Computers & Fluids 80 (2013) 356–364

Contents lists available at SciVerse ScienceDirect

Computers & Fluids

journal homepage: www.elsevier.com/locate/compfluid

Solving incompressible two-phase flows on multi-GPU clusters

Peter Zaspel *, Michael Griebel [1]

Institute for Numerical Simulation, University of Bonn, Wegelerstraße 6, 53115 Bonn, Germany

ARTICLE INFO

Article history:
Received 11 August 2011
Received in revised form 4 January 2012
Accepted 22 January 2012
Available online 31 January 2012

Keywords:
Graphics processing units
Multi-GPU
Two-phase flows
Navier–Stokes equations
Level-set method
Finite difference

ABSTRACT

We present a fully multi-GPU-based double-precision solver for the three-dimensional two-phase incompressible Navier–Stokes equations. It is able to simulate the interaction of two fluids like air and water based on a level-set approach. High-order finite difference schemes and Chorin's projection approach for space and time discretization are applied. An in-depth performance analysis shows a realistic speed-up of the order of three by comparing equally priced GPUs and CPUs and more than a doubling in energy efficiency for GPUs. We observe profound strong and weak scaling on two different multi-GPU clusters.

1. Introduction

Moving forward to Exascale computing, the high performance computing (HPC) community has recognized the application of massively parallel hardware as one of the key ingredients to satisfy future computing requirements. One type of this hardware are graphics processing units (GPUs). They are a prototype for a general class of many-core processors with a high thread-parallelism which is expected to dominate future compute clusters. Consequently, there is now a growing number of multi-GPU-based HPC systems for which scientists need appropriate numerical software.

An important area for multi-GPU applications are computational fluid dynamics (CFD) simulations. Commercial CFD package vendors only slowly start to adapt this technology. But in academia, several groups have already published results on multi-GPU codes in CFD, e.g. for Lattice-Boltzmann applications [1,2], compressible fluids [3,4] or meteorology [5]. For grid-based incompressible flow simulations, which are described by the Navier–Stokes equations, Cohen and Molemaker were in [6] among the first to show multi-GPU results. They implemented a finite difference/volume code for single machine multi-GPU parallelism. The first truly distributed-memory MPI-based Navier–Stokes solver with finite differences was done by Jacobsen et al. [7]. Later, this group also showed extensions to this solver including a full geometric multigrid [8]. Other multi-GPU implementations of the Navier–Stokes equations are based on finite elements [9,3].

In our current research, we are especially focused on *two-phase* flow applications based on the Navier–Stokes equations. Kelly presented in [10] a single-GPU accelerated two-phase solver using the level-set method. Kuo et al. [11] accelerated their shared-memory parallel two-phase flow solver by a single-GPU-based Poisson solver. We are also aware of unpublished work on a multi-GPU two-phase flow solver by the group of Aoki et al. at the Tokyo Institute of Technology. However, to the best of our knowledge, there is no publication on a grid-based double-precision fully GPU-based parallel solver for the two-phase incompressible Navier–Stokes equations, which is able to scale on distributed memory multi-GPU clusters.

In this paper, we now present such a two-phase flow solver. It extends the introducing work by the authors in [12]. Like our original in-house CPU fluid solver NaSt3DGPF [13–16], the new GPU solver uses a finite difference discretization on a staggered grid in complex geometries and the continuum surface force approach to simulate two fluid phases like air and water. The fluid phases are distinguished by a level-set function. A range of applications, e.g. in the domain of droplet/ bubble dynamics [16] or water ways simulations [14] is thus able to profit from the performance available on large GPU clusters.

Note here that this publication contains several significant improvements and extensions in contrast to the authors' first publication on this topic [12]. The most important improvement is based on the seamless and full usage of GPUs for all computations. Now, besides just the Poisson solver and the level-set reinitialization, also all other GPU parallelizable parts of the described numerical method

* Corresponding author.
E-mail addresses: zaspel@ins.uni-bonn.de (P. Zaspel), griebel@ins.uni-bonn.de (M. Griebel).
[1] Principal corresponding author.

