AI-accelerated CFD simulation based on OpenFOAM and CPU/GPU computing*

K. Rojek[1][0000−0002−2635−7345], R. Wyrzykowski[2][0000−0003−1724−1786], and P. Gepner[3][0000−0003−0004−1729]

[1] Czestochowa University of Technology, Czestochowa, Poland
krojek@icis.pcz.pl
[2] Czestochowa University of Technology, Czestochowa, Poland
roman@icis.pcz.pl
[3] Warsaw University of Technology
pawel.gepner@gmail.com

Abstract. In this paper, we propose a method for accelerating CFD (computational fluid dynamics) simulations by integrating a conventional CFD solver with our AI module. The investigated phenomenon is responsible for chemical mixing. The considered CFD simulations belong to a group of steady-state simulations and utilize the MixIT tool, which is based on the OpenFOAM toolbox. The proposed module is implemented as a CNN (convolutional neural network) supervised learning algorithm. Our method distributes the data by creating a separate AI sub-model for each quantity of the simulated phenomenon. These sub-models can then be pipelined during the inference stage to reduce the execution time or called one-by-one to reduce memory requirements.
We examine the performance of the proposed method depending on the usage of the CPU or GPU platforms. For test experiments with varying quantities conditions, we achieve time-to-solution reductions around a factor of 10. Comparing simulation results based on the histogram comparison method shows the average accuracy for all the quantities around 92%.

Keywords: AI acceleration for CFD · convolutional neural networks · chemical mixing · 3D grids· OpenFOAM · MixIT · CPU/GPU computing.

1 Introduction

Machine learning and artificial intelligence (AI) methods have become pervasive in recent years due to numerous algorithmic advances and the accessibility of computational power [1, 7]. In computational fluid dynamics (CFD), these methods have been used to replace, accelerate or enhance existing solvers [13]. In this

* The authors are grateful to the byteLAKE company for their substantive support. We also thank Valerio Rizzo and Robert Daigle from Lenovo Data Center and Andrzej Jankowski from Intel for their support.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_29

work, we focus on the AI-based acceleration of a CFD tool used for chemical mixing simulations.

Chemical mixing is a critical process used in various industries, such as pharmaceutical, cosmetic, food, mineral, and plastic ones. It can include dry blending, emulsification, particle size reduction, paste mixing, and homogenization to achieve your desired custom blend [6].

We propose a collection of domain-specific AI models and a method of integrating them with the stirred tank mixing analysis tool called MixIT. MixIT [11] provides deep insights solutions to solve scale-up and troubleshooting problems. The tool utilizes the OpenFOAM toolbox [14] for meshing, simulation, and data generation. It allows users to design, simulate and visualize phenomena of chemical mixing. More detailed, MixIT provides geometry creation, performs 3-dimensional (3D) CFD flow simulations for stirred reactors, including tracer simulations and heat transfer analysis. Moreover, it allows you to get performance parameters, such as mixing intensity, power per unit volume, blend time, critical suspension speed, gas hold-up, and mass transfer coefficients.

Our goal is to provide an interaction between AI and CFD solvers for much faster analysis and reduced cost of trial & error experiments. The scope of our research includes steady-state simulations, which use an iterative scheme to progress to convergence. Steady-state models perform a mass and energy balance of a process in an equilibrium state, independent of time [2]. In other words, we assume that a solver calculates a set of iterations to achieve the convergence state of the simulated phenomenon. Whence, our method is responsible for predicting the convergence state with the AI models based on a few initial iterations generated by the CFD solver. In this way, we do not need to calculate intermediate iterations to produce the final result, so the time-to-solution is significantly reduced. The proposed AI models make it possible to run many more experiments and better explore the design space before decisions are made.

The contributions of this work are as follows:

- AI-based method that is integrated with a CFD solver and significantly reduces the simulation process by predicting the convergence state of simulation based on initial iterations generated by the CFD solver;
- method of AI integration with the MixIT tool that supports complex simulations with size of ≈ 1 million cells based on the OpenFOAM toolbox and high performance computing with both CPUs and graphic processing units (GPUs);
- performance and accuracy analysis of the AI-accelerated simulations.

2 Related Work

Acceleration of CFD simulations is a long-standing problem in many application domains, from industrial applications to fluid effects for computer graphics and animation.

