CNCA Centro Nacional de Computação Avançada

DEUCALION

EasyBuild @ Deucalion

B. Malaca (CNCA)
April 2026

PRR Plano de Recuperação e Resiliência
REPÚBLICA PORTUGUESA
Financiado pela União Europeia NextGenerationEU
EuroHPC Joint Undertaking
fct Fundação para a Ciência e a Tecnologia
INESCTEC
Universidade do Minho

HPC in Portugal

Pre 2019

Different Tera-scale machines scattered along the territory (biggest one in Coimbra)

There was no national calls, so either PRACE or directly through “local” computing centers

BOB (2019)

1 PFlop system with 800 nodes, donated from TACC (Stampede)

Portugal started having national calls for computing time in academia and others

Deucalion (2024)

10 PFlop system with 2165 nodes, with three architectures

Now researchers can access both Deucalion and Marenostrum 5 via national calls

Deucalion in numbers

ARM partition

1632 nodes

Fujitsu A64FX (48 cores) processor

32 GB HBM2 RAM (50% faster than x86)

+1500 modules

X86 partitions

500 nodes + 33 GPU nodes (4x A100 40/80 GB)

2 x AMD EPYC 7742 per node (128 cores)

256 GB RAM

+2500 modules

Disk space of 10PB in a Lustre parallel filesystem

New website: deucalion.acnca.pt

Using EasyBuild with three architectures

Good things

Module naming is consistent: people can just change the partition in the sbatch options and it'll still work (our logins are x86)

The community is quite fast at shipping new stuff

Difficulties

Usually installations for Zen2/GPU work but ARM was more difficult (getting better)

Performance ARM v. x86 varies on a per-core basis.

Suggestion

Expand lfoss toolchain to incorporate proprietary clang-compatible compilers (armclang, aocc, etc.)

Main difficulties

- ARM and x86 stacks
- Pytorch+ACL
- LLVM toolchains
- Fujitsu BLAS

ARM and x86 stacks

Benchmarks

X86 and GPU partition benchmark similarly to other comparable machines

ARM was slower, with wildly different code-depending performance

Compilers

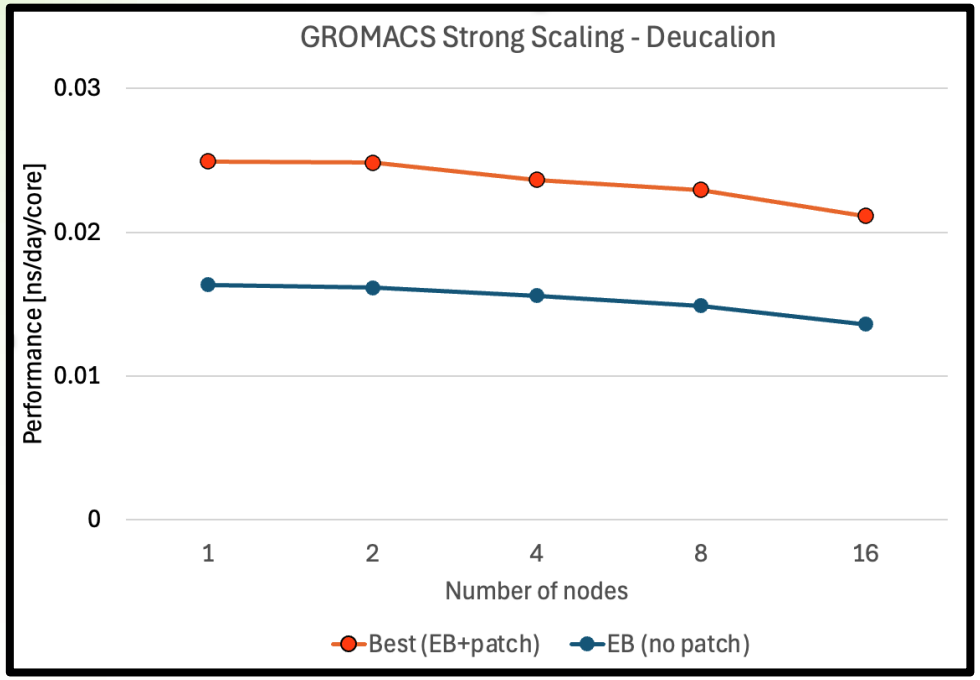
ARM also has gcc, nvcc, armclang, and Fujitsu's