0045-7930/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compfluid.2012.01.021

are completely implemented on GPUs. This eliminates the large overhead of hundreds of GPU ↔ CPU data transfers as further explained in Section 3.1. Altogether, this leads to a major performance improvement of more than 30% compared to the authors paper [12] which is more precisely described in the dedicated Section 4.5. We additionally integrate a methodology based on performance per Watt and performance per Dollar benchmarks, which allows to have more fair performance comparisons, see Sections 4.1, 4.4 and 4.7 also with specific focus on Fermi GPU code optimizations. Finally, we show multi-GPU scaling on a 48 Fermi GPU cluster in Section 4.6.

The remainder of this article is organized as follows: In Section 2, we introduce the governing equations and the numerical methods applied in our solver. Section 3 presents details of the GPU implementation. Then, in Section 4, we review GPU benchmarking in general and give an in-depth performance analysis of the multi-GPU code. In Section 5, we discuss our results and provide final conclusions.

The major contributions of this work are as follows:

- For the first time, results of a fully GPU-based double-precision solver for the two-phase incompressible Navier–Stokes equations discretized on a grid using the level-set method are published.
- The solver is an MPI parallel multi-GPU code which scales on multi-GPU clusters. It thus fulfills the requirements for modern HPC systems.
- We present general guidelines for efficient and future-safe GPU implementations and give best practice rules for multi-GPU benchmarking.

2. Governing equations and numerical solution

We describe three-dimensional incompressible two-phase flow problems using the Navier–Stokes equations which are extended by a level-set formulation [17] to cope with phase-dependent densities and viscosities [18]. Surface tension effects at the free surface between the fluid phases are modeled by the continuum surface force method [19]. This approach has been previously described in detail in [16] for our CPU-based fluid solver NaSt3DGPF. We here just give a short sketch of the idea.

The model for two-phase incompressible fluids can be described in a set of equations with

$$\rho(\phi)\frac{D\vec{u}}{Dt} + \nabla p = \nabla\cdot(\mu(\phi)\mathbf{S}) - \sigma\kappa(\phi)\delta(\phi)\nabla\phi + \rho(\phi)\vec{g}, \tag{1}$$

$$\nabla\cdot\vec{u} = 0, \tag{2}$$

$$\partial_t\phi + \vec{u}\cdot\nabla\phi = 0, \tag{3}$$

where Eq. (1) is the momentum equation with time t, the fluid velocity $\vec{u}$, pressure p and the level-set function ϕ. The level-set function, a signed distance function with $|\nabla\phi| = 1$, implicitly describes the free surface $\Gamma_f = \{\vec{x}\in\Omega \mid \phi(\vec{x}) = 0\}$ and allows to define phase dependent densities $\rho(\phi)$ and viscosities $\mu(\phi)$ by

$$\rho(\phi) := \rho_2 + (\rho_1 - \rho_2)\,H(\phi),$$
$$\mu(\phi) := \mu_2 + (\mu_1 - \mu_2)\,H(\phi),$$

$$\text{with}\quad H(\phi) := \begin{cases} 0 & \text{if } \phi < 0,\\ \frac{1}{2} & \text{if } \phi = 0,\\ 1 & \text{if } \phi > 0,\end{cases}$$

$$\text{and } \phi(\vec{x},t) := \begin{cases} < 0 & \text{if } \vec{x}\in\Omega_1,\\ = 0 & \text{if } \vec{x}\in\Gamma_f,\\ > 0 & \text{if } \vec{x}\in\Omega_2.\end{cases}$$

