VI-HPS

VIRTUAL INSTITUTE – HIGH PRODUCTIVITY SUPERCOMPUTING

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

BSC Barcelona Supercomputing Center Centro Nacional de Supercomputación

JÜLICH Forschungszentrum

lrz Leibniz Supercomputing Centre of the Bavarian Academy of Sciences and Humanities

RWTH AACHEN UNIVERSITY

TECHNISCHE UNIVERSITÄT DRESDEN

UNIVERSITÉ DE VERSAILLES ST-QUENTIN-EN-YVELINES

Universität Stuttgart

FAU FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

Lawrence Livermore National Laboratory

Linaro

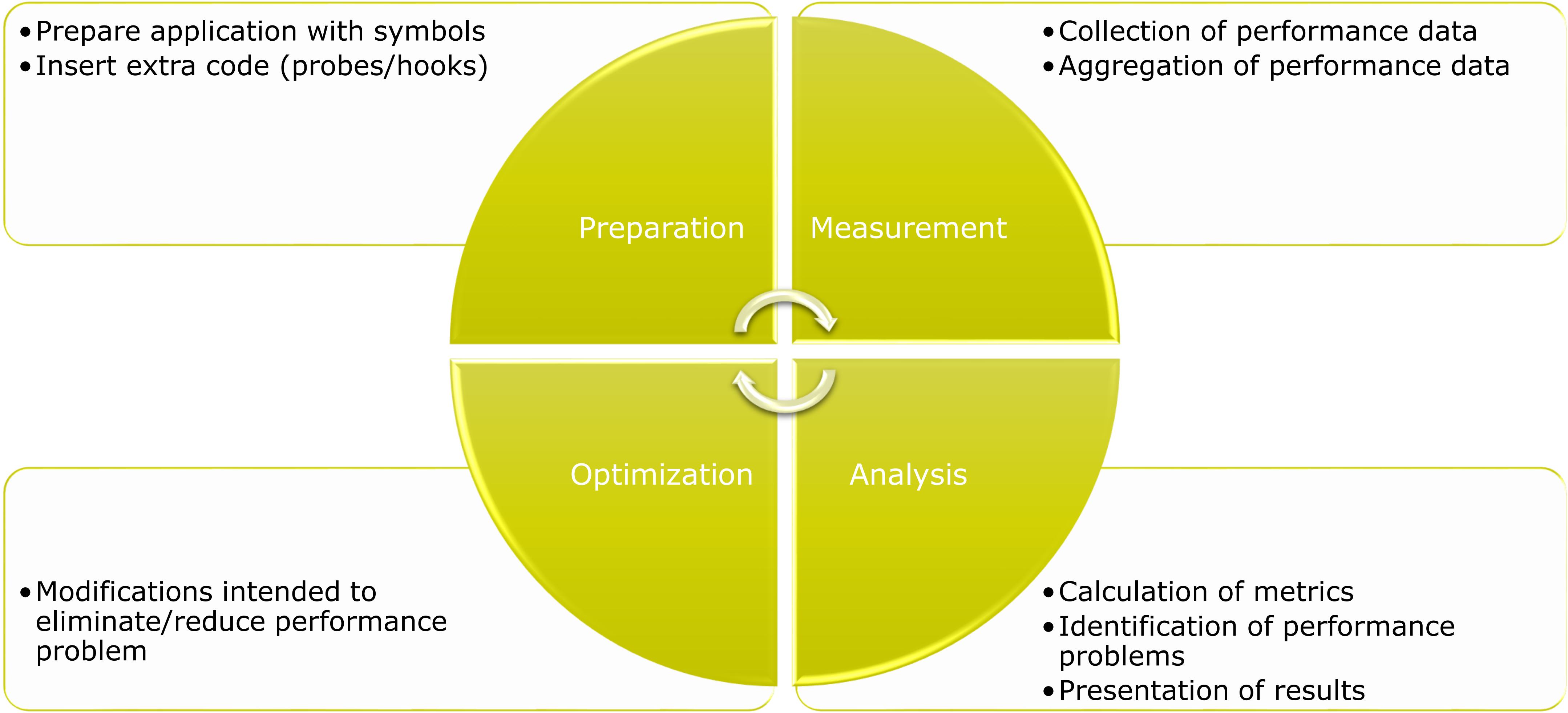
TECHNISCHE UNIVERSITÄT DARMSTADT

TUM Technische Universität München

UNIVERSITY OF OREGON

THE UNIVERSITY of TENNESSEE

Performance engineering workflow



Preparation

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

Measurement

- Collection of performance data
- Aggregation of performance data

Analysis

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

Optimization

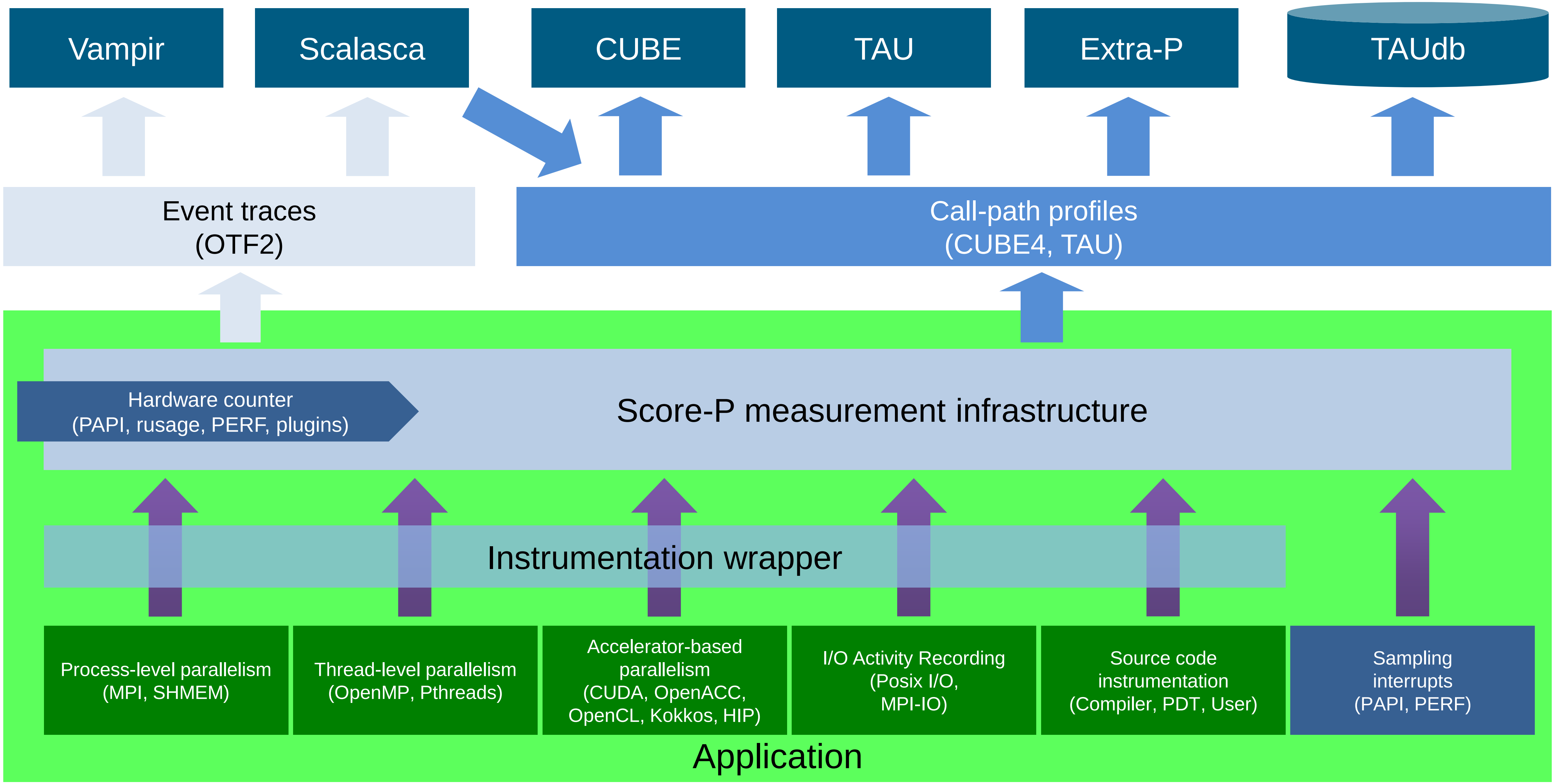
- Modifications intended to eliminate/reduce performance problem