The closer the code to typical BLAS work, the better for A64FX

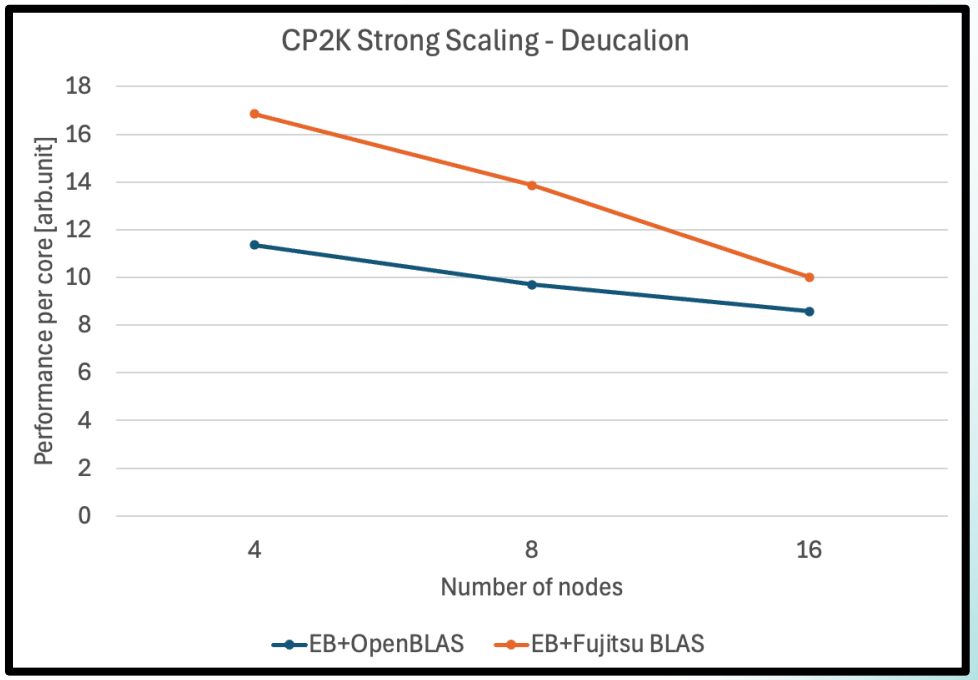
Considering the energy the gap closes further

ARM and x86 stacks

EasyBuild packages for ARM can be optimized further



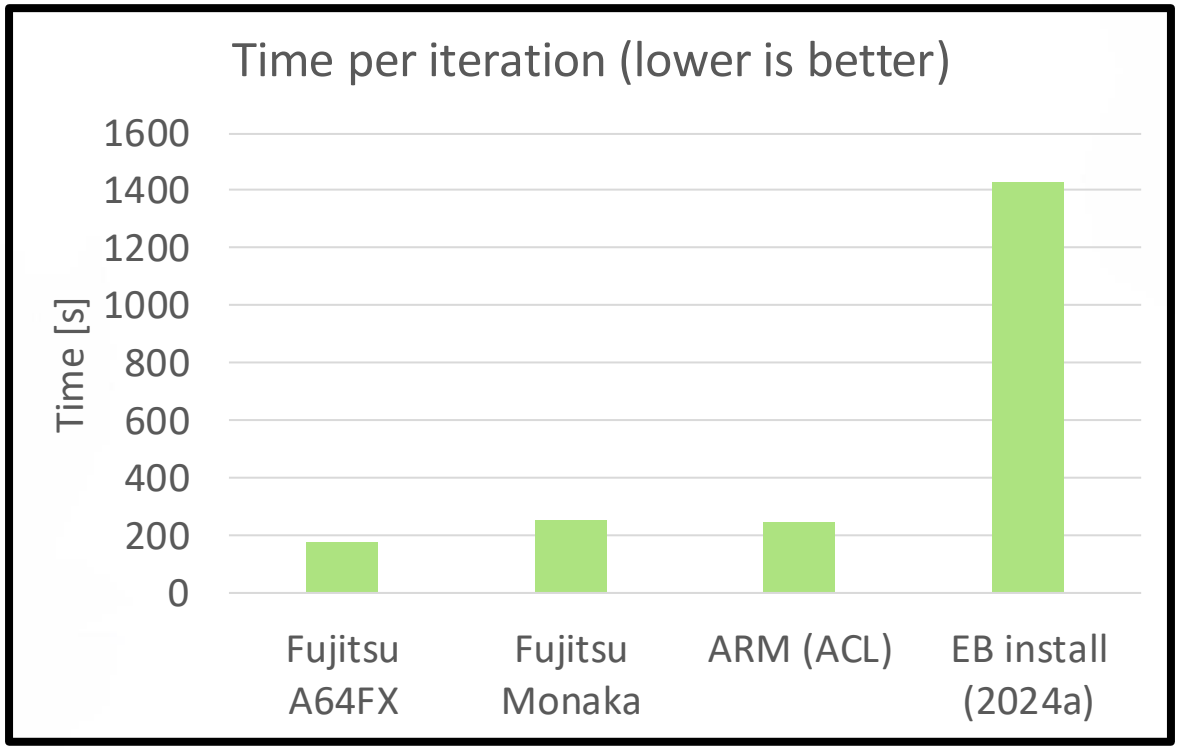
At the same number of cores A64FX spends 1.15x the energy of Rome to solve the problem



