Towards a multi-GPU solver for the three-dimensional two-phase incompressible Navier-Stokes equations

Michael Griebel, *Peter Zaspel*

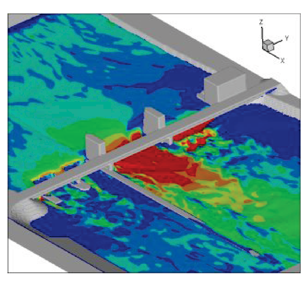
universitätbonn — Institute for Numerical Simulation, Rheinische Friedrich-Wilhelms-Universität Bonn

Project

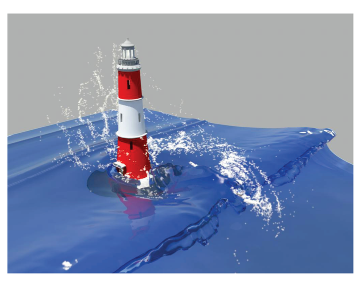
We are in the process of porting our parallel level-set based two-phase solver for the three-dimensional Navier-Stokes equations to the GPU.

NaSt3DGPF – A parallel two-phase solver for computational fluid dynamics

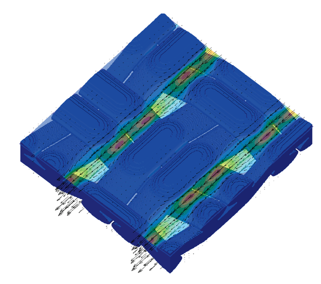
NaSt3DGPF has been developed at the Institute for Numerical Simulation [1],[2],[3]. These are some of its applications:



Flow through and around hydraulic constructions



High quality fluid animations



Flow through porous media

It has the following features:

- Finite difference discretization of the Navier-Stokes equations on a uniform staggered grid using Chorin's projection approach
- Simulation of two-phase flows (e.g. air and water) by the well-known *level set method*
- Computation of surface tension effects (via continuum surface force)
- Simulation of turbulence by a large-eddy approach
- Parallelization using the domain decomposition approach of Schwarz (implemented via MPI)

Current progress

A multi-GPU double-precision solver for the pressure Poisson equation based on the Jacobi preconditioned conjugate gradient method has been implemented using CUDA and MPI.

Model equations and their solution

The two-phase Navier-Stokes equations

The **Navier-Stokes equations** serve as model for fluid behavior.

$$\frac{\delta}{\delta t}\vec{u} + (\vec{u}\cdot\nabla)\vec{u} + \frac{1}{\rho}\nabla p = \frac{\mu}{\rho}\Delta\vec{u} + \vec{g} \quad \text{(momentum equation)}$$

$$\nabla\cdot\vec{u} = 0 \quad \text{(continuity equation)}$$

- computational domain $\Omega \subset \mathbb{R}^3$
- time $t \in [0, t_{end}]$
- velocity $\vec{u} \in \Omega \times [0, t_{end}] \to \mathbb{R}^3$
- pressure $p \in \Omega \times [0, t_{end}] \to \mathbb{R}$
- dynamic viscosity $\mu \in \Omega \times [0, t_{end}] \to \mathbb{R}$
- density $\rho \in \Omega \times [0, t_{end}] \to \mathbb{R}$
- constant volume force $\vec{g} \in \mathbb{R}^3$
- initial conditions $\vec{u}(\cdot, 0) = \vec{u}_0(\cdot)$, $p(\cdot, 0) = p_0(\cdot)$
- boundary conditions

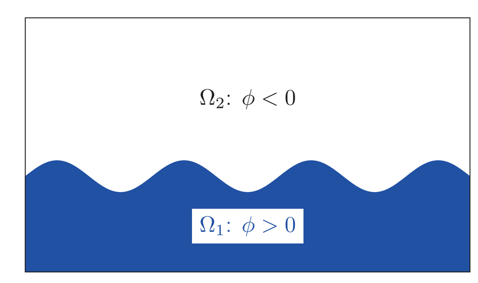
Two phases (e.g. air and water) are distinguished by a **level set function** ϕ.

