Scalable GPU accelerated simulation of multiphase compressible flow

Anand Radhakrishnan
Computational Science and Engineering
Georgia Institute of Technology
aradhakr34@gatech.edu

Henry Le Berre
Computer Science
Georgia Institute of Technology
hberre3@gatech.edu

Spencer H. Bryngelson
Computational Science and Engineering
Georgia Institute of Technology
shb@gatech.edu

***Abstract*—We present a strategy for GPU acceleration of a multiphase compressible flow solver that brings us closer to exascale computing. Given the memory-bound nature of most CFD problems, one must be prudent in implementing algorithms and offloading work to accelerators for efficient use of resources. Through careful choice of OpenACC decorations, we achieve 46% of peak GPU FLOPS on the most expensive kernel, leading to a 500-times speedup on an NVIDIA A100 compared to 1 modern Intel CPU core. The implementation also demonstrates ideal weak scaling for up to 13824 GPUs on OLCF Summit. Strong scaling behavior is typical but improved by reduced communication times via CUDA-aware MPI.**

***Index Terms*—GPU acceleration, exascale, multiphase flow, compressible flow**

I. Introduction

Multiphase compressible flow simulations are critical to understanding various physical problems such as breakup and formation of liquid droplets [1], bubble cavitation [2], [3], and shock wave attenuation [4]. Compressible flow algorithms consist primarily of vector operations, typically of low arithmetic intensity. This limits speedup on GPUs due to memory-bound kernels. Further, bidirectional communication is required between neighboring grid cells every time step for computing derivatives.

Here, we present strategies that we found particularly successful for large compressible flow simulations. The results are demonstrated on the multiphase solver MFC [5]. We show a 300-times speedup on an NVIDIA A100 GPU compared to an Intel Xeon Cascade Lake CPU core. The methodology scales up to 13824 on OLCF Summit at 97% efficiency.

II. Background

MFC uses interface capturing methods [6], [7] that also serve as governing equations. The equations are solved discretely via a finite volume method with high-order accurate WENO reconstruction, representing interfaces without spurious oscillations [8]. The algorithm uses an HLLC Riemann solver [9] and a Total Variation Diminishing (TVD) Runge–Kutta time stepper [10]. MFC is a Fortran90 codebase that offloads all compute kernels to GPUs via OpenACC 2.6. Near-ideal CPU scaling has already been demonstrated [5].

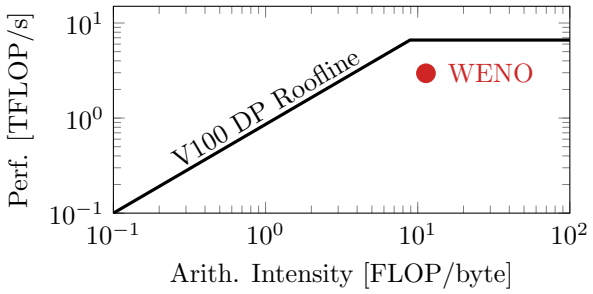

Fig. 1. Double precision (DP) roofline and performance for the most expensive compute kernel, WENO reconstruction, on OLCF Summit.

III. Results

A. Kernel Performance

As the discrete conservation laws act on a system of conservative variables, user-defined data types are employed throughout the code for convenience. Due to dimensional splitting in the process of flux addition, a temporary reshape of the input variables is necessary to ensure coalesced memory access. For this purpose, the use of multidimensional arrays resulted in a 6-times speed up of the most expensive kernel (WENO) on GPUs compared to user-defined derived types.

Performance improvements follow from metaprogramming techniques, including the use of a pre-processor, `Fypp`, that passes user inputs as parameters. These parameters allocate fixed-size private arrays in GPU kernels and are available at compile time. This strategy enables the use of CUDA local memory for efficient memory access. We can also use the fixed parameters to eliminate large conditional blocks in kernels, leading to improved times via judicious use of available GPU registers. This strategy alone results in 8- and 2-times speedups of the two most expensive kernels, which are associated with WENO and the Riemann solver, respectively.

