Auto-tuning Multi-GPU High-Fidelity Numerical Simulations for Urban Air Mobility

Konstantina Koliogeorgi
Microprocessors and Digital Systems Lab
National Technical University of Athens
Athens, Greece
konstantina@microlab.ntua.gr

George Anagnostopoulos
Microprocessors and Digital Systems Lab
National Technical University of Athens
Athens, Greece
geoanagn@microlab.ntua.gr

Gerardo Zampino
FLOW, Engineering Mechanics
KTH Royal Institute of Technology
Stockholm, Sweden
gzampino@kth.se

Marcial Sanchis
FLOW, Engineering Mechanics
KTH Royal Institute of Technology
Stockholm, Sweden
sanchis@kth.se

Ricardo Vinuesa
FLOW, Engineering Mechanics
KTH Royal Institute of Technology
Stockholm, Sweden
rvinuesa@mech.kth.se

Sotirios Xydis
Microprocessors and Digital Systems Lab
National Technical University of Athens
Athens, Greece
sxydis@microlab.ntua.gr

***Abstract*—The aviation field is rapidly evolving towards an era where both typical aviation and Unmanned Aicraft Systems are essential and co-exist in the same airspace. This new territory raises important concerns regarding environmental impact, safety and societal acceptance. The RefMap European Project is an initiative that addresses these issues and aims at optimizing air traffic in terms of the environmental footprint in aviation and drone flights. One of RefMap's objectives is the development of powerful deep-learning models that predict urban flow based on extensive CFD simulations. The excessive time requirements of CFD simulations require the computational power of exascale heterogeneous supercomputer clusters. This work presents RefMap's strategy to mitigate simulation to GPU-enabled high-class solvers and further leverage sophisticated autotuning HPC techniques for creating portable high-performance simulations that can efficiently run on any GPU architecture and parallel system.**

***Index Terms*—Computational Fluid Dynamics, GPU acceleration, Autotuning**

I. Introduction

The convergence of climate concerns, the impact of COVID-19 and energy crises has hastened a societal and political transition towards a more digital, intermodal, and sustainable air transport, where airliners will operate on a environment-neutral manner and co-exist with Unmanned Aircraft Systems, commonly known as UAVs or drones. The paradigm shift towards Urban Air Mobility (UAM) introduces new challenges [1] such as complex air traffic management (ATM) and real-time forecasting as well as safe drone flights within urban areas. At the same time, the growth rate of airspace leads to a critical increase in environmental impacts, such as (CO_2) and non-CO_2 emissions and disturbance on wildlife [2].There is also an important societal aspect to UAS and UAM, as drone flights incur poor local air quality as well as annoyance reported by residents due to noise exposure. All these stress the importance of developing an aviation business model that leads to a more resilient and sustainable multi-scale aviation ecosystem.

Towards this goal, European Horizon Project RefMap [3] has been assembled with the mission to develop a digital cloud-based service aimed at quantifying the environmental footprints of air mobility for both airliners and UAV/UAM and optimizing air traffic optimisation in real time with the intent of minimal environmental impact in communities. RefMap aims to address these goals through the use of high-performance accelerated AI methodologies that empower innovative prediction models. For the UAM case specifically, the goal is to deliver a data-driven framework that enables online airflow predictions to facilitate the safe and publicly-acceptable operation of UAVs in terms of noise, visual pollution and air traffic density. The latter real-time urban flow predictions will rely on deep-learning models. Due to the lack of urban airflow data of high spatial resolution, i.e. idiosyncrasies of buildings, positions, wall configurations etc., the training of these deep-learning models can only be based on data derived by physics simulations of Computational Fluid Dynamics that take into account turbulence effects. RefMap proposes a novel multi-fidelity approach so that the predictive models are trained on both high-fidelity simulation data, i.e. either Large Eddy Simulations (LES) [4] or Direct Numerical Simulations (DNS) [5], and low-fidelity data, i.e. Reynolds-Averaged-Navier-Stokes (RANS) [6] simulations. DNS examines turbulent structures by taking into account all dependent variables of the problem while RANS provide a quicker insight into the impact of a single parameter. In the proposed RefMap approach, data-driven RANS simulations are firstly validated with LES data conducted for a few benchmark cases and then used for the deep-learning model training. However, this multi-fidelity analysis encounters significant bottlenecks, particularly for simulations involving complex geometries and realistic scenarios. For example, a single DNS/LES simulation on KTH supercomputer Dardel with 128 physical CPU cores requires 150 time units in simulation time to achieve the statistical convergence, which is equivalent to about two weeks of computational time. Similarly, despite the computational efficiency of single RANS simulations, multiple RANS simulations are required to explore the influence of a single parameter, adding up to a considerable amount of computational time.

