V European Conference on Computational Fluid Dynamics
ECCOMAS CFD 2010
J. C. F. Pereira and A. Sequeira (Eds)
Lisbon, Portugal,14-17 June 2010

EFFICIENCY OF LARGE-SCALE CFD SIMULATIONS ON MODERN SUPERCOMPUTERS USING THOUSANDS OF CPUS AND HYBRID MPI+OPENMP PARALLELIZATION

A. V. Gorobets[*,†], R. Borrell[†], F. X. Trias[†], T. K. Kozubskaya[*] and A. Oliva[†]

[*]Institute for Mathematical Modeling of RAS,
Miusskaya sq. 4, 125047 Moscow, Russia
e-mail: cherepock@gmail.com
[†]Heat and Mass Transfer Technological Center, Technical University of Catalonia,
ETSEIAT, C/Colom 11, 08222 Terrassa, Spain
e-mail: cttc@cttc.upc.edu

Key words: parallel CFD, hybrid parallelization, DNS, supercomputer

Abstract. *This work represents an experience in using the hybrid parallel model to perform large-scale DNS. Advantages of the hybrid approach compared to the MPI-only approach are presented and discussed. The use of OpenMP in addition to MPI is demonstrated for modelling of compressible and incompressible flows using both structured and unstructured meshes. A parallel Poisson solver for incompressible flows with one periodic direction extended with the hybrid parallelization is presented. A two-level domain decomposition approach is considered for improving parallel algorithms for compressible flows. An alternative strategy with partial data replication is represented as well. Several DNS examples with mesh sizes varying from* 10^6 *to* 10^8 *control volumes are given to demonstrate efficient usage of the upgraded algorithms. Performance tests and simulations have been carried out on several parallel systems including Marenostrum, MVS-100000 and Lomonosov supercomputers.*

1 Introduction

Further progress in computational fluid dynamics is closely related with efficient usage of modern HPC systems. Performance of supercomputers grows rapidly mostly due to increase in the number of CPU cores. Despite more computing power available these new supercomputers bring new problems. In particular, a very large number of CPUs requires highly scalable algorithms. The new architecture with multi-core nodes motivates the use of a more complex parallel model - a hybrid parallel model that combines both distributed and shared memory models. Hence CFD algorithms that can perform well on such a big number of CPU cores are of high interest.

Huge computing power of new HPC systems is required in particular for large-scale direct numerical simulations (DNS). These numerical experiments contribute to the development of turbulence models like RANS, LES, DES etc. For this reason solutions of model academic problems with fine resolution and huge computing demands serves industrial purposes providing more efficient and reliable tools for engineering applications.

This work represents experience in using the hybrid parallel model to perform large-scale DNS. The use of OpenMP in addition to MPI is demonstrated for modelling of compressible and incompressible flows using both structured and unstructured meshes. Advantages of the hybrid approach compared to the MPI-only approach are presented and discussed for two different cases.

On the one hand there is a complex parallel solver for incompressible flows with one periodic direction[1,2]. It has explicit scalability limitations related in particular with requirements of its component, the Schur complement based direct solver[3]. Using shared-memory parallelization in this case is not just desirable but necessary in order to extend efficient range of CPU numbers.

On the other hand an algorithm for compressible flow is considered as an opposite example. It has rather efficient MPI parallelization without explicit limitations on scalability, that allows to use efficiently several thousands of CPUs even for relatively small meshes[4]. Hence using OpenMP in addition to MPI is much rather just desirable than really necessary. But it has also been shown that even in this case the use of OpenMP is profitable and allows to improve the solver performance.

In the next section the parallel Poisson solver extended with OpenMP parallelization is described. Performance with MPI+OpenMP parallelization is demonstrated and compared to MPI-only approach. The hybrid parallelization of the compressible flow algorithm is presented in section 3. Illustrative applications of both solvers are given in section 4. Finally section 5 is devoted to a brief summary of results and conclusions.

2 Parallel Poisson solver

The Poisson equation arises in modelling of incompressible flows from the incompressibility constraint. It has to be solved at least once at each time step and it is usually the main bottleneck from a parallel point of view. In the previous work[1,2] a scalable algorithm for Poisson equation was proposed. It performs well on both small clusters (even obsolete ones with poor network performance) and on supercomputers.

