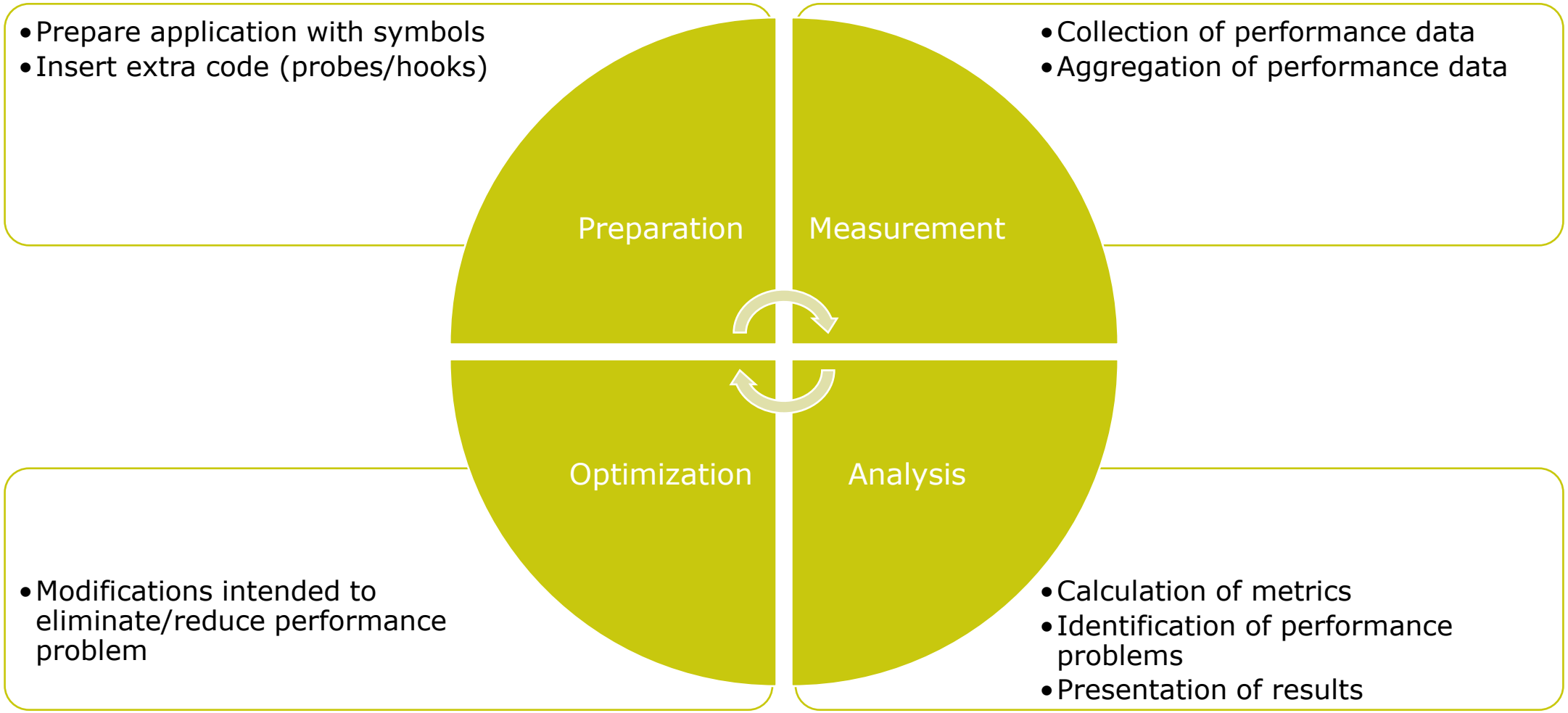
Score-P – A Joint Performance Measurement Run-Time Infrastructure for Periscope, Scalasca, TAU, and Vampir: Basic Usage

