AmgX GPU Solver Developments for OpenFOAM

Matt Martineau,[1] Stan Posey,[2] Filippo Spiga[3]

[1]*NVIDIA Ltd., Reading, UK, Developer Technology, mmartineau@nvidia.com*
[2]*NVIDIA Corporation, Santa Clara, USA, Program Manager, CFD, sposey@nvidia.com*
[3]*NVIDIA Ltd., Cambridge, UK, Developer Relations, fspiga@nvidia.com*

Current trends in high performance computing (HPC) include the use of graphics processing units (GPUs) as massively parallel co-processors to CPUs that can accelerate numerical operations common to computational fluid dynamics (CFD) solvers. This work will preview a new development aiming to accelerate OpenFOAM using the NVIDIA AmgX linear solver library that provides GPU support to the PETSc4FOAM library introduced during 2020 by members of the OpenFOAM HPC Technical Committee. Initial experiments with 8M cells demonstrate that AmgX on a V100 GPU can provide a 7x speedup of the pressure solve vs. an OpenFOAM GAMG-PCG solve on an x86 CPU server node with dual-socket 20 core "Broadwell" CPUs.

1. Introduction

Efficient use of hardware system resources and improved CFD simulation turn-around times continue to be important factors behind engineering decisions to expand CFD as a technology to support product design. While in recent years the progress on CFD simulation techniques verses physical testing has been remarkable, the availability of conventional HPC clusters running CPU-based parallel solvers has not been enough to motivate a substantial industry shift towards high-fidelity CFD modelling and CFD-driven design optimization procedures.

GPU-parallel CFD achieves speedups over CPUs from additional fine grain, or second-level parallelism that conforms to existing CPU-based distributed memory, or first-level scalable parallelism. For most CFD implementations, the focus is on implicit sparse iterative solvers whereby linear algebra matrix operations that would be processed on the CPU are offloaded to the GPU for numerical acceleration resulting in an overall simulation speedup.

Since the introduction of GPU computing in 2007, NVIDIA has continued to invest in high performance solver libraries for applied CFD and related HPC applications:

- **cuBLAS:** library of basic linear algebra subroutines (BLAS) for dense matrices [1]
- **cuSPARSE:** library of kernels and solver components for sparse matrices [2]
- **Thrust:** open source C++ template library of high-performance parallel primitives [3]
- **cuSOLVER:** high-level solver package based on the cuBLAS and cuSPARSE libraries [4]
- **AmgX:** open-source iterative solver library with special focus on multigrid methods [5]

These libraries range from kernel-level single-node math routines typical of BLAS for vector and matrix operations, to multi-node parallel linear solver such as AmgX targeting multi-GPU multi-node system configurations.

2. AmgX for OpenFOAM

AmgX enables developers to offload the linear solver portions of their HPC applications to distributed NVIDIA GPUs using a simple C API, potentially unlocking considerable speedups in CFD simulations with minimal development cost. One primary advantage of AmgX is its flexible solver composition system, which supports nested combinations of various solvers and preconditioners that can be controlled by a configuration file, allowing users to fine-tune for a specific linear system without recompiling. The library exposes advanced solvers and preconditioners to cover a wide range of problem domains with features that include:

- AMG: Ruge-Steuben, Un-smoothed aggregation; V-, W-, F- cycles
- Krylov methods: PCG, GMRES, BiCGStab, and flexible variants.
- Smoothers: Block-Jacobi, Gauss-Seidel, incomplete LU, Polynomial, dense LU.
- Support for MPI and OpenMP; FP64 and FP32; Matrix formats CSR, COO, MM, etc.

Since the first release of NVIDIA AmgX during 2012, there have been at least 4 independent implementations (including the current work) of AmgX-based offload solvers for OpenFOAM. During NVIDIA GTC 2015, Landman [6] presented work on the Culises library that embedded AmgX for OpenFOAM. In 2017, Rathnayake [7] published work that implemented the OFAmgX wrapper library to enable use of AmgX solvers from OpenFOAM. This was inspired by a similar AmgX wrapper developed by Chuang and Barba [8] to add AmgX support to PETSc. Recently, Vratis [9] developed a plug-in solver SpeedIT that integrated AmgX for AMG preconditioning to the OpenFOAM CG solver and achieved GPU speedups for the 3D lid-driven cavity case.

