Ansys Fluent HPC Capabilities for Large-Scale CFD Engineering Simulations

Rongguang Jia[1], Mohammed S. Kamel[2]*, Chunliang Wu[3],
Ansys Inc., 10 Cavendish Court, Lebanon NH 03766, United State

Bharat R. Agrawal[4]
Ansys Software Pvt. Ltd., Pune, Maharashtra 411057, India

Advancements in High-Performance Computing (HPC) deployment and software development over the last few decades have established HPC as a cornerstone of engineering simulation, solving a wide range of real-world problems. Multi-scale, multi-physics Computational Fluid Dynamics (CFD) simulations of complex systems often require deploying meshes that can exceed one billion cells. In Ansys Fluent, various development efforts focus on enhancing scalability and performance to efficiently handle these large-scale simulations on many processor cores. Enhancements in parallelism techniques extend across all aspects of the CFD solver and pre- and post-processing operations. In this paper, Ansys Fluent demonstrates scalable performance for nearly 200,000 CPU cores. Jet-engine gas combustor, gas separation vessel, and full peloton of 121 cyclists are among the showcases which feature Ansys Fluent's HPC capabilities for tackling problems that have remained mostly unexplored. These large-scale CFD simulations prove the scalability and robustness of the Fluent solver across thousands of processors and billions of mesh elements.

Introduction

High Performance Computing (HPC) is a cornerstone of modern Computational Fluid Dynamics (CFD), enabling the simulation of complex fluid flows that are otherwise unfeasible to analyze through experimental measurements alone. As CFD simulation encompasses billions of cell elements, unique challenges emerge that span various aspects of HPC, including algorithms, software, and hardware platforms, and test its current capabilities.

One of the foremost challenges is the development and optimization of software capable of efficiently handling large-scale simulations. This involves creating robust algorithms that can leverage the full potential of HPC resources. Parallelization is crucial, yet it introduces complexities in maintaining the scalability of the simulation software efficiently handle emerging computational resources and more massive mesh sizes.

Partitioning and domain decomposition strategies must be refined to ensure optimal load balancing such that the computational load is evenly distributed across processors, minimizing idle time, and maximizing resource utilization. On the other hand, it is crucial to handle the complexities introduced by partitioning various kinds of mesh topologies, including sliding and overset meshes, fluid-solid interfaces and special boundary conditions.

Another layer of complexity arises from the diversity of HPC platforms, each with its own architecture, operating system, and performance characteristics. Careful tuning and optimization are necessary to ensure the CFD software and other employed libraries such as Message Passing Interface (MPI) and Open Multi-Processing (OpenMP), run seamlessly and efficiently across different platforms, whether they are traditional supercomputers, clusters, or emerging cloud-based systems.

I/O operations and data management are also critical concerns. The massive datasets generated and consumed by large-scale CFD simulations necessitate advanced I/O strategies to ensure data can be read, written, and modified

[1] Distinguished Engineer, Fluid Business Unit.
[2*]Senior R&D Engineer, Fluid Business Unit, Senior Member AIAA. Email: mohammed.kamel@ansys.com
[3] Principal R&D Engineer, Fluid Business Unit.
[4] Lead R&D Engineer, Fluid Business Unit.

efficiently. This includes the use of parallel I/O techniques and scalable storage solutions to handle vast amounts of data without becoming a bottleneck.

The accuracy and efficiency of linear solvers and numerical methods are paramount in maintaining the precision of simulations. Developing solvers that can scale effectively with the problem size and remain stable across billions of cell elements is a significant challenge.

In this paper, we focus on those challenges and explore the current solutions that significantly improve the performance and efficiency of large-scale simulations, enabling more accurate and efficient modeling of complex fluid dynamics.

Various industrial strength large-scale CFD simulations using Ansys Fluent are presented in this paper. The first case is the gas combustor case of 0.83 billion cells [1,2]. The second one is the gas separation vessel of 1.6 billion cells [3]. The third showcase is the first aerodynamic simulations of full pelotons of 121 road cyclists of 3 billion grid cells [4-6].

I. Areas of HPC Enhancement

A. Hardware and Software Advancement

Over the past 20 years, advancements in CPU technology have driven remarkable progress in computational performance. As shown in Fig. 1, the exponential increase in transistor density—approximately six times more transistors per core—has been a key enabler. However, single-core performance has seen only modest improvements, scaling proportionally with the transistor increase per core. While processor frequencies have plateaued and power usage has remained constant, the number of cores per node has grown dramatically, from just 1 core in 2004 to as many as 256 cores in 2024.

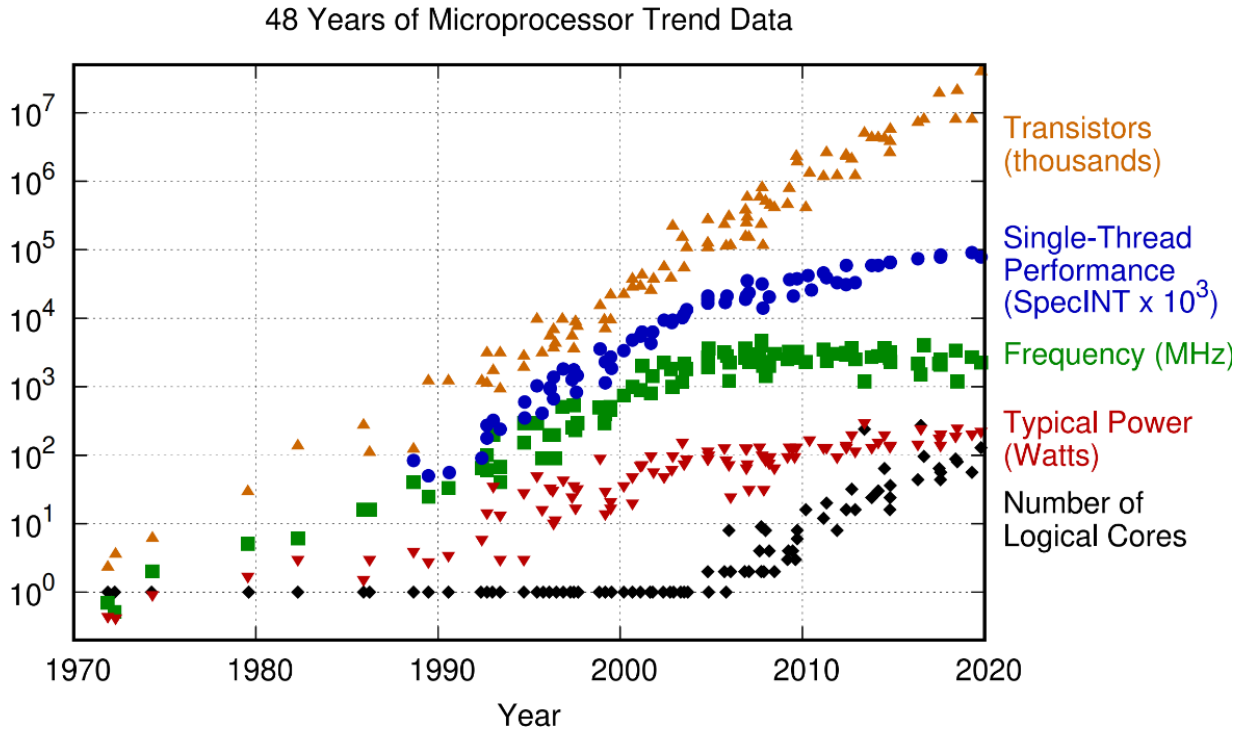


Fig. 1: Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, k. Olukotun, L. Hammond and C. Batten. New plot and data collected for 2010-2019 by K. Rupp [7]