Many papers are focused on the adaptation of CFD codes to hardware architectures exploring modern compute accelerators such as GPU [12, 15, 16], Intel

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_29

Xeon Phi [20] or field-programmable gate array (FPGA) [17]. Building a simulator can entail years of engineering effort and often must trade-off generality for accuracy in a narrow range of settings. Among the main disadvantages of such approaches are requirements of in-depth knowledge about complex and extensive CFD codes, expensive and long-term process of portability across new hardware platforms, and, as a result, relatively low-performance improvement compared with the original CFD solver. In many cases, only a small kernel of the solver is optimized.

Recent works have addressed increasing computational performance of CFD simulations by implementing generalized AI models able to simulate various use cases and geometries of simulations [10, 13]. It gives the opportunity of achieving lower cost of trial & error experiments, faster prototyping, and parametrization. Current AI frameworks support multiple computing platforms that provide code portability with minimum additional effort.

More recently - and most related to this work - some authors have regarded the fluid simulation process as a supervised regression problem. In [8], the authors present a novel generative model to synthesize fluid simulations from a set of reduced parameters. A convolutional neural network (CNN) is trained on a collection of discrete, parameterizable fluid simulation velocity fields.

In work [22], J. Thompson et al. propose a data-driven approach that leverages the approximation of deep learning to obtain fast and highly realistic simulations. They use a CNN with a highly tailored architecture to solve the linear system. The authors rephrase the learning task as an unsupervised learning problem. The key contribution is to incorporate loss training information from multiple time-steps and perform various forms of data-augmentation.

In paper [8], the authors show that linear functions are less efficient than their non-linear counterparts. In this sense, deep generative models implemented by CNNs show promise for representing data in reduced dimensions due to their capability to tailor non-linear functions to input data.

Work [10] introduces a machine learning framework for the acceleration of Reynolds-averaged Navier-Stokes to predict steady-state turbulent eddy viscosities, given the initial conditions. As a result, they proposed a framework that is hybridized with machine learning.

In [18], the authors present a general framework for learning simulation and give a single model implementation that yields state-of-the-art performance across a variety of challenging physical domains, involving fluids, rigid solids, and deformable materials interacting with one another.

Our method for AI-accelerated CFD simulations is based on utilizing a set of sub-models that are separately trained for each simulated quantity. This approach allows to reduce the memory requirements and operate on large CFD meshes. The proposed data-driven approach provides a low entry barrier for future researchers since the method can be easily tuned when the CFD solver evolves.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_29

3 Simulation of Chemical Mixing with MixIT tool

3.1 MixIT: simulation tool based on OpenFOAM

MixIT [11] is the next generation collaborative mixing analysis and scale-up tool designed to facilitate comprehensive stirred tank analysis using lab and plant data, empirical correlations, and advanced 3D CFD models. It combines knowledge management techniques and mixing engineering (science) in a unified environment deployable enterprise-wide.

This tool allows users to solve Euler-Lagrange simulations [9] and momentum transfer from the bubbles to the liquid. The liquid flow is described with the incompressible Reynolds-averaged Navier-Stokes equations using the standard k-ϵ model.

The generation of 3D grids is performed with the OpenFOAM meshing tool *snappyHexMesh* [9]. For Euler-Lagrange simulations, a separate grid for each working volume is created using the preconfigured settings of MixIT. A mesh of the complete domain is generated, and the working volume is defined by the initial condition of the gas volume fraction with the OpenFOAM toolbox.

3.2 Using MixIT tool for simulation of chemical mixing

The chemical mixing simulation is based on the standard k-ϵ model. The goal is to compute the converged state of the liquid mixture in a tank equipped with a single impeller and a set of baffles (Fig. 1). Based on different settings of the input parameters, we simulate a set of quantities, including the velocity vector field U, pressure scalar field p, turbulent kinetic energy k of the substance, turbulent dynamic viscosity mut, and turbulent kinetic energy dissipation rate ϵ.

