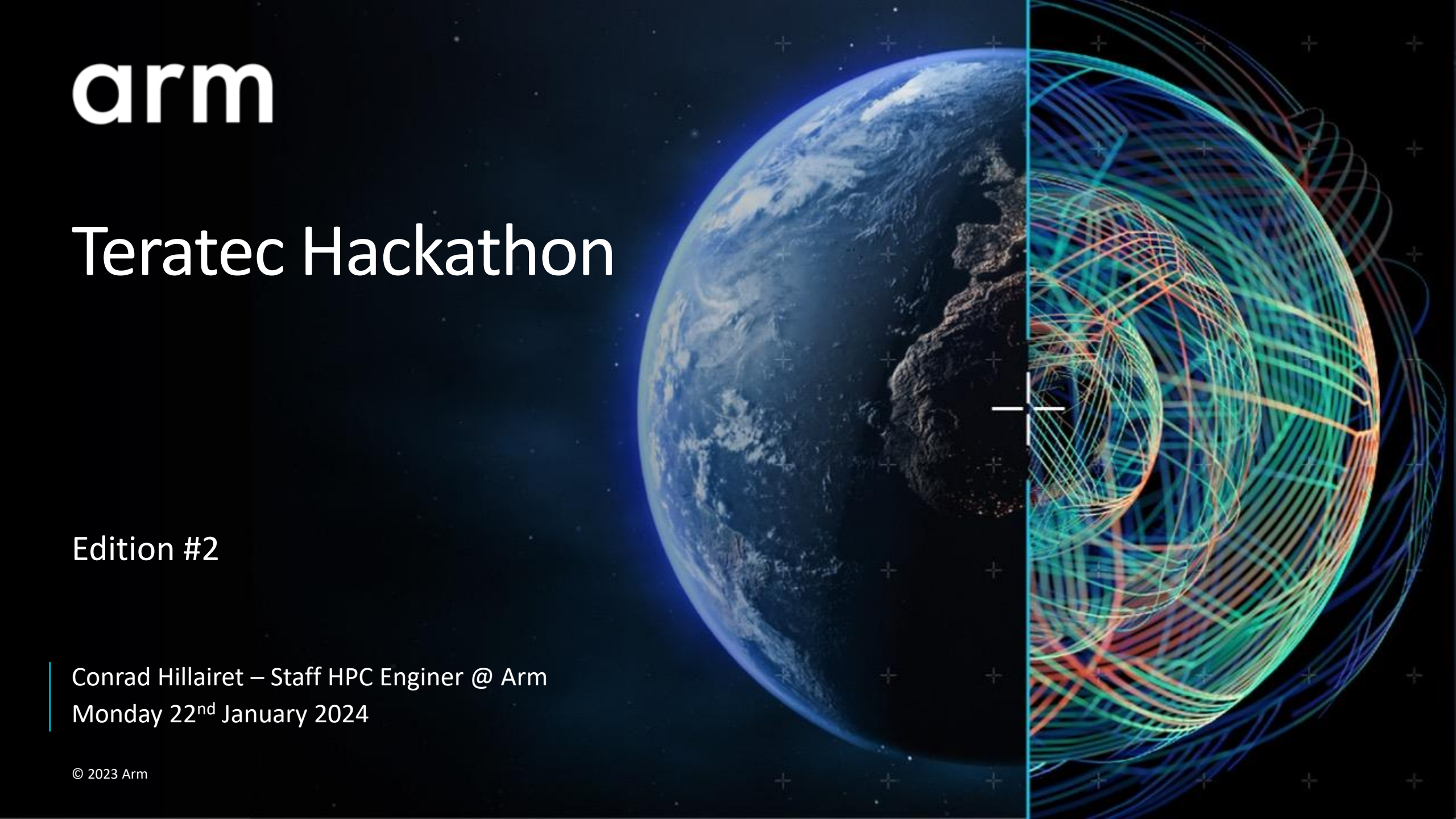


Teratec Hackathon

Edition #2

Conrad Hillairet – Staff HPC Enginer @ Arm

Monday 22nd January 2024

Introduction

Introduction

Context

- During this Hackathon several teams of students will work on two HPC challenges provided by two big industrial partners : CGG and EDF.
- The HPC platform made available is based on AWS Graviton 3 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, CGG, EDF, Linaro, Ucit) will be available for questions. They should be the main point of contact. Help shall not be provided by the teachers.
- 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 or Kévin Tuil
- Email
  - conrad.hillairet@arm.com
  - kevtuil@amazon.fr

