Karlsruhe Institute of Technology

Scalability Study of HiFlow^3 based on a Fluid Flow Channel Benchmark

V. Heuveline, E. Ketelaer, S. Ronnås, M. Schmidtobreick, M. Wlotzka

No. 2012-05

Preprint Series of the Engineering Mathematics and Computing Lab (EMCL)

KIT – University of the State of Baden-Wuerttemberg and
National Research Center of the Helmholtz Association

www.emcl.kit.edu

Preprint Series of the Engineering Mathematics and Computing Lab (EMCL)
ISSN 2191–0693
No. 2012-05

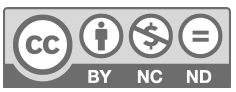
Impressum

Karlsruhe Institute of Technology (KIT)
Engineering Mathematics and Computing Lab (EMCL)

Fritz-Erler-Str. 23, building 01.86
76133 Karlsruhe
Germany

KIT – University of the State of Baden Wuerttemberg and
National Laboratory of the Helmholtz Association

Published on the Internet under the following Creative Commons License:
http://creativecommons.org/licenses/by-nc-nd/3.0/de .



Scalability Study of HiFlow^3 based on a Fluid Flow Channel Benchmark

Vincent Heuveline[*], Eva Ketelaer[†], Staffan Ronnas[‡], Mareike Schmidtobreick[§], Martin Wlotzka[¶]

Engineering Mathematics and Computing Lab (EMCL)
Karlsruhe Institute of Technology (KIT)

Abstract: Exploiting the compute power of high performance computing clusters efficiently is a key ingredient in order to solve large, fully coupled systems modeled by partial differential equations with high accuracy. We study strong and weak scalability properties of the parallel Finite Element software package HiFlow^3 for a challenging instationary 3D fluid flow problem. For this benchmark study, we ran several simulations with up to 10 millions of unknowns using up to 512 cores on the bwGRiD cluster in Karlsruhe. For large problem sizes, the software package showed good characteristics regarding efficiency and speedup.

1 Introduction

Numerical simulations often play a key role in the analysis of complex physical and technical processes. Many real world phenomena can be modeled mathematically by partial differential equations (PDEs). Examples include the numerical simulation of tropical cyclones which influence the global weather forecast, as well as computational studies of the human respiratory system. Performing accurate simulations

[*] vincent.heuveline@kit.edu
[†] eva.ketelaer@kit.edu
[‡] staffan.ronnas@kit.edu
[§] mareike.schmidtobreick@kit.edu
[¶] martin.wlotzka@kit.edu

often relies on computational resources only available on high performance computing (HPC) platforms. Approximate solutions of the underlying system of PDEs can be computed using appropriate numerical discretization methods. A common discretization approach is the Finite Element Method (FEM), which usually results in large, sparse, and fully coupled linear systems with up to several millions of unknowns. In order to be able to solve such large problems, it is essential to use highly scalable finite element software.
In this paper we analyze weak and strong scalability properties of the HiFlow^3 Finite Element package [Heuveline2010] developed at the Engineering Mathematics and Computing Lab (EMCL). This software platform aims at providing efficient and accurate solvers for complex models. The study is based on a 3D channel flow benchmark problem defined by the DFG [Turek1996]. It is defined as a fluid flow problem inside a rectangular channel containing a block-shaped obstacle. This benchmark has previously been performed by a large community using a variety of discretization methods and numerical solvers with a strong emphasis on the accuracy of the solution. On the one hand the benchmark is a challenging problem for numerical simulations, and on the other hand there are published results giving a point of comparison for the quality of our solutions. Our study was carried out on the bwGRiD cluster located in Karlsruhe using up to 512 cores distributed over 64 nodes.
In Section 2, the mathematical problem, the solution method and implementation are described. Additionally, in this section the setup for the scalability study is presented. Section 3 is dedicated to the presentation of the results, as well as their analysis. A conclusion and outlook are given in Section 4.

2 Benchmark Formulation

In this section, we present the mathematical setup on which our scalability study is based and depict how we validated the benchmark implementation. The second part of the section gives an overview of the test setup.

