7th Fall Technical Meeting of the Eastern States Section of the Combustion Institute
Hosted by the University of Connecticut, Storrs, CT
October 9–12, 2011

Turbulence-chemistry closure method using graphics processing units: a preliminary test

K. E. Niemeyer[1], C. J. Sung[2], C. G. Fotache[3], and J. C. Lee[3]

[1]*Department of Mechanical and Aerospace Engineering*
Case Western Reserve University, Cleveland, OH 44106
[2]*Department of Mechanical Engineering*
University of Connecticut, Storrs, CT 06269
[3]*United Technologies Research Center - UTC*
East Hartford, CT 06108

Graphics processing units (GPUs) are attractive for high-performance computing due to their massively parallel architecture, modest power consumption, and relatively low cost. Recent attempts were made to speed up the chemical kinetics in reactive-flow simulations. However, these approaches did not realize the full potential of the GPU as inter-processor communication was intensive. In some popular high-fidelity closure models, there is a very large set (e.g. 10^4–10^6) of independent chemical kinetics integrations to perform. This leads to a new strategy of performing all the chemistry integrations on the GPU to minimize communication between the CPU and GPU and to maximize the computational load on the GPU. We demonstrated this technique with a detailed hydrogen kinetics system (9 species and 38 reactions) and observed a speed up factor of 70. Also, the computing time on the GPU is short enough that when the CPU is designed to handle the transport terms, the impact of chemistry on the overall computational "clock time" is minimal. Thus, for practical-scale LES simulations, the GPU allows the inclusion of detailed kinetics in closure models.

1 Introduction

The utilization of the general-purpose graphics processing unit (GPU) has grown significantly in the past decade due to its increasing performance and inherent parallelism, as well as its potential in reducing capital costs. GPUs are popular in many scientific computing areas such as protein folding, molecular dynamics, CAT scan processing, computational finance, and other large, complex problems that can readily be formulated using a fine grained, massively concurrent computation paradigm [1].

The use of GPUs in reactive-flow simulations is limited to date with only two notable approaches found in the literature: 1) evaluation of the reaction rates on the GPU, which is equipped with hardware-accelerated transcendental function calls, to reduce the CPU time in performing the chemical kinetics time integration [2] and 2) using the GPU to accelerate both the reaction rate evaluations and matrix inversion step in an implicit integrator [3, 4]. However, there are significant limitations to both of these approaches. In the first, the intensive memory transfer between the CPU and the GPU via the PCI bus makes the computation-to-memory-movement ratio unsatisfying. In the second case, order-of-magnitude speedup was realized only with extremely large reaction mechanisms (i.e. $>$ 1000 species). The inversion of a large matrix was performed using a commercial GPU linear algebra library (CULA [5]); the speedup shown in the study mainly demonstrated the performance of the CULA matrix inversion. The kinetic system used in this demonstration was unrealistic and impractical for industrial high-fidelity simulations, which are typically limited to tens of species.

In this work, we propose and demonstrate a new strategy that would allow detailed chemical kinetics to be employed in large-scale, high-fidelity reactive-flow simulation codes that are relevant to both industrial and academic applications. In such applications, the domain-decomposition-based parallelization paradigm and operator-splitting procedure are the norm and these codes are designed primarily to run on medium-sized Linux clusters (i.e. a few hundred nodes). In the domain-decomposition method, each CPU core is assigned a distinct portion of the computational domain arranged in such a way to minimize the inter-processor communication. Whereas, the operator-splitting procedure effectively makes the chemical kinetics calculation for each computational cell independent in every time integration cycle. The turbulent-chemistry closure model employed in these codes is typically the cell-averaged closure model for high–Mach-number simulations. For low–Mach-number applications, a large number of closure models exist to avoid the problem associated with stiff chemical kinetics. Among these closure models, the probability density function (PDF) transport approach [6–8] is perhaps the most promising that is capable of predicting phenomena such as lean blow-off, high-altitude re-ignition, and CO emissions. At the low-level implementation of this method, there is a Monte Carlo sampling [7] of chemical kinetics in each computational cell. This results in a very large set of independent chemical kinetics integrations. Even a modest number, such as 50 samples per cell, would require intensive computational resources. This severely limits the utilization of the PDF transport model in industrial applications. However, a large number of concurrent chemical kinetics integrations, in either the cell-averaged or PDF transport approaches, fits well with the hardware architecture of a GPU.

