Cache-Oblivious Solution of the 2D Navier-Stokes Equations

Tobias Neckel
neckel@in.tum.de
Institut für Informatik / Technische Universität München / Scientific Computing in Computer Science
Boltzmannstraße 3, 85748 Garching

Christoph Zenger
zenger@in.tum.de
Institut für Informatik / Technische Universität München / Scientific Computing in Computer Science
Boltzmannstraße 3, 85748 Garching

Abstract

Numerical solvers in the field of partial differential equations are often faced with a considerable loss of efficiency in data access and storage usage due to jumps within the physical memory space of the computer. To avoid this drawback, the cache hierarchy as provided by many modern computer architectures should be used.

The purpose of this article is to provide the reader with an overview on a cache-efficient implementation of a Finite Element Method (FEM) for the time-dependent incompressible Navier-Stokes equations for isothermal, laminar fluid flow in two dimensions. By using space-filling curves on hierarchically structured cartesian grids and a stack storage system, very high Level 2 cache hit rates can directly be obtained.

The approach seems to be perfectly suited for adaptive cartesian grids, demanding only a minimum information for adaptive refinement. Besides, an extension to three dimensions will be straight forward. Thus, exploiting the full potential of this method promises to be very interesting.

1 Introduction

When using the hierarchical memory system of caches that are smaller than the main memory (RAM) but have less latencies, a processor obtains copies of reused data much faster than from the RAM.

To exploit this storage hierarchy efficiently, an algorithm has to respect locality concerning time and physical memory. Temporal locality means that if data copied to the cache has to be reused, this should be done as fast as possible in order to prevent that it is deleted from the cache by following data. The spatial locality assures that a new data packet which is going to be used right after the currently read packet is located next to it in the physical memory. Because the cache holds a copy of the RAM that has the size of a whole cache line, the new data is likely to be in the cache, too. Thus, the spatial locality reduces the number of cache misses (i.e. number of access to data that is not in the cache).

The incompressible 2D-Navier-Stokes equations have the following form (see [2], e.g.)

$$\begin{aligned} \nabla \cdot \mathbf{u}(x,t) &= 0 \qquad \forall\,(x,t) \in \Omega \times [0,T] && (1)\\ \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\,\mathbf{u} + \frac{1}{\rho}\nabla p - \nu\,\Delta\mathbf{u} &= \mathbf{g} \qquad \forall (x,t) \in \Omega \times [0,T], && (2)\end{aligned}$$

where $\mathbf{u}(x,t)$ is the velocity vector and $p(x,t)$ the scalar pressure on a space point x in the domain Ω at a specific time $t \in [0,T]$. The nabla operator $\nabla_i = \frac{\partial}{\partial x_i}$ represents the gradient or the divergence, resp., whereas Δ denotes the Laplace operator. The fluid properties are given by the density ρ and the kinematic viscosity ν.

The boundary conditions on $\partial\Omega = \Gamma_D \cup \Gamma_N$,

$$\begin{aligned} \mathbf{u}(x,t) &= \mathbf{w}(x,t) \qquad \forall\,(x,t) \in \Gamma_D \times (0,T] && (3)\\ [\mu\,\nabla\mathbf{u}\cdot\mathbf{n} - p\,\mathbf{n}]\,(x,t) &= \mathbf{f}(x,t) \qquad \forall\,(x,t) \in \Gamma_N \times (0,T], && (4)\end{aligned}$$

define Dirichlet and Neumann conditions for the velocity.

Additionally, we need the initial conditions

$$\begin{aligned} \nabla \cdot \mathbf{u}(x, t_0 = 0) &= 0 \qquad \forall\, x \in \Omega && (5)\\ \mathbf{n}\cdot\mathbf{u}(x, t_0 = 0) &= \mathbf{n}\cdot\mathbf{w}_0 \qquad \forall\, x \in \Gamma_D. && (6)\end{aligned}$$

In our simulations, we do not consider body forces ($\mathbf{g} = 0$ in (2)) and use the outflow conditions $\mathbf{f} = \mathbf{0}$ in (4).

