Teratec Hackathon

Edition #3

Conrad Hillairet – Staff HPC Engineer @ Arm
conrad.hillairet@arm.com

Monday 20th January 2025

Introduction

Introduction

Context

- During this Hackathon several teams of students will work on two HPC challenges provided by two big industrial partners : Viridien and EDF.
- The HPC platform made available is based on AWS Graviton3 and Graviton4 processors, fueled by Arm Neoverse technologies.
- Despite it being a contest, the goal is also pedagogical. We want students to learn something out of this event (team work, technical skills).
- Experts (from Arm, AWS, Viridien, EDF, Ucit) will be available for questions. They should be the main point of contact.
- The kick-off webinar recording is available here: https://teratec.eu/gb/activites/Hackathon.html
- The work of the Teams will be assessed (/100 points) to establish a ranking.

Support

Support

How can I get some help during the event ?

- Via Slack
  1. Join the AHUG Slack Workspace
     - You may receive an invitation prior to the event
     - Link available here https://a-hug.org/contact/
  2. Join the **teratec-hackathon-hpc** slack channel
     - Send a private message to Conrad Hillairet
  3. Ask your questions:
     - In the slack channel
     - Using private message to Conrad Hillairet
- Email
  - conrad.hillairet@arm.com

Hardware

Hardware

- Each team will have access to its own cluster made available by AWS and UCit.
- The head node is a reduced node, do not run benchmarks on it !
- Two compute nodes are available : a Graviton3 and a Graviton4. Comparing the performances of the codes on both platforms can be interesting.
  - Compute nodes can also be used for compilation (in addition of the head node) : split and share your work and use your resources wisely !
- Free your resources when you are not using it.

Software

Pre-installed software

Available in /tools

- Compilers:
  - ACFL 24.04 and 24.10
  - GCC 13.2.0 and 14.2.0
- Optimized BLAS, LAPACK (and FFT):
  - ArmPL 24.04 and 24.10
  - OpenBLAS
- MPI
  - OpenMPI 4.1.7 and 5.0.6
- Profiling and Debugging : Linaro Forge
- module use /tools/acfl/<version>/modulefiles
- export PATH and LD_LIBRARY_PATH for the other packages (/!\ Do not mess up your environment with multiple versions : create a source file for instance /!\)

Report

Report

/10 points

- We expect you to document your findings in a report or a presentation.
- We invite you to describe what works and what does not work.
- The report will be shared with the organizers before Monday 27th of January 2025, 9 am.

Viridien

Black-Scholes Tuning

/45 points

- The reference documentation are the documents provided by Viridien and the recording of the kick-off webinar.
- BSM.cxx is the source code of the problem.
- It is very likely that some optimizations that you will try will not work. We invite you to document it in your report nonetheless.
- Creativity and Ideas will be taken into account in the assessment.
- The new versions of the code (documented) will be provided at the end of the Hackathon.

EDF

Scientific Computing

/45 points

- The goal is to port a real industrial HPC application on Arm.
- In the context of this hackathon we consider that the porting includes (but is not limited to) the following steps: Compilation, Runs & Benchmarks, Timings, Scalability studies, Validation, Profiling, and Optimization.
- Concerning documentation, a good starting point is the recording of the kick-off webinar available on the Teratec hackathon webpage.
- Porting an application is almost a never-ending activity. We propose a methodology with different steps. In each category we provide some guidelines. We encourage you to follow them. But you are also free to do more tests if you find them relevant.

Guidelines provided by EDF

- Instructions to build the code are available here :
  - https://gitlab.com/codeaster-opensource-documentation/opensource-installation-development/-/blob/main/install/install-code-aster-native.md
- There is a link to an archive that includes the sources of the prerequired libraries here :
  - https://gitlab.com/codeaster-opensource-documentation/opensource-installation-development/-/blob/main/devel/changelog.md#version-17
  - Version 17 is the one currently under development so with the same prerequired libraries than the main branch.
- If we want to start with a stable version we can use the current tag : 17.1.9 (there will be 17.2.0 in December).
- Take the latest tag 17.x.y from the main branch.
- On these pages there is the command to launch the validation tests : run_ctest.
- For parallelism we do not use OpenMP or threads in code_aster. We distribute using MPI and we let the solver (MUMPS or PETSc) eventually use threads.
- There is a simple test to assess performance on a cube : https://gitlab.com/codeaster-opensource-documentation/opensource-installation-development/-/blob/main/benchmarks/Cube.md

Compilation

/20

- The goal of this part is to build the code.
- Several versions of compilers are made available. It is suggested to start with GCC 13.2.0 .
  - Ideally it would be good to have a version of the code compiled with GCC and a version of the code compiled with ACFL.
- Several implementations of BLAS and LAPACK are available : OpenBLAS, ArmPL, ATLAS (not preinstalled). It is suggested to start with OpenBLAS.
  - Ideally it would be good to have a version of the code compiled with OpenBLAS and a version of the code compiled with ArmPL.
  - Compiling with ArmPL could be challenging and could require to introduce support for it.

Runs, Benchmarks and Scalability

/10

- The goal of this part (now that it compiles) is to :
  - check that the code runs to completion
  - try to run some test cases from the community to do a benchmark comparison
  - have a look at the scalability behavior of the code.

- Start by checking that you can run some simple testcases like the ones mentioned by EDF.

- Explore the strong scalability behavior single node.
- Is the parallel efficiency good ?

- Compare the performances of the code compiled with difference compilers.
- Compare the performances of the code on Graviton3 against Graviton4.

Validation

/5

- The goal of this part (now that it runs) is to check that what is simulated is correct. In other words, that we are computing something meaningful.

- Run the validation testsuite, the more test cases you validate, the better.

Profiling and Optimization

/10

- The goal of this part is to understand the performance behavior of this application and try to improve it.

- Try to understand where in the code you are spending time.
- Can you make the code run faster ?
- What is the impact of different compiler flags ?
- Is vectorization beneficial for this code ?
- What has the compiler done in terms of vectorization ?
- How many loops did the compiler vectorize ?
- Extract some microkernels if they are of interest to speed-up the code.
- When optimizing make sure to double check that you do not break the code (validation).

arm

Merci
Danke
Gracias
Grazie
谢谢
ありがとう
Asante
Thank You
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה
ధన్యవాదములు
Köszönöm