Score-P & Vampir
Comprehensive Multi-Paradigm Performance Analysis

Ronny Brendel
Technische Universität Dresden
ronny.brendel@tu-dresden.de

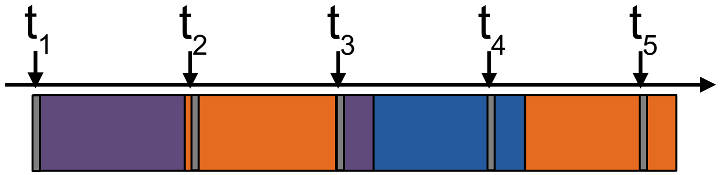
Argonne Training Program on Extreme-Scale Computing (ATPESC), St. Charles, IL, 7th August 2018

Introduction
Why Bother Analyzing Performance

- There are countless ways to leave performance on the table
  - Lots of little function calls due to e.g. constructors/destructors
  - Inefficient parallelization
  - Lack of vectorization
  - Bad memory access patterns / cache usage
  - Bad file I/O usage
  - …
- Many performance tools are really easy to use
  - Just try it out on your code or pet project
- There are also things performance tools cannot help you with
  - Different/better algorithms

Introduction
Methods

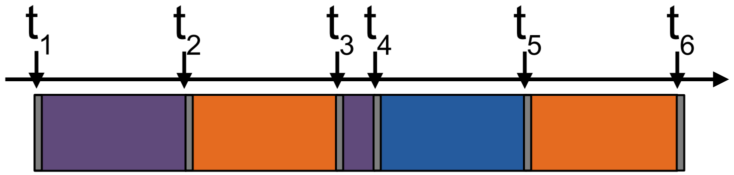
Sampling



- Interrupt in given intervals (typically ~10ms)

