C. R. Mecanique 339 (2011) 141–148

Contents lists available at ScienceDirect

Comptes Rendus Mecanique

www.sciencedirect.com

High Performance Computing / Le Calcul Intensif

Design of a massively parallel CFD code for complex geometries

Une algorithmique optimisée pour le supercalcul appliqué à la mécanique des fluides numérique

Vincent Moureau [*], Pascale Domingo, Luc Vervisch

CORIA, CNRS, INSA & Université de Rouen, 76801 Saint-Etienne-du-Rouvray, France

A R T I C L E   I N F O

Article history:
Available online 30 December 2010

Keywords:
Computer science
Fluid dynamic solvers
Domain decomposition

Mots-clés :
Informatique, algorithmique
Mécanique des fluides numérique
Décomposition de domaine

A B S T R A C T

A strategy to build the next generation of fluid dynamics solvers able to fully benefit from high-performance computing is discussed. The procedure relies on a domain decomposition of unstructured meshes that is organized in two levels. The computing cells are first gathered at an elementary level in cell groups; at a second level, cell groups are dispatched over processors. Compared to the usual single-level domain decomposition, this double domain decomposition allows for easily optimizing the use of processor memory and therefore load balancing in both Eulerian and Lagrangian contexts. Specific communication procedures to handle faces, edges and nodes are associated to this double domain decomposition, which strongly reduce the computing cost; input–output times are optimized as well. In addition, any multi-level solution techniques, as deflated preconditioned conjugate gradient, are well-adapted to such mesh decomposition. This approach has been used to develop the YALES2 code, which also benefits from a non-degenerescent tessellation algorithm for tetrahedra to automatically generate high-resolution meshes on super-computers. To illustrate the capabilities of the YALES2 algorithmic, an aeronautical burner is fully simulated with a mesh of 2.6 billion cells, followed by a demonstration test over 21 billion cells.

R É S U M É

Une décomposition de domaine originale et les structures de données associées sont proposées pour la simulation numérique appliquée aux équations de la mécanique des fluides, afin de tirer pleinement parti de la puissance des supercalculateurs hautement parallèles. La stratégie adoptée dans le logiciel YALES2 repose sur une double décomposition: un niveau élémentaire de groupes de cellules est formé, avant de répartir ces groupes de cellules sur les processeurs à un deuxième niveau. Cette décomposition optimise sur le supercalculateur la répartition des données des maillages non-structurés, pour des simulations eulériennes aussi bien que lagrangiennes. La double décomposition est aussi associée à une procédure de communication spécifique pour la gestion des informations géométriques (faces, arrêtes, nœuds). Ceci se traduit par une gestion optimale du temps calcul et des entrées-sorties, allant de pair avec l'intégration d'un raffinement de maillage systématique. De plus, la double décomposition est particulièrement adaptée aux méthodes de déflation pré-conditionnée avec gradient conjugué. Des calculs démontrant l'efficacité de

[*] Corresponding author.
E-mail address: vincent.moureau@coria.fr (V. Moureau).

1631-0721/$ – see front matter © 2010 Académie des sciences. Published by Elsevier Masson SAS. All rights reserved.
doi:10.1016/j.crme.2010.12.001

cette approche sont présentés, dont une solution convergée sur un maillage contenant 2,6 milliards de cellules et un cas test sur 21 milliards de cellules.

1. Motivation

The evolution of multi-core processing units towards many-core units, as well as the dramatic increase of the number of cores in super-computers enable the tackling of new multi-physics problems. However, this great computing power may be utilized only with suitable softwares and numerics that are sometimes highly tied to the super-computer architecture. Consequently, the design of modern massively parallel solvers has to include these constraints from the early beginning of their development. In the field of Computational Fluid Dynamics (CFD), the key issues are often related to the choice of the parallelism paradigm, the generation and management of large meshes, or the data inputs and outputs, when hundreds of gigabytes are involved. In this scope, the combustion modeling team at CORIA has designed a novel CFD solver called YALES2 that brings innovative solutions to all these key issues. This code alleviates several limitations inherent to distributed memory architectures through a fine-grain partitioning of the mesh and solution files used to start or restart a computation, and of the full solution during the computation. As a result, the small amount of memory available to each computing core in actual super-computers is not a limit and very large computations may be performed. In the first part of the paper, the novel algorithms for the mesh partitioning and management are described, and in the second part, a large computation example is briefly presented.