2.1 Mathematical Problem

The considered benchmark assumes a three dimensional instationary flow around a block-shaped obstacle in a channel as described in [Turek1996], see Fig. 1. We assume the liquid to be an incompressible Newtonian fluid and model the flow by the Navier-Stokes equations

$$\begin{aligned} u_t - \nu\Delta u + (u\cdot\nabla)u + \frac{1}{\rho}\nabla p = 0, \quad & \text{in } \Omega\times[0,T], \\ \nabla\cdot u = 0, \quad & \text{in } \Omega\times[0,T], \end{aligned}$$

where u denotes the velocity field, p the pressure variable and $\Omega\subset\mathbb{R}^3$ holds. At the inflow Γ_{in} (1a) we set the boundary condition to a Poiseuille profile, at the outflow Γ_{out} (1b) we choose the natural or do-nothing boundary condition and at the solid walls of the channel $\Gamma_{\text{wall}} := \partial\Omega\backslash(\overline{\Gamma_{\text{in}}\cup\Gamma_{\text{out}}})$ (1c) we impose homogeneous Dirichlet boundary conditions (no-slip conditions). As initial condition we set the velocity to zero inside of Ω (1d).

$$u = g, \quad \text{on } \Gamma_{\text{in}}\times[0,T], \tag{1a}$$

$$(-\mathcal{I}p + \nu\nabla u)\cdot n = 0, \quad \text{on } \Gamma_{\text{out}}\times[0,T], \tag{1b}$$

$$u = 0, \quad \text{on } \Gamma_{\text{wall}}\times[0,T], \tag{1c}$$

$$u = 0, \quad \text{in } \Omega\times\{0\}, \tag{1d}$$

where

$$g(0,y,z,t) = \left(16U_m yz\frac{(H-y)(H-z)}{H^4}, 0, 0\right)^\top.$$

The height and width of the channel $H = 0.41\,m$ and the maximum inflow speed $U_m = 2.25\,m/s$ are chosen as in [Turek1996], which yields the Reynolds number $Re = 100$. We discretize this equation with an FEM, see e.g. [Girault1986]. The channel is discretized by hexahedral cells, resulting in the initial mesh covering the computational domain which is shown in Fig. 1.

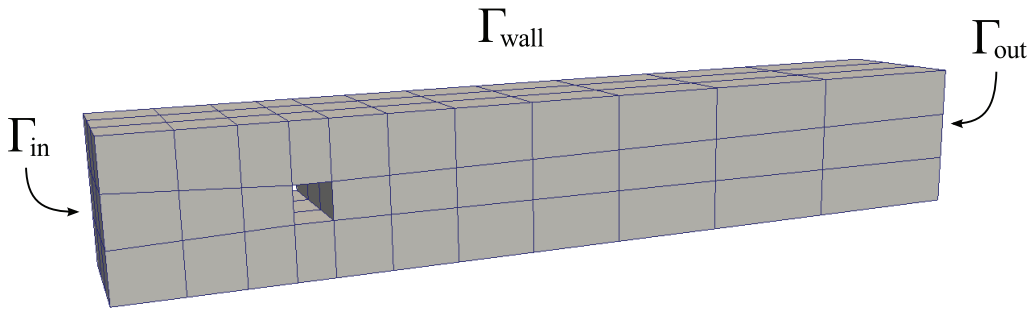


Figure 1: The initial mesh which is a coarse discretization of the computational domain with 96 cells.

The computations for the scalability study were done on meshes resulting from different levels of global refinement applied to the initial mesh. The number of mesh cells and degrees of freedom resulting from these refinements are given in Table 1.

The parallelization concept of HiFlow^3 is based on domain decomposition, as described in [Heuveline2011, Ronnas2011]. In this study, the partitioning of the mesh was computed with the METIS graph partitioner [Karypis2006]. For the spatial discretization of the model problem, we applied a continuous Galerkin FEM. We used the Taylor-Hood element pair, i.e. Q_2-elements for the velocity and Q_1-elements for the pressure. These elements satisfy the discrete inf-sup condition, and thus guarantee stability [Girault1986].

Refinement Level	# Mesh Cells	Problem size, i.e. # DoFs
2	6,144	170,688
3	49,152	1,296,256
4	393,216	10,098,432

Table 1: Information on the number of cells and the number of degrees of freedom (DoFs) that different meshes have. Hereby the mesh level describes how often the mesh has been refined globally on the basis of the initial mesh.