This algorithm named Krylov-Schur-Fourier Decomposition (KSFD) can use efficiently up to around a thousand of CPUs of a supercomputer. The FFT is used to uncouple original 3D system into set of 2D systems that can be solved independently with Schur complement based direct algorithm[3] or preconditioned conjugate gradient algorithm[5].

However, since the FFT decomposition is applied in one direction a mesh is restricted to be uniform in this direction and obstacles in flow can only be 2D shapes extruded along this direction. There are implementations of the solver for both structured meshes and unstructured meshes based on 2D triangular meshing extruded with constant step in the 3-rd direction[2].

The main scalability limitation comes from the Schur complement based solver that has to be used for solving at least one of the 2D systems (that corresponds to the lowest Fourier frequency). The present work is devoted to extend the previous algorithm to make it run efficiently on several times bigger number of CPUs and letting it reach a number around 10000 CPUs. The extension is based on the hybrid two-level parallelization approach that complements the standard MPI parallelization with OpenMP.

2.1 Overview of the CFD algorithm

The non-dimensional incompressible Navier-Stokes equations in primitive variables are considered

$$\nabla \cdot \mathbf{u} = 0 \tag{1a}$$

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\,\mathbf{u} = \frac{1}{Re}\Delta \mathbf{u} - \nabla p \tag{1b}$$

where Re is the non-dimensional Reynolds number.

Equations (1a-1b) are discretised on a staggered grid in space by fourth-order symmetry-preserving schemes[6] in case of the structured solver. The unstructured solver[2] uses a collocated grid and second-order scheme. For the temporal discretisation, a fully explicit dynamic second-order one-leg scheme[6] is used for both convective and diffusive terms. Finally, to solve the pressure-velocity coupling a classical fractional step projection method is used. Further details about the time-integration method can be found in[7,8].

Parallelization of the overall algorithm is straightforward except for the Poisson equation that has be solved on each time step. The parallel Poisson solver is considered further in details.

The original 3D system to be solved is following:

$$Ax = b. \tag{2}$$

Diagonalisation of blocks of the system (2) that correspond to discretisation in periodic direction leads by means of the FFT to the set of independent systems to be solved:

$$A_i^{2D} x_i^{2D} = b_i^{2D}, \qquad i = 1, ..., N_x, \tag{3}$$

where N_x is the number of nodes in the periodic direction. The 2D systems are ordered ascending the corresponding Fourier frequency. Hence first systems that correspond to lower frequencies are much more problematic for an iterative solver. The set is divided into 2 groups by delimiter $1 \leq D \leq N_x$. The first group is solved using a direct method, the second is solved using an iterative method. The algorithm of the Poisson equation solution is following:

1. Transformation of the r.h.s. vector b to the Fourier space by means of the FFT provides the set of r.h.s. b_i^{2D} for 2D systems;
2. Solution of $A_i^{2D} x_i^{2D} = b_i^{2D}, i = 1, ..., D$, using the direct Schur complement based solver;
3. Solution of $A_i^{2D} x_i^{2D} = b_i^{2D}, i = D + 1, ..., N_x$, using the block-preconditioned conjugate gradient iterative solver;
4. Transformation of solution vectors x_i^{2D} by means of FFT restores the solution x of the original 3D system (2).

The 3D domain is discretised uniformly along the periodic direction, x, and consists of N_x planes of N_{yz} nodes. Partitioning is done by decomposing the domain into P_x parts along the periodic direction and each plane is decomposed into P_{yz} parts engaging $P = P_x \times P_{yz}$ CPUs in total. Considering that FFT is hardly possible to parallelise efficiently within distributed memory model (especially for such small vectors around 100 500 elements) it has been replicated inside of the P_x-groups. The FFT is relatively fast and its replication would not affect much the overall efficiency but it also involves a group communication within P_x-groups that limits the P_x number. It has been shown in[1] that P_x can be taken up to 8 with reasonable efficiency and then performance goes down rapidly for bigger numbers. On the other hand P_{yz} is also limited due to the main scalability bottleneck of the Schur solver that is used on the stage 2 of the algorithm. This scalability limitation is described in details in[1]. In the Schur complement algorithm there is an interface system to be solved. This system growth with both the CPU number and the mesh size finally becoming hardly possible to solve. And this interface system is especially problematic for high-order scheme because of its large space stencil. For example with a mesh of 10^8 nodes and the 4-th order scheme the solver works well for P_{yz} around 100 - 200 but it is hardly possible to use it for say $P_{yz} = 500$. Limitations are mainly due to the preprocessing stage requirements, the memory requirements for interface system solution, the size of the group communication of reduction type that takes place in Schur solver to obtain contributions of interface nodes. Considering these limitations and according to our experience this Poisson solver with MPI-only parallelization can be efficiently used on say up to 1000-2000 CPUs. Nowadays that it is quite enough, but considering the fast grows of CPU number in supercomputers this range may soon become insufficient. For this reason MPI parallelization has been extended with additional OpenMP parallelization that works inside of multi-core nodes.

