V International Conference on Computational Methods for Coupled Problems in Science and Engineering
COUPLED PROBLEMS 2013
S. Idelsohn, M. Papadrakakis and B. Schrefler (Eds)

HIGH PERFORMANCE CONJUGATE HEAT TRANSFER WITH THE OPENPALM COUPLER.

Florent Duchaine*, Stéphan Jauré*, Damien Poitou*, Eric Quémerais†, Gabriel Staffelbach*, Thierry Morel* and Laurent Gicquel*

*CERFACS, 42 avenue G. Coriolis,
31 057 Toulouse Cedex 01, France
e-mail: florent.duchaine@cerfacs.fr

†ONERA, DSNA,
29 Avenue de la Division Leclerc
92 322 Châtillon Cedex, France

Key words: Massively Parallel, Conjugate Heat Transfer, Large Eddy Simulation

Abstract. Optimizing gas turbines is a complex multi-physical and multi-component problem that has long been based on expensive experiments. Today, computer simulations can reduce design process costs and are acknowledged as a promising path for optimization. Although the simulations of specific components of gas turbines become accessible, these stand-alone simulations face a new challenge: to improve the quality of the results, new physics must be introduced. Based on the simulation of conjugate heat transfer within an industrial combustor to predict the temperature of its walls, the current work aims at studying the scalability of code coupling on HPC architectures. Coupling accurately solvers on massively parallel architectures while maintaining their scalability is challenging. The strategy investigated relies on recent developments made in a generic parallel coupler. Performance tests have been carried out until 12,288 cores on the CURIE supercomputer (TGCC / CEA). Results show a good behavior and advanced analyzes are carried out in order to draw new paths for future developments in coupled simulations.

1 Introduction

Determination of heat loads such as wall temperatures and heat fluxes, is a key issue in gas turbine design [1, 2]. With the constant increase of computing power, numerical simulations of the thermal interaction between fluid flows and solids offer new design paths to diminish development costs through important reductions of the number of experimental tests. To determine mean heat loads on structures, many authors use Conjugate Heat Transfer (CHT) where the fluid and solid equations are resolved simultaneously. CHT is a difficult field and most existing tools are developed for chained (rather than coupled),

steady (rather than transient) phenomena thanks to Reynolds Averaged Navier-Stokes (RANS) solvers [3, 4]. The accuracy of the CHT predictions largely relies on the Computational Fluid Dynamics (CFD) method. Recent contributions based on Large Eddy Simulation (LES) [5, 6] provide promising results especially for the prediction of heat transfer in complex geometries [7, 8, 9, 10].

Using an unsteady LES flow solver to resolve such problems raises several complexities to address for CHT. LES requires high mesh resolutions to accurately capture the flow physics. It is also more CPU consuming than RANS methods to converge spatial and temporal statistics. These specificities imply the use of specific strategies to accelerate the convergence toward steady heat transfer problems as well as efficient methods to use existing high performance architectures. Previous studies have proposed guidelines to reach stable convergence of CHT problems with LES based on time desynchronization of convection and conduction as well as the use of high frequency information exchanges between the physics [7, 11].

Note that, there are two basic approaches to numerically solve CHT problems. The first one is a direct coupling approach where the different physics are solved simultaneously in a large system of equations by a monolithic solver. The second approach consists in solving each set of equations separately with dedicated solvers that exchange interface conditions through a coupler. The last solution adopted here has the advantage of using existing state-of-the-art codes to solve fluid and solid equations.

In this context and with the convergence recommendations, the resolution of CHT problems puts a lot of pressure on the tool used to couple the solvers. Several communities have investigated the use of code coupler in many different area ranging from climate studies to industrial applications. These communities are now faced to the challenge of running the coupled applications with highly loaded codes on massively parallel machines where the solvers exchange a lot of amount of data at a high frequency. The strategy investigated in this work to address these issues relies on recent developments made in a generic parallel coupler [12]. The first section presents the fluid and solid solvers. Then, implementation details on the coupling library are provided. Finally performance tests carried out until 12,288 cores on the CURIE supercomputer (TGCC / CEA) are proposed.