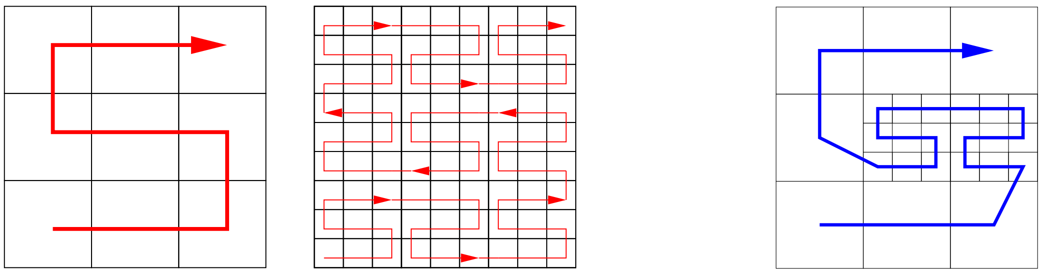
2 Cache-oblivious FEM implementation

In this section, we present a method for solving the Navier-Stokes equations based on space-filling curves. First, an introduction to the concept of the cache-efficient implementation elaborated in [3] is given. Then we describe the Finite Element Method used for the spatial discretisation as well as the time integration scheme of this method (developed in [10], [5]).

2.1 Concept of the cache-oblivious implementation

One possibility to achieve the temporal and spatial locality presented in the previous section is the usage of space-filling curves with suitable data structures and a hierarchical tree arrangement of the grid cells.

The space-filling curve that is used to traverse and linearise the grid cells of the domain in a depth-first strategy is the Peano-curve. One step of regular refinement following the pattern introduced by the Peano-curve in two dimensions is shown in Fig. 1(a). The refinement always subdivides one father cell into nine son cells. The recursive definition of the curve is well suited for adaptive grids (see Fig. 1(b)) and for recursive programming structures to visit all cells of the domain. The property of locality of the Peano-curve assures the temporal locality of the concept by returning quickly to already used corner nodes that hold the degrees of freedom of the velocities (see Fig. 2(a)).



(a) (b)

Figure 1: A regularly (a) and adaptively (b) refined Peano-curve on a square domain.

The data access is organised strictly locally. Thus, the program only needs to know the data of one cell put together completely in data packets. The packets of the whole hierarchical cell arrangement are stored in a system of linear stacks where only two operations are allowed: *push* (put current data packet on top of the stack) and *pop* (get top packet from the stack). By implementing the stacks as arrays, this data usage assures the spatial locality. For regular grids, two stacks would be sufficient to distinguish nodes left and right of the Peano curve. Because of adaptivity and hierarchical demands, $3^d + 1 = 10$ stacks are needed for the grid corners (see [3]).

Even for complex geometries no redundant operations (costs) arise due to the underlying concept of refinement which is very similar to the one of quadtrees and octrees (see [4], e.g.).

2.2 FEM for the Navier-Stokes equations

For solving the Navier-Stokes equations numerically, we have to discretise the equations in space and time. We use a Finite Element Method on the velocities for the spatial discretisation. The corresponding weak form of the momentum equations (2)

$$\begin{aligned} \int_\Omega \dot{\mathbf{u}} \cdot \mathbf{s}\, dx \;+\; \int_\Omega (\mathbf{u} \cdot \nabla)\, \mathbf{u} \cdot \mathbf{s}\, dx - \frac{1}{\rho} \int_\Omega p\, (\nabla \cdot \mathbf{s})\, dx \;+\; \nu \int_\Omega \nabla \mathbf{u} : \nabla \mathbf{s}\, dx \;=& \\ = \int_\Omega \mathbf{g} \cdot \mathbf{s}\, dx \;+\; \int_{\Gamma_N} \mathbf{f} \cdot \mathbf{s}\, da. & \qquad (7) \end{aligned}$$

uses the Neumann boundary condition (4) and ansatz and test functions $\mathbf{u}$ and $\mathbf{s}$ in the spaces

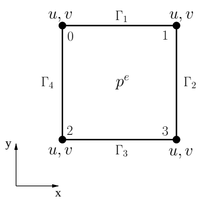
$$\begin{aligned} U &= \left\{ \mathbf{u} \in H^{1,2}(\Omega) \mid \quad \mathbf{u}|_{\Gamma_D} = \mathbf{w} \right\} \\ S &= \left\{ \mathbf{s} \in H^{1,2}(\Omega) \mid \quad \mathbf{s}|_{\Gamma_D} = 0 \right\}. \end{aligned}$$

$H^{1,2}(\Omega)$ denotes the Sobolev space $H^1(\Omega)$ for two dimensional functions.