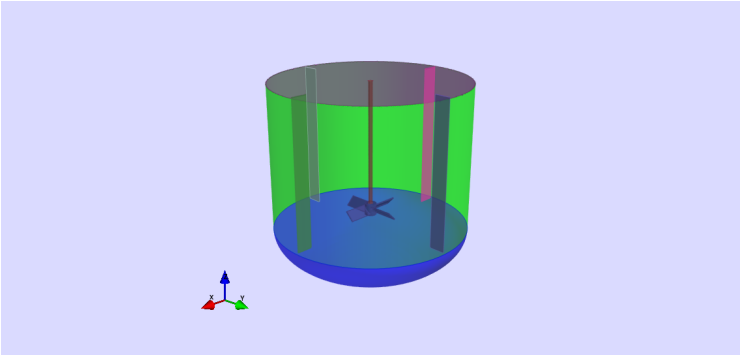


Fig. 1: Scheme of the simulated phenomenon

To simplify the simulation process, we have selected a subset of parameters responsible for the simulation flow. The CFD solver supports many scenarios; however, our research includes three basic case studies:

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_29

– assuming the different liquid level of a mixed substance,
– exploring the full range of rotations per minutes (rpm) of the impeller,
– considering different viscosities of the mixed substance.

4 AI-based acceleration

4.1 Introduction of AI into simulation workflow

Conventional modeling with OpenFOAM involves multiple steps (Fig. 2a). The first step includes pre-processing, where you need to create the geometry and meshing. This step is often carried out with other tools. The next step is the simulation. It is the part that we mainly focus on in this paper by providing the AI-based acceleration. The third step is post-processing (visualization, result analysis).

Our goal is to create solver-specific AI method to ensure the high accuracy of predictions. Our approach belongs to a group of data-driven methods, where we use partial results returned by the CFD solver. The advantage of this method is that it does not require to take into account a complex structure of the simulation, but focus on the data. Such an approach lowers the entry barrier for new CFD adopters compared with other methods, such as a learning-aware approach [5], which is based on the mathematical analysis of solver equations.

Fig. 2b presents the general scheme of the AI-accelerated simulation versus the conventional non-AI simulation. It includes (i) the initial results computed by the CFD solver and (ii) the AI-accelerated part executed by the proposed AI module. The CFD solver produces results sequentially iteration by iteration, where each iteration produces intermediate results of the simulation. All intermediate results wrap up into what is called the simulation results. The proposed method takes a set of initial iterations as an input, sends them to our AI module, and generates the final iteration of the simulation. The AI module consists of three stages: (i) data formatting and normalization, (ii) prediction with AI model (inference), and (iii) data export.

Data formatting and normalization translate the data from the OpenFOAM ASCII format to the set of arrays, where each array stores a respective quantity of

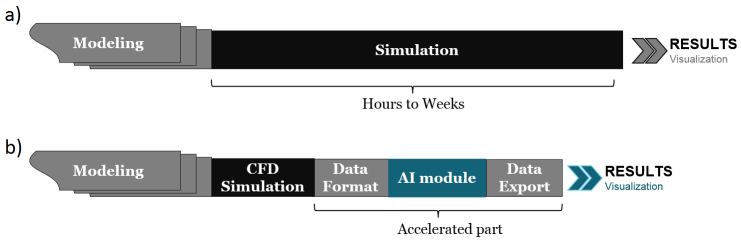


Fig. 2: Scheme of AI-accelerated simulation (b) versus conventional non-AI approach (a)

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_29

the simulation (U, p, ϵ, mut, and k). These quantities are normalized depending on a user configuration. The linear normalization is performed based on the following equation:

$$y_i = x_i/max(|maxV|, |minV|) \cdot R, \tag{1}$$

where y_i is the normalized value, x_i is the input value, $maxV$, $minV$ are the maximum and minimum values from all the initial iterations of a given quantity, R is a radius value (in our experiments $R = 1$). When a dataset has a relatively small value of median compared to the maximum value of the set (median value is about 1% of the maximum), then we use a cube normalization with $y_i^* = y_i^3$.

The AI-accelerated simulation is based on supervised learning, where a set of initial iterations is taken as an input and returns the last iteration. For simulating the selected phenomenon with MixIT and conventional non-AI approach, it is required to execute 5000 iterations. At the same time, only the first N_I iterations create the initial iterations that produce input data for the AI module. Moreover, to reduce the memory requirements, the initial dataset is composed of simulation results corresponding to every S_I-th iteration from the N_I initial iterations. The determination of parameters N_I and S_I is the subject of our future work. At this moment, we have empirically selected them by training the AI model with different values of the parameters, up to achieving an acceptable accuracy. For the analyzed phenomenon, we set $N_I = 480$, and $S_I = 20$. As a result, we take the initial data set composed from iterations $20, 40, 60, ..., 480$, so $480/5000 = 0.096$ of the CFD simulation has to be executed to start the inference.

