Available online at www.prace-ri.eu

Partnership for Advanced Computing in Europe

PRACE Project Airinnova: Automation of High-Fidelity CFD Analysis for Aircraft Design and Optimization

Mengmeng Zhang[a*], Jing Gong[b], Lilit Axner[b], Michaela Barth[b]

[a]Airinnova AB, SE-182 48, Sweden
[b]PDC Center for High Performance Computing, KTH Royal Institute of Technology, SE-100 44, Sweden

Abstract

Airinnova is a start-up company with a key competency in the automation of high-fidelity computational fluid dynamics (CFD) analysis. Following on from our previous PRACE SHAPE project, we have continued collaborating with the PDC Center for High Performance Computing at the KTH Royal Institute of Technology (KTH-PDC), to investigate the performance analysis of the open source CFD code SU2 and further develop the automation process for the field of aerodynamic optimization and design.

1. Introduction

Airinnova AB is a start-up company that provides and develops advanced computational solutions for cutting-edge preliminary designs for aircraft, including computational aerodynamics, and multi-disciplinary optimization. High fidelity CFD analysis is a vital tool in modern aircraft design and optimization, and the computational capability is a limiting factor. High fidelity CFD requires special skills to generate suitable meshes for modelling aircraft and to execute the analysis code, which constrains the use of CFD analysis for aircraft design to a limited number of people who already have those skills.

Airinnova is working on establishing methods to perform CFD analysis automatically, and thus make it possible for CFD to be used far more widely in aircraft design, as researchers and designers without the necessary CFD skills would be able to use the automatic procedures. To achieve this, Airinnova is developing an automated procedure to carry out Reynolds-Averaged Naiver-Stokes (RANS) based CFD analysis, either using watertight aircraft geometries defined by Computer-Aided Design (CAD) files, or using Common Parametric Aircraft Configuration Schema (CPACS) files. Figure 1 illustrates the flow of data for the automated CFD process that Airinnova provides. It starts from a watertight aircraft geometry (left), and then the mesh to represent the aircraft is automatically generated (middle) for CFD analysis to be performance with the selected turbulent models (right).

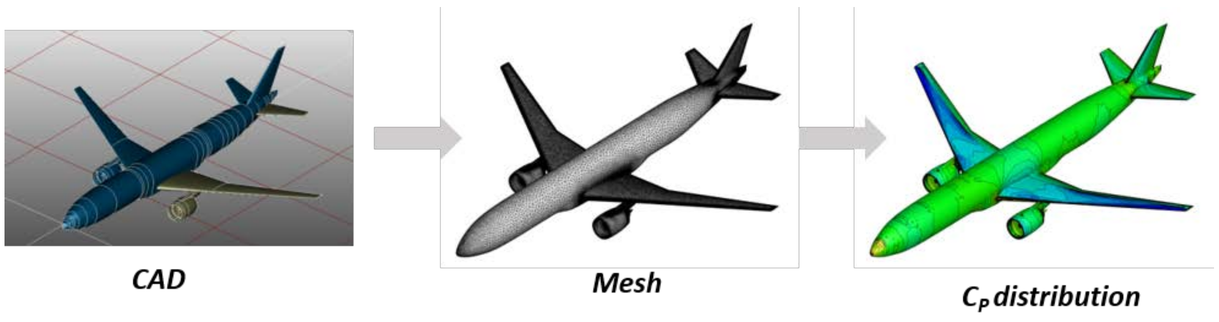


Figure 1: Data flow of the automation CFD prototype that Airinnova provides

[*] Corresponding author. E-mail address: mengmeng.zhang@airinnova.se

The automation of CFD analysis can be applied to many fields within the aviation design industry. Some typical examples of the application of CFD analysis are as follows:

1. *Aerodynamic shape optimization:*
   Set up design parameters for a wing to facilitate the optimization work:
   a. using the Free Formal Deformation box control points around a wing, and
   b. using the standard shape parameters of an aircraft planform, as well as the twist, camber, and thickness for a wing.
2. *Compilation of aero-data for flight simulation:*
   Execute computations for a set of flight states and collect the aero-coefficients in a table/matrix in .csv format (post-processing part).

Both of these applications require high fidelity CFD simulations, such as RANS solvers with turbulence models and/or LES solvers. The computational time (that is, how it takes to run the simulations) is a limiting factor that determines whether it is possible to carry out the simulations on the available computational resources in real time. In order to make it is possible to actually perform such computationally intensive work, the simulations must be parallelized - thus the total run-time can be reduced to make the simulations feasible when run on high performance computing (HPC) resources. Within the PRACE SHAPE project, the CFD computations have been speeded-up using HPC resources; the test cases that were used were the NASA Common Research Model (CRM) [1,2] and a regional jet-liner (A320-alike). The solutions that were obtained are shown for each application respectively. This demonstrated that the automation of CFD analysis for aircraft design is both viable and valuable.

