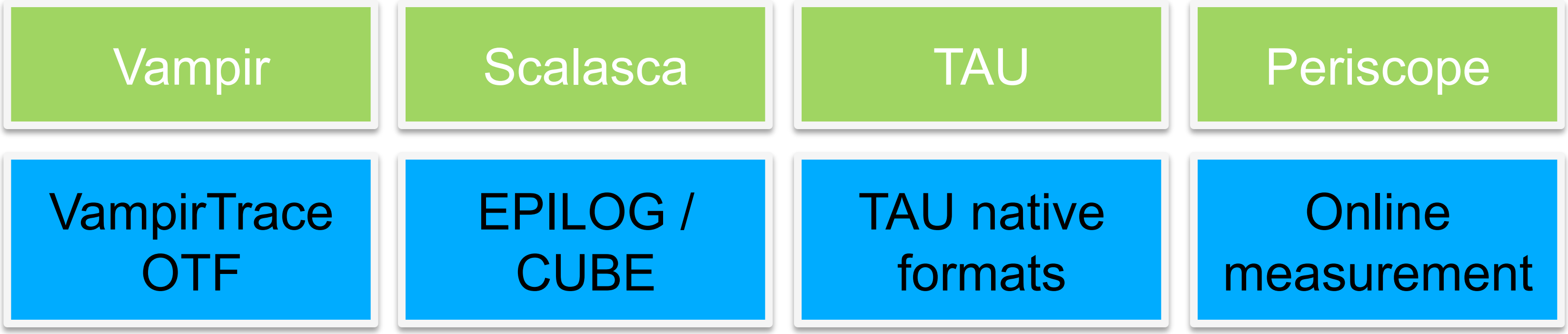
Score-P – A Joint Performance Measurement Run-Time Infrastructure for Periscope, Scalasca, TAU, and Vampir

Peter Philippen
p.philippen@fz-juelich.de

2013-04-03

Fragmentation of tools landscape

- Several performance tools co-exist
  - With own measurement systems and output formats
- Complementary features and overlapping functionality
- Redundant effort for development and maintenance
  - Limited or expensive interoperability
- Complications for user experience, support, training



Vampir	Scalasca	TAU	Periscope
VampirTrace OTF	EPILOG / CUBE	TAU native formats	Online measurement

Initial SILC Project

- Start a community effort for a common infrastructure
  - Score-P instrumentation and measurement system
  - Common data formats OTF2 and CUBE4
- Developer perspective:
  - Save manpower by sharing development resources
  - Invest in new analysis functionality and scalability
  - Save efforts for maintenance, testing, porting, support, training
- User perspective:
  - Single learning curve
  - Single installation, fewer version updates
  - Interoperability and data exchange

Project Overview

- SILC project funded by BMBF
- Close collaboration PRIMA project funded by DOE
- Continued in LMAC project (BMBF)
- Other projects:
  - H4H
  - HOPSA
  - Catwalk

GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

DEPARTMENT OF ENERGY
UNITED STATES OF AMERICA

Partners

- Forschungszentrum Jülich, Germany
- German Research School for Simulation Sciences, Aachen, Germany
- Gesellschaft für numerische Simulation mbH Braunschweig, Germany
- RWTH Aachen, Germany
- Technische Universität Dresden, Germany
- Technische Universität München, Germany
- University of Oregon, Eugene, USA

Score-P Functionality

- Provide typical functionality for HPC performance tools
- Support all fundamental concepts of partner's tools

