UK Research and Innovation

Porting and Optimising TELEMAC–MASCARET for the OpenPOWER Ecosystem

EMiT 2019, 9th–11th April, Huddersfield, UK

Judicaël Grasset(1), Yoann Audouin(2), Stephen Longshaw(1), Charles Moulinec(1), David R. Emerson(1)

2019–04–10

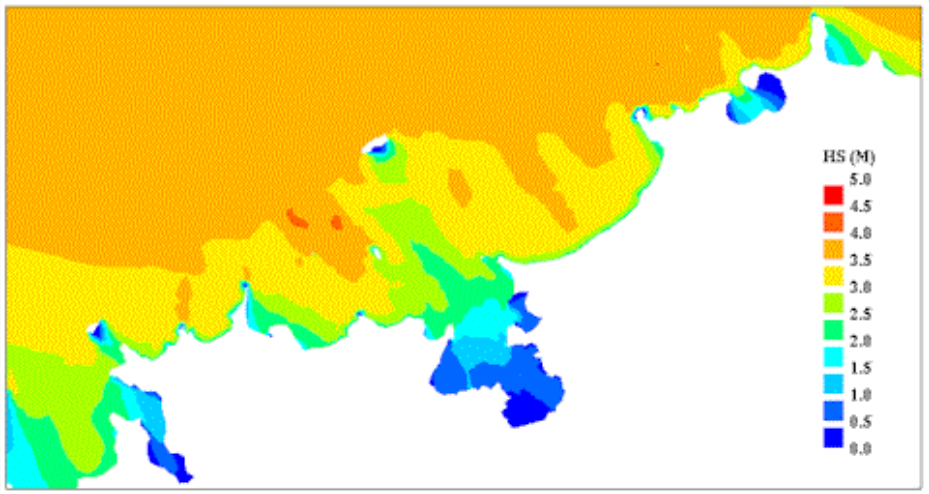
(1) STFC, Daresbury Laboratory, Warrington, United Kingdom
(2) EDF R&D, Chatou, France

What is TELEMAC–MASCARET?

TELEMAC–MASCARET is an open-source suite of hydrodynamic solvers for free-surface flow modelling.

It was originally developed by Électricité de France (EDF) in the 1990s and is now developed through the TELEMAC–MASCARET consortium:

- Artelia
- BAW
- CEREMA
- CERFACS
- UKRI–STFC, Daresbury Laboratory
- EDF
- HR Wallingford



Courtesy of the official TELEMAC–MASCARET website.

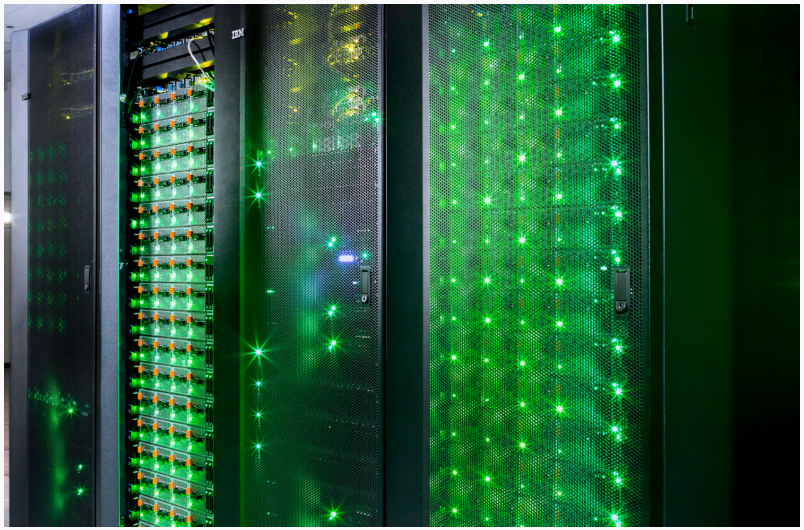
What is TELEMAC–MASCARET?

