VIRTUAL INSTITUTE – HIGH PRODUCTIVITY SUPERCOMPUTING

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

VI-HPS Team

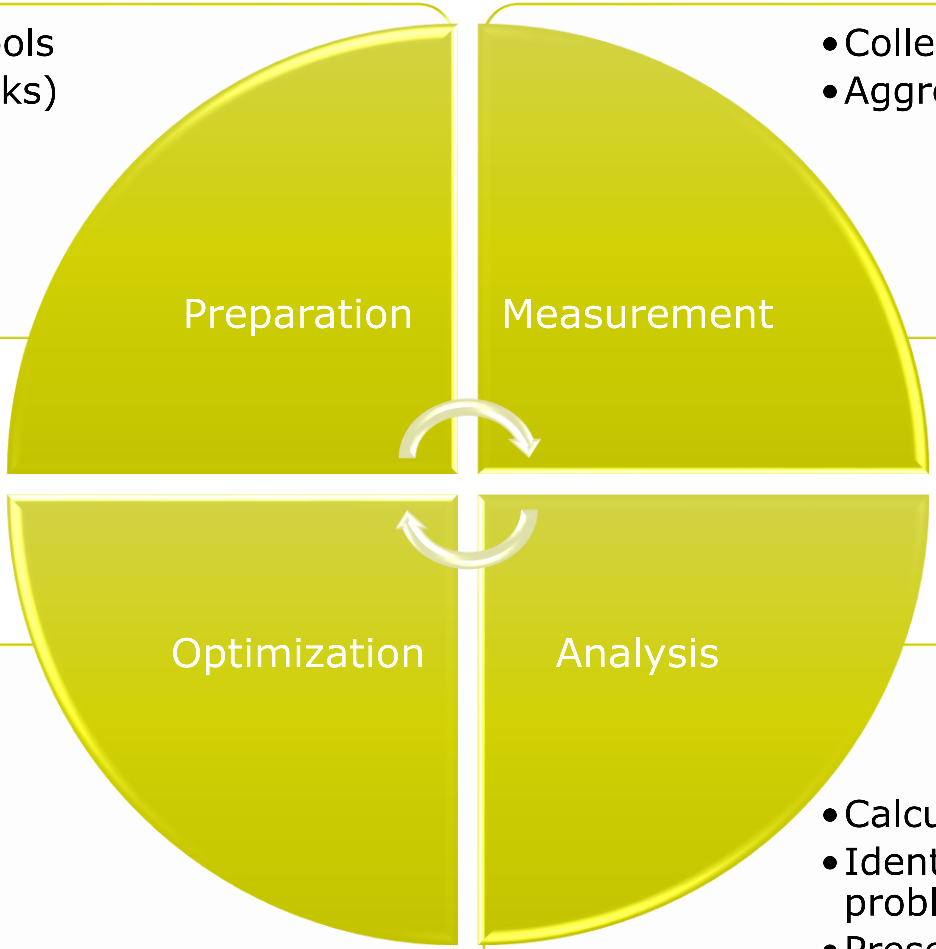
Score-P
Scalable performance measurement infrastructure for parallel codes

arm
JÜLICH Forschungszentrum
lrz Leibniz Supercomputing Centre
TECHNISCHE UNIVERSITÄT DARMSTADT
Technische Universität München
UNIVERSITY OF OREGON
THE UNIVERSITY of TENNESSEE
Barcelona Supercomputing Center Centro Nacional de Supercomputación
Lawrence Livermore National Laboratory
RWTHAACHEN UNIVERSITY
TECHNISCHE UNIVERSITÄT DRESDEN
UNIVERSITÉ DE VERSAILLES ST-QUENTIN-EN-YVELINES
Universität Stuttgart

Performance engineering workflow

- Prepare application with symbols
- Insert extra code (probes/hooks)

- Collection of performance data
- Aggregation of performance data



- Modifications intended to eliminate/reduce performance problem

- Calculation of metrics
- Identification of performance problems
- Presentation of results

