Unsteady Flow Simulations in Compressors

High Performance Computing to Simulate Large Scale Industrial Flows in Multistage Compressors

*N. Gourdain¹, M. Montagnac¹, F. Wlassow¹, M. Gazaix²*

¹Computational Fluid Dynamics Team, CERFACS, Toulouse, 31057, France
²Computational Fluid Dynamics and Aero-acoustics Dpt., ONERA, Châtillon, 92320, France
Unsteady Flow Simulations in Compressors

Corresponding author: Dr. Nicolas Gourdain, Nicolas.Gourdain@cerfacs.fr
CERFACS, CFD Team, 42 avenue Gaspard Coriolis, Toulouse, 31057, France
Phone: +33 5 61 19 30 47
Fax: +33 5 61 19 30 00

Co-author: Dr. Marc Montagnac, Marc.Montagnac@cerfacs.fr
CERFACS, CFD Team, 42 avenue Gaspard Coriolis, Toulouse, 31057, France
Phone: +33 5 61 19 30 48
Fax: +33 5 61 19 30 00

Co-author: Mr. Fabien Wlassow, Fabien.Wlassow@cerfacs.fr
CERFACS, CFD Team, 42 avenue Gaspard Coriolis, Toulouse, 31057, France
Phone: +33 5 61 19 30 06
Fax: +33 5 61 19 30 00

Co-author: Mr. Michel Gazaix, Michel.Gazaix@onera.fr
ONERA, DSNA Dpt., avenue de la Division Leclerc, BP72, Châtillon, 92322, France
Phone: +33 1 46 73 42 56
Fax: +33 1 46 73 41 66
Unsteady Flow Simulations in Compressors

ABSTRACT
The aim of this study is to propose a computing method to obtain a detailed simulation of the unsteady flow that develops in multistage turbomachines. The three-dimensional unsteady Reynolds-Averaged Navier-Stokes equations are solved using a structured multiblock decomposition method. Although this kind of flow solver is very popular in the turbomachine community nowadays, the complex block connectivities used in meshes of industrial configurations can be penalizing for parallel computing. The computing strategy implemented in a modern flow solver is investigated in this paper, with a particular interest for mesh partitioning, communications and load balancing. Advantages and drawbacks of different computing platforms are then discussed, ranging from vector supercomputers to massively scalar platforms. Comparisons are performed regarding criteria such as the elapsed time and the electric power consumption. The results show that the use of a large number of computing cores (>128) is largely penalized by communications and load balancing errors, whereas computing performance with a moderate number of computing cores (<128) is mainly driven by the peak power of the architecture. To help users estimate a priori the parallel performance of a task, a tool based on an extension of the Amdahl’s law is proposed, showing satisfying results when compared with observations. Finally, an unsteady flow simulation is performed in a whole three-stage compressor at the design operating point. While still far beyond industrial resources, this numerical flow simulation shows that potential breakthroughs in the design of compression systems can be expected.

KEYWORDS: parallel computing, scalar, vector, multistage compressor, unsteady flows

NOMENCLATURE

\[ h \] distance from the hub (m)
Unsteady Flow Simulations in Compressors

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H$</td>
<td>compressor span (m)</td>
</tr>
<tr>
<td>$N$</td>
<td>number of computing cores (-)</td>
</tr>
<tr>
<td>$S$</td>
<td>speedup (-)</td>
</tr>
<tr>
<td>$V_x$</td>
<td>axial velocity (m.s$^{-1}$)</td>
</tr>
<tr>
<td>$W$</td>
<td>vector of conservative variables (-)</td>
</tr>
<tr>
<td>$\phi$</td>
<td>normalized mass flow (-)</td>
</tr>
<tr>
<td>$\psi$</td>
<td>relative error between steady and unsteady flow solutions (-)</td>
</tr>
<tr>
<td>IGV</td>
<td>Inlet Guide Vane</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
</tbody>
</table>

INTRODUCTION

Present regulations in terms of pollutant emissions, noise and economical constraints, require new approaches and design in the fields of energy supply and transport domains. Nowadays, rotating machines such as gas and wind turbines are involved in most the energy conversion processes. Increasing the efficiency of these systems is thus a necessary step to meet industrial requirements in terms of pollutant emissions and economical constraints. Unfortunately, the flows that take place in these machines are still not well understood, especially in complex environment such as multistage turbomachines. This lack of knowledge leads to a challenge to design the next engine generation with a more compact size and a larger operability range. In the case of compressors, it is well established that endwall flows and multistage interaction between adjacent blade rows largely affect the overall performance [1]. Nowadays different approaches are available to obtain valuable data for analyzing these complex unsteady flow phenomena. The first method consists in performing experimental campaigns, which is the historic way to provide reliable data and validate the design of a system. But this approach induces high financial costs and long times in the design process.
Unsteady Flow Simulations in Compressors

Another more recent and popular method is the Computational Fluid Dynamics (CFD) that allows many investigation possibilities for more reasonable cost and time. The main drawback of the latter is still its limited predictive capacity for very complex flows due to deficiencies in the physical or numerical modelling. While it is known that the next technological breakthrough will come from a better understanding of the unsteady flow effects, and by considering the whole system, most flow simulations still focus only on a limited part of the machine (e.g. an isolated blade or a periodic sector) in which the flow is solved with a steady-state method (e.g. the Reynolds-Averaged Navier-Stokes -RANS- method). The main reason is that the size and the complexity of rotating machines impose serious constraints on the simulation cost. Therefore, an improvement of the machine design requires that flow solvers efficiently run on high-end computing platforms to get high levels of detail and accuracy.