In the new strategy, the CPU is responsible for the transport terms while the GPU handles the chemistry term. Both sets of calculations can be performed concurrently under the operator-splitting method. The memory-transfer latency is mitigated by only passing the initial conditions and final results back and forth between the CPU and GPU. The chemical kinetics integrator on the GPU is self-contained; the GPU threads include all the information pertaining to the chemical kinetics system, the thermodynamic properties, and the time integration algorithm.

In the following, we discuss the methodology of our approach, then present and discuss results of a performance test between using traditional, CPU-only and combined CPU/GPU-calculations. The computing time of many independent chemical kinetics integrations is measured, ranging the number of concurrent integrations (corresponding to computational cells) over a range realistic to large-scale, high-fidelity simulations.

2 Methodology

In order to determine the potential speedup of a parallel reactive-flow simulation where CPU/GPU pairs are responsible for different portions of the computational domain, we considered a single CPU/GPU pair to carry out our numerical experiment. We implicitly assume that our candidate for GPU acceleration is a reactive-flow simulation code that has good scalability on a cluster of CPUs using the domain-decomposition technique. When the cluster is GPU-capable, each CPU thread will spawn GPU threads, and the fundamental computer unit is a CPU/GPU pair. Thus, we demonstrate the speedup possible with a single GPU card and compare its performance to one CPU core. For this demonstration, we use an explicit integration method on a detailed hydrogen/air system, with high-Mach reactive flow (hypersonics) as the intended area of application. Performance comparisons were made between a single CPU core and a CPU/GPU pair. For the CPU/GPU pair, time integrations of a large set of independent kinetics systems is assigned to the GPU exclusively and the role of the CPU is to handle the data traffic to and from the GPU, specifically the initial conditions input and the integrated results output.

Two versions of the code were written: one used only the CPU, and the other used the CPU only to send the initial conditions to and receive the integrated results from the GPU. The GPU code that performed the chemical kinetics integration is self-contained; it performed all the integration steps and evaluated all the reaction rates and thermodynamic properties on the GPU card. The GPU threads were launched such that each thread corresponds to one distinct instance of a chemical kinetics integration (with different initial conditions). In the test cases shown in this work, one NVIDIA c2070 card was used which has 488 streaming processors; thus, in ideal conditions, 488 threads could be run simultaneously. However, the CPU version necessarily performed the set of integrations sequentially. In our test, the computational time of the GPU code included the memory transfer to and from the GPU, as this is representative of actual an application.

We used the explicit fourth-order Runge-Kutta method to perform the integration of the detailed hydrogen-air reaction mechanism of Yetter et al. [9], with 9 species and 38 irreversible reactions. The code was written in ANSI C for the CPU portions and NVIDIA's Compute Unified Device Architecture (CUDA) C version 4.0 [10] for the GPU portion. The code did not use any existing chemical kinetics libraries (e.g. Chemkin, Cantera). Rate expressions were written out explicitly in a manner to minimize the number of operations. Also, the layout of the code was optimized for both the CPU and GPU versions (e.g. all loops unrolled). Different memory access patterns and sets of optimization compiler (GNU) flags were tested and the optimal sets were used for both the GPU and CPU versions. Calculations were performed on an AMAX ServMax PSC-6n GPU workstation; this machine contains dual six-core Intel Xeon 2.66 GHz CPUs and four NVIDIA Tesla c2070 GPU cards, but for the current tests only a single CPU core and one GPU card were used. In order to compare the performance of the CPU and GPU for cases relevant to large-scale reactive-flow simulations, we varied the number of independent chemical kinetics integrations performed. The number ranged from the small scale (2048) to the largest number of cells expected for one CPU core (16,777,216, or 256^3).