2 Presentation of the flow and conduction solvers

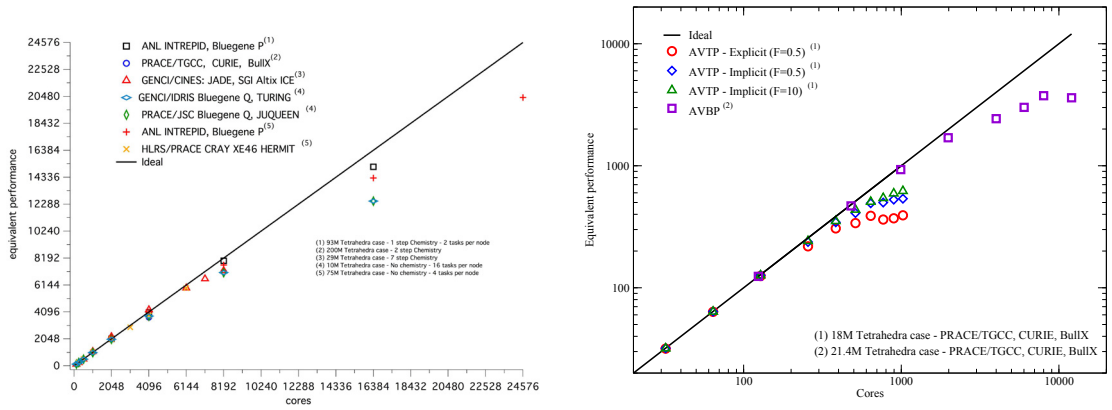
This section aims at presenting the fluid and solid solvers used to construct the CHT tool. Both solvers have a mesh partitioning based parallelism. After a short description of their functionalities, parallel performance of each solver is given.

2.1 The fluid solver AVBP

Recent information about this flow solver can be found in [13]. AVBP solves the compressible Navier-Stokes equations and focuses on unsteady turbulent flows (with and without chemical reactions) for internal flow configurations on unstructured grids. The

prediction of turbulent flows is based on the LES sub-grid scale closure problem [14]. The numerical methods are either 2nd order or 3rd order in time and space [13]. AVBP library includes integrated parallel domain partitioning and data reordering tools, handles message passing and includes supporting routines for dynamic memory allocation, parallel Input/Output and iterative methods.

Typical strong speed-up obtained with AVBP are illustrated on Fig. 1-(a). This flow solver shows an excellent scalability until $8,192$ cores. Results are still good for $24,000$ cores (efficiency is around 85%). The decrease of performance for a very large number of cores underlines a physical limit often encountered on massively parallel applications. Indeed for very large numbers of cores the ratio between computation time and communication time is directly proportional to the problem size (number of grid points and unknowns). For this specific illustration, the configuration tested with $16,000$ cores corresponds to a calculation without enough cells by core or a too low computational workload for each core compared to the amount of exchanges needed to proceed with the CFD computation. It shows that a given task is limited in terms of scalability and no increase of performance is expected beyond a given number of cores.



(a) (b)

Figure 1: Strong speed-up curve obtained for (a) AVBP on different machines and physical configurations, (b) AVBP and AVTP on the CURIE/TGCC supercomputer with the target configuration of the study.

2.2 The solid solver AVTP

The AVTP solver has been written based on the data structure of AVBP. The solver inherits from the mesh capability and the computational performances of AVBP. AVTP solves the time dependent energy conservation equation. A second order centered scheme is used for spacial discretization. Time integration is done either with an explicit or an implicit first order forward Euler scheme.

A typical strong speed-up obtained with AVTP is illustrated on Fig. 1-(b) for explicit

and implicit schemes. Due to the small number of operations per iteration compared to AVBP, the scaling of the explicit integration becomes poor at about 500 cores and then saturates. For the same Fourier number ($F = 0.5$ at which the explicit scheme is limited due to stability reasons), the implicit scheme shows better scaling performances due to the fact that it needs more operations per iteration than the explicit scheme. The price for one iteration on one processor is thus higher when using the implicit scheme than the explicit one. Increasing the Fourier number, the number of operations per iteration increases (the method needs more sub-iterations to converge) and thus the scaling appears to be better ($F = 10$ on Fig. 1). On the other hand, using larger Fourier numbers increase the time step leading to a global decrease of the CPU time consumed to simulate a given physical time, which is why implicit schemes are preferred for this type of solver. Finally, the more the domain is partitioned by increasing the number of core, the more the conjugate gradient algorithm needs sub-iterations to converge which increases the number of operation per iteration and thus participate to the decrease of the efficiency of the solver. After 750 cores, the efficiency of the implicit solver at $F = 10$ is less than 75%.

3 Presentation of the coupler

The OpenPALM software is a code coupler, i.e. a library of functionalities that facilitate the scheduling of existing components execution as well as the exchange of data between these components [12]. This is achieve in part via a collection of primitives that are called in the codes as well as with more complex mechanisms for application scheduling. OpenPALM aims at implementing a general tool allowing to easily integrate high performance computing applications in a flexible and evolutive way proposing a solution to the balance among performance, software reuse and numerical accuracy.