Two main types of computer architecture are generally distinguished and compared. The first kind is the vector-type architecture that has large memory bandwidth and vector processing capabilities. The second type of architecture relies on superscalar cache-based processors interconnected to form systems of symmetric multi-processing nodes. The latter has rapidly developed mainly due to the financial cost effectiveness of its basic components. In general, scientific applications can be tuned specifically to perform well on both platforms. However, the sustained performance of some complex applications is often far from the peak, especially on superscalar machines [2]. Nevertheless, the Gordon Bell Prize for Peak Performance delivered each year at the Supercomputing conference has rewarded applications running on massively parallel scalar platforms these last years, succeeding a period dominated by vector supercomputers [3]. It is thus interesting to evaluate both kinds of platforms for a specific application to determine on which one to invest in for future CPU intensive flow simulations.
Unsteady Flow Simulations in Compressors

In the meantime, attempts have been made on the physical modelling in order to ensure a correct balance between cost and accuracy. Several solutions have been proposed in the literature to reduce the cost of unsteady flow simulations, such as the Adamczyk flow model [4], the phase-lag approach [5] and the spectral methods [6]. While these methods are efficient to simulate periodic unsteady flows in turbomachines, their validity domain is restricted around design operating conditions where frequencies and spatial wavelengths of the flow are known. Unfortunately, such assumptions are no longer true at off-design operating conditions, especially close to the aerodynamic stability limit, which is one of the critical parameter for industrial design. As a consequence, a correct estimation of the entire stable performance map requires a flow simulation in the whole compressor with a full unsteady flow method such as the Large Eddy Simulation [7] method or the unsteady RANS method [8] [9]. Based on an inventive URANS/LES coupling strategy, a high-fidelity simulation of the flow in a whole aircraft gas turbine, including fan, compressor, combustion chamber and turbine, has been achieved by Schluter et al. [10], showing very promising results to understand physics that develop in these complex machines.

Based on this state of the art, this paper describes the computing methodology applied to the flow simulation in a whole three-stage compressor. The experimental facility used as test case for this study is presented in the first section. The flow solver and the numerical model (such as numerical schemes, boundary conditions, meshing strategy, etc.) are then described in a second section. The third section deals with the numerical methodology and the difficulties that have to be addressed to efficiently run on massively parallel platforms. The fourth section presents the flow solver performance on vector and scalar platforms considering different numbers of computing cores (up to 4096). One relevant point is to observe the variation of the elapsed time in relation to the increase
Unsteady Flow Simulations in Compressors

of the number of computing cores. Finally, an efficient strategy in terms of response time and cost is chosen to perform a simulation of the unsteady flow within the multistage compressor. The fifth section briefly summarizes the results and shows a comparison with the standard industrial approach (a steady flow simulation coupled with the mixing plane method [11]).

1- EXPERIMENTAL CONFIGURATION

The testing device considered in this paper is a research multistage compressor dedicated to aero-thermal and aerodynamic studies. This 3½-stage axial compressor named CREATE (Compresseur de Recherche pour l'Etude des effets Aérodynamiques et Technologiques) is representative of high-pressure compressor median-rear blocks of modern turbofan engines. The compressor, the inter-row measurement sections and an overview of the test stand are presented in Fig.1. The number of blades of each rotor and stator is indicated in Table 1. The cylindrical outer casing diameter is 0.52 m and the rotor shaft is driven at the design speed of 11,543 rpm by a 2 MW direct-current drive coupled with a gearbox. At this rotational speed, the first rotor speed is 313 m.s⁻¹ at tip (the corresponding Mach number is 0.92). Indeed the flow is slightly transonic in the first stage and subsonic in the two other ones. More information about measurements can be found in [12] and comparisons of experimental data with numerical simulations are detailed in [13].

2- FLOW SOLVER AND NUMERICAL MODEL

Flow solver description

The flow solver used in this work is the elsA multi-application CFD simulation platform that solves the 3D Navier-Stokes equations using a cell-centered approach on multiblock structured meshes [14]. This solver deals with internal and external aerodynamic flows from the low subsonic to the high supersonic flow regime, considering relative frames of reference and the classical Arbitrary
Unsteady Flow Simulations in Compressors

Lagrangian Eulerian formulation to take into account body deformations [15]. A large panel of RANS turbulence models is implemented ranging from eddy viscosity to full Differential Reynolds Stress models and from low Reynolds models to wall law models. The problem of laminar to turbulent transition modelling can also be addressed, based on the application of criteria allowing the description of Tollmien-Schlichting instabilities, cross-flow instabilities, wall roughness, etc. In order to deal with flows exhibiting large separated regions, the turbulence can be simulated thanks to Detached Eddy Simulations and Large Eddy Simulations. In order to cope with complex geometries, highly flexible advanced techniques of multiblock meshes [16] have been implemented such as totally non-coincident inter-block conditions and Chimera technique for overlapping meshes. Thanks to its efficient Object-Oriented (OO) implementation, the flow solver elsA has the capacity to effectively adapt and integrate new capabilities. It offers an access to a powerful combination of CFD abilities and run with a good CPU efficiency on a large panel of computers. Several programming languages are used: C++ as the main language for implementing the OO design, FORTRAN for the CPU efficiency of calculation loops and Python for the user’s interface. A large effort is currently under progress to improve the code performance on massively parallel platforms since the code was originally written for vector machines.