These hardware advancements, combined with Fluent's scalable solver architecture, have resulted in significant performance gains. Fluent's single-node performance has improved by a factor of 166 over this period, driven by 32 times increase in core count and approximately 5 times enhancement in per-core performance. A CFD testcase of a sedan car with 4 million cells is taken as an example in Fig. 2. These trends underscore the synergy between hardware evolution and software scalability in achieving high computational efficiency.

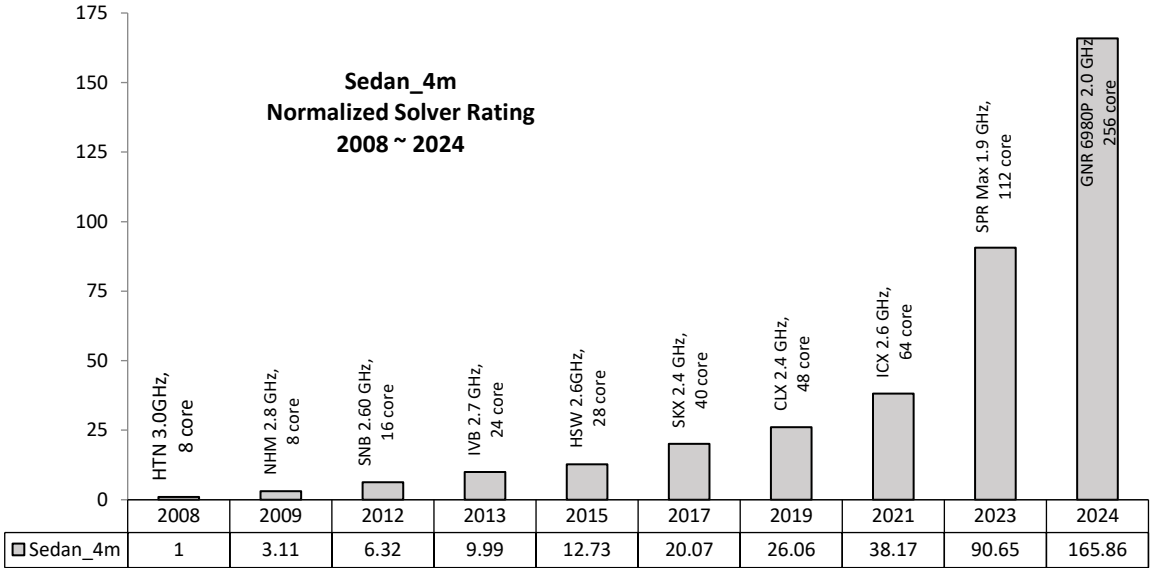


Fig. 2: Normalized solver rating of CFD simulation of flow over a sedan using Ansys Fluent from 2008 to 2024 along with HPC hardware advancements and Fluent scalability enhancements

B. Linear Solvers

Discretization of the governing equations will lead to a large set of linearized equations of the same mesh size. Point-implicit schemes, like the relaxation schemes of Gauss-Seidel (GS) and Incomplete LU Factorization (ILU) are the best candidates to rapidly remove the mesh-resolution dependent high-wavenumber error. But their convergence stalls when trying to use them directly reduce the global errors of lower wavenumbers using the same fine mesh.

Algebraic Multigrid (AMG) is an essential tool to accelerate the convergence of implicit solvers for large unstructured meshes [8,9]. AMG generates a sequence of coarser levels. The equations of coarser levels need no reconstruction of meshes, or re-discretization and computation of fluxes and source terms. The defect at a finer level represented by the global low-wavenumber errors is transferred to a coarser level through the ***restriction*** process. Those errors are considered as locally high-wavenumber errors at the coarse level, where the global corrections using relaxation methods at a coarse level are calculated exponentially more quickly compared to that at a finer one. Those corrections are communicated to the finer level through the ***prolongation*** process. The operators of restriction and prolongation used are based on the additive correction (AC) strategy initially formulated for structured meshes by Hutchinson and Raithby [10].

AMG plays a critical role in achieving parallel performance and scalability, making it a cornerstone of high-performance computing. In particular, handling the coarsest matrix is a key step in enabling efficient grouping and solving processes. However, as the number of cores increases, the collected matrix can grow substantially in size, posing significant challenges.

For instance, consider a scenario where only 10 rows remain, but the computation is distributed across 10,000 cores. In such a case, the collected matrix would expand to 100,000 rows. If these 100,000 rows were to be processed on a single core, it would create a severe bottleneck, undermining the benefits of parallelization.

To address this scalability challenge, we implemented a hierarchical approach to matrix collection. Instead of collecting all rows into a single core at once, the data is first grouped into smaller, more manageable sub-groups. These sub-groups are then progressively aggregated until the entire matrix is collected. This hierarchical method ensures that no single core is overwhelmed, thereby maintaining the efficiency of the computation.

For this hierarchical collection approach to be effective, architecture-aware partitioning is essential. Partitions within each sub-group must be connected topologically to facilitate further grouping, which is the key. This connectivity ensures further grouping of sub-partitions within each sub-group while minimizing ghost-layer communication overhead across machines. By reducing inter-machine communication costs and optimizing data locality, architecture-aware partitioning enhances the overall scalability and performance of the solver as demonstrated in Fig. 3. Architecture-aware partitioning will be discussed in more detail in Section B.

With these innovations — hierarchical matrix collection and architecture-aware partitioning — we successfully scaled AMG to handle systems with over 100,000. This approach not only overcomes traditional bottlenecks but also highlights the importance of tailoring algorithms to the underlying hardware architecture for achieving exceptional scalability.