2. Background

The compressible Navier-Stokes equations can be written in conservative form as [3]

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \mathbf{u}) &= 0, \\ \frac{\partial (\rho \boldsymbol{u})}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \mathbf{u})\boldsymbol{u} + \boldsymbol{\nabla} p &= 0, \\ \frac{\partial E}{\partial t} + \boldsymbol{\nabla} \cdot \big(\boldsymbol{u}(E+p)\big) &= 0, \end{aligned} \tag{1}$$

with suitable initial and boundary conditions, where ρ is the density, $\boldsymbol{u} = (u, v, w)$ is the velocity, E is the total energy per unit mass, and p is the static pressure. For the case of constant viscosity coefficient μ, the shear stress tensor $\boldsymbol{\tau}$ for a Newtonian fluid is

$$\boldsymbol{\tau} = \mu[\boldsymbol{\nabla u} + (\boldsymbol{\nabla u})^{\boldsymbol{T}}] - \tfrac{2}{3}\mu(\boldsymbol{\nabla} \cdot \mathbf{u})\boldsymbol{I} \tag{2}$$

By applying the relations in (2), Equations (1) can be rewritten in vector notation as RANS (Reynolds-averaged Navier-Stokes) equations

$$\tfrac{\partial \boldsymbol{U}}{\partial t} + \boldsymbol{\nabla} \cdot \vec{\boldsymbol{F}}^{c}(\boldsymbol{U}) - \boldsymbol{\nabla} \cdot \vec{\boldsymbol{F}}^{v}(\boldsymbol{U}) = 0, \tag{3}$$

where the conservative variables, the convective fluxes, and viscous fluxes tensor are given by,

$$\boldsymbol{U} = \begin{bmatrix} \rho \\ \rho \boldsymbol{u} \\ \rho E \end{bmatrix}, \quad \boldsymbol{F}^{c}(\boldsymbol{U}) = \begin{bmatrix} \rho \boldsymbol{u} \\ \rho \boldsymbol{u}\boldsymbol{u} + p\boldsymbol{I} \\ \rho H \boldsymbol{u} \end{bmatrix}, \quad \boldsymbol{F}^{v}(\boldsymbol{U}) = \begin{bmatrix} 0 \\ \boldsymbol{\tau} \\ \boldsymbol{\tau} \cdot \boldsymbol{u} + \mu^{*}_{tot} c_p \nabla T \end{bmatrix}$$

where c_p is the specific heat at constant pressure, and T is the temperature. Assuming a perfect gas with a ratio of specific heats γ and gas constant R, one can determine the pressure from $p = (\gamma - 1)$, the temperature is given by $T = p/(\rho R)$, and $c_p = \gamma R/(\gamma - 1)$.

The open source code, SU2 [4] for CFD simulation and shape optimization has been investigated in our previous project. SU2 is a computational analysis and design package that has been developed to solve multi-physics analysis and optimization tasks using unstructured mesh topologies [5-7]. In SU2, the standard edge-based finite volume method is used to discretize the governing equations (3). The computational domain consists of non-overlapping elements and the variables are stored at the nodes of the mesh. For each node, the control volume that constitutes the dual grid is defined as a polygon with its vertices at the centers of gravity of the surrounding elements and at the midpoints of the sides, see [5].

By employing the integral form of Equation (3), we obtain

$$\int_{\Omega} \frac{\partial \boldsymbol{U}}{\partial t} \mathrm{d}V + \int_{\partial\Omega} \boldsymbol{F}^{c}(\mathbf{U}) \cdot \mathbf{n} \mathrm{d}A - \int_{\partial\Omega} \boldsymbol{F}^{v}(\mathbf{U}) \cdot \mathbf{n} \mathrm{d}A = 0, \tag{4}$$

and the implicit form of the semi-discrete system becomes

$$\frac{d}{\mathrm{d}t}(|\Omega_i|U_i) + R_i(U) = 0, \tag{5}$$

where the numerical residual $R_i(U)$ at each vertex is evaluated with each nonlinear iteration using a sequence of loops over the edges and vertices, for details see [6]. Since the discretization in time is independent of that in space, the semi-discrete system in Equation (5) can be calculated using the standard numerical methods (e.g., implicit Euler method) for ordinary differential equations, see [7].

3. Profiling and performance analysis

CrayPat is a profiling analysis tool for the Cray XC platform provided by Cray [8]. In principal, there are two methods of collecting code performance data, namely sampling and tracing. In order to reduce the overhead and capture the MPI features, we use sampling data which provides basic performance analysis information by sampling the program counter (PC) at a particular time interval (1 millisecond by default). Firstly the CrayPat command pat_build is used to generate a new executable, which will have suffix +pat, and then run the executable on the computer nodes. A performance data file will be created when the analysis is finished. The command pat_report prints an ASCII text report with detailed profiling data. CrayPat also provides an application program interface (API). By using this API, we can focus on analyzing the performance of the solution section of the code and skip the initialization section. In this profiling analysis, The CrayPat API is introduced in the main function SU2_CFD.cpp as follows.

