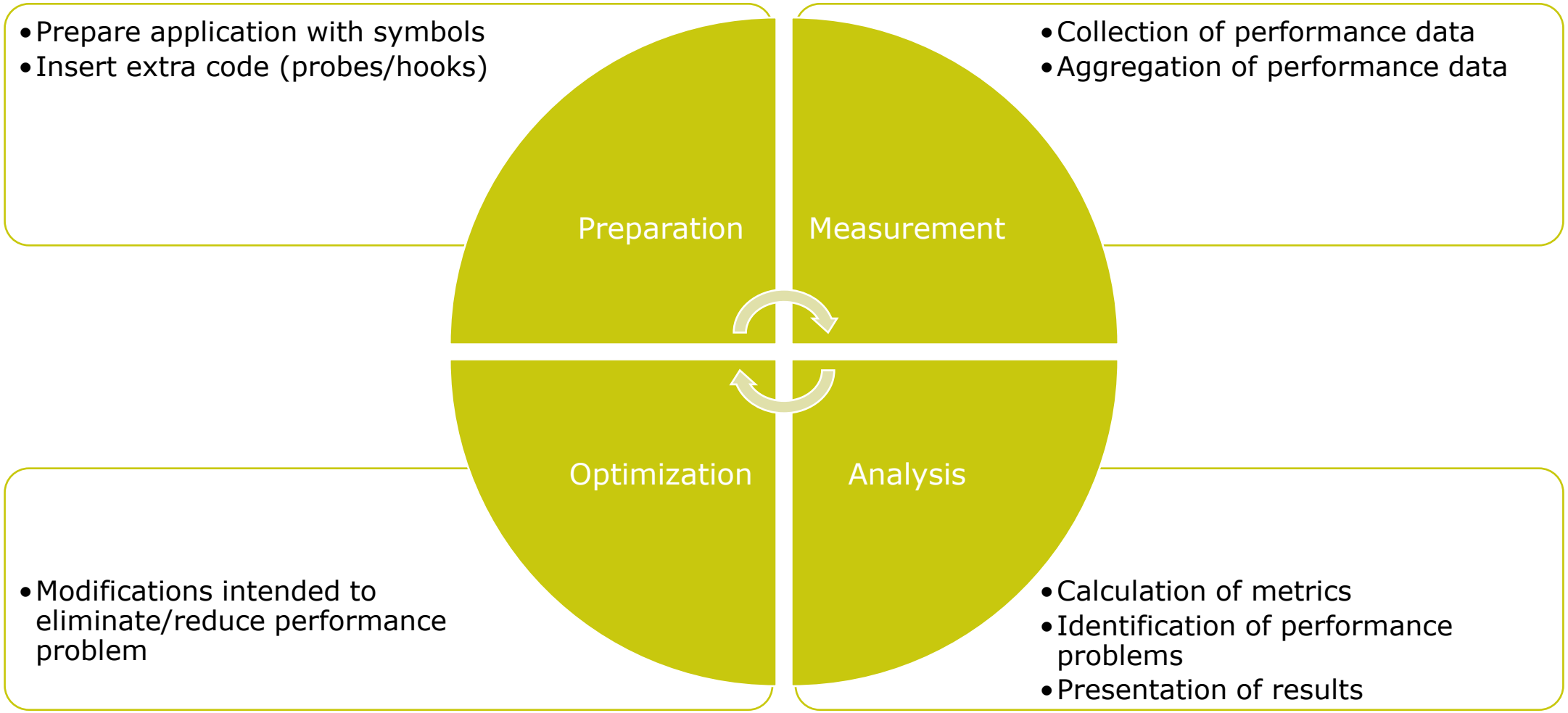
Score-P – A Joint Performance Measurement Run-Time Infrastructure for Periscope, Scalasca, TAU, and Vampir

