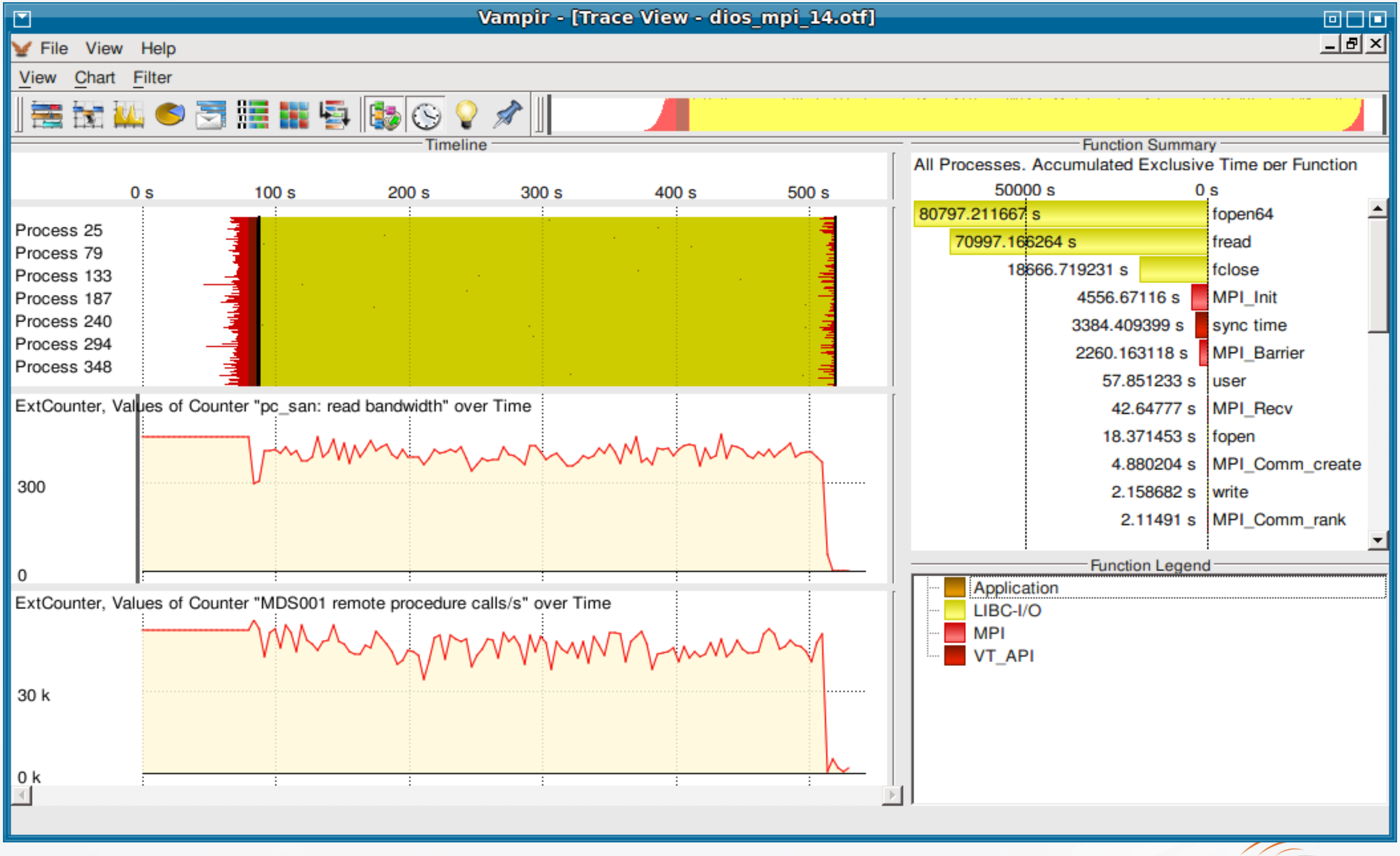
ExtCounter, Values of Counter "MDS001 remote procedure calls/s" over Time

All Processes, Accumulated Exclusive Time per Function

Time	Function
3839.755555 s	fread
253.505656 s	MPI_Recv
96.217507 s	fopen64
81.645559 s	fclose
44.46538 s	sync time
28.230321 s	MPI_Init
11.726846 s	user