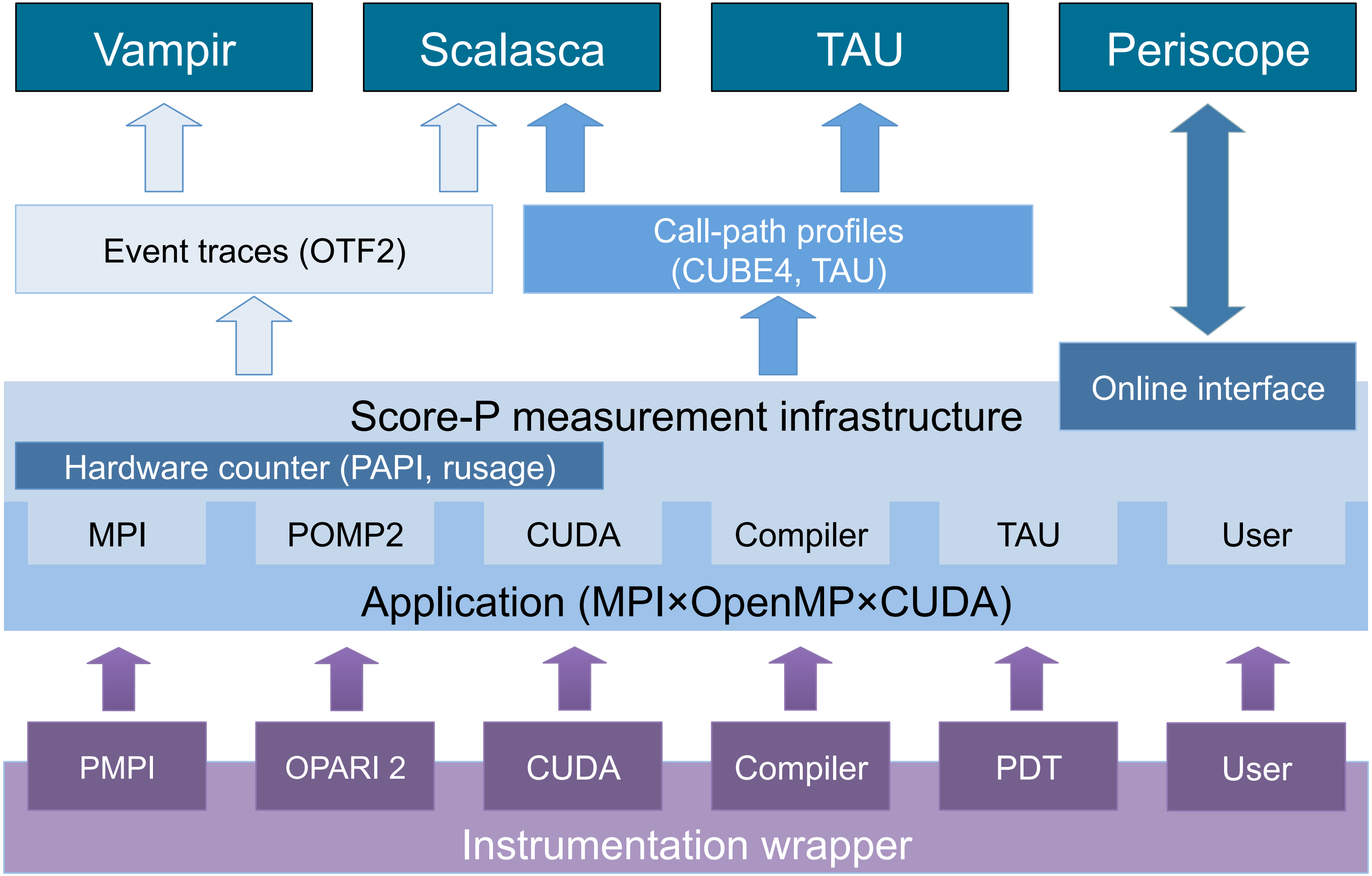
- Instrumentation (various methods)
- Flexible measurement without re-compilation:
  - Basic and advanced profile generation
  - Event trace recording
  - Online access to profiling data

- MPI, OpenMP, and hybrid parallelism (and serial)
- Enhanced functionality (OpenMP 3.0, CUDA, highly scalable I/O)