2. Key components for the design of massively parallel CFD solvers for multi-physics problems

2.1. Governing equations and discretization

A large range of Computational Fluid Dynamics applications relies on the incompressible form of the Navier–Stokes equations, for instance, when the coupling between the flow and its acoustics is not the main driving mechanism. With massively parallel computers, the low-Mach number formulation becomes highly challenging, because it involves solving large linear systems. In this paper, the constant density incompressible equations are considered, but many of the presented algorithms are in fact independent of the governing equations and can be applied to fully compressible flows.

For a laminar or turbulent flow at low Mach number with constant density, the momentum equation reads

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \mathbf{uu} = -\frac{1}{\rho}\nabla P + \nabla \cdot \tau \tag{1}$$

where $\mathbf{u}$ is the velocity, P the pressure, ρ the density and τ the shear stress tensor.

For this type of flow, the mass conservation constraint becomes

$$\nabla \cdot \mathbf{u} = 0 \tag{2}$$

The system of Eqs. (1) and (2), is closed; however, it is useful to derive an equation for P by taking the divergence of Eq. (1), and if the viscous forces are neglected, one obtains the well-known Poisson equation

$$\nabla \cdot \left(\frac{1}{\rho}\nabla P\right) = 2Q \tag{3}$$

where Q defines the so-called Q-criterion [1]. This criterion is the second invariant of the deformation tensor and it is often used for the visualization of vortices in turbulent flows. This criterion may be expressed as the difference between the norms of the rotation and shear tensors Ω_{ij} and S_{ij}:

$$Q = \frac{1}{2}(\Omega_{ij}\Omega_{ij} - S_{ij}S_{ij}) \tag{4}$$

where

$$S_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right), \qquad \Omega_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_i}\right) \tag{5}$$

High Performance Computing (HPC) plays a role of primary importance in CFD because turbulence modeling involves a very broad spectrum of time and space scales. The resolution requirements may be quantified [2] by the number of cells NX needed in each direction to simulate a turbulent flow with a Reynolds number Re_T:

$$NX > Re_T^{3/4} \tag{6}$$

In typical ignition-controlled engines or gas turbines, the turbulent Reynolds number Re_T is of the order of several thousands, which leads to 3D meshes with billions of cells. Consequently, the modeling of turbulent flows with such meshes

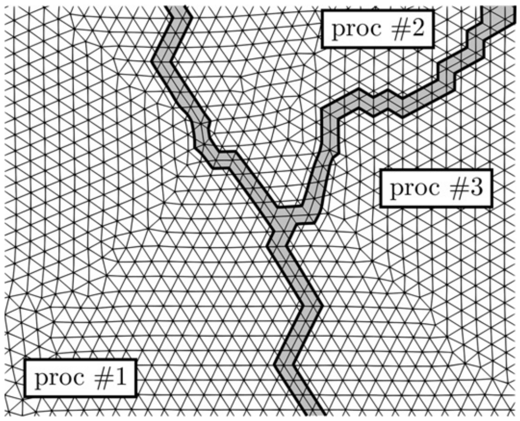

Fig. 1. Single Domain Decomposition. The highlighted elements are participating in the communications between processors.

imposes the use of a few to several thousand cores. Moreover, the nature of Eqs. (1) and (3), which are hyperbolic and elliptic respectively, leads to frequent communications between the cores to advance the flow field in time. All these constraints have a direct impact on the performances of CFD solvers.