2.2 Additional OpenMP parallelization

The use of OpenMP in addition to MPI allows to extend the solver limitations P_t times, where P_t is the number of OpenMP threads. This means on a typical supercomputer with 8-core nodes the solver can run efficiently on 8 times bigger number of CPU cores. Parallelization of explicit parts of the overall CFD algorithm is rather straightforward and is done mainly by decomposing further MPI subdomains. The Poisson solver also fits well the shared memory model. Stages 1 and 4 of the solver algorithm are easily parallelised by second-level decomposition: each thread performs FFTs for its own subset of subvectors (note that each FFT itself is still sequential). Stages 2 and 3 of the algorithm are also easily made parallel with OpenMP: each thread solves its own subset of 2D systems, so threads work with their own sets of data with no intersections. The use of OpenMP gives following advantages:

1. The size of interface in Schur solver decreases P_t times hence decrease in memory requirements, preprocessing stage cost, communications, etc.

2. Size of communications and the number of communicating processes decreases around P_t times

3. More RAM memory available for MPI processes

All these allows the solver to use efficiently around P_t more CPUs. For a typical supercomputer with 8-core nodes it leads to the number around 10000 CPUs.

Performance tests for OpenMP parallelization have been performed on MVS-100000 (RAS) and Lomonosov (MSU) supercomputers on example of a real CFD application, the differentially heated cavity (DHC) DNS[9,10], for reduced-size mesh of $11 * 10^6$ nodes using 4-th order scheme. Figure 1 shows comparison of the Poisson solver speedup on Lomonosov supercomputer for MPI and OpenMP: $P_{yz} = 64$ is fixed, P_x varies from 1 to 8 in case of MPI and P_t varies from 1 to 8 in case of OpenMP. Results show using OpenMP is more efficient than decomposing periodic direction. Figure 2 left shows OpenMP speedup of the Poisson solver for MPI groups of 64 and 128 processes. Figure 2 right shows overall CFD algorithm OpenMP speedup (together with explicit part) for fixed $P_{yz} = 200$ and P_t varying from 1 to 8.

Speedups for OpenMP have been demonstrated on example of reduced mesh size $11*10^6$ nodes in order to save computing time. Typically the mesh size is more than 100 millions of nodes. Considering the previously achieved results[1] on MareNostrum supercomputer with MPI-only parallelization, the new results with MPI+OpenMP allow us to expect

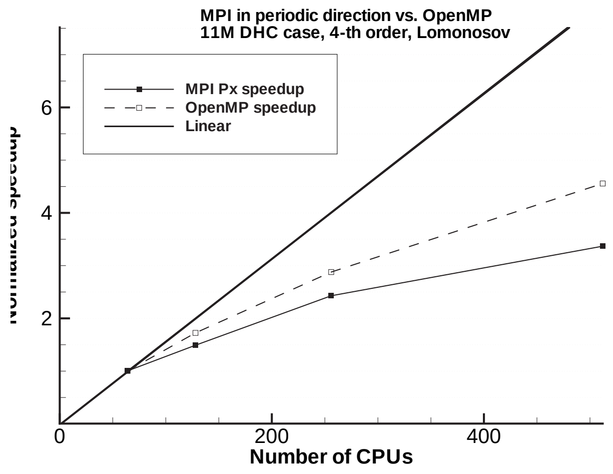


Figure 1: Comparison of MPI and OpenMP speedups on Lomonosov supercomputer

that the solver will run efficiently on around 10 thousands of CPUs when such a big number will be commonly available for single jobs in queue systems of supercomputers.