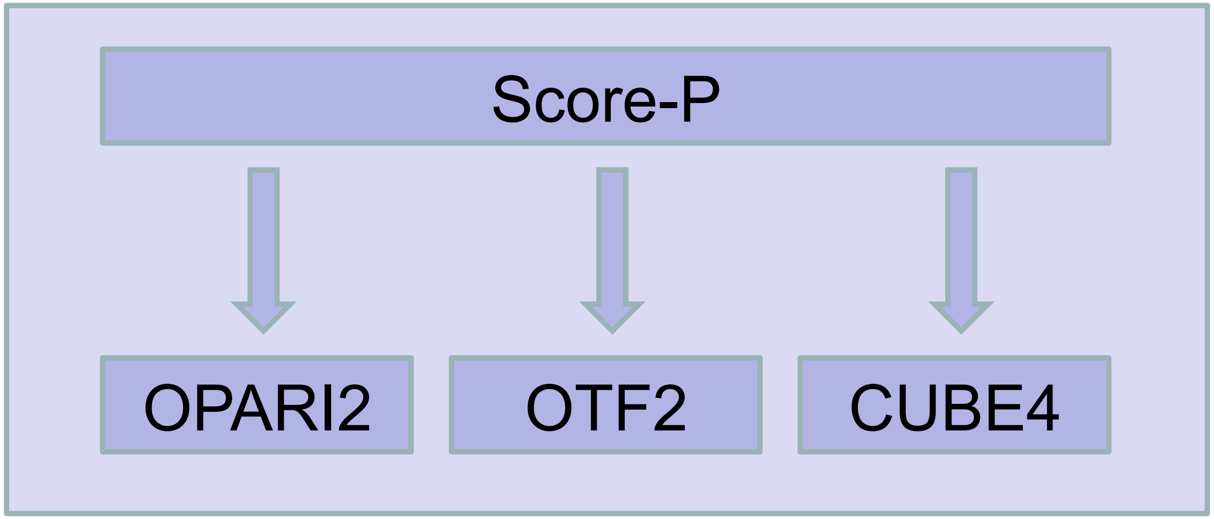
Design Goals

- Functional requirements
  - Generation of call-path profiles and event traces
  - Using direct instrumentation, later also sampling
  - Recording time, visits, communication data, hardware counters
  - Access and reconfiguration also at runtime
  - Support for MPI, OpenMP, basic CUDA, and all combinations
    - Later also OpenCL/HMPP/PTHREAD/…
- Non-functional requirements
  - Portability: all major HPC platforms
  - Scalability: petascale
  - Low measurement overhead
  - Easy and uniform installation through UNITE framework
  - Robustness
  - Open Source: New BSD License

Score-P Architecture



Components



- Separate, stand-alone packages
- Uniform look & feel
- Common functionality factored out
- Automated builds and tests

Score-P

- Instrumenter scorep
- Links application to measurement library
- Records time, visits, communication metrics, hardware counter using internal memory management
  - Store data in thread-local chunks of preallocated memory
  - Efficient utilization of available memory
  - Minimize perturbation/overhead
  - Useful for unification too
  - Variable number of threads
  - Access data during runtime
- Switch modes (tracing/profiling/online) w/o recompilation
- Layer architecture, minimal coupling, easy to extend

Score-P Instrumentation

- Prefix compile/link commands, e.g.

```
mpicc -c foo.c    →    scorep mpicc -c foo.c
```

or, more convenient

```
# in Makefile
MPICC = $(PREP) mpicc

% make PREP="scorep [options]"
```