For the time discretization, we used the Crank-Nicolson method applied to an equidistant partitioning of the time interval. We applied Newton's method to the resulting discrete non-linear problems. In each Newton step, we used a right-preconditioned GMRES-method [Saad2003] to solve the linearized system. An incomplete LU-decomposition from the ILU++ library [Mayer2007, Mayer2008] was employed as a preconditioner. The stopping criteria for both the non-linear Newton solver and the linear GMRES solver were set to a relative tolerance of 10^{-6}. This ensures that the residual norm of both the Newton iteration as well as the GMRES iteration decreases by six orders of magnitude compared to the residual of the starting solution, respectively.
The whole solution process was implemented using HiFlow^3, besides the already mentioned mesh partitioning and the preconditioner library.

For this benchmark no analytical solution and no experimental data are available for comparison. In order to obtain a good indication for the accuracy of the benchmark implementation, we compared the drag coefficient of the obstacle to the results of simulations performed with other software by other researchers as listed in [Turek1996]. We computed the drag coefficient in each time-step by evaluating face integrals as described in [John2002]. The initial mesh, see Fig. 1, was globally refined four times. The benchmark was solved in the time interval [0s, 8s] with a time-step size of 0.025s, i.e. 320 time-steps. For the maximum drag coefficient we get 4.84 which lies in the interval of [4.31, 4.88] from [Turek1996]. The authors of that paper expected a time-periodic solution at a Reynolds number of 100. Here, we made the same observation as in [John2002], namely that the numerical solution tends to a stationary solution instead, which is another indication that our benchmark implementation works correctly.

2.2 Setup of Scalability Tests

We study strong and weak scalability for the following benchmark setup. As benchmark metric, we measure the total run-time of the first ten time-steps of the solution process. The problem size is associated to the successively globally refined meshes mentioned in Subsection 2.1. The refinement levels of the mesh and their resulting problem sizes, i.e. the number of unknowns are listed in Table 1.

This benchmark was done on the local part of the bwGRiD-cluster in Karlsruhe. This system is a x86-cluster featuring 10 Blade centers each of which contains 14 HS21 XM Blades. Each node has two quad-core Intel Xeon E5440 (Harpertown) processors with 2.83 GHz clock frequency and 16 GB main memory. The nodes are connected via Infiniband. The operating system is Scientific Linux 5.5.

We executed the simulation with up to 512 cores; for more details see Table 2. Furthermore we took into account the following two limiting factors with respect to the dependency between the refinement levels and the number of cores. On the one hand the memory available on a computing node is important: a minimum number of nodes is needed to cover the memory required by the solver. On the other hand, problem instances with lower refinement levels require less computational effort, which makes the use of a large number of cores meaningless.

3 Results and Analysis

We consider strong and weak scaling, which are the two common concepts of scalability in HPC. In both cases, the total run-time on different numbers of processors is compared. For the strong scaling, the total problem size is fixed, while for the weak scaling the problem size per process is kept constant.

3.1 Strong Scalability

For the strong scalability, we analyzed the speedup and efficiency for each refinement level. The speedup $S(p, N)$ of a parallel execution

			Run-time [sec]			# of GMRES iter.		
Nodes	x	Cores	Level 2	Level 3	Level 4	Level 2	Level 3	Level 4
1	x	1	2,391	27,517		438	778	
1	x	2	1,470	14,076		1,239	2,376	
1	x	4	906	8,078		1,664	2,510	
1	x	8	605	5,887		2,155	2,819	
2	x	8	345	3,074		2,657	2,976	
4	x	8	226	1,644		3,306	3,214	
8	x	8		1,027	10,752		4,133	5,306
12	x	8			6,923			5,234
16	x	8			5,539			5,520
32	x	8			3,023			5,770
64	x	8			2,007			6,652

Table 2: The run-time in seconds and the total number of GMRES iterations needed over the whole solution process. The benchmark was run with different numbers of cores and different levels of refinement of the intial mesh.

with p processors is defined as

$$S(p,N) = \frac{T(p,N)}{T(1,N)} ,$$

where $T(p,N)$ is the run-time for an execution with p processors and problem size N. The efficiency is defined as

$$E(p,N) = \frac{S(p,N)}{p} .$$

Fig. 2 and Fig. 3 show the obtained speedup and Fig. 4 shows the efficiency of our three test series. Note that for the level 4 test series, these quantities are computed relatively to the execution with $p = 64$ processors, since it was not possible to run these tests with fewer processors due to the limited memory of the computing nodes.

