Sources of Parallel Inefficiency for Incompressible CFD Simulations

Sven H.M. Buijssen[1,2], Stefan Turek[1]

sven.buijssen@math.uni-dortmund.de,

stefan.turek@math.uni-dortmund.de / ture@featflow.de

[1] Institute of Applied Mathematics
University of Dortmund
Germany

[2] Institute of Applied Mathematics
University of Heidelberg (IWR)
Germany

Talk Topic

What will this talk be about?

- Huge systems of (non-)linear equations arising from the discretisation of PDEs, e.g. in CFD, are often solved with parallel multigrid methods.
- Multigrid methods are prefered for their optimal numerical complexity for ill-conditioned PDE problems.
- (Geometric) multigrid methods are strongly depending on the smoothing algorithm used. Many excellent sequential smoothers (SOR, ILU, ...) are highly recursive.
- Their recursive character impedes direct parallelisation.
  $\Rightarrow$ use of block smoothers only $\Rightarrow$ **Consequences?**

Mathematical background

Brief summary of the mathematical background of the program developed:

- parallel, implicit, multigrid-based FEM code for solving the incompressible nonstationary Navier-Stokes equations

$$\mathbf{u}_t - \nu\Delta\mathbf{u} + (\mathbf{u}\cdot\nabla) + \nabla p = \mathbf{f} \quad , \quad \nabla\cdot\mathbf{u} = 0$$

- adaptation of the existing sequential FEATFLOW solver (see `www.featflow.de`),
  a fast and well-known CFD package under constant development since the late 1980s.

Mathematical background

Numerical steps how the Navier-Stokes equations were discretised and solved:

- (implicit) 2nd order discretisation in time
  (Fractional-Step-Θ-, Crank-Nicolson-scheme)
- finite element approach for space discretisation
  (non-parametric non-conforming $\tilde{Q}_1/Q_0$ ansatz)
- Stabilisation of convective term with (Samarskij)
  Upwind scheme
- time steps chosen adaptively

Mathematical background

Numerical steps how the Navier-Stokes equations were discretised and solved (cont.):

- Within each time step:
  - Discrete Projection method to decouple velocity-pressure problem
  - The resulting nonlinear Burgers equation in $\mathbf{u}$ is solved by fixed point defect correction method (outer loop) and multigrid (inner loop)
  - Remaining linear problem in p (ill-conditioned!) is solved with multigrid preconditioned conjugate gradient method

Mathematical background

Numerical steps how the Navier-Stokes equations were discretised and solved:

- FEM in space, implicit discretisation in time
- Within each time step:
  - Discrete Projection method to decouple velocity-pressure problem
  - The resulting nonlinear Burgers equation in $\mathbf{u}$ is solved by fixed point defect correction method (outer loop) and multigrid (inner loop)
  - Remaining linear "Pressure Poisson problem" (ill-conditioned!) is solved with multigrid preconditioned conjugate gradient method

Mathematical background

Parallelisation strategy

- Domain decomposition using graph-oriented partitioning tool (Metis or PARTY)
- Uniform refinement of each parallel block
- local communication between at most two adjacent parallel blocks (due to FEM ansatz: $\tilde{Q}_1/Q_0$!)
- block smoothing

Mathematical background

Idea of block smoothing:

- Avoid direct parallelisation of global smoother (significant amount of communication!)
- Instead: Apply the same smoothing algorithm within each parallel block only

Mathematical background

Consequences of block smoothing:

With increasing number of parallel processes:

- It takes more than 1 iteration to spread information across the grid (weakened smoothing property)

In limit case:
block smoother turns into simple Jacobi iteration!

- Thus, the number of multigrid sweps will increase.

  Significantly?

  – for Burgers equation: probably not, good conditioned (scaling with time step k)

  – **for discrete Pressure Poisson equation: probably, problem with condition of $\mathrm{O}(h^{-2})$**