Fragmentation of tools landscape

- Several performance tools co-exist
  - Separate 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

Score-P project idea

- 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 funded by BMBF
- Close collaboration PRIMA project funded by DOE

GEFÖRDERT VOM
Bundesministerium
für Bildung
und Forschung

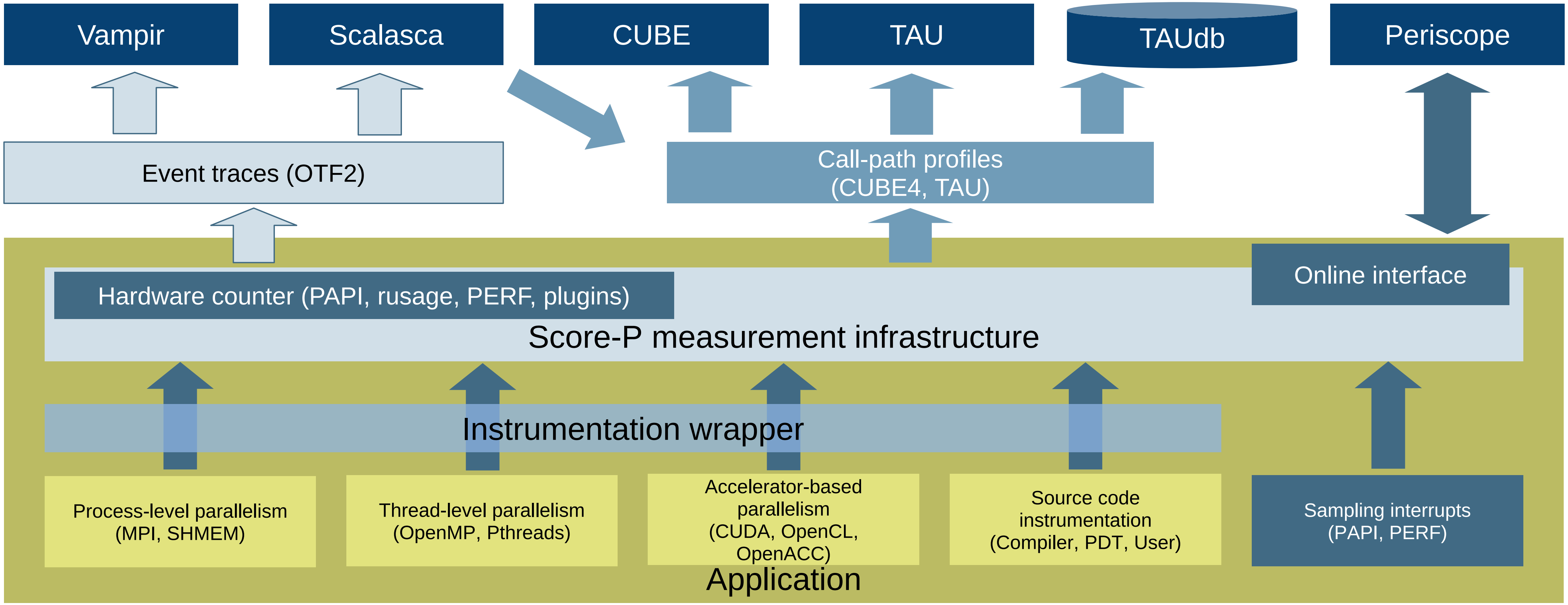
Partners

- Forschungszentrum Jülich, Germany
- Gesellschaft für numerische Simulation mbH Braunschweig, Germany
- RWTH Aachen, Germany
- Technische Universität Darmstadt, Germany
- Technische Universität Dresden, Germany
- Technische Universität München, Germany
- University of Oregon, Eugene, USA