The aforementioned challenges are only aggravated by the

979-8-3503-4859-0/DATE24/© 2024 EDAA

intense data requirements of model training. A large number of both LES and RANS simulations are required to generate this data, leading to prohibitive time requirements. To address this issue and enhance the efficiency of generating training datasets, RefMap planning outlines optimization strategies for both low- and high-fidelity simulations using High-Performance Computing (HPC) techniques and GPU acceleration. This strategy aligns well with state-of-the-art works that leverage HPC parallelization computing paradigms such as MPI to achieve parallel execution of CFD solvers and most recent efforts to bring CFD solvers closer to exascale computing by incorporating GPU accelerators. For instance, leveraging a homogeneous multi-core HPC cluster, Yan et al. [7] developed a scalable parallel simulator for the wind flow in a full-scale urban community utilizing the Newton–Krylov–Schwarz method. Moving towards heterogeneous clusters, initial works [8] studied real-time large-eddy simulation of flow in a neutral atmosphere over realistic urban areas targetting GPGPUs for enhanced performance. More realistic models like CityFFD [9], provide a large-eddy simulation method that captures the turbulence in an urban area and is accelerated through parallel computing on GPUs. Recent works [10] continue to contribute to this field with novel urban microclimate simulation methods and HPC-optimized solvers to set the ground for the creation of sustainable urban environments. Still, there is a gap between the advancements and powerful optimization tools in the HPC domain and their adoption in state-of-the-art CFD solvers. RefMap aims at bridging this gap by developing accurate models for flow prediction and leveraging HPC optimization techniques and GPU accelerators to increase their efficiency.

In this paper, we (i) introduce the vision of RefMap to quantify the safety, environmental and societal impact of UAS and UAM and deliver a cloud-service that predicts air-flow in urban environments based on powerful deep-learning models. RefMap plans on acquiring the training data for these models through numerous a multi-fidelity approach based on both LES and RANS simulations. (ii) Describe the high resolution Large Eddy Simulation method that accurately describes the turbulent structures within a realistic representation of an urban environment and generates high fidelity data for training the deep learning models, (iii) motivate the need for HPC optimization of the LES simulations to acquire training data in a high-throughput fashion and present the acceleration capabilities of powerful state-of-the-art GPU-enabled CFD solvers, (iv) present an advanced optimization approach that leverages GPU-enabled solver with MPI support and extends it with an auto-tuning tool to create an optimization framework for portable and highly efficient simulation deployment on different parallel systems and GPU architectures. The proposed methodology introduces minimal modifications to the solver's source code that is critical to creating a representative search space and allow a robust search that takes both algorithmic and platform capabilities into consideration.

II. Theoretical Background

A. Computational Fluid Dynamics Simulations

RefMap aims to develop a new optimisation approach to reduce the environmental footprint through transformative multi-scale aviation planning. To achieve this goal, the turbulent flow in a urban environment needs to be carefully described in order to produce a more precise model for predicting turbulent structures within the wake of buildings arranged in tandem or a canopy. The turbulent flow in cities is simulated with high-resolution Large Eddy Simulations (LES) for a simplified configuration consisting of a wall-mounted cylinder in a turbulent boundary layer to simulate an athmospheric boundary layer.

In RefMap, we develop the above high-resolution LES using the open-source software Nek5000, developed by Fischer [11]. Nek5000 employs the Spectral-Element Method (SEM), initially introduced by Patera [12], which combines classical finite element methods with spectral methods. The normalized governing equations in the streamwise (x), vertical (y), and spanwise (z) directions

$$\frac{\partial \widetilde{u}_i}{\partial t} + \widetilde{u}_j \frac{\partial \widetilde{u}_i}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{1}{h}\frac{\partial^2 \widetilde{u}_i}{\partial x_j \partial x_j} - \mathcal{H}(\widetilde{u}_i) \tag{1}$$

