Assessment of a heterogeneous computing CFD code in wind farm simulations

Assessment of a heterogeneous computing CFD code in wind farm simulations

B López[1], A Guggeri[1], M Draper[1] and G Usera[1]

[1] Facultad de Ingeniería, Universidad de la República, Julio Herrera y Reissig 565, Montevideo, Uruguay

E-mail: brunolop@fing.edu.uy

Abstract. The use of heterogeneous architectures, based on CPU-GPU processors, has led to a significant increase in the performance of parallel computing applications. In recent years, this approach has been implemented in various computational fluid dynamics (CFD) codes to take advantage of the compute capability of the GPU graphics cards. The objective of this work is to assess the performance of a general purpose CFD open-source code in wind energy applications, running in a heterogeneous architecture. To this aim, a numerical wind turbine model was migrated from a CPU-based Fortran program to the CFD code. Several timing tests were performed on a local computing station, while running simulations of well-documented wind tunnel experiments. The results obtained show a significant reduction in computational time and resource required, indicating a great potential of the GPU-accelerated CFD code to be used in large wind farms simulations or in real-time applications.

1. Introduction

In the last decade, the introduction of heterogeneous architectures for parallel computing programs allowed a considerable increase of performance in High-Performance Computing (HPC), compared to parallel applications executed on homogenous architectures, based on Central Processing Units (CPU). At present, the most extended heterogeneous computing platforms combine CPUs with Graphics Processing Units (GPU), which have been generalized for data-parallel computation-intensive tasks [1].

Recently, scientists have assessed the use of CPU-GPU architectures for solving Computational Fluid Dynamics (CFD) problems, obtaining significant improvement in the computational performance of the codes employed [2–4]. These works also show that the redesign of code programming strategies is needed to maximize the benefits of the GPU architecture. The exploration of heterogeneous computing in wind energy applications is incipient. Wind turbines simulations using the lattice Boltzmann method performed on GPU architecture are presented in [5]. The turbines rotor is represented using the Actuator Line Model [6]. The GPU accelerated code used in this work achieved simulations performance close to real time. In [7] control vibration strategies in onshore wind turbines are analysed using an aeroelastic model developed for GPU architecture, using Python and CUDA C programming languages. These strategies are based on mass damper technics which are tuned using the GPU accelerated code, through massive parallel computation. The wind speed time series are generated using LES simulations and taken as input for the aeroelastic model to compute the equivalent fatigue loads.

The overall objective of this paper is to evaluate the performance of an open-source general purpose CFD code in wind energy applications, running on CPU-GPU architectures. This code is called Chamán (spanish word for Shaman), and had its inception in a recent migration from in-house open-source code caffa3d [8,9] to a heterogeneous architecture based on CPU and GPU. As a result, the new code combines the use of GPU and the communication protocol Message Passing Interface (MPI) to handle different levels of parallelisms and benefit from massive parallel computing platforms. More details of Chamán are given in [4], where it was used to analyse the pollution dispersion in an urban environment.

In the context of this work, further code migration to Chamán had to be done, i.e. the Fortran 90 programming modules of caffa3d oriented to wind turbines simulations, were extended with CUDA C Kernels for GPU computations. Since Chamán allows the possibility of executing the code using a heterogeneous GPU-CPU infrastructure or CPUs only, the speed-up evaluation was performed through a performance comparison between these two modalities, using local computational resources.

The description of wind turbine modules migration and the results of performance comparison are presented in following sections.

2. Methodology

2.1. CFD code: turbine model migration

The Chamán code implements the finite volume method for solving incompressible flows. The discretisation strategy follows a fully implicit scheme with second order accuracy in both space and time. It features a Large Eddy Simulations (LES) approach to simulate turbulent flows. The sub-grid scale model employed in this work is the standard Smagorinsky model [10]. The representation of wind tubrine generators in the computational domain is done using the Actuator Line Model (ALM) [6]. The implementation of the ALM in caffa3d code and its validation has been described in several works [11–14].