- TELEMAC–MASCARET is only parallelized with MPI
- Which is usefull when HPC clusters are made of single core processor
- But HPC clusters have more and more core per processor
- The compute nodes also have more and more GPUs
- Then in order to let TELEMAC–MASCARET use the full computing power of tommorrow's cluster, it is needed to search for new way of adding parallelism

Computing used

OpenPOWER archtecture in a nutshell:

- IBM POWER processors
- NVIDIA GPUs
- NVIDIA NVLink



The machine used for this work, Paragon

In our case, each node of the machine used consists of:

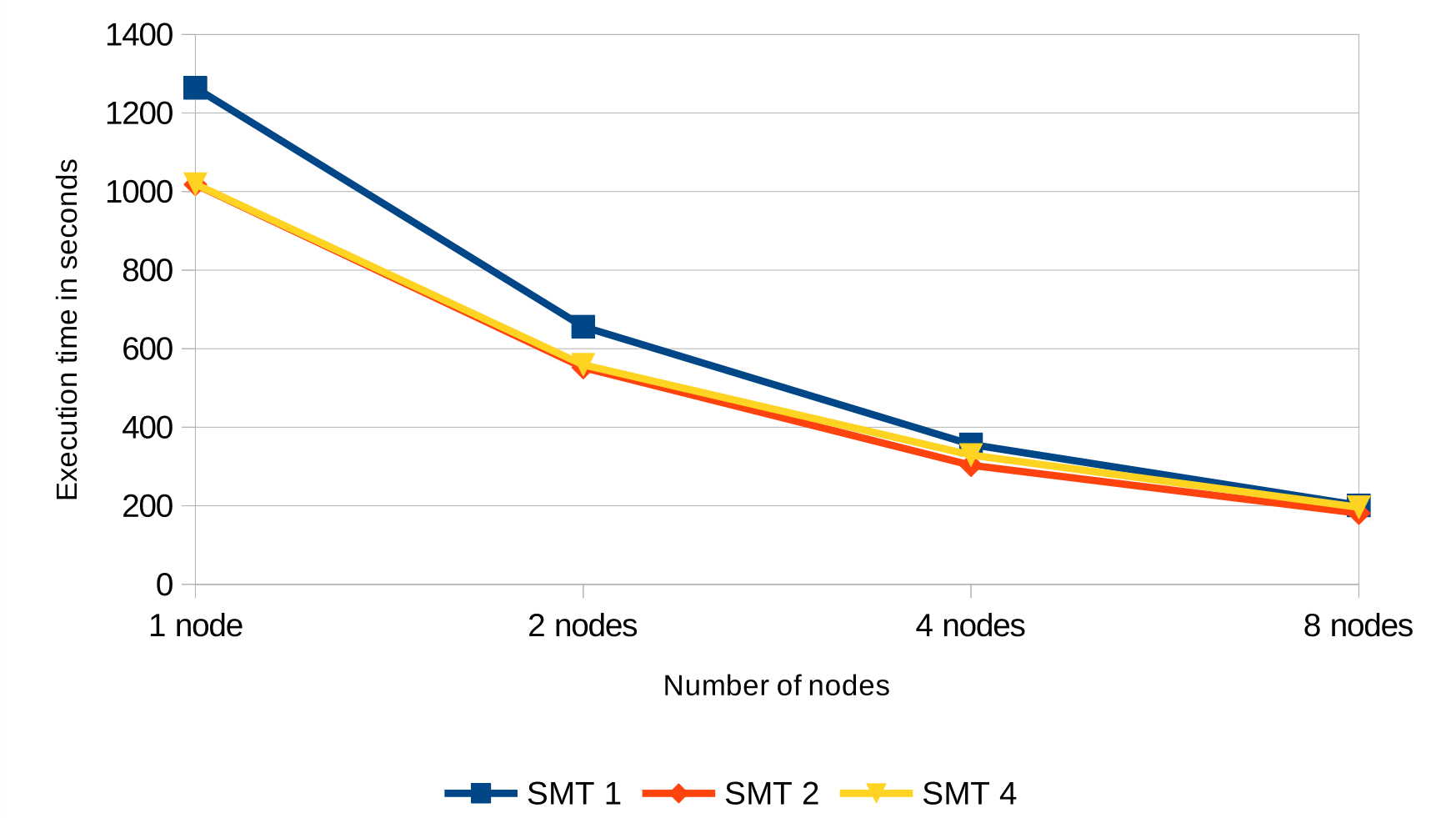
- Two IBM POWER 8 processors, with 8 cores each
- Each core has simultaneous multithreading (SMT) capability
- In this case the cores are able to run either 1 threads (SMT1), 2 threads (SMT2), 4 threads (SMT4) or 8 threads (SMT8) at the same time
- Four NVIDIA P100 GPUs
- NVIDIA NVLink for GPU–GPU and GPU–CPU interconnections

