Hybrid MPI+OpenMP parallelization of an FFT-based 3D Poisson solver that can reach 10^5 CPU cores

A. Gorobets[1,2], F. X. Trias[1], R. Borrell[3] M. Soria[1] A. Oliva[1]
Corresponding author: andrey@cttc.upc.edu

[1] Heat and Mass Transfer Technological Center, Technical University of Catalonia, Spain.
[2] Keldysh Institute of Applied Mathematics of RAS, Russia.
[3] Termo Fluids, S.L., Spain.

Abstract: This work is devoted to the development of efficient parallel algorithms for the direct numerical simulation (DNS) of incompressible flows on modern supercomputers. A Poisson solver for problems with one uniform periodic direction is presented here. It is extended with a two-level hybrid MPI+OpenMP parallelization. Advantages and implementation details for the additional OpenMP parallelization are presented and discussed. This upgrade has allowed to significantly extend the range of efficient scalability. Here, the solver has been tested up to 12800 CPU cores for meshes with up to 10^9 nodes. However, estimations based on the presented results show that this range can be potentially stretched beyond 10^5 cores.

Keywords: Poisson solver, hybrid parallelization MPI + OpenMP, Direct numerical simulation, preconditioned conjugate gradient, Schur complement, FFT

1 Introduction

Since the irruption of multi-core architectures, computing power growth is mainly based on increasing both the number of nodes and the number of cores per node. However, these tendencies bring new problems that must be solved in order to exploit efficiently the new computing potential. In the context of CFD of incompressible flows, the Poisson equation, that has to be solved at least once per time step, is usually the most difficult-to-parallelize part of the DNS algorithm.

The Poisson solver presented here is a combination of a block preconditioned Conjugate Gradient (PCG) method and a Fast Fourier Transformation (FFT). The FFT decomposes the original system into a set of independent 2D systems that are solved by means of the PCG algorithm. For the most ill-conditioned systems that correspond to the lowest frequencies in the spectral space the PCG solver is replaced by a Direct Schur-complement Decomposition (DSD) method to avoid slow convergence. The overall solver, hereafter named Krylov-Schur-Fourier Decomposition (KSFD) was implemented within distributed memory model using message-passing interface (MPI). Its applicability was limited by around 2000-6000 CPU cores depending on the mesh size and the scheme order. Main limiting factors are twofold: (i) the direct solver for the most ill-conditioned 2D system(s), and (ii) the parallelization in the periodic direction. For details about the solver and its components the reader is referred to [1].

In the present paper, the solver is upgraded by means of a two-level hybrid MPI+OpenMP parallelization that exploits better supercomputers with multi-core nodes. MPI couples nodes within the distributed memory model while OpenMP provides shared memory parallelism inside each node. Some

details about OpenMP parallelization issues, e.g. memory access problems, hardware and parallel bottlenecks, can be found for instance in [2]. This hybrid model is rather common in CFD. MPI+OpenMP based algorithms and comparison with the MPI-only approach can be found for example in [3]. In general, the hybrid approach improves the performance but not significantly. In [4], comparison MPI vs. MPI+OpenMP concludes that the latter outperforms the former only on a large number of CPUs. However, migration to a more complex parallelization can hardly be justified by a relatively small gain in performance. Hence, the present work is mainly devoted not just to improve the performance, but to heal the above-mentioned limitations allowing to engage efficiently many times more CPUs.

