Accelerating PETSc-Based CFD Codes with Multi-GPU Computing

Pi-Yueh Chuang
Mechanical and Aerospace Engineering
The George Washington University
Washington, DC 20052
Email: pychuang@gwu.edu

Lorena A. Barba
Mechanical and Aerospace Engineering
The George Washington University
Washington, DC 20052
Email: labarba@gwu.edu

I. Motivation

The motivation of this work is a desire to accelerate existing PETSc[1]-based CFD (computational fluid dynamics) codes with multi-GPU computing. Rewriting their code base with CUDA or OpenCL may be onerous for researchers who focus more on the application of CFD, rather than high-performance computing. Typically, the most expensive part in CFD codes is solving Poisson-like systems, for which NVIDIA's AmgX[2] library provides multi-GPU solvers and preconditioners. We present a wrapper code to bridge AmgX and PETSc so that users can run their existing PETSc-based CFD codes on modern heterogeneous platforms, with minimal code changes. Not only does the wrapper code handle converting between AmgX and PETSc data structures, but it also effectively takes care of the situation where the number of MPI processes is different than the number of GPUs.

We assessed the overall benefits of using multi-GPU systems in CFD applications when replacing PETSc with AmgX. While application speed-up[1]was our top concern, we also studied other possible benefits, such as savings in terms of hardware and cost. To determine such benefits, we used Poisson systems in 2D and 3D as benchmarks and also a CFD application with our open-source code, PetIBM.

II. AmgX and AmgX wrapper

NVIDIA's AmgX library provides various iterative solvers and preconditioners on multiple GPUs, and is free for non-commercial use. Using AmgX in existing PETSc applications would require considerable code modifications, due to the different data structures used by the two libraries. Furthermore, since some calculations other than the linear solvers may remain on CPUs, users may want to take advantage of all the CPU cores available for these tasks. If the number of GPUs is different to the number of CPU cores on a heterogeneous platform (typically fewer GPUs than cores), the data and linear systems on CPUs need to be merged and scattered before and after calling the solvers on GPUs. This indicates nontrivial modifications in existing CFD codes, which could be costly and a deterrent to adopting GPU hardware.

We developed a wrapper code to couple AmgX and PETSc, featuring simple usage and the capability to exploit all available CPU and GPU resources. It hides from users all MPI communications, data conversions, data transfer between CPU and GPU, and memory allocations. In most scenarios, users can directly replace the PETSc calls to `KSPSetOperators` and `KSPSolve` with the wrapper functions `setA(A)` and `solve(x, rhs)`. `A`, `x`, and `rhs` remain PETSc matrix and vectors before and after calling the wrapper. Users need to make no other coding effort. Setting up the properties of the solvers (e.g., type of solvers and preconditioners) can be done by a user-specified input script at runtime.

III. Benchmarks and results

We present three benchmakrs in the poster:

A. Standard Poisson systems

1) Description: This benchmark provides a reference to users whose CFD codes solve standard Poisson systems. For this purpose, we measured only the run time of solving the system, that is, the calls to `KSPSolve` and `solve(x, rhs)`. The function `solve(x, rhs)` covers MPI communications and data conversion and transfer, so timing the call to this function gives a performance indication of our wrapper, rather than only of the solution behavior on GPUs.

The benchmarks use conjugate-gradient solvers on both CPUs and GPUs. The preconditioner for the CPU solver is Hypre BoomerAMG (through PETSc's interface), whereas the preconditioner of the GPU solver is classical algebraic multigrid – described as a GPU implementation of Hypre BoomerAMG in the AmgX manual[see 3, p.130].

2) Results: With 25M unknowns, four[2] NVIDIA K20s can compete with about 100[3] CPU cores in 2D problems; in 3D, eight K20s can compete with about 256 CPUs. For 2D problems with 100M unknowns and 3D problems with 50M unknowns, 32 K20s can compete with about 400 CPU cores.