- Statistical correctness guarantees

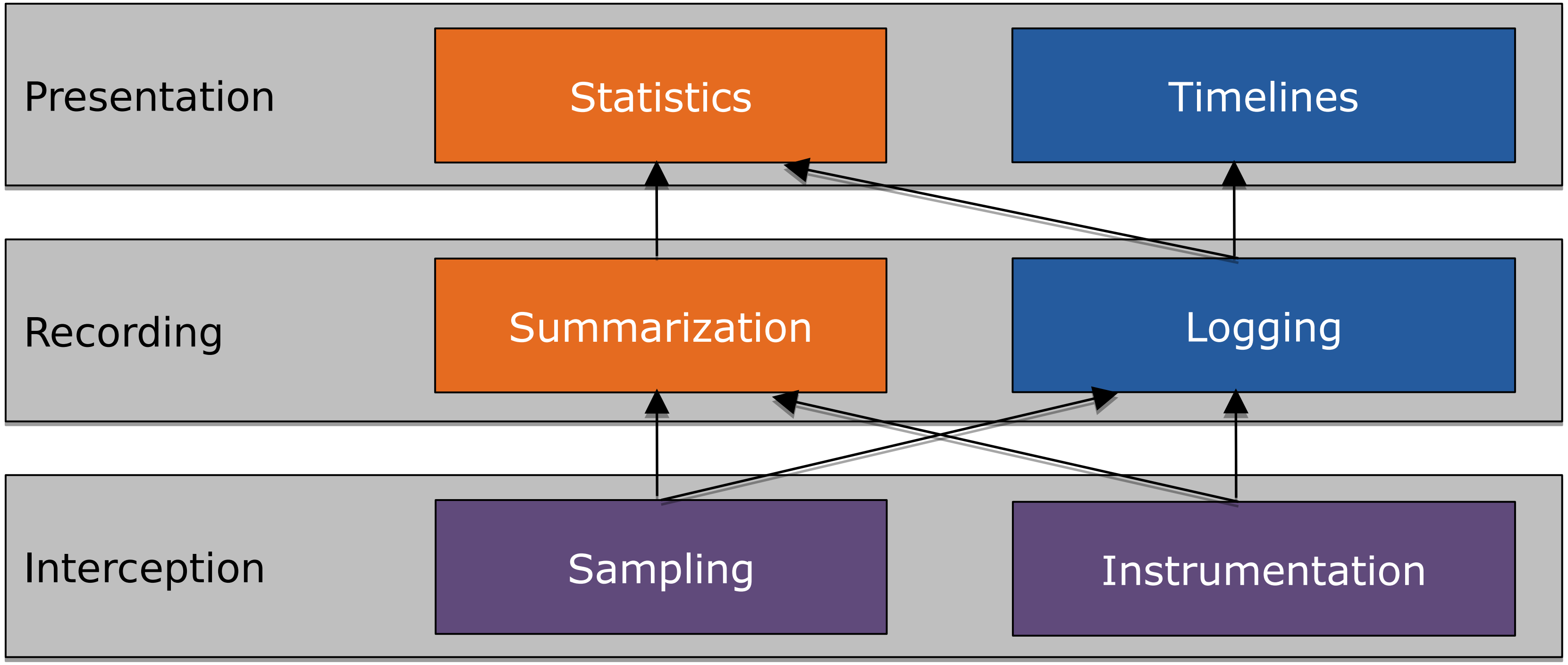
Instrumentation



- Callback before and after event

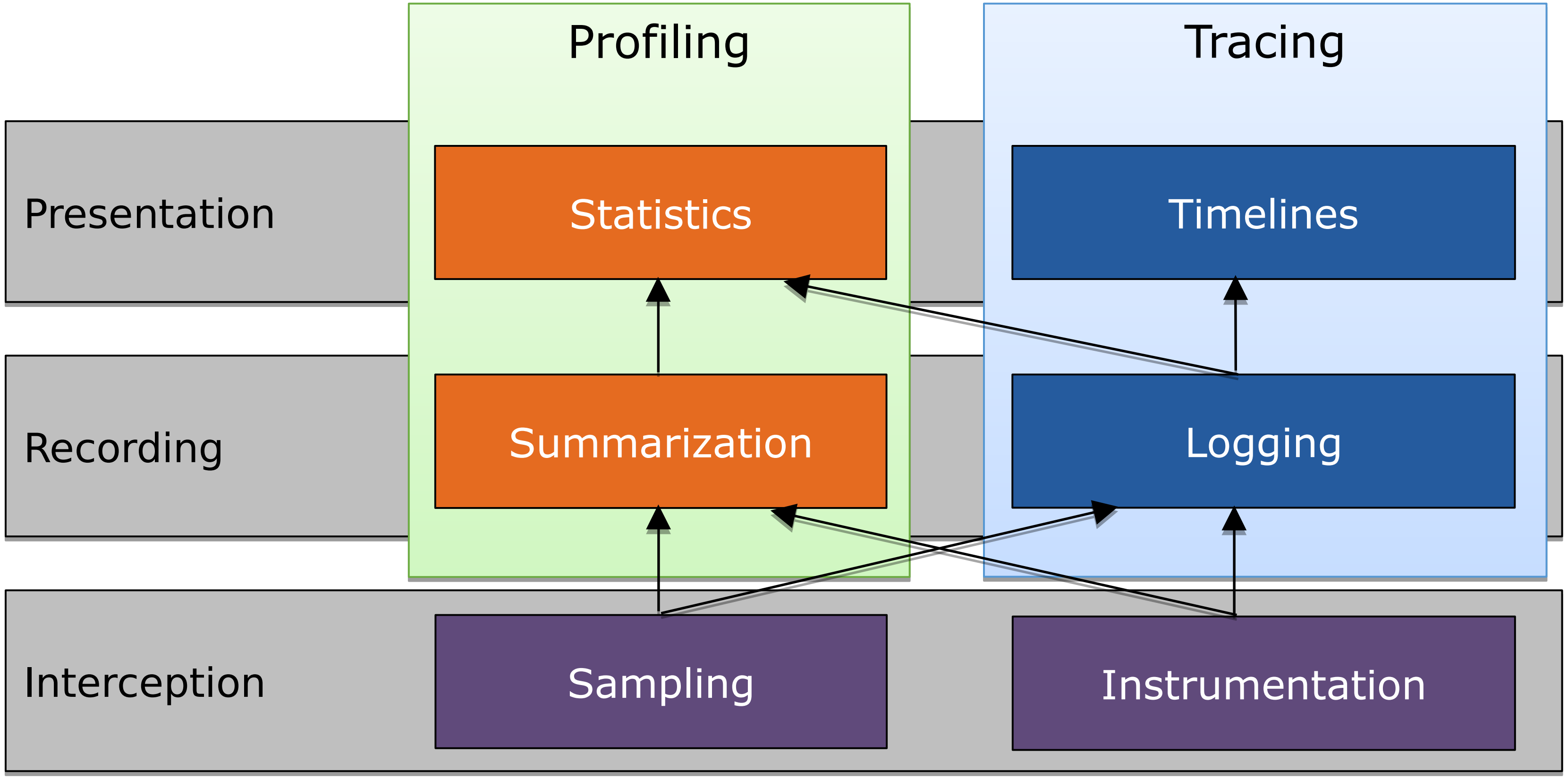
- Exact times and counts
- Wrappers have easy access to function arguments

Introduction
Methods



Introduction

Methods



Introduction
Methods

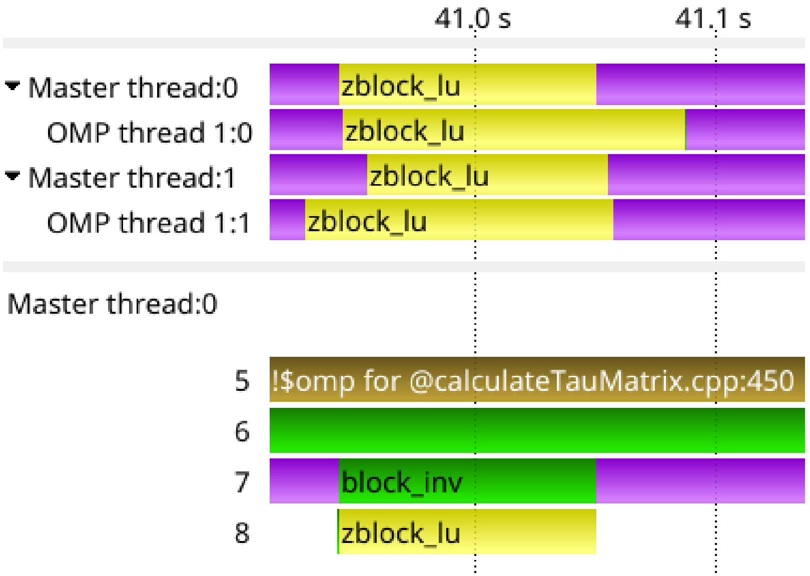
Profile

- Information accumulated into buckets
- Typically small overhead
- Typically Static representation

Time		Function name
(%)	(s)	
5.44	1.21	QListData::isEmpty
2.96	0.66	QHash::findNode
2.67	0.60	QList::last
1.71	0.38	handleEnter
0.58	0.13	QHash::find