Numerical section

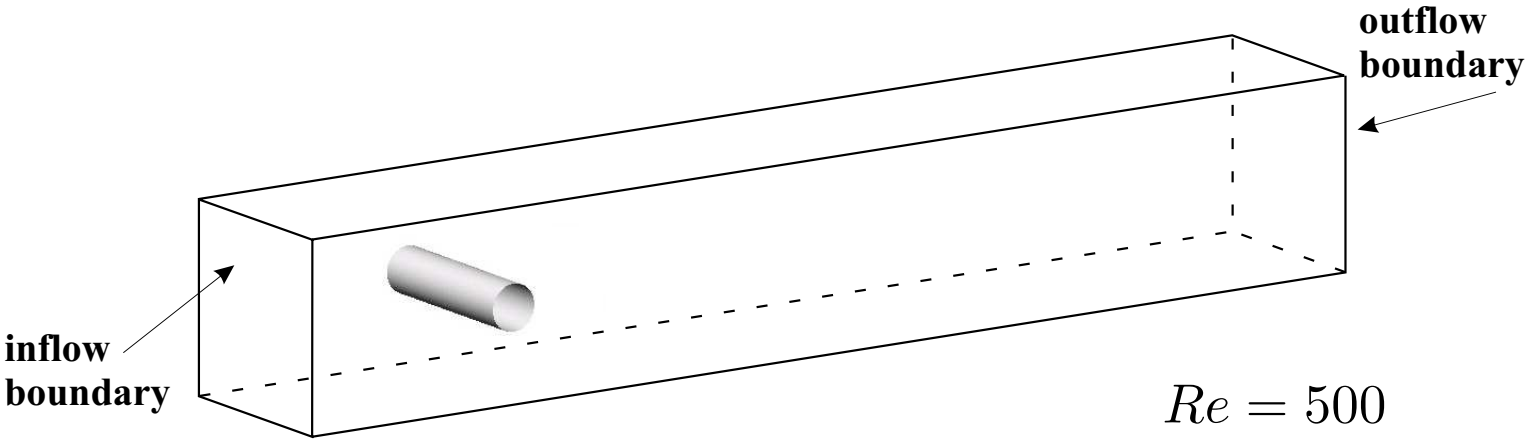
The parallel C++ code has been

- successfully tested on different modern platforms:
  Cray T3E 1200 (Jülich), Hitachi SR8000 (Munich),
  Alpha Workstation Cluster (Wuppertal), Linuxcluster
  HELiCS (Heidelberg), ...
- used to simulate different geometries with varying
  mesh deformations (aspect ratios)
  (typical test problems have been performed as well as
  some relevant industrial problems)
- compared with the sequential F77 counterpart from
  the FEATFLOW package

Numerical section

The typical effects observed, illustrated on the basis of the DFG benchmark from 1995:

channel flow around a cylinder:



$Re = 500$

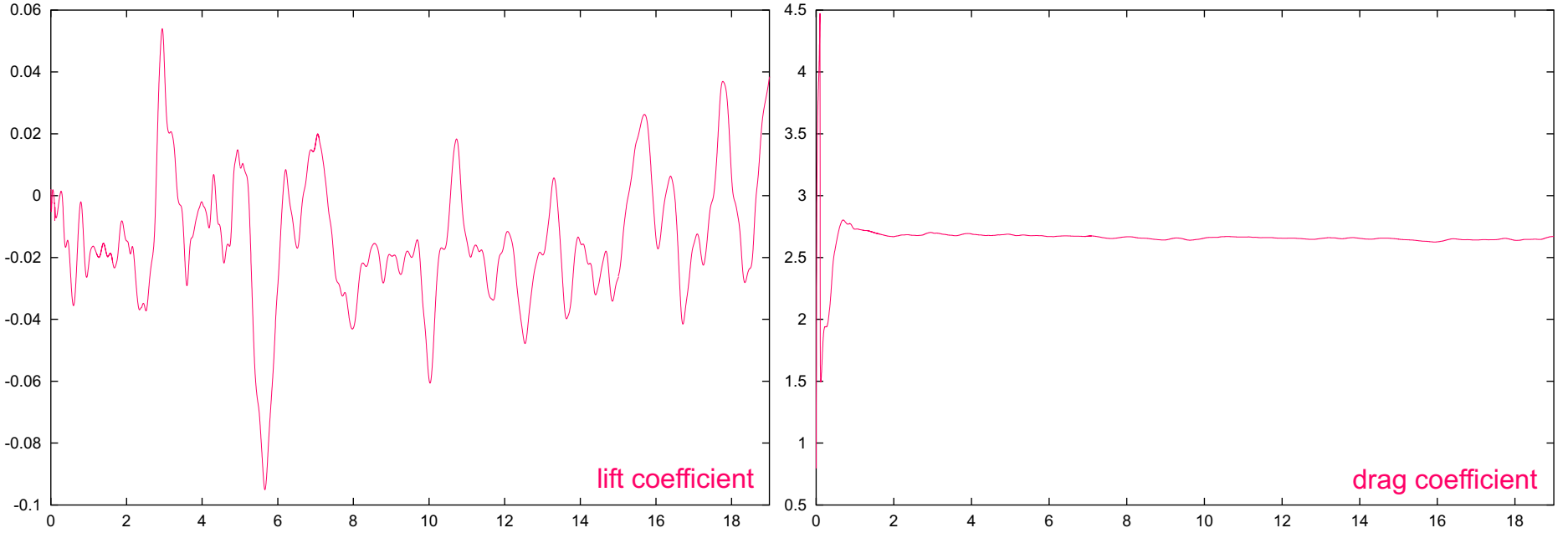
aspect ratio ≈ 20

Numerical section

We did some long term simulation

- d.o.f.: 32 million resp. 250 million
- #time steps: 6.500 ($T_{End} = 20s$)