There are two aspects which generally degrade the efficiency of our test runs as the number of processors is increased. First, for solving

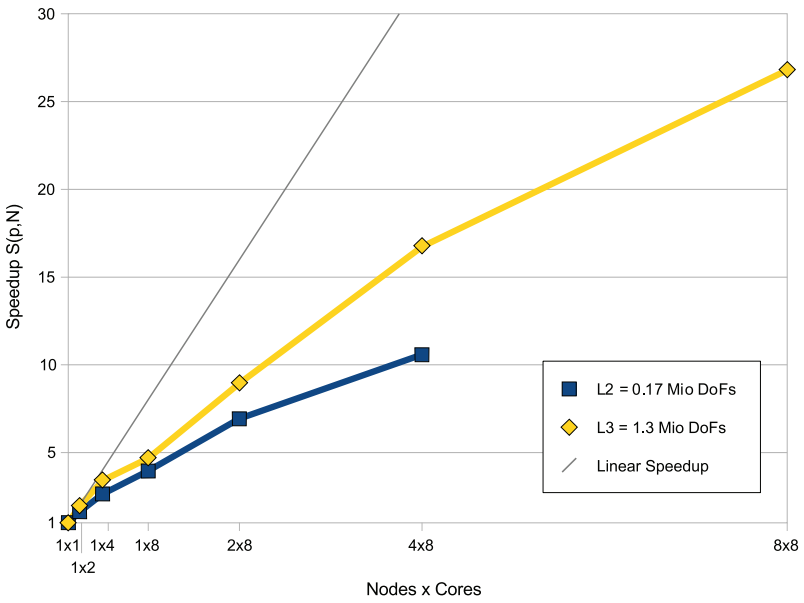


Figure 2: The speedup of the benchmark is plotted for levels of refinement 2 and 3. As reference point the sequential run is used.

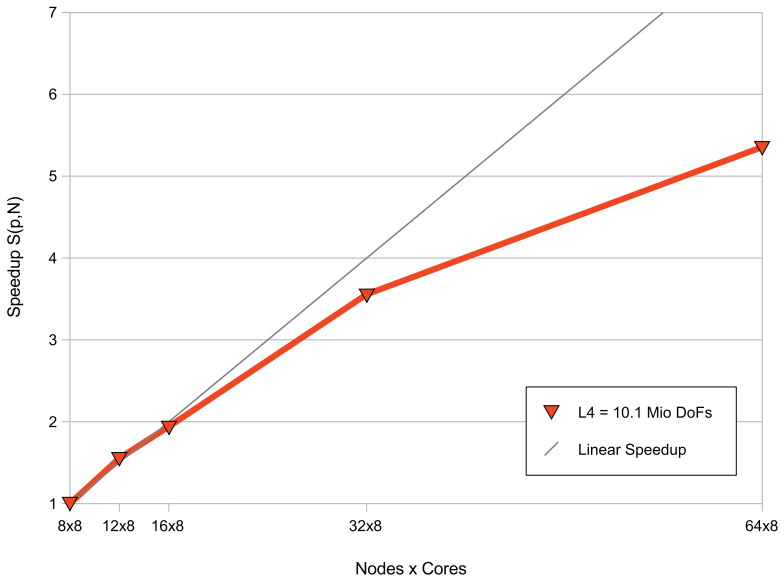


Figure 3: The speedup of the benchmark is plotted for the level 4 refinement. The test run with the lowest possible core number (64 cores) is used as reference point.

the linear systems, communication and synchronization is necessary in every iteration of the GMRES solver, since the systems are fully coupled. The computational domain is divided into the same number of subdomains as the number of processors used. Second, the incomplete LU factorization is applied as a block-Jacobi preconditioner. This means that the couplings from different subdomains are neglected in the preconditioning step. Therefore the preconditioner becomes less effective the more subdomains we have, since more information about the couplings between the unknowns is ignored. In consequence, the GMRES solver needs to perform more iterations to achieve the desired stopping tolerance, which can be seen by comparing the number of GMRES iterations in Table 2.