Trace

- Event log
- Possibly large overhead
- Interactive representation



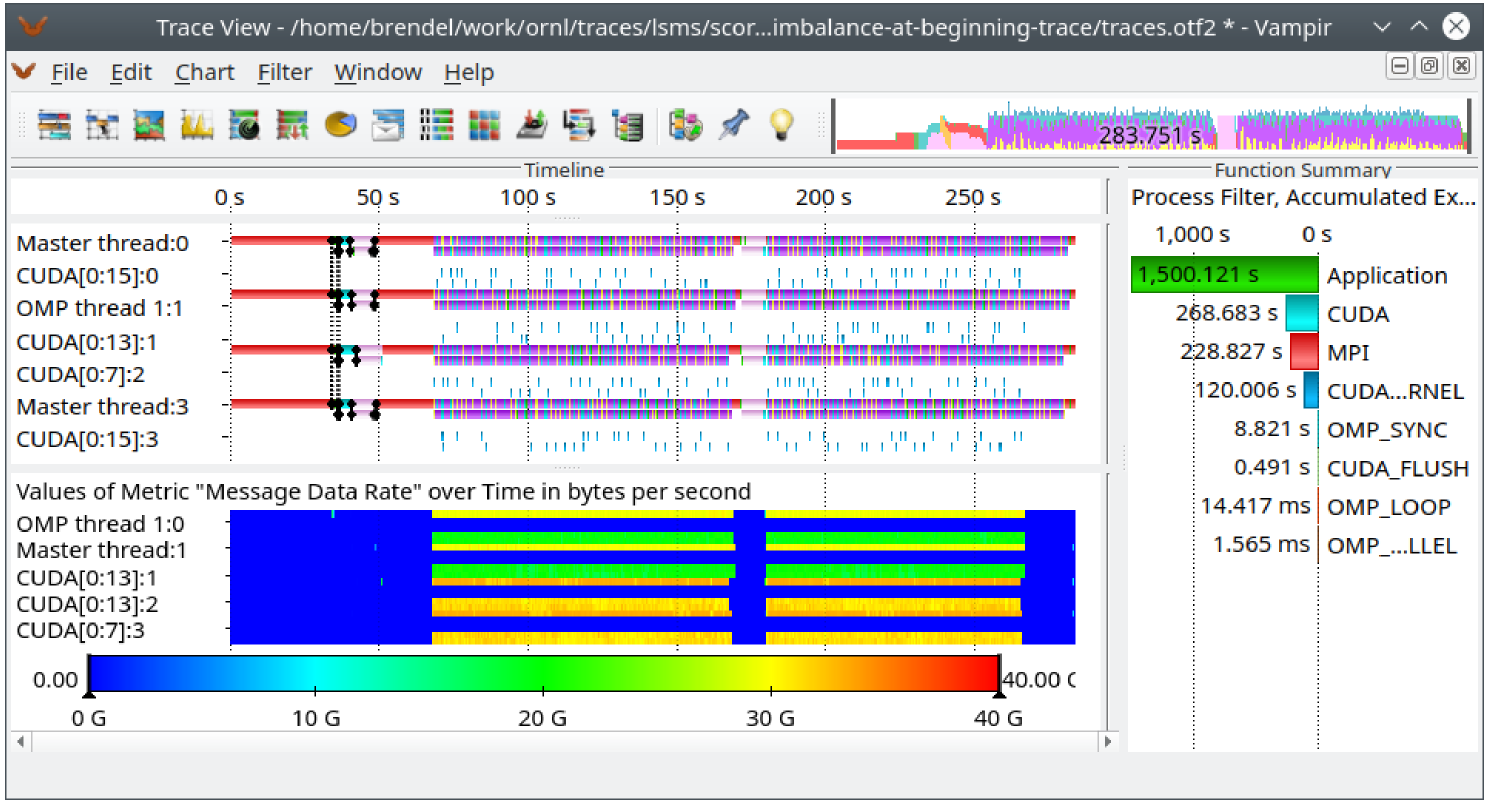
Introduction
Methods

- Trade-offs
  - Ease of use
  - Run time overhead and recording size
  - Accuracy
  - API semantics (e.g. MPI_Send‘s sender and receiver processes and transferred bytes)

- Most tools combine both sampling (+ call stack unwinding) and instrumentation of library events (e.g. MPI, OpenMP and CUDA library functions)
  - To avoid problems with some techniques while gathering enough information

Introduction
Vampir

- Comprehensive, powerful performance data visualization
- Developed since 1996
- Commercial



