11th International Conference on Multiphase Flow,
ICMF 2023, Kobe, Japan, April 2–7, 2023

Fast simulation of multiphase compressible flows through GPU acceleration

Henry Le Berre, Anand Radhakrishnan, and Spencer H. Bryngelson*

School of Computational Science & Engineering, Georgia Institute of Technology, Atlanta, GA 30308
*shb@gatech.edu

Abstract

Simulations of multiphase compressible flows can take several days to produce actionable results on distributed CPU systems. We present computational techniques that accelerate such simulations by leveraging GPUs. The methodology is demonstrated on a multi-GPU simulation of a collapsing bubble cloud. The low arithmetic intensity of common algorithms for compressible flow is addressed by optimizing GPU kernels for the available hardware, leading to a 300 times speed up on an NVIDIA A100 GPU compared to a modern Intel CPU core. The method demonstrates near ideal ($\sim$97%) weak scaling on OLCF Summit for at least 13824 GPUs.

Keywords: GPU acceleration, compressible flow, multi-component flow

Introduction

Multiphase compressible flows are ubiquitous: bubbles nucleate and cavitate (Saurel et al. 2008; Brennen 2015), droplets form and atomize (Meng and Colonius 2018). These flows are important in application. For example, shock waves from collapsing bubbles lead to large pressures that can erode pumps and propellers (Dular et al. 2004; Binama et al. 2016; Sharma et al. 1990). Simulation of these phenomena can accelerate engineering design and prevent mechanical failure.

Compressible flow simulations are usually restricted to small time step sizes. Thus, it is important to minimize the required wall-time for each step. At the same time, many new supercomputers rely on GPU accelerators for most of their compute capabilities; often more than 90% of available FLOPS come from such acceleration.

Leveraging GPUs is thus required to take advantage of the most powerful computers. However, algorithms for compressible flows consist primarily of vector operations, which typically have low arithmetic intensity and do not immediately use the full capability of the GPUs themselves. One must also exchange boundary data between processors, called halo exchanges. This transfer requires fast network communication when problem sizes are small.

Here, we present strategies that lead to large simulation speedups and use them to run a large multi-GPU simulation of a bubble cloud collapsing near a wall. The algorithm scales well to thousands of GPUs, making it practical for large simulations. Similar large-scale simulations of many collapsing bubbles have been conducted on CPUs (Rasthofer et al. 2019), though different strategies are required for the GPUs discussed here.

Numerical Method and Implementation

Interface capturing methods (Kapila et al. 2001; Saurel et al. 2009) are used to generate the governing equations. These equations are discretized and solved via a finite volume method that uses high-order accurate WENO reconstruction of the primitive variables at the finite volume faces, leading to a Riemann problem (Liu et al. 1994). The HLLC approximate Riemann solver handles this problem (Toro 2013). The flow is marched in time using a total variation diminishing Runge–Kutta time stepper (Gottlieb and Shu 1998).

These methods are implemented in the open-source solver MFC (Bryngelson et al. 2021). MFC has been extensively validated on cases involving shock-bubble interactions and spherical bubble dynamics. MFC is a Fortran codebase that exhibits near ideal weak scaling on CPUs (Bryngelson et al. 2021). Acceleration is implemented via OpenACC, which offloads all compute kernels to GPU accelerators (Wienke et al. 2012). Here, we use OpenACC instead of CUDA for portability purposes, though one can expect similar speedups with CUDA.

Methodology for obtaining GPU speedups

GPU acceleration is tested via a 2-component water–air problem with 8 million grid points in 3D, which is close to the DRAM capacity (16 GB) of the V100 GPUs on Summit. Memory realignment of the state variables in 2D and 3D is required to coalesce memory access. Doing so leads to a 6-times speedup of the most expensive GPU kernel, which performs the WENO reconstruction. Performance improvements are also attained via metaprogramming. *Fypp*, a Fortran pre-processor (Aradi 2021), passes user inputs as fixed parameters, enabling compile-time optimizations and efficient use of the thread stack memory. These optimizations lead to 8- and 2-times speedups of the two most expensive kernels, which are associated with the WENO reconstruction and Riemann solver. Together, these optimizations result in a 300-times speedup of a single NVIDIA A100 GPU over one Intel Xeon Cascade Lake CPU core. On OLCF Summit, the 6 NVIDIA V100 GPU and 2 IBM POWER9 CPU compute nodes benefit from a 40-times speedup.