VI-HPS Team

Score-P
Scalable performance measurement infrastructure for parallel codes

allinea
JÜLICH FORSCHUNGSZENTRUM
RWTH AACHEN UNIVERSITY
TECHNISCHE UNIVERSITÄT DRESDEN
UNIVERSITÉ DE VERSAILLES ST-QUENTIN-EN-YVELINES
Universität Stuttgart
Barcelona Supercomputing Center
Lawrence Livermore National Laboratory
TECHNISCHE UNIVERSITÄT DARMSTADT
Technische Universität München
UNIVERSITY OF OREGON
THE UNIVERSITY of TENNESSEE

Performance engineering workflow



Why Score-P? 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

- Performance measurement infrastructure and runtime

- Prepared applications provide measurement data in several ways:
  - Call-path profiling: CUBE4 data format used for data exchange
  - Event-based tracing: OTF2 data format used for data exchange
  - Online profiling: In conjunction with the Periscope Tuning Framework

- Using direct instrumentation and/or sampling

- Supported parallel paradigms:
  - Multi-process: MPI, SHMEM
  - Thread-parallel: OpenMP, Pthreads
  - Accelerator-based: CUDA, OpenCL, OpenACC

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

Score-P (cont.)

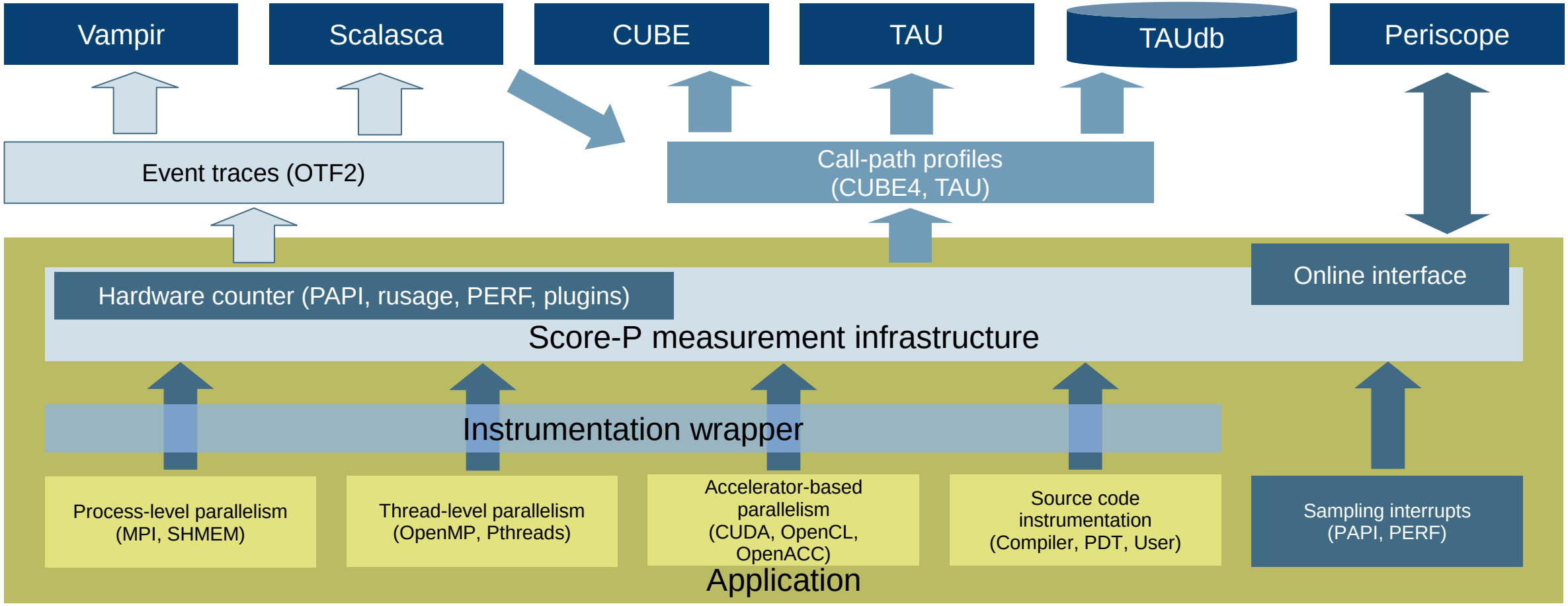
- Developer perspective:
  - Save manpower by sharing development resources
  - Invest in new analysis functionality and scalability
  - Save efforts for maintenance, testing, porting, support, training
- User perspective:
  - Single learning curve
  - Single installation, fewer version updates
  - Interoperability and data exchange
- Open Source; portable and scalable to all major HPC systems
  - Commitment to joint long-term cooperation
  - Development based on meritocratic governance model
  - Open for contributions and new partners
- Initial project funded by BMBF
  - Close collaboration with 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