2.2. Domain decomposition

Several paradigms exist to balance the computational work required by CFD on several cores. In the OpenMP framework, for instance, algorithm loops are divided into independent chunks that are solved by each core before a synchronization point at the end of the loop. This method does not guarantee a linear speed-up on a large number of cores, because all the loops of a CFD solver are not suitable to an OpenMP splitting. Thus, domain decomposition techniques relying on the Message Passing Interface (MPI) are usually preferred for massively parallel solvers. It consists in splitting the computational domain into sub-meshes that are affected to each computational core. Then, the governing equations are solved on each subdomain and communications are performed whenever solved quantities stored in other cores are required. Since the preparation and the receiving of the MPI messages induce some latency, MPI and OpenMP may be coupled to increase the efficiency on multi-core CPUs. In that case, MPI is used for inter-processor communications and OpenMP for intra-processor communications.

2.3. Single Domain Decomposition

In Single Domain Decomposition (SDD) methods, the domain is split in a number of subdomains that corresponds to the number of cores as illustrated in Fig. 1. On each core, all the three-dimensional variables required for the CFD computation are stored in the local memory and some connectivity is necessary to compute the partial derivatives of the governing equations. This approach has some drawbacks because the processing of any variable or any connectivity change will have to deal with the full subdomain mesh, which may count 10,000 to more than one million cells. This strategy does not benefit from any co-location of the data in the memory space. For instance, a three-dimensional array is stored as a one-dimensional array in the memory and the computation of any differential operator involving the three space directions has then to retrieve the data in scattered places in the memory. This introduces some latency along with cache misses during the computation. Another issue concerns mesh connectivity changes that may occur during local mesh refinement or load balancing. In such cases SDD requires to modify all the connectivity of the subdomain, which is costly. These examples show that an additional decomposition level may prove useful to alleviate most of these issues.

2.4. Double Domain Decomposition

In the Double Domain Decomposition (DDD) methodology, an additional level of subdomains is introduced, meaning that the subdomain of each computational core is divided into cell groups, as sketched in Fig. 2. The primary goal of the technique is to localize the data in the memory space to avoid any cache misses. Indeed, if all the data of a cell group fit in the cache memory of the core, all the operations may be performed without unloading and reloading any data in the cache memory.

This approach has also several side benefits when dealing with mesh connectivity changes. First, load balancing between processing units becomes simpler to perform. Instead of balancing the number of cells on all the processor, the DDD technique allows to balance the number of cell groups per core. Then, the cell groups are simply transferred from one core to another and only a limited part of the connectivity has to be rebuilt. Moreover, if Lagrangian particles are used in the CFD computation and are stored in the cell groups, the load balancing of the particles and the Eulerian cells is performed altogether without any additional effort compared to the single phase case.

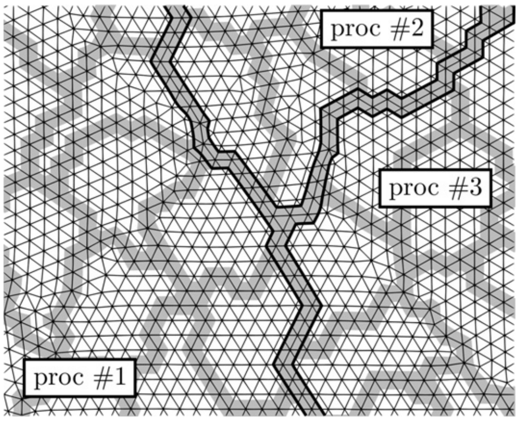

Fig. 2. Double Domain Decomposition. The highlighted elements are participating in the communications inside and outside each processor and those in the black subdomain are participating in the communications between processors.