Each of the previous OpenFOAM-AmgX implementations has demonstrated real potential but ultimately the OpenFOAM community were unable to fully benefit from AmgX. All previous implementations were built on outdated releases of OpenFOAM (v2.4 or earlier) and with the exception of Vratis, have not kept up with the latest innovations in the NVIDIA GPU ecosystem.

Most importantly, all previous implementations were developed without collaborative input from ESI OpenCFD or other OpenFOAM developers.

The key differentiation and point of strength for the current work is the increasing community effort behind the PETSc4FOAM library [10] released during 2020 by core developers of the OpenFOAM HPC Technical Committee [11]. Necessary and frequent software updates that occur to community-based and open source libraries like PETSc [12], 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.

2.1 Current Implementation

The PRACE white paper that introduced PETSc4FOAM [13] provided NVIDIA with the guidance needed to adapt AmgX as an external plug-in solver to OpenFOAM. For validation of the approach, the first step was to replicate results in the paper for CPU-based experiments of the 3D lid-driven cavity flow for both OpenFOAM and PETSc CPU-based solvers. The medium (M) test case of 200x200x200 for 8M total cells shown in Table 1 was used because it fit on a single V100 GPU with 16GB of memory.

Parameters	Test-case		
	S	*M*	*XL*
Δx (m)	0.001	0.0005	0.00025
N. of cells tot. (millions)	1	8	64
N. of cells lin.	100	200	400
ν (m^2/s)	0.01	0.01	0.01
d (m)	0.1	0.1	0.1
Co	1	0.5	0.25
Δt $(sec.)$	0.001	0.00025	0.0000625
N. of Reynolds	10	10	10
U (m/s)	1	1	1
n. of iter.	-	-	100

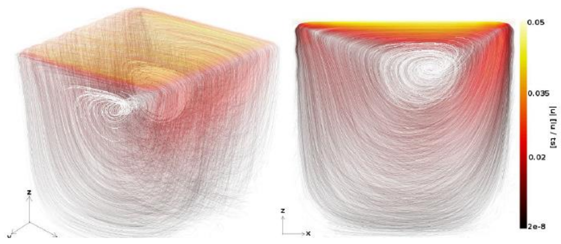

Table 1: Details of test case 3D lid-driven cavity flow

The next steps included GPU developments that embedded AmgX into PETSc4FOAM using the AmgXWrapper from Chuang and Barba [8], which adapts the AmgX interface to accept PETSc data structures. In order to maximize performance, an accelerated version of the LDU2CSR algorithm was developed that uses a radix sort to convert between LDU and CSR matrix formats but stores a permutation array so that successive timesteps can convert with minimal overhead. Further, AmgXWrapper was extended to support shallow updates of matrix coefficients and the lightweight "re-setup", reducing the amount of memory traffic and setup workload. Lastly, AmgX was extended to support the custom OpenFOAM convergence testing, which uses a special normalization factor, as described in the PRACE paper.

2.2 Performance Experiments

Initial investigations targeted a single server node and single NVIDIA V100 GPU [14]. The system used was an NVIDIA DGX-1 comprised of the configuration shown in Figure 1, with 2 x 20 core "Broadwell" CPUs and 8 x NVIDIA V100 GPUs with 16 GB of memory each for a total of 128 GB of GPU system memory.

DGX-1 Feature	System Specifications
GPUs	8 x V100
CPUs	2 x 20 core E5-2698 v4
GPU memory	128 GB total system
GPU bandwidth	900 GB/s per GPU

Figure 1: Details of the test system NVIDIA DGX-1

The actual benchmark configurations are shown in Figure 2. For CPU-only execution with the OpenFOAM and PETSc solvers, the full 2 x 20 cores and 40 MPI ranks were used, and for GPU execution only a single GPU and single CPU core were used.