At the same number of cores A64FX spends the same energy as Rome to solve the problem

Pytorch (ACL)

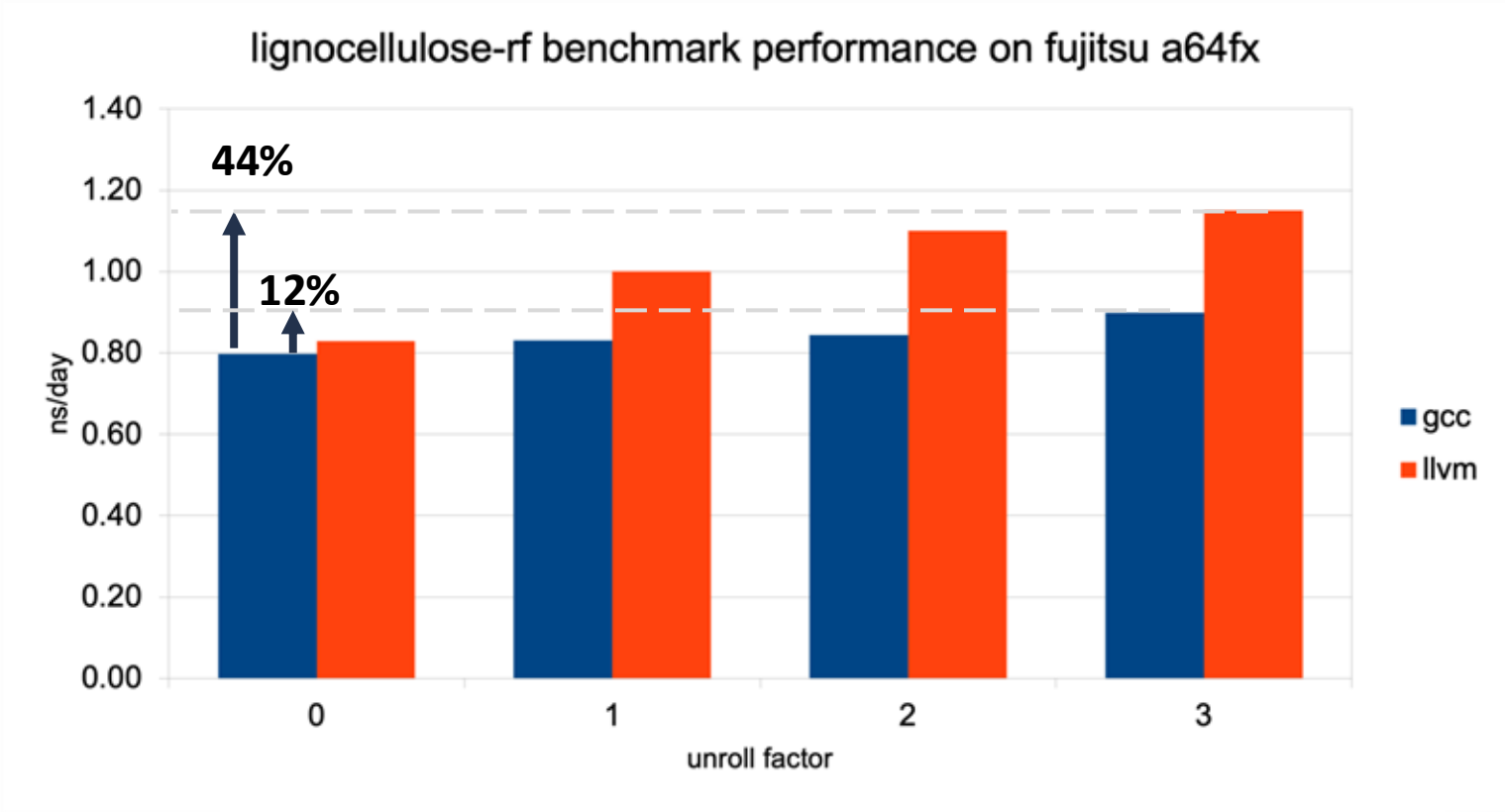
Benchmark: CIFAR-100 (https://github.com/weiaicunzai/pytorch-cifar100/tree/master)