3 Parallel CFD algorithm for compressible flows

In the previous case the use of OpenMP was necessary because it allows to extend limitations by reducing interface size in Schur solver. This case in contrast does not require additional OpenMP parallelization. But it is shown further that even in this case the hybrid model gives advantages while being rather easy to implement.

The in-house code Noisette[11] designed for solving 2D and 3D CFD and computational aeroacoustics (CAA) problems for compressible flows using unstructured triangular and tetrahedral meshes and high-order algorithms[12]. The parallel algorithm is well-scalable for both explicit and implicit time integration.

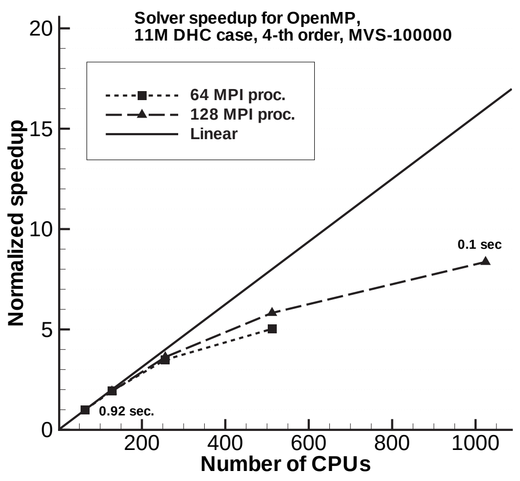
The use of OpenMP in this case improves performance mainly for big numbers of CPUs when the mesh size per core is small.

3.1 Overview of the Noisette

There are several basic models implemented in Noisette including Euler Equations (EE), Navier-Stokes Equations (NSE), Nonlinear Disturbances Equations (NLDE), Linearised Euler Equations (LEE). The system of equations for any of these models is of hyperbolic type and can be represented in a general form:

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{F}(\mathbf{Q})}{\partial x} + \frac{\partial \mathbf{G}(\mathbf{Q})}{\partial y} + \frac{\partial \mathbf{H}(\mathbf{Q})}{\partial z} = \frac{1}{Re}\left(\frac{\partial \mathbf{F}_\nu(\mathbf{Q})}{\partial x} + \frac{\partial \mathbf{G}_\nu(\mathbf{Q})}{\partial y} + \frac{\partial \mathbf{H}_\nu(\mathbf{Q})}{\partial z}\right), \quad (4)$$

where $\mathbf{Q}$ - is a vector of full or linearised conservative variables, $\mathbf{F}, \mathbf{G}, \mathbf{H}$ - vectors of



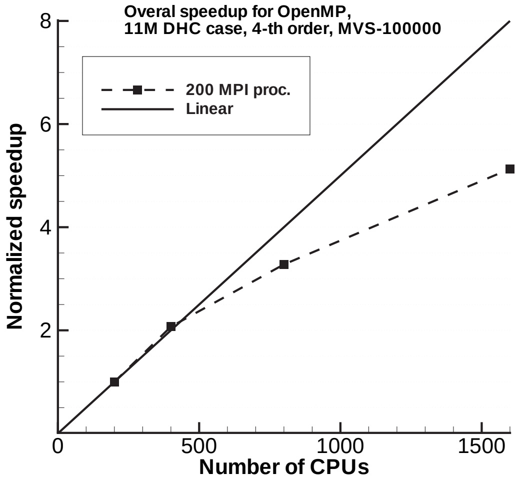


Figure 2: OpenMP speedup of the Poisson solver (left) and overall CFD algorithm (right) on MVS-100000

full or linearised conservative fluxes, $\mathbf{F}_\nu, \mathbf{G}_\nu, \mathbf{H}_\nu$ - vectors of full or linearised dissipative fluxes, Re - Reynolds number. A high order vertex-centred space approximation for unstructured meshes[12,13] is used for numerical solution of (4) regardless of model. The method is based on a finite-volume approach and a 2nd order approximation on arbitrary unstructured triangular or tetrahedral meshes. The method includes elements of a finite-difference approach and on "Cartesian" areas of mesh the finite-volume approximation of the 2nd order is exactly the finite-difference approximation of a high order (up to 6th depending on scheme parameters). Viscosity terms in the Navier-Stokes equations are approximated using finite-element method of 2nd order of accuracy. Time integration can be explicit Runge-Kutta up to 4-th order or implicit based on Newton linearisation. In case of implicit scheme a block-preconditioned stabilised bi-conjugate gradient solver is used.

