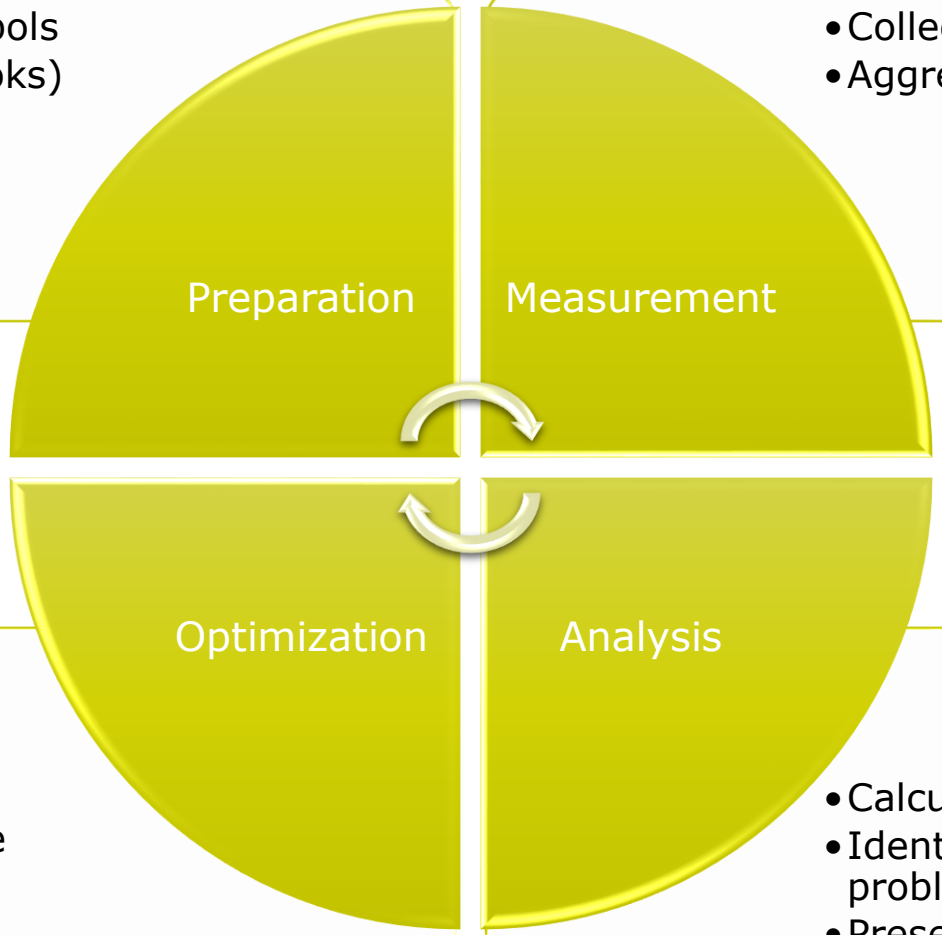
Score-P – A Joint Performance Measurement Run-Time Infrastructure for Periscope, Scalasca, TAU, and Vampir