Figure 1 shows the performance of the most expensive kernel in MFC, WENO, and the double precision (DP) roofline for the NVIDIA V100 GPU it ran on. The WENO kernel achieves 46% of V100 peak FP64 performance. This results from the high computational intensity and careful memory reuse in this kernel, resulting in a 500-times speedup for a 3D problem of size 8 million when using 1 NVIDIA A100 over 1 modern Intel CPU core.

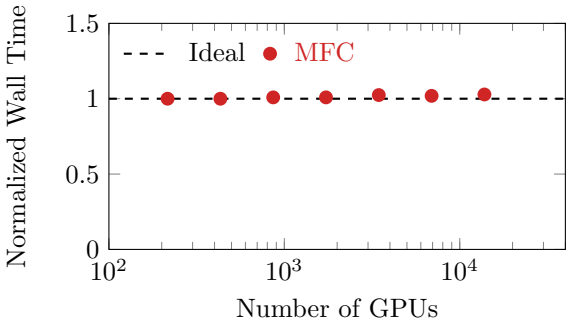

Fig. 2. Weak scaling on Summit for a 3D multiphase problem.

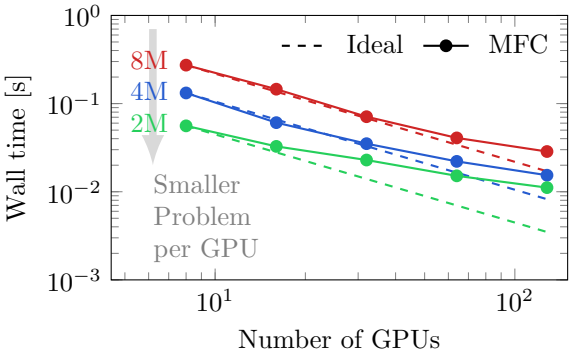

Fig. 3. Strong scaling on OLCF Summit. Labels are problem size per GPU.

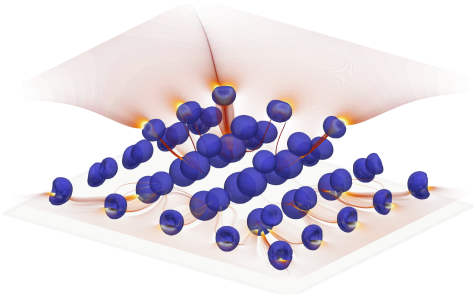

Fig. 4. Streamlines of a collapsing bubble cloud near wall.

Next, we probe performance on an NVIDIA A100 (OLCF Wombat) and V100 (Summit) GPUs by comparing performance against Intel Xeon Cascade lake (PSC's Bridges2) CPUs with compiler optimizations. We observe average wall-clock times and speedups for both GPUs on a 3D problem of size 8 million points, which is close to the memory limit of a V100 GPU. Results show that the algorithm is 1.72-times faster on an A100 when compared to a V100, resulting in better performance than one would anticipate between the two cards based on double precision performance limits (a factor of 1.24). This can be attributed to higher memory bandwidth (1.7-times), and faster GPU interconnects (2-times) on an A100 over a V100.

We also observe that a single A100 or V100 has 7- and 4.5-times faster execution time than two Intel Xeon Cascade lake CPUs (40 cores). Thus, on a PSC Bridges2 node with 8 V100 GPUs, we obtain a 22-times speedup compared to the two Intel CPUs on the compute nodes of the same supercomputer.

While computation is offloaded completely to the GPUs, one must still transfer data, including halo transfers and I/O data dumps. Halo transfers are insignificant on CPUs but constitute a meaningful (∼10%) portion of the overall wall-clock times on GPUs, even for large problems (8 million grid points in 3D). I/O transfers take up a significant amount of time on GPUs, equivalent to about ten time steps. However, their effect on overall simulation times for compressible flows is insignificant, as I/O is often required just once every thousand time steps.