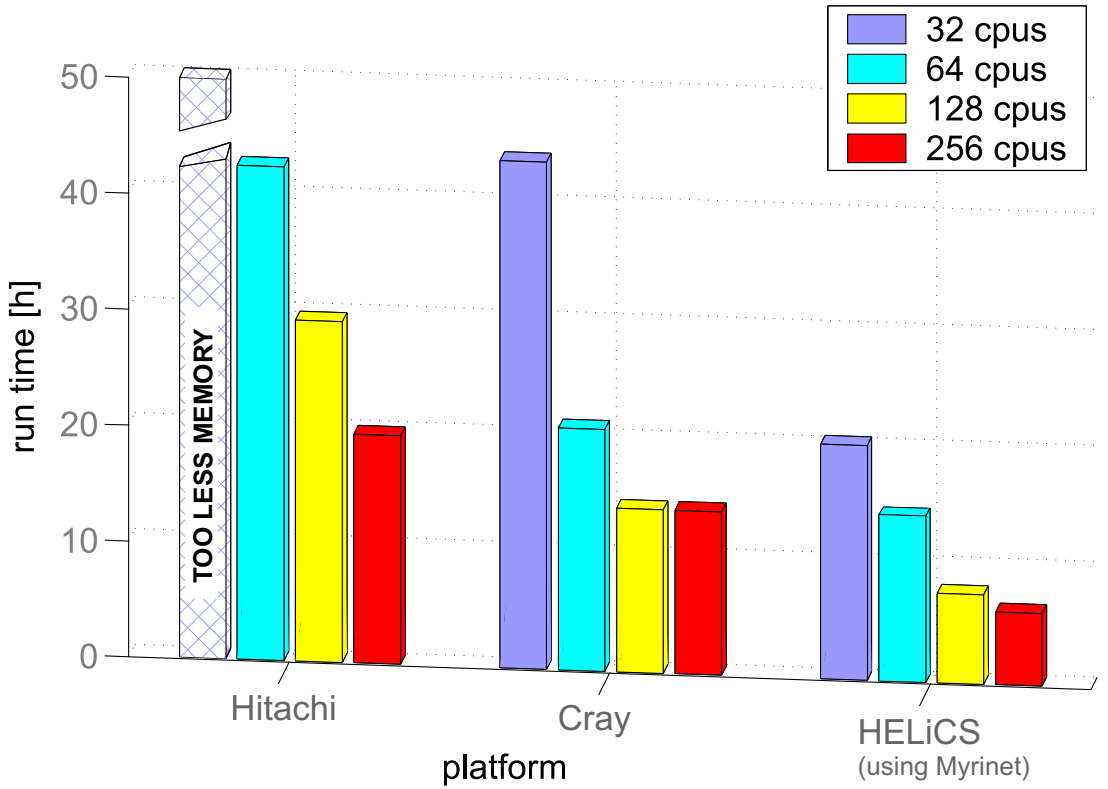
and computed lift and drag coefficients



Numerical section

... and examined how the program scales

(32 million d.o.f., 1300 time step)



One observes:

decrease of run time, but less than expected

Numerical section

Reasons why run times decrease slower than expected:

- Amdahl's law: sequential part of program limits speedup
- increasing communication loss
  with increasing number of cpus
- **solving Pressure Poisson equation takes longer**
  with increasing number of cpus (i.e. parallel blocks):
  #iterations plus **24 percent** when stepping from 32 to 256 cpus.
  (even worse on coarser grids: stepping from 1 to 256 cpus $\hookrightarrow$ increase by a factor of 4-5)

Numerical section

Block-ILU smoother was used for this tests. Why not use some other smoother which numerically deteriorates less?

- A grid with fairly high aspect ratios (≈ 20) was used in order to
  - limit problem size as well as to
  - compute boundary layers near the cylinder accurately.
- Simpler smoothers scarcely converge (Jacobi/SOR) using this grid.
- The very same problems occur with preconditioned Krylov space methods: weakened smoothing property when used as Block smoother

Conclusions

To sum up, we can say

- applying block smoothers in parallel CFD solvers works and helps reducing communication loss, especially in combination with non-conforming FEM.
- Price to pay: On grids with more than moderate aspect ratios, ill-conditioned (sub-)problems will suffer significant deterioration of numerical efficiency with increasing number of cpus used.

  For real-world problems, loss in parallel efficiency can be equally due to communication and significant deterioration of numerical efficiency.