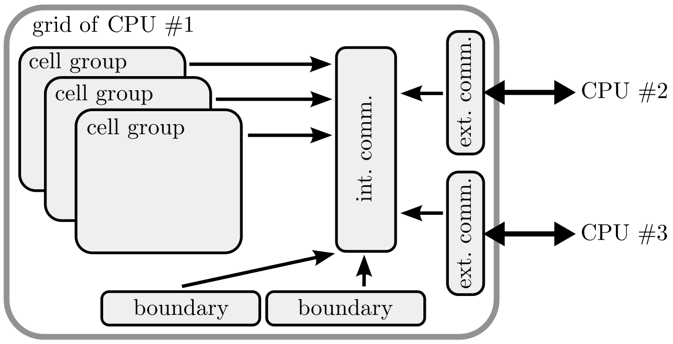

Fig. 3. Internal and external communicators.

A second important feature of the DDD technique is to embed a coarse mesh of the domain. If the cell groups all count 500 cells in three dimensions, a mesh with one million cells features around 2000 cell groups. This coarse mesh may be used for the preconditioning of the Poisson equation (3). For instance, deflated preconditioned conjugate gradient (DPCG) algorithms [3] rely on this coarse mesh in a very efficient manner.

Performing the communications in SDD solvers requires to build sorted lists of geometrical elements that have to be exchanged between different cores. In DDD solvers, some geometrical elements, such as nodes, faces or edges, need to exchange data inside the core during the communication steps. Another data structure is therefore needed to connect the geometrical elements at the border of the cell groups together. The solution adopted in the YALES2 code is to define an internal communicator that contains all the nodes, faces or edges involved in the communications inside or outside the cores, and external communicators that link the nodes, faces or edges of the internal communicator with those contained in other computing cores. This architecture is depicted in Fig. 3, where the boundaries are also represented. As a result, the assembling of a residual consists of the following steps:

1. Compute the residual in each cell group.
2. Update the internal communicator values with the cell group contributions.
3. Add the contributions of the boundaries to the internal communicator values.
4. Update the external communicator values with the values from the internal communicator.
5. Exchange the values between external communicators.
6. Update the internal communicator values with the contributions of the external communicators.
7. Update the residual values in each cell group.

In the YALES2 solver, the partitioning at each level is performed thanks to the METIS library [4]. This library is a set of tools to generate high-quality partitions of undirected graphs with a minimum edge cut.

2.5. Mesh generation and IO handling

The mesh generation becomes an issue when dealing with problems that require meshes of more than 100 million elements because very few meshing softwares are able to cope with such mesh sizes. Solving this issue is of primary

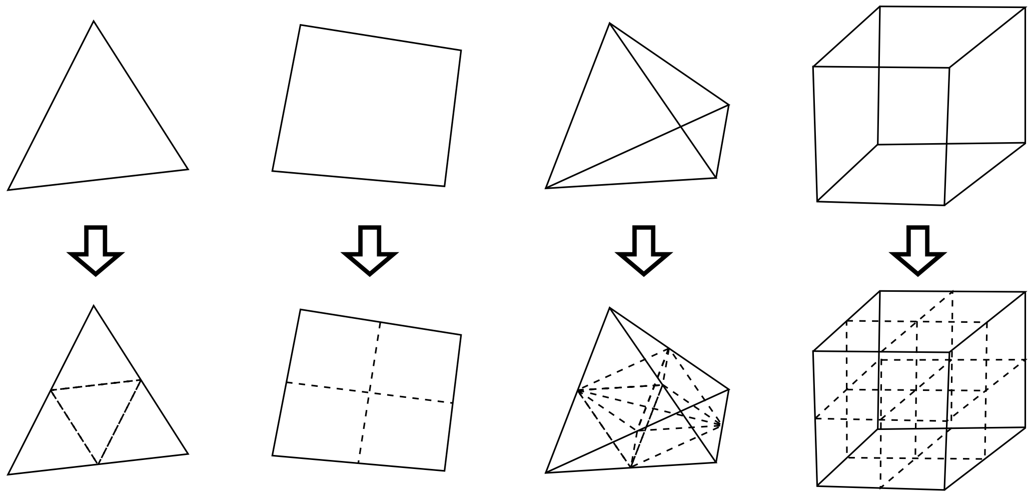

Fig. 4. Refinement of elements.