Within the ALM model, each rotor blade is discretised into several nodes. At each node, and at each time step, a local wind speed is calculated by interpolation in the neighbouring grid cells. From this local wind speed, the Lift and Drag coefficients (C_L and C_D, respectively) are computed at each blade node, using the airfoil data given in [15]. At this step, Prandtl's tip loss correction factor is applied. These coefficients are used to calculate the aerodynamic forces along each blade. To project these forces in the computational domain, a Gaussian smearing function is used to smoothly distribute the forces and thus avoid the occurrence of numerical instabilities [11]. This smearing function is limited to a cylindrical volume confined within the rotor disc, as it has been shown to improve force distribution near the blade tip [16]. The distributed forces are finally used to compute additional source term in the Navier-Stokes equations. The wind turbine nacelle and tower are represented in the code through drag coefficients, using a 3D smearing Gaussian function to project the forces in the computational domain. For more details of ALM implementation in caffa3d, please see [11].

There are some critical time-consuming steps in this implementation that can be accelerated in Chamán using the CPU-GPU strategy: the indexing of neighbouring grid cells of each blade node (at each time step) and subsequent wind speed interpolation, as well as the calculation of smearing weighting factors and force distribution. The optimisation of these code instances is based on replacing their associated loops in the original Fortran 90 code with GPU kernel routines, which are executed by multiple processors at the same time on the GPU board.

Integration between the Fortran 90 code running on the CPU and the CUDA code for the GPU kernels is achieved through extensive use of of the Fortran 90 ISO-C-Binding module [4]. The parallelisation of Chamán execution between CPU processes is based on block-structured meshing (domain decomposition), and is performed using the MPI library, in the same manner as caffa3d. Finally, the compilation of the code is done through a hybrid Makefile, which is

configured to produce two independent versions of the solver, one that computes only on a CPU basis, and another one that includes the use of GPUs for computation. The performance of these two versions is compared in this paper to assess the gain in speed obtained on a heterogeneous CPU-GPU system.

2.2. Simulations setup

The performance comparison of Chamán using only CPU and CPU-GPU processors is conducted by simulating a base case that includes the presence of wind turbines. For this purpose, a test case has been selected from the blind test workshop [17]. This blind test was considered suitable for performance testing as it has a well-documented data set from experimental measurements, under controlled wind conditions and wind turbines operation, in a closed-loop wind tunnel. Furthermore, reasonable experience has been previously obtained employing caffa3d to simulate wind turbines operation in wind tunnels [11, 12, 14].

The experimental data are obtained from wind tunnel tests of two scaled three-bladed wind turbines aligned with the wind flow direction. The test section is 2.71 m wide, 1.81 m high and 11.15 m long. The test case selected is identified as case B_2 in [17]. According to this case, the upstream wind turbine, which has a diameter of $D_1 = 0.944$ m and a hub height of $H_1 = 0.827$ m, is located at a distance of $x_1 = 1.788$ m ($2D_2$) from the inlet section. On the other hand, the downstream wind turbine has a diameter of $D_2 = 0.894$, a hub height of $H_1 = 0.827$ m, and is located at $x_2 = 6.418$ m ($5.18D_2$ from the upstream wind turbine). Both turbines have same blade geometry, which is specified in [17].

The wind inflow conditions is generated by means of a turbulence grid located at the inlet section. The grid is divided in squares of 0.240 m side. The wooden bars that define the grid have a cross section size of 47 mm by 47 mm. This configuration gives the grid a solidity of 35%. Using this turbulence grid in an empty wind tunnel results in a turbulence intensity of TI = 10.0% at the position of the upstream turbine ($x_1 = 1.788$ m). As the flow reaches the position $x = 4.77D_2$ from the inlet section, the turbulence intensity in the empty tunnel drops to TI = 4.8%. The mean wind speed at the inlet section is $U_{ref} = 11.5$ m/s.

In the test case considered (B_2), the upstream wind turbine is operated with a constant tip speed ratio $TSR_1 = 0.6$ and the downstream turbine with $TSR_2 = 4.5$, taking U_{ref} as reference for both turbines. This means that the rotor speed of the upstream and downstream turbines are 1396 RPM and 1106 RPM, respectively.