The test case

Test case used: tomawac/fetch_limited/tom_test6.cas

- This is a limited test with a small mesh: 18k elements, 9.6k points.
- It spends all of its time in a single fortran function: qnlin3.f
- This function was reported to be a bottleneck by some users during the annual TELEMAC User Conference (2018).

Original execution time with MPI (IBM compiler)

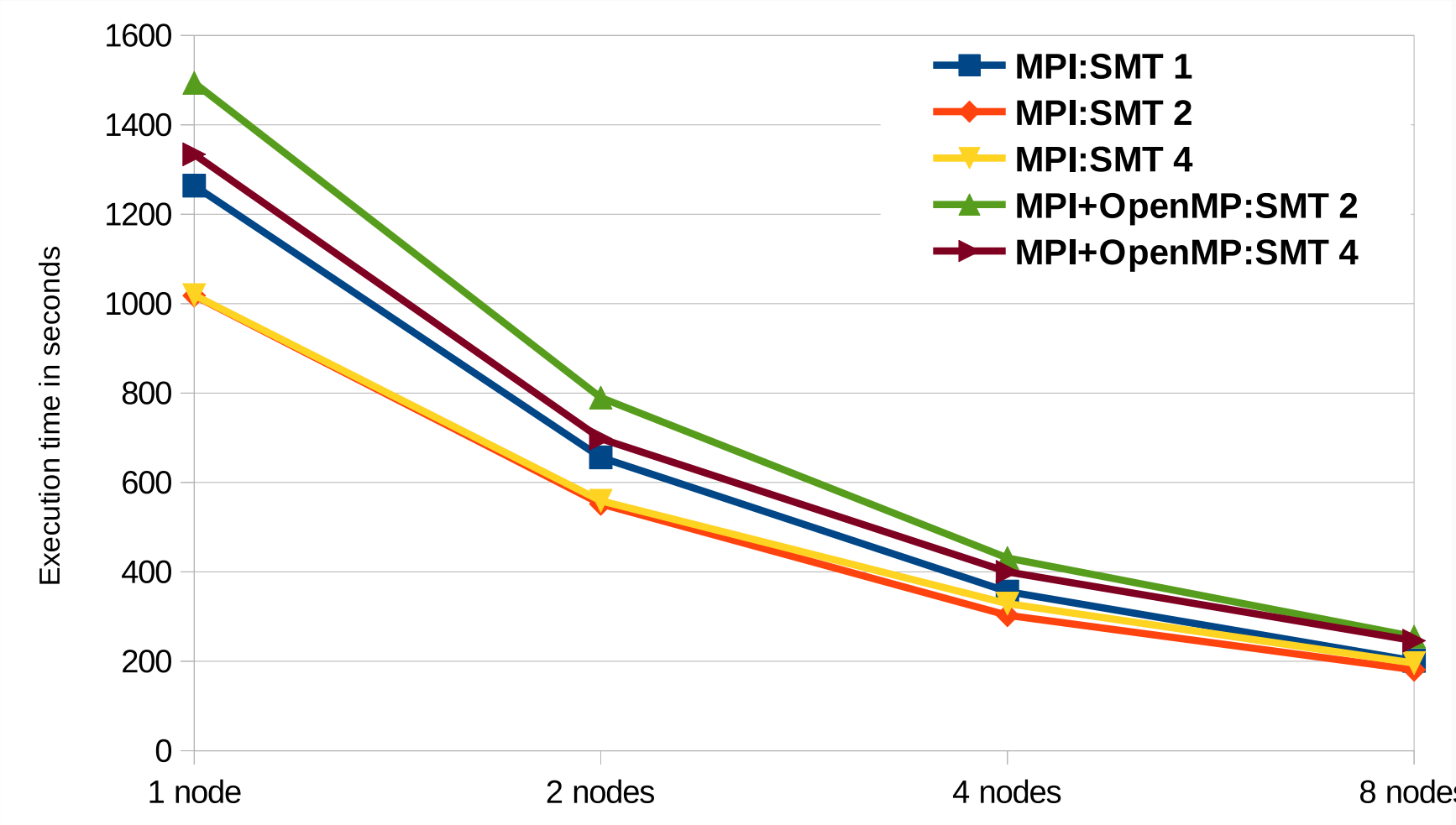


MPI+OpenMP (IBM compiler) on CPU

We can already use the simultaneous multithreading for MPI parallelization.
But would it be better to use it for OpenMP parallelization?

- create and initialise array for reduction
- !$omp parallel do reduction(+:tmp_array)
- do loop
- do loop
- do loop
- do loop
- tmp_array(x,y,z) = tmp_array(x,y,z) + k
- ...
- !$omp end parallel do
- array = array + tmp_array

MPI+OpenMP (IBM compiler)