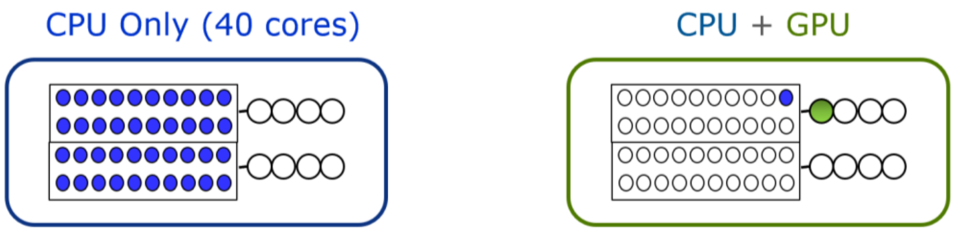

Figure 2: Schematic of the benchmark configurations for the DGX-1 system

Similar to procedures described in the PRACE paper, the AmgX performance experiments focused on preconditioned conjugate gradient (PCG) and investigated the performance impact of different preconditioners. The experiments first replicated OpenFOAM and PETSc results from the PRACE paper to first establish CPU-only baselines for proper CPU verses GPU comparisons. Each performance test of the 8M cell model was run for 20 iterations, collecting and analyzing both the total wall-time and the time spent only in the pressure solve. Figure 3 provides a comparison of the speedups achieved for the pressure solve relative to the FOAM-GAMG-PCG baseline results.

The experiments included the following solution procedures and solver configurations:

- CPU-only standard OpenFOAM procedures with FOAM-GAMG-PCG and FOAM-DIC-PCG
- CPU-only solutions from PETSc through the PETSc4FOAM library with PETSc-ICC-CG, PETSc-AMG-CG, and PETSc-AMG-CG-c (+caching, details described in [13]).
- GPU solutions from AmgX through the PETSc4FOAM library with AmgX-JAC-CG, AmgX-AMG-CG, and AmgX-AMG-CG-c (+caching, details described in [13]).

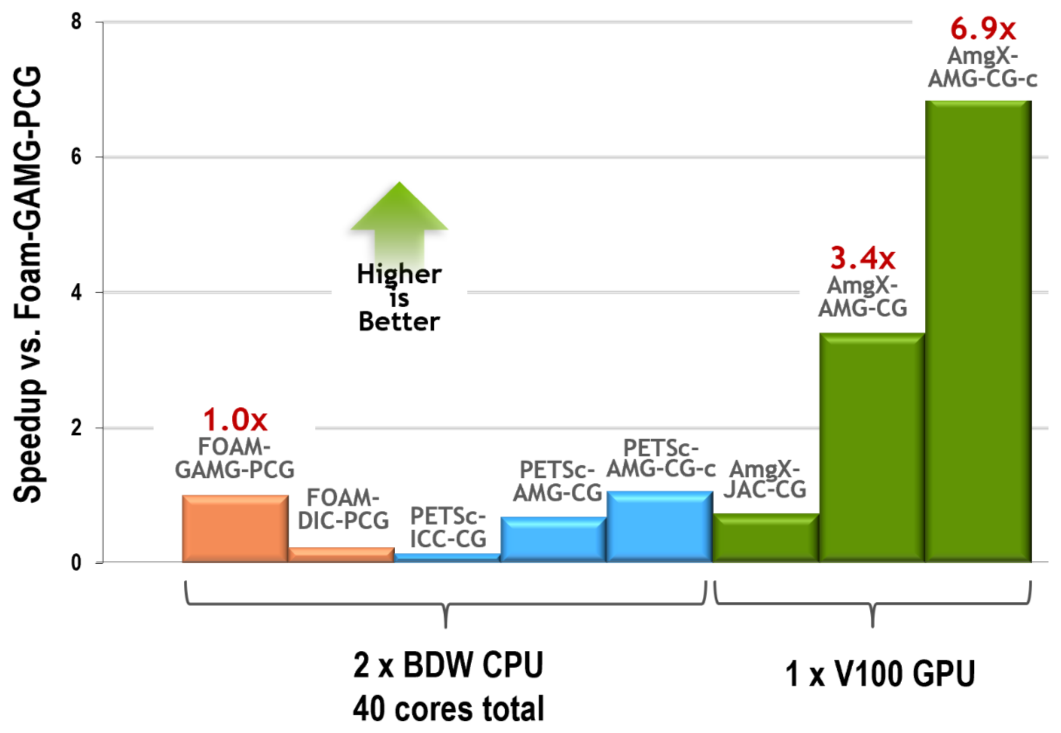

Figure 3: Results of pressure solve comparisons for OpenFOAM, PETSc, and AmgX

Results demonstrate that the best CPU-only pressure solve is obtained using PETSc-AMG-CG-c which has a slight performance speedup over FOAM-GAMG-PCG. The PETSc-AMG-CG solver is described as the best approach overall in the PRACE paper, especially for its scaling performance on multi-node systems. The best GPU result is obtained using the AmgX-AMG-CG-c solver (AMG preconditioner with the CG solver with caching invoked). This solver combination provides a ~7x performance speedup over the baseline FOAM-GAMG-PCG obtained using 40 cores on the CPU node. This means a single V100 has roughly a 14x performance advantage verses a single socket 20 core Broadwell CPU for the pressure solve.