2 Overview of the mathematical model

The turbulent incompressible flow of Newtonian fluid is considered. The velocity field, u, is governed by the Navier-Stokes (NS) and continuity equations given by

$$\frac{\partial u}{\partial t} + (u \cdot \nabla)\, u = \frac{1}{Re}\nabla^2 u - \nabla p\,; \qquad \nabla \cdot u = 0, \tag{1}$$

where Re is the Reynolds number. Periodic boundary conditions are prescribed in the x-direction. The finite-volume symmetry-preserving spatial discretization of these equations reads

$$\Omega \frac{du_h}{dt} + \mathsf{C}(u_h)\, u_h + \mathsf{D} u_h - \mathsf{M}^t p_h = 0_h, \tag{2}$$

where the discrete incompressibility constraint is given by $\mathsf{M} u_h = 0_h$. The diffusion matrix, D, is symmetric and positive semi-definite; the diagonal matrix, Ω, represents the sizes of the control volumes and the convection matrix, $\mathsf{C}(u_h)$, is skew-symmetric, *i.e.* $\mathsf{C}(u_h) + \mathsf{C}^t(u_h) = 0$. For a detailed explanation, the reader is referred to [5].

For the temporal discretization, a second-order explicit one-leg scheme is used for both the convective and diffusive terms. The pressure-velocity coupling is solved with the fractional step projection method in which a predictor velocity, u_h^p, is evaluated without considering the pressure gradient and then the incompressibility constraint, $\mathsf{M} u_h^{n+1} = 0_h$, leads to a Poisson equation for p_h^{n+1}: $\mathsf{L} p_h^{n+1} = \mathsf{M} u_h^p$ with $\mathsf{L} = -\mathsf{M}\Omega^{-1}\mathsf{M}^t$, where the discrete Laplacian operator, L, is a symmetric negative semi-definite matrix. Thus, the original 3D system to be solved is $\mathbf{A}^{3D}\mathbf{x}^{3D} = \mathbf{b}^{3D}$, where $\mathbf{A}^{3D} = -\mathsf{L} \in \mathbb{R}^{N\times N}$ and $N = N_x \times N_y \times N_z$ is the total number of nodes ($3D$ means each unknown is coupled with its neighbors in the three spatial directions).

The Poisson solver is presented in detail in [1]. Shortly, the original 3D system is decomposed by means of FFT into the set of independent 2D systems: $\hat{\mathbf{A}}_i^{2D}\hat{\mathbf{x}}_i^{2D} = \hat{\mathbf{b}}_i^{2D}$, where $i = 1, \cdots, N_x$ .. Since the matrices $\hat{\mathbf{A}}_i^{2D}$ are symmetric and positive-definite a Preconditioned Conjugate Gradient (PCG) method has been chosen. The 2D systems are ordered descending conditioning number, hence first planes that correspond to lower Fourier frequencies are rather ill-conditioned. In this case the PCG algorithm is inefficient and a direct method is used instead for the first D planes, where D is a delimiting parameter that can be chosen. The resulting algorithm is following:

1. The change-of-basis from physical to spectral space for the right-hand-side with a standard FFT.
2. Solve the first D decoupled 2D systems using the direct DSD solver.
3. Solve the remaining 2D systems using a PCG method.
4. Restore in the physical space the solution sub-vectors with inverse FFT.

The domain is partitioned by dividing the periodic direction into P_x parts and the $x-$orthogonal plane into P_{yz} parts. The parallelization of the solver can be divided into two parts: (i) "explicit" steps 1 and 4 of the algorithm, and (ii) steps 2 and 3, that involve the solution of 2D systems.

	N_x	N_y	N_z	N	Ra	Pr	Order
Mesh1	128	192	462	$\approx 11.4\times10^6$	10^{11}	0.71	4^{th}
Mesh2	256	800	1600	$\approx 327.7\times10^6$	10^{11}	0.71	2^{nd}
Mesh3	256	1400	2800	$\approx 1003.5\times10^6$	10^{11}	0.71	2^{nd}

Table 1: Physical and numerical parameters of the test cases.

Parallelization of FFT with MPI was out of consideration, instead it has been replicated in the x direction. This requires additional broadcast communication to restore the whole N_x-subvectors within P_x-groups. Then each process can perform FFTs sequentially. FFTs itself do not take much time but this group communication is the main limiting factor for the P_x number. In [1] was shown that P_x can be taken up to 8 with reasonable efficiency.