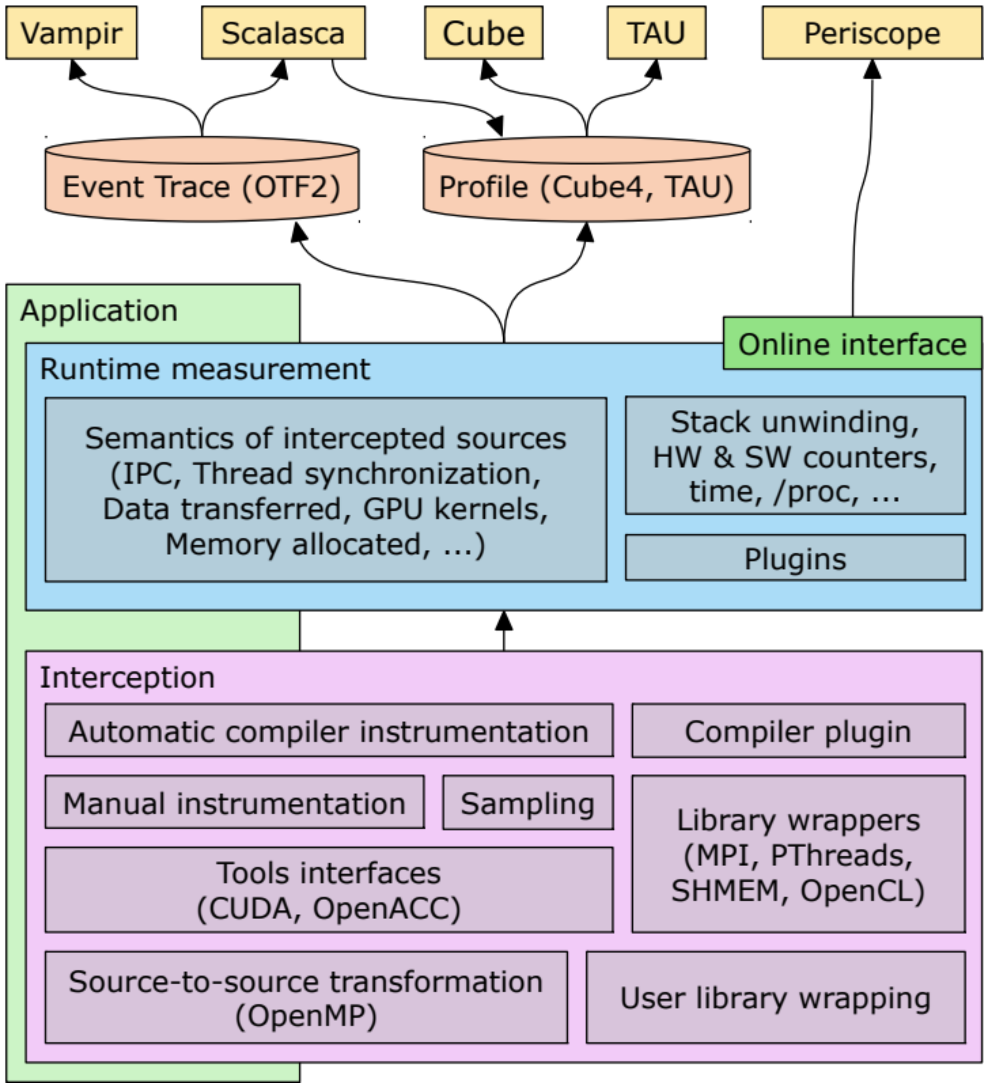
1,500.121 s	Application
268.683 s	CUDA
228.827 s	MPI
120.006 s	CUDA...RNEL
8.821 s	OMP_SYNC
0.491 s	CUDA_FLUSH
14.417 ms	OMP_LOOP
1.565 ms	OMP_...LLEL

Introduction
Score-P

- Jointly developed performance data collector
- Developed since 2009
- Open-source (3-clause BSD)
- Partners:
  - TU Dresden, GER
  - FZ Jülich, GER
  - TU München, GER
  - University of Oregon, USA
  - RWTH Aachen; TU Darmstadt; Gesellschaft für numerische Simulation mbH; German Research School for Simulation Sciences GmbH (all GER)

Introduction
Score-P

- Supports:
  - C, C++, Fortran
  - MPI, SHMEM
  - OpenMP, PThreads
  - CUDA, OpenACC, OpenCL
  - Compilers: Cray, GNU, IBM, Intel, Pathscale, PGI, (LLVM)


Vampir
Scalasca
Cube
TAU
Periscope
Event Trace (OTF2)
Profile (Cube4, TAU)
Application
Online interface
Runtime measurement
Semantics of intercepted sources (IPC, Thread synchronization, Data transferred, GPU kernels, Memory allocated, …)
Stack unwinding, HW & SW counters, time, /proc, …
Plugins
Interception
Automatic compiler instrumentation
Compiler plugin
Manual instrumentation
Sampling
Library wrappers (MPI, PThreads, SHMEM, OpenCL)
Tools interfaces (CUDA, OpenACC)
Source-to-source transformation (OpenMP)
User library wrapping

Tutorial

Data Collection with Score-P

Tutorial
Data Collection with Score-P

- Load Score-P

```
$ module load scorep
```

at ANL

```
$ echo "+vampir" >> ~/.soft && resoft
```

- Compile & Link

```
$ scorep … gcc …  main.c
```

with MPI

```
$ scorep … mpicc … main.c
```

with SHMEM

```
$ scorep … oshcc … main.c
```

  - CMake

```
$ SCOREP_WRAPPER=OFF cmake -DCMAKE_C_COMPILER=scorep-gcc ..
$ SCOREP_WRAPPER_INSTRUMENTER_FLAGS="…" SCOREP_WRAPPER_COMPILER_FLAGS="…" make
```

  - Autotools

```
$ SCOREP_WRAPPER=OFF ../configure CC=scorep-gcc MPICC=scorep-mpicc ..
$ SCOREP_WRAPPER_INSTRUMENTER_FLAGS="…" SCOREP_WRAPPER_COMPILER_FLAGS="…" make
```

Tutorial
Data Collection with Score-P

- Execute

```
$ ./a.out
```

```
$ mpirun -np 2 ./a.out
```

```
$ shmemrun -np 2 ./a.out
```