$$\frac{\partial \widetilde{u}_i}{\partial x_i} = 0, \tag{2}$$

are discretized in each element using the $P_n P_{n-2}$ formulation of the Galerkin projection method [13]. Here, n represents the maximum polynomial order of the trial function for velocity, while $n-2$ corresponds to the polynomial order for the pressure trial function. The main advantage of this method, as opposed to finite element methods, lies in its rapid convergence, minimal dispersion and diffusion errors, and its ability to provide a more accurate description of complex geometries [14]. For the RefMap purpose, using the SEM allows us to increase the spacial resolution by increasing the polynomial order rather than changing the mesh and hence obtain more accurate results with weak impact on the computational cost. An 8th-order Gauss-Lobatto-Legendre (GLL) polynomial was employed, and the domain size was set to 16h, 3h, and 4h in the x, y, and z directions, respectively, where h denotes the height of the obstacle. The dimensions of the mesh elements were constrained to $\Delta x^+ < 18$, $\Delta y^+ < 0.5$, and $\Delta z^+ < 9$. Given these considerations, simulations are carried out for at least 220,000 nodes to have the best compromise between the computational cost and the accuracy of the solution. Fig. 1 displays the wake of a wall-mounted cylinder immersed in a turbulent boundary layer. The aspect ratio of the cylinder, defined as the ratio between the height and its width, is equal to 2 while the friction Reynolds number in front of the obstacle is $Re_\tau \approx 180$. In Fig. 1, we reported the isosurface at $\lambda_2 = -40$ while the colormap refers to the instantaneous streamwise velocity. Although the description of the physics behind the turbulent structures observed in this specific configurations lies beyond the purpose of the present paper (for additional details see [15]), it is worth noting the complexity of turbulent structures behind the obstacle. The three-dimensional nature of the wake makes the less accurate simulations, e.g. RANS, unsuitable for this analysis and necessitates the use of computationally expensive higher order simulations.

B. CFD scalable deployments and GPU acceleration

The computational cost of CFD solvers and complex geometries for realistic scenarios have always pressed the need

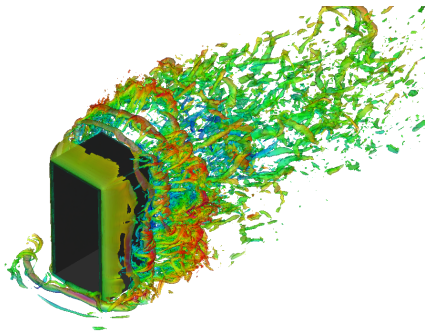

Fig. 1. Isosurface of the $\lambda_2 = -40$ inside the wake of a wall-mounted square cylinder with aspect ratio (AR) 2 and immersed in a turbulent boundary layer. The friction Reynolds number in front of the cylinder is $Re_\tau \approx 180$ based on the height of the cylinder h. The colormap is the instantaneous streamwise velocity.

for utilizing efficient and scalable packages that leverage HPC optimization techniques and support parallel execution on powerful clusters. Efficient deployment on these platforms is based on parallel computing frameworks, such as MPI [16], OpenMP [17] or Hybrid MPI+OpenMP, that split the workload into independent tasks and distribute each task across multiple cores.

The Nek5000 solver leverages MPI to exploit both inter- and intra-node parallelism and has demonstrated considerable scalability [18] across several thousands of nodes on petascale systems. The current version of Nek5000 however is not optimized to exploit the capabilities of modern High-Performance Computing (HPC) architectures that are shifting from traditional homogeneous systems towards modern GPU-accelerated Exascale computing platforms. Therefore, an active field of research focuses on transforming Nek5000 and other solvers so that they can benefit from the use of accelerated technologies, such as GPUs. Several GPU-accelerated solvers [19]–[21] were based on OpenACC and OCCA [22] for GPU programming, while more recent works [23] use CUDA for Nvidia GPUs or a combination of OpenACC and CUDA [24]. NekRS [25] is also a recent promising solution for GPU acceleration that is written in C++/OCCA and employs just-in-time (JIT) compilation for platform portability. Despite promising performance enhancements for some cases, existing GPU-enhanced tools do not yet support all types of simulations and require a tool-specific formulation of the problem, leading to considerable overheads for the scientists, i.e. months for setting up a new mesh for the tool.

The plan within RefMap vision is to mitigate from the golden-standard Nek5000 simulation to SOD2D [26], one of the latest promising attempts at GPU-enabled solvers that has the ability to work well with complex meshes. SOD2D is a CFD algorithm for scalable simulation of turbulent compressible flows that leverages spectral formulation of the Continuous Galerkin Finite Elements model [13] applied to the spatial terms in the Navier-Stokes system coupled with an Entropy Viscosity stabilization model. SOD2D is a Fortran-based solver that supports MPI execution to scale the simulation across multiple CPU or GPU-enhanced nodes. GPU-acceleration is enabled through the use of GPU-oriented OpenACC directives.