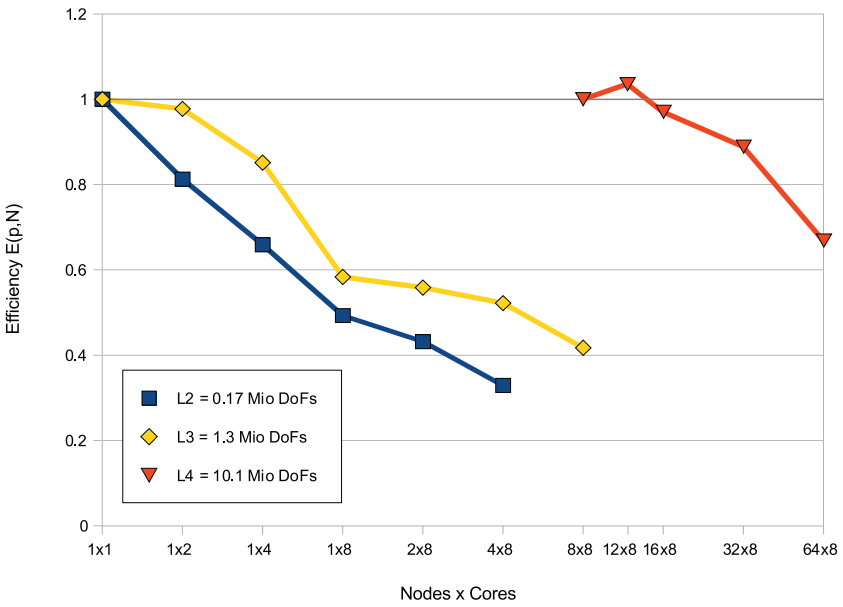


Figure 4: The efficiency of the benchmark for three levels of refinement. The efficiency of the test run with the lowest possible core number is set to 1, the other tests are based on these, respectively.

The problem sizes of the level 2 and 3 series are small enough to run the tests on one processor as a basis for the evaluation of the efficiency. The efficiency decreases to less than 50% for these two series when increasing the number of processors to 32 and 64,

respectively. In addition to the effects regarding the linear solver and preconditioner mentioned above, the computation to communication ratio decreases when using more processors, and therefore the level 2 series with the smallest problem size yields the lowest efficiency. For the level 3 series we observe a better efficiency due to its larger problem size.

The level 4 series shows a better efficiency than the other two series, but it should be kept in mind that this is measured with respect to a different reference, and therefore not directly comparable. For 96 processors, we even observe a superlinear speedup, which corresponds to the fact that fewer GMRES iterations are needed on 96 cores compared to 64 cores. When increasing the number of processors to 128 and 256, the efficiency remains at high values of 97% and 89%, respectively. For a decomposition onto 512 processors, the computation to communication ratio is lowered so that the efficiency decreases to 67%.

3.2 Weak Scalability

For the weak scalability, we want to keep the problem size per core constant. Hence, we increase the problem size, namely the number of DoFs N, and the number of cores p by the same factor s and compare the run-times $T(p, N)$:

$$\mathcal{R} := \frac{T(s \cdot p, s \cdot N)}{T(p, N)}.$$

To evaluate the weak scalability, we compare the run-times from Table 2 of runs where the number of cores are increased by a factor of eight, and the number of DoFs are increased by a factor of about 7.7, which we obtain naturally by the global mesh refinement.

For example, we can compare the run-time of $T(2 \times 8, L3) = T(16, 1.3 \cdot 10^6)$ and $T(1 \times 2, L2) = T(2, 0.17 \cdot 10^6)$ which gives $\mathcal{R} = 2.09$. Assuming a perfect scaling the run-time should remain constant, so that $\mathcal{R} = 1$. Results for selected comparisons from Table 2 are shown