importance because most of the super-computers have a large memory capacity suitable for meshes of several billion elements. Various methodologies address the mesh generation issue. For instance, the mesh may be generated as a collection of small meshes that have to be reconnected at the beginning of the computation. This method may require some efforts especially for the reconnection of the meshes. Thus, the strategy in YALES2 is to generate meshes sufficiently resolved to describe the geometry with around 50 million elements and then to automatically refine them with tessellation algorithms. This method is illustrated in Fig. 4 for 2D and 3D basic elements. The tessellation algorithm for triangles, quadrangles and hexahedra, which consists in splitting all the edges in their middle, is non-degenerescent, which means that the quality of the mesh remains the same after successive mesh refinement steps. However, this is not the case for tetrahedra if applied without caution. Indeed, the tessellation of a tetrahedron into eight subelements is not uniquely defined. The correct choice of the four subelements at the center of the tetrahedron is essential to the quality of the mesh. For this reason, a different non-degenerescent tessellation algorithm [5] is used for tetrahedra, which ensures that the skewness of the mesh remains the same after successive mesh refinement steps.

The only drawback of the mesh refinement methodology concerns the boundaries featuring an important curvature. Indeed, the refined meshes do not modify the original mesh description of the boundary. If this original description fits imperfectly the region of strong curvature, the error will be transmitted to the successive refined meshes. For this reason, it is advised to limit the number of mesh refinement steps and to have a starting mesh that already describes correctly all the geometrical details. An alternative could be to move slightly the created nodes to match the local boundary curvature but such algorithms still have to be implemented in the YALES2 software.

Once the mesh is generated, it has to be stored and then it has to be loaded by each computational core at the beginning of the computation. The mesh cannot be stored in a single file: for instance, a 3D mesh of three billion tetrahedra plus a solution, which contains the velocity, the pressure and other important fields, has a typical size of hundred gigabytes. If the mesh is stored in several files, the number of files should not be dependent on the number of cores of the computation because it would limit a lot the flexibility of the code. It would also lead to a potential issue on machines with a large number of cores and a file system with a limit on the number of files per directory. Therefore, the mesh management has to rely on a partitioning of the grid with just enough files to avoid any memory issue when loading the mesh. With YALES2, each file of a partitioned mesh contains typically less than a million elements. A 2.6 billion cells mesh for instance is usually split into 4096 files. Then, all the inputs and outputs (IO) are managed by 4096 master processors that are in charge of reading and writing the mesh and the solution. No parallel IO are used in YALES2, but the master processors are sorted and prioritized so that only a certain number of them can read or write at the same time to avoid any saturation of the disk accesses.

The file format used in YALES2 is based on the HDF5 library [6] because it brings a binary format with compression, which is platform independent, and it has a convenient API. Moreover, this file format may be read directly in some visualization softwares through the eXtensible Data Model and Format (XDMF) standard [7] even if the mesh and the solution are partitioned in many blocks.

When combined with suitable interpolation algorithms, these tessellation and partitioning algorithms allow to perform mesh refinement and weak scaling studies easily. An example of weak scaling measurement conducted on an IBM Blue Gene/P machine is presented in Fig. 5. In these measurements, only the solving of the Navier–Stokes equations, i.e. the velocity prediction and projection steps, are taken into account. The mesh refinement algorithm has a negligible cost compared to the flow advancement. Two original meshes with 14 and 41 million cells respectively were generated and all the other meshes were obtained through successive mesh refinements. When increasing the number of cores, the size of the

