To Appear in

High Performance Computing in Science and Engineering 2004

Springer, 2004

parpp3d++ – a parallel HPC code for the incompressible nonstationary Navier–Stokes equations

Sven H.M. Buijssen and Stefan Turek

University of Dortmund, Institute for Applied Mathematics and Numerics,
Vogelpothsweg 87, 44227 Dortmund, Germany
sven.buijssen@math.uni-dortmund.de, ture@featflow.de

Parallel multigrid methods belong to the most prominent tools for solving huge systems of (non-)linear equations arising from the discretisation of PDEs, as for instance in Computational Fluid Dynamics (CFD). However, the quality of (parallel) multigrid methods in regard of numerical and computational complexity mainly stands and falls with the smoothing algorithms ('smoother') used. Since the inherent highly recursive character of many global smoothers (SOR, ILU) often impedes a direct parallelisation, the application of block smoothers is an alternative. However, due to the weakened recursive character, the resulting parallel efficiency may decrease in comparison to the sequential performance, due to a weaker total numerical efficiency. Within this paper, we show the consequences of such a strategy for the resulting total efficiency on the Hitachi SR8000-F1 if incorporated into the parallel CFD solver *parpp3d++* for 3D incompressible flow. Moreover, we analyse the numerical losses of parallel efficiency due to communication costs and numerical efficiency on several modern parallel computer platforms.

1 Numerical and Algorithmic Approach

parpp3d++ is a parallel 3D code for the solution of the incompressible nonstationary Navier-Stokes equations

$$\mathbf{u}_t - \nu\Delta\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p = \mathbf{f}, \quad \nabla\cdot\mathbf{u} = 0 \tag{1}$$

This code is an adaptation of the existing sequential FeatFlow solver (see www.featflow.de). For a detailed description of the numerical methods applied see [1, 7]. Here we restrict ourselves to a very brief summary of the mathematical background. Equation (1) is discretised separately in space and time. First, it is discretised in time by one of the usual second order methods known from the treatment of ordinary differential equations (Fractional-Step-θ-scheme, Crank-Nicolson-scheme). Space discretisation is performed by

applying a special finite element approach using the non-conforming $\tilde{Q}_1/Q_0$ spaces (in the non-parametric version). The convective term is stabilised by applying an upwind scheme (weighted Samarskij upwind). Adaptive time stepping for this implicit approach is realised by estimating the local truncation error. Consequently, solutions at different time steps are compared. Within each time step the coupled problem is split into scalar subproblems using the Discrete Projection method. We obtain definite problems in $\mathbf{u}$ (Burgers equations) as well as in p (Pressure-Poisson problems). Then we treat the nonlinear problems in $\mathbf{u}$ by a fixed point defect correction method, the linearised nonsymmetric subproblems are solved with multigrid. For the ill-conditioned linear problems in p a preconditioned conjugate gradient method is applied. As preconditioner, multiplicative as well as additive multigrid (using Jacobi/SOR/ILU smoothers) has been implemented.

In order to parallelise the multigrid method the coarse mesh is split into parallel blocks by a graph-oriented partitioning tool (Metis [4], PARTY [6]). Subsequently, each block is uniformly refined. Consistency with the sequential algorithm (MV application, grid transfer) is guaranteed through local communication between at most two parallel blocks (this is possible because of the face-oriented $\tilde{Q}_1/Q_0$ ansatz). The inherent recursive character of global smoothers impedes a direct parallelisation. Therefore, the global smoothing is replaced by smoothing within each parallel block only (block smoothers). To minimise the communication overhead for solving the coarse grid problem, it is treated on a single processor with an optimised sequential algorithm. The cost is two global communications (setting up the right side and propagation of the solution vector).

2 Experiences on Hitachi SR8000-F1

The code has been written in C++ and uses MPI for communication. It has been tested [1] for many configurations including standard benchmarks like Lid-Driven-Cavity and the 3D configurations of the '1995 DFG-Benchmark' defined in [9] as well as some problems with industrial background: computation of drag values on model car surfaces (automotive industry), simulation of molten steel being poured into a mould (steel industry), design of catalytic coated ceramic wall reactors which are used as micro reactors for heterogeneously catalysed synthetic reactions (chemical engineering). Hexahedral meshes with aspect ratios up to 500 and problems with 250 million degrees of freedom in space and up to several thousand time steps have been handled successfully.