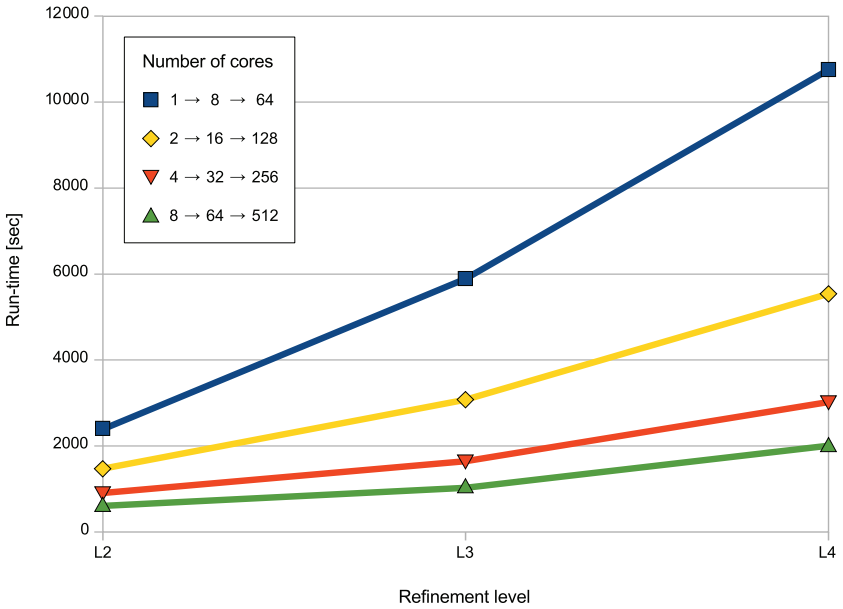


Figure 5: Comparison sequences of the run-times of jobs in order to obtain a measure for weak scalability, where the number of cores and DoFs are increased by a factor of 8 and 7.7, respectively.

in Fig. 5. Each curve stands for a sequence of run-times where the number of cores is multiplied by 8 from level to level. The number of cores for level 2 is taken from $\{1,2,4,8\}$, respectively, for each line. The run-times increase by factors varying from 1.70 to 2.46 when comparing level 2 and 3 and from 1.80 to 1.95 when comparing level 3 and level 4. However, as mentioned in Subsection 3.1, the workload is not completely determined by the number of unknowns, due to the increase in the number of GMRES iterations with an increased degree of parallelization.

3.3 Scalability Results

The obtained results for the strong and weak scalability show a decrease in speedup and efficiency as well as a non-constant behaviour of the ratio $\mathcal{R}$. This can be explained by two aspects. Firstly, the main influence results from the mathematical characteristics of the

block-Jacobi preconditioner as described in Subsection 3.1. Secondly, one cannot expect much better efficiency with a large number of processors for the small problem sizes of the level 2 and 3 test series. Due to memory limitations of the cluster, it was not possible to run the tests with a refinement level higher than level 4, i.e. number of DoFs $N = 10,098,432$.

4 Conclusion

In this paper we presented the mathematical setup of the scalability benchmark implemented with the HiFlow^3 Finite Element package. We executed different tests with varying numbers of cores and different problem sizes. Based on the run-times, the weak and strong scalability were analyzed. Our conclusion is that both the weak and the strong scalability are highly satisfactory and correspond to the expected behaviour of a mature parallel FEM software package. It turns out that the behaviour of the block-Jacobi preconditioner has a strong impact on the decrease of the scalability. Improvements with respect to more efficient preconditioning techniques are planned and will be included in a new release of HiFlow^3. An extension of this benchmark on larger machines considering larger problems will be the subject of further research.

References

[Girault1986] V. Girault and P.-A. Raviart. *Finite element methods for Navier-Stokes equations : theory and algorithms.* Springer series in computational mathematics. Springer, Berlin, 1986.

[Heuveline2010] H. Anzt, W. Augustin, M. Baumann, H. Bockelmann, T. Gengenbach, T. Hahn, V. Heuveline, E. Ketelaer, D. Lukarski, A. Otzen, S. Ritterbusch, B. Rocker, S. Ronnas, M. Schick, C. Subramanian, J.-P. Weiss, and F. Wilhelm. Hiflow3 – a flexible and hardware-aware par-

allel finite element package. EMCL Preprint Series, 2010. 06:1-36.

[Heuveline2011] H. Anzt, W. Augustin, M. Baumann, T. Gengenbach, T. Hahn, A. Helfrich-Schkarbanenko, V. Heuveline, E. Ketelaer, D. Lukarski, A. Nestler, S. Ritterbusch, S. Ronnås, M. Schick, M. Schmidtobreick, C. Subramanian, J.-P. Weiss, F. Wilhelm, and M. Wlotzka. Hiflow3 - a hardware-aware parallel finite element package. In *5th Parallel Tools Workshop*. Springer, accepted.

[John2002] V. John. Higher order finite element methods and multigrid solvers in a benchmark problem for the 3d navier-stokes equations. *International Journal for Numerical Methods in Fluids*, 40(6):775–798, 2002.

[Karypis2006] A. Abou-Rjeili and G. Karypis. Multilevel algorithms for partitioning power-law graphs. In *20th International Parallel and Distributed Processing Symposium, 2006. IPDPS 2006*.

[Mayer2007] J. Mayer. A multilevel crout ilu preconditioner with pivoting and row permutation. *Numerical Linear Algebra with Applications*, 14(10):771–789, 2007.

[Mayer2008] J. Mayer. Symmetric permutations for i-matrices to delay and avoid small pivots during factorization. *SIAM J. Sci. Comput.*, 30(2):982–996, March 2008.