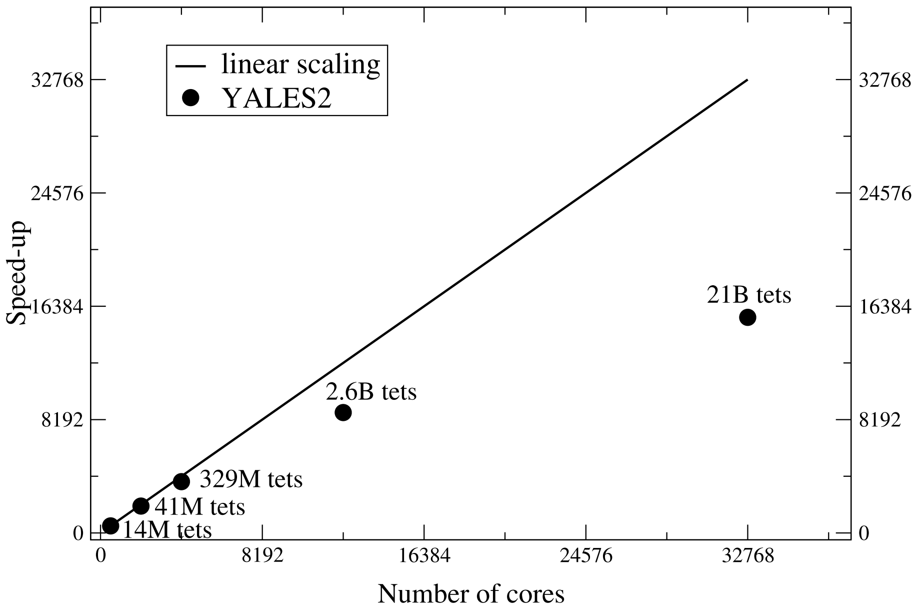

Fig. 5. Weak scaling measurements on an IBM Blue Gene/P machine.

mesh was also increased to keep a reasonable cell count on each core. It may be noticed that the performances of the code are rather good, except for the biggest mesh that features 21 billion tetrahedra. For this mesh, the performances of the Deflated Preconditioned Conjugate Gradient (DPCG) [3] utilized to solve the Poisson equation are not sufficient. This is mostly due to the fact that only one coarse mesh is used to precondition the PCG and more levels are necessary on such large meshes.

3. Application to a semi-industrial swirl burner

Combining the different techniques described in the paper, large computations may be performed efficiently on a large number of cores, typically greater than 16,384. To demonstrate these capabilities, the iso-thermal simulation of a semi-industrial swirl burner was conducted without any chemical reactions, just in order to analyze the flow and the turbulence intensity without thermal expansion. This swirl burner was designed by Turbomeca, SAFRAN group, and studied experimentally at DLR by Meier et al. [8]. The very complete experimental database consists of Particle Image Velocimetry (PIV) velocity measurement for the iso-thermal flow and Raman species and temperature measurements for the reacting case. Consequently, this database was extensively used as a validation test case for combustion models or for Large-Eddy Simulation (LES) solvers [9–12]. The simulated operating conditions are at atmospheric pressure and temperature with a mass flow rate of 13.6 g/s. The flow Reynolds number in this case is approximately 40,000 at the swirler exit based on the injection diameter and an estimation of the Kolmogorov scale based on turbulent dissipation measurement in coarse LES computations gives $\eta = 29$ μm, away from the walls, with Re_T of the order of 1400. From this estimate, a resolution of at least 100 μm is necessary to resolve all the turbulent scales in the flow except in the boundary layers. Such a mesh with 2.6 billion tetrahedra was generated from an original grid featuring 41 million tetrahedra, which was refined homogeneously twice. LES computations were first performed and converged on intermediate refined grids before the final interpolation on the 2.6 billion cells mesh. Then, the flow was converged again to obtain a fully developed flow. Sample results are presented in Figs. 6 and 7. In the first figure, the instantaneous velocity magnitude shows the typical V-shape of swirl injectors and the broad range of turbulent scales. The second figure represents the smallest resolved vortices through the plot of Q-criterion iso-contours [1]. It may be noticed that vortices are concentrated in the swirl region and that this swirl motion promotes the formation of elongated vortices in the shear regions. These computations illustrate the possible use of the methodologies presented in the paper. More results may be found in dedicated publications [13].