OpenPALM is mainly composed of 3 complementary components, (1) the PALM library[1] , (2) the CWIPI library[2] and (3) the graphical interface PrePALM. As the application programming interface is available in Fortran and C/C++, OpenPALM can couple codes written in different languages. The CWIPI library is the part of OpenPALM that is mostly used in this study and is thus explained in more details in the following. CWIPI aims at providing a fully parallel communication layer for mesh based coupling between several parallel codes with MPI communications. Like most existing coupling libraries for multi-executables paradigm [15, 16, 17], CWIPI is a static coupler in the sense that all the components of the simulations are started at the beginning, exchange data during the run phase and finish together at the end. Coupling is made through 1D, 2D or 3D exchange zones that can be discretized in different ways in the coupled codes. The library takes into account all types of geometrical elements (polygon, polyhedral) with an unstructured description. CWIPI functionalities involve the construction of the communication graph between distributed geometric interfaces through geometrical localization, interpolation

[1]Projet d'Assimilation par Logiciel Multiméthodes

[2]Coupling With Interpolation Parallel Interface

on non coincident meshes, exchange of coupling fields for massively parallel applications as well as visualization file building.

3.1 High performance mesh based coupling

Code coupling is an appealing method to develop multiphysics applications. However if it is done incorrectly it can become a performance pitfall and render useless the efforts invested to optimize each individual code. There are at least two important aspects to take into account to manage efficient code coupling in a HPC context: (1) reducing the overhead of data transfer between the solvers and (2) maintaining a global processor idle time low, unless both codes have perfectly equal CPU per iteration times, the fastest code will have to wait the other. Having a good load balancing is the key to maintain a low idle time and thus reduce CPU waste. The first point requires the most attention and a direct point to point communication between each solver's processors is proposed. Also non matching grids being used, a parallel interpolation method is required. The algorithm consists of two parts: the initialization or setup phase, i.e. where the communication routes and the interpolation coefficients are computed, and the run-time phase, or how inter code synchronization is actually executed. The first phase is done just once per coupled simulation except if the geometries are mobile.

For the description of these phases, lets consider that solvers A and B are linked by a coupling through their respective discretized interface I_A and I_B.

3.1.1 Inter-code communication scheme (ICCS) determination

The communication routes construction consists in projecting the discretized interface I_A on I_B and vice-versa in order to prepare the communication phase. To maintain full scalability, coupling massively parallel applications has to remain a distributed process not only during the run-time part, but also the initialization part. Furthermore distributing the workload in the initialization improves the capacity of the coupled application to handle large simulations (which is a key for future applications).

Connecting the interfaces I_A and I_B means being able to perform geometrical searches from a computational domain into the other in order to locate the degrees of freedom of I_A in I_B and vice-versa. To keep the data distribution, the geometrical searches are performed in a parallel way by avoiding data centralization and sequential treatment. This objective faces a clear difficulty: in massively parallel CFD applications or heat transfer solvers the meshes are partitioned into sub-domains each processed by different processors. This partitioning also applies to the coupling interfaces. As the partitioning algorithm is usually not aware of the coupling process, the different distributions have no reason to match, leading to complex associations between interface processors of both solvers. To address these specific difficulties the CWIPI algorithm is composed of an optimized three levels location method (Algo. 1): the first level is the partition number of the mesh, the second one is the cell number in the selected partition, and the next one

is the mean values computed in the selected cell.

As the process is fully symmetric, lets consider that code A is the source code (where the data is localized for the interpolation) and code B is the target code. The corresponding interfaces I_A and I_B are partitioned on processes, leading to n_A sub-interfaces noted I_A^n and n_B I_B^m with $n \in [1, n_A]$ and $m \in [1, n_B]$. It is worth noting that the number of sub-interfaces n_A (respectively n_B) is lower or equal to the total number of total partition of the code A (respectively B). Only the processes which contains a sub-interface are involved in the projection algorithm.

The location algorithm is parametrized to adjust the size of the bounding box around the source process sub-interface I_A^n for step #2 as well as around the source cells for step #4. This parameter takes the form of a tolerance: increasing it leads to helping locating points when geometries are not exactly matching. Note however that increasing tolerance results in an increase of the time requested by the location algorithm to converge.