Fujitsu is not shipping new versions of Pytorch so better to focus on integrating the ACL

GROMACS and LLVM

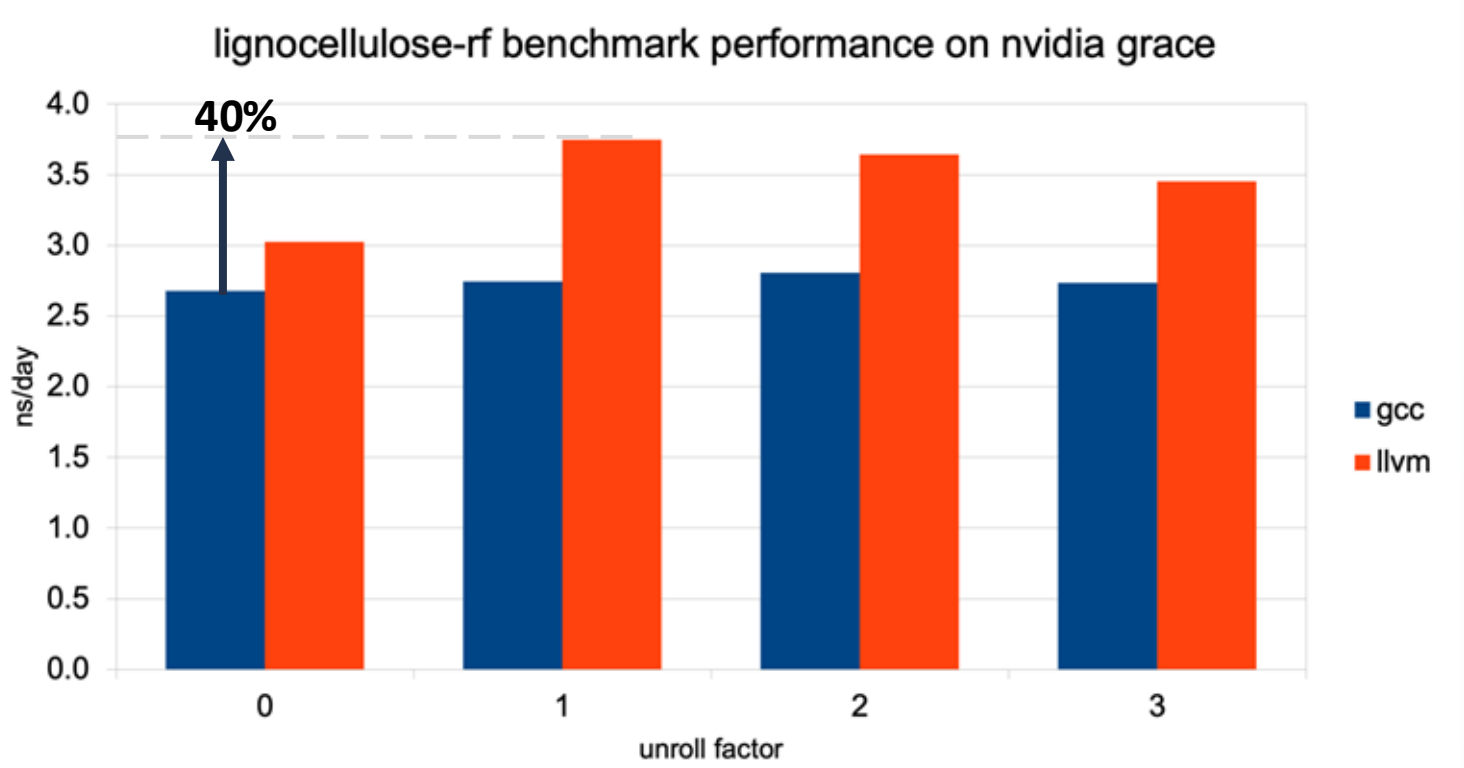
GROMACS got a patch from a researcher to optimize for A64FX (improve software pipelining)



