Porting and Optimising the TELEMAC-MASCARET Hydraulics Suite for POWER8

Judicaël Grasset[1] Yoann Audouin[2] Stephen Longshaw[1] Charles Moulinec[1]

[1]STFC, Scientific Computing Department, Daresbury Laboratory, Warrington, United Kingdom

[2]EDF R&D, Chatou, France

Objectives and Outcomes

- **Making sure TELEMAC-MASCARET compiles and runs with GNU gfortran and IBM xlf on a POWER8 machine.**
- **Using efficiently the simultaneous multithreading (SMT) capability of the POWER8.**
- **Exploiting the NVlink capability by porting some parts of TELEMAC to Nvidia Tesla P100 GPUs.**
- **Enables users to extract more performance from the same number of cores.**

Introduction

TELEMAC-MASCARET is an open-source suite of solvers for free surface flow modelling. This work refers to the wave propagation module TOMAWAC.

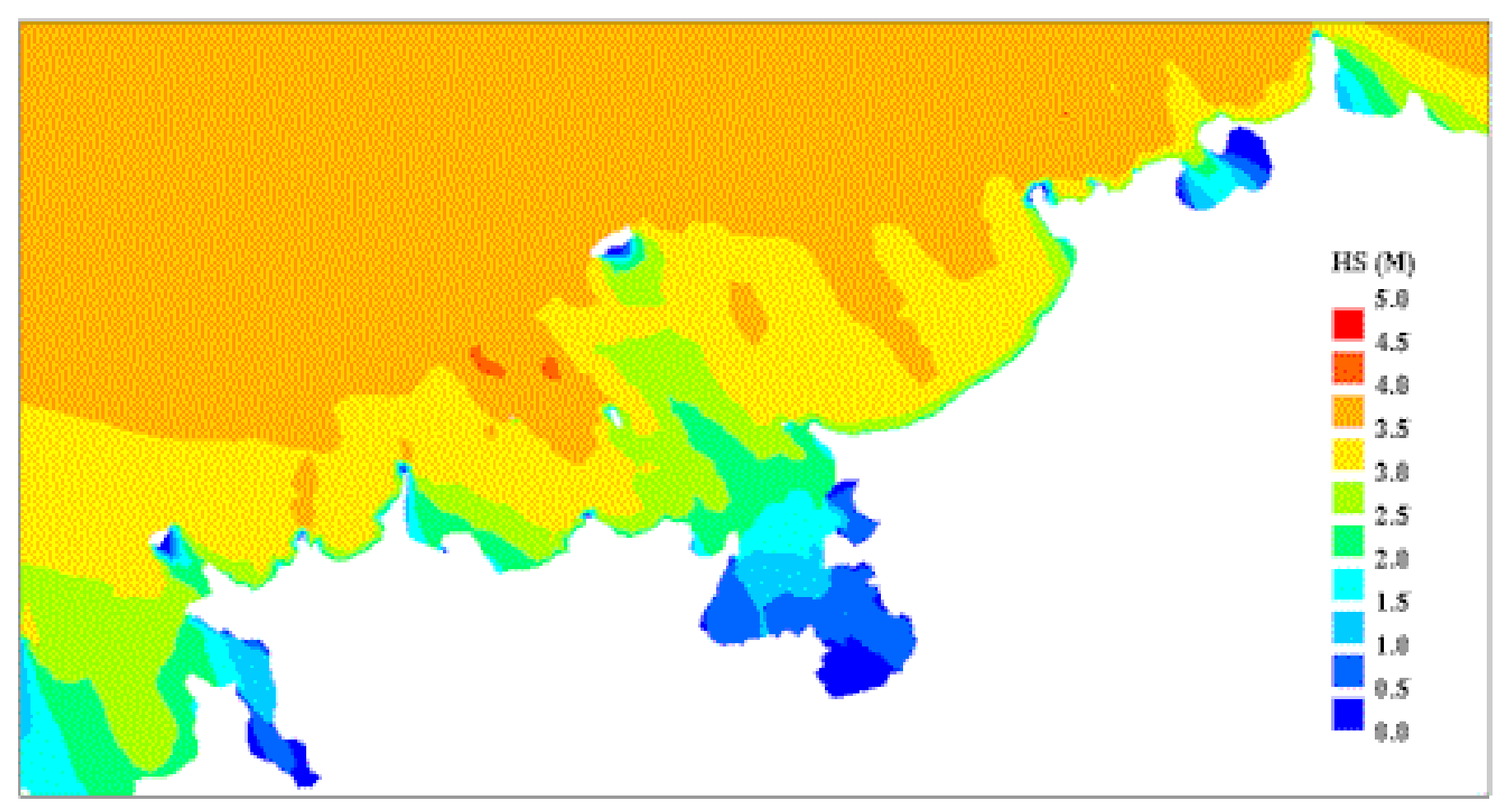

Figure 1: Wave propagation near the port of Saint-Malo. Image taken from the TELEMAC website.

The code is able to run on thousands of cores thanks to efficient parallelisation with MPI. However, the speedup observed using MPI has a limit. Each MPI process computes a fraction of the mesh and as the number of MPI processes increases, the part of the mesh to work on is reduced, and so are the benefits of the parallelisation. In this poster we show how to solve this problem with the use of hybrid MPI/OpenMP.

The testing machine

This work uses the Panther and Paragon POWER8 clusters at the Hartree Centre at Daresbury Laboratory. Each node of the cluster is made of two POWER8 CPUs and 4 GPUs (either K80 or P100-SXM2+NVLink). Each CPU comprises 8 cores and is able to run 128 hardware threads (8 threads on each core) thanks to the SMT8 capability of POWER8.
Our work has shown that using 8 threads per core doesn't provide the best performance for TOMAWAC. In our benchmarks the best execution time was never reached when using 8 threads per core but when using 2 or 4.

Solution

Profiling the code shows that 95% of the time is spent in a computationally demanding quadruple nested loop. These loops have no memory dependency on each other, therefore the outer loop can be parallelised with OpenMP using the *parallel do* directive.