**Numerical parameters**

The target application is a challenging large-scale flow simulation. The choice of the numerical parameters is thus a key point to obtain a fair balance between the cost and quality of results. Spatial convective fluxes are computed with the Roe’s scheme [17] using a fine-tuned Harten’s entropic correction. The van Leer’s Monotone Upstream-centered Scheme for the Conservation Laws (MUSCL) approach [18] is used to reach third-order accuracy in space. A second order Dual Time Stepping (DTS) method [19] is applied for time integration of the unsteady RANS equations.
Unsteady Flow Simulations in Compressors

Based on the criterion proposed by Gourdain and Leboeuf [20], 3200 physical iterations are considered to discretize one rotation of the compressor at the design speed and ensure a correct capture of the main frequencies. The time marching of the inner loop is performed with an efficient implicit time integration scheme based on the backward Euler scheme and a scalar Lower-Upper Symmetric Successive Over-Relaxation (LU-SSOR) method [21]. To achieve a converged state, the number of sub-iterations of the inner loop is defined to decrease the residual magnitude of at least two orders. The conservative values of each block that are near the inter-block connections are exchanged at each relaxation cycle of the implicit method and the linear system solved at each cycle takes into account only one block (i.e. all linear systems are solved simultaneously and independently for each block). The turbulent viscosity is computed with the two equations model of Wilcox [22] based on a k-ω formulation with a fully turbulent flow assumption (the Reynolds number based on the chord is $10^6$). The main advantage of this turbulence model as regards computational aspects is that it belongs to the model category that does not require an explicit knowledge of the wall distance. As a consequence, wall distance computations are avoided at each physical time iteration, resulting in computational time savings.

Meshing strategy and boundary conditions

As a first approximation, the whole length of the experimental domain is not simulated and the present study focuses on the flow inside and close to the compression system (excluding inlet guide vane and plenums). The $2\pi$-periodicity of the compressor is represented, meaning all rotor-stator interaction effects can theoretically be simulated at any operating point, including off-design conditions. The numerical domain extends from section 25A to one axial chord behind section 290 (see Fig. 1 for the definition of experimental sections). The inlet guide vane flow is modelled using the self-similarity law of Lakshminarayana and Davino [23] and experimental data at section 25A.
Unsteady Flow Simulations in Compressors

To avoid numerical problems (such as wave reflexions), inlet and outlet ducts are partially considered: the axial distance between boundary conditions and blade rows corresponds to one rotor chord upstream and two rotor chords downstream the compressor. While tip clearances between rotor blade tip and casing are taken into account, other technological effects such as axial gaps between rotating and non-rotating parts of the hub are neglected due to mesh complexities. However the authors know that these axial gaps have an impact on the compressor overall performance [24] and should be simulated in a future work.

As shown in Fig.2, the flow domain is discretized with a multiblock decomposition approach using an O-H meshing strategy for each passage of the compressor. The typical dimensions of a blade passage mesh are 85, 33 and 57 points, respectively in the axial, tangential and radial directions. An O-H mesh with 13 points in the radial direction is used to discretize the radial tip clearances. To reduce the cost of unsteady flow simulations, a wall function approach is used [25] with a fixed wall cell size that corresponds to a mean normalized wall distance $y^+$ of 20. The whole multistage compressor is represented with a 134M cell grid. It corresponds to a mesh density of 250,000 cells by blade passage, which is a quite-coarse mesh compared to industrial standards (usually around 1M cells for one blade passage). A sliding mesh condition, also called zonal or patched-grid boundary in the literature, is applied at blade row interfaces. This method is based on non-coincident interfaces between blocks [16] and is conservative in the case of plane surfaces, which is roughly the case here. A throttle condition is used to model the outlet duct, coupled with a simplified radial equilibrium condition. By simply increasing the value of the throttle parameter, this condition enables to describe the compressor performance map from choked to stall operating conditions.
3- NUMERICAL METHODOLOGY FOR HIGH PERFORMANCE COMPUTING

The flow solver has been successfully installed on most high-performance computing platforms, including both scalar and vector computers. The message-passing paradigm has been chosen as the way of programming for parallel computers. The Message Passing Interface (MPI) standard library is used to implement communications between computing cores. The parallel approach is based on a coarse-grained approach since each block is allocated to a computing core (several blocks can be allocated to the same core).

**Block connectivity**

The use of structured grids for meshing the configuration leads to a complex multiblock topology. The original configuration is composed of 2,848 blocks and it requires specific treatments to exchange data between adjacent blocks. Two categories of inter-block connections are used in the current context. Either the mesh points are fully coincident at the interface between only two blocks, or the mesh points are totally non coincident at the interface between many blocks (e.g. interface between blade rows). The mostly found case is the coincident block connectivity, which is an interface between two blocks where mesh lines of both blocks remain continuous. In the present flow solver, each block is surrounded by two layers of ghost cells that are used opportunely to set information coming from the adjacent block. If the two blocks are placed on two different computing cores, then a blocking point-to-point message passing communication occurs through the “MPI_Sendrecv_replace” routine. This step requires storing data in buffer arrays to exchange them between computing cores through the MPI library. Otherwise, if the two blocks are placed on the same core, ghost cells are directly filled by a memory-to-memory operation that does not need any MPI communication. Once the ghost cells are filled with the variable values (conservative variables, pressure, etc.), the solver can compute the flow solution of each block independently of
Unsteady Flow Simulations in Compressors