Warning: lfoss-2025b uses LLVM-20, which doesn't work well with SVE (either patch or use lfoss-2026.1 directly)

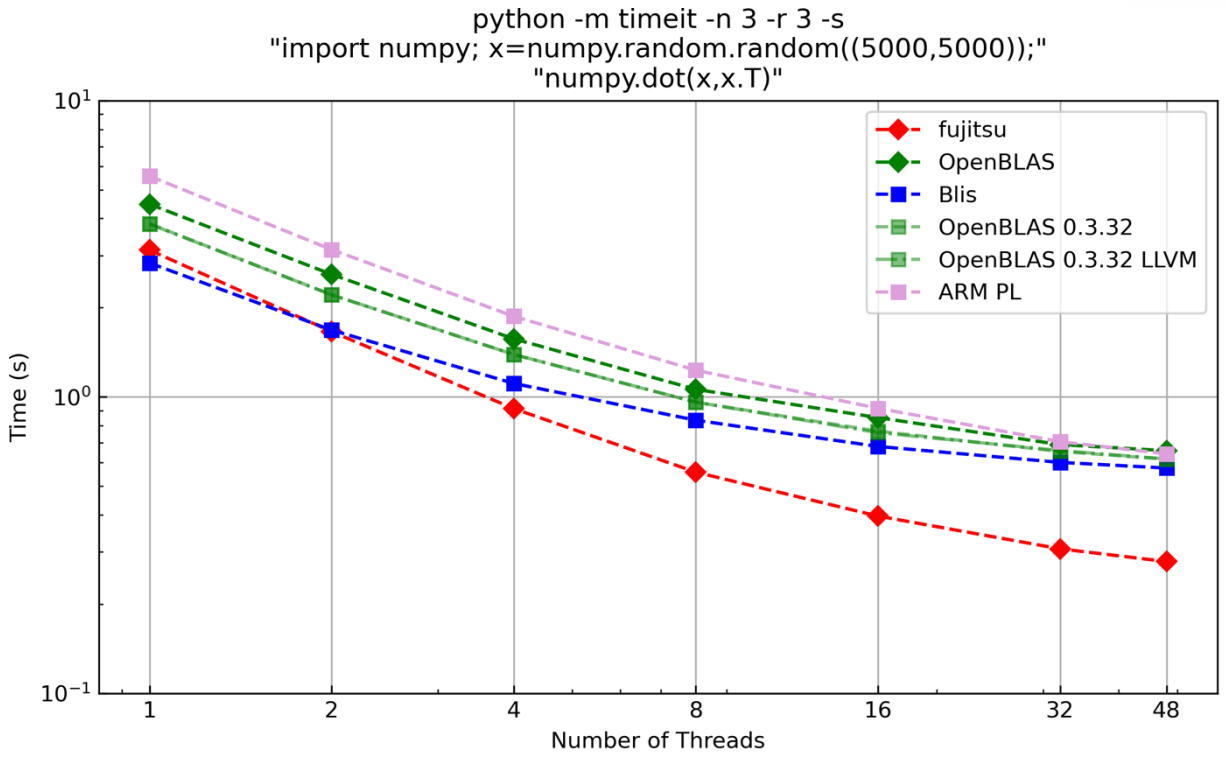
GROMACS and LLVM

GROMACS got a patch from a researcher to optimize in A64FX (improve software pipelining)



Warning: lfoss-2025b uses LLVM-20, which doesn't work well with SVE (either patch or use lfoss-2026.1 directly)

