ParCFD'2020
32^{nd} International Conference on Parallel Computational Fluid Dynamics
May-11-13 2020, Nice FRANCE

ADAPTIVE LOAD BALANCING AND AUTOMATED CODE GENERATION FOR EFFICIENT SIMULATION OF REACTING FLOWS

Thorsten ZIRWES[*,†], Feichi ZHANG[†], Peter HABISREUTHER[†], Jordan A. DENEV[*], Henning BOCKHORN[†] AND Dimosthenis TRIMIS[†]

[*] Karlsruhe Institute of Technology
Steinbuch Centre for Computing
76344 Eggenstein-Leopoldshafen, Germany
e-mail: thorsten.zirwes@kit.edu, jordan.denev@kit.edu

[†]Karlsruhe Institute of Technology
Engler-Bunte-Institute
76131 Karlsruhe, Germany
e-mail: feichi.zhang@kit.edu, peter.habisreuther@kit.edu, henning.bockhorn@kit.edu, dimosthenis.trimis@kit.edu

Key words: Adaptive Load Balancing, Automated Code Generation, Turbulent Combustion, Direct Numerical Simulation, High Performance Computing, OpenFOAM

Abstract. Direct numerical simulation of turbulent combustion processes including molecular diffusion and complex chemical reaction mechanisms constitutes a challenge even on modern supercomputers. Because of this, two developments are presented which help to reduce the simulation time of reacting flows without affecting the accuracy of the results:

The first development targets node-level performance of the computation of chemical reaction rates. A converter tool has been developed which reads a reaction mechanism as input and generates C++ source code. It contains the routines for computing chemical reaction rates for that specific reaction mechanism. This allows to generate highly optimized code that makes use of cache locality, vectorization and enables additional compiler optimizations, leading to simulation time reductions of up to 70 %.

The second development targets parallel load imbalances. Because chemical reaction rates are usually computed from an operator splitting approach, they are integrated over the CFD time step by a solver for stiff ordinary differential equations. However, in regions of the simulation where no combustion takes place, this integration requires much less internal time steps than in regions of active combustion. This creates load imbalances between the processes. Therefore, a load balancing technique has been developed which identifies pairs of processes that adaptively share their workload. This has been shown to reduce simulation times by about 30 %.

These developments have been included in a reacting flow solver in OpenFOAM and used to perform the simulation of a turbulent flame on Germany's largest supercomputer on up to 28 800 CPU cores. The detailed results allow generating a DNS database of mixed-mode flames for developing improved combustion models.

1 INTRODUCTION

To address the climate challenges of the future by improving current combustion devices and designing new combustion concepts, a deeper understanding of the complex multi-physics given by the interaction of turbulent flow and chemical reactions is necessary. Because of the multi-scale nature of combustion, which spans many length and time scales, direct numerical simulations (DNS) for practical applications are still a challenge on today's supercomputers. It is therefore important to reduce the simulation time on high performance computers without affecting the accuracy of the results [1].

The most accurate results are obtained by direct numerical simulations, where all relevant time and length scales of both the flow and flame are resolved. This, however, requires to include detailed reaction mechanisms which can incorporate thousands of chemical reactions. Because of this, an automated code generation approach for speeding up the computation of chemical reaction rates is presented in section 2. Section 3 presents an adaptive load balancing strategy specifically for the computation of chemical reaction rates. An application of the code is shown in section 4, where a turbulent flame is simulated on one of Germany's fastest supercomputers.

2 AUTOMATED CODE GENERATION

In most simulation tools for reactive flows, the information about which chemical reactions can occur and their specific modeling constants are read from a text file (*reaction mechanism*) at the start of the simulation. Based on these information, the chemical reaction rates are computed. In order to speed this up, a converter tool [1] has been developed which converts the text based reaction mechanism file into C++ source code containing routines for computing chemical reaction rates. Instead of using a general implementation which can use arbitrary reaction mechanisms, the generated source code only computes the reaction rates for one specific reaction mechanism of the user's choice. This technique is also used in other fields like GPU computing [2].

During the conversion process, chemical reactions are re-ordered so that reaction of the same type are grouped together. This allows to compute the reaction rates in loops that can easily be auto-vectorized by the compiler. All data is tightly packed and accessed sequentially to maximize CPU cache usage. Information like the number of reactions and chemical species, which differ between reaction mechanisms, are now compile time constants and enable additional compiler optimizations.

Compiling the automatically generated C++ source code creates a library that can directly be used in OpenFOAM. Depending on the size of the reaction mechanism, total simulation times can be reduced by up to 70 % [1] using the optimized code.

3 ADAPTIVE LOAD BALANCING

Because the time scale of many chemical reactions is lower than for solving the Navier-Stokes equations, chemical reaction rates are often computed using an operator splitting approach: First, the chemical reaction rates are integrated over the flow time step neglecting convection and diffusion. The average value of the reaction rate is then used as

non-stiff source term in the transport equations for chemical species and energy. This integration is performed with special solvers for stiff ordinary differential equations (ODE), which use an adaptive internal time step for the integration. However, the number of internal time steps depends on how reactive the gas mixture is locally. For example, within the flame front where chemical reactions occur at high temperatures, the integrator has to take many internal time steps. In regions where no combustion takes place, e.g. in parts of a nozzle for pure air, the integrator only takes a few internal time steps. This creates an imbalance between processes working on different parts of the simulation domain.