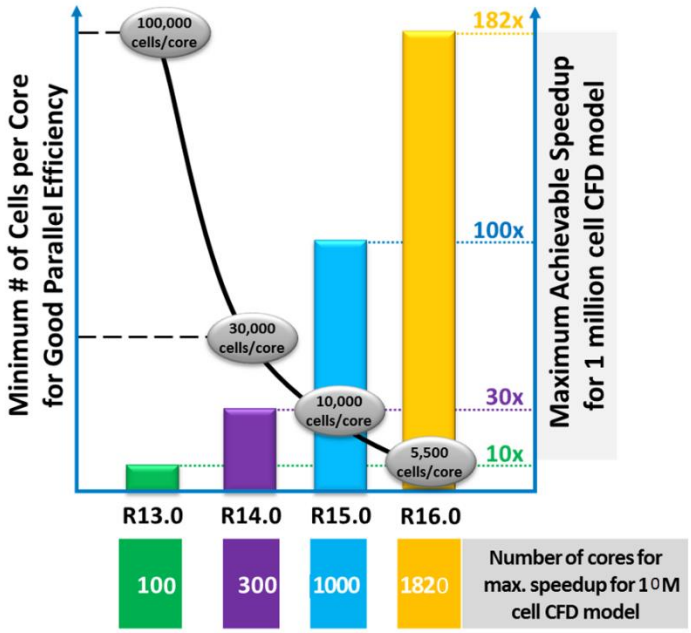


Fig. 3: With optimization of the linear solver at various versions R13.0~R16.0, Fluent could scale to 80% parallel efficiency with a smaller number of cells per core to achieve faster simulation turnaround time.

C. Domain Decomposition

Parallelization of the flow solver requires domain decomposition methods to efficiently divide, or to ***partition***, a general mesh among multiple processing units. The partitioning methods should minimize the deviation of the cell and face element per partition and the inter-communications between partitions.

The graph partitioning method is a cornerstone process of subdividing a general graph into smaller subgraphs, aiming to minimize the number of cut-edges between these subgraphs and to let every partition have the same number of vertices.

Ansys Fluent uses the MPI-based library, **ParMeTiS**, as the default method for partitioning and repartitioning unstructured graphs [11,12]. It is used in the auto-partitioning process when reading the mesh and automatically distributes the mesh partitions among the processing units. Also, it is used in the manual or automated repartitioning processes when attaining better load balancing after changes in either the mesh, as in Adaptive Mesh Refined (AMR), or after changes in the physical model setup.

Multi-constraint partitioning [13] provides additional controls in ParMeTiS besides the vertex/edge mesh-related criteria. Multi-constraint model-weighted partition add physical-based criteria to achieve optimal load balancing when simulating multi-physics flows, as in the computations of Conjugate Heat Transfer (CHT), Volume of Fluid model (VOF) of immiscible fluids, Discrete Phase Model (DPM) of partition, or In-Situ Adaptive Tabulation (ISAT) to accelerate detailed stiff chemistry in reacting flows [8,9]. The weights of additional constraints, added to the cell elements, correspond to the anticipated computational load of the newly introduced physics models. For instance, in CHT model, the associated fluid cells weights are larger than those of solid ones. As energy equation is solved at the solid and fluid cells, where flow computations are conducted on the fluid cells only. Similarly, the DPM weight of a cell is proportional to the number of enclosed particles.

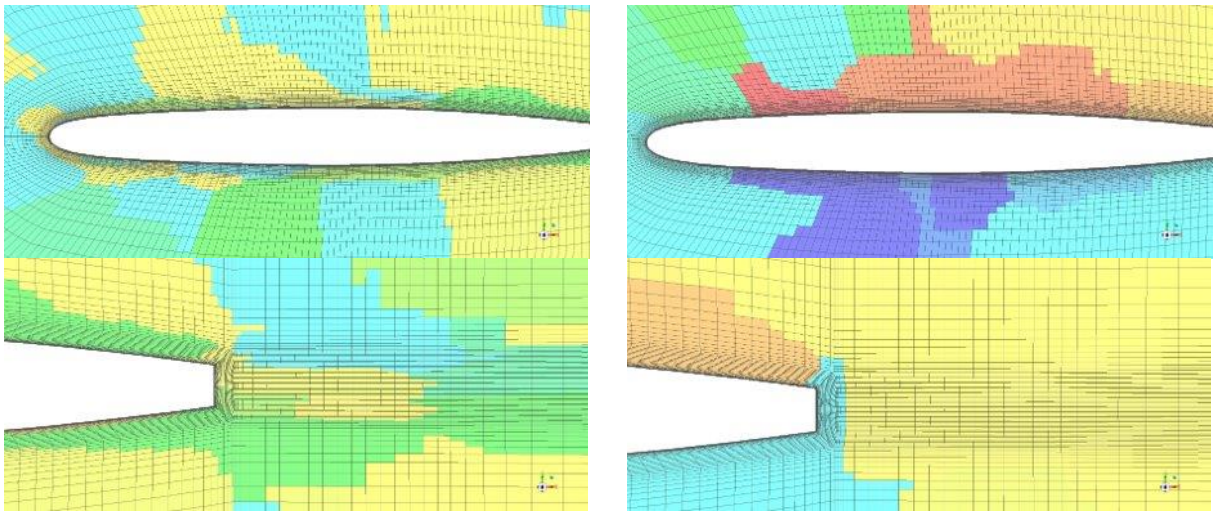
Cell-Encapsulated Partitioning is adopted in Fluent to enhance parallel convergence and scalability independently of the mesh and partition topologies. Cell grouping, using **Laplace smoothing** [8,9], takes place before partitioning to collect all cells of high strength factor represented by geometric-based Laplace coefficients. All adjacent cells sharing stretched faces of an aspect ratio above a given threshold are grouped together as one vertex in the graph. The multi-constrained weights of the cell group are added up to the vertex weight. This will ensure rapid parallelized AMG coarsening as discussed in Section A.

Also, Fluent provides additional functionality to encapsulate cells containing poor-quality mesh elements. Partitioning of such cell groups will prevent problematic face elements from existing along partition interfaces as shown in Fig. 4 and 5. Therefore, it will streamline the AMG hierarchical matrix collection across the cluster architecture, which is crucial to be considered when distributing the partitions across the cluster machines.

Architecture-aware partitioning is critical for high-performance cluster computing. By mapping original partitions to the cluster while considering network topology and latencies, as sketched in Fig. 6, systems can be optimized for better resource utilization and faster data exchanges.

