Code_Saturne's Extreme Scaling on IBM Blue Gene/Qs

Charles Moulinec[1], Yvan Fournier[2], Ales Ronovsky[3], Vendel Szeremi[1], Pascal Vezolle[4], David R. Emerson[1]

[1]SCD, STFC Daresbury Laboratory, Sci-Tech Daresbury, Keckwick Lane, Daresbury, Warrington, WA4 4AD, UK
[2]EDF R&D, 6 quai Watier, 78400 Chatou, FR
[3]Department of Applied Mathematics, VSB-Technical University of Ostrava, 708 00 Ostrava, CZ
[4]IBM France, 1 Rue de la Vieille Poste - 34006 Montpellier, FR

Introduction

Code_Saturne [1, 2] is an open-source multi-purpose CFD software developed by EDF-R&D (see Fig. 1 for its toolchain). It is currently being prepared for Exascale and with this objective tested on some of the largest existing high-end machines.

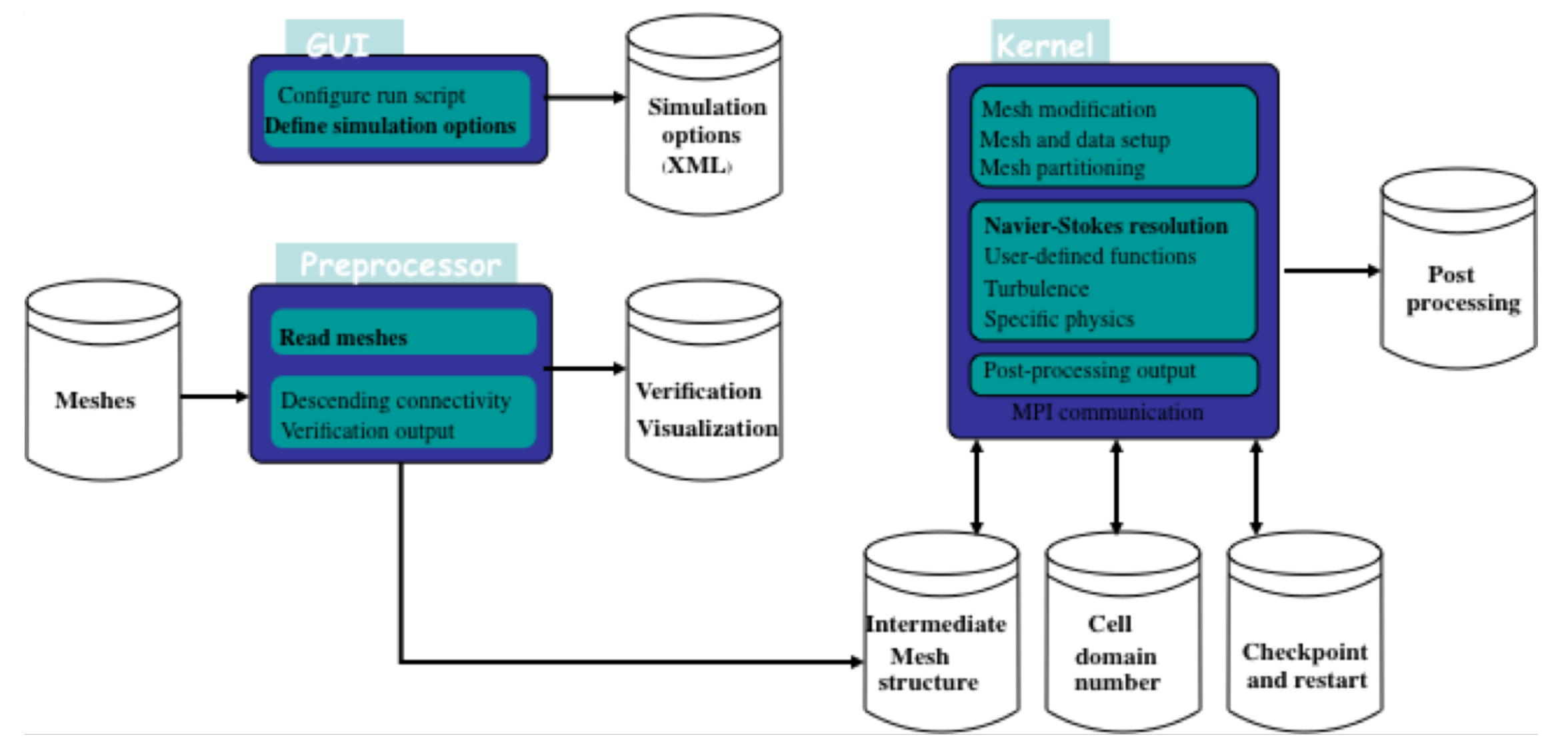

Figure 1: *Code_Saturne*'s toolchain

This works focuses on assessing *Code_Saturne* 4.2.0's performance on IBM Blue Gene/Qs (BGQs), namely Mira (Argonne) [3] and Juqueen (Jülich) [4], ranked #5 and #11 (November 2015's Top500 list).

Hardware - Settings

Mira and Juqueen are made of 1,024 node racks with 16 cores or 64 threads per node (16GiB RAM per node). Mira has got 48 racks and Juqueen 28.
Code_Saturne uses MPI/OpenMP for parallelism, and the Modified Compressed Sparse Row (MSR) algorithm is used here as a storage format.

