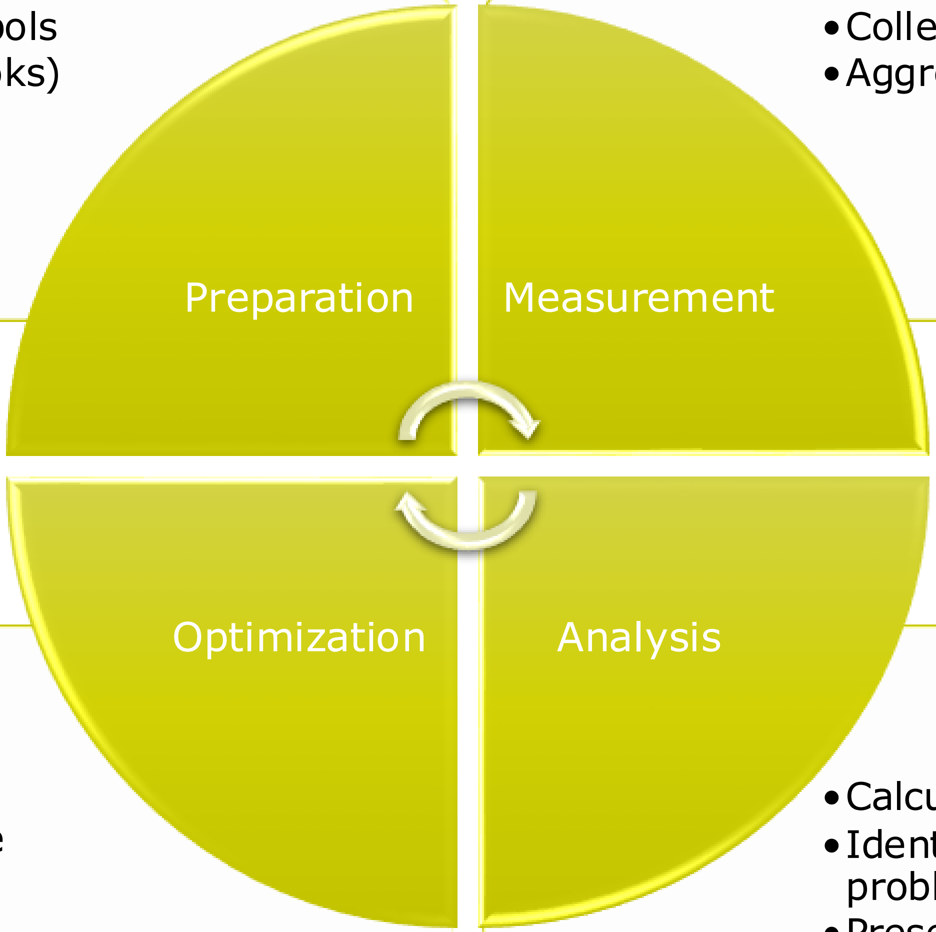
Score-P – A Joint Performance Measurement Run-Time Infrastructure for Scalasca, TAU, and Vampir

VI-HPS Team

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

Hands-on: CloverLeaf_OpenACC

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 (Leonardo)

- Load the modules for the compiler and MPI (NVHPC + OpenMPI) :

```
% module load nvhpc/23.1 openmpi/4.1.4--nvhpc--23.1-cuda-11.8
```

- Load the tools for NVHPC + OpenMPI toolchain via helper script :

```
% source /leonardo/pub/userexternal/bwylie00/tools/nvompi/setup.sh
```

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

```
% cd /leonardo_work/tra24_epicure/
% mkdir -p <your_user_name> ; cd <your_user_name>
% tar xvf /leonardo/pub/userexternal/izhukov0/Hackathon/material/handsons/\
CloverLeaf_OpenACC.tar.gz
% cd CloverLeaf_OpenACC
```

CloverLeaf

- Overview
  - Mini-app for solving compressible Euler equations on a Cartesian grid.
  - Uses an explicit, second-order accurate method with a staggered grid.
- Structure
  - Computation broken into “kernels,” each updating specific grid variables.
  - Minimal control logic for high compiler optimization.
  - Avoids dependencies by updating copies of the mesh.
- Implementations
  - Serial, MPI & OpenMP (default), OpenMP, OpenACC, CUDA, OpenCL, OpenSHMEM, etc.
- More details here: http://uk-mac.github.io/CloverLeaf/

- For our hands-on, we use “clover_bm1024_short.in” which executes 87 steps

Reference run

- Make sure the compiler and MPI (NVHPC + OpenMPI) are loaded:

```
% module load nvhpc/23.1 openmpi/4.1.4--nvhpc--23.1-cuda-11.8
```

- Compile:

```
% cd /leonardo_work/tra24_epicure/<your_user_name>/CloverLeaf_OpenACC
% make
```

- Check that the executable is available / copy the batch script and submit it

```
% cd bin/ ; ls -l
% cp ../jobscripts/reference.sh .
% sbatch reference.sh
```

CloverLeaf_OpenACC reference run

```
% cat slurm-<job_id>.out
+ srun ./clover_leaf
MPI rank    1 (1) using device 1/4 on lrdn1337.leonardo.local
MPI rank    7 (3) using device 3/4 on lrdn1635.leonardo.local
MPI rank    2 (2) using device 2/4 on lrdn1337.leonardo.local
MPI rank    0 (0) using device 0/4 on lrdn1337.leonardo.local
MPI rank    3 (3) using device 3/4 on lrdn1337.leonardo.local
MPI rank    6 (2) using device 2/4 on lrdn1635.leonardo.local
MPI rank    5 (1) using device 1/4 on lrdn1635.leonardo.local
MPI rank    4 (0) using device 0/4 on lrdn1635.leonardo.local

    Clover Version 1.300
       MPI Version
   OpenACC Version 201711
   Task Count      8

 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     13.85142779350281
 First step overhead   0.8581387996673584
```

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

Note “Wall clock” time

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) mpif90

# 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.scorep/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

- Record CUDA application events and device activities

```
% export SCOREP_CUDA_ENABLE=default
```

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

- 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
% cp ../jobscripts/scorep.sbatch.
...
# Score-P measurement configuration
export SCOREP_OPENACC_ENABLE=default
export SCOREP_CUDA_ENABLE=default
export SCOREP_EXPERIMENT_DIRECTORY=scorep_clover_leaf_8_sum


#export SCOREP_FILTERING_FILE=../config/scorep.filter
#export SCOREP_ENABLE_TRACING=true
#export SCOREP_TOTAL_MEMORY=17M

# Run the application
srun ./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
+ srun ./clover_leaf
MPI rank    1 (1) using device 1/4 on lrdn1337.leonardo.local
MPI rank    7 (3) using device 3/4 on lrdn1635.leonardo.local
MPI rank    2 (2) using device 2/4 on lrdn1337.leonardo.local
MPI rank    0 (0) using device 0/4 on lrdn1337.leonardo.local
MPI rank    3 (3) using device 3/4 on lrdn1337.leonardo.local
MPI rank    6 (2) using device 2/4 on lrdn1635.leonardo.local
MPI rank    5 (1) using device 1/4 on lrdn1635.leonardo.local
MPI rank    4 (0) using device 0/4 on lrdn1635.leonardo.local

    Clover Version 1.300
       MPI Version
   OpenACC Version 201711
   Task Count      8

 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     13.94159698486328
 First step overhead   0.8847539424896240
```

- 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_8_sum
MANIFEST.md  profile.cubex  scorep.cfg




% cube scorep_clover_leaf_8_sum/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