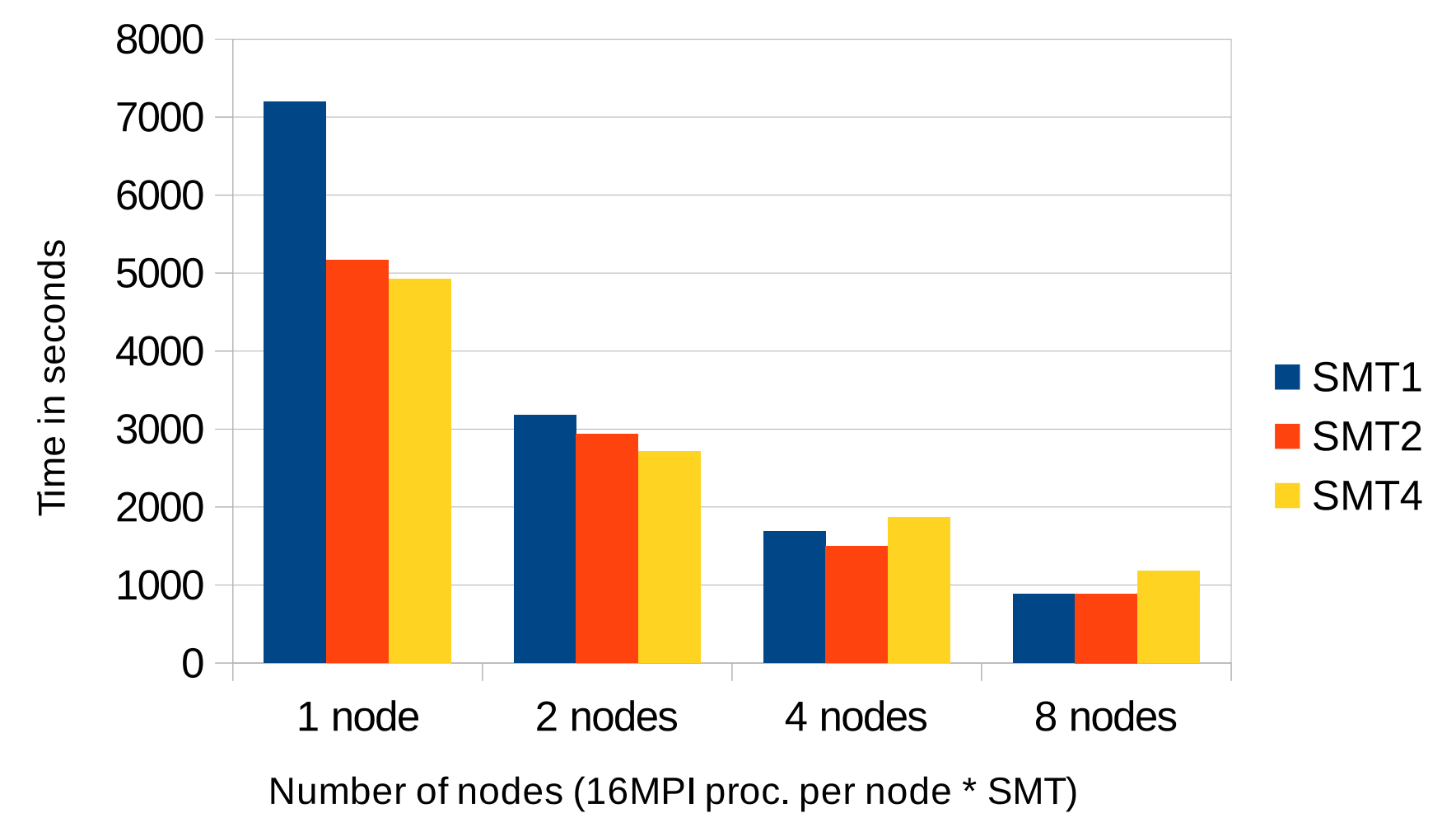

Figure 2: Time to solution for the original pure MPI implementation

Results

Figure 2 shows the timing for TOMAWAC to complete a simulation as a function of the number of nodes and SMTs used. Using SMT2 is always useful, but not SMT4. This is mainly because the number of elements that each MPI process uses to carry out the computations is too low in our example case, future work will look at bigger cases that are not currently typical of current users of TELEMAC-MASCARET. The same figure also shows that even if SMT is useful it is not as good as having a single physical core for each thread: using 1 node with SMT4 allows use of 64 MPI processes. This is also true for 4 nodes with SMT1, but the latter is much faster.

Figure 3 shows the time to solution of TOMAWAC when using the loop parallelised with OpenMP in addition to the existing MPI implementation. One MPI process is used per processor but another possibility would be to use 16 MPI processes per processor and use SMT to run the OpenMP threads. This last method has proven to be a little less efficient in our benchmarks.

