Eleventh International Conference on Computational Fluid Dynamics (ICCFD11), Maui, Hawaii, USA, July 11-15, 2022 **ICCFD11-3803**

GPU-Based HPC and AI Developments for CFD

S. Posey*, J. Luitjens*, O. Hennigh* and S. Oberlin*
Corresponding author: sposey@nvidia.com

* NVIDIA Corporation, USA.

Abstract: Current trends in computational fluid dynamics (CFD) include the use of graphics processing units (GPUs) as parallel co-processors to CPUs in order to accelerate numerical operations and algorithms common to CFD solvers. The first topic will examine advances in GPU method development for various CFD software including FUN3D from NASA and OpenFOAM from OpenCFD. The second topic will introduce novel CFD methods in artificial intelligence (AI) capable of encoding the Navier-Stokes equations into physics-informed neural networks (PINNs), while being agnostic to geometry, or initial and boundary conditions. Examples will include NVIDIA use of such techniques applied in electronics cooling design.

Keywords: GPU, HPC, AI, Computational Fluid Dynamics, Artificial Neural Network.

1 Introduction

Efficient use of computational resources and CFD simulation turn-around times are critically important factors behind engineering decisions to expand CFD technology to support more product design. Recent developments in GPU-based high-performance computing (HPC) and AI have improved computational speeds by orders of magnitude for a broad range of CFD simulations relevant to engineering practice.

The continual increase in processor speeds is limited due to power and thermal constraints, and to achieve gains in performance without increasing clock speeds, application software parallelism of numerical methods must be implemented. Other approaches can include deployment of artificial neural networks (ANNs) to emulate physical processes that can be coupled to numerical CFD solvers, or the use of a physics-informed neural network (PINN) that can emulate the full PDE solution of conventional numerical solvers. For these methods, parallelism can come in the form of task parallelism, data parallelism, use of deep learning frameworks, and a combination of these with the use of GPUs to accelerate conventional multi-core CPU computations for CFD simulation.

2 HPC Developments

HPC systems with GPUs provide significant increases in levels of parallel processing for CFD software that is developed using GPU programming models such as CUDA, OpenACC, or various API approaches. The programming strategies of choice can depend on several factors, such as availability of suitable GPU-based libraries including at a linear solver level, performance portability requirements for cross-platform deployment, and execution profile characteristics.

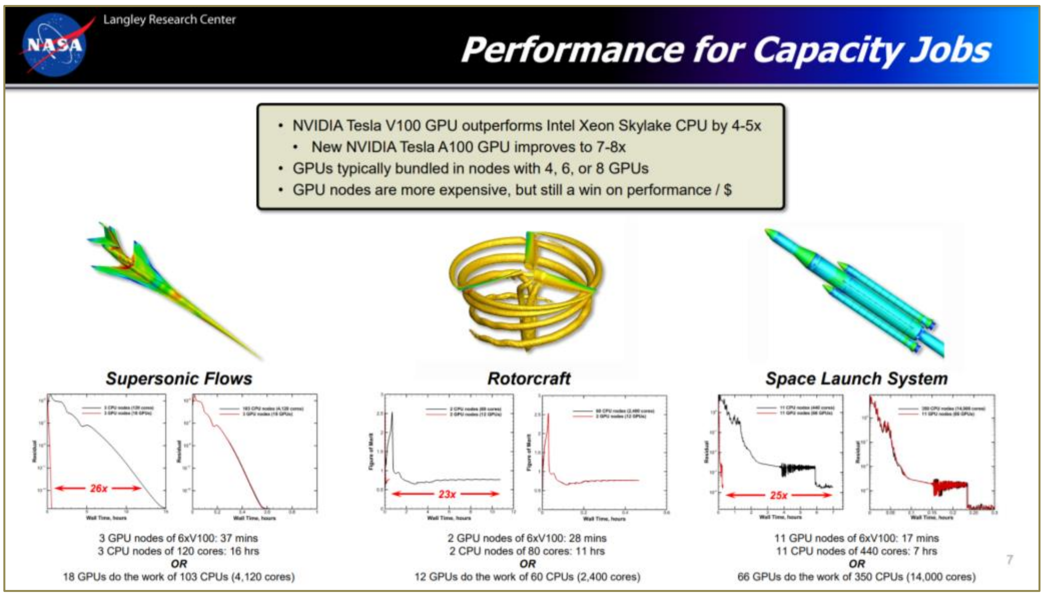
Developments in GPU-parallel CFD achieve substantial speedups from a fine grain, or second-level parallelism under an existing distributed memory, or first-level ‘scalable’ parallelism. For most GPU implementations in CFD, the focus is on implicit sparse iterative solvers whereby linear algebra matrix operations that are normally processed on CPUs, are off-loaded to the GPU for acceleration, resulting in an overall simulation speedup. GPU acceleration for end-to-end simulation can be limited however if the GPU focus is the linear solver only, which might represent ~50% of the total execution profile and therefore yield a "best-case" overall speedup of 2x. For more substantial overall speedups, non-solver code such as matrix assembly and construction should be included in GPU developments.