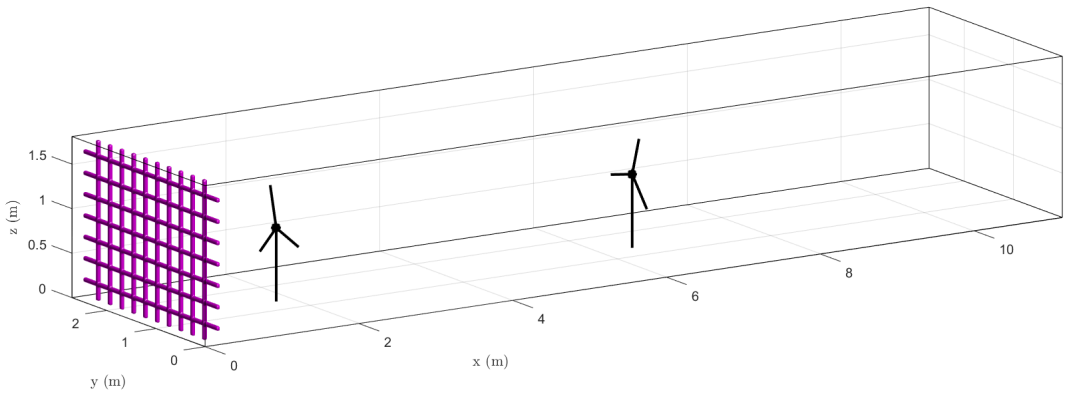

Figure 1. Setup of computational domain simulated with Chamán for perfomance evaluation. The setup corresponds to test case B_2 of the blind test workshop [17]

As for the computational domain, it has the same dimensions as the wind tunnel test section and is uniformly divided into 16 million mesh cells, each 21.7 mm long, 10.6 mm wide and 14.1

mm high. This resolution covers the turbine rotor with 64 cells in the vertical direction and 84 cells in the spanwise direction. The time step of the simulations is set to 0.001 s.

The numerical inlet boundary conditions are modelled with a uniform flow at U_{ref} velocity superimposed with small disturbances (TI = 0.2%), which are generated synthetically following the Mann turbulence approach ([18]). Instead of explicitly solving the turbulence grid, it is modelled using the immersed boundary method [19]. This method applies a force field at the points of the computational mesh to obtain a velocity field consistent with the presence of objects immersed in the flow. The turbulence grid is located $2D_1$ upstream the first wind turbine model.

The setup of the computational domain is represented in Figure 1.

2.3. Computing resources

The computational load is distributed in two processes, using one core each. This means that each process handles the calculations of eight million cells. In simulations where GPU computing is enabled, each core execute the most time-consuming instructions on the GPU card (one card per core was used).

All simulations were performed at a local computing station, provided with an Intel Core i7-6700K CPU processor and four NVIDIA GeForce GTX 980 Ti graphics cards. The processor has a base frequency of 4.00 GHz and four physical cores. Its theoretical power consumption, specified in terms of Thermal Design Power (TDP), is 91 W. On the other hand, each graphic card has 6 GB of memory and 2816 CUDA Cores with a base clock of 1000 MHz. The power consumption of each is specified as 250 W.

In order to measure the performance of the simulations, the time at the beginning and end of each run is recorded.

3. Results

Chamán code is first calibrated to match the wind conditions in the empty wind tunnel. As previously stated, in test case B_2 the turbulent flow reaches the position of the upstream wind turbine (x_1) with TI = 10%. At the downstream wind turbine position (x_2), the turbulence intensity decays to TI = 4.8%. Once calibration is achieved, the wind turbines are modelled in the simulations.

The mean velocity in the wake at position $x = 4.264$ m ($4.77D_2$) from the inlet section was measured in the wind tunnel using a single hot-wire anemometer at a frecuency of 20 kHz for 45 s. The results obtained with Chamán are evaluated at the same position, giving a good overall agreement with the corresponding experimental values. The simulation results are averaged over a period of 10 s (10.000 time steps). The comparison of the results of these two approaches is shown in Figure 2. An additional wake profile has been included, which corresponds to the results obtained with the DTU model, during the blind test workshop. This model consists of the CFD code EllipSys3D, which uses a finite volume approach and LES models to solve the Navier-Stokes equations, combined with the ALM method to represent the turbines rotor. The convective terms are discretised following a combination of third and fourth order schemes. The inlet boundary condition used by this model is a pre-generated synthetic turbulence.