On the other hand, the number P_{yz} is also limited due to the scalability limitations of the DSD solver that is used on the step 2 of the algorithm. The main bottleneck comes from the solution of the Schur complement (or interface) system that grows fast with P_{yz} and with the mesh size and with the scheme order. According to our experience, for real CFD applications this DSD solver is feasible for P_{yz} up to $200\sim300$ and $500\sim800$ with the fourth- and second-order scheme, respectively. In summary this MPI-only Poisson solver can be efficient only on up to $P = P_x \times P_{yz} = 2000\sim6000$ CPUs (depending on the scheme order).

3 Extension with shared memory parallelization

If a parallel system has P_t cores per node then each MPI process can spawn P_t OpenMP threads engaging $P_x \times P_{yz} \times P_t$ CPU cores in total. The two-level decomposition is now used and MPI subdomains are divided further into P_t parts. In this way, to engage a given number of CPU cores, P, the size of MPI group, $P_x \times P_{yz}$, can be decreased by a factor of P_t meaning that either P_x or/and P_{yz} can be taken smaller compared to the MPI-only approach. Doing so, the size of the Schur complement system of the DSD solver reduces proportionally to P_{yz} healing its scalability limitations. Similarly, the cost of group communication in the periodic reduces proportionally to P_x improving the parallel efficiency. Moreover, the explicit part of the algorithm also benefits because the total size of MPI communications reduces proportionally to $P_x \times P_{yz}$. Altogether, it allows the solver to engage efficiently around P_t more CPUs and adapt the code easily to different computer architectures.

Parallelization of the explicit parts of the CFD algorithm (momentum and energy equations, etc.) is rather straightforward. Computations are divided among threads while MPI communications are performed by the master thread only. The computations over a subdomain are mainly organized by means of three nested loops along z-, y- and x-directions, where the inner loop corresponds to the periodic x-direction. Only the outer loop, in z-direction, is decomposed.

The FFT steps 1 and 4 of the Poisson solver are parallelized in the same vein. There are two nested loops along the z-and y-directions. FFTs are called inside the inner loop for each subvector in the periodic x-direction. Again, the outer loop is decomposed and each thread performs independently its own subset of sequential FFTs. Finally, on steps 2 and 3 there is a set of independent $2D$ systems that is divided between threads. Each thread just solves its own systems completely isolated in memory.

The test case chosen to measure the performance corresponds to a DNS of a turbulent natural convection flow in a differentially heated cavity (DHC) [6] illustrated on Figure 1 (right). Mesh size ranges from 11 million to 1 billion of grid points (see Table 1).

The first test shows the speedup with OpenMP for both the Poisson solver itself and the overall CFD algorithm using Mesh1. The numbers $P_{yz} = 64$, $P_x = 1$ are fixed while P_t varies from 1 to 8. The test has been carried out on the MVS-100K supercomputer and results are shown in Figure 1

(left). For MPI communications there is no parallel speedup as only the master thread communicates. Nevertheless communications are around 2.5 times faster with OpenMP because MPI processes are not queueing to access the network. For computations speedup is higher and the overall speedup is around 5.5. The average wall clock time for the Poisson solver and the overall algorithm is 0.11 and 0.29 seconds respectively on 1024 CPU cores.