4. Conclusions

In this article, the design of modern CFD solvers is discussed with a particular attention paid to mesh management and partitioning. This focus is motivated by the continuous increase of computing power of massively parallel computers. To fully benefit from these new computing architectures, specifically tailored algorithms have to be used. For instance, load balancing techniques and linear solvers preconditioned with a coarse grid can be implemented easily, specifically when relying on a

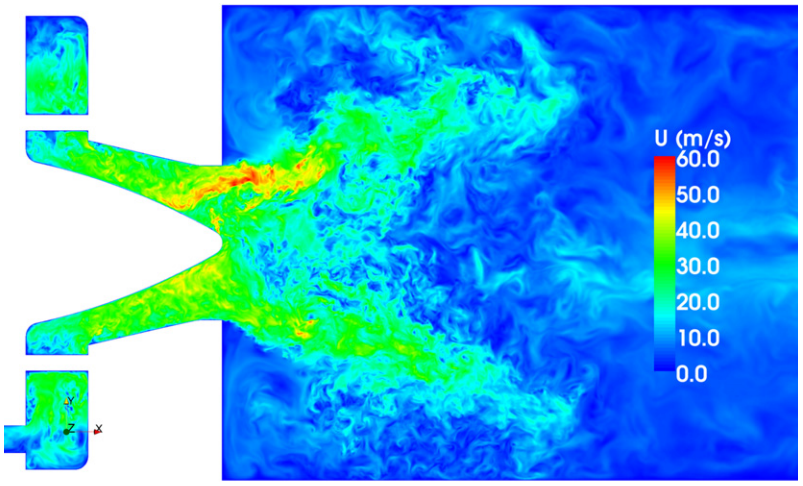

Fig. 6. Instantaneous velocity field.

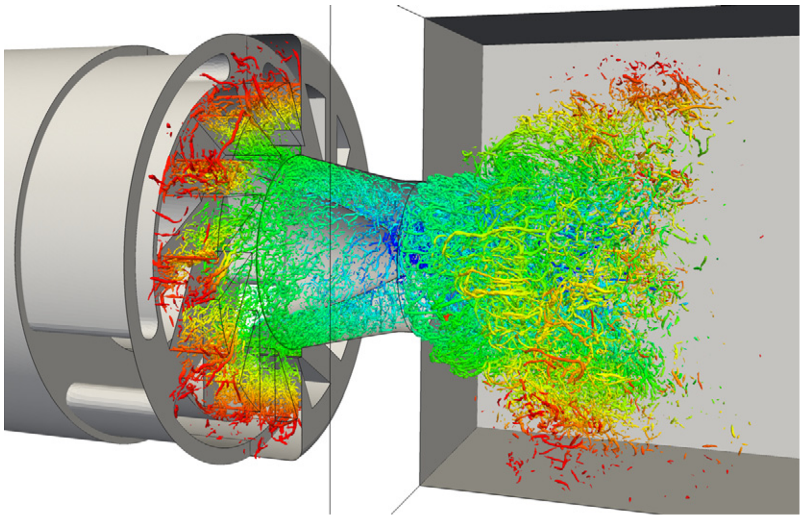

Fig. 7. Q-criterion iso-contours representing the smallest resolved vortices.

double domain decomposition. This type of fine-grain partitioning may also be applied to the mesh and solution files used to restart or visualize computations. When these very large files are split into a manageable number of blocks, the small amount of memory per CPU is not a constraint anymore.