Iterative solvers have become the standard for most CFD software owing to their computational and storage efficiency. In order to aid the adoption of GPUs for applied CFD with the off-load computing model and assist in the CFD community's broad range of GPU interests, NVIDIA continues to invest in high performance iterative solvers in several library developments:

- cuBLAS lib of basic linear algebra subroutines (BLAS) for dense matrices.
- cuSPARSE lib of kernels and solver components for sparse matrices with a variety of formats.
- Thrust, an open-source C++ template library of high performance parallel primitives.
- AmgX complete iterative solver lib with emphasis on multigrid solvers and smoothers.

GPU implementations of CFD software FUN3D developed by NASA and OpenFOAM developed by ESI-OpenCFD have made use of one or more GPU programming methods and GPU-based libraries, and have reported substantial computational acceleration performance:

- FUN3D: GPU developments in CUDA where 1 x A100 GPU achieves equivalent performance of up to 8 x Skylake CPUs for a range of "capacity" CFD simulations as shown in Figure 1.


Langley Research Center

Performance for Capacity Jobs

- NVIDIA Tesla V100 GPU outperforms Intel Xeon Skylake CPU by 4-5x
  - New NVIDIA Tesla A100 GPU improves to 7-8x
- GPUs typically bundled in nodes with 4, 6, or 8 GPUs
- GPU nodes are more expensive, but still a win on performance / $

Supersonic Flows

3 GPU nodes of 6xV100: 37 mins
3 CPU nodes of 120 cores: 16 hrs
OR
18 GPUs do the work of 103 CPUs (4,120 cores)

Rotorcraft

2 GPU nodes of 6xV100: 28 mins
2 CPU nodes of 80 cores: 11 hrs
OR
12 GPUs do the work of 60 CPUs (2,400 cores)

Space Launch System

11 GPU nodes of 6xV100: 17 mins
11 CPU nodes of 440 cores: 7 hrs
OR
66 GPUs do the work of 350 CPUs (14,000 cores)

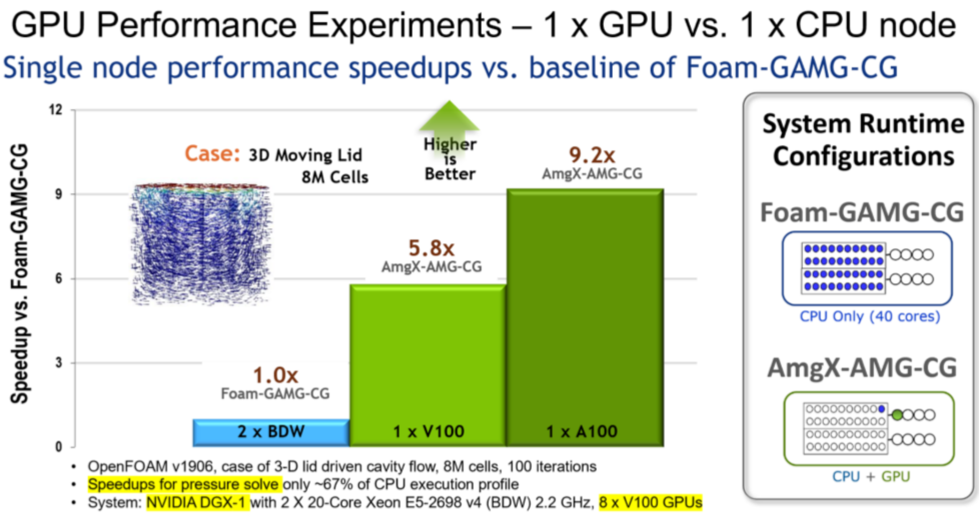
7

Figure 1: NASA FUN3D performance for selected "capacity" jobs on NVIDIA GPUs

- OpenFOAM: GPU developments using the AmgX library that demonstrate speedups single GPU (V100) over single CPU, and intranode strong scaling efficiency of 65% for a 25M cell model.