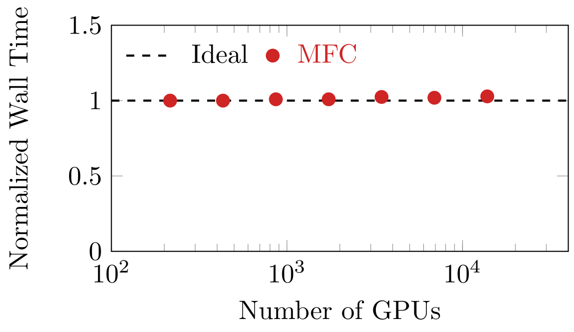

Figure 1: Weak scaling on Summit for a 3D multi-component test problem.

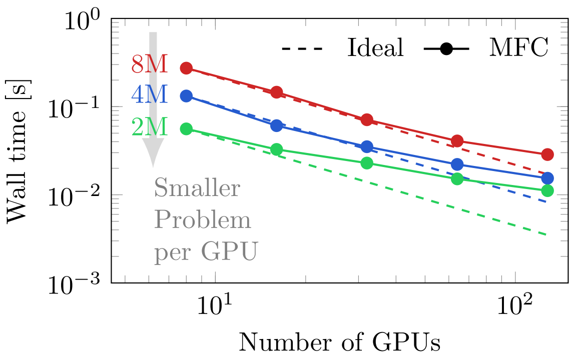

Figure 2: Strong scaling on OLCF Summit. Labels indicate problem size per GPU.

Scalability

We next examine multi-GPU performance for the 3D 2-component water–air problem with 1 million grid points per GPU on OLCF Summit. In fig. 1, the problem size is increased in proportion to the number of GPUs, called weak scaling. The wall-times are normalized using the 216 GPU base case. We observe within 3% of ideal efficiency up to at least 13824 GPUs.

We next increase the number of GPUs for a fixed problem size to observe strong scaling performance. Figure 2 shows this performance on Summit. The baseline case uses 8 NVIDIA V100 GPUs for a 64 million grid point problem. Halo exchange times, while less than 1% of the time step cost on CPUs for this problem, are more prominent on GPUs due to the faster kernels. Smaller problem sizes require fast communication to retain GPU speedups due to the increasing fraction of data in the halo region. We use CUDA-aware MPI to transfer boundary data between processors, which uses the direct interconnects between GPUs, resulting in 4-times faster halo exchanges. For example, 84% of ideal performance is retained when the node count is increased by a factor of 10 for the same problem size.

Collapsing Bubble Cloud

We demonstrate the benefits of GPU speedups for large multiphase simulations by conducting a test run. The test case simulates the collapse of 50 air bubbles in water at ambient pressure ($1\,\mathrm{atm}$) near a wall when subject to a $10\,\mathrm{atm}$ pressure wave. The density of the fluids follows from those at standard temperature and pressure. The stiffened-gas equation of state represents both fluids (Menikoff and Plohr 1989). This example simulation uses 216 million grid points, corresponding to 60 for each bubble diameter. The simulation time requires 3×10^5 time steps.

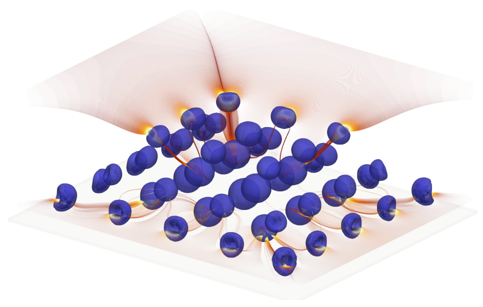

Figure 3: Streamlines of a collapsing bubble cloud near a wall.