OpenACC in SOD2D: OpenACC [27] programming model targeting mainly GPU accelerators that achieves source code portability and good performance through the use of high-level platform-independent compiler directives annotating the source code of the application. Parallelizing compilers leverage theses directives, called *pragmas*, to expose parallelism in the code and build optimized code for a target device architecture. Computationally heavy regions in the code are annotated with OpenACC pragmas to indicate off-loading computation to the GPU and determine its *parallelism factor* of each off-loaded segment. OpenAcc directives achieve three levels of parallelism: *gang, worker and vector*. Gang, worker and vector sizes correspond to number of ThreadBlocks, Warps and Threads, respectively, in typical CUDA naming convention. Specifically, a gang consists of one or more workers, while each worker operates on a vector of some length. Therefore, gang refers to coarse-grained parallelism while vector to SIMD parallelism.

In SOD2D, the algorithmic structure for the most demanding functions, i.e. convective and diffusive kernels, has been designed to distribute one element of the mesh per thread block, with threads in a block handling either nodal or gaussian quadrature operations [26]. OpenACC directives are placed strategically above the corresponding loops to achieve parallelization across the nodes and dimensions.

SOD2D MPI Parallelization: SOD2D utilizes MPI for multi-scale parallel execution on CPU or GPU nodes. Parallel execution of the solver is based on the idea of splitting the mesh into partitions and independent tasks and then distributing each task across different MPI ranks. The grid points of the mesh are partitioned before runtime into a connectivity matrix that indicates how the mesh should be distributed across ranks. Special attention and synchronization is required for communication between ranks computing boundary and periodic nodes.

III. Proposed Autotuning Framework

This section describes the proposed RefMap methodology for efficiently deploying LES simulations on heterogeneous supercomputing clusters. Our strategy leverages the baseline architecture-agnostic GPU-acceleration capabilities of existing SOD2D [26] solver and further enhances them by coupling them with an autotuning algorithm to guarantee portability and highly efficient code for any given system.

A. Key Idea: Exploration on Decoupled OpenAcc Pragmas

As mentioned in Section II-B, SOD2D places OpenACC pragmas in computational intensive functions in a strategic manner that adheres to data dependencies and supports parallel execution. The OpenAcc pragmas are placed mainly above for-loop regions to indicate the degree of parallelism by the number of gangs and vector length. SOD2D adopts global values for this parameters and applies them to all pragmas across SOD2D. However, this is an arbitrary decision that ignores the target GPU architecture and leaves out heterogeneous configurations for pragmas applied at different locations. As a result, the achieved parallelization may not be the optimal one. The key idea of the proposed methodology is to decouple OpenACC pragmas and create a parameter space that better represents the problem and captures the acceleration capabilities. An

Fig. 2. Overview of Proposed Autotuning Framework on SOD2D.

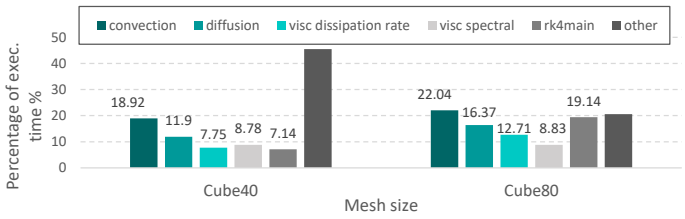

Fig. 3. Hotspot functions of SOD2D as identified by Nsight profiler on Taylor Green Vortex simulation for different mesh sizes.

autotuning framework will then be leveraged to efficiently explore the search space and reach a superior, if not global, optimal configuration. This strategy combines the algorithmic parallelization prospects and architecture-specific acceleration capabilities and therefore achieves higher performance.

1) Parameter space definition: OpenACC pragmas can be found in various places in SOD2D starting from the initialization phase to the time integration loop. Depending on their placement, they have a different impact on the parallelization and overall performance of SOD2D. Based on our proposed strategy, decoupling all pragmas leads to an exponentially large space for the autotuner that immensely explodes the complexity of the exploration. Fig.2 presents our strategy for creating a targeted yet compact parameter space: we first perform a profiling to identify the most time consuming functions and focus on the OpenACC pragmas included in these functions.

Using the NVIDIA Nsight Systems tool [28] we gather profiling information for the SOD2D execution of a representative benchmark, i.e. Taylor-Green Vortex [29] simulation that shows the capability of the model to handle both discontinuities and turbulent structures. Fig.3 presents an histogram of the distribution time across SOD2D functions for two different meshes, referenced as Cube40 and Cube80, that differ in size, i.e., 64000 and 512000 elements respectively. In both cases, the results reveal that the computational overhead primarily comes from the $convection$ and $diffusion$ terms of Navier Stokes equations and secondarily from the $viscosity\ dissipation\ rate$ and $smart\ viscosity\ spectral$ functions. Function $rk4main$ poses a considerable overhead too, but it is distributed across multiple smaller tasks. We target the four first functions that require 50-60% of the total time in the examined cases and are expected to further dominate for larger mesh sizes.