It was not before the completion of the program's implementation that access to Hitachi SR8000-F1 at Leibniz-Rechenzentrum Munich was gained. Moreover, the design of the program has been chosen to incorporate only basic elements of the ISO92 reference on C++ and to solely rely on the MPI 1.2 specification. This to guarantee the utmost level of portability. As a conse-

quence, none of the SR8000-F1's vector processing capabilities are explicitly deployed. The system is merely used as a MPP unit among others. For code optimisation we rely on the Hitachi C/C++ compiler.

During the first year on Hitachi SR8000-F1, KCC and g++ had been employed. Having to overcome serious compilation errors with both of them, the run times we finally observed with g++ were rather disappointing. The problems with KCC could never be solved.

Since the first beta release of the vendors own C++ compiler (sCC) in June 2002, things have improved – as have run times. A comparison with Cray T3E-1200 (Research Centre Jülich) and the Linux PC cluster HELICS (IWR Heidelberg) is performed in section 3. Annoying but seemingly inevitable[1] are sCC's long compilation times of 8–9 hours whereas g++ needs as less as ten minutes on a Pentium 4 with 1.8 GHz – despite the fact that the usual suspects, C++ templates, are rarely used.

3 Comparison of Run Times

This section will deal with a comparison of run times on three different types of MPP units: a low-cost Linux PC cluster consisting of 256 dual-processor nodes of AMD Athlon MP 1.4 GHz type (HELICS, IWR Heidelberg), a Cray T3E-1200 (Research Centre Jülich) and LRZ's SR8000-F1.

3.1 Definition of Benchmark Problem '1995 DFG-3D2Z'

We give merely a brief summary of the test configuration. The complete information containing all definitions (and results) can be found in [9]. An incompressible Newtonian fluid is considered for which the conservation equations of mass and momentum read

$$\frac{\partial U_i}{\partial x_i} = 0 \,, \qquad \rho\frac{\partial U_i}{\partial t} + \rho\frac{\partial}{\partial x_j}\left(U_j U_i\right) = \rho\nu\frac{\partial}{\partial x_j}\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right) - \frac{\partial P}{\partial x_i} \,.$$

The notations are: time t, cartesian coordinates $(x_1, x_2, x_3) = (x, y, z)$, pressure P and velocity components $(U_1, U_2, U_3) = (U, V, W)$. The kinematic viscosity is defined as $\nu = 10^{-3}\,\mathrm{m}^2/\mathrm{s}$, and the fluid density is $\rho = 1.0\,\mathrm{kg}/\mathrm{m}^3$. As problem configuration the flow around a cylinder with circular cross–section in a channel is considered. See Fig. 1 for geometry and boundary conditions. The channel height and width is $H = 0.41\,\mathrm{m}$ and $D = 0.1\,\mathrm{m}$ is the cylinder diameter. The Reynolds number is defined by $Re = \bar{U}D/\nu$ with the mean velocity $\bar{U}(t) = 4U(0, H/2, H/2)/9$. The inflow condition is

$$U(0, y, z) = 16U_m yz(H - y)(H - z)/H^4, \quad V = W = 0$$

with $U_m = 2.25\,\mathrm{m}/\mathrm{s}$.

[1] As explained on LRZ's web pages, see [5].

