on arm

Teratec Hackathon

AHUG Workshop
ISC 2023

Gilles Tourpe - HPC Business Developer - AWS
gtourpe@amazon.fr
Conrad Hillairet - Staff HPC Engineer - Arm
conrad.hillairet@arm.com

25th May 2023

Overview and genesis

TERATEC

- AWS and TERATEC
- Proposal / Objectives
  - Educate next HPC talents
  - Think HPC differently / Reality check
  - Promote/Support arm64 ecosystem
    - Accelerate adoption
    - Accelerate tools maturation

Big love to…

Teratec en partenariat avec arm aws CGG eDF UCit

Modus operati

Provide students with a Cloud HPC infrastructure to use AWS c7g instances using AWS Graviton3

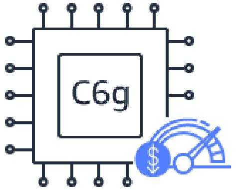
Provide them a 1st hand experience on using new ARM processors, new architectures, new tools to optimise a mini-app provided bv CGG and port a production grade code on a HPC Cloud infrastructure

100% of the participating teams (10) were to port CGG Stencil code (50% of the evaluation) and then either choose to port Code Telemac or Code Saturne from EDF R&D (50% of the evaluation)

AWS Graviton3

Hardware based on Arm technologies

Graviton2 Processor



Frequency
2.5 GHz

Peak Flops
1280 GFlops

Peak Memory B/W
204 GB/s

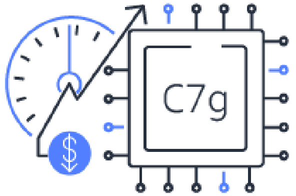
Many core architecture
64 cores

Non-NUMA

7 nm

Arm Neoverse N1

Graviton3 Processor



Frequency
2.6 GHz

Peak Flops
2662 GFlops

Peak Memory B/W
307 GB/s

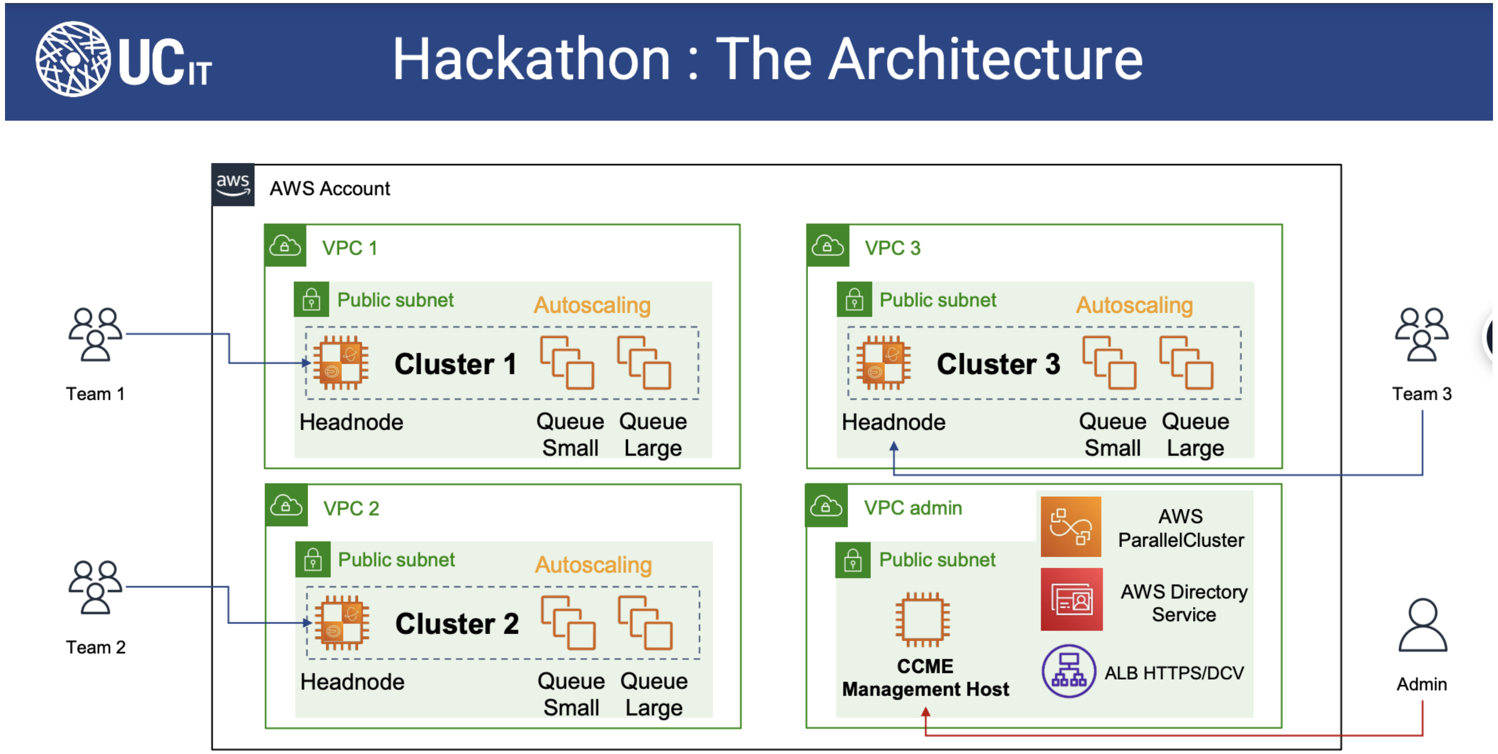
Many core architecture
64 cores

Non-NUMA

Energy efficiency
5 nm

Arm Neoverse V1

Architecture overview and UCit platform used to deliver it



- Each cluster is fully isolated from the others (network, accounts…)