- Application
- LIBC-I/O
- MPI
- VT_API

Example – threader simulation with 400 processes

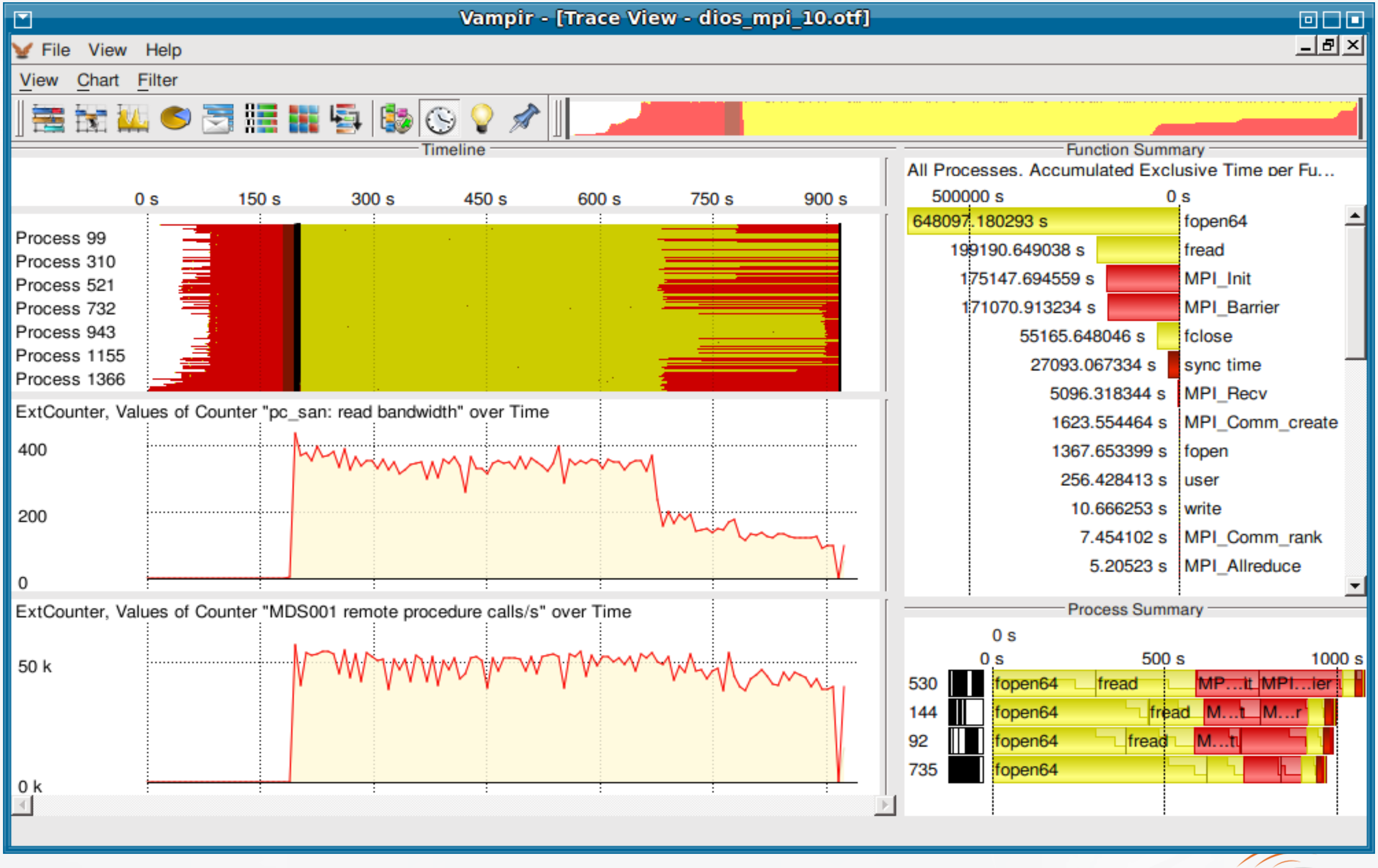


All Processes. Accumulated Exclusive Time per Function

Time	Function
80797.211667 s	fopen64
70997.166264 s	fread
18666.719231 s	fclose
4556.67116 s	MPI_Init
3384.409399 s	sync time
2260.163118 s	MPI_Barrier
57.851233 s	user
42.64777 s	MPI_Recv
18.371453 s	fopen
4.880204 s	MPI_Comm_create
2.158682 s	write
2.11491 s	MPI_Comm_rank