The data export stage includes denormalization of the data and converting them to the OpenFOAM ASCII format. This stage and data formatting one are executed on the CPU, but the prediction with AI model (inference) can be executed on either the CPU or GPU, depending on user preferences.

4.2 Idea of using AI for accelerating simulation

Our neural network is based on the ResNet network [3] organized as residual blocks. In a network with residual blocks, each layer feeds into the next layer and directly into the layers about two hops away. To handle relatively large meshes (about 1 million cells), we have to reduce the original ResNet network to 6 CNN layers.

To train the network, we use 90% of the total data set, referred to as the training data. The remaining 10% are kept as the validation data for model selection, allowing detection of any potential overfitting of the model.

Our AI model is responsible for getting results from 24 simulation iterations (from iterations $20, 40, 60, ..., 480$) as the input, feed the network, and return the final iteration. Each iteration has a constant geometry and processes the 3D mesh with one million cells in our scenario. Moreover, we have five quantities that are taken as the input and returned as the output of the simulation. One of the main challenges here is to store all those data in the memory during the learning. To reduce memory requirements, we create a set of sub-models

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_29

that independently work on a single quantity. Thanks to this approach, all the sub-models are learned sequentially, which significantly reduces memory requirements. This effect is especially important when the learning process is performed on the GPU.

The proposed strategy also impacts the prediction (inference) part. Since we have a set of sub-models, we can predict the result by calling each sub-model one-by-one to reduce the memory requirements or perform pipeline predictions for each quantity and improve the performance. The created pipelines simultaneously call all the sub-models, where each quantity is predicted independently.

In this way, our method can be executed on the GPU platform (in one-by-one mode), or the CPU platform with a large amount of RAM (in a pipelined mode).

5 Experimental evaluation

5.1 Hardware and software platform

All experiments are performed on the Lenovo platform equipped with two Intel Xeon Gold 6148 CPUs clocked at 2.40GHz and two NVIDIA V100 GPUs with 16GB of the global HBM2 memory. The host memory is 400GB.

For training the models, the CUDA framework v.10.1 with cuDNN v.7.6.5 is used. As a machine learning environment, we utilize TensorFlow v.2.3, the Keras API v.2.4, and python v.3.8. The operating system is Ubuntu 20.04 LTS. For the compilation of codes responsible for data formatting and export from/to OpenFOAM, we use GCC v.9.3. The CFD simulations are executed using the OpenFOAM toolbox v.1906 and MixIT v.4.2.

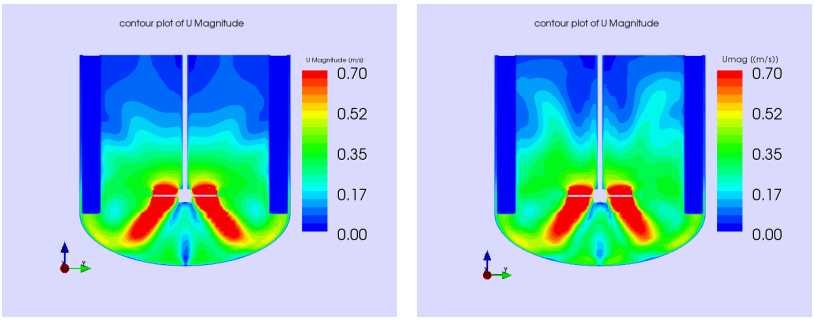
5.2 Performance and accuracy analysis

The first part of the analysis is focused on the accuracy results, while the second one investigates the performance aspects. Since the accuracy evaluation for the regression-based estimation is not so evident as for the classification method, we have selected a set of metrics to validate the achieved results. In the beginning, we compare the contour plots of each quantity created across the XZ cutting plane defined in the center point of the impeller. The results are shown in Figs.3-5, where we can see the converged states computed by the conventional CFD solver (left side) and AI-accelerated approach (right side).