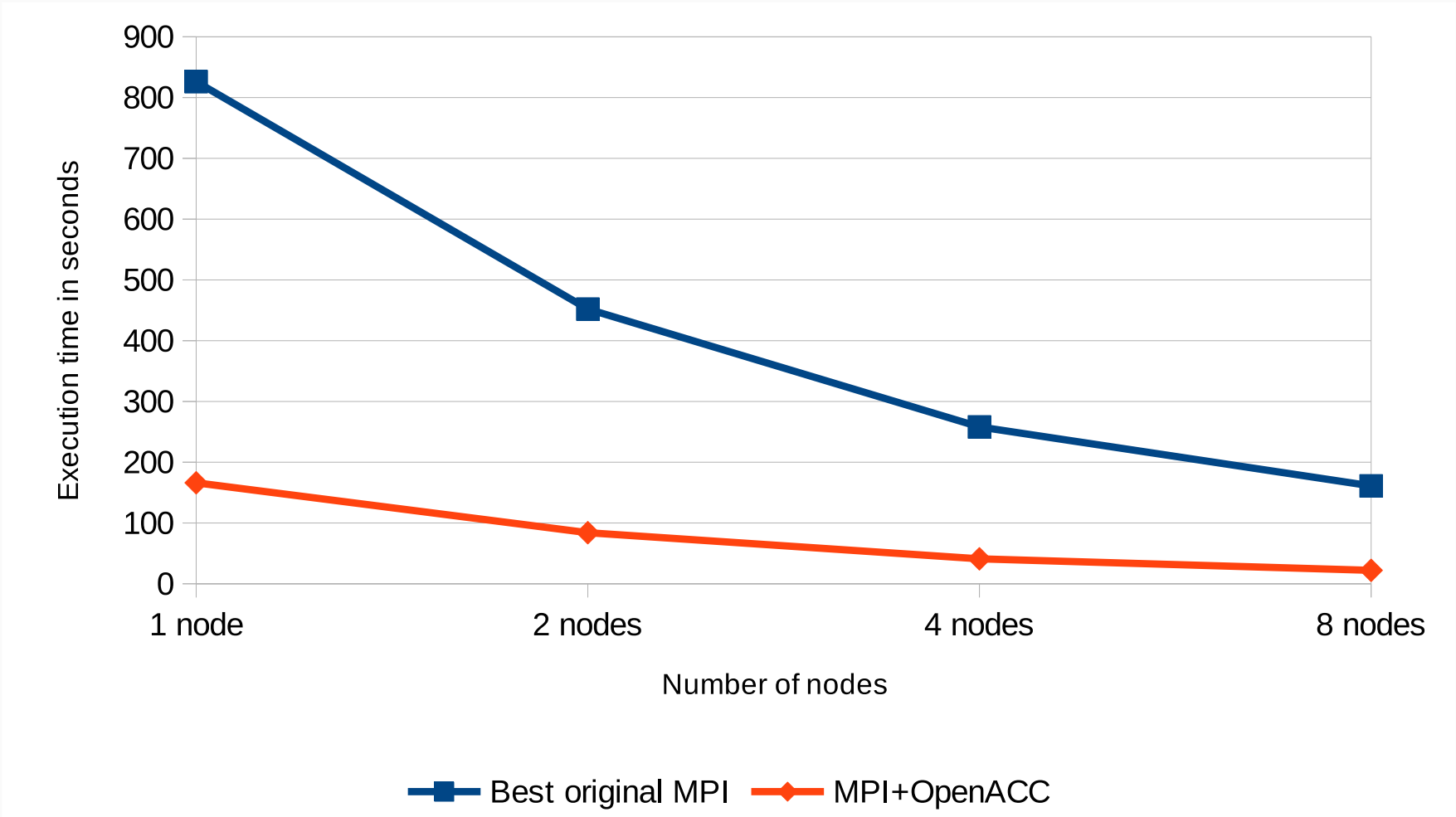
MPI+OpenACC (PGI compiler) on GPU

Move data to GPU and execute the loop on it.

- !$acc data copy(array)
- !$acc parallel loop collapse(4)
- do loop
- do loop
- do loop
- !$acc atomic
- array(x,y,z) = array(x,y,z) + k
- ...
- !$acc end data

Elsewhere during the initialisation of the code, we have linked each MPI task to a specific GPU.