The power and thrust coefficients, C_P and C_T respectively, were also measured for both wind turbines during the experiments. These are given by equations 1 and 2, as follows:

$$C_P = \frac{2P}{\rho U_{ref}^3 A} \tag{1}$$

$$C_T = \frac{2T}{\rho U_{ref}^2 A} \tag{2}$$

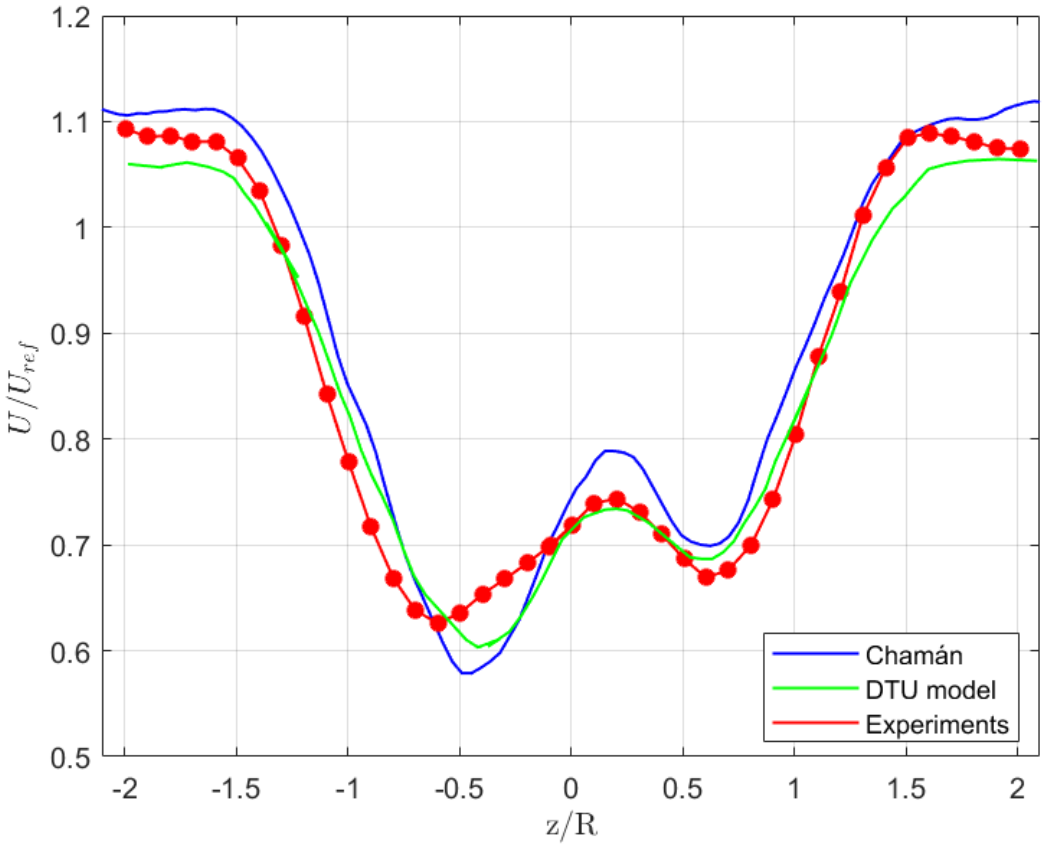

Figure 2. Wake profile obtained at position $x = 4.264$ m $(4.77D_2)$ from the inlet section. The results obtained with Chamán are compared with the experimental test as well as with DTU CFD model described in [17].

Where P is the mechanical power determined at the turbine shaft, while T is , and A the rotor swept area. The Table 1 summarises the results obtained in the wind tunnel experiments and with both CFD models, corresponding to test case B_2, in terms of C_P and C_T measured at each wind turbine. As it is shown in the table, the errors incurred by the cfd models with respect to the experimental results are of the same order.