- Inspect

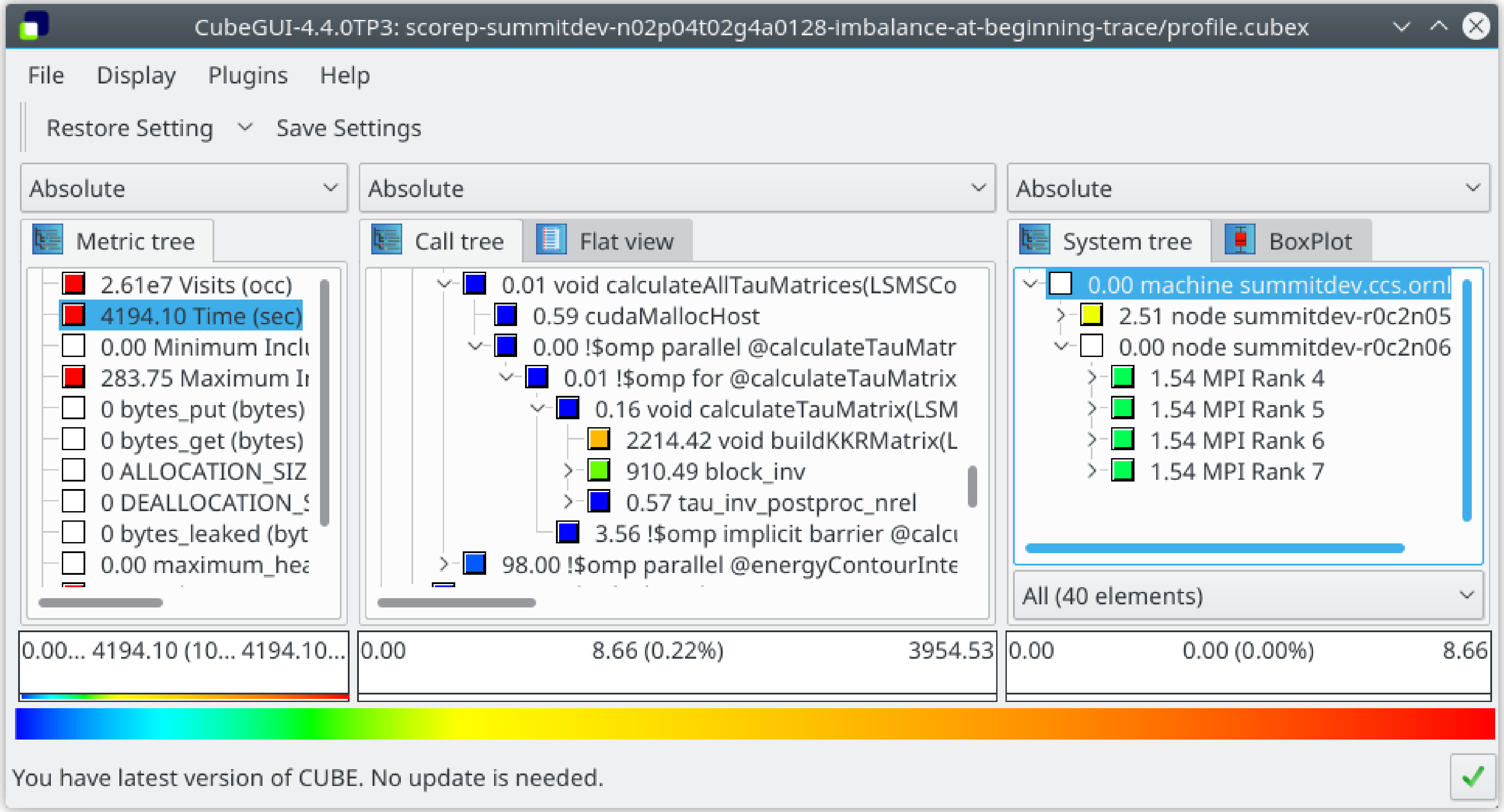
```
$ ls -R
scorep-20170323_1309_7243761919249966  a.out

./scorep-20170323_1309_7243761919249966:
profile.cubex  scorep.cfg
```

- Inspect > Cube

```
$ cube scorep-20170323_1309_7243761919249966/profile.cubex
```

Tutorial

Profile Visualization with Cube



CubeGUI-4.4.0TP3: scorep-summitdev-n02p04t02g4a0128-imbalance-at-beginning-trace/profile.cubex

Metric tree:

- 🟥 2.61e7 Visits (occ)
- 🟥 4194.10 Time (sec)
- □ 0.00 Minimum Inclu
- 🟥 283.75 Maximum In
- □ 0 bytes_put (bytes)
- □ 0 bytes_get (bytes)
- □ 0 ALLOCATION_SIZ
- □ 0 DEALLOCATION_S
- □ 0 bytes_leaked (byt
- □ 0.00 maximum_hea

Call tree:

- 🟦 0.01 void calculateAllTauMatrices(LSMSCo
  - 🟦 0.59 cudaMallocHost
  - 🟦 0.00 !$omp parallel @calculateTauMatr
    - 🟦 0.01 !$omp for @calculateTauMatrix
      - 🟦 0.16 void calculateTauMatrix(LSM
        - 🟧 2214.42 void buildKKRMatrix(L
        - 🟩 910.49 block_inv
        - 🟦 0.57 tau_inv_postproc_nrel
      - 🟦 3.56 !$omp implicit barrier @calcu
- 🟦 98.00 !$omp parallel @energyContourInte

System tree:

- □ 0.00 machine summitdev.ccs.ornl
  - 🟨 2.51 node summitdev-r0c2n05
  - □ 0.00 node summitdev-r0c2n06
    - 🟩 1.54 MPI Rank 4
    - 🟩 1.54 MPI Rank 5
    - 🟩 1.54 MPI Rank 6
    - 🟩 1.54 MPI Rank 7

All (40 elements)

0.00... 4194.10 (10... 4194.10...