Algorithm 1 Inter-code communication scheme (ICCS) determination algorithm

Step0:
- Each partition n of the source code defines its discretized source sub-interface I_A^n to the coupler (nodes coordinates & connectivity of the cells).
- Each partition m of the target code defines its discretized target sub-interface I_B^m to the coupler (nodes coordinates & connectivity of the cells)

Step1: Each process of the the source code defines a surrounding box of its partition I_A^n

Step2: Each process of the source code checks for geometrical intersections of its surrounding box of sub-partition I_A^n with target nodes of the different target sub-interface I_B^m
return Determination of a reduce number of target nodes per source process n
return Construction a first communication graph between source and target processes

Step3: Each process of the source code classifies the previous target nodes in an octree structure in order to optimize the next research step
return Octree structure containing the target nodes

Step4: Each source process defines a sub-box per mesh element of its sub-interface I_A^n

Step5: Each source process checks the intersection between each source cell sub-box of I_A^n and the target nodes classified in the octree
return Determination of a limited number of candidate target nodes per source cell

Step6: For each target node, the source process identifies the closest element of the source sub-interface I_A^n and defines the final communication graph
return Final communication graph from the source processes to the target ones

3.1.2 Communication phase

The communication phase consists in the interpolation of the fields and the exchange of the data between the solvers. The data can be stored either at the center of the cells for cell-centered solvers or at the nodes for cell-vertex solvers. Interpolation is done directly by the source solver via linear methods. Note that user can customize the interpolation with call-back definition. To ensure communication scalability the communication scheme between each solver is based on direct point to point communications between the processors which share a common interface following the communication graph setup in the previous phase. Each processor generally has several counter parts, which have to provide a portion of its data field.

Two communication schemes exist in the library: (1) synchronous and (2) asynchronous. In the synchronous mode, each process of code A that treats a sub-interface I_A^n is involved in a loop of communications with processes of code B that share partially this sub-interface. The bounds of this loop are obtained thanks to the inter-code communication scheme determination phase and follow the natural order of the process numbering. The exchanges are based on the MPI_Sendrecv primitive so that they can be mono or bi-directional. This primitive is a blocking one implying that an exchange between process n of code A with process m of code B has to be finished before starting the exchange between n and $m+1$. This method is not well optimized when a very large number of cores is involved in the coupling. Concerning the asynchronous mode, the exchanges are based on loops around the primitives MPI_Issend for the sending and MPI_Irecv for reception. The completion monitoring of the exchanges is achieve thanks to loops around primitives MPI_Wait. In this mode, the communication times can overlap in a fully transparent way which is prone to be more performant than the synchronous mode. The other advantage of the method is its potential to overlap the communication times with other treatments in the code that don't affect the exchanged fields.

4 Application to an industrial combustion chamber

The configuration of interest is a sector of an annular helicopter combustion chamber (Fig. 2), including the secondary air flow (A), the flame tube (B) and the high pressure distributor (C). The flame tube is fed with air and kerosene through an injector (D). The flame stabilizes in the primary zone (E). The thermal problem studied here by conjugate heat transfer consists in the determination of the temperature of combustor wall as well as of the stator.

The fluid domain is discretized with 3.8 million nodes and 21.4 million tetrahedra. To describe the flame with sufficient accuracy, the mesh is refined in the primary zone where the flame stabilizes. The solid domain is discretized with 3.8 million nodes and 18.2 million tetrahedra. The high number of solid cells is linked to a resolution constraint that requires at least 4 elements in the wall thickness. The conjunction of very thin walls with the use of tetrahedra leads to a large number of elements.

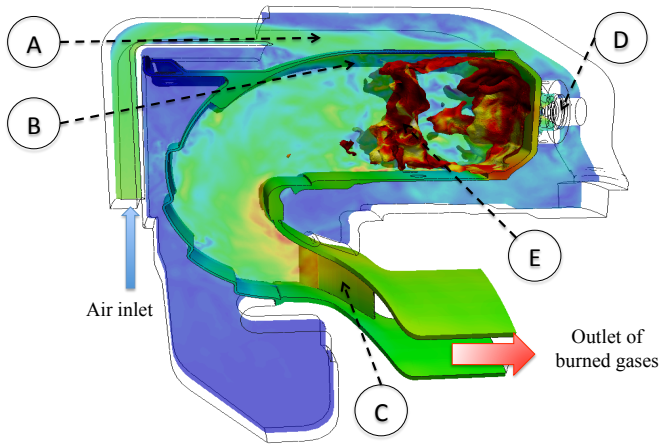


Figure 2: View of the configuration, fluid and solid models. Only one sector of the annular combustion chamber is investigated.

Computations are done on the CURIE supercomputer, owned by GENCI and operated at the TGCC by CEA. CURIE offers different fractions of x86-64 computing resources. Thin nodes among the 5040 B510 bullx nodes of the architectures are used in the present study. Each node is composed of 2 eight-core Intel processors Sandy Bridge EP (E5-2680) with 2.7 GHz, 64 GB and 1 local SSD disk. A total of 80,640 cores are availables on the machine. The coupling has been tested on 768 processors, *ie* 12,288 cores.