Another important point concerns the mesh generation. In the case of unstructured grids, that are well-adapted to complex geometries, the generation of very large meshes with several billion cells is challenging. To overcome this issue, mesh refinement and interpolation algorithms may be utilized. All these methodologies are coded in the YALES2 solver, which is able to perform very large computations of iso-thermal and reacting turbulent flows. The example given in the paper highlights the great benefit of an increased mesh resolution. In many lab-scale experiments, such a resolution is sufficient to reach the Direct Numerical Simulation (DNS).

This type of computations opens new perspectives in fluid dynamic research. First, by simulating the finest scales in realistic flow Reynolds numbers, it revitalizes the use of DNS databases to understand the physics and thereby improving the accuracy of physical models for sub-grid scale non-linear phenomena, models that are mandatory to perform coarser simulations for everyday design optimization in industry. Second, improving resolution brings out the details of complex multi-physics flow problems, as for instance primary liquid atomization or the coupling between chemistry and radiative heat transfer, including detailed radiative spectral-response of gases. Third, a flow solution strategy ables to handle billions of points within an acceptable cpu time, means a code that can return unsteady flow solutions over meshes of moderate size in less than a day on affordable computers; hence, generalizing the inclusion of major unsteadiness induced effects in computational fluid dynamics of real systems.

Acknowledgements

The authors were granted access to IDRIS HPC resources under the allocation 2010-090052 made by GENCI (Grand Equipement National de Calcul Intensif) and to Juelich Supercomputing Centre resources during the 2010 Blue Gene Extreme Scaling workshop.

References

[1] Y. Dubief, F. Delcayre, On coherent-vortex interaction identification in turbulence, J. Turbulence 1 (2000) 1–22.
[2] T. Poinsot, D. Veynante, Theoretical and Numerical Combustion, R.T. Edwards, Inc., Philadelphia, 2001.
[3] R. Nicolaides, Deflation of conjugate gradients with applications to boundary value problems, SIAM J. Numer. Anal. 24 (2) (1987) 355–365.
[4] G. Karypis, V. Kumar, A fast and high quality multilevel scheme for partitioning irregular graphs, SIAM J. Sci. Comput. 20 (1) (1999) 359–392.
[5] M.-C. Rivara, Mesh refinement processes based on the generalized bisection of simplices, SIAM J. Numer. Anal. 21 (3) (1984) 604–613.
[6] The HDF Group, Hierarchical data format version 5, http://www.hdfgroup.org/hdf5, 2000–2010.
[7] The XDMF Group, Extensible data model and format, http://www.xdmf.org.
[8] W. Meier, P. Weigand, X. Duan, R. Giezendanner-Thoben, Detailed characterization of the dynamics of thermoacoustic pulsations in a lean premixed swirl flame, Combust. Flame 150 (1–2) (2007) 2–26.
[9] V. Moureau, C. Bérat, H. Pitsch, An efficient semi-implicit compressible solver for large-eddy simulations, J. Comput. Phys. 226 (2) (2007) 1256–1270.
[10] V. Moureau, P. Minot, H. Pitsch, C. Bérat, A ghost-fluid method for large-eddy simulations of premixed combustion in complex geometries, J. Comput. Phys. 221 (2) (2007) 600–614.
[11] J. Galpin, A. Naudin, L. Vervisch, C. Angelberger, O. Colin, P. Domingo, Large-eddy simulation of a fuel-lean premixed turbulent swirl-burner, Combust. Flame 155 (1–2) (2008) 247–266.
[12] S. Roux, G. Lartigue, T. Poinsot, U. Meier, C. Berat, Studies of mean and unsteady flow in a swirled combustor using experiments, acoustic analysis, and large eddy simulations, Combust. Flame 141 (1–2) (2005) 40–54.
[13] V. Moureau, P. Domingo, L. Vervisch, From large-eddy simulation to direct numerical simulation of a lean premixed swirl flame: Filtered laminar flame-pdf modeling, Combust. Flame (2010), doi:10.1016/j.combustflame.2010.12.004.