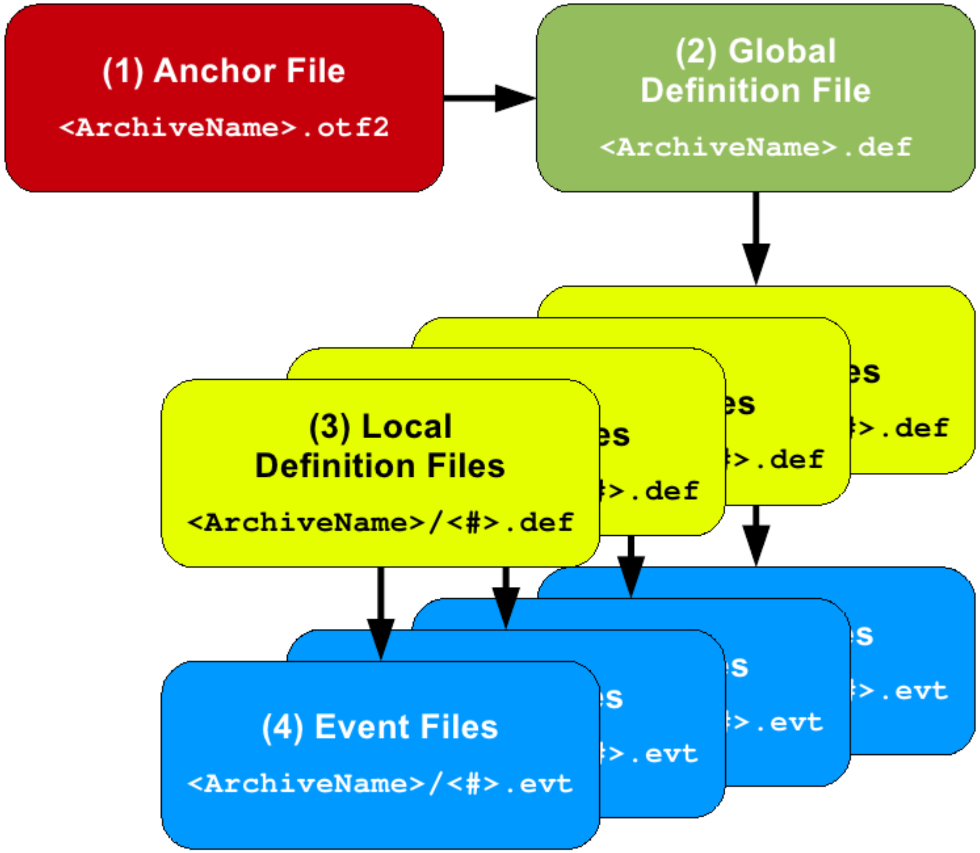
- Customization via options, e.g. --pdt --user
  % scorep --help for all options
- Automatic paradigm detection (serial/OpenMP/MPI/hybrid)
- Manual approach, get libs and includes via scorep_config

Score-P Run-Time Recording

- Just run your instrumented program!
- Customize via environment variables
  - Switch between profiling/tracing
  - Select MPI groups to record, e.g. P2P, COLL, …
  - Specify total measurement memory
  - Trace output options, e.g. compression, SION
  - Hardware counter (PAPI, rusage)
- Customize via files
  - In/exclude regions by name/wildcards from recording (filtering)
  - Restrict trace recording to specified executions of a region (selective tracing)
- Data written to uniquely named directory

The Open Trace Format Version 2 (OTF2)