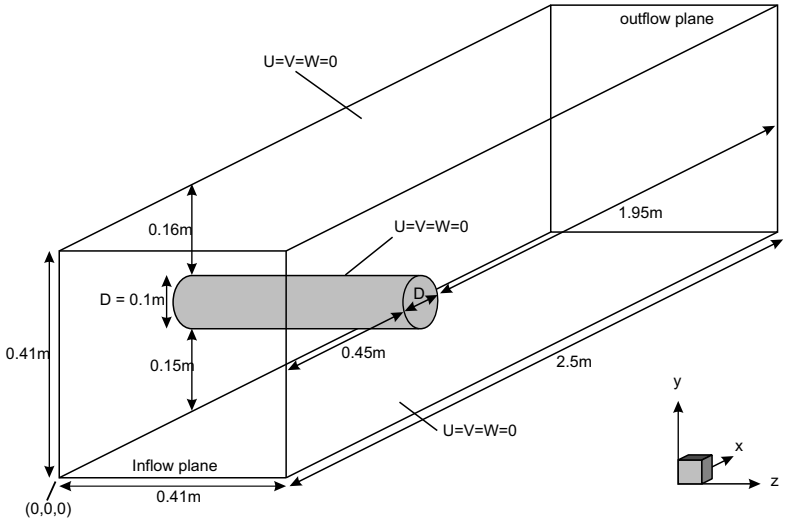

Fig. 1. Geometry of 3-d test case '1995 DFG-3D2Z' with boundary conditions

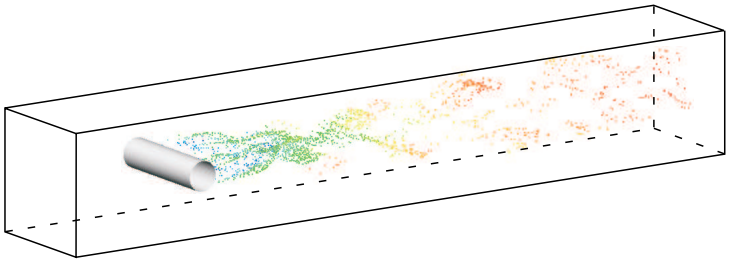

Fig. 2. Resulting Van Kármán vortex shedding behind the cylinder (particle tracing)

Table 1. Run times of benchmark problem '1995 DFG-3D2Z'

	#procs	time	comm.
Cray T3E-1200	64	20 h 51'	17%
	128	14 h 06'	29%
	256	14 h 05'	42%
HELICS	32	20 h 13'	18%
	64	16 h 37'	25%
	128	7 h 42'	37%
	256	9 h 46'	62%
SR8000-F1	64	42 h 34'	16%
	128	29 h 22'	19%
	256	19 h 41'	39%

3.2 Results of Benchmark Problem '1995 DFG-3D2Z'

A triangulation of the geometry was made leading to a problem size in space of 32 millions degrees of freedom. For this test suite time steps were fixed a priori such that exactly 434 time steps with Fractional-Step-θ-scheme were necessary to simulate $T = [0, 1]$.

Table 1 shows the resulting run times with 64, 128 and 256 processes on each of the platforms stated above. Additionally, the relative amount of time spent in communication routines was gathered. From this data it can be easily seen that the scaling on Hitachi SR8000-F1 is satisfying (relative speedups of 1.4 and 1.5 respectively). The increase in communication loss is least of the triple. As far as actual run times are concerned, however, things look more sombre. SR8000-F1 is conspicuously in last position. It needs as much as 256 processes to beat run times on the PC cluster when applying only 32 processes.

This is not an isolated observation, but has been perceived for other compiler settings, problem sizes, degrees of parallelism and geometries, too. Nevertheless, we proceed with optimising the code on the Hitachi system; SR8000-F1 is still used as a host to simulate current research projects like the BMBF project introduced in the subsequent section.

4 Current Computations

Currently, SR8000-F1 is used to optimise the design of ceramic wall reactors as part of BMBF project 03C0348A.[2] The intension is to develop ceramic wall reactors and ceramic plate heat exchangers as micro reactors for heterogeneously catalysed gas phase reactions. By appropriate calibration of the catalytic activity, diffusive mass transport and heat removal an optimal temperature distribution can be attained which in turn leads to a significant increase in performance of the reactor. A general and economical reactor concept demanding low development efforts is strived for.

The outer dimensions of the workpiece are fixed as are inflow and outflow nozzle. Number, shape and position of the "obstacles" in the interior are parameters to generate a uniformly distributed flow. Figure 3 gives a general survey of the geometry.

Refining the initial coarse grid four times leads to problem sizes which are in the range of 30–80 million degrees of freedom. To reach the stationary limit between 20 and 40 time steps are necessary. Availing 128 parallel processes, the computations take 12–18.5 h on SR8000-F1. Figure 5 shows the velocity distribution in x-direction on a cutplane through the centre of gravity of the geometry for several of the tested designs.

[2] This project is a cooperation with the Institute of Chemical Engineering, Chair of Reaction Engineering (TCB), University of Dortmund and the Hermsdorfer Institute for Technical Ceramics.