- Job scheduler: SLURM
- Compute: 2 partitions
  - Small: c7g.4xlarge (4 vCPUs, 32 GiB) – limit 4 instances
  - Large: c7g.16xlarge (64 vCPUs, 128 GiB) – limit 4 instances
  - ➢ 272 vCPUs available
- Storage: Shared NFS – 500GiB
- Remote access
  - SSH connection to frontend node through login/password
  - Web portal EnginFrame + remote desktop on frontend node

Results

- 10 teams registered
- 7 actively participated

En s’appuyant sur ces codes industriels, ce **hackathon HPC organisé par Teratec et AWS** avec le soutien d'**ARM** et d'**UCit** a permis aux étudiants d’accroitre leur compréhension des enjeux industriels autour de la simulation haute performance et de se familiariser à l'utilisation du Cloud Computing pour le développement, l'analyse de performance et l'exécution de codes de calculs dits HPC.

L'UVSQ a remporté cette compétition en inscrivant 3 équipes sur le Podium. Bravo à l'équipe de Hugo BATTISTON, Guillaume BIGAND, Mathys JAM et Benjamin LOZES qui termine première.
Les équipes "The Assembler" et "Arm yourself" se partagent la seconde place.

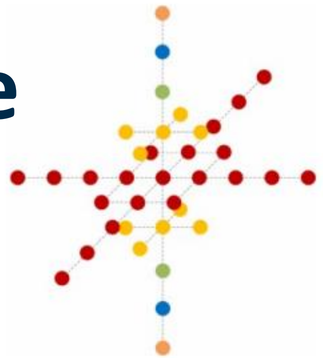
Félicitations à ces trois équipes qui pourront présenter leur travail et échanger avec la communauté HPC lors du Forum Teratec 2023 qui se tiendra au Parc Floral de Paris les 31 mai et 1er juin 2023. En attendant, l’équipe vainqueure s’est vue offrir 4 Macbok

Teratec Hackathon

Two codes – Representative of industial challenges

code_saturne

Stencil code



EDF

CGG

Teratec Hackathon

You have until Friday to « hack it » !
Good luck.

The Organising Committee.

CGG Stencils

What did they do ?