The OpenAcc pragmas included in the hotspot functions define the parameter space for the autotuning exploration. These dominant pragmas only include the gang and vector parallelization factors. Based on our decoupling approach, we define a single point in the parameter space as a configuration of $gang\_number$ and $vector\_length$ values for all located pragmas in the hotspots. Table I reports the parameter space and the range of values for each one as well as their place in SOD2D source code. The values are selected based on the minimum number of threads per GPU warp. The searchable solution space, i.e. all possible combinations for the field value, amount to $\prod_{i=1}^{K} gang\_values_i \times vector\_lengths_i$ configurations, where K is the number of functions to perform the exploration on and $gang\_values$, $vector\_lengths$ is cardinality of the respective sets, i.e. the number of possible values for number of gangs and vector length respectively. In our case, this amounts to $250^4 * 20^4$.

TABLE I
CONFIGURATION FIELDS AND RANGE OF VALUE FOR THE APPLIED DIRECTIVES AND THEIR PARAMETERS

Function	OpenACC pragma Field	Range of values	Step
full_convec_ijk	num_gangs	[2048-512000]	2048
	vector_length	[32-640]	32
full_diffusion_ijk	num_gangs	[2048-512000]	2048
	vector_length	[32-640]	32
visc_dissipationRate	num_gangs	[2048-512000]	2048
	vector_length	[32-640]	32
smart_visc_spectral	num_gangs	[2048-512000]	2048
	vector_length	[32-640]	32

2) Optimization objective and Exploration strategy: We employ the OpenTuner [30] autotuning framework to drive the performance optimization. Fig.4 describes the major steps of our performance autotuning process. OpenTuner implements the main optimization loop by sampling the parameter space in each iteration, executing the selected configuration on the available infrastructure and gathering the performance metrics. Based on these metrics, the optimizer component evaluates the previous selections and the most recent one and decides on the next configuration.

Optimizer: For the optimizer, we select the exploration strategy that is based on the Multi-Arm Banded approach, i.e., *AUC Bandit*. *AUC Bandit* is a multi-armed bandit with sliding window, area under the curve credit assignment meta technique. This meta technique uses the AUC Bandit approach to combine greedy mutation, differential evolution, and two hill climber instances.

B. Ensuring simulation correctness

Simulation for cases such as flow prediction have very strict accuracy requirements. At the same time, they are very sensitive to changes to computational patterns due to floating point operations. The proposed methodology examines multiple configurations that distribute the initial computations to a different number of blocks on the GPU or number of threads per blocks. The result is changing the execution order of floating point operations or generating different intermediate values. This can lead to reduced precision that does not meet the requirements.

We take this challenge into account by meticulously examining the accuracy of each examined configuration. We compare each output to the reference output of SOD2D simulation and define an acceptable deviation of corresponding values greater than 10^-6. To further guarantee the accuracy of the results, we perform a 10-fold validation of the results for

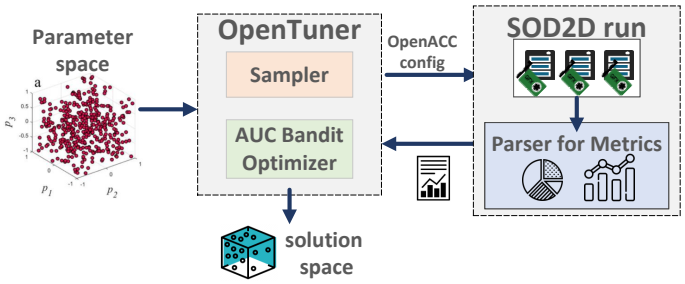

Fig. 4. Overview of Proposed Autotuning Framework on SOD2D.

the top configurations by averaging the output values over 10 executions and then comparing them to the reference values.

IV. Experimental Evaluation

A. Experimental Setup

The proposed framework has been deployed on BerzeLiUS [31] supercomputing platform. Each node of the platform is equipped with AMD Epyc 7742 CPUs and 8 NVIDIA A100 Tensor Core GPUs. All modules of our framework are containerized to facilitate portability on any infrastructure and GPU architectures. The container used on BerzeLiUS is build on a base image with CUDA toolkit 11.8 and NVHPC version 23.3. All experiments are performed on the Cube40 mesh.