- 🟧 Application
- 🟨 LIBC-I/O
- 🟥 MPI
- 🟥 VT_API

Example – threader simulation with 1500 processes

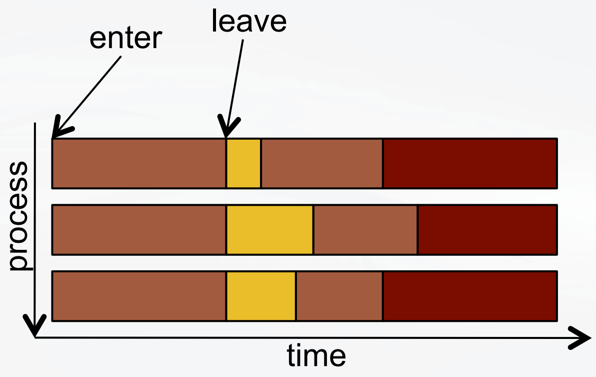


All Processes. Accumulated Exclusive Time per Fu...

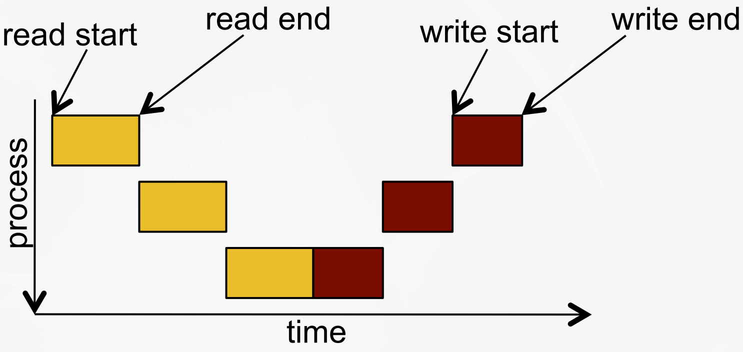
500000 s	0 s
648097.180293 s	fopen64
199190.649038 s	fread
175147.694559 s	MPI_Init
171070.913234 s	MPI_Barrier
55165.648046 s	fclose
27093.067334 s	sync time
5096.318344 s	MPI_Recv
1623.554464 s	MPI_Comm_create
1367.653399 s	fopen
256.428413 s	user
10.666253 s	write
7.454102 s	MPI_Comm_rank
5.20523 s	MPI_Allreduce

File Access Pattern Visualization

- what does the Vampir Master Timeline show:
  - X axis: time
  - Y axis: processes



- what is needed for file access patterns:
  - X axis: byte range within a single file
  - Y axis: processes



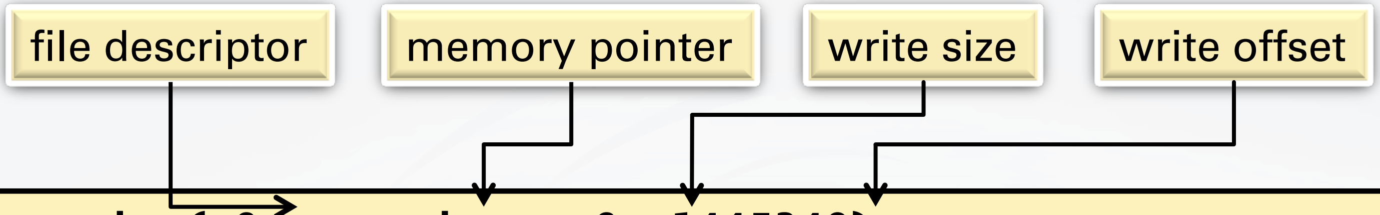
- record each write/read access for each byte range in a vector
- merge consecutive accesses
- write an appropriate enter/leave pair based on the process number and the access offset/access range into a new trace file

Flash I/O and HDF5

- Multiphysics astronomy code
- solves a broad range of astrophysical problems
- uses HDF5 or pNetCDF for parallel storage access
- I/O kernel available since 2001
- revised implementation available since 2006

- What has been changed?
  - instrument both versions
  - check differences in the I/O patterns