Math functions calls (pow)

OpenMP parallelization

Limit number of divisions

Vectorization (Neon, SVE)

Compiler Flags

Compiler optimization remarks

Tools : MAP, MAQAO

Remove unnecessary matrix copies

Reordering & unrolling loops

Cache blocking

Intrinsics

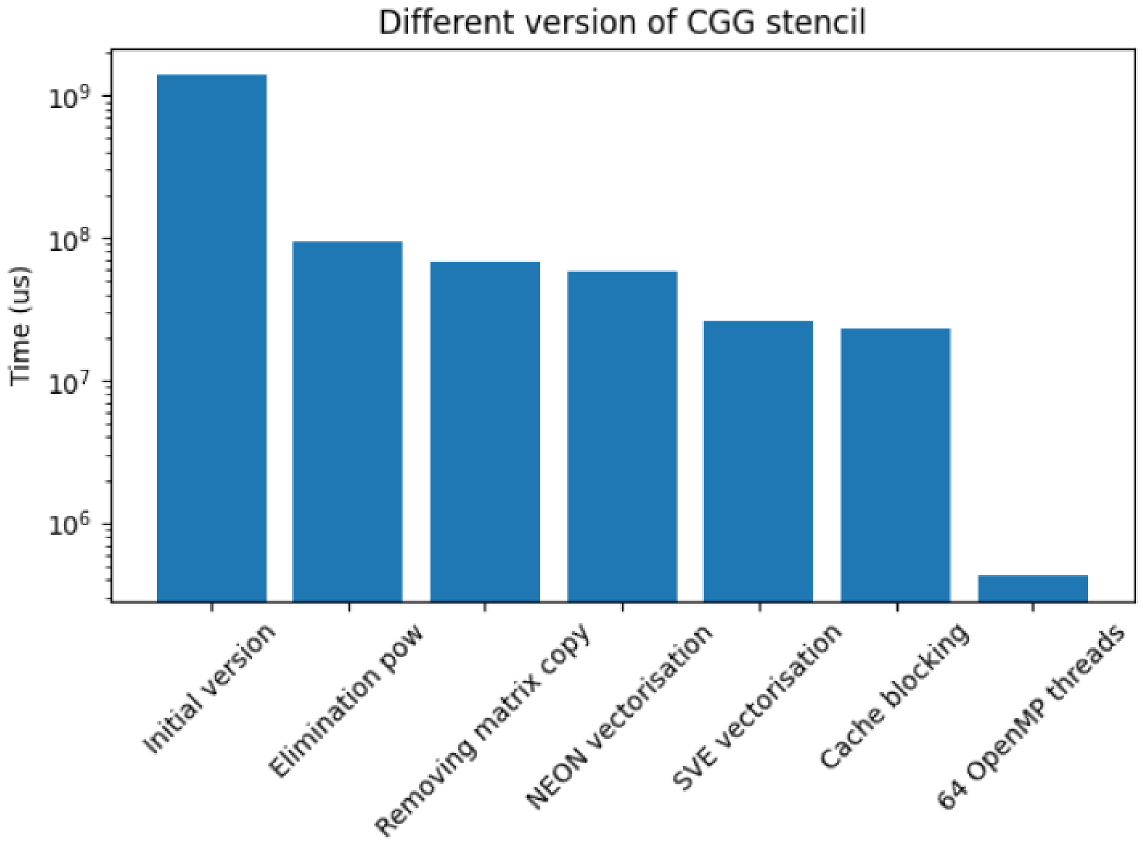


FIGURE 6 – Histogram of the different optimized versions

Best speed-up
7176x

Code_Saturne

Reference - Build instructions on x86 with Intel toolchain

Install

- Download Code_Saturne 7.2.0 (https://github.com/code-saturne/code_saturne/archive/refs/tags/v7.2.0.tar.gz)
- Use the archive
- Install the code locally
  - source ~/intel/oneapi/setvars.sh
  - CC="mpiicc"
  - CXX="mpiicpc"
  - FC="ifort"
  - DEST=$HOME/code_saturne_7.2.0
  - mkdir build
  - cd build
  - ../configure CC="$CC" CXX="$CXX" FC="$FC" \

