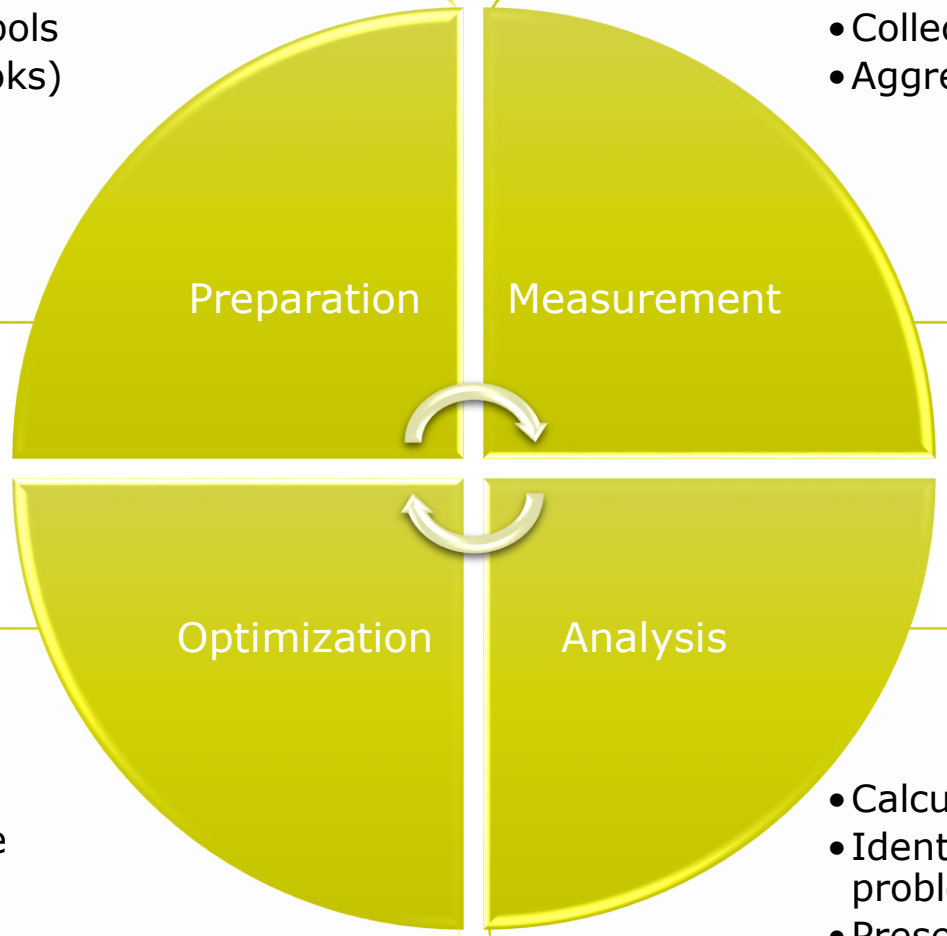
Score-P – A Joint Performance Measurement Run-Time Infrastructure for Scalasca, TAU, and Vampir