Definition:

$$\phi : \Omega \times [0, t_{end}] \to \mathbb{R}$$
$$|\nabla\phi| = 1$$

Free surface / water surface:

$$\Gamma_f(t) = \{\vec{x} \in \Omega |\, \phi(\vec{x}, t) = 0\}$$



$$\Omega = \Omega_1 \cup \Omega_2 \cup \Gamma_f$$

The domain-dependent density and viscosity is thus given by

$$\rho(\phi) = \rho_2 + (\rho_1 - \rho_2)\, H(\phi) \qquad \mu(\phi) = \mu_2 + (\mu_1 - \mu_2)\, H(\phi) \quad \text{with} \quad H = (\phi)\begin{cases} 0 & \text{if } \phi < 0 \\ \frac{1}{2} & \text{if } \phi = 0 \\ 1 & \text{if } \phi > 0 \end{cases}.$$

Chorin's projection approach

For each time step n compute a new time step $n+1$ by

1. calculating an intermediate velocity field $\vec{u}^*$
$$\frac{\vec{u}^* - \vec{u}^n}{\delta t} = -(\vec{u}^n\cdot\nabla)\vec{u}^n + \frac{\mu}{\rho}\Delta\vec{u}^n + \vec{g},$$
2. solving the Poisson equation for the pressure p^{n+1}
$$\nabla\cdot\left(\frac{1}{\rho}\nabla p^{n+1}\right) = \nabla\cdot\frac{\vec{u}^*}{\delta t},$$
3. applying a pressure correction to $\vec{u}^*$
$$\vec{u}^{n+1} = \vec{u}^* - \frac{\delta t}{\rho}\nabla p.$$

The solution of the Poisson equation dominates the overall simulation time

- Discretization of the pressure Poisson equation results in a very large, but sparse linear system:
$$Ax = b \quad \text{with} \quad \#\text{unknowns} \equiv \#\text{grid points}$$
- Iterative solvers for systems of linear equations take full advantage of sparsity
  ⇒ iterative linear solvers are used to solve the pressure Poisson equation
- Non-constant density ρ in a two-phase fluid simulation leads to large matrix condition number
  ⇒ a conjugate gradient solver with Jacobi preconditioner is necessary

Implementation

The preconditioned conjugate gradient solver for linear systems

The linear system $Ax = b$ is solved iteratively to a given threshold ϵ by the following algorithm:

Algorithm ConjugateGradient(A, b, ϵ)

$init\ x_0 \in \mathbb{R}^n$
$r_0 = b - Ax_0$
$q_0 = B^{-1} r_0$
$p_0 = q_0$
for $k = 0, 1, \ldots$
do
$a_k = \frac{r_k^T q_k}{p_k^T A p_k}$
$x_{k+1} = x_k + a_k p_k$
$r_{k+1} = r_k - a_k A p_k$
if $(r_{k+1} < \epsilon)$
then return (x_{k+1})
$q_{k+1} = B^{-1} r_{k+1}$
$b_k = \frac{r_{k+1}^T q_{k+1}}{r_k^T q_k}$
$p_{k+1} = q_{k+1} + b_k p_k$

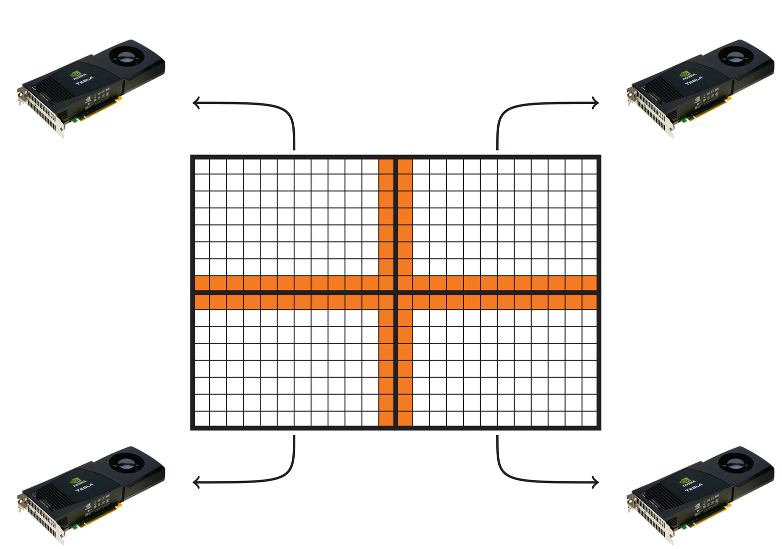
Matrix B is given by

$$B_{ij} = \begin{cases} A_{ii} & \text{if } i = j \\ 0 & \text{else} \end{cases}$$

for a *Jacobi* preconditioned conjugate gradient solver.

Multi-GPU parallelized conjugate gradient solver

Multi-GPU parallelization is performed according to the domain decomposition method of Schwarz. → Each GPU holds one part of the domain Ω.



Data exchange necessary at boundary cells for matrix-vector and inner products.