The obtained results show high similarity, especially for the values from the upper bound of the range. Some quantities, such as k, and ϵ have a meager median value (below 1% of the maximum value). As a result, a relatively small error of the prediction impacts the sharp shape of contours for the near-zero values. The higher absolute values, the more smooth shape of the contour plot can be observed.

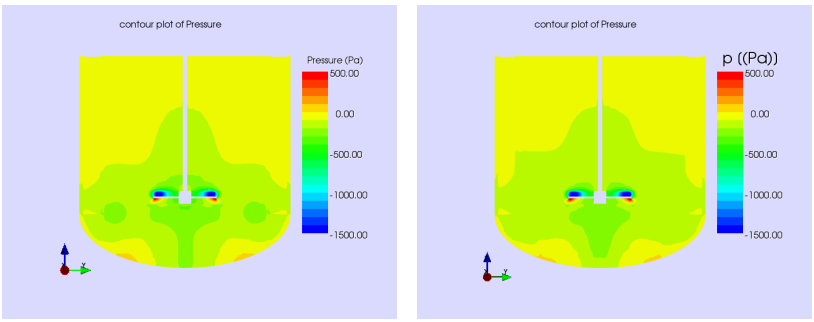
To estimate the accuracy, we also use a set of statistical metrics. The first two ones are correlation coefficients that measure the extent to which two variables tend to change together. These coefficients describe both the strength and the

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_29



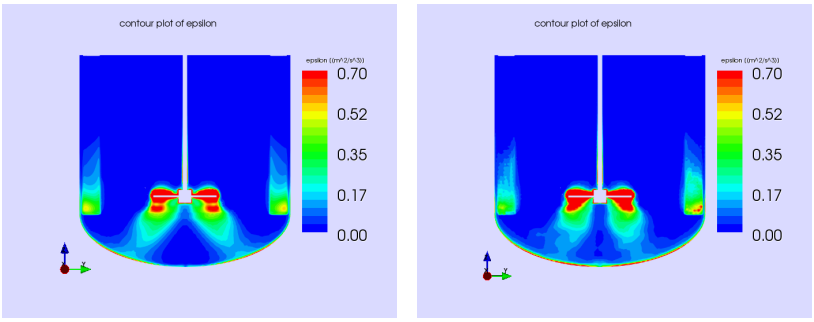
(a) CFD (b) AI

Fig. 3: Contour plot of the velocity magnitude vector field (U) using either the conventional CFD solver (a) or AI-accelerated approach (b)



(a) CFD (b) AI

Fig. 4: Contour plot of the pressure scalar field (p)



(a) CFD (b) AI

Fig. 5: Contour plot of turbulent kinetic energy dissipation rate (ϵ)

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_29

Table 1: Accuracy results with statistical metrics

Quantity	Pearson's coeff.	Spearman's coeff.	RMSE	Histogram equaliz. [%]
U	0.99	0.935	0.016	89.1
p	0.993	0.929	0.004	90.1
ϵ	0.983	0.973	0.023	90.3
k	0.943	0.934	0.036	99.4
mut	0.937	0.919	0.147	93.5
Average	0.969	0.938	0.045	92.5

direction of the relationship. Here, we use two coefficients, including the Pearson correlation that estimates the linear relationship between two continuous variables, as well as the Spearman correlation that assesses the monotonic relationship between two continuous or ordinal variables. The Pearson correlation varies from 0.93 for the mut quantity to 0.99 for U and p. The average Pearson correlation for all the quantities is 0.97. It shows a high degree correlation between the CFD (computed by solver) and AI (predicted) values. The Spearman correlation varies from 0.92 to 0.97 with the average value equal to 0.94. It shows a strong monotonic association between the CFD and AI results.

The next statistical metric is the Root Mean Square Error (RMSE). It is the standard deviation of the residuals (differences between the predicted and observed values). Residuals are a measure of how far from the regression line data points are. The implemented data normalization methods ensure that the maximum distance from the X-axis is 1. RMSE varies from 0.004 for the p quantity to 0.15 for the mut quantity. The average RMSE for all the quantities is 0.05. Based on these results, we can conclude that the proposed AI models are well fit.

The last method of accuracy assessment is histogram comparison. In this method, we create histograms for the CFD solver and AI module results and estimate a numerical parameter that expresses how well two histograms match with each other. The histogram comparison is made with the coefficient of determination, which is the percentage of the variance of the dependent variable predicted from the independent variable. The results vary from 89.1% to 99.4%, with an average accuracy of 92.5%. All metrics are included in Table 1.

