Advances in Engineering Software 000 (2016) 1–6

Contents lists available at ScienceDirect

Advances in Engineering Software

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

Coupling of lattice-Boltzmann solvers with suspended particles using the MPI intercommunication framework

T.A. Puurtinen[a,*], J.I. Toivanen[a], K. Mattila[b], J. Hyväluoma[c], R.W. Nash[d], P.V. Coveney[d], J. Timonen[a]

[a] Department of Physics, University of Jyväskylä, P.O. Box 35, FI-40014 University of Jyväskylä, Finland
[b] Federal University of Santa Catarina, 88040-900 Florianópolis, SC, Brazil
[c] MTT Agrifood Research Finland, FI-31600 Jokioinen, Finland
[d] Centre for Computational Science, Department of Chemistry, University College London, London WC1H 0AJ, UK

ARTICLE INFO

Article history:
Received 27 April 2015
Revised 14 July 2016
Accepted 18 July 2016
Available online xxx

Keywords:
MPI
Coupling
Lattice-Boltzmann
Advection-diffusion

ABSTRACT

The MPI intercommunication framework was used for coupling of two lattice-Boltzmann solvers with suspended particles, which model advection and diffusion respectively of these particles in a carrier fluid. Simulation domain was divided into two parts, one with advection and diffusion, and the other with diffusion only (no macroscopic flow). Particles were exchanged between these domains at their common boundary by a direct process to process communication. By analysing weak and strong scaling, it was shown that the linear scaling characteristics of the lattice-Boltzmann solvers were not compromised by their coupling.

1. Introduction

The lattice-Boltzmann method (LBM) has attracted interest in several fields of computational physics, such as fluid dynamics [1], quantum mechanics [2] and electrodynamics [3]. LBM can easily exploit massive MPI parallelism, threading [4] and co-processors [5] offered by contemporary HPC systems. It is well suited for HPC because of its theoretically linear scalability arising from the locality of the method. Problem domain can be divided into subdomains that exchange information only at their boundary, and communication can usually take place alongside with computation. LBM is efficient especially in problems related to heterogeneous and porous media, which are difficult to describe by mesh based numerical methods. It has been recently demonstrated that LBM can scale to very large, and complicated systems, with several hundred thousand to more than million cores. [6,7] It is largely acknowledged in the HPC community that parallelism keeps increasing in the foreseeable future. This makes LBM one of the best candidates for the above simulation problems when HPC is needed.

HemeLB is one of the co-design codes in the CRESTA project. HemeLB has been developed to simulate blood flow in a complete physiological model of the human body. [8] There are several recent works that address computational hemodynamics by coupling of models. For instance, Golbert et al. [9] use coupled method to model fluid structure interactions between blood and arterial walls, Amati et al. [10] use a multi-scale procedure for blood flow by coupling LB and micro-scale particle dynamics models, and Bernaschi et al. [11] use coupled LB and molecular dynamics to study red blood cell suspension flows in arteries with realistic shapes and sizes.

Coupling of different models is a natural solution for constructing the whole model because of the variety of different physics and length scales involved. An efficient solution of such a system can involve the use of fundamentally different methods, which must however exchange information related to a set of their variables. External commercial and open source solver couplers such as MpCCI [12], OpenPALM [13] and OASIS [14] provide means to run two or more solvers simultaneously or in a serial mannner. They also provide various interpolation algorithms for the conversion of field variables between solvers and performance analysis tools. Use of external coupler modules requires however allocation of additional computational resources, and can become a bottleneck in large scale simulations because of possibly inefficient interpolation algorithms and serialised communication patterns. Communication strategies relevant to massively parallel simulations are reviewed e.g. in [15].

In this article we consider application of our MPI parallel lattice-Boltzmann code with suspended particles to simulation of

* Corresponding author.
E-mail addresses: tuomas.a.puurtinen@jyu.fi (T.A. Puurtinen), jussi.timonen@jyu.fi (J. Timonen).

http://dx.doi.org/10.1016/j.advengsoft.2016.07.008
0965-9978/

colloidal drug particles in blood flow with coupling to a diffusive particle system. For a review of such systems see e.g. [16,17]. The purpose of this work was to investigate the parallel efficiency of coupling built into the original solvers.

Section 2 introduces the LB solver with suspended particles used here to test the coupling interface. In Section 3 coupling of two versions of this solver is described, and the weak and strong scaling of the coupled system are both analysed. Conclusions are then drawn in Section 4.

2. The lattice-Boltzmann solver

The lattice-Boltzmann method can be considered as a finite difference discretization of the discrete-velocity Boltzmann equation (DVBE),

$$\frac{\partial f_i}{\partial t} + \vec{c}_i \cdot \frac{\partial f_i}{\partial \vec{r}} = \mathcal{J}_i(\vec{f}), \quad i = 1, 2, \ldots, q, \tag{1}$$

which describes the evolution of mass-density distribution function $\vec{f}$ of 'fluid particles' in the case when each of these 'particles' has a velocity belonging to a discrete set $\{\vec{c}_1, \vec{c}_2, \ldots, \vec{c}_q\}$. Here $f_i = f_i(\vec{r}, \vec{c}_i, t)$, t is the time and $\vec{r}$ the position of a fluid particle, operator $\mathcal{J}_i$ describes collisions of these particles and $\vec{f} = \left(f_1, f_2, \ldots, f_q\right)^T$. It can be shown that the incompressible Navier–Stokes equation can be recovered from the above equation in the nearly incompressible limit [18].