VI-HPS Team

Score-P
Scalable performance measurement infrastructure for parallel codes

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

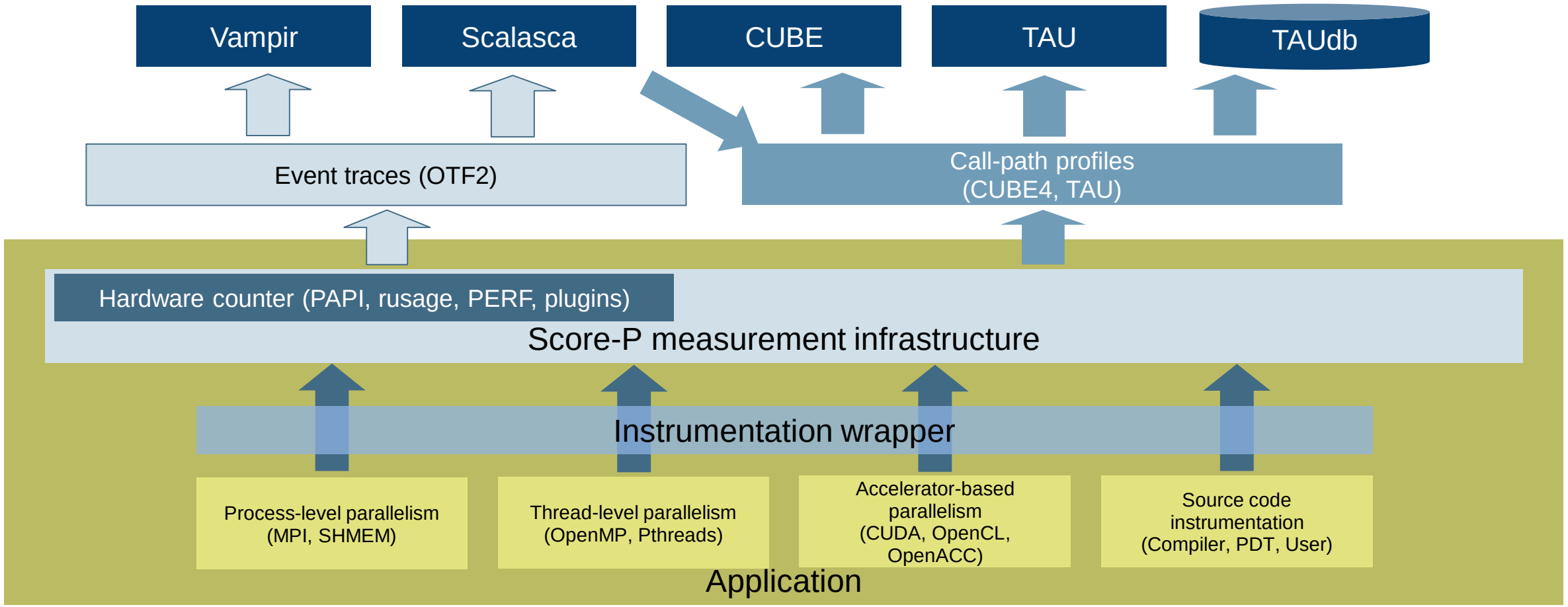
Score-P

- Infrastructure for instrumentation and performance measurements
- Instrumented application can be used to produce several results:
  - 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
- Supported parallel paradigms:
  - Multi-process: MPI, SHMEM
  - Thread-parallel: OpenMP, Pthreads
  - Accelerator-based: CUDA, OpenCL, OpenACC

- Open Source; portable and scalable to all major HPC systems
- Initial project funded by BMBF
- Close collaboration with PRIMA project funded by DOE

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

Score-P overview



Hands-on: TeaLeaf_CUDA

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

Local installation (JUWELS Booster)

- Set account and default environment (NVHPC + ParaStationMPI) via helper script:

```
% source $PROJECT_training2123/setup.sh
```

- Load the modules for the tool environment:

```
% module load Score-P/7.1 CubeGUI/4.6
```

- Copy tutorial sources to your WORK directory (or your personal workspace)
  - Only required if not done already (for opening exercise)

```
% cd $SCRATCH_training2123/$USER
% tar zxvf  $PROJECT_training2123/examples/tea_leaf.tar.gz
% cd TeaLeaf_CUDA
```

TeaLeaf_CUDA: Makefile

```
#Crown Copyright 2014 AWE
#
# This file is part of TeaLeaf.
#
# TeaLeaf is free software...
#
# Agnostic, platform independent Makefile for the TeaLeaf benchmark code.
# It is not meant to be clever in any way, just a simple build script.
#
# this works as well:-
#
# make COMPILER=PGI [OPENMP=1]
#

...

#PREP=scorep --cuda

MPI_COMPILER=$(PREP) mpifort
C_MPI_COMPILER=$(PREP) mpicc
# No preposition for CXX_MPI_COMPILER!
CXX_MPI_COMPILER=mpic++
NVCC=$(PREP) nvcc -ccbin $(CXX_MPI_COMPILER)

...
```