In[14] it was demonstrated that Noisette can efficiently run on several thousands of CPUs even with relatively small meshes. To go further OpenMP parallelization has been implemented.

3.2 Additional OpenMP parallelization

The main advantage that OpenMP gives is reduction of time spent on communications due to the following reasons:

1. Number of communicating processes reduced P_t times
2. Size of interfaces between subdomains reduced around P_t times
3. Communications are faster also because there are no multiple processes queueing for network hardware inside of a multi-core the node.

The main problem for OpenMP is intersection in memory between threads on writing data. Such an intersection is considered here on a simple example of calculation of gradients: there is a loop over tetrahedrons and resulting data is added to nodes. Each node can belong to several tetrahedrons so when the loop over tetrahedrons is decomposed there can be several threads modifying data in the same node at the same time. This leads to inconsistency of data and must be avoided. This is done by the second level decomposition. MPI subdomain is decomposed further into P_t parts, mesh elements are reordered in a way that element from a part with higher rank has a bigger number. Then each thread only iterates tetrahedrons that have at least one node belonging to its part and each thread only updates values in nodes that belong to its part so the intersection is naturally avoided. There is some small overlap of computations in interface tetrahedrons that have nodes from different parts but this interface is minimised by proper partitioning with Metis. This kind of intersection also occurs in calculation of fluxes, viscosity, boundary conditions, and generally in matrix vector products. An alternative approach with replication of data can be used as well: certain arrays that are affected by intersection are replicated between threads, each thread writes to its own array, then master thread performs summation to resulting arrays. This approach is more simple but leads in particular to unnecessary memory consumption. It works well for example in case of calculation of fluxes because arrays to replicate are few but it doesn't work in case of gradients because there are too many arrays to replicate.

3.3 Performance tests on Lomonosov supercomputer

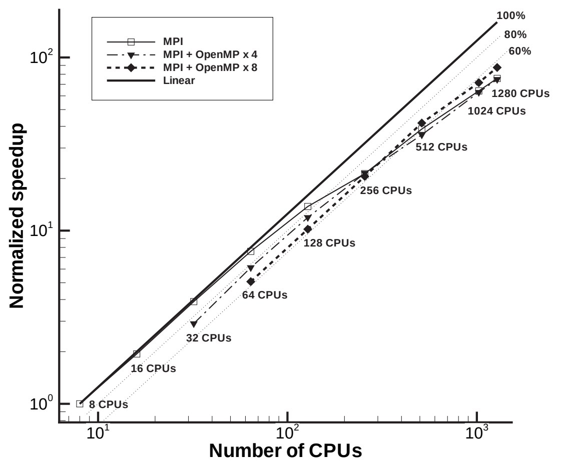
Computing time was measured using manual instrumentation profiling with high-resolution timing built in Noisette. Two test cases with reduced mesh size were taken from real application configurations:

1. "Flow around a finite cylinder", mesh with 783650 nodes and 4605496 tetrahedrons, full NSE, 4-th order explicit time integration

2. "Round jet and cylinder", mesh with 2047992 nodes and 12066956 tetrahedrons, full NSE, 4-th order explicit time integration

These cases are related with studying of acoustic sources in a turbulent wake. Speedup results are shown in figure 3. It can be seen that MPI+OpenMP only outperforms MPI at rather big numbers of CPUs. The point of intersection of the plots for MPI and MPI+OpenMP moves to the bigger number of CPUs when the mesh size increases. In the first case (fig 3 left) it is around 250 CPU while in the second case (fig 3 right) that has 2.5 times bigger mesh it is above 500 CPUs.

The major time consuming components of the algorithm are represented and compared in table 1 for 4-step explicit Runge-Kutta time integration. The wall-clock time was measured for 30 time steps when running the case 1 on 1280 CPUs of Lomonosov supercomputer.