B. Scaling

Figure 2 shows the weak scaling performance for a 3D multiphase problem of size 1 million points per GPU on OLCF Summit. The base case uses 216 GPUs (36 nodes on Summit), and all subsequent wall times are normalized with the base case. We observe near ideal (within 3%) efficiency for up to 13824 GPUs (2304 nodes), thus facilitating the ability to run large multi-GPU simulations while preserving speedups.

Halo exchange times are prominent at smaller problem sizes on GPUs. Thus, minimizing communication time is of paramount importance to achieve ideal strong scaling. CUDA-aware MPI can drastically (4x) reduce halo exchange time by using faster GPU interconnects to transfer buffers while simultaneously preventing the need for data transfer between host and device.

Figure 3 shows MFC's strong scaling performance on Summit for the same problem as the weak scaling analysis with the base case using 8 NVIDIA V100 GPUs. The largest problem that can fit on a V100 GPU consisting of 64 million grid points (8M per GPU, red) retains 84% of ideal performance even when the node count is increased by a factor of 10. For smaller problems (2M, green) and a large number of GPUs, a larger deviation from ideal performance is observed due to the prominence of MPI transfers (∼50% of run time).

We test the ability to conduct large multiphase simulations on GPUs by running a sample problem of size 216 million on 216 GPUs (36 nodes on Summit). Figure 4 shows the corresponding simulation of 50 bubbles in water collapsing near a wall.

IV. Conclusion

Kernel optimizations coupled with careful memory use can improve the arithmetic intensity of multiphase compressible flow simulations, resulting in large GPU speedups. Using effective MPI communication techniques to ensure scalability will also facilitate the preservation of such speedups for large multi-GPU runs.

Acknowledgment

The authors acknowledge use of OLCF Summit and OLCF Wombat under Director's Discretionary allocation CFD154 and XSEDE under allocation TG-PHY210084.

REFERENCES

[1] J. C. Meng and T. Colonius, "Numerical simulation of the aerobreakup of a water droplet," *Journal of Fluid Mechanics*, vol. 835, p. 1108–1135, 2018.

[2] R. Saurel, F. Petitpas, and R. Abgrall, "Modelling phase transition in metastable liquids: application to cavitating and flashing flows," *Journal of Fluid Mechanics*, vol. 607, p. 313–350, 2008.

[3] C. E. Brennen, "Cavitation in medicine," *Interface focus*, vol. 5, no. 5, p. 20150022, 2015.

[4] A. Chauvin, G. Jourdan, E. Daniel, L. Houas, and R. Tosello, "Experimental investigation of the propagation of a planar shock wave through a two-phase gas-liquid medium," *Physics of fluids*, vol. 23, no. 11, p. 113301, 2011.

[5] S. H. Bryngelson, K. Schmidmayer, V. Coralic, J. C. Meng, K. Maeda, and T. Colonius, "Mfc: An open-source high-order multi-component, multi-phase, and multi-scale compressible flow solver," *Computer Physics Communications*, vol. 266, p. 107396, 2021.

[6] A. Kapila, R. Menikoff, J. Bdzil, S. Son, and D. S. Stewart, "Two-phase modeling of deflagration-to-detonation transition in granular materials: Reduced equations," *Physics of fluids*, vol. 13, no. 10, pp. 3002–3024, 2001.

[7] R. Saurel, F. Petitpas, and R. A. Berry, "Simple and efficient relaxation methods for interfaces separating compressible fluids, cavitating flows and shocks in multiphase mixtures," *journal of Computational Physics*, vol. 228, no. 5, pp. 1678–1712, 2009.

[8] X.-D. Liu, S. Osher, and T. Chan, "Weighted essentially non-oscillatory schemes," *Journal of computational physics*, vol. 115, no. 1, pp. 200–212, 1994.

[9] E. F. Toro, *Riemann solvers and numerical methods for fluid dynamics: a practical introduction*. Springer Science & Business Media, 2013.

[10] S. Gottlieb and C.-W. Shu, "Total variation diminishing runge-kutta schemes," *Mathematics of computation*, vol. 67, no. 221, pp. 73–85, 1998.