Design goals

- Functional requirements
  - Generation of call-path profiles and event traces
  - Using direct instrumentation and sampling
  - Flexible measurement without re-compilation
  - Recording time, visits, communication data, hardware counters
  - Access and reconfiguration also at runtime
  - Support for MPI, SHMEM, OpenMP, Pthreads, CUDA, OpenCL, OpenACC and their valid combinations
  - Highly scalable I/O
- Non-functional requirements
  - Portability: all major HPC platforms
  - Scalability: petascale
  - Low measurement overhead
  - Robustness
  - Open Source: 3-clause BSD license

Score-P overview


Vampir

Scalasca

CUBE

TAU

TAUdb

Periscope

Event traces (OTF2)

Call-path profiles (CUBE4, TAU)

Hardware counter (PAPI, rusage, PERF, plugins)

Online interface

Score-P measurement infrastructure

Instrumentation wrapper

Process-level parallelism (MPI, SHMEM)

Thread-level parallelism (OpenMP, Pthreads)

Accelerator-based parallelism (CUDA, OpenCL, OpenACC)

Source code instrumentation (Compiler, PDT, User)

Sampling interrupts (PAPI, PERF)

Application

Future features and management

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

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

- Commitment to joint long-term cooperation
  - Development based on meritocratic governance model
  - Open for contributions and new partners

Hands-on:
NPB-MZ-MPI / BT

Score-P

Performance analysis steps

- 0.0 Reference preparation for validation

- 1.0 Program instrumentation
- 1.1 Summary measurement collection
- 1.2 Summary analysis report examination

- 2.0 Summary experiment scoring
- 2.1 Summary measurement collection with filtering
- 2.2 Filtered summary analysis report examination

- 3.0 Event trace collection
- 3.1 Event trace examination & analysis

Recap: Local installation (Isambard)

- Select appropriate PrgEnv (cray or gnu: gnu suggested/assumed)

```
% module switch PrgEnv-cray PrgEnv-gnu
```

- VI-HPS tools not yet installed system-wide
  - Add extra module path
  - Required for each shell session

```
% module use /projects/bristol/vi-hps/modules
% module load scalasca                       # implicitly loads scorep and cube
```

- Copy tutorial sources to your working directory (e.g., $HOME)

```
% cd $HOME
% tar zxvf /projects/bristol/vi-hps/tutorial/NPB3.3-MZ-MPI.tar.gz
% cd NPB3.3-MZ-MPI
```

Recap: Local installation (Archer)

- Select appropriate PrgEnv (cray, gnu, or intel: gnu suggested/assumed)

```
% module switch PrgEnv-cray PrgEnv-gnu
```

- Latest/recent versions of VI-HPS tools not yet installed system-wide
  - Add extra module path
  - Required for each shell session

```
% module use /home/y07/y07/scalasca/modules
% module load scalasca                       # implicitly loads scorep and cube
```

- Copy tutorial sources to your working directory ($WORK)

```
% cd $WORK
% tar zxvf /work/y14/shared/tutorial/NPB3.3-MZ-MPI.tar.gz
% cd NPB3.3-MZ-MPI
```

NPB-MZ-MPI / BT instrumentation

```
#---------------------------------------------------------------
# The Fortran compiler used for MPI programs
#---------------------------------------------------------------
#MPIF77 = ftn

# Alternative variants to perform instrumentation
...
MPIF77 = scorep --user  ftn

# This links MPI Fortran programs; usually the same as ${MPIF77}
FLINK   = $(MPIF77)
...
```

- Edit config/make.def to adjust build configuration
  - Modify specification of compiler/linker: MPIF77

Uncomment the Score-P compiler wrapper specification

NPB-MZ-MPI / BT instrumented build

```
% make clean

% make bt-mz CLASS=C NPROCS=16
cd BT-MZ; make CLASS=C NPROCS=16 VERSION=
make: Entering directory 'BT-MZ'
cd ../sys; cc  -o setparams setparams.c -lm
../sys/setparams bt-mz 16 C
scorep --user ftn -c  -O3 -fopenmp bt.f
 [...]
cd ../common;  scorep --user ftn -c  -O3 -fopenmp timers.f
 [...]
scorep --user ftn -O3 -fopenmp -o ../bin.scorep/bt-mz_C.16 \
bt.o initialize.o exact_solution.o exact_rhs.o set_constants.o \
adi.o rhs.o zone_setup.o x_solve.o y_solve.o exch_qbc.o \
solve_subs.o z_solve.o add.o error.o verify.o mpi_setup.o \
../common/print_results.o ../common/timers.o
Built executable ../bin.scorep/bt-mz_C.16
make: Leaving directory 'BT-MZ`
```