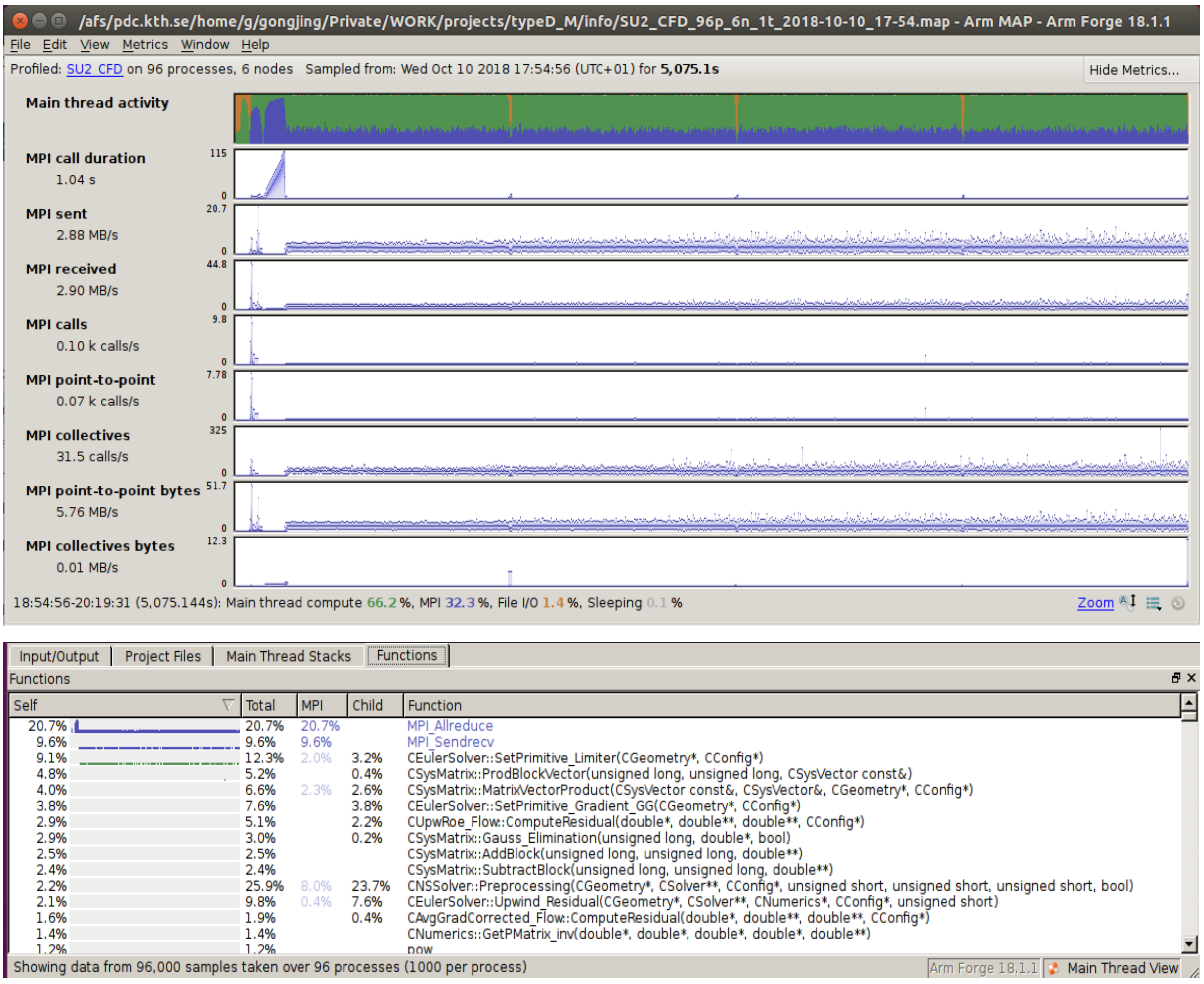
```
#ifdef CRAYPAT
istatpat = PAT_record(PAT_STATE_ON); // Set PAT_STATE on
#endif
Driver->StartSolver();             // Launch the solver
#ifdef CRAYPAT
istatpat = PAT_record(PAT_STATE_OFF); // Set PAT_STATE off
#endif
```

As an example, the profile by function with 96 MPI-rank on ARCHER system is shown in Table 1.

```
  Samp% |      Samp |     Imb. |  Imb. | Group
        |           |     Samp | Samp% |  Function
 100.0% | 512,491.2 |       -- |    -- | Total
|---------------------------------------------------------------------------
|  64.9% | 332,820.2 |       -- |    -- | USER
||--------------------------------------------------------------------------
||   9.2% |  47,372.9 | 20,609.1 | 30.6% | CEulerSolver::SetPrimitive_Limiter
||   4.8% |  24,495.9 |  9,369.1 | 28.0% | CSysMatrix::ProdBlockVector
||   4.1% |  20,829.4 |  9,433.6 | 31.5% | CEulerSolver::SetPrimitive_Gradient_GG
||   3.9% |  20,106.4 | 11,017.6 | 35.8% | CSysMatrix::MatrixVectorProduct
||   3.0% |  15,138.4 |  8,359.6 | 36.0% | CUpwRoe_Flow::ComputeResidual
||   2.7% |  14,024.7 |    894.3 |  6.1% | CSysMatrix::Gauss_Elimination
||   2.6% |  13,433.5 |  8,984.5 | 40.5% | CSysMatrix::AddBlock
||   2.5% |  12,821.7 |  8,827.3 | 41.2% | CSysMatrix::SubtractBlock
||   2.2% |  11,429.7 | 11,241.3 | 50.1% | CNSSolver::Preprocessing
||   2.1% |  10,954.9 |  5,562.1 | 34.0% | CEulerSolver::Upwind_Residual
||   1.8% |   9,229.9 |  2,120.1 | 18.9% | CEulerVariable::GetPrimitive
||   1.6% |   8,380.0 |  3,666.0 | 30.8% | CAvgGradCorrected_Flow::ComputeResidual
||   1.5% |   7,854.2 |  4,304.8 | 35.8% | CNumerics::GetPMatrix_inv
||   1.0% |   5,352.5 |  2,787.5 | 34.6% | CEulerVariable::SubtractGradient_Primitive
||   1.0% |   5,265.0 |  2,176.0 | 29.6% | CNSSolver::SetTime_Step
||==========================================================================
|  32.4% | 166,061.1 |       -- |    -- | MPI
||--------------------------------------------------------------------------
||  20.6% | 105,486.0 | 80,681.0 | 43.8% | MPI_Allreduce
||   9.7% |  49,905.7 | 54,535.3 | 52.8% | MPI_Sendrecv
||==========================================================================
```