    --with-blas=$MKLROOT --prefix=$DEST \

    --disable-gui --without-med \

    --without-hdf5 --without-cgns \

    --without-metis --disable-salome \

    --without-salome --without-eos \

    --disable-static --enable-long-gnum
  - make
  - make install

Run

- Download test cases https://github.com/code-saturne/saturne-open-cases
- Go inside folder BUNDLE
- Two test cases available : C016 et F128 , the main difference is the size of the mesh, C016 allows very fast computations (single core) and F128 more precise simulations (single node)
- First step : generate the meshes that will be used later
  - cd $REPBASE/BENCH_C016_PREPROCESS/DATA
  - $DEST/bin/code_saturne run
    - Outputted data are in $REPBASE/BENCH_C016_PREPROCESS/RESU/extrude_16
    - C016 : has 1 024 cells as an input and 17 408 as the output
  - cd $REPBASE/BENCH_F128_PREPROCESS/DATA
  - $DEST/bin/code_saturne run
    - Outputted data are in $REPBASE/BENCH_F128_PREPROCESS/RESU/extrude_128
    - F128 : 100 040 cells in input, 12 905 160 cells in output
- 2 ème étape : run testcase C016 on one node with 48 cores and 1 thread per MPI task :
  - cd $REPBASE/BENCH_C016_01/DATA
  - $DEST/bin/code_saturne submit --wckey dtsi:undefined --nodes=1 -n 48 --exclusive --ntasks-per-node=48 --cpus-per-task=1

on arm

Code_Saturne

Reference results for F128 on Cronos supercomputer

CRONOS (https://top500.org/system/179899/)

			reference CRONOS								
			F128_01			F128_02			F128_04		
nombre de nœuds	nombre de cœurs	nombre threads	User CPU time	Total CPU time	Elapsed time	User CPU time	Total CPU time	Elapsed time	User CPU time	Total CPU time	Elapsed time
1	48	1	241	12017	258	1019	50643	1069			
2	96	1	131	13184	142	534	53274	565			
4	192	1	89	17908	q97	295	58426	313	1103	220235	1168
8	384	1	50	20602	56	155	61143	165	599	234654	627
16	768	1	38	32345	45	141	111944	155	333	262441	360
32	1536	1	32	53595	38	59	99592	69	205	321137	229

on arm

Adapt configure command to your environment

x86 to aarch64

```
CC="mpiicc"
CXX="mpiicpc"
FC="ifort"
DEST=$HOME/code_saturne_7.2.0
mkdir build
cd build
../configure CC="$CC" CXX="$CXX" FC="$FC" \
             --with-blas=$MKLROOT --prefix=$DEST \
             --disable-gui --without-med \
             --without-hdf5 --without-cgns \
             --without-metis --disable-salome \
             --without-salome --without-eos \
             --disable-static --enable-long-gnum
make
make install
```

```
CC=mpicc CXX=mpicxx FC=armflang ./configure \
 --with-blas=$ARMPL_DIR --prefix=$PWD/build \
 --disable-gui --without-med                \
 --without-hdf5 --without-cgns              \
 --without-metis --disable-Salome           \
 --without-salome --without-eos             \
--disable-static --enable-long-gnum
```

Adapt configure command to your environment

x86 to aarch64

```
[...]
checking for python3 extension module
directory...
${exec_prefix}/lib64/python3.7/site-packages
checking for dlopen... dlopen
checking for MKL libraries... no
checking for threaded ATLAS BLAS... no
checking for ATLAS BLAS... no
checking for legacy C BLAS... no
configure: error: in `/home/ec2-
user/code_saturne-7.2.0':
configure: error: BLAS support is requested,
but test for BLAS failed!
See `config.log' for more details
```

Need to introduce support for ArmPL

Modify one file

Adapt ATLAS and MKL detection mechanisms

Adapt configure command to your environment

x86 to aarch64

```
[...]
${exec_prefix}/lib64/python3.7/site-packages
checking for dlopen... dlopen
checking for MKL libraries... no
checking for threaded ATLAS BLAS... no
checking for ATLAS BLAS... no
checking for legacy C BLAS... no
checking for ArmPL libraries... yes
checking for MPI (MPI compiler wrapper
test)... yes
checking for MPI I/O... yes
checking for MPI in place... yes
checking for MPI nonblocking barrier... yes
CCMIO headers not found
[...]
```