The strong speed-up obtained with AVBP on the present configuration is shown on Fig. 1-(b). The performance of the solver are very good till 2,048 cores. After, a significant loss of performance is observed. This is directly linked to two main reasons: (1) the MPI_allreduce that are not optimized in the version of Bullxmpi used for these tests and (2) the size of the fluid mesh which doesn't contain enough degrees of freedom to reach good scaling properties up to 8,000 cores. The strong speed-up obtained with AVTP based on 8 cores on the present configuration is also shown on Fig. 1-(b). The curve is the one already discussed during the AVTP description. A good scalability is observed till 650 cores with the implicit scheme at Fourier $F = 10$ used for the coupled computations.

5 Parallel efficiency results

This section presents the strong scaling analysis of the coupling between AVBP and AVTP from 128 to 12,288 cores. In a first part, the simulations conditions are detailed. Some important features resulting from the application that directly impact the efficiency results are underlined. Finally, the performance of the coupled application are analyzed.

5.1 Simulations conditions

The physics and the numerics of the coupling methodology to reach a steady thermal state of the solid are detailed in [7, 11]. The main idea relies on a desynchronization of

Case #	1	2	3	4	5	6	7	8	9
AVBP	124	224	480	992	1,984	4,000	6,016	8,000	12,032
AVTP	3	31	31	31	63	95	127	191	255
OpenPALM	1	1	1	1	1	1	1	1	1
Total	128	256	512	1,024	2,048	4,096	6,144	8,192	12,288

Table 1: Repartition of cores between AVBP and AVTP for the coupled simulations. The OpenPALM driver takes one core.

the temporal evolutions in the fluid and solid with a high frequency data exchange to reduce the CPU costs while ensuring stability of conjugate heat transfer computations. It results that for the present case, both solvers run in parallel and exchange data at a fixed frequency corresponding to 20 times steps for the fluid code and 1 time step for the solid one.

Table 1 presents the repartition of cores between AVBP and AVTP for the 9 cases tested. Knowing the performances of the codes on the target machine, these repartitions aim to not slow down the fluid code which is the more CPU greedy in case of not perfect synchronization at meeting points (*ie* the cores of the solid solver wait for the fluid ones for data exchange). As a result, the performances of the coupling are compared to AVBP performances in the following of the paper. Note that the driver of OpenPALM runs on one core and that AVBP always runs on a whole processor count in order to avoid processor sharing between AVBP and AVTP.

AVBP treats around 0.48 million cells on its discretized coupling interface and AVTP about 1.7 million elements. The core distribution among the solvers as well as the number of cores on which these coupling interfaces are partitioned for the 9 cases are shown on Fig. 3. It is worth noting that 100% of the solid cores are involved in the coupling process for all cases. On the other hand, the proportion of AVBP processes involved in the coupling interface decreases almost exponentially when the number of cores increases. This behavior is linked to the ratio between the volume of the configuration and its surface which is higher in the case of the fluid solver. It is important to note that mesh partitioning is managed independently in each solver resulting in non conformal patterns of the partitioned discretized surfaces.

It is thus of interest to analyze the evolution of the distribution of the number of mesh elements in the sub-interfaces as the number of cores increases. Indeed, a reduced amount of AVBP cores have more task to perform than the others in order to perform the exchanges with AVTP. Moreover, the way this additional load is distributed and its evolution when the number of processing cores increases is particularly important to ensure the scaling of the coupled application. To analyze these distributions, Fig. 4 and 5 show the probability density functions (PDF) of the number of cores treating n_c cells (these PDFs are constructed based only on the cores with coupling cells). Abscissa scales are logarithmic in order to facilitate the interpretation. Concerning the fluid solver AVBP,