In standard Galerkin manner, we project the infinite spaces U and S to finite subspaces U_h and S_h, resp. By using bilinear basis functions $\Phi_i(\mathbf{x})$ on our regular cartesian grid nodes to represent $\mathbf{u}_h(\mathbf{x})$ and $\mathbf{s}_h(\mathbf{x})$, we get a time dependent system of equations from (7)

$$\begin{aligned} A\dot{u}_h + Du_h + C(u_h, v_h)u_h - M_x^T p_h/\rho &= f_x && (8)\\ A\dot{v}_h + Dv_h + C(u_h, v_h)v_h - M_y^T p_h/\rho &= f_y && (9)\\ M_x u_h + M_y v_h &= 0, && (10)\end{aligned}$$

where the x and y components of the semi-discrete momentum equations are represented separately. The global matrices A (mass matrix), D (diffusion), $C(u_h, v_h)$ (convection) and M_x, M_y (divergence) can be assembled by their local counterparts (see Fig. 2(b)).



$$\left.\begin{aligned} 0 &= \int_{\Omega^e} \nabla\cdot\mathbf{u}_h\,dx = \int_{\Gamma^e} \mathbf{u}_h\cdot\mathbf{n}\,da\\ &= \int_{\Gamma_1} v_h da + \int_{\Gamma_2} u_h da - \int_{\Gamma_3} v_h da - \int_{\Gamma_4} u_h da\\ &= \tfrac{h}{2}\left[v_0 + v_1 + u_1 + u_3 - v_2 - v_3 - u_0 - u_2\right]\end{aligned}\right\} \Rightarrow \begin{aligned} M_x^e &= \tfrac{h}{2}(-1,1,-1,1)\\ M_y^e &= \tfrac{h}{2}(1,1,-1,-1)\end{aligned}$$

(a) (b)

Figure 2: Visualisation of one grid cell and the corresponding degrees of freedom (a). Derivation of the local discrete divergence operators M_x^e, M_y^e (b).

The pressure p is not treated by FEM but has to assure the continuity equation (1) on the discrete cell level as a Lagrange multiplier of this omnipotent constraint. With constant values p^e on each cell, the local gradient operators $M_x^{e\,T}, M_y^{e\,T}$ from $-p^e \int_{\Omega^e} (\nabla\cdot\mathbf{s}_h)\,dx$ of (7) go with the local divergence operators M_x^e, M_y^e derived in Fig. 2(b).

Together with the semi-discrete continuity equation (10), differentiated once with respect to time, the usage of equations (8) and (9) leads to one linear system of equations for the discrete pressure (see [2])

$$\begin{aligned}\left(M_x A^{-1} M_x^T + M_y A^{-1} M_y^T\right) p_h &= M_x A^{-1}\left[-f_x + Du_h + C(u_h, v_h)u_h\right]\\ &\quad + M_y A^{-1}\left[-f_y + Dv_h + C(u_h, v_h)v_h\right]. \quad (11)\end{aligned}$$

This system can be interpreted as a discrete pressure Poisson equation and allows the calculation of the pressure at any time if velocity values are known.

As for the time integration, a simple explicit Euler method can be used to approximate the velocity time derivative $\dot{\mathbf{u}}_h$ by $(\mathbf{u}_h^{n+1} - \mathbf{u}_h^n)/\Delta t$. This allows to calculate the new velocity values $\mathbf{u}_h^{n+1}$ by a simple update using the pressure gradient in each time step like in the Chorin projection method.

Although the linear system of equations is globally presented in (11), a SOR-solver is used that works strictly locally on each grid cell. By that means, large memory needs and complex data structures for the sparse system matrix can be avoided and a cell-wise view compatible to the cache concept can be obtained.

3 Results

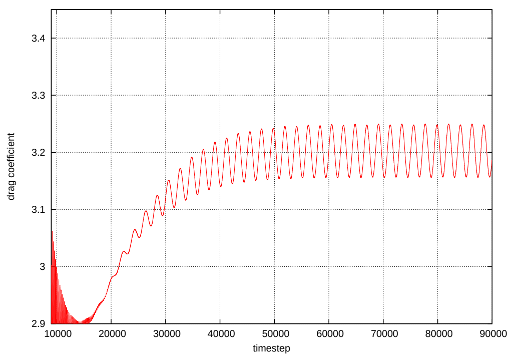
In this section, the results of the DFG-benchmark scenario [8] for the channel flow around a cylinder at a Reynolds number of 100 are presented, resulting in the typical Kármán vortex street. More results of further scenarios are given in [5].

Two different levels of regular refinement are used for the simulations: level 1 with 4374 cells and level 2 with 39366 cells for the channel representing 13502 and 119070 degrees of freedom respectively. We restrict to these levels due to our explicit time scheme that causes a very large number of time steps for a further refinement of the grid. The simulations have been carried out on an Intel XEON 2.4 GHz architecture with a Level 2 cache size of 512 KB. Concerning the memory requirement (see Tab. 1), the program has not been optimised yet. The L2 cache hit rate shown in Tab. 1 has been measured with the tool perfex [6] which exploits the performance counters of the CPU. The very high hit rates of more than 98% confirm the efficiency of the implementation concept. Standard Gauß-Seidel smoothers for multigrid methods only reach L2 hit rates less 55% as shown in [1].