Adapt configure command to your environment

x86 to aarch64

```
[...]
checking for a sed that does not truncate
output... /usr/bin/sed
checking for ar... ar
checking the archiver (ar) interface... ar
configure: sourcing config/cs_auto_flags.sh
(test for known compilers)
compiler 'mpicc' is NVIDIA compiler
compiler 'mpicxx' is NVIDIA compiler
compiler 'armflang' is NVIDIA compiler
checking how to print strings... printf
[...]
```

Bug in compiler detection mechanism

Arm Compiler for Linux detection overwritten by NVIDIA detection

Two files to modify (~10-20 lines)

Adapt configure command to your environment

x86 to aarch64

```
[...]
checking for a sed that does not trun
checking for ar... ar
checking the archiver (ar) interface.
configure: sourcing config/cs_auto_fl
compilers)
compiler 'mpicc' is NVIDIA compiler
compiler 'mpicxx' is NVIDIA compiler
compiler 'armflang' is NVIDIA compile
checking how to print strings... prin
[...]
```

```
[...]
checking for ar... ar
checking the archiver (ar) interface... ar
configure: sourcing config/cs_auto_flags.sh
(test for known compilers)
compiler 'mpicc' is Arm C compiler
compiler 'mpicxx' is Arm C++
compiler 'armflang' is Arm Fortran compiler
checking how to print strings... printf
[...]
```

etection

Linux detection

VIDIA detection

fy (~10-20 lines)

Compile

x86 to aarch64

```
make -j 4 > resComp >&1
make install


grep -i err ./resComp
```

No problem, compiles OoB

on arm

Flat MPI version

x86 to aarch64

```
CC=mpicc CXX=mpicxx FC=gfortran ./configure --
prefix=$PWD/build-noomp --disable-gui --without-hdf5 --
without-cgns --without-med --without-metis --disable-static
--disable-salome --without-salome --without-eos  --enable-
long-gnum --with-blas=$ARMPL_DIR --disable-openmp > resConf
2>&1

grep -i openmp ./resConf
[…]
AArch64_RHEL-7_aarch64-linux/bin/../lib/gcc/aarch64-linux-
gnu/11.2.0/../../.. -lgfortran -lm
checking for OpenMP (C)... yes
checking for OpenMP (Fortran)... yes
checking for Fortran libraries of gfortran... (cached) OpenMP
support: no
 OpenMP support: yes
 OpenMP accelerator support: no
 OpenMP Fortran support: yes
 Auto load environment modules:  binutils/11.2.0 gnu/11.2.0
armpl/22.1.0 openmpi/gcc/4.1.4
```

Bug in OpenMP configuration
not Arm specific
also happens on x86

One file to modify (~1 line)

Flat MPI version

x86 to aarch64

```
CC=mpicc CXX=mpicxx FC=gfortran ./configu
noomp --disable-gui --without-hdf5 --with
without-metis --disable-static --disable
-without-eos --enable-long-gnum --with-b
openmp > resConf 2>&1

grep -i openmp ./resConf
[…]
AArch64_RHEL-7_aarch64-linux/bin/../lib/g
gnu/11.2.0/../../.. -lgfortran -lm
checking for OpenMP (C)... yes
checking for OpenMP (Fortran)... yes
checking for Fortran libraries of gfortra
support: no
 OpenMP support: yes
 OpenMP accelerator support: no
 OpenMP Fortran support: yes
 Auto load environment modules: binutils
armpl/22.1.0 openmpi/gcc/4.1.4
```

```
CC=mpicc CXX=mpicxx FC=gfortran ./configure --
prefix=$PWD/build-noomp --disable-gui --without-hdf5 --
without-cgns --without-med --without-metis --disable-static
--disable-salome --without-salome --without-eos  --enable-
long-gnum --with-blas=$ARMPL_DIR --disable-openmp > resConf
2>&1

grep -i openmp ./resConf
[…]
OpenMP support: no
OpenMP support: no
 Auto load environment modules:  binutils/11.2.0 gnu/11.2.0
armpl/22.1.0 openmpi/gcc/4.1.4


make > resComp 2>&1
grep fopenmp ./resComp | wc -l
0
```