Hardware

AWS Graviton3

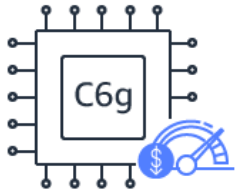
Hardware based on Arm technologies

Graviton2 Processor

Frequency
2.5 GHz

Peak Flops
1280 Gflops
2x128bits NEON

Peak Memory B/W
204 GB/s
DDR4



Many core
architecture
64 cores

Non-NUMA

7 nm

Arm Neoverse N1

Graviton3 Processor

Frequency
2.6 GHz

Peak Flops
2662 Gflops
2x256bits SVE
or 4x128bits NEON

Peak Memory B/W
307 GB/s
DDR5

Many core
architecture
64 cores

Non-NUMA

Energy
efficiency
5 nm

Arm Neoverse V1

Software

Pre-installed software

- Compilers: ACFL and GCC
- Optimized BLAS, LAPACK and FFT: ArmPL
- MPI : OpenMPI
- Profiling and Debugging : Linaro Forge

- module use /fsx/acfl/modulefiles
- module use /fsx/Libs/modulefiles

arm

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 29th of January 2024, 9 am.

arm

CGG

The Zeros of Rieman

/45 points

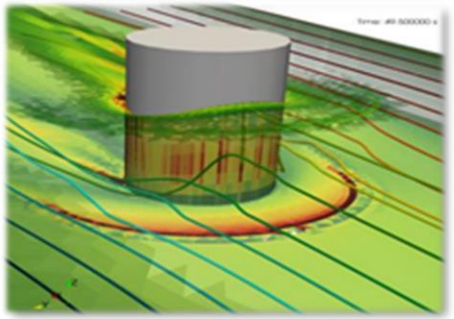
- The goal is to optimize on a single node a fairly small code written in C++ (you are free to rewrite it in Fortran if you prefer) that computes the number of zeros of Riemann's Zeta function on a given interval.
- The reference documentation are the slides provided by CGG and the kick-off webinar recording.
- 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.
- /25 points on the results obtained.
- /20 points on the creativity and ideas.

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.
- We will use this version of the code : https://gitlab.pam-retd.fr/otm/telemac-mascaret/-/archive/v8p5r0/telemac-mascaret-v8p5r0.tar.gz
- 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.



Compilation

/12

- The goal of this part it to build the code.

- TELEMAC can use several libraries (METIS, MED, MUMPS,…). To start with, just build it with METIS (only).
- Try to configure and compile TELEMAC with GNU compiler.
- Try to configure and compile TELEMAC with ACFL compiler.

- In case of success, you can try to build TELEMAC with more libraries.

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 of several test cases (single node and multi-node).
  - gaia/turbidity-3d can be a good starting point
- Is the parallel efficiency good ?

- Try to run some test cases from : https://gitlab.nicodet.fr/nicogodet/telemac-mascaret-benchmark

  And compare your results to the one presented.

Validation

/10

- 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.

- To start with, try to validate the following test cases :
  - examples/telemac2d/gouttedo/(t2d_gouttedo.cas)
    - The goal is only to check that everything work: this is the usual small test to run just after compilation.
  - examples/telemac2d/malpasset/(t2d_malpasset-nerd.cas)
    - To do first, because this is a test case with a standard numerical scheme called NERD
  - examples/telemac2d/malpasset/(t2d_malpasset-fine.cas)
    - For a geometry with a faire amount of number of cells, scalable on 8 or 10 cores
  - examples/telemac3d/bump_static/(t3d_bump_static.cas)
  - examples/telemac3d/malpasset/(t3d_malpasset-fine_p2.cas)

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

Profiling and Optimization

/13

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

- Try to understand where in the code you are spending time. It can be interresting to do it for several test cases to see if the bottlenecks are the same.
- 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).
- Try to explain the scalability behavior of the code.

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