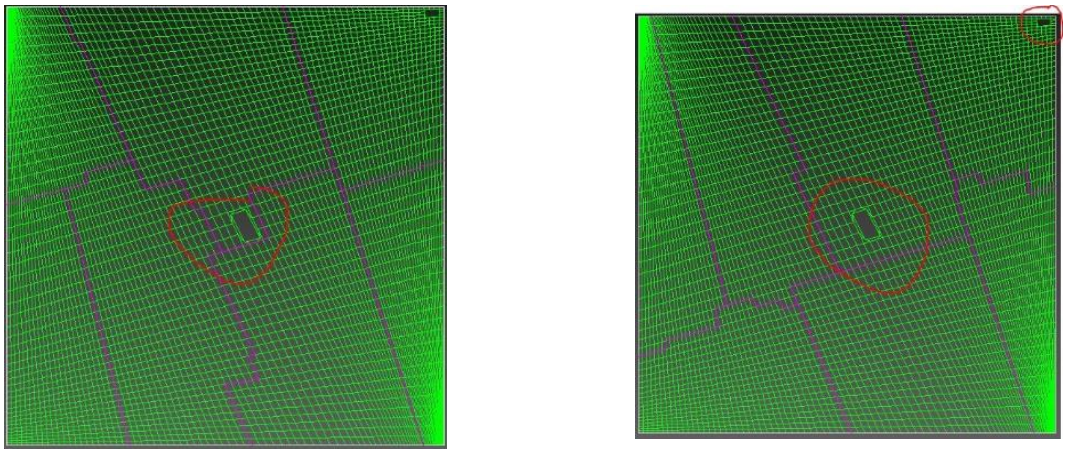
One key advantage of this approach is the reduction of inter-machine traffic. By localizing data exchanges or routing them through low-latency connections, the load on the network is significantly decreased. This ensures more efficient bandwidth usage, critical for maintaining high solver throughput.

Such optimization is especially beneficial in environments with slow interconnects or large-scale clusters, where latency and network inefficiencies can become major bottlenecks. By addressing these issues, the performance of the system improves substantially.



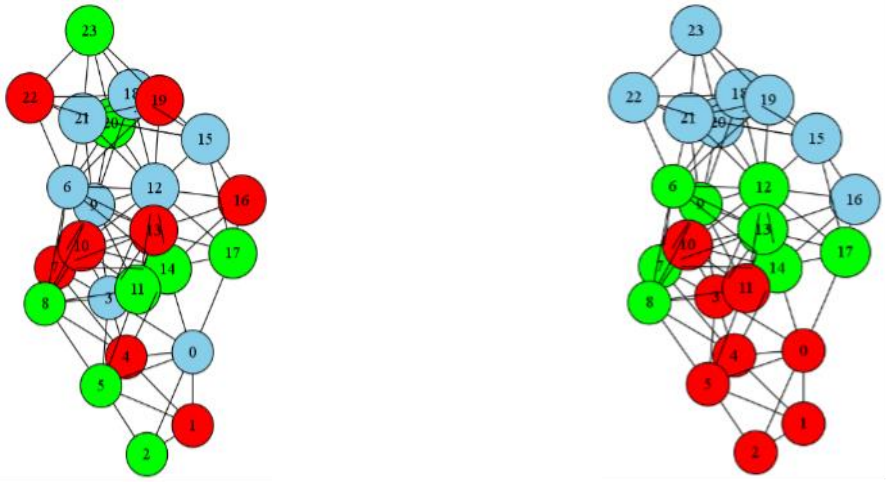
(a) **(b)**

Fig. 4: Contours of active partitions; (a) original partition topology; (b) after repartitioning with cell encapsulation using Laplace smoothing



(a) **(b)**

Fig. 5: Lines of partitions interface; (a) original partition topology; (b) After repartitioning with cell encapsulation using Laplace smoothing and poor-mesh elements grouping



(a) **(b)**

Fig. 6: Partition Graph: 3 machines, 8 cores each Colors indicate machines; (a) original; (b) after mapping with improved clustering of processes to machines

D. I/O and Lightweight Workflow

By default, Ansys Fluent writes case and data files in the Common Fluids Format (CFF). CFF files offer improved read and write performance, and they can be postprocessed in Ansys CFD-Post and Ansys EnSight, as well as in third-party products. A case file contains a mesh and settings for a specific problem. Solution data are typically saved in a separate data file related to the case file where the mesh and settings are stored.

The CFF format is based on the HDF5 format [14], which is an open-source hierarchical data format and a library of APIs that provides near-random access to any dataset in both serial and parallel processing environments. All settings are saved as formatted strings in multiple groups of CFF files. **Heavy-weight** data such as mesh connectivity and coordinates are stored as a 1D or 2D array in the HDF5 dataset. Additional **lightweight** data (such as minimum and maximum IDs, total number of elements/subgroups, and so on) are added to the groups and dataset as attributes.

As a part of preprocessing, setting up the simulation, which includes defining boundary conditions, solver numerics and solution monitors or animations etc.is often interactive. This process typically involves loading the mesh into the solver, which can be both memory-intensive and time-consuming, particularly for cases with hundreds of millions or billions of mesh elements, even with robust HPC resources. To address this challenge, we developed a streamlined lightweight workflow specifically for Ansys Fluent. This workflow minimizes CPU and memory usage by allowing users to configure simulation settings through the standard Fluent GUI/TUI, without the need of loading the full mesh data. As a result, it significantly improves the solver responsiveness during problem setup. Notably, the user can use any PC or laptop for this purpose for any large case, eliminating the need for additional HPC resources.

The lightweight workflow prefers a single processor but also works in parallel. We compare the performance of reading and writing a case with 130 million elements between normal and lightweight workflows, as shown in Table 1. For reading, it is almost impossible to read such a large case with a single processor, but very swift in lightweight workflow with 70 times faster. Though it is not preferred to have the lightweight workflow in parallel, as shown in Table 1(a), Fluent loads the case settings 10 times faster with 120 cores, compared to the normal workflow. For writing the case file, lightweight workflow takes almost constant CPU time no matter how many processors are used, which is less than 2 seconds and is much faster than the normal workflow. The total memory usage by the solver in the lightweight workflow is usually less than 1 GB. With this feature, the user can easily set up a case from mesh, view or modify an existing case from a laptop or a desktop PC, eliminating the need for HPC resources when setting up the simulation.

	Normal (s)	Lightweight (s)	Speedup (-)
Single process	934	13.3	70
4 processes	524	11.3	46
120 processes	182	18.3	10

(a)

	Normal (s)	Lightweight (s)	Speedup (-)
Single process	423	1.3	325
4 processes	151	1.8	84
120 processes	39.4	2.0	20

(b)

Table 1: Comparison of IO performance for the normal and lightweight workflows of 130 million cell mesh under single process, 4-way parallel and 120-way parallel: (a) reading case; (b) writing case.