Figure 3 shows the streamlines of the collapsing bubble cloud and the isovolumes of air volume fraction in the range 0.95–1. Here, we use 216 GPUs (36 nodes) on Summit, or 1 million points per GPU, which amounts to 2 hours of wall-time for this simulation. Using POWER9 CPUs on the same compute nodes would require more than 2 days for the same problem. We save the simulation state to disk every 1000 time steps. Each data export requires about ten time steps worth of wall-time, so this cost has a negligible impact on performance.

Conclusion

Efficient memory use and kernel optimizations lead to large GPU speedups for multiphase compressible flow simulations. The weak scaling behavior is ideal, and the strong scaling behavior is comparable to other GPU flow solvers. Large multiphase flow simulations can be conducted in a few hours on NVIDIA GPUs as opposed to a few days on Intel Xeon and IBM POWER9 CPUs.

Acknowledgment

S. H. Bryngelson acknowledges the use of the Extreme Science and Engineering Discovery Environment (XSEDE) under allocation TG-PHY210084, OLCF Summit under allocation CFD154, hardware awards from the NVIDIA Academic Hardware Grants program, and support from the US Office of Naval Research under Grand No. N000142212519 (PM Dr. Julie Young).

References

B Aradi. Fypp: Python-powered Fortran metaprogramming, 2021. URL `https://github.com/aradi/fypp`.

M Binama, A Muhirwa, and E Bisengimana. Cavitation effects in centrifugal pumps-a review. *International Journal of Engineering Research and Applications*, 6(5):52–63, 2016.

CE Brennen. Cavitation in medicine. *Interface Focus*, 5(5): 20150022, 2015.

SH Bryngelson, K Schmidmayer, V Coralic, JC 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*, 266:107396, 2021.

M Dular, B Bachert, B Stoffel, and B Širok. Relationship between cavitation structures and cavitation damage. *Wear*, 257(11):1176–1184, 2004.

S Gottlieb and CW Shu. Total variation diminishing runge-kutta schemes. *Mathematics of Computation*, 67(221):73–85, 1998.

AK Kapila, R Menikoff, JB Bdzil, SF Son, and DS Stewart. Two-phase modeling of deflagration-to-detonation transition in granular materials: Reduced equations. *Physics of Fluids*, 13(10):3002–3024, 2001.

XD Liu, S Osher, and T Chan. Weighted essentially non-oscillatory schemes. *Journal of Computational Physics*, 115 (1):200–212, 1994.

JC Meng and T Colonius. Numerical simulation of the aerobreakup of a water droplet. *Journal of Fluid Mechanics*, 835: 1108–1135, 2018.

R Menikoff and BJ Plohr. The riemann problem for fluid flow of real materials. *Reviews of Modern Physics*, 61(1):75, 1989.

U Rasthofer, F Wermelinger, P Karnakov, J Šukys, and P Koumoutsakos. Computational study of the collapse of a cloud with 12 500 gas bubbles in a liquid. *Physical Review Fluids*, 4(6):063602, 2019.

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

R Saurel, F Petitpas, and RA Berry. Simple and efficient relaxation methods for interfaces separating compressible fluids, cavitating flows and shocks in multiphase mixtures. *Journal of Computational Physics*, 228(5):1678–1712, 2009.

SD Sharma, K Mani, and VH Arakeri. Cavitation noise studies on marine propellers. *Journal of Sound and Vibration*, 138(2):255–283, 1990.

EF Toro. *Riemann solvers and numerical methods for fluid dynamics: a practical introduction*. Springer Science & Business Media, 2013.

S Wienke, P Springer, C Terboven, et al. OpenACC — first experiences with real-world applications. In *European Conference on Parallel Processing*, pages 859–870. Springer, 2012.