Here, Ω_1 and Ω_2 are the domains of the two fluid phases and ρ_1, ρ_2, μ_1 and μ_2 denote the respective material parameters. The material derivative $\frac{D\vec{u}}{Dt}$ is given by $\frac{D\vec{u}}{Dt} := \partial_t\vec{u} + (\vec{u}\cdot\nabla)\vec{u}$. In the surface tension force term $\sigma\kappa(\phi)\delta(\phi)\nabla\phi$ of Eq. (1), κ identifies the curvature of the free surface and σ denotes the surface tension coefficient which is a material constant. Furthermore, δ is the Dirac-delta functional. Finally, $\vec{g}$ stands for volume forces, e.g. gravity and $\mathbf{S}$ is the stress tensor $\mathbf{S} := \nabla\vec{u} + \{\nabla\vec{u}\}^T$. Eq. (2) is the continuity equation and describes the incompressibility constraint. The last Eq. (3) models the transport of the level-set function, i.e. the dynamics of the free surface.

We discretize the above equations with the finite difference method on a staggered uniform grid. For numerical reasons, we apply a smoothing [18] to the Heaviside functional $H(\phi)$ and the Dirac-delta functional in an ϵ-environment of the free surface. Chorin's projection approach [20] then leads to a solution method which is described in detail in Algorithm 1. Here, we employed a first-order Euler time integration to get a simple formulation. In practical application problems, we use a second-order Adams–Bashforth time integration to compute the intermediate velocity field and to transport the level-set function. The time derivative for the artificial time τ in the level-set reinitialization process is discretized by a third-order Runge–Kutta method. Note that we have to reinitialize the level-set function after each transport step to recover the distance property $|\nabla\phi| = 1$ which is necessary for a correct evaluation of the free surface's normal and curvature κ. All transport terms and the level-set gradient in the reinitialization step are discretized using a fifth-order weighted essentially non-oscillatory (WENO) scheme, while the diffusion term in the second step is computed using second-order central differences. The Poisson equation is discretized by a second order method and solved with a Jacobi-preconditioned conjugate gradient (CG) method for sparse linear systems. Here, the Poisson equation's non-constant coefficients, namely the density jump for two-phase flows, lead to a high condition number for the linear system and thus to a slow convergence of the iterative solver. It is therefore a well-known fact that the Poisson solver dominates the overall run-time for such a solution method.

The full GPU solver supports different kinds of boundary conditions. These include slip and no-slip solid boundaries. Inflows and outflows can be controlled by Dirichlet and homogeneous Neumann boundary conditions for the velocity field. We introduce complex geometries by flagging out some cells of the simulation domain and apply appropriate boundary conditions at the fluid–solid interface cells of the obstacles.

Algorithm 1. Chorin's projection approach

for $n = 1, 2, \ldots$ do:

1. set boundary conditions for $\vec{u}^n$
2. compute intermediate velocity field $\vec{u}^*$:
$$\frac{\vec{u}^* - \vec{u}^n}{\delta t} = -(\vec{u}^n\cdot\nabla)\vec{u}^n + \frac{1}{\rho(\phi^n)}\nabla\cdot(\mu(\phi^n)\mathbf{S}^n) - \frac{1}{\rho(\phi^n)}\sigma\kappa(\phi^n)\delta(\phi^n)\nabla\phi^n + \vec{g}$$
3. apply boundary conditions and transport level-set function:
$$\phi^* = \phi^n + \delta t(\vec{u}^n\cdot\nabla\phi^n)$$
4. reinitialize level-set function by solving
$$\partial_\tau d + sign(\phi^*)(|\nabla d| - 1) = 0, \quad d^0 = \phi^*$$
5. solve the pressure Poisson equation with $\phi^{n+1} = d$:
$$\nabla\cdot\left(\frac{\delta t}{\rho(\phi^{n+1})}\nabla p^{n+1}\right) = \nabla\cdot\vec{u}^*$$
6. apply velocity correction:
$$\vec{u}^{n+1} = \vec{u}^* - \frac{\delta t}{\rho(\phi^{n+1})}\nabla p^{n+1}$$