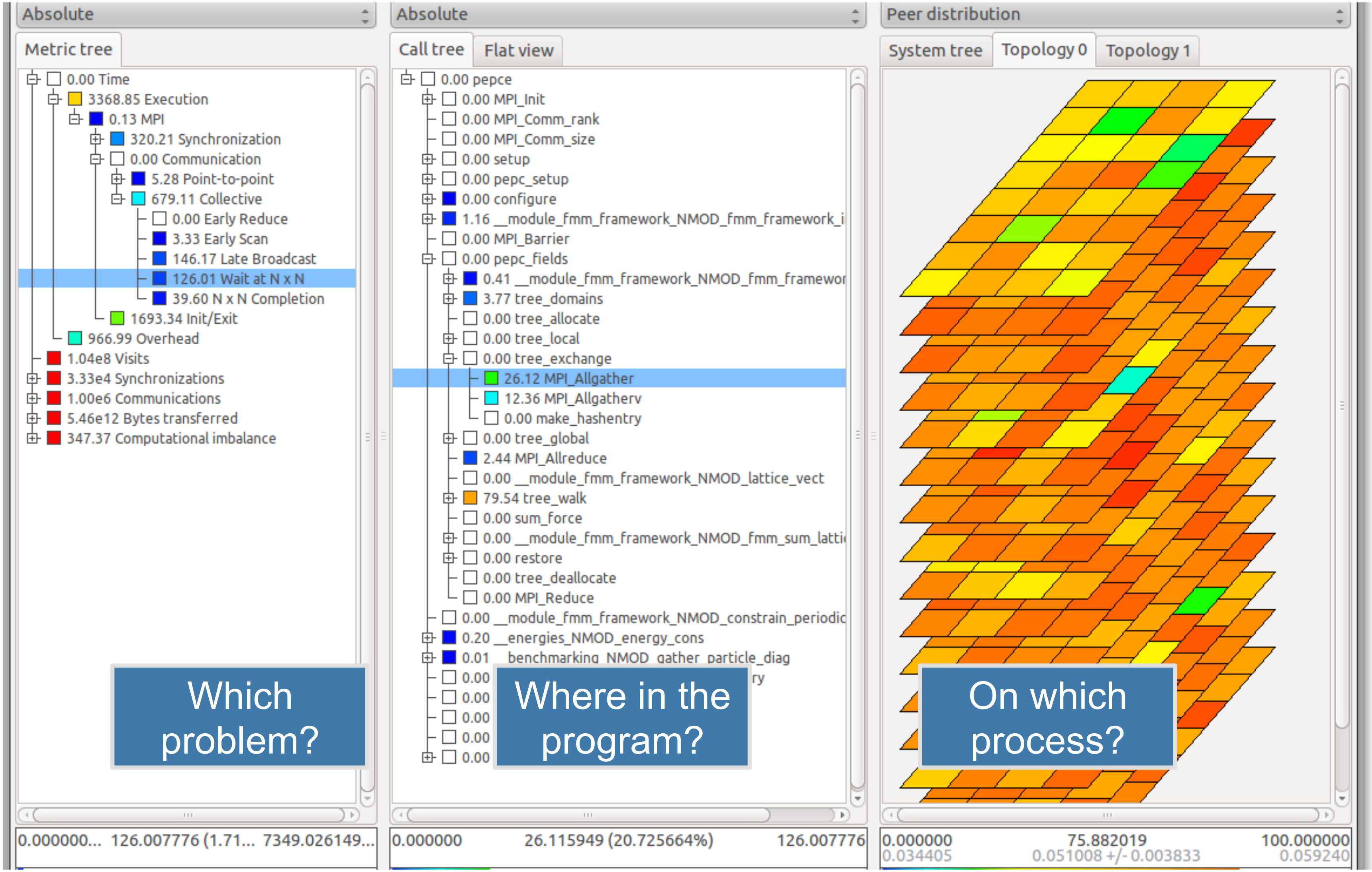
- Typical event trace data format
  - Event record types + definition record types
  - Stored per process/thread in temporal order
- Multi-file format
  - Anchor file
  - Global and local definitions + mappings
  - Event files
- OTF2 API + read/write library + support tools



CUBE4

- Generic format for call-path profiles of parallel apps, three-dimensional performance space:
  Metric hierarchy x Call-tree hierarchy x System hierarchy
- File organization, since version 4
  - Metadata stored in XML format
  - Metric values stored in binary format
    (Two files per metric: data + index for storage-efficient sparse representation)
  - Bundled into one file
- Optimized for
  - High write bandwidth
  - Fast interactive analysis through incremental loading