We have also performed a collective comparison of the results, where we plot a function y(x), where x represents the results obtained from the CFD solver, while y is the prediction. The results are shown in Fig. 6. The black line shows the perfect similarity, while the blue dots reveal the prediction uncertainty.

Now we focus on the performance analysis. We start with comparing the execution time for the AI module executed on the CPU and GPU. In this experiment, the mesh size is one million cells, and the CFD solver is run only on the CPU. The AI-accelerated part is executed in three steps, including data formatting and data export (implemented on CPU), as well as the AI prediction (performed on the CPU or GPU platform). This part includes 90.4% of the

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_29

Table 2: Comparison of execution time for CPU and GPU

	CPU	GPU	Ratio GPU/CPU
Data formatting [s]	65.1		
Prediction (inference) [s]	41.6	290.6	7.0
Data export [s]	9.5		
Entire AI module [s]	116.2	365.2	3.1
Full simulation [s]	1483.3	1732.3	1.2

simulation. For the AI-accelerated approach, the full simulation includes 9.6% of all CFD iterations executed by the CFD solver and the AI-accelerated part.

Data formatting takes 65.1 s, while the data export takes 9.5 s. The AI prediction time depends on the selected platform, taking 41.6 s on the CPU and 290.6 s on the GPU platform. So, somewhat surprisingly, the AI-accelerated part (formatting + prediction + export) runs 3.1 times faster on the CPU than in the case when the the GPU is engaged. Considering the CFD solver overhead (9.6% of all iterations), we can observe that this is the most time-consuming component of the entire AI-accelerated simulation. So the final speedup of the CPU-based simulation is 1.2 against the case when the GPU is engaged. The performance details are summarized in Table 2, where the execution time for the full simulation (last row) includes executing both the entire AI module and the first 9.6% of the simulation, which takes 1367.2 s.

The reason for the performance loss for the GPU prediction (inference) is a high time of data allocation on the GPU compared with CPU and multiple data transfers between the host and GPU global memory. These transfers are required

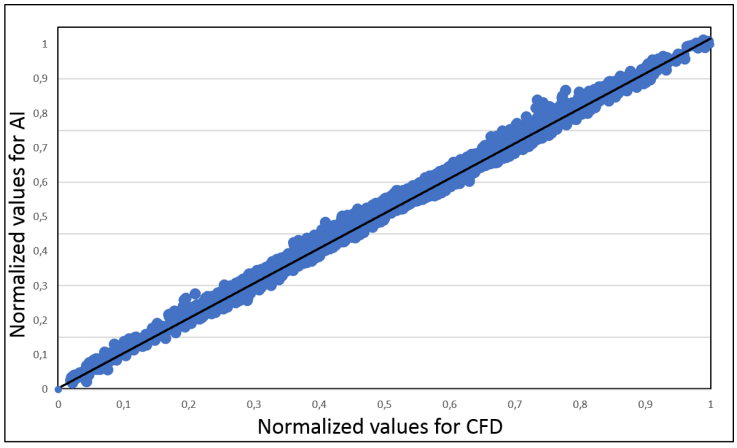


Fig. 6: Comparison of simulation results for the conventional CFD solver and AI-accelerated approach

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_29

Table 3: Comparison of execution time for simulation of chemical mixing using either the conventional CFD solver or AI-accelerated approach

	9.6% of sim.	90.4% of sim.	100% of sim.
CFD solver [s]	1367.2	12874.1	14241.2
AI-accelerated (CPU) [s]	1367.2	116.2	1483.3
Speedup	1	110.8	9.6
AI-accelerated (engaging GPU) [s]	1367.2	365.2	1732.3
Speedup	1	35.3	8.2

because each quantity has its sub-model that needs to be sequentially loaded. On the CPU platform, all the predictions of sub-models can be pipelined, and the memory overhead becomes much lower.