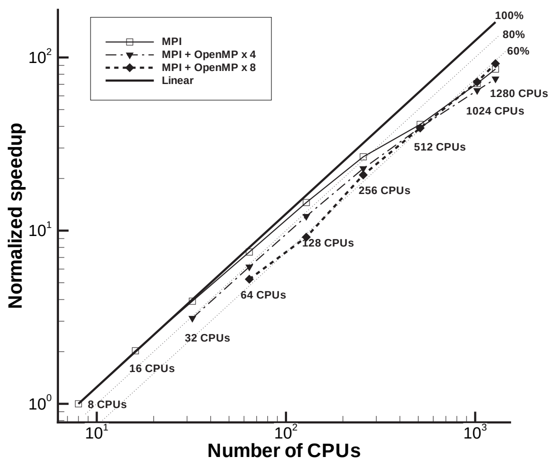


Figure 3: OpenMP speedup of the Poisson solver (left) and overall CFD algorithm (right) on MVS-100000 (log. scale)

Name of channel	N of calls	t, sec. MPI	t, sec. MPI+OpenMP
Reduction communication (max)	30	0.2252	0.1347
Calculation of gradients	120	1.1647	1.3875
Calculation of fluxes	120	1.5750	1.1740
P2P communication	120	1.3333	0.6086

Table 1: Comparison of major time consuming components of the algorithm when running on 1280 CPUs

It must be noted that by the time the tests were performed the second level decomposition for OpenMP was not yet fully implemented and calculation of gradients was done with a substantial overlap (which later has been substantially reduced). Results in table 1 also show that the time spent on communications is substantially reduced in case of MPI+OpeMP. Finally MPI+OpenMP outperformed MPI in total around 16% for case 1 and 8% for case 2 when running on 1280 CPUs.

4 Illustrative applications

The described algorithms with hybrid parallelization are currently used to perform several DNS cases. In particular:

1. "Impinging jet" DNS case, incompressible flow, mesh 100 millions of nodes, 4-th order scheme, Re = 20000, running on 320 CPUs of MareNostrum supercomputer
2. "Round jet and cylinder", compressible flow, Mach 0.1, Re = 14000, mesh sizes 16 and 64 millions of nodes, full NSE, high-order scheme

Some illustrations of instantaneous flow fields are given in figure 4.

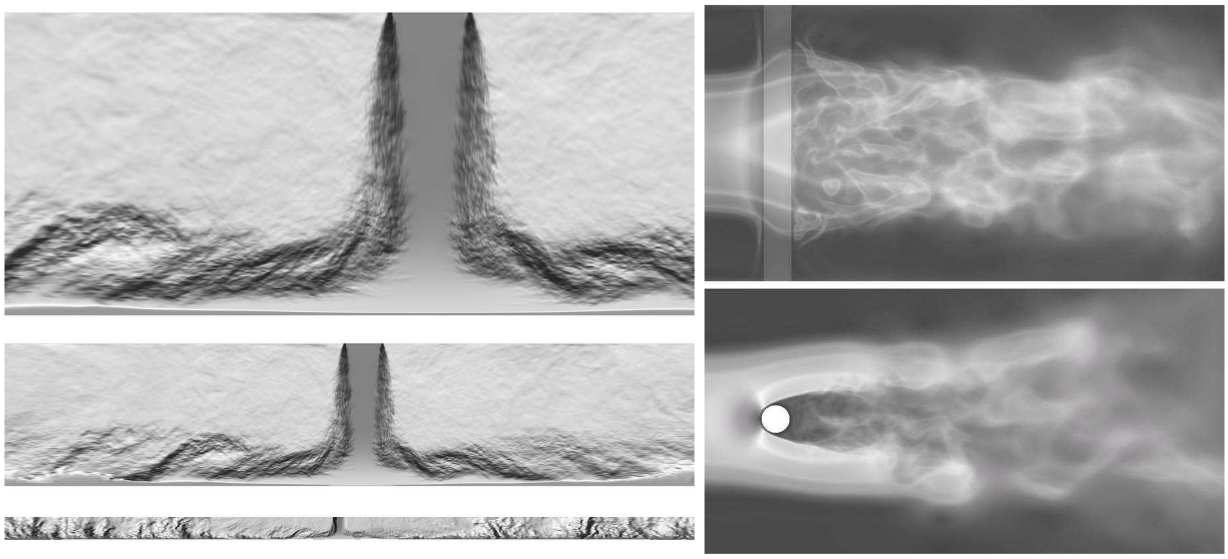


Figure 4: "Impinging jet" DNS case (left) is shown in three fragments: the whole domain (bottom) and zoom to the jet region (middle and top). "Round jet and cylinder" (right): top view (top) and lateral view (bottom). This visualisation of instantaneous flow fields is based on velocity module.