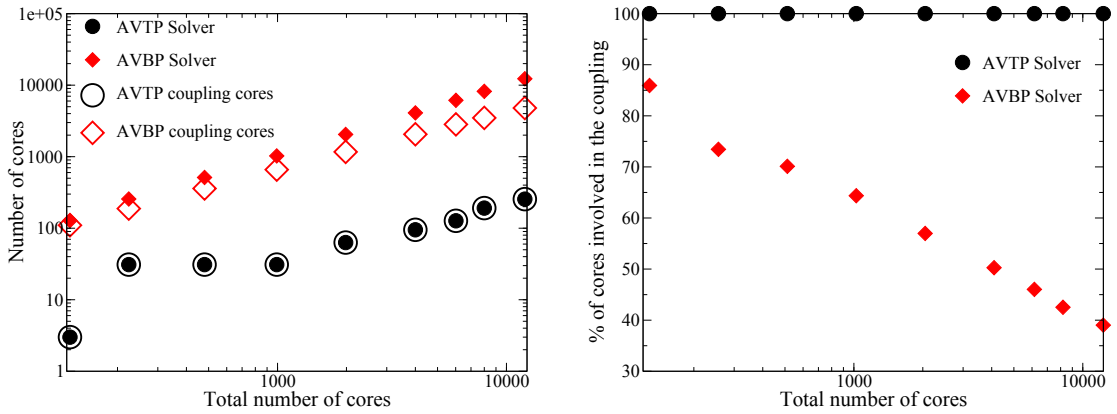


Figure 3: (a) Repartition of the total cores allocated to the solvers as well as evolution of the number of cores involved in the coupling and (b) corresponding percentage of cores involved in the coupling process for each solver .

the PDFs at low number of cores are rather spread out with distinct peaks, reflecting an inhomogeneous partitioning of the coupled discretized interface among the computing cores. Increasing the number of cores narrows the range of the PDF around the small number of cells. A significant peak emerges indicating a homogenization of the distribution of coupling cells on the cores. Nevertheless, a peak at very low number of cells exists highlighting the participation of cores in the process of inter-code exchanges for a small number of information. From case #3 to 9, the peak has the same order of magnitude as the main peak of the distribution.

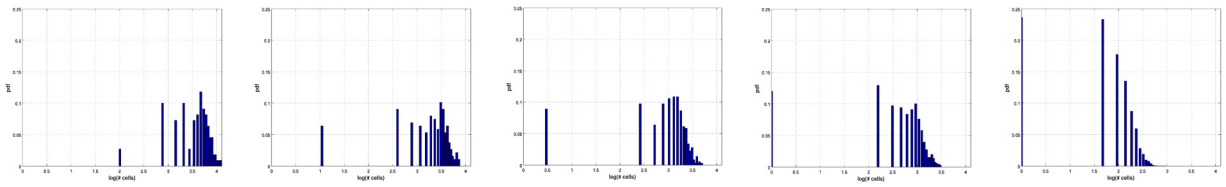
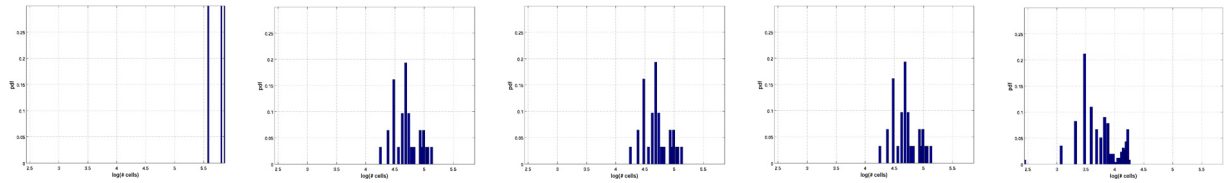


Figure 4: Probability density functions of the distribution of coupling cells per AVBP core depending on the total number of cores of the coupled simulations.

As far as AVTP is concerned, case #1 exhibits a perfectly homogenous distribution on the 3 cores. Then all the distributions show a multimodal profile with a marked peak in the range of small numbers of coupling cells per core.

The analyzes of this section underline phenomenological issues resulting from the coupler as well as the CHT application. Some of these results consist of potential weaknesses for the performance results: imbalance between the number of cores allocated to the



case #1 case #2 case #3 case #4 case #9

Figure 5: Probability density functions of the distribution of coupling cells per AVTP core depending on the total number of cores of the coupled simulations.

solvers, all the cores of the AVBP solver don't participate to the coupling creating a difference of computing load, imbalance repartition of the sub-interface among the coupling cores on each solver, complex communication scheme with a lot of connection between processes of the solvers, and finally a combination of these features. Knowing these features, the next section analyzes the scalability of the application.

5.2 Performance of the coupled application

The time needed to construct the ICCS is presented on Fig. 6-(a). Except for case #1, this time is almost constant when the number of cores increases. For case #1, the ratio of cores involved in the coupling for AVBP over the one involved in the coupling for AVTP is the most important: this imbalance of cores that have to communicate together leads to a very poor efficiency of the algorithm. Increasing the number of cores for AVTP from case #1 to case #2 leads to a drastic reduction of the consumed time. From case #2 to 4, the number of cores allocated to the AVTP coupling interface is fixed while the number of AVBP cores increases. It results a slight increase of the time spent to construct the communication scheme. Then, the ratio of coupling cores between the solvers is almost constant resulting in a slight decrease of the time consumed by the ICCS algorithm until 8,000 cores. From this analyzes, one can note that the scaling of the ICCS algorithm largely depends on the number of processes involved in the coupling on each side of the coupled problem. Increasing the number of cores of just one code is not sufficient to ensure good scaling of this step.