The final performance comparison considers the execution time for the conventional CFD solver and AI-accelerated approach. The first 9.6% of the simulation takes 1367.2 s. The remaining part takes 12874.1 s for the conventional CFD solver. Using the AI-accelerated module, this part is computed 110.8 times faster when executed on CPU, and 35.3 times faster when GPU is involved. As result, the entire simulation time is reduced from 14241.2 s to 1483.3 s (9.6x speedup) for the CPU, and to 1732.3 s (8.2x speedup) when the GPU is engaged. These results are summarized in Table 3.

Fig. 7 illustrates the performance advantages of the proposed AI-accelerated solution against the conventional CFD solver. Here the blue bars show the execution time for the whole simulation, while the orange ones correspond to executing 90.4% of the simulation. The two bars on the left side correspond to using exclusively the conventional CFD solver, while other bars demonstrate the advantages of using the AI module to reduce the execution time of the simulation. Fig. 7 not only illustrates the speedup achieved in this way but also demonstrates that this speedup is finally limited by the time required to perform the initial 9.4% of the simulation using the OpenFOAM CFD solver.

6 Conclusions and Future Works

The proposed approach to accelerate the CFD simulation of chemical mixing allows us to reduce the simulation time by almost ten times compared to using the conventional OpenFOAM CFD solver exclusively. The proposed AI module uses 9.6% of the initial iterations of the solver and predicts the converged state with 92.5% accuracy. It is expected that this reduction in the execution time will translate [21] into decreasing the energy consumption significantly, which means reducing the environmental footprint, including the carbon footprint [19]. However, the reliable evaluation of this effect is the subject of our future work since it requires considering the whole workflow, including both the inference and training stages.

Our method is fully integrated with the MixIT tool and supports 3D meshes with one million cells. Thanks to a data-driven approach, this method does not

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_29

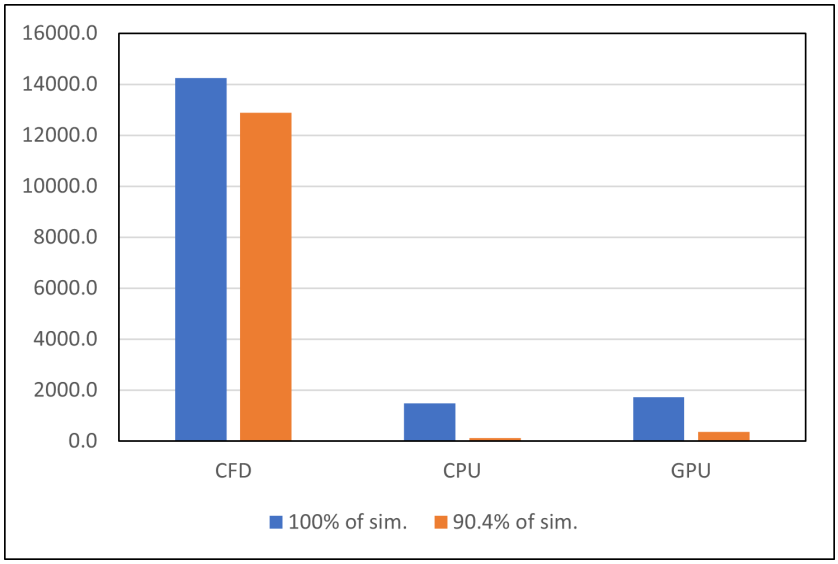


Fig. 7: Advantages in execution time (in seconds) achieved for the AI-accelerated approach over the conventional CFD solver

require a high knowledge of the CFD solvers to integrate it with the proposed solution. Such an integration gives a promising perspective to apply the method for CFD solvers that constantly evolve since it does not require going deep into CFD models.

The AI module is portable across the CPU and GPU platforms, which allows us to utilize the GPU power in the training stage and provides high performance of prediction using the CPU. The proposed pipelined model can separately train each quantity that significantly reduces memory requirements and supports larger meshes on a single node platform.

Our method is still under development. Particular attention will be paid to support more parameters of the CFD simulation of chemical mixing, including shape, position and number of impellers, the shape of the tube, number of baffles, and mesh geometry. Another direction of our research is providing further accuracy improvement and reduce the number of initial iterations required by the AI module.