CUBE4 GUI



```
Absolute
Metric tree
□ 0.00 Time
  ■ 3368.85 Execution
    ■ 0.13 MPI
      ■ 320.21 Synchronization
      □ 0.00 Communication
        ■ 5.28 Point-to-point
        ■ 679.11 Collective
          □ 0.00 Early Reduce
          ■ 3.33 Early Scan
          ■ 146.17 Late Broadcast
          ■ 126.01 Wait at N x N
          ■ 39.60 N x N Completion
      ■ 1693.34 Init/Exit
    ■ 966.99 Overhead
■ 1.04e8 Visits
■ 3.33e4 Synchronizations
■ 1.00e6 Communications
■ 5.46e12 Bytes transferred
■ 347.37 Computational imbalance

0.000000...   126.007776 (1.71...   7349.026149...
```

```
Absolute
Call tree   Flat view
□ 0.00 pepce
  □ 0.00 MPI_Init
  □ 0.00 MPI_Comm_rank
  □ 0.00 MPI_Comm_size
  □ 0.00 setup
  □ 0.00 pepc_setup
  ■ 0.00 configure
  ■ 1.16 __module_fmm_framework_NMOD_fmm_framework_i
  □ 0.00 MPI_Barrier
  □ 0.00 pepc_fields
    ■ 0.41 __module_fmm_framework_NMOD_fmm_framewor
    ■ 3.77 tree_domains
    □ 0.00 tree_allocate
    □ 0.00 tree_local
    □ 0.00 tree_exchange
      ■ 26.12 MPI_Allgather
      ■ 12.36 MPI_Allgatherv
      □ 0.00 make_hashentry
    □ 0.00 tree_global
    ■ 2.44 MPI_Allreduce
    □ 0.00 __module_fmm_framework_NMOD_lattice_vect
    ■ 79.54 tree_walk
    □ 0.00 sum_force
    □ 0.00 __module_fmm_framework_NMOD_fmm_sum_latti
    □ 0.00 restore
    □ 0.00 tree_deallocate
    □ 0.00 MPI_Reduce
  □ 0.00 __module_fmm_framework_NMOD_constrain_periodic
  ■ 0.20 __energies_NMOD_energy_cons
  ■ 0.01 benchmarking NMOD gather particle diag
  □ 0.00                                   ry
  □ 0.00
  □ 0.00
  □ 0.00
  □ 0.00

0.000000        26.115949 (20.725664%)        126.007776
```

```
Peer distribution
System tree   Topology 0   Topology 1

0.000000            75.882019              100.000000
0.034405   0.051008 +/- 0.003833           0.059240
```

Which problem?

Where in the program?

On which process?

Online Access Interface

- The Online Access Interface (OA) allows to:
  - Retrieve measured performance data while application is still running
  - Control application execution, interrupt between phases
  - Configure and re-configure measurement settings
  - Successive measurement iterations with refined settings
- Designed according to Periscope‘s analysis method
  - Multiple performance experiments within one run
  - Remote analysis with measurements acquisition over sockets
  - Faster analysis: one iteration of the time loop could be enough
- May induce new analysis methods

Releases

- Early preview release at SC’10
- Score-P BOF at SC’11
- Latest release: V 1.1.1

Future Features and Management

- Scalability to maximum available CPU core count
- Support for OpenCL, and OpenACC
- Support for sampling, binary instrumentation
- Support for new programming models, e.g. PGAS
- Support for new architectures

- Ensure a single official release version at every time which will always work with the tools
- Allow experimental versions for new features or research

- Open for new partners after SILC funding period
- Commitment to joint long-term cooperation

- Future integration in Open MPI releases

Long-term Cooperation, Other Projects

- SILC (2009 – 2011)
  - Initial version
- PRIMA (2009 – 2012)
  - Integration with TAU
- H4H (2010 – 2013)
  - Heterogeneous architectures
- HOPSA (EU-Russia project, 2011-2012)
  - Integration with system monitoring
- LMAC (2011 – 2014)
  - Evolution of Score-P, performance dynamics
- GASPI (2011-2014)
  - PGAS, performance API, tool support

GEFÖRDERT VOM
Bundesministerium
für Bildung
und Forschung

DEPARTMENT OF ENERGY
UNITED STATES OF AMERICA

GEFÖRDERT VOM
Bundesministerium
für Bildung
und Forschung

GEFÖRDERT VOM
Bundesministerium
für Bildung
und Forschung

Conclusions

- Common measurement part is community effort
  - Use released resources for analysis
- Unite file formats, open for adoption
- Online access interface, open for adoption
- Scalability/flexibility/overhead limitations addressed
- Easy to extend due to layered architecture
- Robust due to extensive and continuous testing
- Long-term commitment
  - Partners have extensive history of collaboration

SILC & PRIMA Team

- Dieter an Mey, Scott Biersdorf, Kai Diethelm, Dominic Eschweiler, Markus Geimer, Michael Gerndt, Houssam Haitof, Rene Jäkel, Koutaiba Kassem, Andreas Knüpfer, Daniel Lorenz, Suzanne Millstein, Bernd Mohr, Yury Oleynik, Peter Philippen, Christian Rössel, Pavel Saviankou, Dirk Schmidl, Sameer Shende, Wyatt Spear, Ronny Tschüter, Michael Wagner, Bert Wesarg, Felix Wolf, Brian Wylie