A Massively Parallel High Order Flow Solver

Gregor J. Gassner, Andrea D. Beck
Institute for Aerodynamics and Gasdynamics, University of Stuttgart, Germany
EMail: gassner / beck@iag.uni-stuttgart.de

Abstract

In this work we present a discontinuous Galerkin based framework for the solution of compressible flow. The code is based on a multi-domain decomposition strategy, where the domain is subdivided into (unstructured) hexahedral grid cells. Each grid cell is mapped from physical space to a unit reference element, where for each grid cell a nodal tensor product trial function is used to approximate the flow solution. The nodal points are chosen equal to the quadrature points used to evaluate the inner products in the finite element like variational formulation. This and the fact that the tensor product structure of the ansatz maps itself to the structure of the resulting DG operator yields an highly efficient framework. To classify the efficiency of this framework, we compare the specific CPU time needed to evaluate one spatial degree of freedom for the three-dimensional compressible Navier-Stokes equations with a state of the art finite difference based DNS solver. Whereas the $6th$ order accurate finite difference code needs $4\mu s$ on one Nehalem core (Taubench needed about 7.6sec in average on this architecture), the DG framework with polynomial degree $N = 5$ needs $2\mu s$ (Gauss-Legendre points) or $1.6\mu s$ (Gauss-Lobatto Legendre points), respectively.

Additionally to this high serial efficiency, a major concern of the presented implementation was the high performance computing aspect. As we are interested in massive parallel simulations of unsteady flow, an explicit global time stepping 5 stage $4th$ order accurate Runge-Kutta time integrator was implemented. The MPI parallelization of the code was introduced without additional computational overhead (no “double-computing” of values at MPI boundaries!) with a special emphasis on communication latency hiding: the DG operator is naturally split in two parts, the volume integral part which depends only on local processor local data and the surface integral part which needs neighbor processor data to compute the numerical flux functions. This can be used to hide MPI communication latency by using non-blocking communication and performing processor local data computations first (volume integral) during the communication. The goal of this implementation is to demonstrate the excellent scaling capabilities of an explicit DG based software with an absolute limit of only *one* grid cell on a processor. Figure 1 shows selected results of such an MPI scaling. The left part of the plot demonstrates the weak scaling of the code for different combinations of polynomial degree N and number of grid cells, whereas the right plot shows the strong scaling of the code from 1 up to 4096 processors for different polynomial degrees

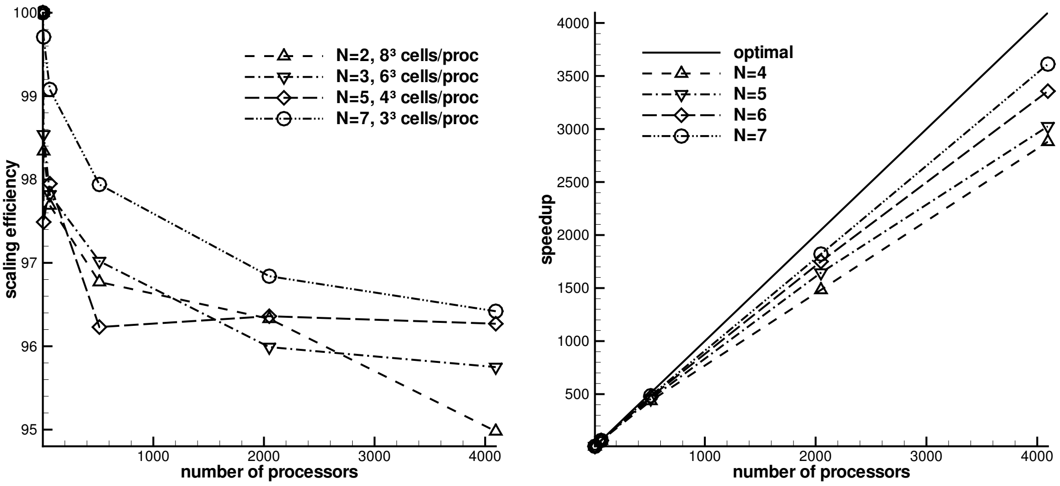

Figure 1: Weak (left) and strong (right) scaling result on an IBM Blue Gene P system (Jülich, Germany).

N. What makes those strong scaling results unique is the fact that the total grid cell number used for this test is 4096. Thus for the largest computation with 4096 processors, only one grid cell was left on the processor while still reaching 88% scaling with $N = 7$ (only 512 DOF per equation on a processor).

In this work, the presented framework is used to simulate the test case *Radial Expansion Wave* ($C1.5$); an unsteady three dimensional flow problem governed by the non-linear Euler equations with a defined adiabatic coefficient $g = 1.4$. As specified in the problem description, the domain size is $[-4; 4]^3$ discretized with a (structured) Cartesian hexahedral grid. The analytic initial solution is propagated in time with supersonic outflow applied at all the domain boundaries. The global L_2 entropy error is computed over time to assess the accuracy of the method. In contrast to the one-dimensional version of the problem, the three-dimensional version of the problem is not smooth in the time period $0 < t < 3$ anymore. The exact solution starts to develop a 'kink-type' structure and thus only $1st$ order for high resolutions can be observed at time $t = 3$. Investigations seem to point towards a smooth solution for the two-dimensional case up to $t = 2$, but convergence results for the three dimensional version are unclear and up to now, no specific time interval with guaranteed smooth solution has been found for the three dimensional test case in our investigations. Table 1 shows the convergence results for an approximation with polynomial degree $N = 4$ for different end times $t = 0, 1, 2, 3$. No clear convergence order can be observed with a general trend that the convergence order is sub-optimal. This effect seems to be magnified for higher order approximations which can be observed in Tbl. 2, where the errors for a calculation with polynomial degree $N = 5$ are plotted. In this test suite, the experimental order of convergence (EOC) has a tendency towards 1 at end time $t = 3$. All computations where performed on a Nehalem cluster using 64 processors. The Taubench on a single core needs about 7.6 seconds in average. The computation for $N = 4$ and 32^3 grid cells needs about 8 minutes wall clock time and the case $N = 5$ with 32^3 cells needs 14 minutes on 64 processors.

	$t = 0$		$t = 1$		$t = 2$		$t = 3$	
cells	L_2	EOC	L_2	EOC	L_2	EOC	L_2	EOC
8^3	2.10E-3	-	8.30E-3	-	4.22E-2	-	6.11E-2	-
16^3	1.68E-5	7.0	1.41E-3	2.6	2.09E-3	4.3	2.35E-3	4.7
32^3	3.13E-7	5.7	1.88E-5	6.2	1.50E-4	3.8	7.43E-4	1.7

Table 1: Experimental order of convergence for different end times t with polynomial degree $N = 4$.

	$t = 0$		$t = 1$		$t = 2$		$t = 3$	
cells	L_2	EOC	L_2	EOC	L_2	EOC	L_2	EOC
8^3	6.18E-4	-	4.20E-3	-	9.81E-3	-	1.37E-2	
16^3	2.71E-6	7.8	7.50E-4	2.5	1.93E-3	2.3	1.63E-3	3.1
32^3	2.42E-8	6.8	7.25E-6	6.7	3.28E-5	5.9	5.46E-4	1.6

Table 2: Experimental order of convergence for different end times t with polynomial degree $N = 5$.

The remaining task for this test case is to find a suitable time interval for the three-dimensional version where the solution remains smooth or to increase the adiabatic coefficient g from 1.4 up to the value 3.0 for final data submission. Those results and an investigation of the breakdown of the order of convergence in the three dimensional test case will be presented at the conference.