E. Robust Handling of Partitions Neighborhood for Moving Meshes

The Fluent partitions' neighborhood represents the additional cells and faces in the mesh in the parallel partitioning context. These additional mesh elements have two or more copies across the partitions and participate in the core solver only in one partition to compute updated physical fields while being used to prescribe these physical quantities on the partition boundary of the other partitions. Optimal parallel scaling during the solver phase critically depends upon the setup of these partitions neighborhood as these require regular updates across one or more partitions using the new field data computed on one partition. In the presence of moving meshes such as sliding, dynamic, morphing, etc. meshes, the neighborhood may change whenever mesh topology changes, causing an overhead for the parallel scalability. These aspects have impacted many practical applications in recent years when progressively higher scalability requirements were needed to achieve meaningful scalability in the overall workflow. To speed up such workflows, successive optimizations were implemented in multiple new releases over the past few years to help these applications meet their overall simulation requirements as shown in Fig. 7. These optimizations carefully leveraged the reusable aspects of the mesh and provided speed-ups for all cases. An improvement in the timing of updating the moving mesh grid has been observed, making the process up to three times faster.

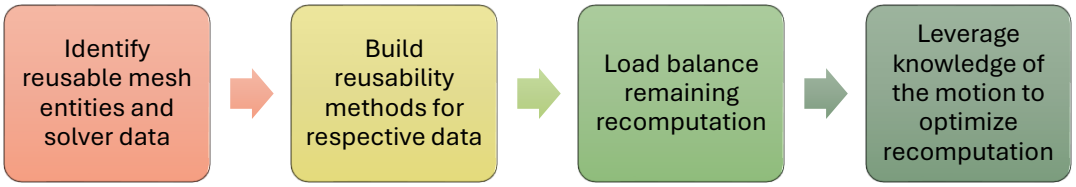


Fig. 7: General methodology adopted to optimize multiple major bottlenecks

F. Wall Distance Computation of Moving Meshes

Wall distance computation is essential in most CFD simulations where the turbulence model uses the perpendicular distances of the cell centroids to the nearest no-slip wall. These turbulence models capture the boundary layer phenomenon, which is critical to accurately predicting the flow characteristics near the walls, directly impacting the predicted physical structure of the flow field not just in the nearfield but also far from these walls. For external aerodynamics, among other applications, the accuracy of the wall distance highly impacts the overall solver results. To compute an accurate wall distance field, Fluent uses exact tree-based search algorithms to compute the exact wall distances free from errors. These algorithms are expensive, and for moving meshes again, these become a dominant bottleneck in the scalability of the overall simulation. Multiple optimizations were implemented for wall distance computations in the search tree construction and the lookup methods over the past few years to achieve substantial improvement in the parallel efficiency of the overall simulations. This is shown in Fig. 8 the wall distance calculation time for the brake pack case.

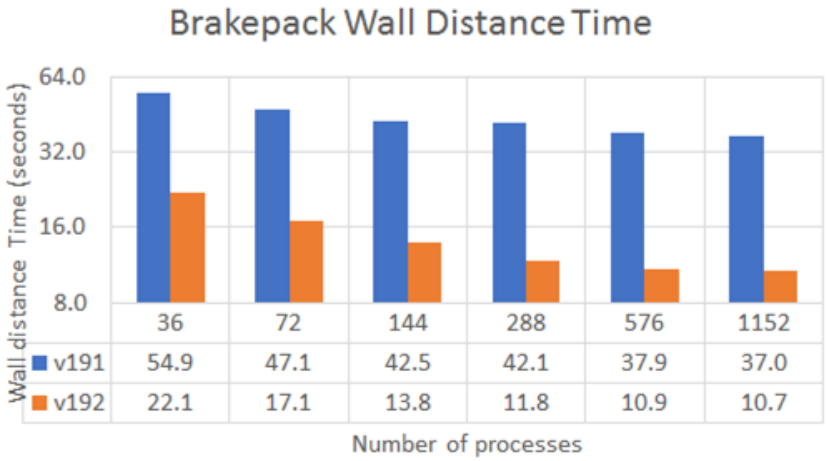


Fig. 8: Continuous improvements made in Ansys Fluent v192 compared to v191 and measured for an open case of brake pack

G. Hybrid MPI/OpenMP: A Unified Approach to Computational Efficiency

Message Passing Interface (MPI) is the ubiquitous communication exchanger for data transfer between different solver partitions in the distributed memory model. However, OpenMP (Open Multi-Processing) uses a shared memory model that enables running multiple threads to access a common memory space.

The HPC industry appears to be in a divergent phase regarding HPC hardware, including the CPU itself and the interconnect options available to the users. Faster interconnects are essential to scale HPC workloads like CFD with linear efficiency beyond a few machines by delivering bandwidths and latencies that allow the core solver to sufficiently exploit the CPU compute power available on each machine. The default setup of these libraries is automatically adjusted to efficiently run all large-scale studies. This requires regular investigations, including benchmarking studies over a wide range of CFD testcases with the newer hardware and software updates, to maintain the broad applicability of the defaults on the desired range of hardware types.

The **Hybrid MPI/OpenMP method** merges the strengths of shared and distributed memory models to create a robust, efficient computational framework. This dual approach leverages the best of both paradigms, ensuring seamless operation across multi-core systems while maintaining exceptional scalability and performance.

A standout feature of this hybrid model is its ability to fully exploit the potential of multi-core processors. By employing multiple threads, workloads are dynamically balanced, ensuring all processing units remain actively engaged. This maximizes CPU utilization, keeping all cores productive and achieving superior computational efficiency.

One of its most significant advantages lies in its simplicity. The model eliminates the need for cell migration or additional data collection processes, reducing overhead and boosting performance. By minimizing unnecessary data movement, the framework enhances both computational speed and system responsiveness. Particle tracking is one such use case.

Another noteworthy benefit is the reduced memory footprint when large, duplicated data blocks, such as those used for wall distance calculations, need to be shared among processes on a machine. This efficient memory usage is particularly advantageous for resource-intensive tasks like particle tracking, where scalability and precision are critical. Wall distance calculation is one such use case.

By integrating shared and distributed memory models, the hybrid MPI/OpenMP design delivers impressive performance, scalability, and resource optimization, making it an ideal solution for HPC. As shown in Fig. 10, Hybrid MPI/OpenMP approach is almost 4 times faster than using MPI only for the LM combustor case with a single injector [15].

H. Multi-domain Architecture

Multi-physics simulations need to address a broad spectrum of temporal and spatial scales. Using a single computational domain for discretizing these models can result in stiff linear systems of equations due to the significant disparity in eigenvalues. This can severely impact the solver's robustness, convergence, and parallel scalability. On the other hand, managing multi-physics numerical models within a multi-domain architecture that orchestrates concurrent co-simulations across multiple computational domains offers an effective solution to maintain solver accuracy and performance.

