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

Michael Griebel, Peter Zaspel

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 porous media
• High quality fluid animation
• 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:

\[ \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} = -\nabla p + \frac{1}{\rho} \nabla \cdot \mathbf{f} + \mathbf{b} \]

(continuity equation)

\[ \nabla \cdot \mathbf{u} = 0 \]

(momentum equation)

- Computational domain \( \Omega \subset \mathbb{R}^3 \)
- Time \( t \in [0, T] \)
- Velocity \( \mathbf{u} \in \mathbb{R}^3 \)
- Density \( \rho \in \mathbb{R}^+ \)
- Dynamic viscosity \( \mu \in \mathbb{R}^+ \)
- Boundary conditions

Two phases (e.g. air and water) are distinguished by a level set function \( \phi \):

\[ \phi(\mathbf{x}, t) \]

The domain-dependent density and viscosity is then given by

\[ \rho(\mathbf{x}, t) = \rho_1 + (\rho_2 - \rho_1) \phi(\mathbf{x}, t) \]

with \( \rho_1, \rho_2 \in \mathbb{R}^+ \).

Chorin’s projection approach

For each time step \( n \):

1. Compute intermediate velocity \( \mathbf{u}_0 \) by:

\[ \frac{\mathbf{u}_0 - \mathbf{u}_{n-1}}{\Delta t} = \nabla \times \left( \frac{1}{\rho} \nabla p \right) + \mathbf{b} \]

2. Solve the Poisson equation for the pressure \( p_n \) using conjugate gradient

\[ -\nabla^2 p_n + \frac{1}{\rho_n} \nabla \phi = -\nabla \cdot \mathbf{u}_0 \]

3. Applying a pressure correction to \( \mathbf{u}_0 \):

\[ \mathbf{u}_n = \mathbf{u}_0 + \Delta t \nabla \phi \]

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
- 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 \( \rho \) in a two-phase fluid simulation leads to large matrix condition number \( \Rightarrow \) a conjugate gradient solver with Jacobi preconditioner is necessary

Implementation

The preconditioned conjugate gradient solver for linear systems

The linear system \( A x = b \) is solved iteratively to a given threshold \( \epsilon \) by the following algorithm:

Algorithm for Jacobi-GMRES(1,1,\epsilon)

\[ x_0 \in \mathbb{R}^3 \]

\[ b_0 = A x_0 \]

for \( k = 0, 1, \ldots \)

\[ x_{k+1} = x_k + \frac{b - A x_k}{\|A x_k\|_2} \]

then return \( x_{k+1} \)

\[ x_{k+1} = x_k + \frac{b - A x_k}{\|A x_k\|_2} \]

Main challenge:

Making time saved for data transfers by a well-chosen parallelization

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 \( \Omega \).

Algorithm for combined data transfer and matrix-vector product

Stream 1

Matrix-vector product on inner cells

Stream 2

Exchange boundary data

(\( \approx \) fast method for boundary data exchange)

Stream 0

Matrix-vector product on boundary cells

Results

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

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

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


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

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

Performance measurements

Benchmarkerking problem

A large air bubble rising in water

Simulation properties:

- Water box size: \( 0.2 \text{ cm} \times 0.2 \text{ cm} \times 0.2 \text{ cm} \) (0.2 cm = 0.04 m)
- Bubble radius: \( 0.03 \text{ cm} \) (0.0003 m)
- Volume force: gravity \( \mathbf{f} = (0.81, 0, 0)^T \)

Visualization of the simulation results: time: 80 s, 0.016 s

Results for the conjugate gradient solver

Speedup with growing problem sizes

Speedup using multiple GPUs

Results for the whole fluid solver

Speedup using multiple GPUs

Benchmarking platform

NVIDIA Tesla S1070 connected to two workstations with Intel Core 2 Duo CPU E8500 (3.33GHz) communicating over gigabit Ethernet

Time measurements

- Getting ready: compute mean
- During first 10 time steps
- 1000 CG iterations per time step
- Including time necessary for data transfers

Performance measurements on large distributed memory clusters

References


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

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