3 Results and discussion

The computing time per integration time step comparison between the CPU and GPU as a function of independent integrations to perform (or CFD cells) is depicted in Fig. 1, displayed in logarithmic scale. Even at small numbers of independent integrations, the GPU performs better than the CPU, and at larger numbers, the difference reaches nearly two orders of magnitude. The largest speedup is 75x at 8,388,608 independent integrations. At very small numbers of independent integrations (GPU threads), the streaming processors are underutilized and the performance dropped to the CPU's level, but with more threads the streaming processors are saturated and the overall computational performance increases.

The upper limit of independent integrations, 16,777,216, was chosen for two reasons. First, this is the maximum thread size allowable on the current GPU hardware, without splitting the

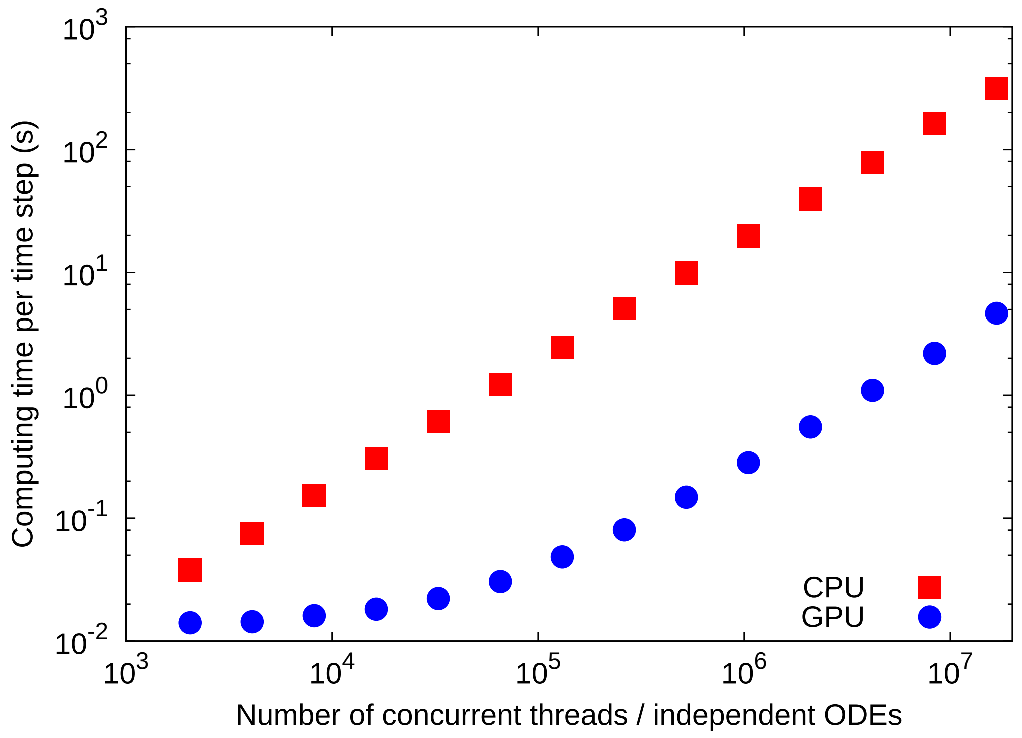

Figure 1: Comparison of CPU and GPU computational time per integration time step as a function of number of independent chemical kinetics time integrations. The GPU time includes the memory transfer to and from the GPU at the beginning and end of the GPU solver calls, respectively.

integration into multiple calls. Additionally, this is the largest number of computational cells expected for a single CPU core; we considered a realistic range for most applications.