- Return to root directory and clean-up
- Re-build executable using Score-P compiler wrapper

Measurement configuration: scorep-info

```
% scorep-info config-vars --full
SCOREP_ENABLE_PROFILING
  Description: Enable profiling
 [...]
SCOREP_ENABLE_TRACING
  Description: Enable tracing
 [...]
SCOREP_TOTAL_MEMORY
  Description: Total memory in bytes for the measurement system
 [...]
SCOREP_EXPERIMENT_DIRECTORY
  Description: Name of the experiment directory
 [...]
SCOREP_FILTERING_FILE
  Description: A file name which contain the filter rules
 [...]
SCOREP_METRIC_PAPI
  Description: PAPI metric names to measure
 [...]
SCOREP_METRIC_RUSAGE
  Description: Resource usage metric names to measure
 [... More configuration variables ...]
```

- Score-P measurements are configured via environmental variables

Summary measurement collection

```
% cd bin.scorep
% cp ../jobscript/isambard/scorep.pbs .
% vim scorep.pbs

# Score-P measurement configuration
export SCOREP_EXPERIMENT_DIRECTORY=scorep_bt-mz_sum
#export SCOREP_FILTERING_FILE=../config/scorep.filt
#export SCOREP_METRIC_PAPI=PAPI_TOT_INS,PAPI_FP_INS
#export SCOREP_METRIC_RUSAGE=ru_stime
#export SCOREP_METRIC_RUSAGE_PER_PROCESS=ru_maxrss
#export SCOREP_TOTAL_MEMORY=90M
#export SCOREP_ENABLE_TRACING=true

# Run the application
aprun -n $NPROCS -d $OMP_NUM_THREADS $EXE

% qsub scorep.pbs
```

- Change to the directory containing the new executable before running it with the desired configuration
- Check settings

Leave these lines commented out for the moment

- Submit job

Summary measurement collection

```
% less scorep_mzmpibt.o<job_id>

 NAS Parallel Benchmarks (NPB3.3-MZ-MPI) - BT-MZ MPI+OpenMP \
>Benchmark

 Number of zones:  16 x  16
 Iterations: 200    dt:   0.000100
 Number of active processes:    16

 Use the default load factors with threads
 Total number of threads:    128  (  8.0 threads/process)

 Calculated speedup =    127.77

 Time step    1

 [... More application output ...]
```

- Check the output of the application run

BT-MZ summary analysis report examination

```
% ls
bt-mz_C.16  scorep_mzmpibt.o<job_id> scorep_bt-mz_sum
% ls scorep_bt-mz_sum
MANIFEST.md profile.cubex  scorep.cfg









% cube scorep_bt-mz_sum/profile.cubex

       [CUBE GUI showing summary analysis report]
```

- Creates experiment directory including
  - A brief content overview (MANIFEST.md)
  - A record of the measurement configuration (scorep.cfg)
  - The analysis report that was collated after measurement (profile.cubex)
- Interactive exploration with Cube

Hint:
Copy ‘profile.cubex’ to local system (laptop) using ‘scp’ to improve responsiveness of GUI

Further information

- Community instrumentation & measurement infrastructure
  - Instrumentation (various methods)
  - Basic and advanced profile generation
  - Event trace recording
  - Online access to profiling data
- Available under 3-clause BSD open-source license
- Documentation & Sources:
  - http://www.score-p.org
- User guide also part of installation:
  - <prefix>/share/doc/scorep/{pdf,html}/
- Support and feedback: support@score-p.org
- Subscribe to news@score-p.org, to be up to date