0.00 8.66 (0.22%) 3954.53

0.00 0.00 (0.00%) 8.66

You have latest version of CUBE. No update is needed. ✓

Tutorial
Data Collection with Score-P

- Runtime Options
  - Profiling (default)

    ```
    $ export SCOREP_ENABLE_PROFILING=true
    ```

  - Tracing

    ```
    $ export SCOREP_ENABLE_TRACING=true
    ```

  - Performance counters

    ```
    $ export SCOREP_METRIC_PAPI=PAPI_L2_TCM,…
    ```

  - Filtering

    ```
    $ export SCOREP_FILTERING_FILE=my.filt
    ```

  - Memory (default: 16M)

    ```
    $ export SCOREP_TOTAL_MEMORY=400M
    ```

  - And many more…

    ```
    $ scorep-info config-vars
    ```

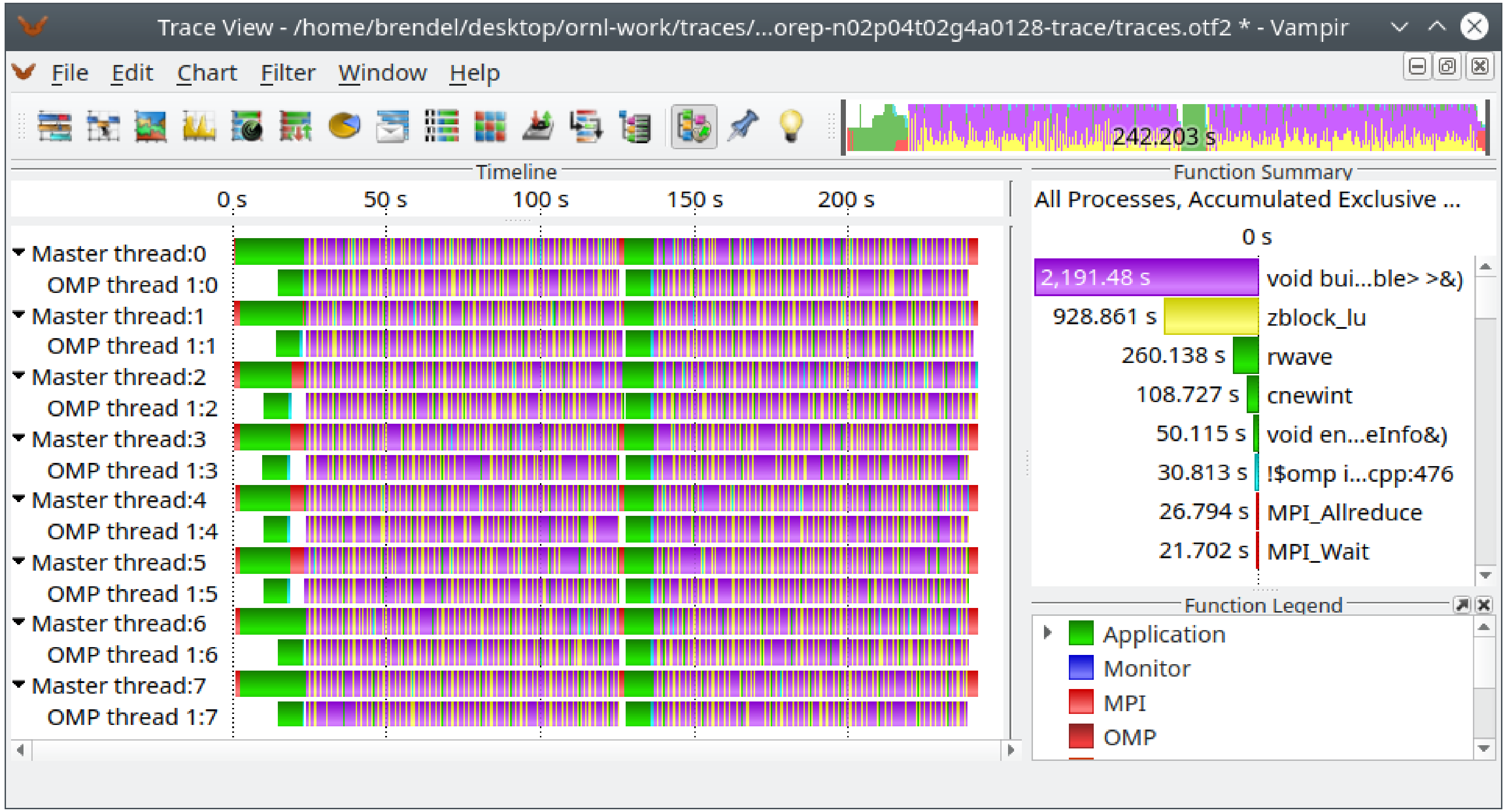
Tutorial

Trace Visualization with Vampir

```
$ export SCOREP_ENABLE_PROFILING=false
$ export SCOREP_ENABLE_TRACING=true
$ export SCOREP_METRIC_PAPI=PAPI_TOT_INS,PAPI_TOT_CYC

$ mpirun -np 4 ./a.out

$ ls -R
scorep-20170323_1309_7243761919249966  a.out

./scorep-20170323_1309_7243761919249966:
scorep.cfg  traces/  traces.def  traces.otf2

$ module load vampir

$ vampir scorep-20170323_1309_7243761919249966/traces.otf2
```

Tutorial

Trace Visualization with Vampir



Trace View - /home/brendel/desktop/ornl-work/traces/...orep-n02p04t02g4a0128-trace/traces.otf2 * - Vampir

All Processes, Accumulated Exclusive ...