MPI+OpenACC (PGI compiler) on GPU



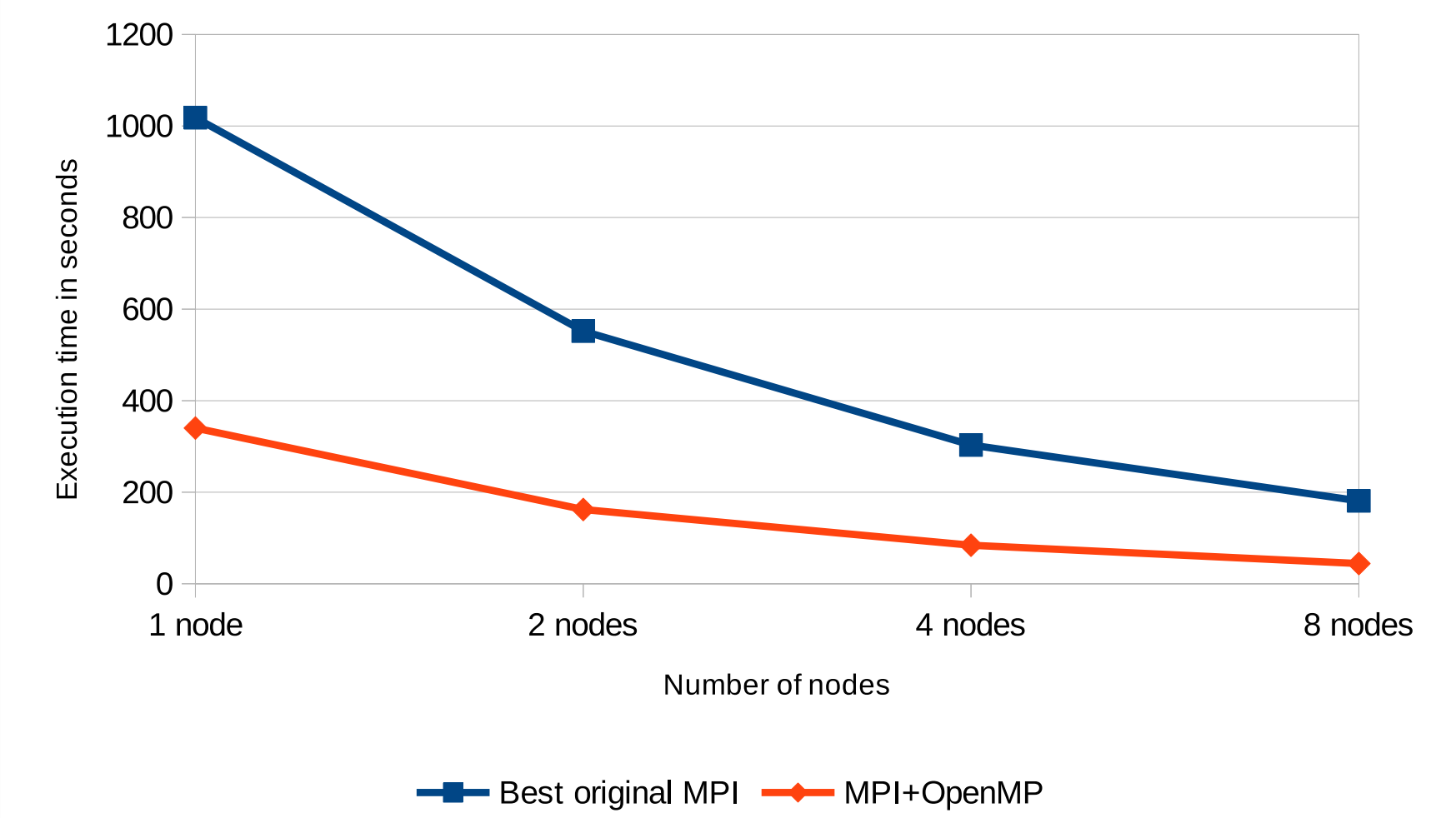
MPI+OpenMP (IBM compiler) on GPU

Move data to GPU and execute the loop on it.