Table 2: The profile by function with 96 MPI-rank on ARCHER using the CrayPat

ARM MAP is another parallel profiler with a simple graphical user interface [9]. After completing the run, MAP displays the collected performance data using GUI. A screenshot for the profiling results with 96 MPI-rank is shown in Figure 2.


/afs/pdc.kth.se/home/g/gongjing/Private/WORK/projects/typeD_M/info/SU2_CFD_96p_6n_1t_2018-10-10_17-54.map - Arm MAP - Arm Forge 18.1.1
File Edit View Metrics Window Help
Hide Metrics...
Zoom
Input/Output
Project Files
Main Thread Stacks
Functions

Profiled: SU2_CFD on 96 processes, 6 nodes Sampled from: Wed Oct 10 2018 17:54:56 (UTC+01) for 5,075.1s

Main thread activity

MPI call duration 1.04 s

MPI sent 2.88 MB/s

MPI received 2.90 MB/s

MPI calls 0.10 k calls/s

MPI point-to-point 0.07 k calls/s

MPI collectives 31.5 calls/s

MPI point-to-point bytes 5.76 MB/s

MPI collectives bytes 0.01 MB/s

18:54:56-20:19:31 (5,075.144s): Main thread compute 66.2 %, MPI 32.3 %, File I/O 1.4 %, Sleeping 0.1 %

Functions

Self	Total	MPI	Child	Function
20.7%	20.7%	20.7%		MPI_Allreduce
9.6%	9.6%	9.6%		MPI_Sendrecv
9.1%	12.3%	2.0%	3.2%	CEulerSolver::SetPrimitive_Limiter(CGeometry*, CConfig*)
4.8%	5.2%		0.4%	CSysMatrix::ProdBlockVector(unsigned long, unsigned long, CSysVector const&)
4.0%	6.6%	2.3%	2.6%	CSysMatrix::MatrixVectorProduct(CSysVector const&, CSysVector&, CGeometry*, CConfig*)
3.8%	7.6%		3.8%	CEulerSolver::SetPrimitive_Gradient_GG(CGeometry*, CConfig*)
2.9%	5.1%		2.2%	CUpwRoe_Flow::ComputeResidual(double*, double**, double**, CConfig*)
2.9%	3.0%		0.2%	CSysMatrix::Gauss_Elimination(unsigned long, double*, bool)
2.5%	2.5%			CSysMatrix::AddBlock(unsigned long, unsigned long, double**)
2.4%	2.4%			CSysMatrix::SubtractBlock(unsigned long, unsigned long, double**)
2.2%	25.9%	8.0%	23.7%	CNSSolver::Preprocessing(CGeometry*, CSolver**, CConfig*, unsigned short, unsigned short, unsigned short, bool)
2.1%	9.8%	0.4%	7.6%	CEulerSolver::Upwind_Residual(CGeometry*, CSolver**, CNumerics*, CConfig*, unsigned short)
1.6%	1.9%		0.4%	CAvgGradCorrected_Flow::ComputeResidual(double*, double**, double**, CConfig*)
1.4%	1.4%			CNumerics::GetPMatrix_inv(double*, double*, double*, double*, double**)
1.2%	1.2%			pow

Showing data from 96,000 samples taken over 96 processes (1000 per process) Arm Forge 18.1.1 Main Thread View

Figure 2: The screenshot of profiling results of 96 MPI-rank on Archer using ARM MAP

ARM MAP provides various viewpoints to present collected performance data in different sections. In Figure 2, the top section displays the “Metrics view” with MPI groups such as “MPI call duration”, “MPI point-to-point” etc. The “Functions View" in the bottom displays the profiles of the functions in the SU2 code. The exclusive time or the time spent in the function itself is shown in self. The value in the Total column is for the time in the function itself and all its calltrees or inclusive time while the one in the Child column shows the time spent in the calltrees only. To improve the scalability at large scale, it would be important to conduct identity analysis and eliminate the MPI collection communications (e.g., MPI_Allreduce()) just for verbose information. From Table 1 and Figure 2, we find that the MPI communications take around 30% of the total execution time.