2,191.48 s void bui...ble> >&)
928.861 s zblock_lu
260.138 s rwave
108.727 s cnewint
50.115 s void en...eInfo&)
30.813 s !$omp i...cpp:476
26.794 s MPI_Allreduce
21.702 s MPI_Wait

🟩 Application
🟦 Monitor
🟥 MPI
🟥 OMP

Tutorial
Data Collection Overhead

- Trace size and overhead varies greatly with event rate
  - Make a reference run and check wall clock time!
  - Rule of thumb: Try to stay below 10% overhead

→ Filtering is an integral part of Score-P‘s workflow

Tutorial
Data Collection with Score-P

- Score-P workflow as presented so far:
  1) Instrument & build
  2) Execute
  3) Analyze profile using Cube

Tutorial
Data Collection with Score-P

- Score-P workflow with filtering

1) Instrument & build

2) Execute (profiling)

3) Analyze overhead

   If the estimated trace size is too large, filter and goto 3

4) Execute using the filter (tracing)

5) Analyze trace using Vampir

Tutorial
Data Collection with Score-P

3) Analyze Overhead

```
$ scorep-score scorep-20170323_1309_7243761919249966/profile.cubex

Estimated aggregate size of event trace:                   40GB
Estimated requirements for largest trace buffer (max_buf): 6GB
Estimated memory requirements (SCOREP_TOTAL_MEMORY):       6GB
(warning: The memory requirements cannot be satisfied by Score-P to avoid
 intermediate flushes when tracing. Set SCOREP_TOTAL_MEMORY=4G to get the
 maximum supported memory or reduce requirements using USR regions filters.)

flt     type    max_buf[B]        visits time[s] time[%] time/visit[us]  region
         ALL 5,383,272,006 1,635,443,611  579.23   100.0           0.35  ALL
         USR 5,358,738,138 1,631,138,913  253.00    43.7           0.16  USR
         OMP    23,580,522     4,089,856  318.79    55.0          77.95  OMP
         COM       665,210       182,120    0.90     0.2           4.95  COM
         MPI       288,136        32,722    6.55     1.1         200.11  MPI
```

Tutorial
Data Collection with Score-P

3) Analyze Overhead

```
$ scorep-score -r scorep-20170323_1309_7243761919249966/profile.cubex
  [...]

flt     type    max_buf[B]        visits time[s] time[%] time/visit[us]  region
         ALL 5,383,272,006 1,635,443,611  579.23   100.0           0.35  ALL
         USR 5,358,738,138 1,631,138,913  253.00    43.7           0.16  USR
         OMP    23,580,522     4,089,856  318.79    55.0          77.95  OMP
         COM       665,210       182,120    0.90     0.2           4.95  COM
         MPI       288,136        32,722    6.55     1.1         200.11  MPI

         USR 1,716,505,830   522,844,416   79.32    13.7           0.15  matmul_sub_
         USR 1,716,505,830   522,844,416   53.44     9.2           0.10  matvec_sub_
         USR 1,716,505,830   522,844,416  111.47    19.2           0.21  binvcrhs_
         USR    76,195,080    22,692,096    2.76     0.5           0.12  binvrhs_
         USR    76,195,080    22,692,096    4.37     0.8           0.19  lhsinit_
         USR    56,825,184    17,219,840    1.63     0.3           0.09  exact_solution_
```

Tutorial
Data Collection with Score-P

3) Create filter

```
$ cat myfilter.filt
SCOREP_REGION_NAMES_BEGIN
  EXCLUDE
    matmul_sub*
    matvec_sub*
    binvcrhs*
    Binvrhs*
    exact_solution*
    lhs*init*
    timer_*
SCOREP_REGION_NAMES_END

$ scorep-score -f myfilter.filt scorep-20170323*/profile.cubex

Estimated aggregate size of event trace:                   409MB
Estimated requirements for largest trace buffer (max_buf): 58MB
Estimated memory requirements (SCOREP_TOTAL_MEMORY):       70MB
(hint: When tracing set SCOREP_TOTAL_MEMORY=70M to avoid
[..]
```

Tutorial
Data Collection with Score-P

4) Execute using the filter

```
$ export SCOREP_ENABLE_TRACING=true
$ export SCOREP_TOTAL_MEMORY=70M
$ export SCOREP_FILTERING_FILE=myfilter.filt

$ mpirun -np 8 ./a.out
```