Multi-domain architecture (MDA) in Ansys Fluent refers to an umbrella of workflows for multi-physics co-simulations, including discrete phase model (DPM) for particle tracking, conjugate heat transfer across fluid/solid interface and aerooptics [16]. DPM-based particle tracking problem is considered here as an example to demonstrate the significant impact of using a multi-domain approach to enhance load balancing and performance.

Figure 7(a) illustrates the CFD computational domain with particles enclosed by mesh cells and partitions. In a single domain approach, both CFD and DPM calculations share the same computational domain, as shown in Fig. 9(a), with load balancing considering only the additional constraint of the number of particles per cell, as mentioned in Section B. However, the load balancing of the hybrid MPI/OpenMP approach used by DPM is constrained to each machine and does not account for multiple machines, thereby negatively affecting overall performance scalability.

Figure 9(b) illustrates the DPM domain, which is an independently created second domain with a distinct partition topology dedicated to DPM calculations. Cells containing particles are evenly distributed among machines and stored in the primary ranks, marked in red in Fig. 9(c). This distribution achieves optimal load balancing for the hybrid MPI/OpenMP approach used in particle tracking. CFD cells without particles are equally distributed across secondary ranks in each machine, ensuring that each machine receives a fair share of the particle tracking load. By concentrating particles in one partition per machine, the overhead of small threads is alleviated. As shown in Fig. 10, DPM second domain with hybrid MPI/OpenMP employment is almost 2 times faster than the hybrid approach using a single domain.

However, creating a second domain necessitates additional data exchange between the two computational domains. And it increases memory usage by approximately 50% and that should be considered when adopting this approach for large cases.

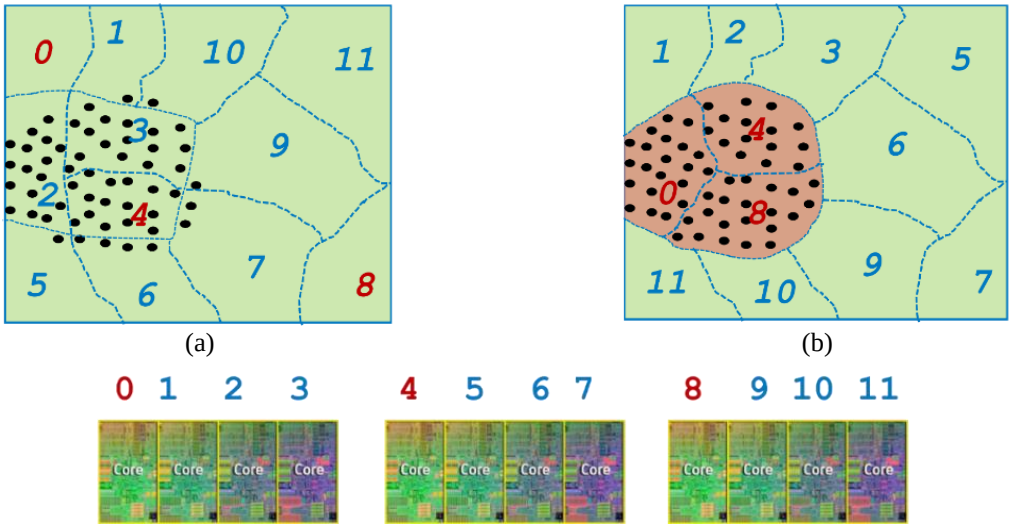
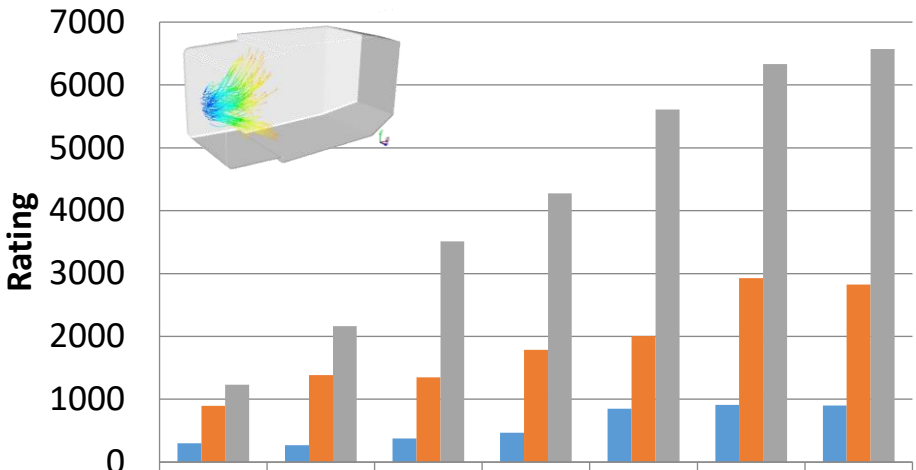


Fig. 9: Illustration of Multi-domain approaches in DPM-based particle tracking for load balancing among cluster machines: (a) CFD root domain; (b) DPM second domain; (c) cluster machines with primary ranks marked in red



	16	32	64	128	256	384	512
MPI	298.449	268.197	375.727	468.605	847.105	912.465	901.627
Hybrid	896.412	1383.59	1348.14	1785.95	2003.01	2925.04	2822.73
2Domain	1232.1	2161.51	3510.01	4276.93	5609.42	6333.05	6572.29

Fig. 10: LM Combustor, Single injector rig, Experiments by Joshi et. al [16], Particle evaporation, 246K cells, 1M particles, Hybrid method is about 4X as fast as the MPI only method. DPM 2nd Domain is about 2X+ as fast as Hybrid MPI/OpenMP

II. Showcases of Large-Scale Simulations

A. Jet Engine Gas Combustor

CFD simulation using Ansys Fluent for a generic gas combustor case of 830 million cells set a new world record for HPC scalability [1]. Ansys collaborated with SGI to successfully run this testcase on 145,152 CPU cores, reducing the solver wall clock time from 20 minutes on 1,296 cores to just 13 seconds. This collaboration highlights the efficiency and scalability of Ansys Fluent on SGI's ICE XA supercomputer platform, demonstrating an overall scaling efficiency of 83%, as shown in Fig. 11. The same case was studied in a collaboration with the High-Performance Computing Center Stuttgart (HLRS) and Cray, and Ansys Fluent shows scalability up to 172,032 cores on the Cray XC40 supercomputer, hosted at HLRS, running at 82% efficiency [2].