Because of this, an adaptive load balancing algorithm was introduced [3]. In this algorithm, all processes are sorted by the time they need to integrate the chemical reaction rates for all cells in their part of the computational domain. The fastest process then forms a pair with the slowest process. The second fastest forms a pair with the second slowest and so on. The two processes within a pair then share their workload based on the relative timings between them. This has the advantage that only pair communication has to be done. Because chemical reaction rates can be computed from local thermo-chemical quantities without knowledge of any neighbor cells, the workload can be freely shared which makes the method attractive for unstructured grids. Additionally, this method can be combined with other load balancing approaches, e.g. arising from adaptive mesh refinement. Figure 1 shows measurements for a flame with and without the load balancing approach [3]. It shows the time required for computing chemical reaction rates on each MPI process. Because the slowest process determines the overall time, using the load balance approach reduces the time by about 30 %.

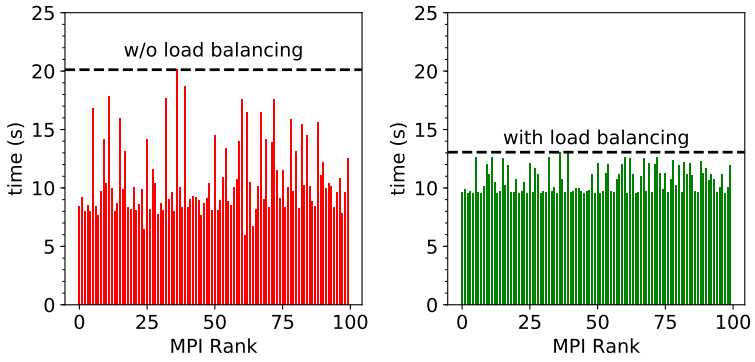

Figure 1: Time required to compute chemical reaction rates on each process with (right) and without (left) load balancing.

4 HPC SIMULATION OF TURBULENT COMBUSTION

A high performance computing (HPC) application is given by the simulation of the turbulent Sydney/Sandia flame [4]. A modified OpenFOAM solver for direct numerical simulation utilizing the two optimizations is used for the simulation, which is additionally coupled to Cantera [5] for computing detailed diffusion coefficients. Two consecutive simulations have been performed: the non-reactive mixing of fuel and oxidizer (Fig. 2 top) and the reactive flame simulation (Fig. 2 bottom). The simulations have been run

on Germany's fastest supercomputer at that time *Hazel-Hen* on up to 28 800 CPU cores, showing linear scaling behavior up to 10,000 CPU cores [1]. Comparison with experimental measurements of that flame show excellent agreement [4] and the results allow the detailed study of mixed-mode combustion.

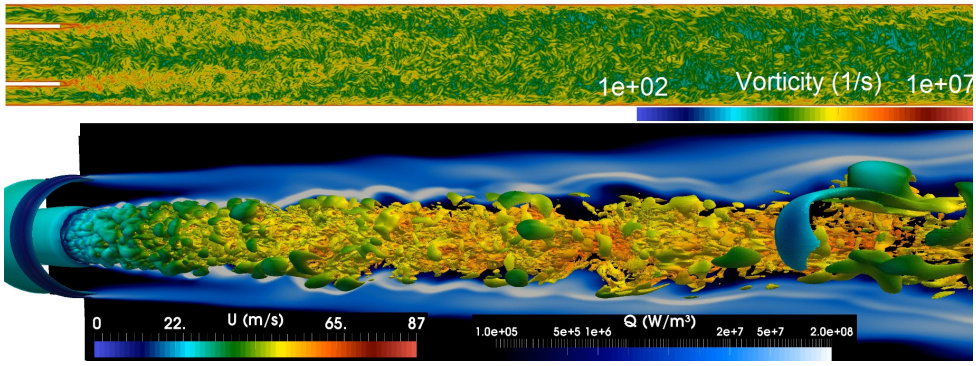

Figure 2: Vorticity during the mixing of fuel and air (top) and vorticity iso-surface colored by velocity with heat release cutting plane through the flame (bottom).

5 CONCLUSIONS

- Optimized code from automated code generation can speed up simulations by 70 %
- Adaptive pair-wise load balancing can reduce chemistry calculation times by 30 %
- These two optimizations are included in an OpenFOAM solver for combustion DNS, which is additionally coupled to Cantera and Sundials
- The code shows excellent scalability up to 28 800 CPU cores
- Results of the Sydney/Sandia flame show very good agreement with experiments

REFERENCES

[1] Zirwes, T., Zhang, F., Denev, J., Habisreuther, P., Bockhorn, H. Automated code generation for maximizing performance of detailed chemistry calculations in OpenFOAM, in *High Performance Computing in Science and Engineering ' 17* (Nagel, W, Kröner, F and Resch, M. eds.), (Cham), pp. 189–204, Springer, 2018.

[2] Niemeyer, K.E. and Curtis, N.J. pyJac v1.0.6, 2018.

[3] Zirwes, T., Zhang, F., Habisreuther, P., Denev, J.A., Bockhorn, H., and Trimis, D. Optimizing load balancing of reacting flow solvers in openFOAM for high performance computing, *6th OpenFOAM conference*, vol. 6, 2018.

[4] Zirwes, T., Zhang, F., Habisreuther, P., Hansinger, M., Bockhorn, H., Pfitzner, M. and Trimis, D. Quasi-DNS dataset of a piloted flame with inhomogeneous inlet conditions, *Flow, Turbulence and Combustion*, 2019.

[5] Goodwin, D., Moffat, H. and Speth, R. Cantera 2.4.0 2019. http://www.cantera.org.