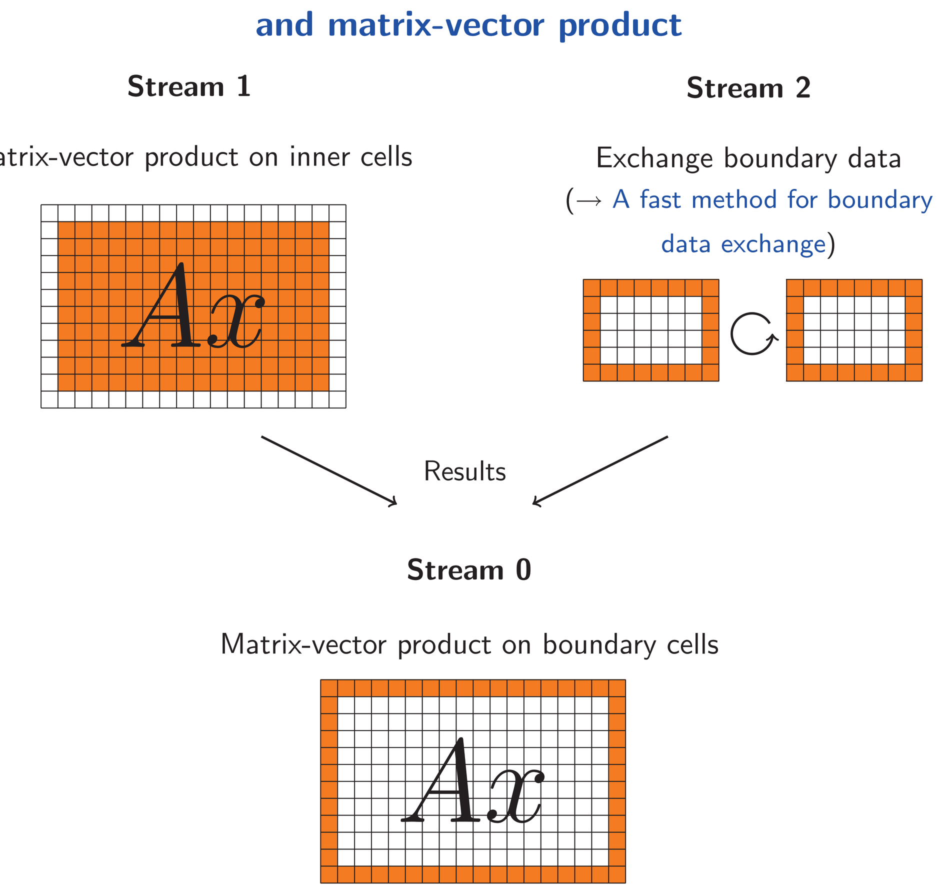
Main challenge:

Hiding time needed for data transfers by a well-chosen parallelization.

Our solution:

CUDA Streams parallelize data transfers and expensive matrix-vector products.[1]

Algorithm for combined data transfer and matrix-vector product



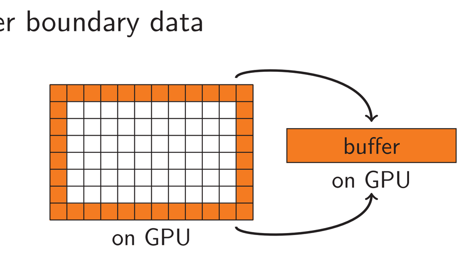
The usage of MPI allows our multi-GPU solver to scale on large distributed memory clusters

[1]available on devices with "concurrent copy and execution" capability

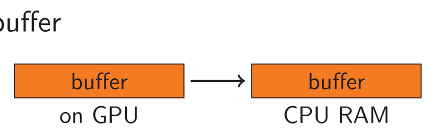
A fast method for boundary data exchange

The fast boundary data exchange is performed using a buffer:

