Hands-on: *JUWELS Booster* (AMD EPYC Rome + 4 x A100) TeaLeaf_CUDA

VI-HPS Team

Tutorial exercise objectives

- Familiarise with usage of VI-HPS tools
  - complementary tools' capabilities & interoperability
- Prepare to apply tools productively to *your* applications(s)
- Exercise is based on a small portable benchmark code
  - unlikely to have significant optimisation opportunities

- Optional (recommended) exercise extensions
  - analyse performance of alternative configurations
  - investigate effectiveness of system-specific compiler/MPI optimisations and/or placement/binding/affinity capabilities
  - investigate scalability and analyse scalability limiters
  - compare performance on different HPC platforms
  - …

Setup for exercises

- Connect to your training account on JUWELS Booster (with X11-forwarding)

```
% ssh -X <yourid>@juwels-booster.fz-juelich.de
```

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

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

- Copy tutorial sources to your WORK directory

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

Case study: TeaLeaf_CUDA

- HPC mini-app developed by the UK Mini-App Consortium
  - Solves the linear 2D heat conduction equation on a spatially decomposed regular grid using a 5 point stencil with implicit solvers
  - Part of the Mantevo 3.0 suite
  - Available on GitHub: https://uk-mac.github.io/TeaLeaf/

- CUDA-enabled MPI version written in Fortran90, run using default testcase
  - Optional OpenMP (only used during initialization): vary the number of threads for each MPI process
  - Run with 4 MPI tasks-per-node so that each has a dedicated GPU
  - Run on 1 or more nodes to see its strong scaling performance: 2 is sufficient for exercise
  - Experiment with different MPI libraries, compilers & compiler optimisations
  - Experiment with different bindings/affinities of MPI processes and OpenMP threads
  - Experiment with different solvers (and other testcases)

- Provided version of Makefile and sources customized for this tutorial
  - builds **tea_leaf** executable in separate directory when using instrumentation

TeaLeaf_CUDA source directory

```
% ls
Benchmarks/                  generate_chunk.f90                  set_field_kernels_cuda.cu
Makefile                     generate_chunk_kernel_cuda.cu       start.f90
README.md                    global_mpi.f90                      tea.f90
build_field.f90              host_reductions_kernel_cuda.cu      tea.in
calc_dt.f90                  init_cuda.cu                        tea_leaf.f90
config/                      initialise.f90                      tea_leaf_cg.f90
cuda_common.hpp              initialise_chunk.f90                tea_leaf_cheby.f90
cuda_errors.cu               initialise_chunk_kernel_cuda.cu     tea_leaf_common.f90
cuda_strings.cu              jobscripts/                         tea_leaf_kernel_cuda.cu
cuda_strings.hpp             kernel_files/                       tea_leaf_ppcg.f90
data.f90                     makefile.deps                       tea_solve.f90
definitions.f90              pack_kernel_cuda.cu                 timer.f90
diffuse.f90                  parse.f90                           timer_c.c
field_summary.f90            read_input.f90                      timestep.f90
field_summary_kernel_cuda.cu report.f90                          update_halo.f90
ftocmacros.h                 set_field.f90                       update_halo_kernel_cuda.cu
```

25 Fortran90 modules, 1 C module, 10 CUDA modules

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

Building tea_leaf

```
% make COMPILER=PGI
mpif90 -fastsse -gopt -Mipa=fast    -g -c data.f90 -o data.o
[ ... ]
mpif90 -fastsse -gopt -Mipa=fast    -g -c tea_leaf.f90 -o tea_leaf.o
mpif90 -fastsse -gopt -Mipa=fast    -g -c diffuse.f90 -o diffuse.o
mpicc -fastsse -gopt -Mipa=fast    -c -g -c timer_c.c -o timer_c.o
nvcc -ccbin mpicxx -std=c++14 -I/p/software/juwelsbooster/stages/2022/software/CUDA/11.5/include
 -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
[ ... ]
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/tea_leaf
```

TeaLeaf_CUDA jobscript for reference execution

```
% cd bin
% cp ../jobscripts/juwelsbooster/run.sbatch .
% cat run.sbatch

#!/bin/bash
#SBATCH --job-name=TeaLeaf         # Job name
#SBATCH --partition=develbooster   # Job partition
#SBATCH --nodes=2                  # Total number of nodes requested
#SBATCH --gres=gpu:4               # Number of GPUs per node
#SBATCH --ntasks-per-node=4        # Number of MPI tasks per node (one per GPU)
#SBATCH --time=00:05:00            # Max. wall-clock time (hh:mm:ss)
#SBATCH --account=training2123     # Project account to be charged
#SBATCH --output=%x.%j.out         # Output files
#SBATCH --error=%x.%j.out

srun ./tea_leaf

% sbatch run.sbatch
```

- Copy jobscript and check/modify its contents
- then submit

TeaLeaf_CUDA Reference Execution

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

Tea Version    1.400
       MPI Version
   Task Count      8

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

CUDA in rank 1 using NVIDIA A100-SXM4-40GB (0:68:0)
CUDA in rank 3 using NVIDIA A100-SXM4-40GB (0:196:0)
CUDA in rank 2 using NVIDIA A100-SXM4-40GB (0:132:0)
CUDA in rank 0 using NVIDIA A100-SXM4-40GB (0:3:0)
Solver to use: PPCG
Preconditioner to use: None
CUDA in rank 5 using NVIDIA A100-SXM4-40GB (0:68:0)
 Step       1 time   0.0000000 timestep   4.00E-03
CUDA in rank 7 using NVIDIA A100-SXM4-40GB (0:196:0)
CUDA in rank 6 using NVIDIA A100-SXM4-40GB (0:132:0)
CUDA in rank 4 using NVIDIA A100-SXM4-40GB (0:3:0)
[...]
 This test is considered PASSED
 First step overhead  -0.3670756816864014
 Wall clock     33.43737912178040
```

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

Hint: save the benchmark output (or note the run time) to be able to refer to it later

Homework: GUI installation

mailto: **scalasca@fz-juelich.de**

cube scalasca

- Install & run ***local*** *(alternate solution)*
  - install Cube GUI locally on desktop
    - binary packages available for MacOS & Windows and externally provided by OpenHPC and various Linux distributions
    - source package available for Linux, requires Qt
      - configure/build/install manually or use your favourite framework (e.g. Spack or EasyBuild)
  - copy .cubex file (or entire scorep directory) to desktop from remote system
    OR locally mount remote filesystem
  - start cube locally

```
desk$ mkdir $HOME/mnt
desk$ sshfs [user@]remote.sys:[dir] $HOME/mnt
desk$ cd $HOME/mnt
desk$ cube ./scorep_sum/profile.cubex
```

https://www.**scalasca.org**/scalasca/software/cube-4.x/download.html