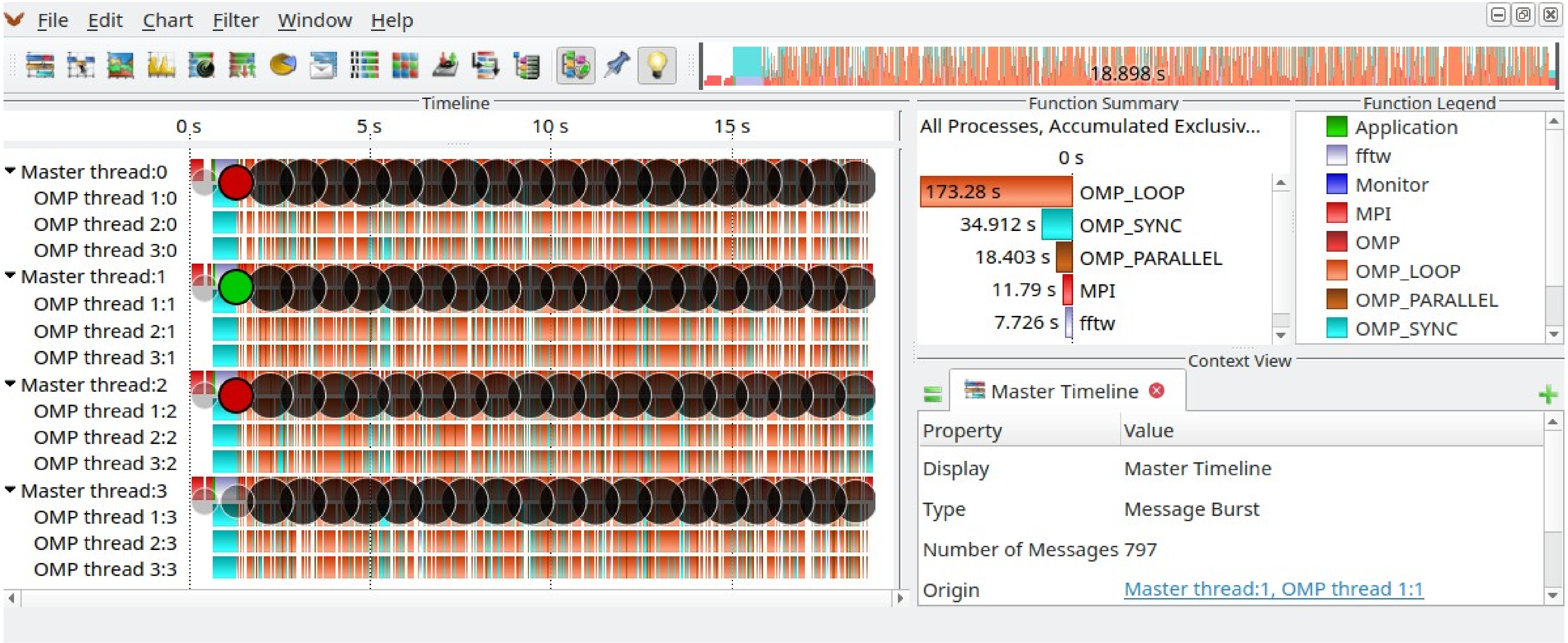
Compile-time filtering (GCC-only)

```
$ scorep --instrument-filter=myfilter.filt gcc main.c

$ export SCOREP_ENABLE_TRACING=true
$ export SCOREP_TOTAL_MEMORY=70M

$ mpirun -np 8 ./a.out          # no runtime filtering needed
```

Tutorial

Trace Visualization with Vampir (Live)

Tutorial
Trace Visualization with Vampir



Function Summary

All Processes, Accumulated Exclusiv...

0 s

173.28 s	OMP_LOOP
34.912 s	OMP_SYNC
18.403 s	OMP_PARALLEL
11.79 s	MPI
7.726 s	fftw

Function Legend

- 🟩 Application
- ▨ fftw
- 🟦 Monitor
- 🟥 MPI
- 🟥 OMP
- 🟧 OMP_LOOP
- 🟫 OMP_PARALLEL
- ▨ OMP_SYNC

Context View

Master Timeline

Property	Value
Display	Master Timeline
Type	Message Burst
Number of Messages	797
Origin	Master thread:1, OMP thread 1:1

Tutorial
Getting Help

-
  ```
  $ scorep --help
  ```

-
  ```
  $ scorep-wrapper --help
  ```

-
  ```
  $ scorep-info config-vars
  ```
- Manuals: `$SCOREP_DIR/share/doc/scorep/pdf/scorep.pdf`

  `$VAMPIR_ROOT/doc/vampir-manual.pdf`
- https://www.alcf.anl.gov/vampir
- support@score-p.org, service@vampir.eu

http://score-p.org

https://vampir.eu

- VI-HPS offers trainings (Invite them!)
  - http://www.vi-hps.org/training/tws/
  - http://www.vi-hps.org/training/material/

Conclusions, Acknowledgments

Conclusions

- Holistic, powerful and detailed software performance analysis
  - Everything in one picture
  - Extremely customizable
  - Extremely scalable
  - Advanced features
  - Very active in adopting new features
- Active research and development community
- Continuously selected by the OLCF
- **Enabler for science at extreme scale**

Sponsors & Projects

Federal Ministry of Education and Research

OAK RIDGE National Laboratory

Score-E

SILC

HAEC

European Commission

Horizon 2020 European Union funding for Research & Innovation

H4H

READEX

LMAC

nextgenio

U.S. DEPARTMENT OF ENERGY

Office of Science

pop

ELP

eeClust

INDIANA UNIVERSITY

ITEA 2

Microsoft

HLRS

High-Performance Computing Center | Stuttgart

SEVENTH FRAMEWORK PROGRAMME

ParMA

Contributors

- **Score-P**

  Andreas Knüpfer, Bert Wesarg, Christian Feld, Daniel Lorenz, Dirk Schmidl, Dominic Eschweiler, Felix Schmitt, Frank Winkler, Ilya Zhukov, Johannes Spazier, Johannes Ziegenbalg, Marc Schlütter, Markus Geimer, Michael Knobloch, Michael Wagner, Pavel Saviankou, René Jäkel, Robert Dietrich, Robert Mijaković, Robert Schöne, Robin Geyer, Ronny Brendel, Ronny Tschüter, Sameer Shende, Scott Biersdorff, Sebastian Oeste Suzanne Millstein, Thomas Ilsche, Yury Oleynik

- **Vampir**

  Alfred Arnold, Andreas Knüpfer, Bert Wesarg, Frank Winkler, Hartmut Mix, Heide Rohling, Holger Brunst, Jens Doleschal, Johannes Ziegenbalg, Matthias Weber, Laszlo Barabas, Michael Heyde, Michael Peter, Reinhard Neumann, Ronald Geisler, Ronny Brendel, Thomas William, Wolfgang E. Nagel

Hands-On

On Titan

https://goo.gl/A12mta

or

titan.ccs.ornl.gov → $WORLDWORK/stf010/brendel/atpesc