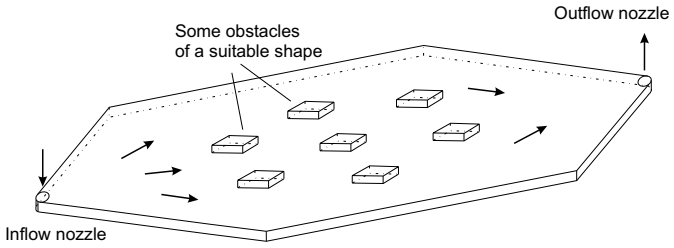

Fig. 3. Sketch of overall geometry of ceramic wall reactors and flow directions

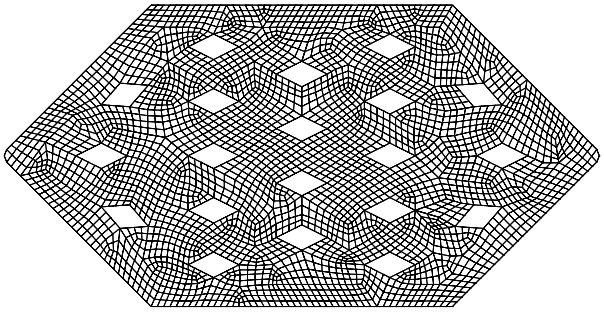

Fig. 4. Typical grid of a ceramic wall reactor (refinement level 2, 2-d top view)

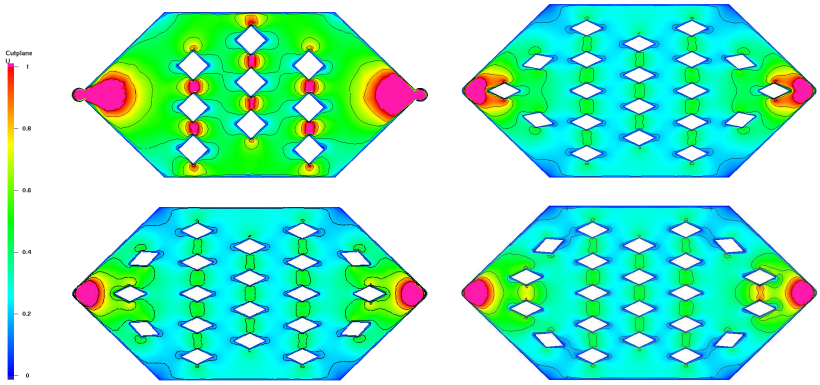

Fig. 5. Some of the two dozen different geometries examined so far

In order to investigate whether a grid-independent solution had been gained, computations on an even finer level were started (245–670 million degrees of freedom in space). But it turned out that each time step would take roughly 2.5 h wall clock time (using 128 processes) such that further investigations at this problem size were cancelled to not deplete the remaining CPU quota.

5 Examination of Parallel Efficiency

Over and above that we did some studies on the scalability of the parallel implementation. As already mentioned in [2] and [1], there are two major effects which affect the run times for a fixed problem size at varying degrees of parallelism. First, there is the inevitable contribution of communication loss to run times. But due to the chosen non-conforming trilinear finite element spaces, the communication needs are limited to a near minimal amount (face–neighbouring elements of at most two parallel blocks only). In general, communication accounts within this code for half of the losses in parallel efficiency. The second major effect is the deterioration of the multigrid solver for the Pressure–Poisson problems: the number of iterations necessary to solve these subproblems usually increases by a factor of 3-6 if stepping from 1 to 256 processes. The factor is problem-dependent, or to state it more precisely, it is dependent on the triangulation's aspect ratio. The bigger the aspect ratios, the worse the deterioration. This dependency is rather simple to explain: multigrid methods fatefully depend on the smoothing property of their smoothers applied internally. As mentioned in Sect. 1, the parallel algorithm replaces global smoothing by block smoothing. This means that with each additional parallel block (i.e. each additional parallel process), it will take more iterations to spread information from one end of the domain to the other. A process that takes only a single iteration in sequential. Any arbitrary smoothing algorithm applied successfully in sequential multigrid (SOR, ILU etc.) will suffer from the blocking strategy and in the marginal case end up as a Block-Jacobi algorithm. But before actually implementing the parallel algorithm, the impact of this theoretical considerations could not be estimated.