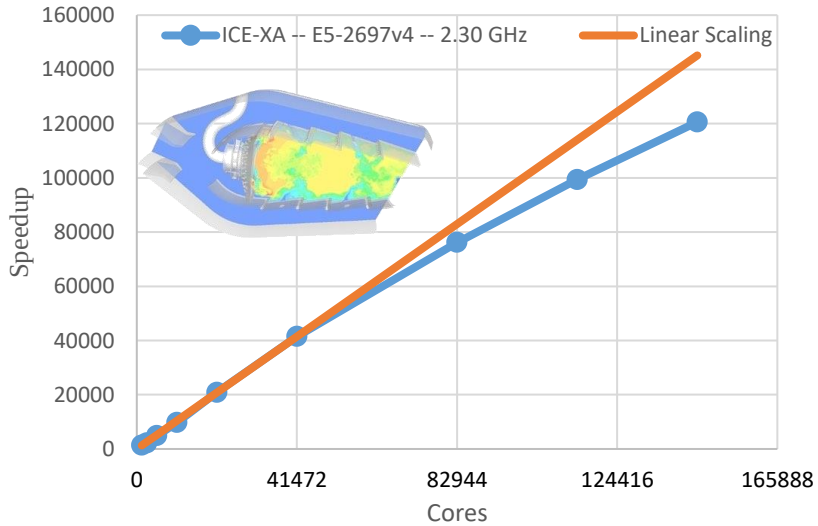


Fig. 11: Simulation of a combustor case with 830 million cells using Fluent R17.2 run on NCAR cluster "Cheyenne" with 145,152 cores achieved 83% parallel efficiency [13].

B. Gas Separation Vessel

Ansys, Saudi Aramco, and the King Abdullah University of Science and Technology (KAUST) have set a new supercomputing record by running Ansys Fluent to 200,000 processors on Shaheen II, a Cray XC40 supercomputer at KAUST [3]. The test case was benchmarked using a mesh of 1.6 billion grid cells and complex multi-phase flow models for four-phase gas separation (gas, oil, water, and solids). Population balance was used to simulate bubbles. Fluent demonstrated excellent scalability up to 68,000 cores, as shown in Fig. 12. This capability enables organizations to make critical and cost-effective decisions faster and increases the overall efficiency of oil and gas production facilities.

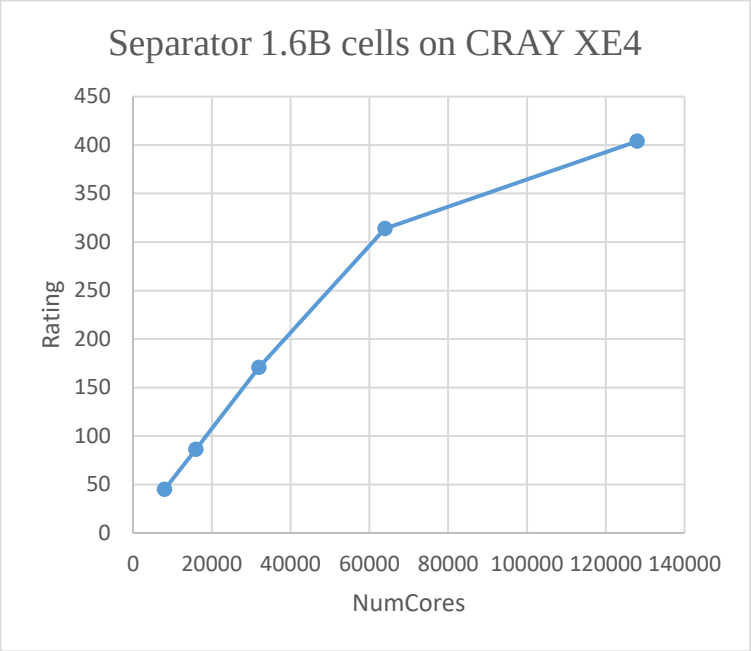


Fig. 12: Scalability of the gas separation vessel case of 1.6 billion grid cells

C. Full Pelotons of 121 Cyclists

Blocken et al. [4] studied the wake flow topology of cyclists and its impact on aerodynamic drag in pelotons. Using Ansys Fluent and RANS equations, they simulated two 121-cyclist configurations, validated with wind tunnel experiments. Each simulation on a Cray XC-40 supercomputer utilized a mesh of approximately 3 billion cells and required 54 hours [4-6]. As shown in Fig. 13 and 14, results showed significant drag reductions for mid-rear cyclists (5–10% of isolated drag) due to sheltering by others. The leading cyclists experienced higher drag but still benefited from a reduction due to the subsonic upstream disturbance created by the cyclists behind. This research underscores the aerodynamic benefits of pelotons, aiding optimal cyclist positioning and equipment design for improved performance in competitive cycling.

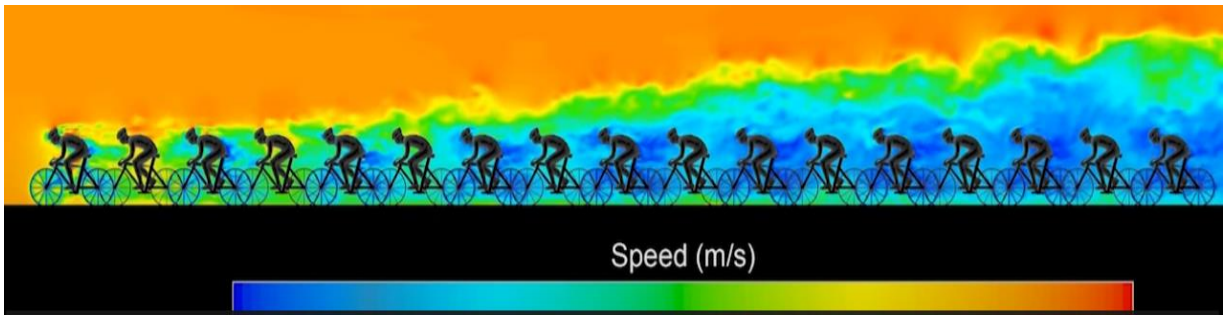


Fig. 13: CFD simulation of 3 billion cells of full peloton of 121 (configuration A) from Blocken et al. [4]: velocity magnitude visualization.