3. Future Work

The initial implementation described is undergoing improvements towards generalization of the method and extensions to distributed parallel computing, to reach production-ready status for a future OpenFOAM release. Strong scaling benchmarks are planned and underway, including experiments with the larger (XL) version of the test case, and others. In partnership with the HPC Technical Committee there are plans to evaluate the AmgX solver performance using the latest NVIDIA A100 Tensor Core GPU based on the Ampere architecture [15]. Further GPU speedups will be observed owing to the A100 improved memory bandwidth rating of 1555 GB/s, which is 1.7x faster than V100.

References

1. NVIDIA cuBLAS — NVIDIA (2020). https://docs.nvidia.com/cuda/cublas/

2. NVIDIA cuSPARSE — NVIDIA (2020). https://docs.nvidia.com/cuda/cusparse/

3. NVIDIA Thrust — NVIDIA (2020). https://docs.nvidia.com/cuda/thrust/

4. NVIDIA cuSOLVER — NVIDIA (2020). https://docs.nvidia.com/cuda/cusolver/

5. NVIDIA AmgX — Community (2020). https://github.com/NVIDIA/AMGX

6. B. Landmann, K. Wieczorek, S. Bachschuster. AeroFluidX: A Next Generation GPU-Based CFD Solver for Engineering Applications, Presentation at NVIDIA GTC Conference, San Jose, CA (2015). https://on-demand.gputechconf.com/gtc/2015/presentation/S5189-Bjoern-Landmann.pdf

7. T. Rathnayake, S. Jayasena, M. Narayana. OPENFOAM ON GPUS USING AMGX, SpringSim-HPC 2017, April 23-26, Virginia Beach, VA, USA, ©2017 Society for Modeling & Simulation International (SCS)

8. P.-Y. Chuang, L. Barba. Using AmgX to Accelerate PETSc-Based CFD Codes, Presentation at NVIDIA GTC Conference, San Jose, CA (2016).

9. What is SpeedIT? – Vratis (2020). http://vratis.com/what-is-speedit/

10. Community / PETSc4FOAM external-solver · GitLab (2020). https://develop.openfoam.com/Community/external-solver

11. High Performance Computing (HPC) Technical Committee - OpenFOAM Wiki (2020). https://wiki.openfoam.com/High_Performance_Computing_(HPC)_Technical_Committee

12. Satish Balay and Shrirang Abhyankar and Mark F. Adams and Jed Brown and Peter Brune, and Kris Buschelman and Lisandro Dalcin and Alp Dener and Victor Eijkhout and William D. Gropp, and Dmitry Karpeyev and Dinesh Kaushik and Matthew G. Knepley and Dave A. May and Lois Curfman McInnes, and Richard Tran Mills and Todd Munson and Karl Rupp and Patrick Sanan, and Barry F. Smith and Stefano Zampini and Hong Zhang and Hong Zhang, PETSc Web page (2019). URL https://www.mcs.anl.gov/petsc

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

14. NVIDIA V100 — NVIDIA (2020). https://www.nvidia.com/en-us/data-center/v100/

15. NVIDIA A100 — NVIDIA (2020). https://www.nvidia.com/en-us/data-center/a100/

16. S. Posey, F. Pariente, Opportunities for GPU Acceleration of OpenFOAM, Tech. rep., 7th ESI OpenFOAM Conference, Berlin (2019). http://tiny.cc/4moalz

17. ESI-OpenCFD, User Guide: Solution and algorithm control (2020). http://tiny.cc/z657kz

18. OpenFOAM - Official home of The Open Source Computational Fluid Dynamics (CFD) Toolbox (2020). https://www.openfoam.com/