Pattern - 2001 Version



file descriptor

memory pointer

write size

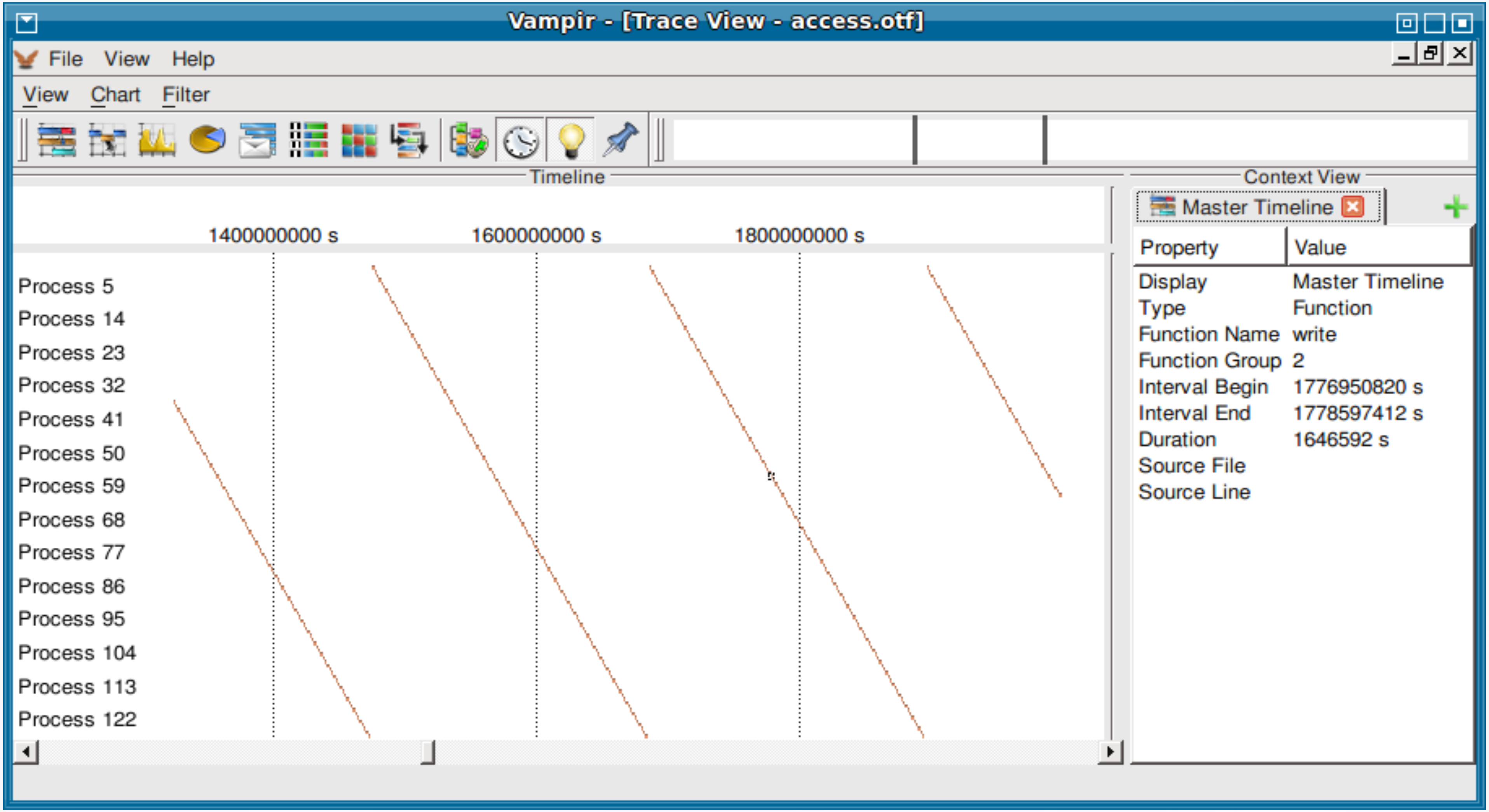
write offset

```
1:        pwrite( 8, somewhere, 8, 1445340);
1:        pwrite( 8, somewhere, 8, 1445348);
1:        pwrite( 8, somewhere, 8, 1445356);
1:        pwrite( 8, somewhere, 8, 1445364);
1:        pwrite( 8, somewhere, 8, 1445372);
1:        pwrite( 8, somewhere, 8, 1445380);
1:        pwrite( 8, somewhere, 8, 1445388);
1:        pwrite( 8, somewhere, 8, 1445396);
1:        pwrite( 8, somewhere, 8, 1445404);
1:        pwrite( 8, somewhere, 8, 1445412);
1:        pwrite( 8, somewhere, 8, 1445420);
1:        pwrite( 8, somewhere, 8, 1445428);
```

