Exploiting heterogeneous HPC systems for combustion simulations with detailed chemistry via CPU-GPU load distribution algorithms

Álvaro Moure[1]*, Anurag Surapaneni[1], Daniel Mira[1]
*Lead presenter: alvaro.moure@bsc.es
[1] Barcelona Supercomputing Center, Spain

Combustion is a multi-physics and multi-scale phenomenon that is part of industries and sectors that modern society heavily relies on, such as transportation, heating systems, industrial furnaces, electrical energy generation, etc. Numerical simulations of combustion have become a essential tool in the design process of these intricate systems, providing deeper insights into the underlying physics and enabling access to data that is either impractical or prohibitively expensive to obtain through experiments. However, performing high-fidelity numerical simulations of real-scale industrial devices requires the use of large computing resources. Historically, combustion codes have been developed and optimized for running in CPU-based computers, but the new generation of supercomputers comes with CPU-GPU heterogeneous architectures where GPUs bring the opportunity to highly accelerate arithmetic-intensive operations.
The current work presents a novel workload distribution algorithm that seeks for optimizing the use of CPUs and GPUs in detailed chemistry combustion simulations. Applying operator splitting [1] to the species equation, Eq (1), commonly used in reacting flows modelling, the computation is divided in two parts:

$$\frac{\partial\left(\rho Y_k\right)}{\partial t}+\nabla\cdot\left(\rho\mathbf{u}Y_k\right)=-\nabla\cdot\left(\rho\mathbf{V}_kY_k\right)+\dot{w}_k, \tag{1}$$

- Transport sub-step. Advection and diffusion terms are solved in the CPU, using third-order Runge-Kutta explicit scheme in the framework of the Finite Element Method-based Alya code [2].
- Chemical sub-step. The stiff Ordinary Differential Equation that involves the source term is computed in the GPU using the implicit Backward-Differentiation Formula bridging Sundials software [3] with the chemical reaction source code generator, pyJac [4].

The MPI-GPU algorithm assigns to some of Alya's MPI processes the role of *GPUManager*, that will carry the job of computing the chemical integration of the MPI ranks designed to it, in the GPU, while the non-*GPUManager* will only perform the transport sub-step. Different distribution strategies are being tested to minimize GPU data transfers, MPI communication primitives and synchronizations.

References

[1] G. Strang, "On the Construction and Comparison of Difference Schemes", *SIAM Journal on Numerical Analysis*, vol. 5, no. 3, pp. 506–517, 1968. doi:10.1137/0705041

[2] M. Vázquez, G. Houzeaux, S. Koric, A. Artigues, J. Aguado-Sierra, R. Arís, D. Mira, H. Calmet, F. Cucchietti, H. Owen, A. Taha, E. D. Burness, J. M. Cela, and M. Valero, ”Alya: Multiphysics engineering simulation toward exascale”, *Journal of Computational Science*, vol. 14, pp. 15–27, 2016. doi:10.1016/j.jocs.2015.12.007

[3] A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward, ”SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers”, *ACM Transactions on Mathematical Software*, vol. 31, no. 3, pp. 363–396, Sep. 2005. doi:10.1145/1089014.1089020

[4] K. E. Niemeyer, N. J. Curtis, and C.-J. Sung, ”pyJac: Analytical Jacobian generator for chemical kinetics”, *Computer Physics Communications*, vol. 215, pp. 188–203, 2017. doi:10.1016/j.cpc.2017.02.004