Figure 6-(b) presents the evolution of the time taken by an AVBP iteration compared to the time of an exchange between the solvers as a function of the total number of cores of the coupled system. The exchanges are made in the synchronous mode. As previously mentioned, the fluid solver exhibits a rather good scaling until 4,000 cores. Above 8,000 cores, the time of an iteration reaches a plateau. The time needed for the data exchange is almost constant on the whole range of cores. As for the first phase, the time of the communication step exhibits an important decrease from case #1 to 2. This reveals that the balance of processes that are coupled continues to play an important part in this phase. Then, the communication time slightly increases as the number of cores grows. Until several thousand of cores, the time requested by an exchange is several order of

magnitude lower than the time of an AVBP iteration. Then both times are of the same order. As the coupling is done every 20 iterations of the fluid solver AVBP, the global scaling of the application is conserved as shown on Fig. 7. The communication phase is thus fully transparent in term of restitution time for the user.

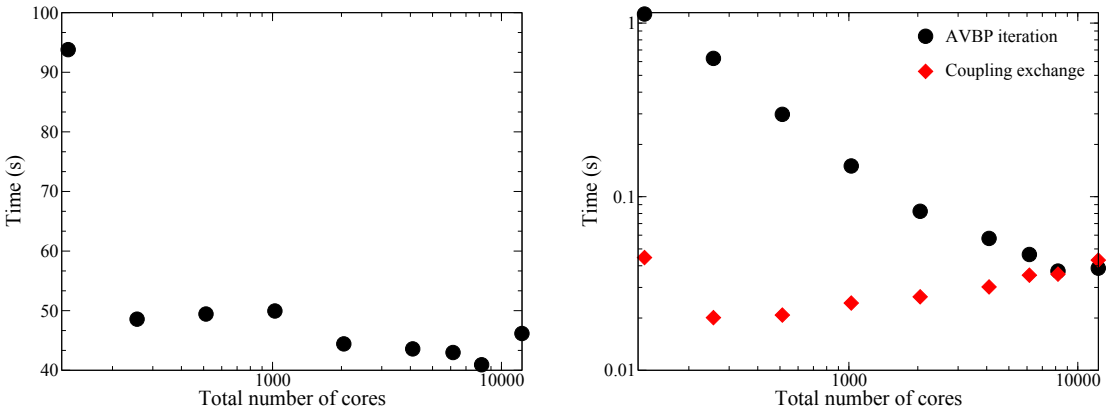


Figure 6: (a) Time taken by the inter-code communication scheme determination algorithm, (b) comparison of the time for one iteration of AVBP and the time to do one exchange as a function of the total number of cores of the coupling.

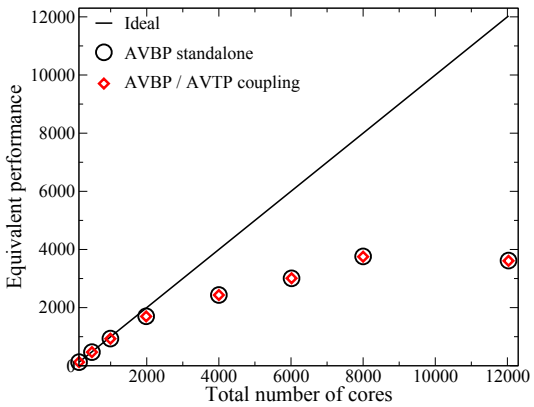


Figure 7: Comparison of the strong speed-up curve obtained for AVBP / AVTP coupling with the AVBP standalone one.

6 Conclusion

The different choices made for the resolution of conjugate heat transfer problems on complex geometries put a lot of pressure on the tool used for coupling. As in several

communities, developers are now faced to the challenge of running coupled applications with highly loaded codes on massively parallel machines where the solvers exchange a lot of data at a high frequency. The strategy investigated in this work to address these issues relies on recent developments made in a generic parallel coupler. The characteristics of the coupled tool including the fluid and solid solvers as well as the coupler are detailed. Then, the tool is applied to an industrial combustion chamber. Performance tests are carried out until 12,288 cores on the CURIE supercomputer (TGCC / CEA) and relevant informations that influence the coupling scalability are detailed. The coupler exhibits a very good behavior up to 12,288 cores implying that the use of HPC can drastically reduce the restitution time of coupled applications for industrial design with high fidelity solvers. Analyzes of the scaling response underline the impact of imbalanced repartition of cores among the codes, imbalance repartition of the sub-interface among the coupling cores on each solver, as well as the complex communication scheme with a lot of connections between processes of the solvers on coupling overhead. These points are independent from the coupler and can be addressed by incorporating the knowledge of the coupling in the preprocessing step of the solvers (constraint and co-partitioning). Moreover, recent tests on asynchronous communication show an important improvement of the scalability of the coupler indicating development paths for the future.