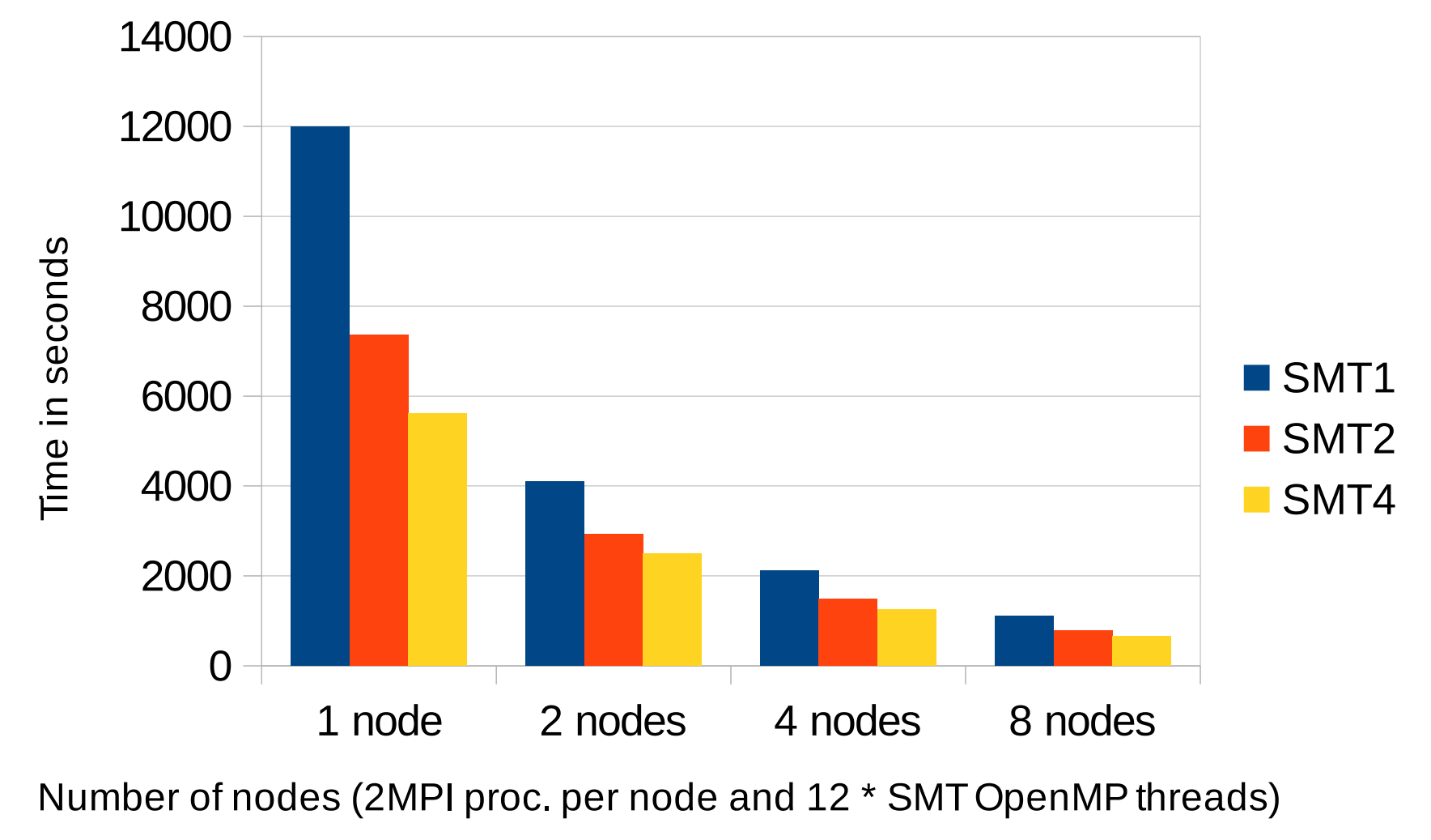

Figure 3: Time to solution for the hybrid MPI/OpenMP implementation

Table 1 presents the execution time of the two versions, using SMT or not. It can been seen that without SMT, the hybrid MPI/OpenMP version shows worse performance than pure MPI. However with SMT, the MPI version is not always the best. This is because the number of elements per subdomain, and hence per MPI process is too low. Conversely when we use the hybrid version which spawns less MPI processes and use SMT to create more OpenMP threads there is still a clear acceleration.

	1 node	2 nodes	4 nodes	8 nodes
MPI	7195	3178	1687	887
MPI SMT(2/4)	4927	2716	1493	888
MPI/OpenMP	11986	4104	2123	1108
MPI/OpenMP SMT4	5622	2492	1259	655

Table 1: Comparison of the time to solution for the pure MPI and the hybrid MPI/OpenMP versions

Conclusions

A method has been presented to improve the execution speed of the TELEMAC-MASCARET wave propagation module (TOMAWAC) when the number of elements per MPI process is relatively small (about 140 elements or less). This method demonstrates that it is possible to get up to 35% more performance by using existing hardware more efficiently.

Future work

In the future we will make use of the GPUs in the POWER8 platform to offload the loop. We hope that thanks to NVLink the time spent in sending data to the GPU and retrieving data from the GPU will be negligible. We will implement this offloading with the two main set of directives available: OpenACC and OpenMP.

Miscellaneous

While porting TELEMAC-MASCARET on POWER8 we found bugs in both compilers gfortran and xlf that prevented the compilation of the suite of solvers. All of them have been reported to the respective maintainers with a reproductible test case. The bug in xlf has been fixed, the one in gfortran is still under investigation.

Acknowledgements

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

Contact Information

- Email: judicael.grasset@stfc.ac.uk

Science & Technology Facilities Council | UK Research and Innovation

Hartree Centre
Science & Technology Facilities Council

eDF