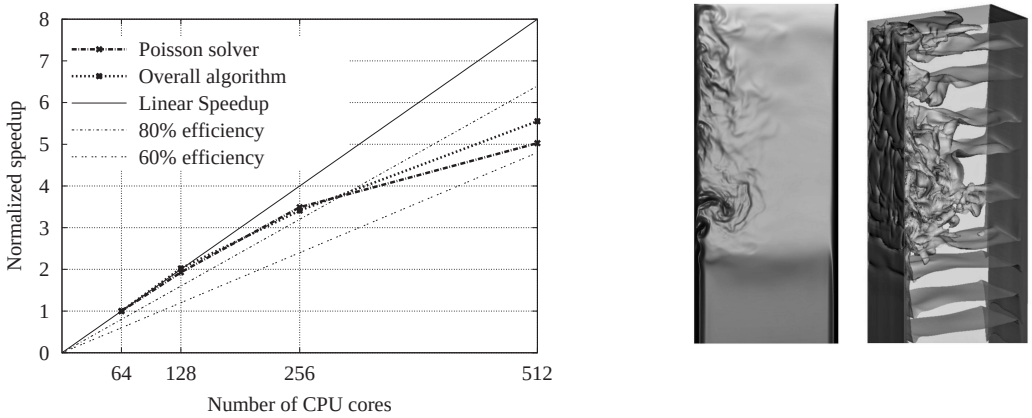

Figure 1: OpenMP speedups ($P_t = 1 \sim 8$) of the Poisson solver and the overall algorithm on Mesh1with 64 MPI processes (left) and instantaneous temperature field in the DHC case (right).

The following tests were performed on the Lomonosov supercomputer to demonstrate the applicability of the algorithm for bigger meshes and numbers of CPUs. In this case, $P_{yz} = 200$ and $P_t = 8$ are kept constant while P_x varies from 1 (1600 CPU cores) to 8 (12800 CPU cores). The wall clock computing time for the Poisson solver reduces from 0.78 to 0.22 (Mesh2) and from 3.82 to 1.00 (Mesh3) seconds, respectively. Speedup results for both the Poisson solver and the overall CFD code are displayed in Figure 2 (left). Measurements show that speedup is not linear, the parallel efficiency of the Poisson solver reduces around 20% each time P_x is doubled.

There were only 12800 cores available for the tests, therefore, $P_{yz} = 200$ was chosen to test the limiting configuration when P_x and P_t reach its maximal value, 8, at the end. Of course, it would be more efficient to increase P_{yz} instead of P_x to engage more CPUs and, in fact, P_{yz} could be taken much bigger. The following test demonstrates this. Figure 2 (right) shows speedup results for Mesh2 and Mesh3when P_{yz} ranges from 200 up to 800 while $P_x = 1$ and $P_t = 8$ are fixed. For the coarser Mesh2, solver already starts loosing efficiency at the end whereas for the finer Mesh3 the speedup is still near linear. Taking $P_x = 8$ would allow us to engage $800 \times 8 \times 8 = 51200$ CPU cores. In the light of this results, we can estimate that the range of efficiency goes beyond 50 thousand cores for meshes with 1 billion grid points or more.

Moreover, the Poisson solver can extend further by exploiting mesh symmetries that are common in DNS. Let us consider the limitations on P_{yz} related with the DSD solver. If there is a symmetry in at least one of the two non-periodic directions then the solution a 2D system $\mathbf{Ax} = \mathbf{b}$ with DSD can be reduced to the independent solution of two twice smaller symmetric counterparts $\mathbf{A}^+\mathbf{x}^+ = \mathbf{b}^+$ and $\mathbf{A}^-\mathbf{x}^- = \mathbf{b}^-$. Doing so, instead of solving one system with P_{yz} processes and N_{yz} unknowns, 2 systems with $N_{yz}/2$ unknowns are simultaneously solved by two groups of $P_{yz}/2$ processes. This way the limitation on the CPU number extends twice leading for the cases with a spatial symmetry beyond horizon of 10^5 CPU cores. Unfortunately, it is far beyond the commonly available computational resources and cannot be demonstrated yet.