B. Flying snake simulations – an application of PetIBM

1) Description: This benchmark intends to show how multi-GPU computing can accelerate a real PETSc-based CFD

[1]"Speed-up" in this work refers to *application speed-up*: the acceleration that results from directly replacing PETSc linear solvers with AmgX solvers in an application. In other words, these application speed-ups represent the overall reductions of run times that users of the applications can experience.

code through an actual application. The CFD code we chose is PetIBM[4], and the application is simulating the flow around a flying-snake's cross-section[5].

PetIBM, our group's PETSc-based CFD code, offers a Navier-Stokes solver based on the formulation of Taira and Colonius[6], in which a modified Poisson system is solved at each time step. Solving this modified Poisson system takes over 90% of run rime on CPUs. We only applied multi-GPU linear solvers on this part, while keeping all other calculations on CPUs.

We attempt a fair comparison using the same nodes of the cluster, with the GPU runs exploiting two available NVIDIA K20s on each node (each node has 12 physical CPU cores). We also ran the benchmark on a workstation, having 6 physical CPU cores and up to two NVIDIA K40c GPUs.

We obtain the best run times on the GPU cases with conjugate-gradient solvers and aggregation multigrid preconditioners. For the CPU cases, we used stabilized biconjugate-gradient solvers and GAMG preconditioners, given that (to the best of our knowledge) there is no counterpart implementation of aggregation multigrid in PETSc, and the combination of stabilized biconjugate-gradient and GAMG gave the best result in this case.

2) Results: The overall speed-up provided by only one GPU node is competitive with about 20 nodes of a CPU cluster. On a workstation, one K40c GPU can compete with about 16 nodes of a CPU cluster.

C. Amazon EC2

1) Description: The previous benchmark shows that enabling multi-GPU computing in a CFD code can size down the clusters required for running simulations. This comes with a cost saving to research based on CFD simulations. We experimented with the same flying-snake simulations on Amazon EC2 to give an example of the cost savings. We compared the simulation executed on an 8-node CPU cluster (c4.8xlarge) versus a single GPU node (g2.8xlarge).

2) Results: The result indicates a 3.1x speed-up and a 16x cost saving on Amazon EC2 when exploiting the GPU nodes.

IV. Conclusion

With our wrapper, existing PETSc-based CFD codes can now exploit all CPU and GPU resources with affordable coding effort. In additions, our benchmarks show that multi-GPU computing can size down the cluster systems required and money cost for scientific simulations.

[2]The number of GPUs presented in the text is the minimum number of GPUs needed due to limited memory on a single GPU.

[3]The comparable numbers of CPU cores are estimated based on good scaling shown in the benchmarks (see figures on the poster). This is a conservative estimate for our purposes (i.e., favors the CPU case).

References

[1] S. Balay, S. Abhyankar, M. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. Gropp, D. Kaushik, M. Knepley, L. C. McInnes, K. Rupp, B. Smith, S. Zampini, H. Zhang, and H. Zhang, *PETSc web page*, 2016. [Online]. Available: http://www.mcs.anl.gov/petsc (visited on 07/13/2016).

[2] *AmgX*. [Online]. Available: https://developer.nvidia.com/amgx (visited on 07/13/2016).

[3] *AmgX reference manual, API version 2*, English, NVIDIA, Nov. 2014.

[4] *PetIBM*. [Online]. Available: http : / / github . com / barbagroup/PetIBM (visited on 07/13/2016).

[5] A. Krishnan, J. J. Socha, P. P. Vlachos, and L. A. Barba, "Lift and wakes of flying snakes," *Physics of Fluids*, vol. 26, no. 3, p. 031 901, 2014. DOI: 10.1063/1.4866444.

[6] K. Taira and T. Colonius, "The immersed boundary method: A projection approach," *Journal of Computational Physics*, vol. 225, no. 2, pp. 2118–2137, 2007. DOI: 10.1016/j.jcp.2007.03.005.