other ones. However, due to computational costs or meshing constraints, this is not always desired to respect the hypothesis of mesh line continuity through an interface between fixed or moving blocks. A non-coincident interface can be considered to stop the spread of a mesh refinement of a block to another in which it would be unnecessary. The other reason to use non-coincident grids is that relative motion between rotating (rotor) and fixed (stator) parts impose that grids move at each physical time iteration and breaks the mesh line continuity at rotor-stator interfaces. The main characteristic of this condition is that a set of blocks is adjacent to another set of blocks. That means one block of the first set of blocks may have to exchange information with each block of the other set during the unsteady flow simulation. To handle message passing between these two sets, a MPI communication group is built with all the blocks related to the interface condition. Collective communications are used to exchange data between computing cores with the “MPI_Allgatherv” routine. If only one computing core handles the whole interblock connection, data exchanges come down to memory-to-memory copies without MPI messages. Note also that ghost cells are not used to compute the flux on the interface. A special routine is dedicated to this task. Depending on the block distribution, the block connectivity may require data communications at some iteration. The two types of communications, that is to say point-to-point communication for coincident interfaces and collective communication for non-coincident interfaces, are serialized. In other words, point-to-point communications take place first and once they are completed, collective communications come in play. Authors are aware that this strategy may be dramatic for speedup and this issue is currently being assessed.

**Estimation of the theoretical speedup**

As the simulation requires several computing cores to compute the flow with a multiblock mesh, it is obviously necessary to allocate blocks to computing cores with the target of maximizing the
Unsteady Flow Simulations in Compressors

speedup $S$, defined as the time ratio between a sequential calculation and a parallel calculation. To estimate this parameter, the most popular method is the Amdahl’s law [26] [27] [28] that gives the maximum theoretical speedup $S_{Amdahl}$ with respect to the number of computing cores $N$:

$$S_{Amdahl} = NP + (1 - P)$$ (1)

where $P$ is the part of the task that can be made parallel and $(1-P)$ the part that remains sequential.

The Amdahl’s law assumes that all tasks attributed to the computing cores finish at the same time ($P$), excepting one core that is dedicated to a sequential task $(1-P)$. However, this assumption is usually not verified since it proves to be difficult to equitably distribute the workload on computing cores, especially with industrial configurations. Indeed, each core ends its own task at a different time. The difficulty thus lies in the estimation of the part of the task that is really parallel in order to apply the Amdahl’s law. To overcome this problem, it is proposed to estimate the theoretical speedup by better taking into account the workload errors. A typical scheme of a parallel computation is shown in Fig. 3: each computing core works a priori for a different time, due to load-balancing errors and communications. Based on this observation, the Amdahl’s law can easily be extended by estimating the working time of each computing core. As a convention, the elapsed time $T_p$ of a parallel computation is set to 1. The sequential time $T_s$ is then obtained by adding the time spent by each core, following the relation:

$$T_{sequential} = NP_0 + (N - 1)P_1 + \ldots + 2P_{N-2} + P_{N-1}$$ (2)

where $P_i$ is the part of the work that is computed with $(N-i)$ computing cores and $P_0+P_1+\ldots+P_{N-1}=T_p=1$.

The strong speedup $S_{EA}$ is directly expressed as the ratio between $T_s$ and $T_p$:
Unsteady Flow Simulations in Compressors

\[ S_{EA} = \sum_{i=0}^{N-1} (N - i)P_i \]  \hspace{1cm} (3)

This law represents an extension of the Amdahl’s law which is simply recovered by taking \( P_1 = P_2 = \ldots = P_{N-2} = 0 \). To apply the extended Amdahl's law (EA) given by Eq. (3), the difficulty relies on the estimation of the coefficients \( P_i \).

**Mesh partitioning: load balancing**

In the case of structured meshes, the size of the blocks could be very heterogeneous, ranging typically from \( 10^3 \) to \( 100^3 \), and taking only into account this criterion alone, it may be very difficult to distribute the same amount of work on all computing cores. As already said in the previous paragraph, this matter can be drawn on a diagram as in Fig. 3. The core with the largest amount of work will end the last one and the other ones will be idle during this period of time. In this paper, the criterion chosen to quantify the amount of work is the total number of mesh cells that belong to blocks handled by the computing core. It can be argued that this criterion is not really perfect since it does not take into account ghost cells that come into play in the flux computation on coincident block-to-block interfaces. They also sometimes take part in the calculation cost as for vector machines some routines are explicitly coded so as to run over all cells of a block in only one “For” loop. Furthermore, this criterion does not take into account some communications. Indeed, some computing cores can be concerned by specific communications through groups of communications. For example, the computation of output data (mass flow rate, etc.) brings only into play a specific pool of computing cores. Non-coincident inter-block connections also concern only a specific pool. The third weak point that deserves to be underlined is the fact that some blocks may have different numerical modelling. The first simple example is that they do not own the same boundary
Unsteady Flow Simulations in Compressors

conditions. Blocks near walls are impacted by turbulence computations whereas blocks at far field conditions are not. Finally, blocks may have different integration schemes or physical modelling depending on locations (typically near walls) or on grid properties (anisotropy, stretching...). According to the criterion used to estimate Amdahl’s (Eq. (1)) and extended Amdahl’s (Eq. (3)) laws, the time spent is supposed to be proportional to the number of cells, meaning that the load balancing error is null only if the same number of cells is allocated to each computing core. Thus the difference with the observed speedup probably indicates the part related to parameters not taken into account such as communications, boundary conditions, block connectivity and ghost cells.