The strengths of the method include simplicity of coding, straightforward incorporation of microscopic interactions and suitability for parallel computing. Moreover, it is relatively straightforward to construct the computation geometry directly from images, since the method only requires a binary representation of the fluid and solid phases of the domain instead of triangulation of the boundaries or volumes (which can of course be used).

The degrees of freedom in LBM are real-valued distribution functions $f_i(\vec{r}, t)$, where $\vec{r}$ is now a lattice site. Our implementation of LBM uses the D3Q19 grid model [19] which has 19 discrete velocities in a three dimensional lattice. In the single relaxation time approximation (SRT), known as the BGK model [20], distribution functions are updated such that

$$f_i(\vec{r} + \Delta t c_i, t + \Delta t) = f_i(\vec{r}, t) - \frac{1}{\tau}\left(f_i(\vec{r}, t) - f_i^{\mathrm{eq}}(\vec{r}, t)\right). \tag{2}$$

The last term on the right hand side of the equation represents the collisions through which the distribution function is relaxed towards the local equilibrium,

$$f_i^{\mathrm{eq}} = \rho(\vec{r}, t) w_i \left(1 + \frac{\vec{c}_i \cdot \vec{v}}{c_s^2} + \frac{(\vec{c}_i \cdot \vec{v})^2}{2c_s^4} - \frac{||\vec{v}||^2}{2c_s^2}\right). \tag{3}$$

Parameter τ is the relaxation time that characterises the collision process, and is related to the kinematic viscosity of the fluid by the relation $\nu = (2\tau - 1)/6$. Constant c_s is the speed of sound, in this case $c_s = 1/\sqrt{3}$ in lattice units, and w_i are weight factors.

After a collision step the distribution values are transferred to neighbouring lattice sites according to their velocities, as evidenced by the left hand side of Eq. (2). This is called the streaming step, and in the present code it is realised using the swap algorithm described in [21].

The macroscopic mass and momentum densities can be computed using

$$\begin{aligned} \rho(\vec{r}, t) &= \sum_i f_i(\vec{r}, t), \\ \rho(\vec{r}, t)\vec{v}(\vec{r}, t) &= \sum_i \vec{c}_i f_i(\vec{r}, t), \end{aligned} \tag{4}$$

from which the macroscopic velocity $\vec{v}$ can be obtained.

We remark that SRT has some known deficiencies in terms of physical accuracy and stability for no-slip boundaries (see e.g. [22]). However, the choice of the relaxation time model does not affect the implementation of coupling of the particle system, and could be changed in the case of a more accurate fluid model is required. In the subsequent scalability tests we choose parameters that permit convergence for the SRT model.

In addition to the fluid dynamics model, we also consider advection and diffusion of tracer particles suspended in the fluid, which are penetrable to the fluid so that the volume interactions can be excluded. We assume that the motion of these particles is governed by the Langevin equation,

$$m\ddot{X} = \gamma m(\vec{v} - \dot{X}) + \gamma m\sqrt{2D}R(t) + \vec{F}(t), \tag{5}$$

where m is the mass of the particle, γ is a damping factor characterising the amount of inertia, D is the molecular diffusion coefficient, the stationary Gaussian process $R(t)$ satisfies $\langle R(t)\rangle = 0$ and $\langle R(t)R(t')\rangle = \delta(t - t')$, and F represents all the other forces present in the system.

Particle positions are updated by integrating Eq. (5) in time using the velocity Verlet algorithm [23]. To this end, the macroscopic flow velocity $\vec{v}$ in the position of the particle is required. Particle positions are not restricted to the lattice sites, and thus interpolation between the lattice sites of the velocity is performed as described in [24].

The LB code was parallelised with MPI [25] using the single program multiple data (SPMD) paradigm. We used a regular box domain decomposition, where the domain can be divided into any given number of slices along all three coordinate axes, but the subdomains must be of the same size. For details of the parallel implementation, see [24].

We note that a simple box decomposition is often inadequate for complicated realistic vascular models, and instead, more sophisticated decomposition methods should be used. One can use external partitioner libraries, such as ParMETIS [28] (which is used in HemeLB [8]), or choose one of the commonly used methods including orthogonal recursive bisection [26] or cell-based decomposition [27] which can be built into the simulation code.

3. Coupling of suspended particles in two LB solvers

So as to provide a simple model for colloidal drug particles in blood flow we couple two copies of the LB solver, which both govern a distinct part of the full domain. One solver is responsible for the vasculature and the other controls the diffusion in the surrounding tissue. When a particle in the vascular flow moves into a boundary cell of the LB lattice, it has a predefined probability for transfer through the interface to the surrounding tissue, where it can only move by diffusion. The effective permeability of the interface is controlled by a probability coefficient and a probability cut-off function defined by cell centres of the LB lattice (cf. the cross sections in Fig. 2). This configuration can be used as an advection-diffusion model for a piece of artificial vasculature (see Fig. 1). This advection-diffusion model has numerous other medical and industrial applications, it can describe for example spreading of tracer particles in the soil or ground water.

Our strategy in the coupling of particle systems of the two solvers is direct process to process MPI communication. Choice of this coupling model was natural because communication pairs will be static during the simulation. Instead of using an existing coupler, we construct a communication interface within each solver. The benefit of this approach is that the interface can be designed as light as possible, and that communication can coincide with computation, whereas many external couplers include sequential steps, which may cause performance issues in highly parallel setups.

Please cite this article as: T.A. Puurtinen et al., Coupling of lattice-Boltzmann solvers with suspended particles using the MPI intercommunication framework, Advances in Engineering Software (2016), http://dx.doi.org/10.1016/j.advengsoft.2016.07.008