Members of the OpenFOAM HPC Technical Committee developed an add-on library to the standard OpenFOAM software named PETSc4FOAM that permits the plug-in of external solvers to OpenFOAM that conform to PETSc formats. NVIDIA further developed an external solver for GPU offload of OpenFOAM based on the AmgX solver library that was first introduced in 2012. In the initial NVIDIA implementation, the OpenFOAM system matrix is copied from the CPU to the GPU, and the AmgX library applies an AMG preconditioner to a preconditioned conjugate gradient (PCG) linear solve for overall linear solver acceleration. Results are copied back to the CPU which completes the OpenFOAM job on the CPU as the end-user normally observes. This solution requires no code changes to the standard OpenFOAM software and only GPU accelerates those linear solver operations off-loaded to the external AmgX solver library through the PETSc4FOAM library interface.

AmgX development was extended for its integration with OpenFOAM for multi-GPU and multi-node computations. Initial experiments with standard benchmarks of the 3d lid-driven cavity case with 8M cells shown in Figure 2. demonstrate that AmgX can achieve as much as a 9x speedup of the pressure solver on an NVIDIA A100 GPU vs. OpenFOAM GAMG-PCG solve on an x86 CPU server node with dual-socket 20 core "Broadwell" CPUs.



- OpenFOAM v1906, case of 3-D lid driven cavity flow, 8M cells, 100 iterations
- Speedups for pressure solve only ~67% of CPU execution profile
- System: NVIDIA DGX-1 with 2 X 20-Core Xeon E5-2698 v4 (BDW) 2.2 GHz, 8 x V100 GPUs

Figure 2: OpenFOAM performance for 1 x GPU vs. 1 x CPU node on NVIDIA GPUs

With this community supported solution, frequent software updates that are made to community-based libraries like PETSc, NVIDIA AmgX, and PETSc4FOAM will ensure that OpenFOAM users naturally benefit from the latest system software, compilers, system and processor hardware architectures, and OpenFOAM future releases.

3 AI Developments

AI research has given rise to applications of physics-informed neural networks (PINNs) that leverage the underlying laws of physics, often described in the form of partial differential equations (PDEs), to solve forward, inverse/data-assimilation and model discovery problems. Advantages over traditional methods of solving PDEs include (i) usability: not requiring arduous meshing, (ii) speed: ability to solve multiple geometries simultaneously, (iii) scalability: embarrassingly parallel across clusters of GPUs, and (iv) expertise: ability to leverage training experience.

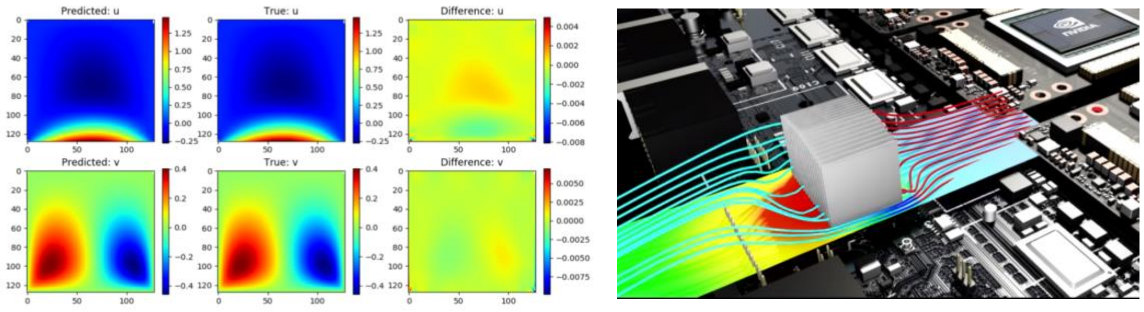


Figure 3: Validation of PINN predicted results vs. truth by an open source CFD solver for lid driven cavity case (left), and PINN predicted thermal results of an NVIDIA GPU heat sink design candidate.

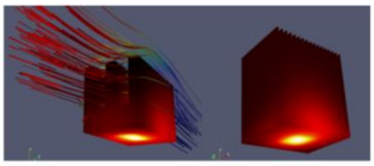
NVIDIA thermal management engineers applied PINN research in an evaluation to improve the design and effectiveness of heat sinks where thousands of design configurations could be analyzed within hours as opposed to weeks with traditional numerical CFD simulations. The PINN method provides a forward solution of parameterized, multi-physics problems, starting with only the geometry and other physical parameters like material properties, boundary conditions, etc., typical of any RANS-based CFD solver. Initial validation of the PINN approach is shown in Figure 3. along with early application of the PINN method to a candidate NVIDIA GPU heat sink design.