VI-HPS Team

Score-P
Scalable performance measurement infrastructure for parallel codes

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

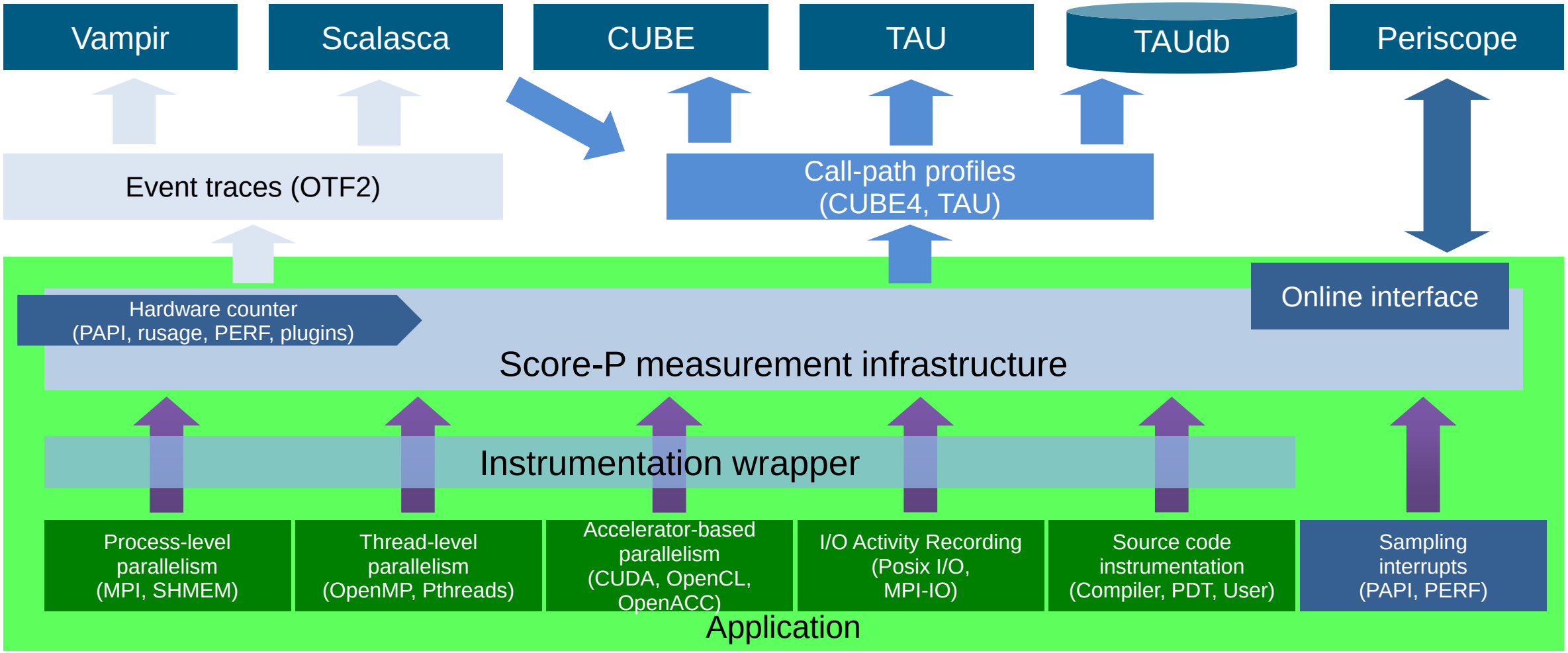
Score-P

- 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



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

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 customisation & 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 (m100)

- Latest/recent versions/combinations of VI-HPS tools not yet installed system-wide
  - Required for each shell session
  - Source the appropriate setup script according to your **compiler-MPI**

```
# module load  cuda gnu spectrum_mpi
% source /m100_work/tra20_TW36/tools/sourceme.scorep-scalasca.gnu-spectrummpi
```

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

```
% cd $CINECA_SCRATCH
% tar zxvf  /m100_work/tra20_TW36/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=GNU [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=GNU OPENMP=1  PREP="scorep --cuda"
scorep --cuda  mpifort -O3 -mcpu=native -mno-float128 -funroll-loops -cpp -fopenmp -c data.f90
[…]
scorep --cuda  mpifort -O3 -mcpu=native -mno-float128 -funroll-loops -cpp -fopenmp -c tea_leaf.f90
scorep --cuda  mpifort -O3 -mcpu=native -mno-float128 -funroll-loops -cpp -fopenmp -c diffuse.f90
scorep --cuda  mpicc -O3 -mcpu=native -mno-float128 -funroll-loops -fopenmp -c timer_c.c
scorep --cuda  nvcc -ccbin mpic++ -restrict -O3 -I$(CUDA_INC) -gencode arch=compute_70,code=sm_70 \
    -Xcompiler "-O3 -mcpu=native -mno-float128 -funroll-loops -fopenmp" -c cuda_errors.cu
[…]
scorep --cuda  nvcc -ccbin mpic++ -restrict -O3 -I$(CUDA_INC) -gencode arch=compute_70,code=sm_70 \
    -Xcompiler "-O3 -mcpu=native -mno-float128 -funroll-loops -fopenmp" -c update_halo_kernel_cuda.cu
mkdir -p bin
scorep --cuda  mpifort -O3 -mcpu=native -mno-float128 -funroll-loops -cpp -fopenmp \
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$(CUDA_LIB) -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
 [... More configuration variables ...]
```

- Score-P measurements are configured via environmental variables

Summary measurement collection

```
% cd bin.scorep
% cp ../jobscript/marconi100/scorep.sbatch .
% cat scorep.sbatch
...
# Score-P measurement configuration
export SCOREP_EXPERIMENT_DIRECTORY=scorep_tea_leaf_sum
#export SCOREP_FILTERING_FILE=../config/scorep.filt
#export SCOREP_METRIC_PAPI=PAPI_TOT_INS,PAPI_TOT_CYC,…
#export SCOREP_METRIC_PAPI_PER_PROCESS=PAPI_L2_TCM
#export SCOREP_METRIC_RUSAGE=ru_stime
#export SCOREP_METRIC_RUSAGE_PER_PROCESS=ru_maxrss
#export SCOREP_TIMER=gettimeofday

# Run the application
export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}
mpirun ./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 tea_leaf.o<job_id>

   Tea Version 1.400
       MPI Version
    OpenMP Version
   Task Count:      8
 Thread Count:      3

Input read finished.
Using CUDA Kernels

[…]

Solver to use: PPCG
Preconditioner to use: None
Test problem   5 is within 0.2741549E-06% of the expected solution
 This test is considered PASSED
```

- 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_leaf.o<job_id>  scorep_tea_leaf_sum/

% ls scorep_tea_leaf_sum
MANIFEST.md  profile.cubex  scorep.cfg









% paraprof scorep_tea_leaf_sum/profile.cubex
% cube scorep_tea_leaf_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 Paraprof & 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