Mesh partitioning: splitting strategy

To share the workload among all computing cores, the load-balancing algorithm follows a so-called “greedy” algorithm described by Ytterström [29]. This algorithm loops over all the blocks searching for the largest one in terms of cells and allocates it on the computing core having the fewest number of cells until all blocks are allocated. Finally the load balancing error is given by the relative error between the number of cells allocated to the computing core and the ideal number of cells. This basic algorithm may fail in the presence of some large blocks in the set of all blocks. Therefore, to offer a higher flexibility when regarding a large number of cores, a block splitting tool is integrated in the load-balancing algorithm to split the original configuration in several new blocks if necessary. In the load-balancing algorithm, before allocating the largest block, it is checked that the load balancing error of the computing core would not exceed a given user’s value. If this error is exceeded then the largest block is split along its largest dimension at the position to obtain the best load balancing error. One of the new blocks is attributed to the computing core, while the other created block is added to a list of remaining blocks. The choice of the maximal load balancing threshold is challenging because if this parameter is too small, the algorithm will split the
Unsteady Flow Simulations in Compressors

configuration in a very large number of blocks, which is detrimental to CPU and memory as the number of ghost cells and the MPI buffer sizes increase with the number of blocks. On the other hand, a large load balancing error is directly penalizing for the parallel efficiency.

Coincident block interfaces

In this study, communications between blocks connected with coincident interfaces are implemented through MPI blocking calls. As a consequence, the ordering of communications is crucial for parallel scalability and for avoiding deadlock. The scheduling of blocking MPI point-to-point communications comes from a heuristic algorithm adapted from graph theory. The algorithm relies on the construction of a weighted multi-graph as shown in Fig. 4. The graph vertices represent the available computing cores represented by letters P and therefore an edge connects two cores. An edge represents a coincident interface that links two structured blocks symbolized by letters B. The weight of an edge is the number of connection interfaces (cell faces). Since many blocks can be assigned to only one computing core, two vertices of the graph can be connected with many edges, hence the name multi-graph. The underlying principle of the ordering algorithm is to structure the point-to-point communication pattern into successive message passing stages. Indeed, the communication pattern comes into play at some points during an iteration of the solver, whenever blocks need information coming from their neighbours. Each communication stage is represented by a list containing the graph edges that have no computing core in common. In other words, if one edge is taken from this list, then the cores represented by its two vertices will not appear among the other edges. The load-balancing process must ensure that all computing cores send roughly the same number of messages with almost the same size so that the scheduler can minimize the number of communication stages. More precisely, this scheduler is well adapted to complete graphs $K_n$ with an even number of vertices and with edges having the same weight. It is
Unsteady Flow Simulations in Compressors

It is straightforward to see that if the graph has an odd number of vertices then one of the computing cores will be idle during a communication stage since point-to-point messages are used. In fact, this is not really a huge constraint to impose the use of an even number of computing cores. A communication stage deduced from a complete graph $K_n$ involves $N/2$ computing cores and as a consequence the communication pattern holds $(N-1)$ communication stages. Of course, the graph coming from the load-balancing process is probably and fortunately often not complete and its edges have certainly different weights. Hence, some simple criteria can help to assess the behaviour of this load-balancing process. The maximum degree of the graph is the minimum number of communication stages that may take place. The ratio between the maximum and minimum degrees of the graph vertices indicates the rate of idleness of the computing core.

Non-coincident block interfaces

In the current version of the flow solver, communications between blocks connected by non-coincident interfaces are treated after the point-to-point communications and are implemented with collective blocking MPI instructions. The scheduler described above does not handle them. Some basic rules must be followed to manually distribute blocks owning such connectivity. Indeed, a computing core that does not handle this kind of blocks will be idle during the blocking collective MPI instructions. As already said, this type of conditions introduces the idea of a computing core group that manages the exchange of data between blocks. For example, if a CFD configuration has three non-coincident interfaces, then this generates three groups. In that case, the set of all computing cores should be the direct sum of these three groups. Thus, all cores would be involved within a collective communication stage. If only one core was belonging to two groups, then only two groups would be involved within the first collective communication stage and the third group would stay idle during that time and would start its communication in a second stage. As a
Unsteady Flow Simulations in Compressors

conclusion, authors are aware that this implementation suffers from the above drawback that creates an important constraint on the user’s side. This issue is currently being investigated and more precisely, the collective communications are being replaced by point-to-point communications so as they can be taken into account by the load-balancing tool like coincident interfaces.

Influence of the MPI implementation

In this paper, communications are implemented with blocking point-to-point messages. Beforehand, benchmarks have been performed to compare this scheme against non-blocking point-to-point communications implemented with the “MPI_Irecv” and “MPI_Isend” routines. The two implementations have been tested with and without the use of the scheduler. Note that if the scheduler is not active then messages are still scheduled but implicitly. More precisely, the scheduling follows the order in which boundary conditions are read at the initialization of the computation. This scheduling turned out to be bad and it gives an idea of a computation that can be considered in a way not having scheduled messages. The two implementations were tested on a whole aircraft configuration with a 30M cells grid and 1,774 blocks. Results are shown in Fig. 5. The reference time is the computational time observed when using the blocking MPI instructions and it is used to normalize other elapsed times. Without the scheduler, the computational efficiency using the blocking MPI implementation is reduced by 40% while no effect is observed with a non-blocking implementation. So the use of a non-blocking communication scheme allows to get rid of the scheduler. The other major interest of the latter approach is to enable the overlapping of communication and computation but this remains to be investigated. All in all, both methods can be used interchangeably.
4- PERFORMANCE ON VECTOR/SCALAR PLATFORMS