Pattern - 2006 Version

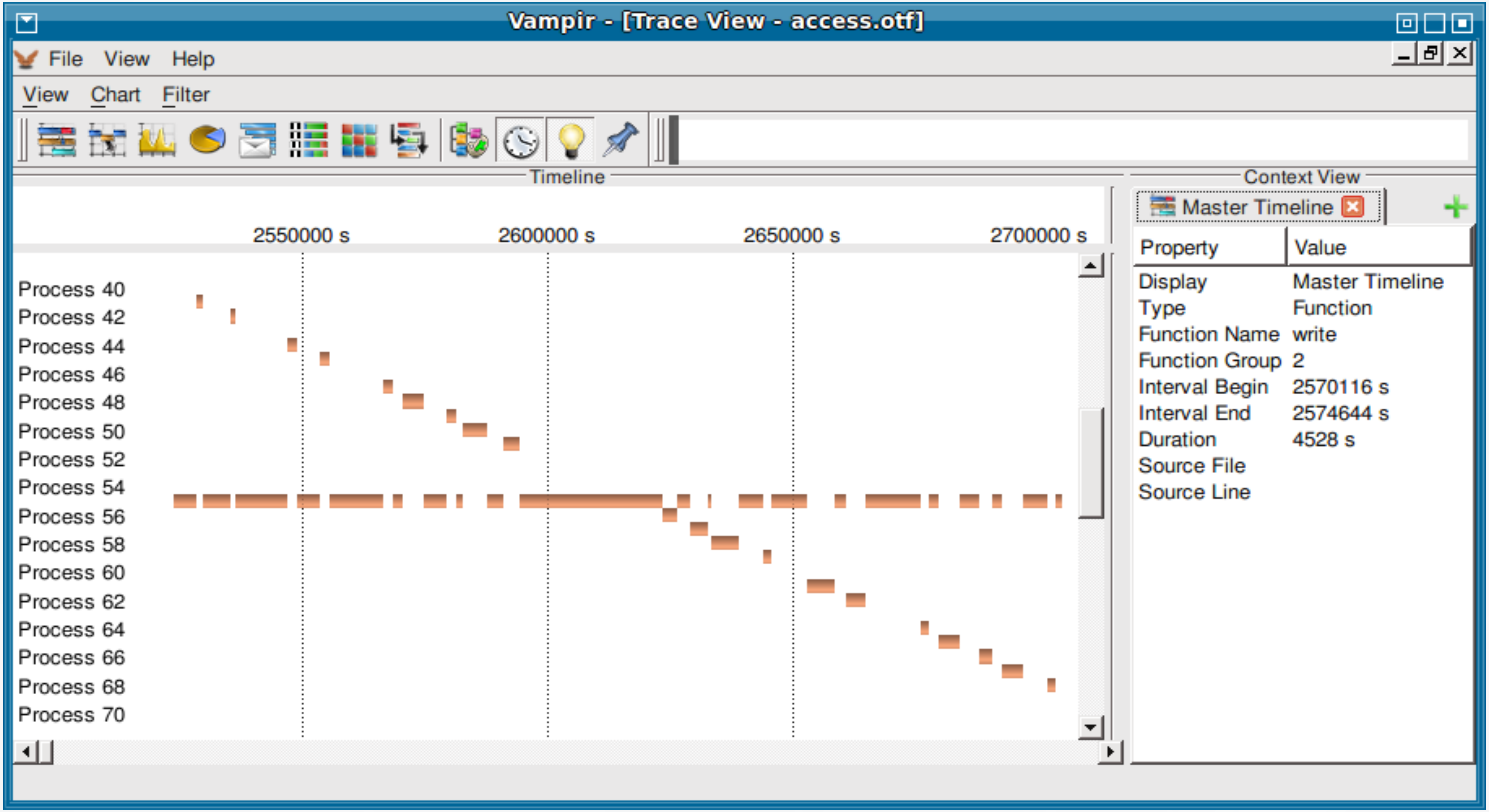
```
1:        pwrite( 8, somewhere, 1638400, 1054266916);
1:        pwrite( 8, somewhere, 1638400, 1264502308);
1:        pwrite( 8, somewhere, 1638400, 1474737700);
1:        pwrite( 8, somewhere, 1638400, 1684975140);
1:        pwrite( 8, somewhere, 1638400, 1895210532);
1:        pwrite( 8, somewhere, 1638400, 2105445924);
1:        pwrite( 8, somewhere, 1638400, 2315681316);
1:        pwrite( 8, somewhere, 1638400, 2525916708);

        ...
```

Access Pattern Visualization



Property	Value
Display	Master Timeline
Type	Function
Function Name	write
Function Group	2
Interval Begin	1776950820 s
Interval End	1778597412 s
Duration	1646592 s
Source File	
Source Line	

Access Pattern Visualization



Property	Value
Display	Master Timeline
Type	Function
Function Name	write
Function Group	2
Interval Begin	2570116 s
Interval End	2574644 s
Duration	4528 s
Source File	
Source Line	

Time for Questions