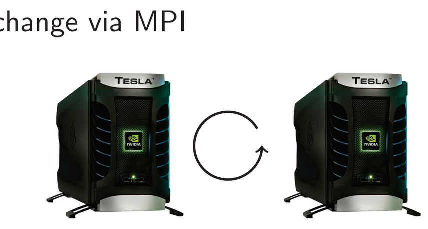
1. Transfer boundary data



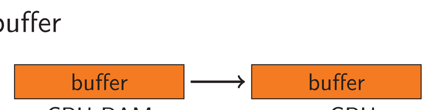
2. Transfer buffer



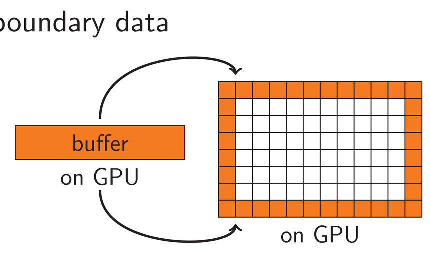
3. Data exchange via MPI



4. Transfer buffer



5. Transfer boundary data



Performance measurements

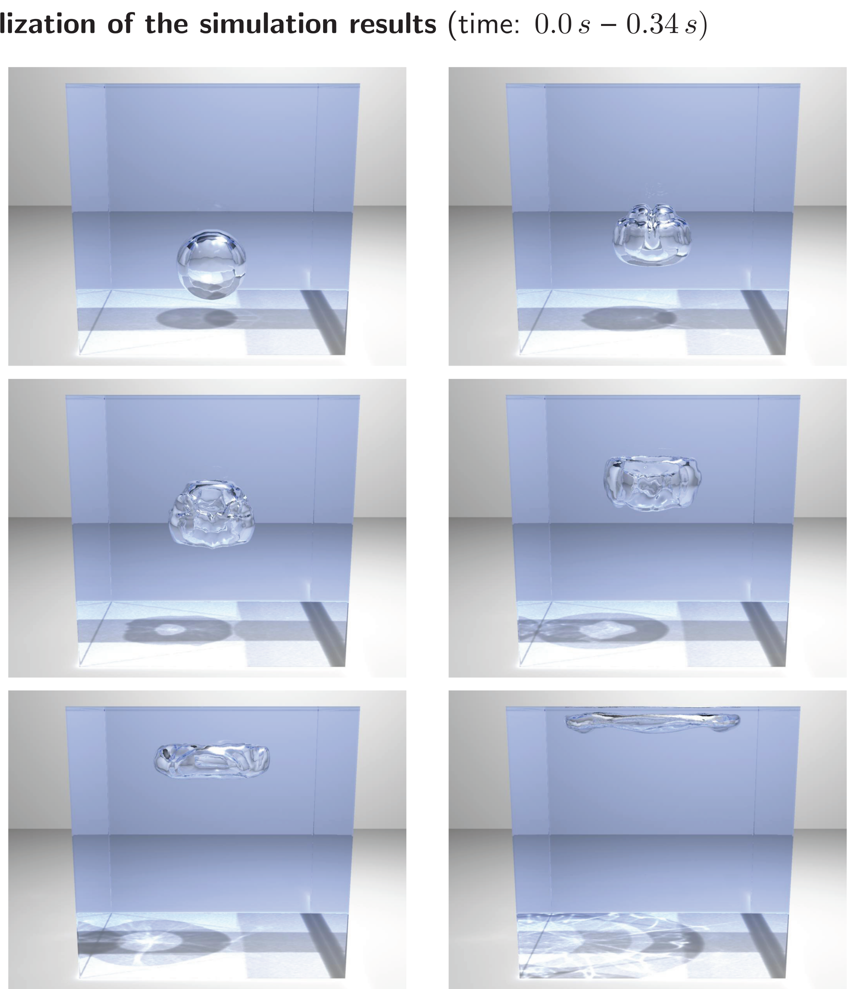
Benchmarking problem

A large air bubble rising in water

Simulation properties

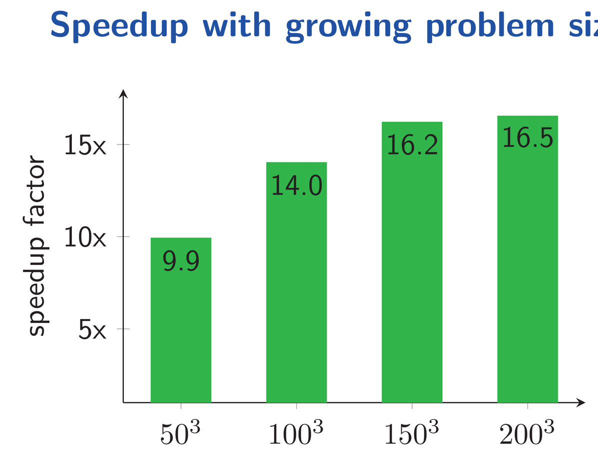
- Water box size: $0.2\,cm \times 0.2\,cm \times 0.2\,cm$ $(0.2\,cm \approx 0.58\,in)$
- Bubble radius: $0.03\,cm$ $(\approx 0.087\,in)$
- Volume forces: gravity $\vec{g} = (0, -9.81, 0)^t\,\frac{m}{s^2}$