Test Cases

The first test case consists of the flow in staggered-distributed tube bundles. The computational domain is made of 2 full tubes leading to an original 26 million cell (26M) hexa mesh. Mesh multiplication (MM) [5] is applied 4 times to this mesh to generate a 105 billion cell mesh (**TB_105B**). The second test case deals with the 3-D flow in a lid-driven cubic cavity. The orginal mesh has got 13 million tetra cells and MM is applied 3 times to generate a 7 billion cell mesh (**LDC_7B**).

Partitioning - Halos

Partitioning is carried out by Space Filling Curve (Hilbert) and its performance is shown in Tab. 1. This stage requires using almost 1 GiB per MPI task and 16 ranks/node are used for all the simulations.

LDC_7B		TB_105B	
#nodes × #ranks	Time	#nodes × #ranks	Time
4,096 × 16	56 s	32,768 × 16	112 s
8,192 × 16	57 s	49,152 × 16	89 s
16,384 × 16	56 s	-	-
28,672 × 16	73 s	-	-

Table 1: Time to perform each partition

Ghost cells are set to exchange information between neighboring subdomains (see timings in Tab. 2).

LDC_7B		TB_105B	
#nodes × #ranks	Time	#nodes × #ranks	Time
4,096 × 16	32 s	32,768 × 16	112 s
8,192 × 16	40 s	49,152 × 16	376 s
16,384 × 16	56 s	-	-
28,672 × 16	77 s	-	-

Table 2: Time to create the halo cells

IOs

Both BGQs file systems are GPFS. *Code_Saturne* relies on MPI-IO for IOs, using a single 'shared file'. Table 3 (resp. 4) shows the time to write (resp. read) the mesh file on (resp. from) disk. For the **TB_105B** case, the file is **19 TiB** large.

TB_105B - 19 TiB	
#nodes × #ranks	Time
16,384 × 16	2,697 s
32,768 × 16	2,536 s

Table 3: Time to write the mesh_output file on disk

LDC_7B - 620 GiB		TB_105B - 19 TiB	
#nodes × #ranks	Time	#nodes × #ranks	Time
4,096 × 16	46 s	32,768 × 16	112 s
8,192 × 16	28 s	49,152 × 16	262 s
16,384 × 16	32 s	-	-
28,672 × 16	40 s	-	-

Table 4: Time to read the mesh_input file from disk

Solver

The Navier-Stokes solver is segregated and the velocity-pressure coupling computed by a fractional-step method. The 3 velocity components are coupled and solved using the Jacobi algorithm. The pressure is computed either using the Algebraic Multigrid (AMG) algorithm as a solver with a diagonal preconditioner (**D + AMG**) or as a preconditioner with the Conjugate Gradient algorithm as a solver (**AMG + CG**). The first five time-steps are computed for all the simulations.

LDC_7B - D + AMG			
MPIs	1 thread	2 threads	4 threads
4,096 × 16	394 s	255 s	255 s
8,192 × 16	251 s	173 s	148 s
16,384 × 16	132 s	96 s	81 s
28,672 × 16	82 s	62 s	56 s
LDC_7B - AMG + CG			
MPIs	1 thread	2 threads	4 threads
4,096 × 16	363 s	231 s	191 s
8,192 × 16	214 s	144 s	117 s
16,384 × 16	111 s	76 s	63 s
28,672 × 16	67 s	48 s	41 s

Table 5: CPU time per time-step

TB_105B - D + AMG			
MPIs	1 thread	2 threads	4 threads
32,768 × 16	236 s	185 s	117 s
49,152 × 16	170 s	141 s	135 s
TB_105B - AMG + CG			
MPIs	1 thread	2 threads	4 threads
32,768 × 16	133 s	112 s	108 s
49,152 × 16	95 s	82 s	78 s

Table 6: CPU time per time-step

Tables 5 and 6 gather the CPU time per time-step as a function of the number of MPI tasks and threads for **LDC_7B** and **TB_105B**. All the tests show that using 4 threads per MPI tasks is faster than using 1 or 2 threads. Moreover, for a given number of threads per MPI task, a speed-up is observed in all the cases when the number of MPI tasks is increased. **AMG + CG** is always faster than **D + AMG**.

Final Remarks - Future Work

This work assesses *Code_Saturne* 4.2.0's performance for 2 types of meshes (hexas only or tetras only), using MPI/OPenMP on two of the largest existing BGQs. Simulations were run up to **3,145,728 threads** on Mira and **1,835,008 threads** on Juqueen, and reasonable scaling was observed for both, with a clear gain using 4 threads per MPI task instead of 1 or 2. The IO tests carried out using MPI-IO were conclusive, showing that reading the mesh_input file has the same cost as a few time-steps.
Exascale performance will only be achieved by using accelerators. Several teams are exploring various options to optimise the code for Intel Xeon Phis and Nvidia GPUs.

Acknowledgements

This research used resources of the Argonne Leadership Computing Facility at Argonne National Laboratory (INCITE-PEAC), which is supported by the U.S. Department of Energy under contract DE-AC02-06CH11357. The authors would like to thank the Forschung Zentrum Jülich for performing their tests during the 2016 Extreme Scaling Workshop.

References

[1] F. Archambeau *et al.* Int. J. on Fin. Vol. (2004).
[2] Y. Fournier *et al.* Comp. & Fluids. (2011).
[3] `https://www.alcf.anl.gov`
[4] `http://www.fz-juelich.de`
[5] A. Ronovsky *et al.* 3^{rd} PARENG. (2013).