As a consequence of this numerical deterioration more time is spent solving the Pressure–Poisson problem in each time step, increasing from rougly 10 percent of overall run time for quasi-sequential program runs to more than 50 percent for massive parallel runs.

One aspect about the deterioration (whether to be appraised positive or negative is left to the reader) is as follows: the deterioration drops as the number of processes increases. Comparing the total iteration count for the benchmark problem presented in Sect. 3 reveals that it is basically identical if a certain degree of parallelism is reached (while maintaining a moderate problem size for each individual process): 2604 iterations for a 64-process-run, 2606 iterations for 128 processes, 2612 iterations for 256 processes. Similar results hold for different geometries, too. Thus, for massive parallel runs (≥ 64 processes) the additional losses in parallel efficiency are due to increased communication needs.

6 Conclusion and Outlook

The detailed examinations in [1] show that the realised parallel version of an optimised sequential 3D-CFD solver has (at least) three sources of parallel in-

efficiency: Besides the obvious overhead due to inter-process communication, the quality of the C++ compilers and the special structure of the Hitachi is an important factor which requires further research activities. However, the biggest loss is due to the weakened numerical efficiency since only blockwise smoothers can be applied. Consequently, the number of multigrid cycles strongly depends on the anisotropic details in the computational mesh and the number of parallel processes. As a conclusion, for many realistic configurations, more than 10 processors are needed to beat the optimised sequential version in FeatFlow. Thus, new and improved numerical and algorithmic techniques have to be developed to exploit the potential of recent parallel supercomputers and of modern Mathematics at the same time (see [8] for a discussion).

Therefore, the central point of our present and future research is the development of new mathematical components – FEM discretisations, adaptivity and (parallel) multigrid solvers – and their realisation in software packages which directly include tools for parallelism and hardware-adapted high-performance in low level kernel routines. The code generation uses the new FEAST software in order to achieve highest computational efficiency. These software developments can be viewed as 'basic research' in the field of mathematical software for PDEs. Hence we will continue our work with the parallel 3D adaptation parpp3d++ from the FeatFlow package which is presently applied on several parallel computers to prototypical configurations similar to the shown geometries. This parallel 3D code is our candidate for all further developments which aim to incorporate the high-performance FEAST techniques into this CFD tool in order to achieve highest computational efficiency on modern computers in combination with the 'best' numerical approaches.

References

1. Buijssen, Sven H.M. Numerische Analyse eines parallelen 3-D-Navier-Stokes-Lösers. Master's thesis, Universität Heidelberg, October 2002. `http://www.mathematik.uni-dortmund.de/lsiii/php/showpdffile.php?Buijssen2002`.
2. Buijssen, Sven H.M. and Turek, Stefan. Sources of parallel inefficiency for incompressible CFD simulation. In Monien, B. and Feldmann, R., editors, *Proceedings 8th International Euro-Par Conference*, LNCS. Springer, January 2002. Paderborn, Germany, August 27-30.
3. HELICS – HEidelberg LInux Cluster System. `http://www.helics.de/`.
4. Karypis, G. and Kumar, V. METIS - A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices. `http://www-users.cs.umn.edu/~karypis/metis/index.html`, January 1998.
5. LRZ Munich. System Description. `http://www.lrz-muenchen.de/services/compute/hlrb/system-en/`.
6. Preis, R. and Diekmann, R. The PARTY Partitioning - Library, User Guide - Version 1.1. `http://www.uni-paderborn.de/fachbereich/AG/monien/RESEARCH/PART/party.html`, January 1996.

7. Turek, S. *Efficient solvers for incompressible flow problems: An algorithmic and computational approach*. Springer, 1999.
8. Turek, S., Becker, C., and Kilian, S. Hardware-oriented Numerics and cocepts for PDE software. Technical report, Universität Dortmund, Vogelpothsweg 87, 44227 Dortmund, June 2003. to appear in ICCS.
9. Turek, S. and Schäfer, M. Benchmark computations of laminar flow around cylinder. In E.H. Hirschel, editor, *Flow Simulation with High-Performance Computers II*, volume 52 of *Notes on Numerical Fluid Mechanics*. Vieweg, 1996. co. F. Durst, E. Krause, R. Rannacher.