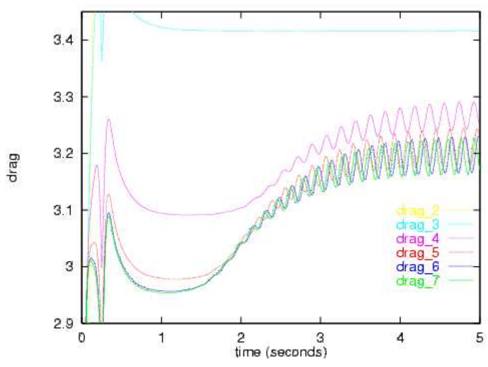
Table 1: Flow around a cylinder at Re = 100: survey of reference values and measured data.

	$c_{d,max}$	$c_{l,max}$	St	Δp	L2 cache hit rate	MEM (MB)
level 1	2.63	0.49	0.298	1.87	98.48%	2.7
level 2	3.25	1.05	0.297	2.23	98.13%	18
Schäfer et al. [8]	3.23	1.00	0.299	2.48		

The numerical results are satisfying for the regular discretisation of the domain and the number of degrees of freedom. The first three reference values of level 2 in Tab. 1 – maximum drag coefficient $c_{d,max}$, maximum lift coefficient $c_{l,max}$ and the Strouhal number St – match those in [8]. Due to the cell centered location of the pressure, an extrapolation would further improve the difference Δp between front and back side of the cylinder.



(a)



(b)

Figure 3: Results for the drag coefficient of level 2 (a) and of [9] (b).

A comparison plot of the drag coefficient with respect to time is given in Fig. 3. The mean value of 3.2 for level 2 corresponds well to the one of [9] on level 6 (667264 d.o.f.'s).

4 Conclusion

In this paper, we have presented a cache-efficient FEM implementation for the incompressible 2D-Navier-Stokes equations. It has been shown that a very high L2 cache hit rate can directly be obtained without any postprocessing optimisation of the code.

As we are still at the beginning of our work, further research is necessary to evaluate the whole spectrum of benefits of our approach. However, current results and first experiences of [7] seem to be promising with respect to developing an efficient three dimensional Navier-Stokes-solver for adaptive cartesian grids. This could be a possibility to address complex problems like fluid structure interaction or direct numerical simulation of turbulent scenarios.

References

[1] *Douglas, C.C., Hu, J., Kowarschik, M., Rüde, U., Weiß, C.:* Cache Optimization for Structured and Unstructured Grid Multigrid, Electronic Transactions on Numerical Analysis, 10:21-40, 2000.

[2] *Gresho, P. M., Sani, R. L.:* Incompressible flow and the finite element method, Chichester, John Wiley & Sons Ltd, 1998.

[3] *Günther, F.:* Eine cache-optimale Implementierung der Finite-Elemente-Methode, Doctoral Thesis, Institut für Informatik, TU München, 2004.

[4] *Mundani, R., Bungartz, H.-J., Rank, E., Romberg, R., Niggl, E.:* Efficient Algorithms for Octree-Based Geometric Modelling, Proceedings of the Ninth International Conference on Civil and Structural Engineering Computing, Civil-Comp Press, 2003.

[5] *Neckel, T.:* Einfache 2D-Fluid-Struktur-Wechselwirkungen mit einer cache-optimalen Finite-Elemente-Methode, Master Thesis, Institut für Informatik, TU München, 2005.

[6] *Perfex:* Linux perfex, http://www.complang.tuwien.ac.at/anton/linux-perfex/, 2000.

[7] *Pögl, M.:* Entwicklung eines cache-optimalen 3D Finite-Element-Verfahrens für große Probleme, Doctoral Thesis, Institut für Informatik, TU München, 2004.

[8] *Schäfer, M., Turek, S.:* Benchmark Computations of Laminar Flow around a Cylinder, http://www.featflow.de/bench_all/paper.html, 1997.

[9] *Turek, S.:* The ‘DFG-Benchmark 1995/6’: Channel Flow around a Circle at Re=100, http://www.featflow.de/album/catalog/bench_cyl_2d/data.html, 1998.

[10] *Weinzierl, T.:* Eine cache-optimale Implementierung eines Navier-Stokes-Lösers unter besonderer Berücksichtigung physikalischer Erhaltungssätze, Master Thesis, Institut für Informatik, TU München, 2005.