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, HIP

- 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


Vampir
Scalasca
CUBE
TAU
Extra-P
TAUdb
Event traces
(OTF2)
Call-path profiles
(CUBE4, TAU)
Hardware counter
(PAPI, rusage, PERF, plugins)
Score-P measurement infrastructure
Instrumentation wrapper
Process-level parallelism
(MPI, SHMEM)
Thread-level parallelism
(OpenMP, Pthreads)
Accelerator-based
parallelism
(CUDA, OpenACC,
OpenCL, Kokkos, HIP)
I/O Activity Recording
(Posix I/O,
MPI-IO)
Source code
instrumentation
(Compiler, PDT, User)
Sampling
interrupts
(PAPI, PERF)
Application

Hands-on:
CloverLeaf_OpenACC

Score-P

Barcelona Supercomputing Center Centro Nacional de Supercomputación
JÜLICH Forschungszentrum
lrz Leibniz Supercomputing Centre of the Bavarian Academy of Sciences and Humanities
RWTHAACHEN UNIVERSITY
TECHNISCHE UNIVERSITÄT DRESDEN
UNIVERSITÉ DE VERSAILLES ST-QUENTIN-EN-YVELINES
Universität Stuttgart
FAU FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG
Lawrence Livermore National Laboratory
Linaro
TECHNISCHE UNIVERSITÄT DARMSTADT
TUM Technische Universität München
UNIVERSITY OF OREGON
THE UNIVERSITY of TENNESSEE

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 (Turpan)

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

```
% source /tmpdir/vi-hps/opt/setup.sh
```

- Load the modules for the tool environment:

```
% module load  scorep/8.3-nvhpc-openmpi-gpu  cube
```

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

```
% cd $WORK
% tar xf /tmpdir/vi-hps/material/handsons/CloverLeaf_OpenACC.tar.gz
% cd CloverLeaf_OpenACC
```

CloverLeaf_OpenACC: Makefile

```
#Crown Copyright 2012 AWE
#
# This file is part of CloverLeaf.
#
# CloverLeaf is free software...
#
# Agnostic, platform independent Makefile for the CloverLeaf 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 --openacc --cuda --user

MPI_COMPILER=$(PREP) mpifort

# No preposition for C/CXX_MPI_COMPILER!
C_MPI_COMPILER=mpicc
CXX_MPI_COMPILER=mpic++

...
```

Specify the suite of compilers (and optionally OpenMP)

No instrumentation by default

Set/uncomment PREP macro for instrumenter preposition

Instrumenting clover_leaf

Score-P instrumenter options:

--compiler: source code routines (default)

--mpp=mpi: MPI determined by heuristics

--openacc: enable OpenACC

--cuda: enable CUDA

--user: enable Score-P user API (optional)

```
% make clean
% make PREP="scorep --openacc --cuda --user"

mpicc -c timer_c.c
scorep --openacc --cuda --user  mpif90 -O3 -acc=gpu -ta=nvidia \
    data.f90 definitions.f90 pack_kernel.f90 clover.F90 report.f90 timer.f90 \
    parse.f90 read_input.f90 initialise_chunk_kernel.f90 initialise_chunk.f90 build_field.f90 \
    update_tile_halo_kernel.f90 update_tile_halo.f90 update_halo_kernel.f90 update_halo.f90 \
    ideal_gas_kernel.f90 ideal_gas.f90 start.f90 generate_chunk_kernel.f90 generate_chunk.f90 \
    initialise.f90 field_summary_kernel.f90 field_summary.f90 viscosity_kernel.f90 viscosity.f90 \
    calc_dt_kernel.f90 calc_dt.f90 timestep.f90 accelerate_kernel.f90 accelerate.f90 \
    revert_kernel.f90 revert.f90 PdV_kernel.f90 PdV.f90 flux_calc_kernel.f90 flux_calc.f90 \
    advec_cell_kernel.f90 advec_cell_driver.f90 advec_mom_kernel.f90 advec_mom_driver.f90 \
    reset_field_kernel.f90 reset_field.f90 hydro.F90 clover_leaf.F90 visit.f90 \
    timer_c.o \
    -o bin/clover_leaf
```