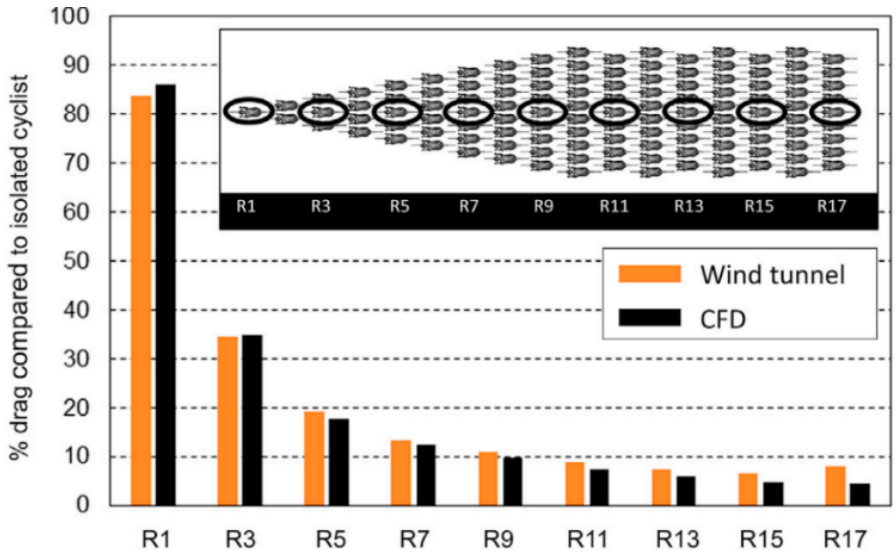


Fig. 14: Wind tunnel and CFD results show the drag reduction percentages for nine cyclists along the centerline, compared to an isolated cyclist at the same speed from Ref. [4].

III. Conclusion

The advancements in High-Performance Computing (HPC) and software development have significantly enhanced the capabilities of Computational Fluid Dynamics (CFD) simulations. This paper has demonstrated the scalability and robustness of Ansys Fluent in handling large-scale simulations, particularly through the example of aerodynamic simulations of gas turbine combustors, gas separators, and full pelotons. The improvements in parallelism techniques, linear solvers, domain decomposition, and I/O operations have enabled efficient handling of meshes exceeding one billion cells across thousands of processors. These developments not only validate the performance of Ansys Fluent but also open new avenues for exploring complex fluid dynamics problems that were previously unfeasible. Future and ongoing efforts will continue to focus on introducing and developing a native multi-GPU solver for further enhancements in the solver throughput.

IV. Acknowledgments

The authors extend their gratitude to the other members of the Ansys Fluent HPC team for their valuable contributions over the years. We also appreciate the computational resources generously provided by our partnership vendors, HPE Cray and Intel. Finally, we thank Ansys for their unwavering support and the opportunity to conduct such impactful studies.

V. References

[1] InsideHPC, "SGI and ANSYS Achieve New World Record in HPC," insideHPC [online news], URL: https://insidehpc.com/2016/09/ansys-world-record/ [retrieved September 25, 2016].

[2] Ansys Inc., "Ansys, HLRS And Cray Set New Supercomputing Record," [Online Press Release]. https://www.ansys.com/About-ANSYS/News-Center/11-15-16-ansys-hlrs-cray-set-new-supercomputing-record [retrieved on November 15, 2016]

[3] Ansys Inc., "Saudi Aramco, KAUST & Ansys Collaboration Breaks Record, Makes Safety and Efficiency Gains, Case Study," [Online resource]. https://www.ansys.com/-/media/ansys/corporate/resourcelibrary/casestudy/cs-cray-saudi-aramco-kaust.pdf [retrieved on July 18, 2017]

[4] Blocken, B., Van Druenen, T., Toparlar, Y., Malizia, F., Mannion, P., Andrianne, T., Marchal, T., Maas, G.-J., and Diepens, J., "Aerodynamic drag in cycling pelotons: New insights by CFD simulation and wind tunnel testing," Journal of Wind Engineering and Industrial Aerodynamics, Vol. 179, 2018, pp. 319–337. doi:10.1016/j.jweia.2018.06.011

[5] Feldman, M., "Worlds Largest CFD Simulation Upends Conventional Wisdom in Bicycle Racing," TOP500 [online news and resources], URL: http://www.top500.org/news/worlds-largest-cfd-simulation-upends-conventional-wisdom-in-bicycle-racing/ [retrieved July 3, 2018].

[6] HPCwire staff, "Aerodynamic Simulation Reveals Best Position in a Peloton of Cyclists," HPCwire [online news], URL: http://www.hpcwire.com/2018/07/05/aerodynamic-simulation-reveals-best-position-in-a-peloton-of-cyclists/ [retrieved July 5, 2018].

[7] Rupp, K. "40 Years of Microprocessor Trend Data," Karl Rupp's Blog [online blog], June 25, 2015. https://www.karlrupp.net/2015/06/40-years-of-microprocessor-trend-data/ [retrieved on June 25, 2015].

[8] Ansys Inc., 2024. Ansys Fluent 2024R2, Fluent Theory Guide. Ansys Inc., Canonsburg, PA.

[9] Ansys Inc., 2024. Ansys Fluent 2024R2, Fluent User's Guide. Ansys Inc., Canonsburg, PA.

[10] Hutchinson, B. R., and Raithby, G. D., "A Multigrid Method Based on the Additive Correction Strategy," Numerical Heat Transfer, Vol. 9, 1986, pp. 511–537.

[11] Karypis, G., Schloegel, K., and Kumar, V., "PARMETIS: Parallel Graph Partitioning and Sparse Matrix Ordering Library," University of Minnesota. 1997. Retrieved from http://conservancy.umn.edu/items/f479f2bc-6470-4ed4-9481-bbcba60e2e7b

[12] Karypis, G., and Schloegel, K., "PARMETIS, Parallel Graph Partitioning and Sparse Matrix Ordering Library Version 4.0," University of Minnesota, Department of Computer Science and Engineering, Minneapolis, MN 55455, August 2011.

[13] Karypis, G., and V. Kumar, "Multilevel Algorithms for Multi-Constraint Graph Partitioning," SC '98: Proceedings of the 1998 ACM/IEEE Conference on Supercomputing, Orlando, FL, USA, 1998, pp. 28-28, doi: 10.1109/SC.1998.10018.

[14] The HDF Group, "Hierarchical Data Format, version 5," [Computer software]. URL: https://github.com/HDFGroup/hdf5.

[15] Joshi, N.D., Epstein, M.J., Durlak, S., Marakovits, S., and Sabla, P.E. "Development of a Fuel Air Premixer for Aero-Derivative Dry Low Emissions Combustors," Proceedings of the ASME 1994 International Gas Turbine and Aeroengine Congress and Exposition. Volume 3: Coal, Biomass and Alternative Fuels; Combustion and Fuels; Oil and Gas Applications; Cycle Innovations. The Hague, Netherlands. June 13–16, 1994. V003T06A011. ASME. https://doi.org/10.1115/94-GT-253.

[16] Kamel, M. S., Dragojlovic, Z., Viti, V., Mercado, F., Jia, R., and La Cava, S., "Aero-Optical Simulation in Ansys FLUENT; Application and Validation," AIAA Paper 2024-1029, AIAA SciTech Forum, National Harbor, MD, Jan. 2024. https://doi.org/10.2514/6.2024-1029.