VI-HPS Team

Performance engineering workflow



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 functionality

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

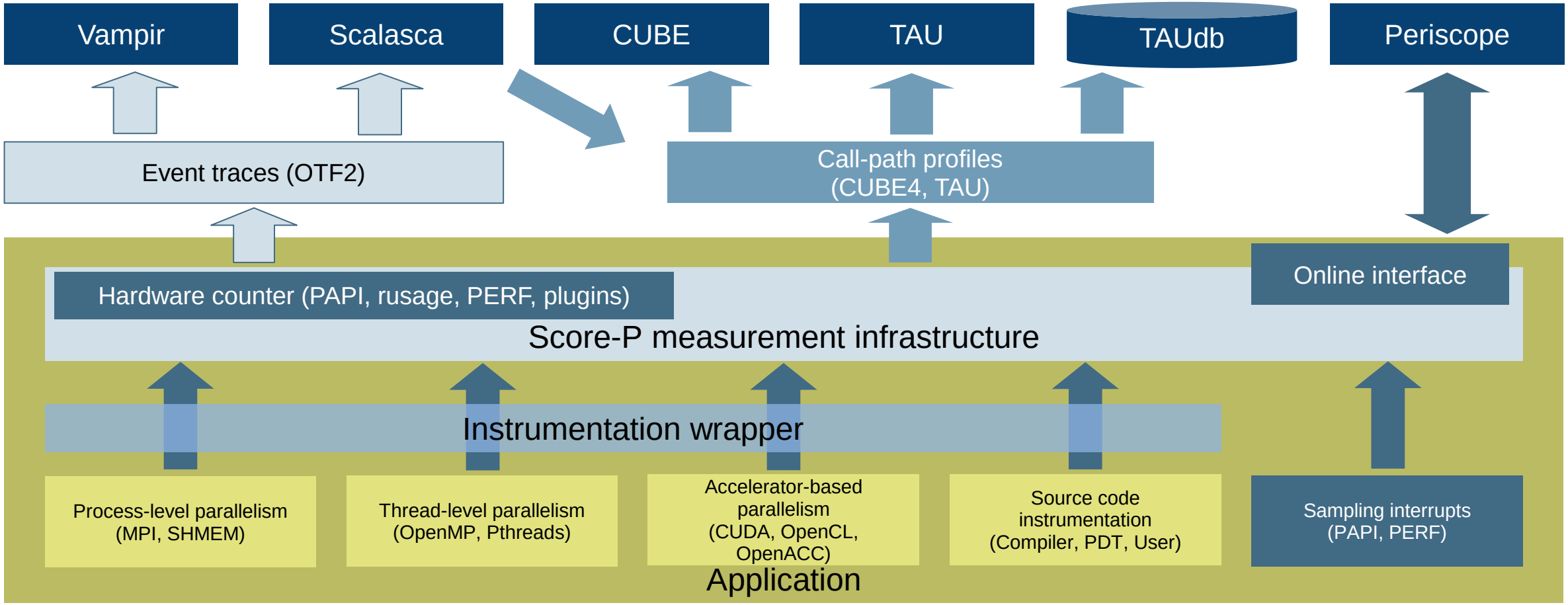
- Instrumentation (various methods)
- Flexible measurement without re-compilation:
  - Basic and advanced profile generation
  - Event trace recording
  - Online access to profiling data

- MPI/SHMEM, OpenMP/Pthreads, and hybrid parallelism (and serial)
- Enhanced functionality (CUDA, OpenCL, OpenACC, highly scalable I/O)

Design goals

- Functional requirements
  - Generation of call-path profiles and event traces
  - Using direct instrumentation and sampling
  - 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
- Non-functional requirements
  - Portability: all major HPC platforms
  - Scalability: petascale
  - Low measurement overhead
  - Robustness
  - Open Source: 3-clause BSD license

Score-P overview



Hands-on:
NPB-MZ-MPI / BT

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
  - Ensure that desired compiler and MPI modules (toolchain) are loaded first

```
% module list
Currently Loaded Modulefiles:
1) DEVELOP     2) intel/16.0  3) openmpi/1.10.4
```

Alternatively switch compilers (gcc) and/or MPI (intelmpi) …

  - Load correct module to add local tool installations to $PATH (required for each shell session)

```
% module load UNITE
UNITE loaded
% module load scorep/3.1-beta-openmpi-intel-papi
scorep/3.1-beta-openmpi-intel-papi loaded
```

… but then you **must** specify the corresponding versions of tools!

- Move to working directory with tutorial exercise

```
% cd $WORK/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 = mpif77

# Alternative variants to perform instrumentation
...
MPIF77 = scorep --user  mpif77
...
#MPIF77 = $(PREP) mpif77

# 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 compiler wrapper specification

NPB-MZ-MPI / BT instrumented build

```
% make clean

% make bt-mz CLASS=C NPROCS=8
cd BT-MZ; make CLASS=C NPROCS=8 VERSION=
make: Entering directory 'BT-MZ'
cd ../sys; cc -o setparams setparams.c -lm
../sys/setparams bt-mz 8 B
scorep --user mpif77 -c  -O3 -fopenmp bt.f
 [...]
cd ../common; scorep --user mpif77 -c  -O3 -qopenmp timers.f
scorep --user mpif77 -O3 -qopenmp -o ../bin.scorep/bt-mz_C.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_C.8
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/claix/scorep.lsf .
% vi scorep.lsf

[...]
export SCOREP_EXPERIMENT_DIRECTORY=scorep_bt-mz_sum
[...]

% bsub < scorep.lsf
```

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

Leave other lines commented out for the moment

- Submit job

Summary measurement collection

```
% less mzmpibt_scorep.<job_id>

 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:    8

 Use the default load factors with threads
 Total number of threads:    48   (  6.0 threads/process)

 Calculated speedup =   47.97

 Time step    1

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

- Check the output of the application run

BT-MZ summary analysis report examination

```
% ls
bt-mz_C.8  mzmpibt-<job_id>.out    scorep_bt-mz_sum
% ls scorep_bt-mz_sum
profile.cubex  scorep.cfg
```

```
% module load cube/4.3.4-gnu
% cube scorep_bt-mz_sum/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