As previously explained, the fundamental difference between scalar and vector approaches is that scalar platforms are well adapted to massively parallel computing by using thousands of computing cores concurrently (but with a limited memory by core) while the vector approach uses a few computing cores with a large memory. Several benchmarks have been performed in order to evaluate the benefits of these approaches and the scalability of the industrial compressor test case in terms of elapsed time (Fig. 6), speedup (Fig. 7, Fig. 8) and power consumption (Fig. 9). For all benchmarks, the performance is evaluated by averaging the time needed to compute ten physical time steps. Three platforms have been selected to show an overview of current available supercomputers. The first one is a vector supercomputer (NEC SX8++) with 16 nodes of 8 computing cores; each node is equipped with 128 Go of RAM and each computing core develops 32 GFlops (peak power). The second platform is a massively scalar computer (IBM Blue Gene /P) that is composed of 8,192 nodes of 4 computing cores; each node is equipped with 2 Go of RAM and each core develops 3.3 GFlops. The third supercomputer is a scalar platform (SGI Altix) composed of 1536 bi-quad cores (i.e. the total is 12288 computing cores); each bi-quad core is equipped with 32 Go of RAM memory and each core develops 12 GFlops.

Computational time

The first comparison is done to estimate the number of computing cores needed to achieve a given elapsed time. Results shown in Fig. 6 (a) and Fig. 6 (b) are normalized by the time required to compute one physical time step with four vector cores (i.e. 2783 s). An identical elapsed time is obtained with 512 scalar cores (IBM), 30 vector cores (NEC) and 72 scalar cores (SGI). These results are useful to underline the efficiency and the interest of parallel computations. On the one hand, the use of a small number of computing cores (<128) does not significantly affect the
Unsteady Flow Simulations in Compressors

computational time that is still largely related to the computing core peak power: to achieve a given elapsed time, the ratio between the required number of vector (NEC) and scalar (SGI) computing cores (=2.7) is very similar to their core peak power ratio (=2.4). On the other hand, computations that consider a large number of computing cores (>128) are more affected by communication cost and load balancing errors, even with an optimized communication network: while the peak power ratio between scalar (IBM) and vector (NEC) computing cores is close to 9.7, the ratio between the number of computing cores required to achieve an identical elapsed time is increased to 17.1. Similar observations are done when comparing IBM and SGI platforms (the ratio between their core peak power is 3.6 and the ratio between the number of computing cores is 7.1). Results obtained for scalar platforms (both IBM and SGI) also indicate that a plateau is reached for the elapsed time when the number of computing cores is increased, showing the limit in terms of scalability.

**Normalized speedup and communication cost**

The previous observation is confirmed by the normalized speedup. Due to memory limitations, the elapsed time with one computing core is not known, so results are normalized with the smallest number of computing core available for the calculation. Comparisons between NEC and IBM platforms are shown in Fig. 7 and comparisons with theoretical speedup are indicated in Fig. 8. A quasi-linear value of the normalized speedup is observed when using a moderate number of computing cores (less than 32 with NEC platform and less than 2048 with IBM platform). However, the parallel efficiency is dramatically reduced when using 4096 computing cores on IBM platform. This observation is compared with theoretical laws (Amdahl’s and extended Amdahl’s laws) for the simulation with the IBM platform. These two laws do not predict the same behaviour: the Amdahl’s law predicts a weak speedup very close to the linear law (the maximum load
Unsteady Flow Simulations in Compressors

balancing error is small) while the EA law predicts a lower value of the normalized speedup (around 6). This comparison shows that the EA law better estimates the real speedup than the Amdahl’s law. However, it only partially explains the large reduction of the normalized speedup when using 4096 computing cores of the IBM platform. To underline the role of communications, Fig. 10 presents the communication ordering obtained on the IBM platform with the MPI Trace library. From left to right, the time represents one sub-iteration of the flow solver. For clarity reasons, data is presented for only 60 cores (simulation is performed with 4096 cores). The white region is related to the computation work while grey and black regions correspond to MPI calls (resp. collective and point-to-point calls). MPI calls are related to the necessity for exchanging information, such as auxiliary quantities (gradients, etc.) at the beginning of the iteration or the increments of conservative variables $\Delta W_i$ during the implicit stages. At the end of the iteration, all the computing cores have also to exchange the vector of conservative variables $W$ by means of MPI collective calls. Two groups of processes are identified in Fig. 10: group 1 is linked to point-to-point and collective calls while group 2 is linked only to point-to-point communications. At the beginning of the flow solver iteration, while the group 1 is still doing collective MPI calls, the group 2 has already started to compute fluxes. During the implicit stage, all processes have to be synchronized (to exchange $\Delta W_i$), explaining why point-to-point communications related to group 2 appear so long. All the computing cores that belong to group 2 are waiting for the information coming from cores of group 1. In this case, collective MPI calls are not correctly balanced, reducing the parallel efficiency.