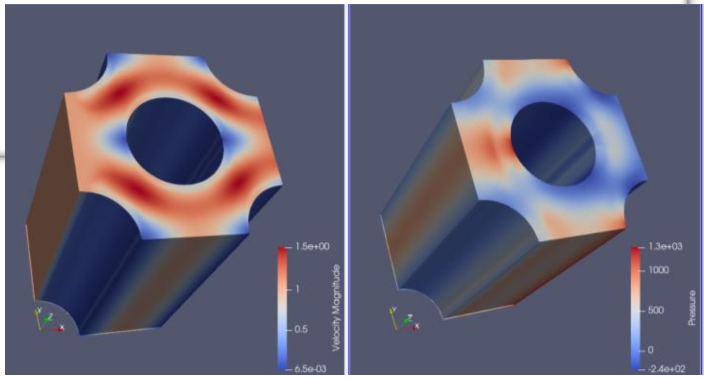

Run

x86 to aarch64

No problem at run time

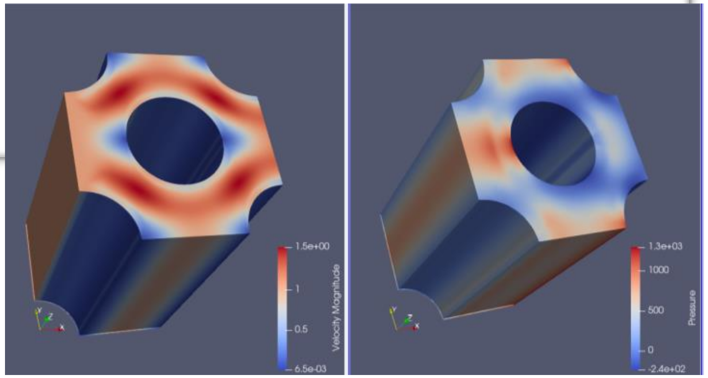
```
First step : generate the meshes that will be used later
cd $REPBASE/BENCH_F128_PREPROCESS/DATA
$DEST/bin/code_saturne run

Second step: run testcase C016 on one node with 48 cores and 1
thread per MPI task :
cd $REPBASE/BENCH_C016_01/DATA
$DEST/bin/code_saturne submit --wckey dtsi:undefined --nodes=1 -n
48 --exclusive --ntasks-per-node=48 --cpus-per-task=1
```



```
First step : generate the meshes that will be used later
cd $REPBASE/BENCH_F128_PREPROCESS/DATA
$DEST/bin/code_saturne run

Second step: run testcase C016 on one node with 64 cores and 1
thread per MPI task :
cd $REPBASE/BENCH_C016_01/DATA
$DEST/bin/code_saturne submit -n 64 --cpus-per-task=1
```