[Ronnas2011] S. Ronnås, T. Gengenbach, E. Ketelaer, and V. Heuveline. Design and implementation of distributed meshes in hiflow3. In C. Bischof, H.-G. Hegering, W. E. Nagel, and G. Wittum, editors, *Competence in High Performance Computing 2010*, pages 61–71. Springer Berlin Heidelberg, 2012.

[Saad2003] Y. Saad. *Iterative methods for sparse linear systems.* SIAM, Society for Industrial and Applied Mathematics, Philadelphia, 2nd edition, 2003.

[Turek1996] S. Turek and M. Schäfer. 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*, pages 547–566. Vieweg, 1996. co. F. Durst, E. Krause, R. Rannacher.

Acknowledgments

We thank Mehmet Soysal for his constant technical support as administrator of the bwGRiD cluster in Karlsruhe. Further the authors thank Prof. Dr. Rudolf Lohner and Dr. Marcel Kunze for their kind support in the preparation of this paper.

Preprint Series of the Engineering Mathematics and Computing Lab

recent issues

No. 2012-04 Hartwig Anzt, Armen Beglarian, Suren Chilingaryan, Andrew Ferrone, Vincent Heuveline, Andreas Kopmann: A unified Energy Footprint for Simulation Software

No. 2012-03 Vincent Heuveline, Chandramowli Subramanian: The Coffee-table Book of Pseudospectra

No. 2012-02 Dominik P.J. Barz, Hendryk Bockelmann, Vincent Heuveline: Electrokinetic optimization of a micromixer for lab-on-chip applications

No. 2012-01 Sven Janko, Björn Rocker, Martin Schindewolf, Vincent Heuveline, Wolfgang Karl: Software Transactional Memory, OpenMP and Pthread implementations of the Conjugate Gradients Method - a Preliminary Evaluation

No. 2011-17 Hartwig Anzt, Jack Dongarra, Vincent Heuveline, Piotr Luszczek: GPU-Accelerated Asynchronous Error Correction for Mixed Precision Iterative Refinement

No. 2011-16 Vincent Heuveline, Sebastian Ritterbusch, Staffan Ronnås: Augmented Reality for Urban Simulation Visualization

No. 2011-15 Hartwig Anzt, Jack Dongarra, Mark Gates, Stanimire Tomov: Block-asynchronous multigrid smoothers for GPU-accelerated systems

No. 2011-14 Hartwig Anzt, Jack Dongarra, Vincent Heuveline, Stanimire Tomov: A Block-Asynchronous Relaxation Method for Graphics Processing Units

No. 2011-13 Vincent Heuveline, Wolfgang Karl, Fabian Nowak, Mareike Schmidtobreick, Florian Wilhelm: Employing a High-Level Language for Porting Numerical Applications to Reconfigurable Hardware

No. 2011-12 Vincent Heuveline, Gudrun Thäter: Proceedings of the 4th EMCL-Workshop Numerical Simulation, Optimization and High Performance Computing

No. 2011-11 Thomas Gengenbach, Vincent Heuveline, Mathias J. Krause: Numerical Simulation of the Human Lung: A Two–scale Approach

No. 2011-10 Vincent Heuveline, Dimitar Lukarski, Fabian Oboril, Mehdi B. Tahoori, Jan-Philipp Weiss: Numerical Defect Correction as an Algorithm-Based Fault Tolerance Technique for Iterative Solvers

No. 2011-09 Vincent Heuveline, Dimitar Lukarski, Nico Trost, Jan-Philipp Weiss: Parallel Smoothers for Matrix-based Multigrid Methods on Unstructured Meshes Using Multicore CPUs and GPUs

No. 2011-08 Vincent Heuveline, Dimitar Lukarski, Jan-Philipp Weiss: Enhanced Parallel ILU(p)-based Preconditioners for Multi-core CPUs and GPUs – The Power(q)-pattern Method

No. 2011-07 Thomas Gengenbach, Vincent Heuveline, Rolf Mayer, Mathias J. Krause, Simon Zimny: A Preprocessing Approach for Innovative Patient-specific Intranasal Flow Simulations

The responsibility for the contents of the working papers rests with the authors, not the Institute. Since working papers are of a preliminary nature, it may be useful to contact the authors of a particular working paper about results or caveats before referring to, or quoting, a paper. Any comments on working papers should be sent directly to the authors.