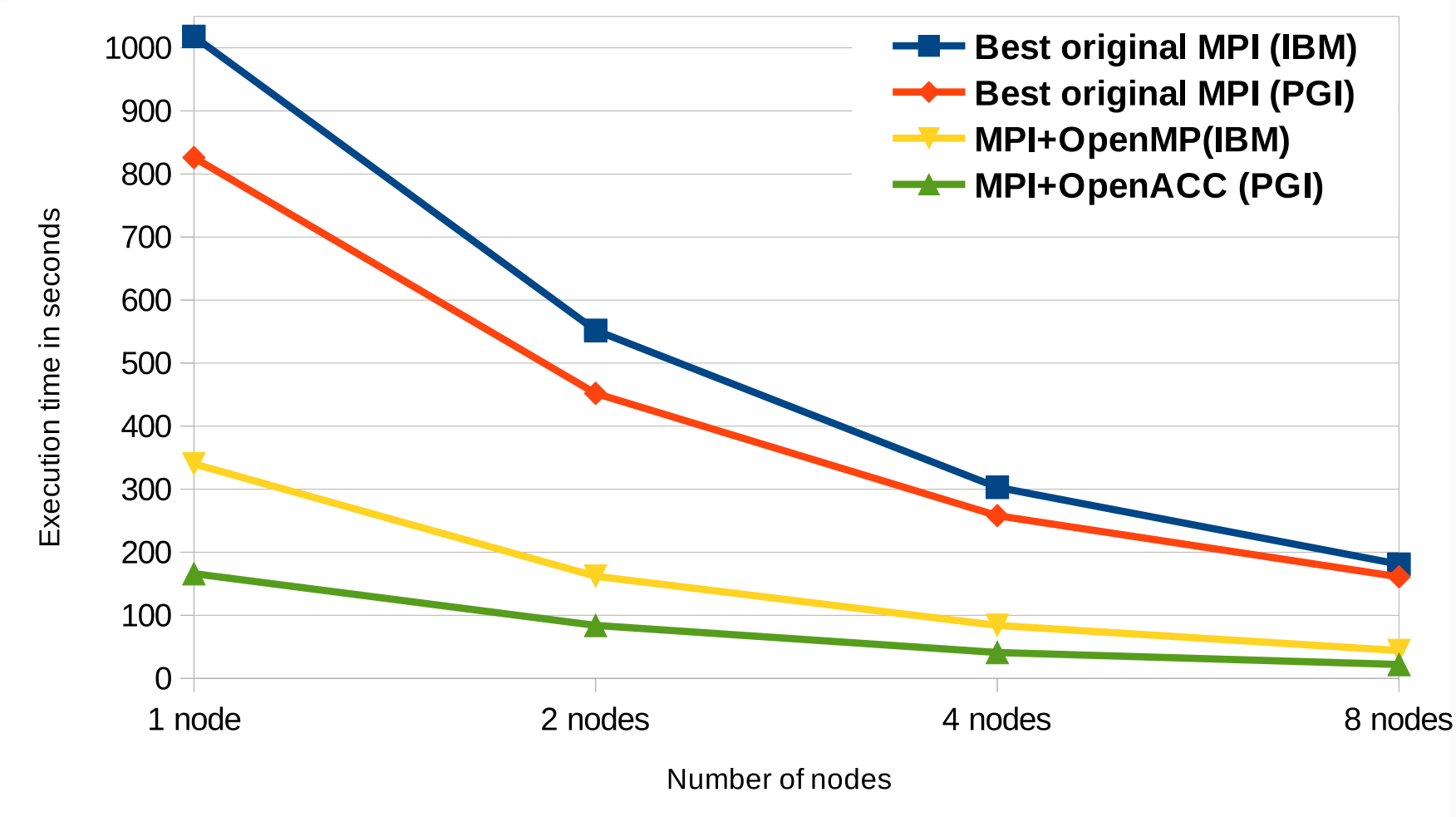
- !$omp target data map(array)
- !$omp target teams distribute parallel do collapse(4)
- do loop
- do loop
- do loop
- !$omp atomic
- array(x,y,z) = array(x,y,z) + k
- ...
- !$omp end target data

Elsewhere during the initialisation of the code, we have linked each MPI task to a specific GPU.

MPI+OpenMP (IBM compiler) on GPU



MPI+OpenMP (IBM compiler) VS MPI+OpenACC (PGI compiler)



Conclusion

Results achieved:

- No improvement when using SMT with OpenMP
- Good improvement when using GPU
- Between 4.8 and 7.3 speedup with OpenACC
- Between 3 and 4.1 speedup with OpenMP

Technical advices:

- PGI compiler is helpful and gives informative messages about how the compiler translates the OpenACC directives
- The nvprof profiler is able to profile the OpenACC code, which lets you efficiently visualize when data transfers occur
- We have been unable to use it with OpenMP code

Future work

- Offloading more parts of TELEMAC–MASCARET to GPU
- Keeping track of the enhancements of OpenACC and OpenMP implementations across different compilers
- Producing increasingly large simulation meshes and proving better convergence, provided by higher resolutions, enabled by faster processing

Acknowledgements

- This work is supported by the Hartree Centre through the Innovation Return on Research (IROR) programme.

Science & Technology Facilities Council
Scientific Computing Department

Science & Technology Facilities Council | UK Research and Innovation

Hartree Centre
Science & Technology Facilities Council

Hartree Centre
Science & Technology Facilities Council

Contact:
judicael.grasset@stfc.ac.uk