The benchmarking tests were performed on the ARCHER system at EPCC, UK [10]. ARCHER is a Cray XC30 system supercomputer based on Intel Ivy Bridge processors. It has a total of 4920 computer nodes (118,080 cores). The local Cray XC40 system, Beskow at PDC [12], was also used for debugging and tests. Beskow is based on Intel Haswell processors and Cray Aries interconnect technology. It consists of 2060 compute nodes, each of which consists of 32 Intel Xeon E5-2698v3 cores respectively 36 Intel Xeon E5-2695v4 cores.

The scalability tests for two cases were carried out on ARCHER. One case consisted of mesh with 7.0M grid points and the other had 8.5M grid points. Firstly, we compare with the base-line using the SU2 v6.0.0 and Cray GNU environment mode PrgEnv-gnu. The average execution time per step with different MPICH environment variables is shown in Table 2.

Flags	Time/Step (s)
“-O2”	0.620737
“-O2” + MPICH_USE_DMAPP_COLL=1	0.625222
“-O2” + MPICH_RMA_OVER_DMAPP=1	0.615493
“-O2” + craype-hugepages16M	0.595895
“-O2” + MPICH_RANK_ORDER	0.611220
“-O2” + “-j2 -d2” (hyperthreading)	0.692701

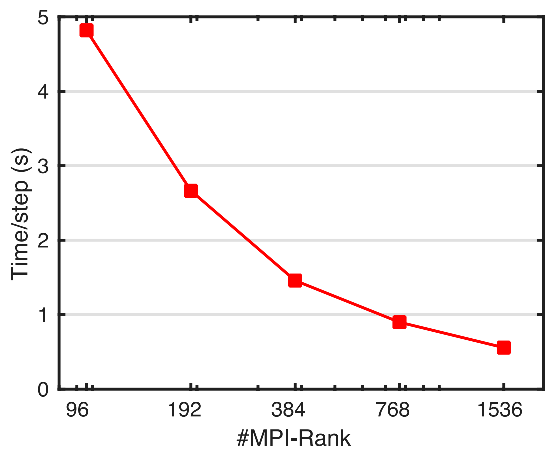
Table 2 The average executive time per step using different MPICH environment variables

Figure 3(a) shows the performance results for the mesh consisting of 7.0 million points on ARCHER. The benchmarking tests ran for 1000 iterations. The strong scaling performance measured in total execution time in seconds is presented in this figure. The parallel efficiency $\eta(\%)$ at maximum used cores is defined as [13]:

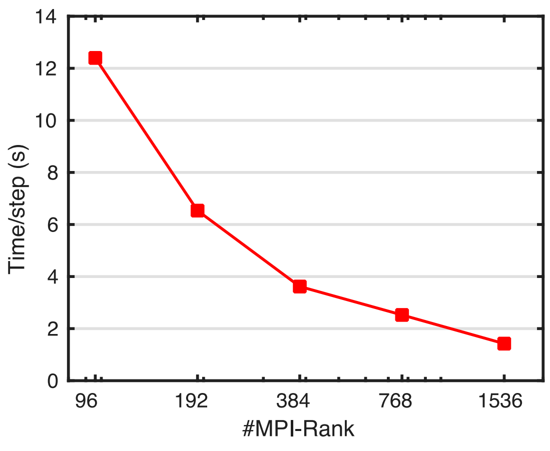
$$\eta(\%) = \frac{T_{min} \cdot N_{min}}{T_{max} \cdot N_{max}} \cdot 100$$

where T_{min} is the total execution time for the minimum number of cores N_{min}, while T_{max} is the total execution time using the maximum number of cores N_{max}.

A parallel efficiency $\eta = 54.1\%$ can be achieved for up to 1536 cores compared with 96 cores. Figure 3(b) shows the performance results for a mesh of baseline with 8.5 million mesh points on ARCHER. A parallel efficiency $\eta = 54.7\%$ of can be obtained for 1536 cores compared with 96 cores. As the MPI-rank increases, the parallel efficiency drops down to 50.0%. The MPI communication time dominates the execution time in comparison to the computation time.