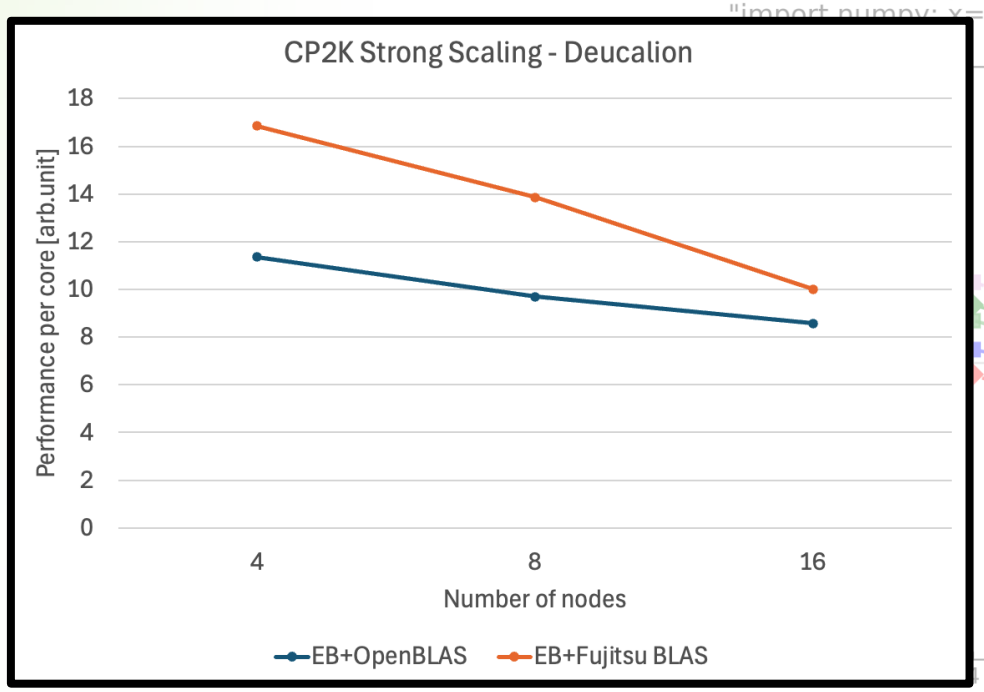
Fujitsu BLAS



For non-threaded BLAS, it's possible to use FlexiBLAS (combined with LD_PRELOAD) to use Fujitsu's BLAS (Fujitsu's compiler is needed for threaded work)

OpenBLAS 0.3.32 incorporated specific improvements (around 5%)

Fujitsu BLAS




```
ml QuantumESPRESSO/7.5-foss-2025b
export FLEXIBLAS=/opt/FJSVstclanga/
cp-1.0.21.02a/lib64/libfjlapacksve.
so
export LD_PRELOAD=/opt/
FJSVstclanga/cp-1.0.21.02a/lib64/
libfj90f.so:/opt/FJSVstclanga/cp-1.
0.21.02a/lib64/libfj90rt.so:/opt/
FJSVstclanga/cp-1.0.21.02a/lib64/
libfj90i.so:/projects/easybuild/
extra/aarch64/software/FFTW.FUJ/1.
1.1-sve-Fujitsu/lib64/libfftw3.so
```

For non-threaded BLAS, it's possible to use FlexiBLAS (combined with LD_PRELOAD) to use Fujitsu's BLAS (Fujitsu's compiler is needed for threaded work)

OpenBLAS 0.3.32 incorporated specific improvements (around 5%)

EESSI

There was significant effort to include A64FX as a testing and building platform @Deucalion

Bunch of Hooks added to EESSI

BLIS

```
Pre-configure hook
for BLIS:
- fix "Illegal
instruction" problem
on A64FX by adding -
DCACHE_SECTOR_SIZE_RE
ADONLY to $CFLAGS for
BLIS 0.9.0
```

Rust

```
Replace build
dependency on Rust
1.88.0 by 1.91.1, as
1.88.0 causes
segmentation faults
on A64FX
```

Numpy

```
Pre-extension hook for
numpy:
- change -march=native
to -march=armv8.2-a
for numpy 1.24.2 when
building for
aarch64/a64fx CPU
target
```

Thanks to everyone that participated in that effort

Conclusions

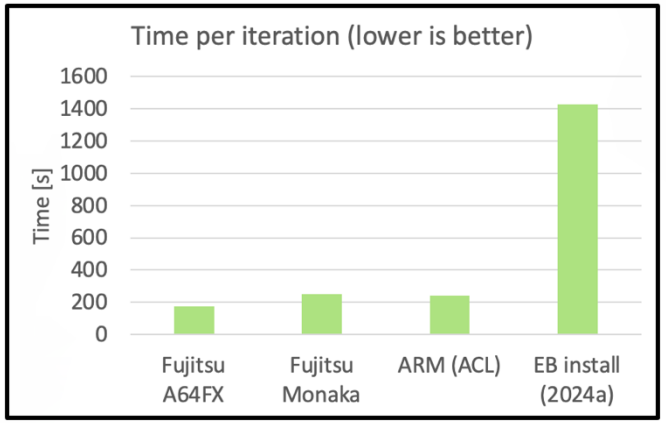
Deucalion

Heterogenous cluster (need to compile every module twice)

Different performances

EESSI available in all partitions

ARM partition



Probably good to be able to test different LLVM forks in a simpler way

Proprietary BLAS/FFT still leads