B. Evaluation of Proposed Framework

Experimental methodology: Our experimental campaign aims at a fair and thorough evaluation of the proposed methodology. We therefore compare three frameworks: (i) SOD2D-ORIG: the original SOD2D code configured with the default values for the OpenAcc pragmas without leveraging an autotuning tool, (ii) SOD2D-TUN-DECOUPLED: the implementation of the proposed methodology described in Section III where we apply OpenTuner after decoupling the OpenAcc pragma values and (iii) SOD2D-TUN-COUPLED: an alternative exploration approach that applies OpenTuner over a more compact search space that results from coupling the values for the examined pragmas, i.e. forcing the same value for all gang numbers and vector lengths for the four hotspots. For the SOD2D-TUN-DECOUPLED and SOD2D-TUN-COUPLED approaches we constraint the examined configurations by setting a limit on the exploration run time approximate 4 hours.

Single-GPU analysis: We first evaluate each framework on a single GPU in terms of performance, accuracy and number of configurations examined. We evaluate accuracy as an error rate, i.e. the percentage of output mesh values that deviate from the original SOD2D output by a value greater than 10^-6. Table II includes the OpenACC pragmas value and evaluation for the configurations proposed by each framework. The abbreviations ng and vl correspond to pragmas num_gangs and vector_length respectively and the subscripts fc, fd, vdr, svs indicate the hotspot functions. The SOD2D-TUN-COUPLED and SOD2D-TUN-DECOUPLED approaches delivers a 6% and 15% performance gain respectively without any loss at the accuracy of the output values. We validate that the optimal solution includes different configuration values for all pragmas. Despite the limited end-to-end speedups acquired, it is important to note

TABLE II
Configuration and Evaluation metrics for the optimal solution proposed by each framework.

Framework	OpenACC pragma	Value Field	Error Rate	Time (sec)	Total Configs
SOD2D-ORIG	ng, vl ng_{svs}, vl_{svs}	[8000, 128] [8000, 64]	N/A	15.384	N/A
SOD2D-TUN COUPLED	ng, vl	[131072, 64]	0%	14.402	416
SOD2D-TUN DECOUPLED	ng_{fc}, vl_{fc} ng_{fd}, vl_{fd} ng_{vdr}, vl_{vdr} ng_{svs}, vl_{svs}	[397312, 96] [253952, 96] [28672, 32] [290816, 32]	0%	13.013	457

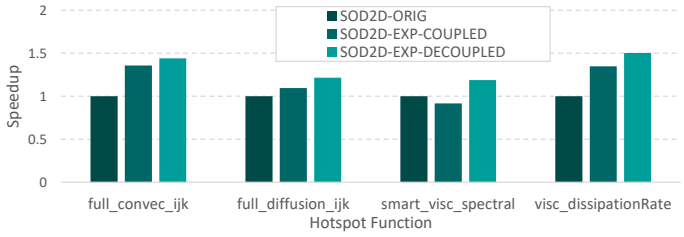

Fig. 5. Speedup evaluation for each hotspot function leveraring three different frameworks and the capabilities of a single GPU.

that this speedup aggregates to significant time savings, e.g. it saves days of computations for 2-weeks-long simulations that need to be executed numerous times.

Figure 5 showcases the speedup achieved for each hotspot function individually normalized to the execution time of SOD2D original configuration. For all functions, the SOD2D-TUN-DECOUPLED outperforms the two other frameworks and delivers speedup ranging from $\times$1.19 to $\times$1.5. SOD2D-TUN-COUPLED outperforms SOD2D-ORIG except for the $smart\_visc\_spectral$ function due to sharing the same values for OpenACC pragmas with the other functions rather than a function-specific optimal configuration.

Multi-GPU analysis: We further evaluate the frameworks over the MPI scaled implementation of SOD2D. Fig. 6 shows the speedup of the autotuning approaches over the original SOD2D configuration for an increasing number of GPUs, i.e. 1, 2, 4 and 8 GPUs. All speedups are calculated with respect to the single-GPU original SOD2D configuration. Both autotuning approaches always outperform the single-GPU and respective multi-GPU original SOD2D and achieve similar performance. Note that the optimal solution found during autotuning for 4 GPUs, reaches the same speedup found when employing 8 GPUs, i.e. approximately $\times$11. Interestingly, as shown in Table III, the best OpenACC configuration for both frameworks changes when employing a different number of GPUs.

V. Conclusions

The RefMap European Project studies the shift of aviation towards Unmaned Air Mobility and its impacts on safety and environmental concerns. A major goal of RefMap is to deliver flow prediction for drone flights within an urban environment using powerful deep-learning models trained on data acquired by both high-resolution and RANS simulations. This work

TABLE III
OPTIMAL CONFIGURATIONS FOR MULTI-GPU ANALYSIS.