Mastering build systems

- Hooking up the Score-P instrumenter `scorep` into complex build environments like *Autotools* or *CMake* was always challenging
- Score-P provides convenience wrapper scripts to simplify this
- *Autotools* and *CMake* need the used compiler already in the *configure step,* but instrumentation should not happen in this step, only in the *build step*

```
% SCOREP_WRAPPER=off \
> cmake .. \
> -DCMAKE_C_COMPILER=scorep-nvcc \
> -DCMAKE_CXX_COMPILER=scorep-nvc++ \
> -DCMAKE_Fortran_COMPILER=scorep-nvfortran
```

Disable instrumentation in the *configure step*

Specify the wrapper scripts as the compiler to use

- Allows to pass addition options to the Score-P instrumenter and the compiler via environment variables without modifying the *Makefiles*
- Run `scorep-wrapper --help` for a detailed description and the available wrapper scripts of the Score-P installation

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_OPENACC_ENABLE
  Description: OpenACC measurement features
 [... More configuration variables ...]
```

- Score-P measurements are configured via environmental variables

Required for OpenACC measurements.
[yes|default] recommended to start.
Additional CUDA measurement optional.

Mastering heterogeneous applications

For all available options check:
scorep-info config-vars --full

- Record CUDA application events and device activities

```
% export SCOREP_CUDA_ENABLE=default
```

- Record OpenCL application events and device activities

```
% export SCOREP_OPENCL_ENABLE=api,kernel
```

- Record OpenACC application events

```
% export SCOREP_OPENACC_ENABLE=regions,wait,enqueue
```

  - Can be combined with CUDA if it is a NVIDIA device

```
% export SCOREP_CUDA_ENABLE=kernel,kernel_callsite,idle
```

CloverLeaf_OpenACC summary measurement collection

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

#export SCOREP_EXPERIMENT_DIRECTORY=scorep-clover_leaf-6
#export SCOREP_FILTERING_FILE=../config/scorep.filter
#export SCOREP_ENABLE_TRACING=true
#export SCOREP_TOTAL_MEMORY=250M

# Run the application
mpiexec ./clover_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

CloverLeaf_OpenACC summary measurement execution

```
% cat slurm-<job_id>.out
+ mpiexec ./clover_leaf
MPI rank    0 (0) using device   0/2 on turpancomp0
MPI rank    1 (1) using device   1/2 on turpancomp0
MPI rank    4 (0) using device   0/2 on turpancomp2
MPI rank    5 (1) using device   1/2 on turpancomp2
MPI rank    2 (0) using device   0/2 on turpancomp1
MPI rank    3 (1) using device   1/2 on turpancomp1

    Clover Version 1.300
       MPI Version
   OpenACC Version 201711
   Task Count      6

 Output file clover.out opened. All output will go there.

 Step    1 time   0.0000000 control sound timestep  3.85E-04   1, 1 x  3.26E-04 y  3.26E-04
 Step    2 time   0.0003852 control sound timestep  2.35E-04   1, 1 x  3.26E-04 y  3.26E-04
 Step    3 time   0.0006203 control sound timestep  2.99E-04   1, 1 x  3.26E-04 y  3.26E-04
[...]
 Step   87 time   0.0310703 control sound timestep  3.66E-04   1, 1 x  3.26E-04 y  3.26E-04

 Wall clock     6.195748090744019
 First step overhead   2.0323038101196289E-002
```

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

Compare to previous reference execution without instrumentation

CloverLeaf_OpenACC summary analysis report examination

```
% ls
clover_leaf* clover.in clover.out  slurm-<job_id>.out
scorep.sbatch  scorep-clover_leaf-6/

% ls scorep-clover_leaf-6
MANIFEST.md  profile.cubex  scorep.cfg







% cube scorep-clover_leaf-6/profile.cubex

       [CUBE GUI showing summary analysis report]
```

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

- 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

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:
  - https://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 kept informed