Parallelisation of implicit DGFEM schemes for fluid flow problems by the Schwarz alternating method

Aleš Pecka[1], Ondřej Bublík[2], Jan Vimmr[3]

Parallelisation of computational algorithms rapidly decreases computational time needed to solve a given boundary value problem. Hence, more accurate results can be achieved. The domain decomposition method is a commonly used method for distributing the computation among several computers in a computer network. It is based on parting the boundary value problem into subproblems by dividing the domain into subdomains.

The DG method along with an appropriate implicit time integrating scheme is a powerful tool for solving fluid flow problems for its stability, robustness and high order of accuracy. We subject the previously developed implicit DG scheme to parallelisation. Following the Schwarz alternating method (an example of domain decomposition methods), originally proposed by Schwarz (1870), we divide the computational domain Ω into a set of overlapping subdomains $\omega_1, \omega_2, \ldots$ Each of the computers in the computer network performs an independent computation on a different subdomain ω_i as illustrated in Figure 1. The boundary condition for each subproblem needs to be defined on the part of the boundary $\partial\omega_i$ whereever it is distinct from $\partial\Omega$. To this end, we take the values of elements that lie at the boundary $\partial\omega_i$ and use their values for the Dirichlet boundary condition. After each time step, the data from the intersection $\omega_{i,j} = \omega_i \cap \omega_j$ is exchanged between the i-th and the j-th computer. This ensures that the solution may propagate from one subdomain to another. We propose that only one iteration of the Schwarz alternating method is needed if the minimum width of overlaps $\omega_{i,j}$ is chosen larger enough so that the particles of the fluid cannot cross the overlaps in one time step.

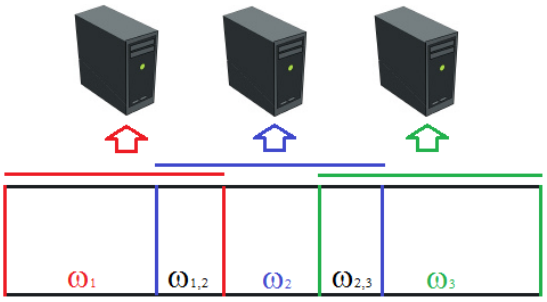

Figure 1: Partition of computational domain Ω into three overlapping subdomains ω_1, ω_2 and ω_3 with overlaps $\omega_{1,2}$ and $\omega_{2,3}$.

Apart from the parallelisation at the level of the computer network, we also employ parallelisation within individual computers. On each computer, a system of linear equations is solved by the GMRES algorithm with the block diagonal Jacobi preconditioner. Here the subject for parallelisation is the GMRES algorithm itself. More specifically, the individual vector

[1] Ph.D. student of Applied Science and Informatics, field Applied Mechanics, e-mail: pecka@ntis.zcu.cz
[2] NTIS - New Technologies for Information Society, e-mail: obublik@ntis.zcu.cz
[3] NTIS - New Technologies for Information Society, e-mail: jvimmr@ntis.zcu.cz

operations involved in GMRES (i.e. matrix-vector multiplications) are parallelised among CPU cores of a single computer. This type of parallelisation is inefficient when applied to a computer network, as the data transfer among computers would be to frequent. On the other hand, it is suitable to be performed among CPU cores of each computer because they share memory.

The following test problem shows speedup of parallelisation among computers by the Schwarz alternating method and parallelisation of GMRES solver among cores of a single computer and a comparison of the two parallelisotion approaches. We performed the benchmark on a cluster of nine computers each with two Intel Xeon E5-2630 v2 (6 cores, 12 threads, 2.60 GHz) processors and 64 GB RAM. This corresponds to a total of 108 cores and 576 GB RAM. Let us consider a viscous flow in a convergent channel with the Reynolds number $\mathrm{Re} = 5000$. The geometry of the problem and the Mach contour at steady state are shown in Figures 2 and 3. We consider an unstructured triangular mesh of 82 299 fairly evenly distributed elements with maximum size $h = 0.005$. We set stagnation pressure $p_0^{\mathrm{in}} = 1$, stagnation density $\rho_0^{\mathrm{in}} = 1$ and angle of attack $\alpha^{\mathrm{in}} = 0$ at the inlet boundary on the left-hand side. At the outlet on the right-hand side we prescribe pressure $p^{\mathrm{out}} = 0.9$. The top and the bottom is a solid boundary on which we apply no-slip boundary condition. In this benchmark, we use backward Euler method in time and the DG method of 4th order in space. We measure the computational time after 50 time steps. Speedup vs. number of nodes for various numbers of threads is plotted in Figure 4.

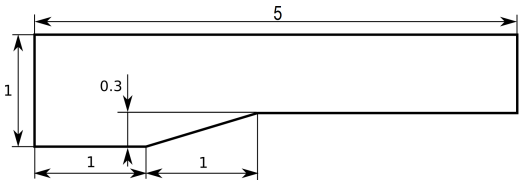

Figure 2: Geometry

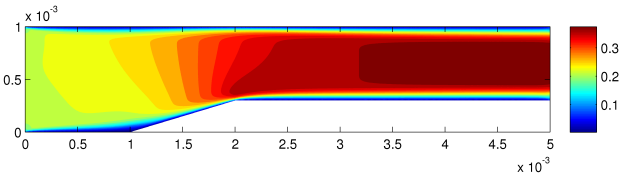

Figure 3: Mach contour at steady state

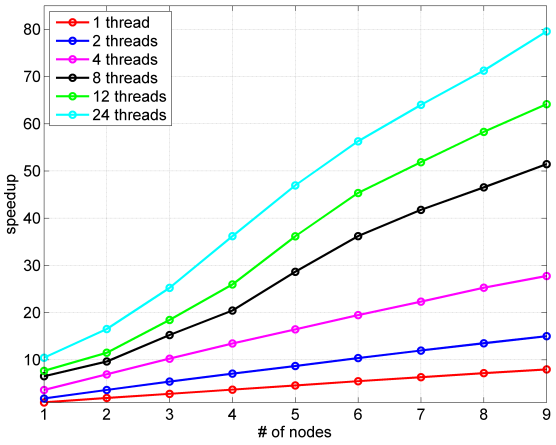

Figure 4: Speedup vs. number of nodes for various numbers of threads

The test problem demonstrates how speedup increases (i.e. computational time drops) with increasing number of computers and CPU cores involved in the computation, see Figure 4. The dependency however is not linear. The overhead is caused by the data transfer, thread management and other imperfections of parallelisation. We achieved speedup of 6.5 when 1 computer with 8 threads was used and speedup of 7.2 when 8 computers each with a single thread was employed. The latter is very close to the ideal speedup of 8. We measured speedup of 79.5 when all 9 computers each with 24 threads was exploited.

Acknowledgement

This work was supported by the project SGS-2016-038.

References

Schwarz, H. A., 1870. Über einen Grenzübergang durch alternierendes Verfahren *Vierteljahrsschrift der Naturforschenden Gesellschaft*. Vol. 15. pp. 272-286.