(a) Case 1



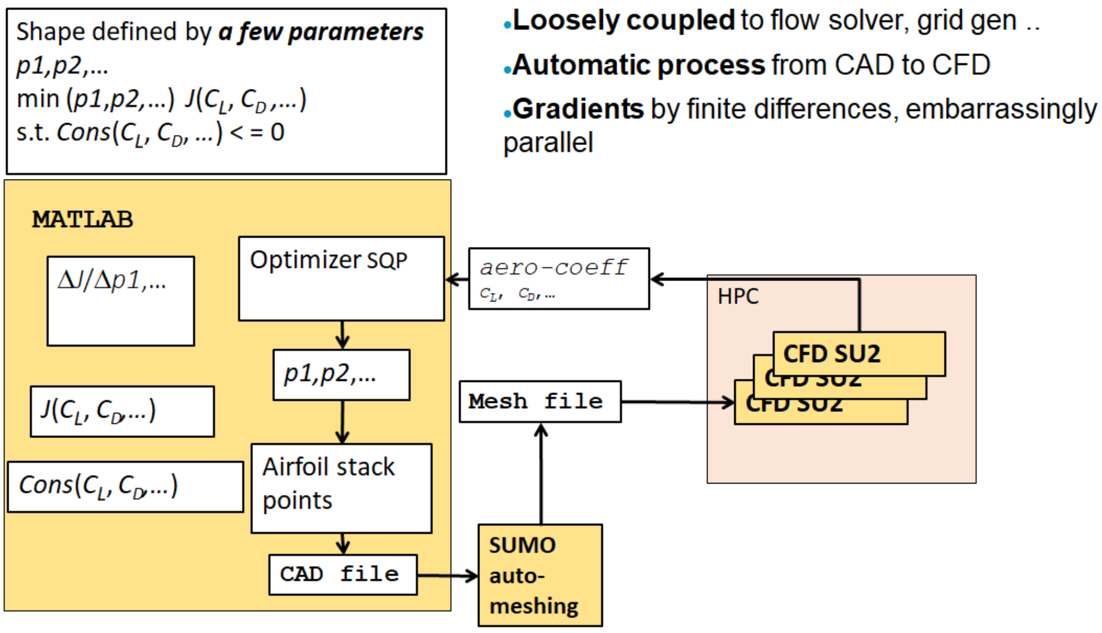
(b) Case 2

Figure 3: Performance results on ARCHER.

4. Results

4.1 Automation work flow within wing shape optimization

An automation workflow for wing shape optimization, which includes the CFD automation process, has been developed. The key idea is to make a *loosely-coupled* system with all the necessary tools, including a flow solver, grid generation and a CAD modeller, see Figure 4. The optimization is carried out using an external optimizer in Python/Matlab. All the modules are linked by scripting in Matlab, Python and Bash shell. Details can be found in [1]. The gradient for the adjoint method of optimization is computed by finite differences using embarrassingly parallel.



Shape defined by ***a few parameters***
p1,p2,...
min (*p1,p2,...*) $J(C_L, C_D,...)$
s.t. $Cons(C_L, C_D, ...) <= 0$
•**Loosely coupled** to flow solver, grid gen ..
•**Automatic process** from CAD to CFD
•**Gradients** by finite differences, embarrassingly parallel
MATLAB
$\Delta J/\Delta p1,...$
Optimizer SQP
aero-coeff
$C_L, C_D, ...$
HPC
CFD SU2
p1,p2,...
$J(C_L, C_D,...)$
Mesh file
$Cons(C_L, C_D,...)$
Airfoil stack points
CAD file
SUMO auto-meshing

Figure 4: The automation work flow within the wing shape optimization scope.

4.2 CRM wing optimization results

The CRM wing optimization task follows the requirements given in AIAA ADODG (Aerodynamic Design Optimization Discussion Group), namely, the drag is optimized at Mach = 0.85, CL=0.85 and with chord Reynolds number 5,000,000, with the thickness constraints that the thicknesses of the optimized wing must not be thinner than the baseline. The turbulence model that is used is the Spalart-Allmaras (S-A) model and the mesh size is 28 million cells.

Figure 5 shows the streamlines, sectional pressure contours and wakes of the simulated baseline wing. Figure 6 shows the wing shape comparison (front view) and the Mach contours between the baseline and the optimized wing. It can be seen that the baseline wing has a stronger shock and the optimized wing has double reduced shocks. Note that the lambda shock raises on the leading edge of the optimized wing, which is favourable from the aerodynamic point of view. However, as the thickness of the optimized wing is increased, the likelihood of the formation of the shockwaves grows. The optimized wing has a drag count reduction of 20 compared to the baseline wing.

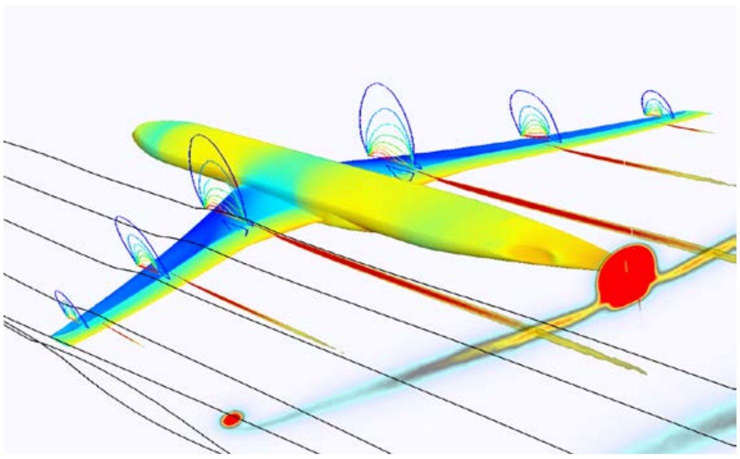


Figure 5: Streamlines, sectional pressure contours and wakes of the CRM baseline wing, SU2 S-A model, CL=0.5, Mach =0.85, chord Reynolds number 5,000,000.