Specify the suite of compilers (and optionally OpenMP)

No instrumentation by default

Uncomment or set PREP to instrumenter preposition

Instrumenting tea_leaf

```
% make COMPILER=PGI PREP="scorep --cuda"

scorep --cuda mpif90 -fastsse -gopt -Mipa=fast    -g -c data.f90 -o data.o
[…]
mpicc -fastsse -gopt -Mipa=fast    -c -g -c timer_c.c -o timer_c.o
scorep --cuda nvcc -ccbin mpicxx -I/p/software/juwelsbooster/stages/2022/software/CUDA/11.5/include \
 -std=c++14 -gencode arch=compute_80,code=sm_80 -restrict -Xcompiler "-fastsse -gopt -Mipa=fast  \
 -c -g" -DNO_ERR_CHK -O3 -c cuda_errors.cu -o cuda_errors.o
[…]
scorep --cuda mpif90 -fastsse -gopt -Mipa=fast    -g        \
data.o definitions.o global_mpi.o tea.o report.o timer.o parse.o read_input.o initialise_chunk.o \
build_field.o update_halo.o start.o generate_chunk.o initialise.o field_summary.o calc_dt.o \
timestep.o set_field.o tea_leaf_common.o tea_leaf_cg.o tea_leaf_cheby.o tea_leaf_ppcg.o \
tea_leaf_jacobi.o tea_solve.o tea_leaf.o diffuse.o  timer_c.o        \
cuda_errors.o cuda_strings.o field_summary_kernel_cuda.o generate_chunk_kernel_cuda.o init_cuda.o \
initialise_chunk_kernel_cuda.o pack_kernel_cuda.o set_field_kernel_cuda.o tea_leaf_kernel_cuda.o \
update_halo_kernel_cuda.o \
-L/p/software/juwelsbooster/stages/2022/software/CUDA/11.5/lib64 \
-lstdc++ -lcudart \
-o bin.scorep/tea_leaf
```

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
 [...]
SCOREP_CUDA_ENABLE
  Description: CUDA measurement features
 [... More configuration variables ...]
```

- Score-P measurements are configured via environmental variables

Required for CUDA measurements. [yes|default] recommended to start with.

Summary measurement collection

```
% cd bin.scorep
% cp ../jobscripts/juwelsbooster/scorep.sbatch .
% cat scorep.sbatch
...
# Score-P measurement configuration
configuration
export SCOREP_CUDA_ENABLE=default
export SCOREP_CUDA_BUFFER=48M

export SCOREP_EXPERIMENT_DIRECTORY=scorep-tea_leaf-8
#export SCOREP_FILTERING_FILE=../config/scorep.filter

#export SCOREP_ENABLE_TRACING=true
#export SCOREP_TOTAL_MEMORY=120M

# Run the application
srun ./tea_leaf

% sbatch scorep.sbatch
```

- 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

TeaLeaf_CUDA Reference Execution

```
% cat TeaLeaf_scorep.<job_id>.out

   Tea Version 1.400
       MPI Version
    OpenMP Version
   Task Count:     8

Input read finished.
Using CUDA Kernels

[…]

Solver to use: PPCG
Preconditioner to use: None
Test problem   6 is within   0.1397839E-05% of the expected solution
 This test is considered PASSED
 First step overhead  -0.5253252983093262
 Wall clock     38.01989197731018
```

- Verify the reported execution configuration and that the test execution passed

Compare to previous reference execution without instrumentation

TeaLeaf summary analysis report examination

```
% ls
tea_leaf tea.in TeaLeaf_scorep.<job_id>.out
scorep.sbatch scorep-tea_leaf-8/

% ls scorep-tea_leaf-8
MANIFEST.md  profile.cubex  scorep.cfg









% cube scorep-tea_leaf-8/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