Table 1. Power (C_P) and thrust (C_T) coefficients comparison between wind tunnel experiments and CFD models (test case B_2).

Model	$C_{P,T1}$	$C_{T,T1}$	$C_{P,T2}$	$C_{T,T2}$
Chamán	0.488	0.704	0.227	0.413
DTU model	0.447	0.758	0.152	0.423
Experiment	0.468	0.833	0.188	0.500

No code profiling was performed during the simulations. Instead, time was recorded at the beginning and end of each programme execution. Running 10000 time steps using only the CPU processor took an average time of 1700 minutes (about 28.3 hours), while the time spent on the

heterogeneous CPU-GPU architecture was about 153 minutes (2.5 hours). This represents a considerable increase in code speed of approximately 11.1 times.

4. Conclusions

A successful code migration was achieved, which allowed for a significant reduction in computational time and resource requirements, in the context of wind turbines simulations. The results obtained in this work indicate a great potential of the GPU accelerated CFD code to be used in large wind farms simulations or in real-time applications. The simulations were performed in a local computing station, equipped with affordable CPU and GPU components.

In future work, we intend to perform a detailed profiling of the wind module migrated in this work, in order to evaluate the possibilities of obtaining additional speedups. Actual power consumption will also be analysed, allowing a more comprehensive analysis of the suitability of using heterogeneous CPU-GPU platforms. Finally, the advantages suggested by the results of this work will be tested in full-scale wind farm simulations. It is expected that the higher the number of turbines simulated, the higher the speedup potential.

References

[1] Cheng J, Grossman M and McKercher T 2014 *Professional CUDA c programming* (John Wiley & Sons)
[2] Niemeyer K E and Sung C J 2014 *The Journal of Supercomputing* **67** 528–564
[3] Afzal A, Ansari Z, Faizabadi A R and Ramis M 2017 *Archives of Computational Methods in Engineering* **24** 337–363
[4] Fernandez G, Mendina M and Usera G 2020 *Computation* **8** 3
[5] Asmuth H, Olivares-Espinosa H and Ivanell S 2020 *Wind Energy Science* **5** 623–645
[6] Sorensen J N and Shen W Z 2002 *J. Fluids Eng.* **124** 393–399
[7] Liu Z, Wang Y, Nyangi P, Zhu Z and Hua X 2021 *Renewable Energy* **168** 516–543
[8] Usera G, Vernet A and Ferré J 2008 *Flow, Turbulence and Combustion* **81** 471
[9] Mendina M, Draper M, Soares A P K, Narancio G and Usera G 2014 *Cluster Computing* **17** 231–241
[10] Smagorinsky J 1963 *Monthly weather review* **91** 99–164
[11] Draper M and Usera G 2015 Evaluation of the actuator line model with coarse resolutions *Journal of Physics: Conference Series* vol 625 (IOP Publishing) p 012021
[12] Draper M, Guggeri A, López B, Díaz A, Campagnolo F and Usera G 2018 A large eddy simulation framework to assess wind farm power maximization strategies: Validation of maximization by yawing *Journal of Physics: Conference Series* vol 1037 (IOP Publishing) p 072051
[13] Guggeri A and Draper M 2019 *Energies* **12** 3508
[14] Mühle F, Schottler J, Bartl J, Futrzynski R, Evans S, Bernini L, Schito P, Draper M, Guggeri A, Kleusberg E *et al.* 2018 *Wind Energy Science* **3** 883–903
[15] Sætran L and Bartl J 2015 *Technical document NTNU* **5**
[16] Draper M, López B, Guggeri A, Campagnolo F and Usera G 2020 Influence of limiting the projection region on coarse large eddy simulation-actuator line model simulations *Journal of Physics: Conference Series* vol 1618 (IOP Publishing) p 022051
[17] Bartl J and Sætran L 2017 *Wind energy science* **2** 55–76
[18] Mann J 1998 *Probabilistic engineering mechanics* **13** 269–282
[19] Peskin C S 1972 *Journal of computational physics* **10** 252–271