The work was supported by the Fondation de Recherche en Aéronautique et Espace in the project STRASS. The TGCC is gratefully acknowledged for providing access of the CURIE machine during the *Grand Challenge*. Turbomeca is acknowledged for supporting code developments and permission for publishing results.

REFERENCES

[1] Lakshminarayana, B., 1996. *Fluid Dynamics and Heat Transfer of Turbomachinery*. Wiley.

[2] Bunker, R. S., 2006. "Gas turbine heat transfer: 10 remaning hot gas path challanges". In Procceedings of GT2006. ASME Turbo Expo 2006.

[3] Schiele, R., and Wittig, S., 2000. "Gas turbine heat transfer: Past and future challenges". *J. Prop. Power ,* ***16***(4), July, pp. 583–589.

[4] Garg, V., 2002. "Heat transfer research on gas turbine airfoils at NASA GRC". *Int. J. Heat Fluid Flow ,* ***23***(2), April, pp. 109–136.

[5] Sagaut, P., and Deck, S., 2009. "Large-eddy simulation for aeronadymics: status and perspectives". *Phil. Trans. R. Soc. Lond.,* ***367***, pp. 2849–2860.

[6] Leonard, T., Duchaine, F., Gourdain, N., and Gicquel, L., 2010. "Steady / unsteady reynolds averaged navier-stokes and large eddy simulastions of a turbine blade at high subsonic outlet mach number". In ASME Turbo Expo, GT2010-22469, ed.

[7] Duchaine, F., Corpron, A., Pons, L., Moureau, V., Nicoud, F., and Poinsot, T., 2009. "Development and assessment of a coupled strategy for conjugate heat transfer with large eddy simulation: Application to a cooled turbine blade". *Int. J. Heat Fluid Flow ,* ***30***(6), pp. 1129–1141.

[8] Bhaskaran, R., and Lele, S. K., 2010. "Large eddy simulation of free-stream turbulence effects on heat transfer to a high-pressure turbine cascade". *Journal of Turbulence*, p. N6.

[9] Maheu, N., Moureau, V., Domingo, P., Duchaine, F., and Balarac, G., 2012. "Large-eddy simulations of flow and heat transfer around a low-mach number turbine blade". In Proc. of the Summer Program , N. A. U. Center for Turbulence Research, ed.

[10] Morata, E. C., Gourdain, N., Duchaine, F., and Gicquel, L., 2012. "Effects of free-stream turbulence on high pressure turbine blade heat transfer predicted by structured and unstructured les". *International Journal of Heat and Mass Transfer,* ***55***(21-22), pp. 5754 – 5768.

[11] Jauré, S., Duchaine, F., Staffelbach, G., and Gicquel, L., 2013. "Massively parallel conjugate heat transfer solver based on large eddy simulation and application to an aeronautical combustion chamber". *Comput. Sci. Disc.,* ***Submitted***.

[12] Piacentini, A., Morel, T., Thevenin, A., and Duchaine, F., 2011. "Open-palm: an open source dynamic parallel coupler.". In IV International Conference on Computational Methods for Coupled Problems in Science and Engineering.

[13] Gicquel, L., Gourdain, N., Boussuge, J.-F., Deniau, H., Staffelbach, G., Wolf, P., and Poinsot, T., 2011. "High performance parallel computing of flows in complex geometries". *Comptes Rendus Mécanique,* ***339***(2-3), pp. 104 – 124.

[14] Sagaut, P., 2000. *Large Eddy Simulation for incompressible flows*. Scientific computation series. Springer-Verlag.

[15] Joppich, W., and Kürschner, M., 2006. "Mpcci - a tool for the simulation of coupled applications". *Concurrency and Computation: Practice and Experience,* ***18***(2), pp. 183–192.

[16] DeCecchis, D., Drummond, L., and Castillo, J., 2011. "Design of a distributed coupling toolkit for high performance computing environment". *Mathematical and Computer Modelling*.

[17] Valcke, S., Balaji, V., Craig, A., DeLuca, C., Dunlap, R., Ford, R. W., Jacob, R., Larson, J., O'Kuinghttons, R., Riley, G. D., , and Vertenstein, M., 2012. "Coupling technologies for earth system modelling". *Geosci. Model Dev. Discuss.,* ***5***, pp. 1987–2006.