**Electric consumption**

Based on the manufacturer specifications (*i.e.* 8W for an IBM core, 780W for a NEC core and 49W for a SGI core), it is possible to obtain the power consumption to compute one iteration (Fig. 9).
Unsteady Flow Simulations in Compressors

The massively scalar platform (IBM) requires 1.9 MJ with 512 computing cores, the vector supercomputer (NEC) needs 11.0 MJ with 30 cores and the scalar platform (SGI) requires 1.7 MJ with 72 cores. At identical performance in terms of elapsed time, the vector supercomputer is always more energy consuming than scalar supercomputers. When comparing IBM and SGI platforms, the result depends on the number of computing cores that are considered: for small numbers of computing cores, the SGI platform appears to be more interesting, while the IBM platform is better adapted to the use of a large number of cores. The same electric power is required for 768 cores (IBM) and 128 cores (SGI). The simulation of one compressor rotation is usually sufficient to obtain an unsteady flow solution at design operating conditions. Considering an identical elapsed time, the required power is 5,440 MJ with the SGI scalar computer, 6,080 MJ with the IBM platform and 35,200 MJ with the NEC vector computer. To give an idea of the magnitude order related to this consumption, it is equivalent to run the experimental facility during 45 min (simulation with the SGI platform) or 4 h 54 min (simulation with the NEC supercomputer).

5- APPLICATION TO UNSTEADY FLOW SIMULATION IN COMPRESSORS

Previous observations show that scalar platforms provide the most interesting performance regarding electric cost and elapsed time. A short overview of the contribution that HPC can offer to the scientific and industrial communities is presented in this section. A direct application for turbomachines is to allow the simulation of unsteady flows in multistage compressors, by taking into account all the blades in the totality of the machine. The major interest from the scientific point of view is to understand the development of aerodynamic instabilities and the impact of rotor-stator interactions in these systems. The prediction of the stability limit (the “surge” line) is also a design parameter that cannot be correctly predicted by current industrial CFD methods. A comparison is done between the steady RANS approach combined with the mixing plane assumption proposed by
Unsteady Flow Simulations in Compressors

Denton and Singh [11] (Steady Mixing Plane - SMP) and the unsteady RANS method applied to the whole compressor (unsteady whole compressor - UWC). An overview of the required computational effort is shown Table 2 for SMP and UWC approaches. The fine mesh corresponds to the industrial standard in terms of grid density while the coarse mesh corresponds to the low-quality mesh considered in the first part of this paper. The computational effort is based on performance obtained with the IBM Blue Gene /P supercomputer and at design operating conditions for the compressor. The SMP method considers only a single passage for each blade row, so only 1.5M of points are required for the whole coarse grid. On IBM Blue Gene architectures with 16 computing cores, the computational time to obtain the solution with the SMP method is 55 CPU hours at design conditions and 80 CPU hours at off-design conditions. On the same architecture, the UWC approach requires 280,000 CPU hours to obtain a periodic solution at design operating conditions (with 512 computing cores). This example shows that the computational effort should be increased by a factor 5000 with respect to steady flow simulations to obtain the complete description of unsteady flows generated by rotor-stator interactions. At near stall conditions (off-design) the computational effort can be increased by an additional factor 10 due to the difficulty to achieve a periodic flow state.

A few physical results obtained with SMP and UWC approaches are compared. The normalized total-to-total pressure ratio and the normalized isentropic efficiency are plotted on Fig. 11 with respect to the mass flow $\phi$. While both methods predict similar overall performance at design operating conditions ($\phi=0.965$), discrepancies appear at partial mass flow and tend to increase when the mass flow is reduced. The stability limit is also under-estimated with the SMP method (in terms of mass flow), leading to an increase of the predicted stable operating range by $+11\%$. To quantify more precisely the sources of error, Fig. 12 indicates the relative difference $\psi$ between time-
averaged solutions of axial velocity obtained with SMP and UWC approaches, following the relation:

$$\psi = \left| \frac{\bar{V}_x(SMP) - \bar{V}_x(UWC)}{\bar{V}_x(SMP)} \right|$$

Results are presented at four sections (26A, 27A, 280 and 28A) corresponding to rotor-stator and stator-rotor interfaces. For visibility reasons, only a part of the circumference is shown (a 22.5° sector). In the case of purely steady flows, the SMP method should provide results close to the time-averaged results obtained with the UWC method. Observed differences are thus related directly to mean unsteady flow effects. At section 26A, the solution computed with both methods is similar: the relative difference is very small in the blade passages (<2%) and highest values are observed near the casing and hub walls (locally difference reaches around 10%). However this observation is no longer true for other locations (27A, 280 and 28A). The most visible feature is that the time-averaged flow does not remain identical in all the compressor passages, mainly due to the impact of adjacent blade rows. For example, a periodic flow appears 96 times around the circumference at section 27A, corresponding to the influence of the first stator row. The same remark can be done for section 28A (impacted by the second stator wakes) and section 280 (impacted by the first stator wakes). At section 280, differences with UWC approach are particularly strong: the peak-to-peak fluctuation induced by stator wakes is under-estimated by a factor 2 and adjacent row effects induce differences on the axial velocity flow field close to 10%. This short comparison shows that the SMP approach is not able to reproduce the mean unsteady flow effects, leading to differences with respect to the UWC approach that grow with the stage axial position. Moreover differences tend to increase at off-design operating conditions.
6- CONCLUSIONS

The paper has been written following three objectives. First, the different problems that flow solvers have to overcome to obtain the best performance on modern supercomputers have been presented, with a particular interest for parallel computing. Second, vector and scalar platforms have been evaluated to compute the flow in large industrial configurations in a reasonable amount of time. Finally, an example of the contribution that HPC can offer to the scientific community has been described. The following points summarize this study:

- A mesh of industrial configurations usually requires complex block connectivity in the case of structured multi-blocks flow solver. These methods (sliding mesh, Chimera, etc.) induce communication cost and load-balancing errors that are complex to manage;
- The classical Amdahl’s law is not well suited to the estimation of the elapsed time when the amount of work is different for each computing core (which is usually the case for industrial configurations). An extension to the Amdahl’s law has been proposed to take into account this specificity, showing a good capacity to estimate the speedup for complex configurations;
- The scheduling strategy and the load balancing error largely affect the speedup, especially when thousands of computing cores have to communicate together. It is clear that both aspects have to be correctly estimated and minimized to obtain the best benefit on massively parallel platforms. From this consideration, computing platforms that uses small number of powerful cores leads to better performance;
- Performance obtained with three computing platforms has been compared. An identical elapsed time is observed when using 512 scalar computing cores (IBM Blue Gene /P), 30 vector cores (NEC SX8++) and 72 scalar cores (SGI Altix);
Unsteady Flow Simulations in Compressors

• The performance of a parallel computing that uses a small number of computing cores (≤128) is mainly related to the peak-to-peak power of the computing core. With large number of computing cores, communications counter-balance the architecture peak power;

• The criterion based on the electric power consumption is largely favourable to scalar platforms. For an identical elapsed time, scalar platforms reduce by a factor 5 the consumed electric energy compared to a vector platform. To be concrete, the total energy used to compute the unsteady flow solution in the whole compressor is around 5,000-6,000 MJ (with scalar supercomputers), which represents the total capacity of a nuclear plant (1,000 MW) during a few seconds;

• The computational effort to describe unsteady flow in a whole multistage compressor must be increased by a factor 5000 with respect to steady flow simulations.

Among the perspectives offered by this work, the improvement of the parallel strategy already implemented in the flow solver is necessary, especially regarding communication cost for massively parallel simulations. A work is currently under progress to suppress the blocking collective MPI instructions. Another perspective is to compute unsteady flows in compressors at off-design conditions, which is a promising way to design more efficient and secure gas turbines.

ACKNOWLEDGEMENTS

The authors are grateful to SNECMA for permission to publish results for the CREATE compressor and to LMFA (Ecole Centrale Lyon) for its collaborative work around the CREATE project. The unsteady flow simulation of the whole compressor has been performed on the CERFACS Blue Gene/L facility. The authors thank the Computing Support Group at CERFACS for their help and anonymous reviewers for interesting comments and recommendations. Finally, the authors also acknowledge EDF, GENCI-CINES and Météo-France for providing computational resources used
Unsteady Flow Simulations in Compressors

for benchmarks and numerical simulations (respectively through Blue Gene /P, SGI Altix and NEC-SX8++ platforms).

REFERENCES


Unsteady Flow Simulations in Compressors


The International Journal of High Performance Computing Applications
Unsteady Flow Simulations in Compressors


Unsteady Flow Simulations in Compressors


Unsteady Flow Simulations in Compressors

FIGURES

Fig. 1 The CREATE compressor and its test-rig at Ecole Centrale Lyon.

Fig. 2 The whole simulated domain and partial view of the mesh (1 on 2 points).
Fig. 3 Typical scheme of a parallel computation.

Fig. 4 Simplified example of a multi-graph used for the scheduling process.
Unsteady Flow Simulations in Compressors

Fig. 5 Effect of the MPI implementation on the code efficiency: method 1 is the blocking MPI implementation and method 2 is the non-blocking MPI implementation.

Fig. 6 CPU time needed for a single physical iteration – (a) comparison between vector and scalar platforms, (b) comparison between two scalar platforms.
Unsteady Flow Simulations in Compressors

Fig. 7 Comparison of the speedup for vector and scalar approaches.

Fig. 8 Comparison of the theoretical speedup with observed speedup for the scalar approach.

Fig. 9 Consumed energy for a single physical iteration – (a) comparison between vector and scalar platforms, (b) comparison between two scalar platforms.
Unsteady Flow Simulations in Compressors

Fig. 10 Communication graph during a typical sub-iteration of the DTS scheme
(each line corresponds to one computing core).

Fig. 11 Mean aerodynamic performances – (a) Pressure ratio, (b) isentropic efficiency.
Unsteady Flow Simulations in Compressors

Fig. 12 Relative error between the time-averaged unsteady flow and the steady flow (axial velocity) solutions.
Unsteady Flow Simulations in Compressors

**TABLE**

<table>
<thead>
<tr>
<th>Row</th>
<th>IGV</th>
<th>R1</th>
<th>S1</th>
<th>R2</th>
<th>S2</th>
<th>R3</th>
<th>S3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of blades per row (for 2π)</td>
<td>32</td>
<td>64</td>
<td>96</td>
<td>80</td>
<td>112</td>
<td>80</td>
<td>128</td>
</tr>
<tr>
<td>Number of blades for 2π/16</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>5</td>
<td>7</td>
<td>5</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 1: Number of blades of the compressor rows.

<table>
<thead>
<tr>
<th></th>
<th><strong>Steady RANS (SMP)</strong></th>
<th></th>
<th><strong>Unsteady RANS (UWC)</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Coarse mesh</td>
<td>Fine mesh</td>
<td>Coarse mesh</td>
<td>Fine mesh</td>
</tr>
<tr>
<td>Number of points</td>
<td>1,500,000</td>
<td>6,700,000</td>
<td>134,000,000</td>
<td>600,000,000</td>
</tr>
<tr>
<td>Total CPU (hours)</td>
<td>55</td>
<td>220</td>
<td>280,000</td>
<td>1,250,000</td>
</tr>
<tr>
<td>effort for</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>converged state (RANS) or 5 ms</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Typical computing effort related to steady RANS and unsteady RANS simulations in a multistage compressor (IBM Blue Gene /P and design operating conditions).