5 Conclusions

The use of OpenMP in addition to MPI for CFD applications has been demonstrated. A parallel Poisson solver for incompressible flows with one periodic direction has been adapted to modern supercomputer architecture with multi-core nodes by hybrid parallelization that will allow to use the solver efficiently on around 10000 CPUs. A parallel CFD/CAA algorithm for compressible flows has been upgraded with the two-level parallelization using OpenMP on the second level. In both cases implementation of additional OpenMP parallelization appeared to be profitable considering the extension in efficient CPU number range it provided and the relatively small effort it required. It must be noted that OpenMP is not the only choice for the second-level parallelization. It could also be done with POSIX Threads or with MPI again, but OpenMP was chosen for the sake of simplicity and portability.

6 Acknowledgements

This work has been financially supported by the *Ministerio de Ciencia e Innovación*, Spain; Contract/grant numbers ENE2007-67185 (Project: "Development of high performance parallel codes for the optimal design of thermal equipments")

This work has been financially supported by the Russian foundation for basic research, grant numbers 09-01-12022-ofi_m and 09-01-00833-a.

Our calculations have been performed on the MVS-100000 supercomputer at the Joint Supercomputer Center of the Russian Academy of Sciences, Lomonosov supercomputer at the Moscow State University, IBM MareNostrum supercomputer at the Barcelona

Supercomputing Center. We thank all these institutions.

References

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

[2] R.Borrell, O.Lehmkuhl, F.X.Trias, M.Soria, A.Oliva, Parallel direct Poisson solver for DNS of complex turbulent flows using Unstructured Meshes, *Parallel CFD 2008*, Lyon (France), May 2008

[3] M. Soria and C. D. Pérez-Segarra and A. Oliva, A Direct Parallel Algorithm for the Efficient Solution of the Pressure-Correction Equation of Incompressible Flow Problems Using Loosely Coupled Computers, *Numerical Heat Transfer, Part B*, 41 (2002) 117-138

[4] A .V. Gorobets, T.K. Kozubskaya, S.A. Soukov, On efficiency of supercomputers in CFD simulations, *Parallel CFD 2008*, Lyon (France), May 2008

[5] Yousef Saad, Iterative Methods for Sparse Linear Systems, *PWS*, Boston, 1996

[6] R. W. C. P. Verstappen and A. E. P. Veldman. Symmetry-Preserving Discretization of Turbulent Flow. *Journal of Computational Physics*, 187:343–368, 2003.

[7] F. X. Trias, M. Soria, C. D. Pérez-Segarra, and A. Oliva. A Direct Schur-Fourier Decomposition for the Efficient Solution of High-Order Poisson Equations on Loosely Coupled Parallel Computers. *Numerical Linear Algebra with Applications*, 13:303–326, 2006.

[8] F. X. Trias, M. Soria, A. Oliva, and C. D. Pérez-Segarra. Direct numerical simulations of two- and three-dimensional turbulent natural convection flows in a differentially heated cavity of aspect ratio 4. *Journal of Fluid Mechanics*, 586:259–293, 2007.

[9] 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, Int. J. Heat and Mass Transfer, 53 (2010) 665-673, Elsevier

[10] 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 II: Heat transfer and flow dynamics, Int. J. Heat and Mass Transfer, 53 (2010) 674-683, Elsevier

[11] T.Kozubskaya A.Gorobets. Technology of parallelization of explicit high-accuracy algorithms on unstructured meshes in computational fluid dynamics and aeroacoustics. *Matematicheskoe modelirovanie*, 19 (2):68–86, 2007.

[12] Tatiana Kozubskaya Ilya Abalakin, Alain Dervieux. High Accuracy Finite Volume Method for Solving Nonlinear Aeroacoustics Problems on Unstructured Meshes. *Chinese Journal of Aeroanautics*, pages 97–104, 2006.

[13] C. Debiez and A. Dervieux, Mixed element volume MUSCL methods with weak viscosity for steady and unsteady flow calculation, Computer and Fluids, Vol. 29, 1999, pp. 89-118.

[14] A.V. Gorobets, T.K. Kozubskaya, S.A. Soukov, On efficiency of supercomputers in CFD simulations, Parallel CFD 2008, Lyon (France), May 2008