Visualization of the simulation results (time: $0.0\,s - 0.34\,s$)



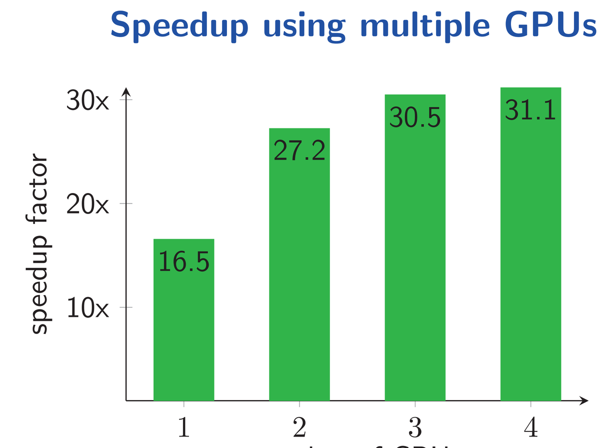
Results for the conjugate gradient solver

Speedup with growing problem sizes



(speedup: 1 CPU ↔ 1 GPU)

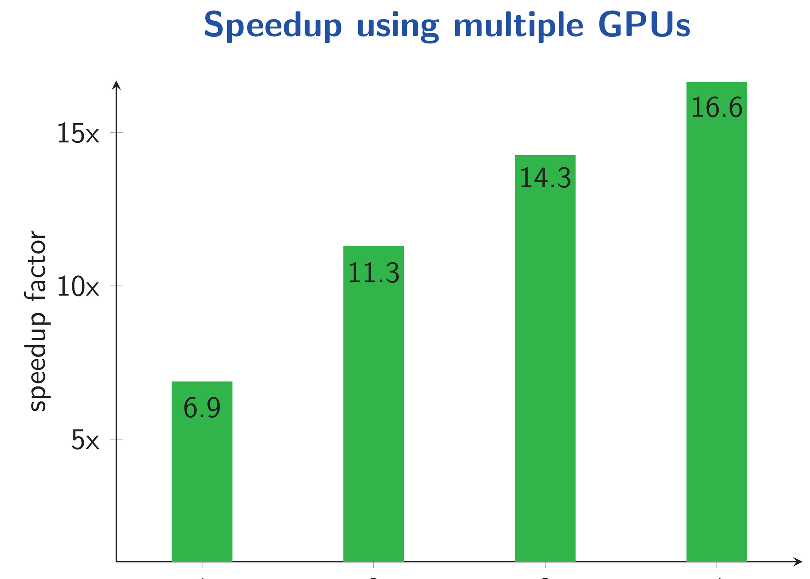
Speedup using multiple GPUs



(speedup: compared to one CPU, grid resolution: 200^3)

Results for the whole fluid solver

Speedup using multiple GPUs



(speedup: compared to one CPU, grid resolution: 200^3)

Benchmarking platform

NVIDIA Tesla S1070 connected to two workstations with Intel Core 2 Duo CPU (E8500/E7200) communicating over gigabit Ethernet

Time measurements

- *gettimeofday* command used
- during first 80 time steps
- 1000 CG iterations per time step
- including time necessary for data transfers

Outlook

Porting further parts of the solver

- Currently: Port of the time consuming level set reinitialization
- Later: Port of the advection scheme to get optimal overall speedup
- Finally: Every computation ported to the GPU

Improvements in scalability

- Investigation of the impact of different network connections on scalability
- Benchmarking different host systems to get optimal GPU speed
- Performance measurements on larger distributed memory clusters

References

[1] Croce, R., M. Engel, J. Strybny and C. Thorenz: *A Parallel 3D Free Surface Navier-Stokes Solver For High Performance Computing at the German Waterways Administration.* In *The 7th Int. Conf. on Hydroscience and Engineering (ICHE-2006)*, Philadelphia, USA, September 2006.

[2] Croce, R., M. Griebel and M. A. Schweitzer: *A Parallel Level-Set Approach for Two-Phase Flow Problems with Surface Tension in Three Space Dimensions.* Preprint 157, Sonderforschungsbereich 611, University of Bonn, 2004.

[3] Croce, R., M. Griebel and M. A. Schweitzer: *Numerical Simulation of Droplet-Deformation by a Level Set Approach with Surface Tension.* Preprint 395, Sonderforschungsbereich 611, University of Bonn, Bonn, Germany, 2008.

NaSt3DGPF project page: http://www.nast3dgpf.com

Peter Zaspel (Institute for Numerical Simulation - University of Bonn)
Email: zaspel@ins.uni-bonn.de