References

1. Archibald, R. et al.: Integrating Deep Learning in Domain Sciences at Exascale. arXiv preprint arXiv:2011.11188v1 (2020)
2. Bhatt, D., Zhang, B. W., Zuckerman, D. M.: Steady-state simulations using weighted ensemble path sampling. The Journal of Chemical Physics, **133**(1) (2010)

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_29

3. Chen D., Hu F., Nian G., Yang T.: Deep Residual Learning for Nonlinear Regression. Entropy **22**(2):193, (2020)
4. Ferziger, J.H., Peric, M., Street, R.L.: Computational Methods for Fluid Dynamics. 4th edn. Springer Nature Switzerland (2020)
5. Jim, M., Ray, D., Hesthaven, J.S., Rohde, C.: Constraint-Aware Neural Networks for Riemann Problems. arXiv preprint arXiv:3327.50678 (2019)
6. Handbook of Industrial Mixing: Science and Practice. Paul, E.L., Atiemo-Obeng, V., Kresta, S.M. (eds.), John Wiley & Sons, Hoboken NJ (2004)
7. Jouppi, N.P., Young, C., Patil, N., Patterson, D.: A Domain-Specific Architecture for Deep Neural Networks. Commm. ACM, **61**(9), 50–59 (2018)
8. Kim, B., et al.: Deep Fluids: A Generative Network for Parameterized Fluid Simulations. arXiv preprint arXiv:1806.02071v2 (2019)
9. Kreitmayer, D. et al.: CFD-based Characterization of the Single-use Bioreactor XcellerexTM XDR-10 for Cell Culture Process Optimization. arXiv preprint arXiv:3461.77983 (2020)
10. Maulik, R., Sharma, H., Patel, S., Lusch, B., Jennings, E.: Accelerating RANS turbulence modeling using potential flow and machine learning. arXiv preprint arXiv:1910.10878 (2019)
11. MixIT: the enterprise mixing analysis tool, https://mixing-solution.com/. Last accessed 5 Feb 2021
12. Mostafazadeh, B., Marti, F., Pourghassemi, B., Liu, F., Chandramowlishwaran, A.: Unsteady Navier-Stokes Computations on GPU Architectures. In: 23rd AIAA Computational Fluid Dynamics Conference (2017) https://doi.org/10.2514/6.2017-4508
13. Obiols-Sales, O., Vishnu, A., Malaya, N., Chandramowlishwaran, A.: CFDNet: a deep learning-based accelerator for fluid simulations. In: Proc. 34th ACM Int. Conf. on Supercomputing (ICS'20), pp. 1-12, ACM (2020)
14. OpenFOAM: the open source CFD toolbox, https://www.openfoam.com. Last accessed 5 Feb 2021
15. Rojek, K. et al.: Adaptation of fluid model EULAG to graphics processing unit architecture. Concurrency and Computation: Practice and Experience **27**(4), 937–957 (2015)
16. Rojek, K., Wyrzykowski, R., Kuczynski, L.: Systematic adaptation of stencil-based 3D MPDATA to GPU architectures. Concurrency and Computation: Practice and Experience **29**(9), e3970 (2017)
17. Rojek, K. Halbiniak, K., Kuczynski, L.: CFD code adaptation to the FPGA architecture. The International Journal of High Performance Computing Applications **35**(1), 33–46 (2021)
18. Sanchez-Gonzalez, A. et al.: Learning to Simulate Complex Physics with Graph Networks. arXiv preprint arXiv:3394.45567 (2020)
19. Schwartz, R., Dodge, J., Smith, N.A., Etzioni, O.: Green AI. Commm. ACM, **63**(12), 54–63 (2020)
20. Szustak, L., Rojek, K., Olas, T., Kuczynski, L., Halbiniak, K., Gepner, P.: Adaptation of MPDATA heterogeneous stencil computation to Intel Xeon Phi coprocessor. Scientific Programming, 14 pages (2015) https://doi.org/10.1155/2015/642705
21. Szustak, L., Wyrzykowski, R. Olas, T. Mele, V.: Correlation of performance optimizations and energy consumption for stencil-based application on Intel Xeon scalable processors. IEEE Trans. Parallel and Distributed Systems, **31**(11), 2582–2593 (2020)

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_29

22. Tompson, J., Schlachter, K., Sprechmann, P., Perlin, K.H.: Accelerating Eulerian Fluid Simulation with Convolutional Networks. In: ICML'17: Proc. of the 34th International Conference on Machine Learning, PLMR 70, pp. 3424–3433 (2017)

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77964-1_29