Framework	Pragmas	1-GPU	2-GPU	4-GPU	8-GPU
SOD2D-TUN-COUPLED	ng, vl	131072 64	374784 32	323584 160	423936 96
SOD2D-TUN-DECOUPLED	ng_{fc}, vl_{fc}	397312 96	18432 64	378880 192	292864 32
	ng_{fd}, vl_{fd}	253952 96	116736 160	446464 160	507904 320
	ng_{vdr}, vl_{vdr}	28672 32	360448 32	417792 320	417792 192
	ng_{svs}, vl_{svs}	290816 32	389120 32	432128 320	329728 192

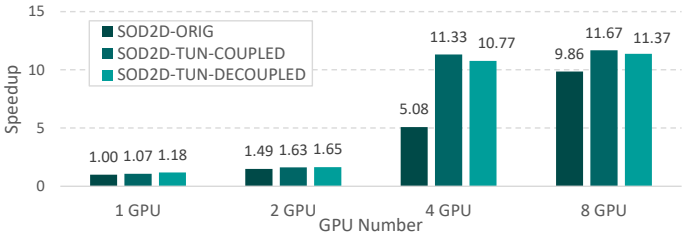

Fig. 6. Speedup evaluation of SOD2D best configurations resulting from autotuning on a scaling number of GPUs.

presents RefMap's plan to tackle the excessive computation requirements of LES simulations through GPU acceleration of CFD solvers and sophisticated autotuning HPC techniques to create portable high-performance simulations that can efficiently run on any GPU architecture and parallel system.

ACKNOWLEDGMENT

This work has been partially funded by EU Horizon 2022 program under grant agreement No 10109669 REFMAP (https://www.refmap.eu/).

REFERENCES

[1] A. Straubinger, R. Rothfeld, M. Shamiyeh, K.-D. Büchter, J. Kaiser, and K. O. Plötner, "An overview of current research and developments in urban air mobility–setting the scene for uam introduction," *Journal of Air Transport Management*, vol. 87, p. 101852, 2020.

[2] S. H. Yim, G. L. Lee, I. H. Lee, F. Allroggen, A. Ashok, F. Caiazzo, S. D. Eastham, R. Malina, and S. R. Barrett, "Global, regional and local health impacts of civil aviation emissions," *Environmental Research Letters*, vol. 10, no. 3, p. 034001, 2015.

[3] https://www.refmap.eu/, "Refmap eu project."

[4] P. J. Mason, "Large-eddy simulation: A critical review of the technique," *Quarterly Journal of the Royal Meteorological Society*, vol. 120, no. 515, pp. 1–26, 1994.

[5] P. Moin and K. Mahesh, "Direct numerical simulation: a tool in turbulence research," *Annual review of fluid mechanics*, vol. 30, no. 1, pp. 539–578, 1998.

[6] C. G. Speziale, "Turbulence modeling for time-dependent rans and vles: a review," *AIAA journal*, vol. 36, no. 2, pp. 173–184, 1998.

[7] Z. Yan, R. Chen, and X.-C. Cai, "Large eddy simulation of the wind flow in a realistic full-scale urban community with a scalable parallel algorithm," *Computer Physics Communications*, vol. 270, p. 108170, 2022.

[8] S. Lenz, M. Schönherr, M. Geier, M. Krafczyk, A. Pasquali, A. Christen, and M. Giometto, "Towards real-time simulation of turbulent air flow over a resolved urban canopy using the cumulant lattice boltzmann method on a gpgpu," *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 189, pp. 151–162, 2019.

[9] M. Mortezazadeh, L. L. Wang, M. Albettar, and S. Yang, "Cityffd–city fast fluid dynamics for urban microclimate simulations on graphics processing units," *Urban Climate*, vol. 41, p. 101063, 2022.

[10] M. Yang, G. Oh, T. Xu, J. Kim, J.-H. Kang, and J.-I. Choi, "Multi-gpu-based real-time large-eddy simulations for urban microclimate," *Building and Environment*, vol. 245, p. 110856, 2023.

[11] P. Fischer, J. Lottes, and S. Kerkemeier, "NEK5000: Open Source Spectral Element CFD Solver," 2008.

[12] A. T. Patera, "A spectral element method for fluid dynamics: Laminar flow in a channel expansion," *J. Comput. Phys.*, vol. 54, p. 468, 1984.

[13] V. Thomée, *Galerkin finite element methods for parabolic problems*. Springer Science & Business Media, 2007, vol. 25.

[14] G. Karniadakis and S. Sherwin, *Spectral/hp Element Methods for Computational Fluid Dynamics*. Oxford University Press, 2005.