Load balancing, or ensuring each device (CPU and GPU) is not waiting for the other to finish calculations, is important for an optimally-designed system. This is particularly important in our hybrid concurrent-computation paradigm. The chemical kinetics time integration on the GPU should balance with what is performed on the CPU, especially transport phenomena. In fact, the operator-splitting procedure ensures that the transport and chemical kinetics terms can be performed concurrently on the CPU and the GPU, respectively. Generally speaking, for typical large-scale reactive-flow simulations that run on a modest number of processors, the number of CPU cores employed and the problem size are varied such that each time integration takes about 10 seconds (or longer, for more complex cases). It can be seen that the chemistry calculations on the GPU take less time than this, even at the largest number of computational cells. Using the GPU makes the inclusion of detailed chemical kinetics in LES codes running on modest-sized computing clusters feasible.

4 Conclusions

A new method for incorporating detailed chemistry into large-scale reactive-flow simulations using graphics processing units (GPUs) is described and demonstrated using a detailed hydrogen/air reaction mechanism (9 species and 38 irreversible reactions). The strategy consists of: (1) using the CPU to handle the transport terms; (2) performing the chemical kinetics integrations (of the whole computational domain) entirely on the GPU concurrently with the CPU's calculations; and (3) passing only the initial conditions and final results back and forth between the CPU and GPU. We demonstrated an almost two orders of magnitude performance speedup comparing a CPU/GPU pair to a CPU alone, for large numbers of computational cells, showing the potential of chemistry integration performed on a GPU card having little or no impact on the overall clock time of the simulation. By performing the entire chemistry term calculation on the GPU, and only transferring the initial conditions and final results between the CPU and GPU, we made fuller utilization of the GPU computing power while minimizing the time-consuming data transfer between the CPU and the GPU. The use of GPUs enables the inclusion of detailed chemical kinetics in reactive flow codes.

Future work will focus on optimizing the memory configuration on the GPU, as well as expanding to hydrocarbon chemical kinetics systems of larger sizes. However, the explicit integration scheme used here is unsuitable for stiff chemistry, so GPU-based implicit methods will be explored. Both approaches will be incorporated into existing CFD codes for demonstrations.

Acknowledgments

This work was supported by National Science Foundation under Grant No. 0932559, the United Technologies Research Center, and the Department of Defense through the National Defense Science and Engineering Graduate Fellowship program.

References

[1] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, J. C. Phillips, Proc. IEEE 96 (2008) 879–899.

[2] K. Spafford, J. Meredith, J. Vetter, J. Chen, R. Grout, R. Sankaran, in: Euro-Par 2009 Workshops, LNCS 6043, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 122–131.

[3] Y. Shi, W. H. Green Jr., H.-W. Wong, O. O. Oluwole, Combust. Flame 158 (2011) 836–847.

[4] Y. Shi, O. O. Oluwole, H.-W. Wong, W. H. Green, A multi-approach algorithm for enabling efficient application of very large, highly detailed reaction mechanisms in multi-dimensional HCCI engine simulations, 7th National Combustion Meeting, 2011.

[5] J. R. Humphrey, D. K. Price, K. E. Spagnoli, A. L. Paolini, E. J. Kelmelis, CULA: Hybrid GPU accelerated linear algebra routines, SPIE Defense and Security Symposium (DSS), 2010.

[6] D. C. Haworth, Prog. Energy Combust. Sci. 36 (2010) 168–259.

[7] H. Möbus, P. Gerlinger, D. Brüggemann, Combust. Flame 132 (2003) 3–24.

[8] S. B. Pope, Prog. Energy Combust. Sci. 11 (1985) 119–192.

[9] R. A. Yetter, F. L. Dryer, H. Rabitz, Combust. Sci. Tech. 79 (1991) 97–128.

[10] NVIDIA, CUDA C programming guide version 4.0, http://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/CUDA_C_Programming_Guide.pdf, 2011.