Score-P overview



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

- VI-HPS tools installed system-wide
  - Load correct module to add local tool installations to $PATH
    - Three modules required for different parts of the workshop:
      scorep: instrumentation, scoring
      cube: graphical examination of measurements/analysis
      scalasca: automatic trace analysis
  - Required for each shell session

```
% module scorep [cube scalasca]
```

-

```
% cd $SCRATCH/NPB-3.3-MZ-MPI
% ls -F
BT-MZ/   Makefile    README.install     SP-MZ/    config/      sys/
LU-MZ/   README      README.tutorial    bin/      common/      jobscript/
```

NPB-MZ-MPI / BT instrumentation

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

# Alternative variants to perform instrumentation
...
MPIF77 = scorep --user ftn
...
#MPIF77 = $(PREP) 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: ftn

Uncomment the Score-P compiler wrapper specification

NPB-MZ-MPI / BT instrumented build

```
% make clean

% make bt-mz CLASS=B NPROCS=8
cd BT-MZ; make CLASS=B NPROCS=8 VERSION=
make: Entering directory 'BT-MZ'
cd ../sys; cc  -o setparams setparams.c -lm
../sys/setparams bt-mz 8 B
scorep --user ftn -c  -O3 -openmp bt.f
 [...]
cd ../common;  scorep --user ftn -c  -O3 \
-openmp timers.f
scorep --user ftn -O3 -openmp \
-o ../bin.scorep/bt-mz_B.8 \
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_B.8
make: Leaving directory 'BT-MZ'
```

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

Summary measurement collection

```
% cd bin.scorep
% cp ../jobscript/cori-p1/scorep.sbatch.B.8 .
% less scorep.sbatch.B.8

[...]
# Score-P measurement configuration
export SCOREP_EXPERIMENT_DIRECTORY=\
scorep_bt-mz_${CLASS}.${PROCS}x${OMP_NUM_THREADS}.${SLURM_JOB_ID}
#export SCOREP_FILTERING_FILE=../config/scorep.filt
#export SCOREP_TOTAL_MEMORY=32M
#export SCOREP_METRIC_PAPI=PAPI_TOT_INS,PAPI_TOT_CYC
#export SCOREP_ENABLE_TRACING=true
[...]


% sbatch scorep.sbatch.B.8
```

- 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

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

```
% less scorep-B.8-<job_id>.out

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

 Number of zones:   8 x   8
 Iterations: 200    dt:   0.000300
 Number of active processes:    16

 Use the default load factors with threads
 Total number of threads:    32  (  4.0 threads/process)

 Calculated speedup =     31.52

 Time step    1

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

- Check the output of the application run

BT-MZ summary analysis report examination

```
% ls -1F
bt-mz_B.8
scorep.sbatch.B.8
scorep-B.8-<jobid>.err
scorep-B.8-<jobid>.out
scorep_bt-mz_B.8x7.<jobid>/

% ls scorep_bt-mz_B.8x7.<jobid>
profile.cubex  scorep.cfg





[% module load cube # if not already done]
% cube scorep_bt-mz_B.8x7.<jobid>/profile.cubex

       [CUBE GUI showing summary analysis report]
```

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