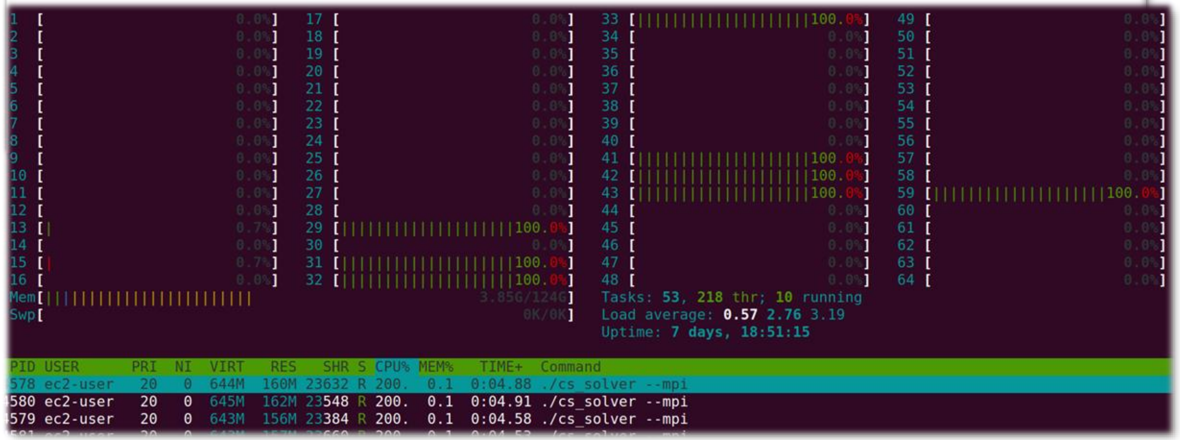


Another run with MPI & OpenMP

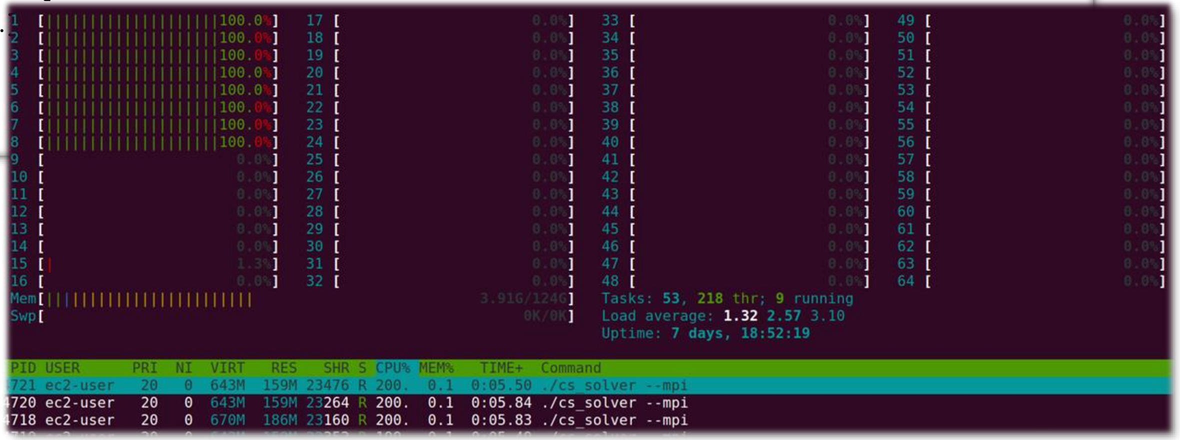
x86 to aarch64

Not Arm specific

```
code_saturne submit -n 4 --nt 2 --cpus-per-task=2
```

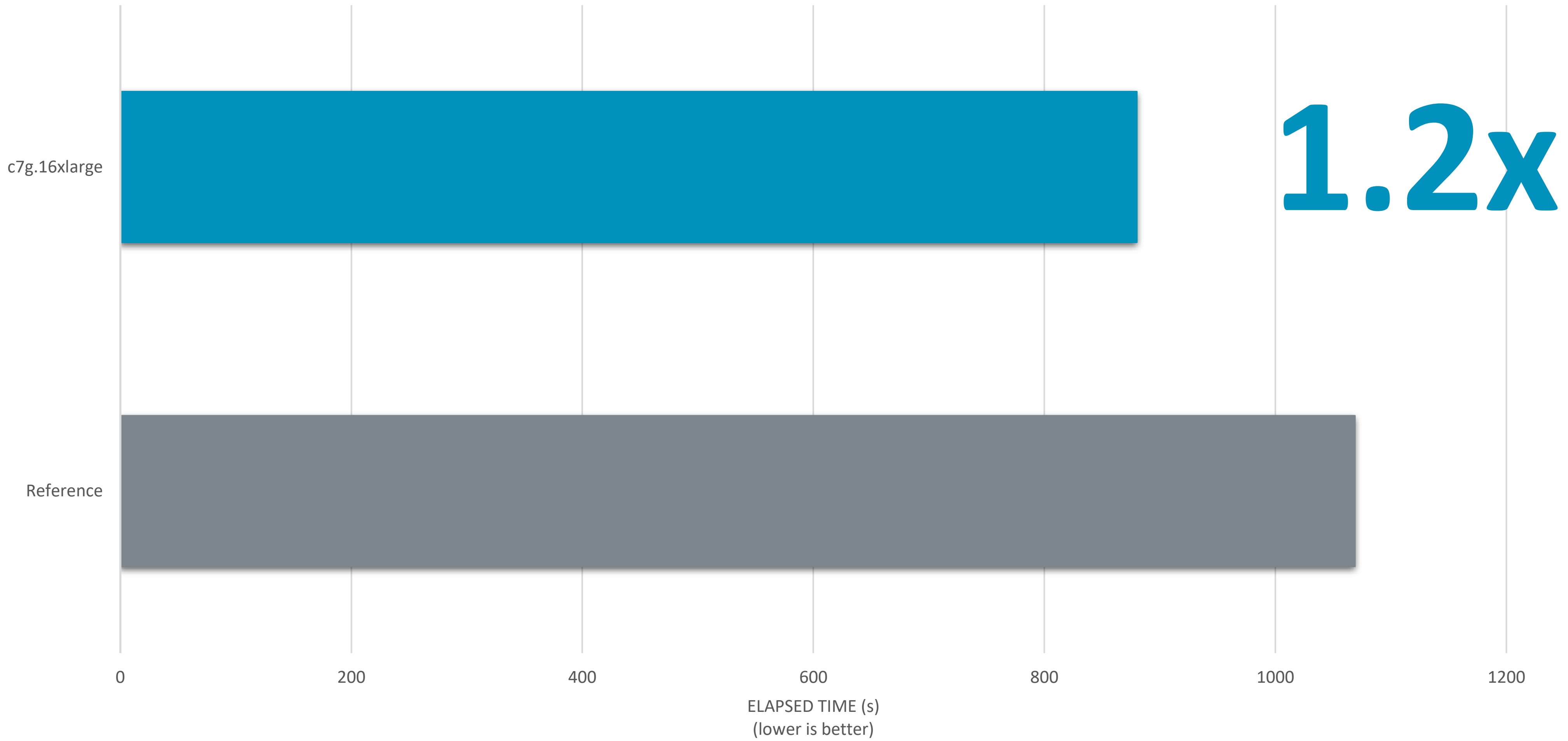



```
export CS_MPIEXEC_OPTIONS="--map-by ppr:4:node:PE=2 --report-bindings"
code_saturne submit -n 4 --nt 2
[ip-172-31-77-39:04714] MCW rank 0 bound to socket 0[core 0[hwt 0]], socket 0[core 1[hwt 0]]: [B/B/./././././././././././././././././././././././././././././././././././././././././././././././././././././././././././././.]
[ip-172-31-77-39:04714] MCW rank 1 bound to socket 0[core 2[hwt 0]], socket 0[core 3[hwt 0]]: [././B/B/././././././././././././././././././././././././././././././././././././././././././././././././././././././././././.]
[…
```



Code_Saturne

AWS Graviton 3 – Arm Neoverse V1



Conclusion

"This hackathon was the opportunity for us to test an architecture we never explored before. But before all, it challenged our skills, and forced us to reconsider our weaknesses. While most members of the team are experienced programmers, we had to use a lot of different tools to speedup our analysis process. Tools we overlooked before, or did not take the time to learn. This is especially true for Code Saturne. **We really appreciate the effort put in by the organizing committee, for the quality of the infrastructure provided, and especially for the difficulty and variety of the problems we had to solve.**

We strongly believe this challenge was of value, and hope that our efforts presented here will be appreciated."

And most importantly

We found the source of all our problems …

"Either the intrinsic calls we did were not the best ones or **the compiler did not understand what we wanted to do."**

Next

Teratec

https://teratec.eu/activites/Hackathon.html

More to come for aarch64 adoption

- Euromaster4HPC Summer School (07/23)
- EPITA HPC new major (CY24)
- University of Luxemburg Summer School (06/23)
- …

on arm

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