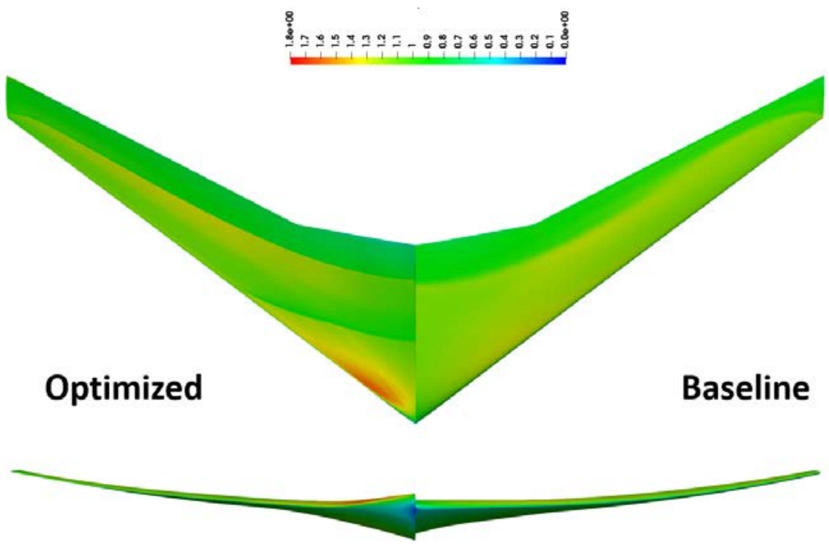


Figure 6: Shape and Mach contour comparison between the CRM baseline wing and the optimized wing, SU2 S-A model, CL=0.5, Mach =0.85, chord Reynolds number 5,000,000

4.3 A320-alike: Design, Simulation and Aero-Data Generation

The A320-alike aircraft is a prototype regional jet-liner without any existing experimental data as yet. It is being designed “virtually” and the aero-data is being generated by computer simulations in order to “fly” in flight simulation software. The initial design is only a wing, a fuselage and tails, without any engine.

The flight conditions that are simulated with the cruise conditions: Mach 0.78, lift coefficient 0.47, and Reynolds number 3.7 million. The simulation was performed on Beskow using the SU2 S-A turbulence model.

The image on the left in Figure 7 shows the RANS mesh of the A320-alike (without any engine) with automatical mesh generation algorithm using *SUMO*. A more realistic design was made by shifting towards a complete configuration through adding pylons and nacelles. The net thrust is given as 10800 Newton per engine. Such a mesh was made by *Pointwise* with the same gridding density as in *SUMO*, with a mesh size of 18 million cells, see the image on the right in Figure 7.

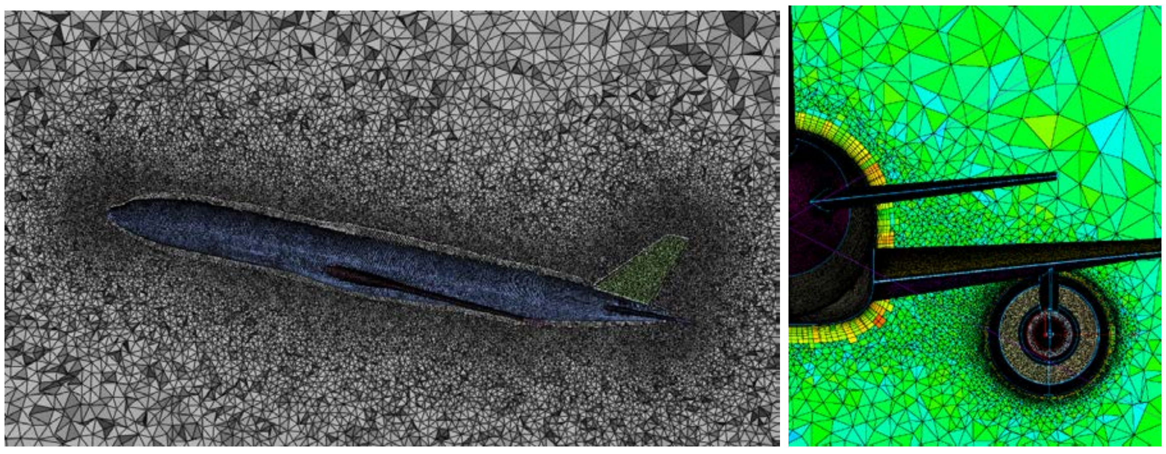


Figure 7: RANS mesh of the A320-alike aircraft, Left: without engine, generated by *SUMO* via automatic meshing scripts. Right: with engine nacelles modelled, generated by *Pointwise* via automatic meshing scripts.

Figure 8 shows the jet flow visualization and turbulence wake around the A320-alike aircraft. The surface of the complete aircraft configuration is contoured by the Pressure Coefficient (CP). Note that a shock is formed around the 75% chord of the main wing, as well as the nacelle inlet. This results in wave drag, which is one of the most important components of drag that designers aim to reduce when developing new aircraft profiles.