Standard neural networks that are driven by data alone are inadequate for modeling such engineering multi-physics problems on various geometries. Considerations must include specific features of a neural network architecture, such as sampling insensitivity, impact of the order of derivatives on the network structure, weighting the various PDEs for loss convergence acceleration, activation functions that do not reduce down to constants, or vanish when differentiated, and gradients and discontinuities due to geometric effects and considerations of local vs. global mass balance equations.

Requirements and tradeoffs were examined for this PINN evaluation vs. a conventional CFD approach. Results in Figure 4. provide final validation of quantities pressure drop and peak temperature of the actual heat sink geometry and set-up vs. the use of conventional numerical open source and commercial CFD solvers. HPC benefits are also provided for the inference phase of the PINN simulations which provides a single design evaluation in just 3 seconds, or near real-time per evaluation.

Numerical Validation of PINN vs. CFD

Physical Quantity	Neural Network Solver	OpenFOAM (pimpleFoam)	Commercial CFD Solver
Pressure Drop (Pa)	6.9	7.3	7.3
Peak Temperature (C)	79.8	81.3	80.2



HPC Benefits of PINN (Inference) vs. CFD

HPC Quantity	Neural Network Solver	Commercial CFD Solver	Factor
Total compute time for 2500 design evaluations	2 hrs*: 3s each on 1 x NVIDIA V100 GPU *Inference time only	104 days: 1h each on 1 x Intel Gold 6128 (SKL) 12 cores @ 3.4 GHz	1200x
Memory capacity (each eval)	0.216 GB	64 GB	296x
Output file size (each eval)	0.5 GB	2 GB	4x

https://news.developer.nvidia.com/nvidia-engineers-apply-ai-advances-in-fluid-mechanics

Figure 4: Validation of PINN predicted results vs. truth by open source and commercial CFD solvers for an NVIDIA GPU heat sink design, and HPC benefits of PINN (inference) vs. conventional CFD.

4 Conclusion and Future Work

As CFD simulation demands increase and motivate the need for more transients, higher-resolutions, and multi-scale, multi-physics simulations, GPUs will become an essential HPC technology. Based on current trends, GPU-based HPC combined with novel AI techniques will enable a level of applied CFD that can grow as a common practice to support engineering design and optimization procedures.

References

[1] Corrigan, A., Camelli, F., Lohner R., & Mut, F., 2010. Porting of an edge-based CFD solver to GPUs. In: 48th AIAA Aerospace Sciences Meeting Including The New Horizons Forum and Aerospace Exposition, number AIAA-2010-522. Orlando, FL, January 2010.

[2] M. Zubair, E. Nielsen, J. Luitjens and D. Hammond, "An Optimized Multicolor Point-Implicit Solver for Unstructured Grid Applications on Graphics Processing Units," 6th Workshop on

Irregular Applications: Architecture and Algorithms (IA3), Salt Lake City, UT, 2016, pp. 18-25.

[3] Walden, A., Nielsen, E., Nastac, G., “Accelerating FUN3D 13.7 Solutions Using GPU Hardware”, NVIDIA GTC 2021: https://fun3d.larc.nasa.gov/GPU_March_2021.pdf

[4] Spisso, I., Amati, G., HPC Comparison of Hypre Petsc vs Pstream as external linear algebra library for OpenFOAM : Project Presentation, ESI OpenFOAM Conference, Hamburg, 23-25 Oct 2018.

[5] S. Bnà, I. Spisso, M. Olesen, G. Rossi *PETSc4FOAM: A Library to plug-in PETSc into the OpenFOAM Framework* PRACE White paper

[6] Naumov, M., M. Arsaev, P. Castonguay, J. Cohen, J. Demouth, J. Eaton, S. Layton, N. Markovskiy, N. Sakharnykh, R. Strzodka et al. 2014. “AmgX: Scalability and performance on massively parallel platforms”. SIAM workshop on exascale applied mathematics challenges and opportunities. SIAM.

[7] M. Raissi, P. Perdikaris, and G. E. Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics* 378 (2019): 686-707.

[8] Xiaowei Jin, Shengze Cai, Hui Li, and George Em Karniadakis. Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations. arXiv:2003.06496, 2020.

[9] ICEnet Consortium: https://icenetcfd.github.io/website/

[10]NVIDIA Technical Blog: “NVIDIA Engineers Apply AI Advances in Engineering Physics Problems”, Jan 2020: https://developer.nvidia.com/blog/nvidia-engineers-apply-ai-advances-in-fluid-mechanics/.