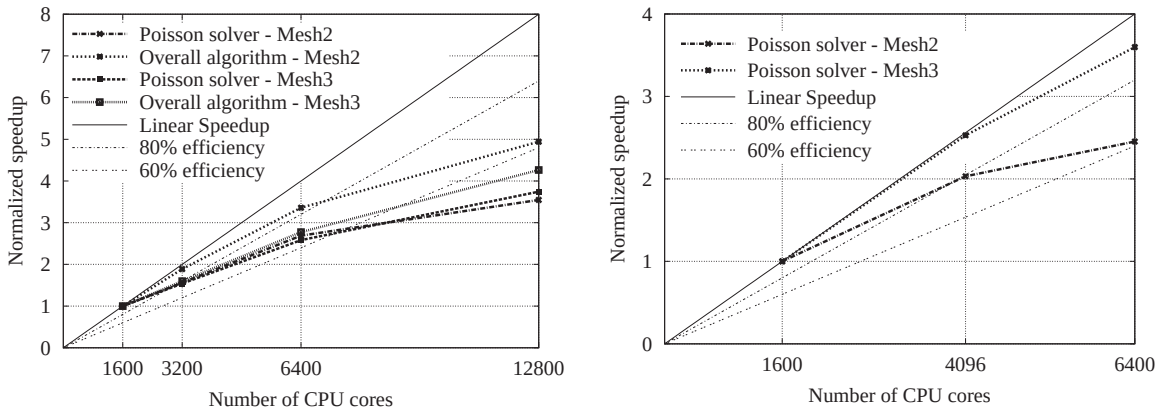

Figure 2: Speedups on the Lomonosov supercomputer of the Poisson solver and the overall algorithm on meshes Mesh2 (left) and Mesh3when varying $P_x = 1 \sim 8$ and keeping $P_{yz} = 200$ and $P_t = 8$ constant (left).

4 Conclusion

A parallel Poisson solver with MPI+OpenMP parallelization for incompressible flows with one periodic direction has been presented. OpenMP was preferred over other options such as POSIX Threads or a second level of MPI because it combines simplicity, efficiency and portability. The hybrid two-level MPI+OpenMP approach has significantly extended the range of scalability allowing the solver to use efficiently many times more CPUs. Speedup results up to 12800 cores for meshes up to 10^9 nodes have demonstrated the feasibility in solving large-scale DNS problems. Moreover, presented estimations show that more than 10^5 cores can potentially be engaged.

This work has been financially supported by the *Ministerio de Ciencia e Innovación*, (ENE2010-17801), (JCI-2009-04910) Spain, and the Russian Federation President grant MK-7559.2010.1. Calculations have been performed on the Lomonosov, MVS-100K (Russia), and MareNostrum (Spain) supercomputers. The authors thankfully acknowledge these institutions.

References

[1] A. Gorobets, F. X. Trias, M. Soria, A. Oliva, A scalable parallel Poisson solver for three-dimensional problems with one periodic direction, Computers & Fluids 39 (2010) 525–538.

[2] R. Aubry, G. Houzeaux, M. Vázquez, J. M. Cela, Some useful strategies for unstructured edge-based solvers on shared memory machines, International Journal for Numerical Methods in Engineering (2010) n/a. doi: 10.1002/nme.2973.

[3] Kengo Nakajima, Three-level hybrid vs. flat MPI on the Earth Simulator: Parallel iterative solvers for finite-element method, Applied Numerical Mathematics 54 (2) (2005) 237–255.

[4] Martin J. Chorley, David W. Walker, Performance analysis of a hybrid MPI/OpenMP application on multi-core clusters, Journal of Computational Science 1 (3) (2010) 168–174.

[5] R. W. C. P. Verstappen, A. E. P. Veldman, Symmetry-Preserving Discretization of Turbulent Flow, Journal of Computational Physics 187 (2003) 343–368.

[6] F. X. Trias, A. Gorobets, M. Soria, A. Oliva, Direct numerical simulation of a differentially heated cavity of aspect ratio 4 with *Ra*-number up to 10^{11} - Part I: Numerical methods and time-averaged flow, International Journal of Heat and Mass Transfer 53 (2010) 665–673.