[15] M. Atzori, P. Torres, A. Vidal, S. Le Clainche, S. Hoyas, and R. Vinuesa, "High-resolution simulations of a turbulent boundary layer impacting two obstacles in tandem," *Phys. Rev. Fluids*, vol. 8, p. 063801, Jun 2023. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevFluids.8.063801

[16] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine *et al.*, "Open mpi: Goals, concept, and design of a next generation mpi implementation," in *Recent Advances in Parallel Virtual Machine and Message Passing Interface: 11th European PVM/MPI Users' Group Meeting Budapest, Hungary, September 19-22, 2004. Proceedings 11*. Springer, 2004, pp. 97–104.

[17] R. Gayatri, C. Yang, T. Kurth, and J. Deslippe, "A case study for performance portability using openmp 4.5," in *Accelerator Programming Using Directives: 5th International Workshop, WACCPD 2018, Dallas, TX, USA, November 11-17, 2018, Proceedings 5*. Springer, 2019, pp. 75–95.

[18] N. Offermans, O. Marin, M. Schanen, J. Gong, P. Fischer, P. Schlatter, A. Obabko, A. Peplinski, M. Hutchinson, and E. Merzari, "On the strong scaling of the spectral element solver nek5000 on petascale systems," in *Proceedings of the Exascale Applications and Software Conference 2016*, 2016, pp. 1–10.

[19] S. Markidis, J. Gong, M. Schliephake, E. Laure, A. Hart, D. Henty, K. Heisey, and P. Fischer, "Openacc acceleration of the nek5000 spectral element code," *The International Journal of High Performance Computing Applications*, vol. 29, no. 3, pp. 311–319, 2015.

[20] M. Otten, J. Gong, A. Mametjanov, A. Vose, J. Levesque, P. Fischer, and M. Min, "An mpi/openacc implementation of a high-order electromagnetics solver with gpudirect communication," *The International Journal of High Performance Computing Applications*, vol. 30, no. 3, pp. 320–334, 2016.

[21] K. Świrydowicz, N. Chalmers, A. Karakus, and T. Warburton, "Acceleration of tensor-product operations for high-order finite element methods," *The International Journal of High Performance Computing Applications*, vol. 33, no. 4, pp. 735–757, 2019.

[22] D. S. Medina, A. St-Cyr, and T. Warburton, "Occa: A unified approach to multi-threading languages," *arXiv preprint arXiv:1403.0968*, 2014.

[23] M. Karp, N. Jansson, A. Podobas, P. Schlatter, and S. Markidis, "Optimization of tensor-product operations in nekbone on gpus," *arXiv preprint arXiv:2005.13425*, 2020.

[24] J. Vincent, J. Gong, M. Karp, A. Peplinski, N. Jansson, A. Podobas, A. Jocksch, J. Yao, F. Hussain, S. Markidis *et al.*, "Strong scaling of openacc enabled nek5000 on several gpu based hpc systems," in *International Conference on High Performance Computing in Asia-Pacific Region*, 2022, pp. 94–102.

[25] P. Fischer, S. Kerkemeier, M. Min, Y.-H. Lan, M. Phillips, T. Rathnayake, E. Merzari, A. Tomboulides, A. Karakus, N. Chalmers *et al.*, "Nekrs, a gpu-accelerated spectral element navier–stokes solver," *Parallel Computing*, vol. 114, p. 102982, 2022.

[26] L. Gasparino Ferreira da Silva, F. Spiga, and O. Lehmkuhl, "Sod2d: A gpu-enabled spectral finite elements method for compressible scale-resolving simulations," *Filippo and Lehmkuhl, Oriol, Sod2d: A Gpu-Enabled Spectral Finite Elements Method for Compressible Scale-Resolving Simulations*.

[27] R. Farber, *Parallel Programming with OpenACC*, 1st ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2016.

[28] https://developer.nvidia.com/nsight systems, "Nvidia nsight systems."

[29] C.-W. Shu, W.-S. Don, D. Gottlieb, O. Schilling, and L. Jameson, "Numerical convergence study of nearly incompressible, inviscid taylor–green vortex flow," *Journal of Scientific Computing*, vol. 24, pp. 1–27, 2005.

[30] J. Ansel, S. Kamil, K. Veeramachaneni, J. Ragan-Kelley, J. Bosboom, U.-M. O'Reilly, and S. Amarasinghe, "Opentuner: An extensible framework for program autotuning," in *Proceedings of the 23rd international conference on Parallel architectures and compilation*, 2014, pp. 303–316.

[31] https://www.nsc.liu.se/systems/berzelius/, "Berzelius cluster."