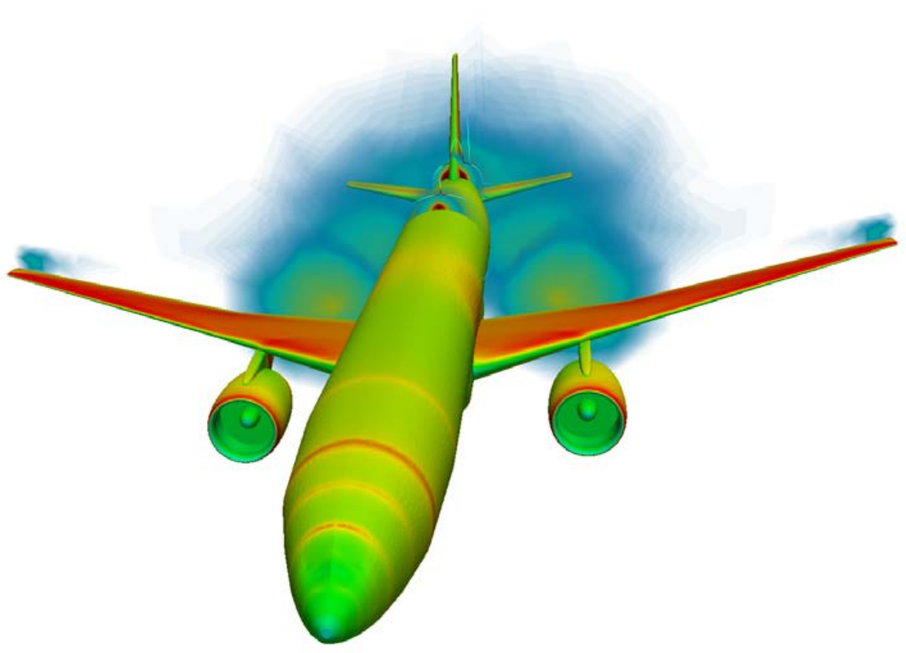


Figure 8: Visualized turbulence wake

4. Conclusions and further work

In the project, we have developed an automation process to run high-fidelity CFD analysis by starting from given geometry format. An abstract regarding this project was submitted to a conference, and additional computing time on the local PRACE Tier-1 resource Beskow at PDC-KTH was granted. Airinnova will subsequently apply for a PRACE Access Type-A project, which follows up on the work that was undertaken during this project and take advantage of both the optimization results that were produced and the existing scripts. Together with the main developers of SU2, we will investigate the non-block or one-side MPI communication for point-to-point communication, which is the main bottleneck for large scale simulations in the current version of the code. The ultimate technical goal of this ongoing research is to develop a flexible and smooth integration of the whole automation process into an MDO (Multi-Disciplinary Optimization) design environment.

References

[1] Mengmeng Zhang, *Contributions to Variable Fidelity MDO Framework for Collaborative and Integrated Aircraft Design*, Doctoral Thesis, Royal Institute of Technology KTH, Stockholm, Sweden, 2015.

[2] Tomac M., *Towards Automated CFD for Engineering Methods in Aircraft Design*, Stockholm, Sweden, Royal Institute of Technology, KTH, 2014, ISSN 1651-7660.

[3] Arthur Rizzi, Computational Aerodynamics in Aircraft Design, TRITA-AVE 2008:06, ISSN 1651-7660, KTH Aeronautical and Vehicle Engineering, Stockholm, Sweden 2008

[4] SU2 the open-source CFD code, http://su2.stanford.edu

[5] Francisco Palacios, Thomas D. Economon, Andrew D. Wendorff, and Juan J. Alonso, Large-scale aircraft design using SU2, AIAA SciTech 5-9 January 2015, Kissimmee, Florida, 53rd AIAA Aerospace Sciences Meeting.

[6] Thomas D. Economon, Dheevatsa Mudigereb, Gaurav Bansalc, Alexander Heinecked, Francisco Palaciose, Jongsoo Parkd, Mikhail Smelyanskiyd, Juan J. Alonsoa, Pradeep Dubeyd, *Performance optimizations for scalable implicit RANS calculations with SU2*, Computers & Fluids, Vol.129(28), 2016, pp. 146—158

[7] Thomas D. Economon, Francisco Palacios, Sean R. Copeland, Trent W. Lukaczyk, and Juan J. Alonso. *SU2: An Open-Source Suite for Multiphysics Simulation and Design*, AIAA Journal, Vol. 54, No. 3 (2016), pp. 828-846.

[8] CrayPat, XC Series Cray Performance Measurement and Analysis Tool User Guide, https://pubs.cray.com/content/S-2376/6.4.3/xc-series-cray-performance-measurement-and-analysis-tools-user-guide-643-s-2376/craypat

[9] ARM MAP, https://www.arm.com/products/development-tools/server-and-hpc/forge/map

[10] Archer, https://www.epcc.ed.ac.uk/facilities/archer

[11] Beskow, https://www.pdc.kth.se/resources/computers/beskow

[12] Nicolas Offermans, Oana Marin, Michel Schanen, Jing Gong, Paul Fischer, and Philpp Schlatter, On the Strong Scaling of the Spectral Element Solver Nek5000 on Petascale system. In the proceeding of EASC'16, April 26-29, 2016, Stockholm, Sweden.

Acknowledgements

This work was financially supported by the PRACE project funded in part by the EU's Horizon 2020 research and innovation programme (2014-2020) under grant agreement 730913. The work was done using the PRACE research infrastructure Tier-1 resources Archer at EPCC, UK, and Beskow at KTH-PDC, Sweden. The support for the technical work via the PRACE type D programme is also gratefully acknowledged. Drs. Thomas Economon and Francisco Palacios are thankful for full discussion.