Journal of Computational Science 2 (2011) 105–112

Contents lists available at ScienceDirect

Journal of Computational Science

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

WaLBerla: HPC software design for computational engineering simulations

C. Feichtinger*, S. Donath, H. Köstler, J. Götz, U. Rüde

Friedrich-Alexander University of Erlangen-Nuremberg, Erlangen, Germany

ARTICLE INFO

Article history:
Received 6 April 2010
Received in revised form 8 November 2010
Accepted 17 January 2011
Available online 29 March 2011

Keywords:
Lattice-Boltzmann method
MPI
Software Quality
Software Engineering

ABSTRACT

WaLBerla (Widely applicable Lattice-Boltzmann from Erlangen) is a massively parallel software framework supporting a wide range of physical phenomena. This article describes the software designs realizing the major goal of the framework, a good balance between expandability and scalable, highly optimized, hardware-dependent, special purpose kernels. To demonstrate our designs, we discuss the coupling of our Lattice-Boltzmann fluid flow solver and a method for fluid structure interaction. Additionally, we show a software design for heterogeneous computations on GPU and CPU utilizing optimized kernels. Finally, we estimate the software quality of the framework on the basis of software quality factors.

1. Introduction

Simulation becomes a more and more important fundamental approach to scientific discoveries that complements theory and experiment. The multi-disciplinary field of computational science and engineering (CSE) has been established in order to deal with large scale computer simulations and optimization of mathematical models. CSE is used successfully e.g. by aerospace, automotive, and processing industries, as well as in medical technology. Many applications in CSE require large scale computation and are performed on high performance computing (HPC) clusters, therefore software development in CSE is dominated by the need of efficient and scalable codes. On the one hand, the various applications from different fields show a significant overlap in the underlying physical or mathematical models and therefore basic numerical algorithms can be reused. On the other hand, the trend towards multiphysics simulation forces current frameworks to also support various numerical algorithms. This requires the development of modular and easily extendable software frameworks that cover a wide range of applications. In this article, we describe our massively parallel multiphysics software framework WaLBerla that is originally centered around the Lattice-Boltzmann method (LBM), but its applicability is not limited to this algorithm. The LBM within WaLBerla serves as an alternative to a Navier Stokes solver for computing instationary flow phenomena. A description of the LBM is given in Section 2. Examples of established Lattice-Boltzmann frameworks are, e.g. Peano [6], Virtual Fluids [7], ILBDC [8], Sailfish [9], Palabos [10], Muphy [11], and Ludwig [12]. Compared to WaLBerla these frameworks differ in fundamental design decisions, some e.g. use different programming languages like python instead of C++, differ in the underlying data structures, e.g. list based data structures instead of block structured grids, are optimized for a single hardware or simulation task, provide an interface for interactive steering, and some do not support massively parallel simulations.

Right from the start WaLBerla has been implemented utilizing software engineering concepts and common design patterns [13]. The general software development process in WaLBerla is a mixture of the spiral model and prototyping [14,15]. During this process, prototypes are defined with certain specifications, which are implemented by our software designs. The designs introduced in this work enable us to realize a general and expandable framework while maintaining runtime efficiency and scalability. In detail, the WaLBerla 1.0 prototype is designed to fulfill the following goals:

- *Understandability and usability*: Easy integration of new simulation scenarios and numerical methods also by non-programming experts.
- *Portability*: Portable to various HPC supercomputer architectures and operating system environments.
- *Maintainability and expandability*: Integration of new functionality without major restructuring of code or modification of core parts in the framework.
- *Efficency*: Possibility to integrate optimized kernels to enable efficient, hardware-adapted simulations.
- *Scalability*: Support of massively parallel simulations.

* Corresponding author.
E-mail address: Christian.Feichtinger@informatik.uni-erlangen.de (C. Feichtinger).

1877-7503/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.jocs.2011.01.004

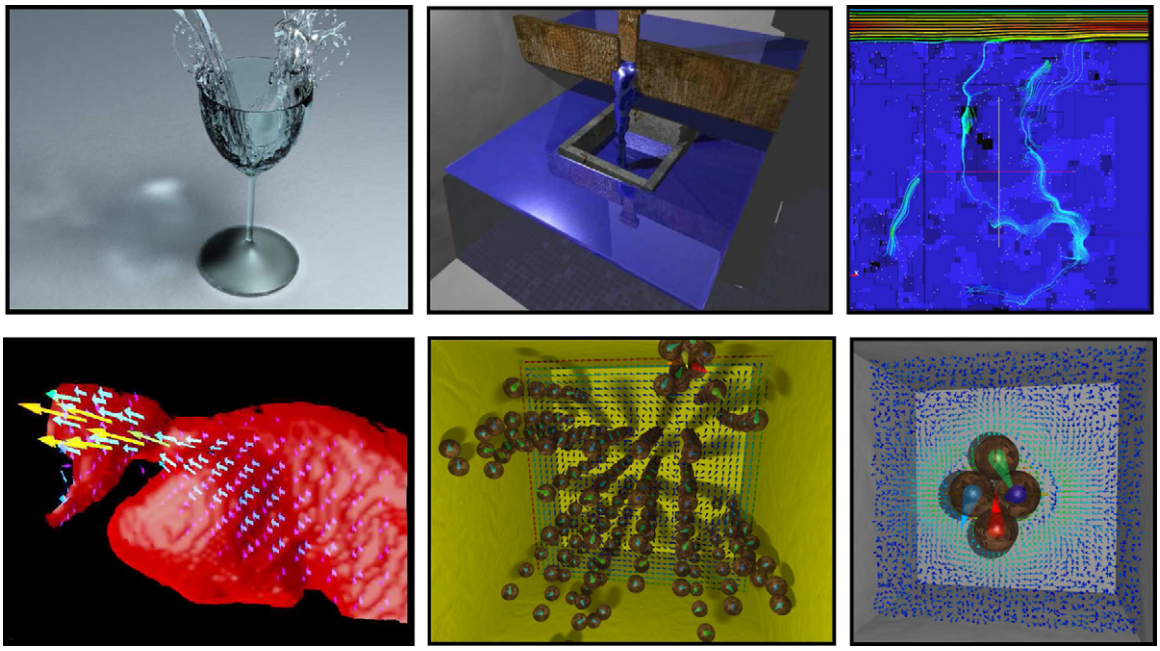

Fig. 1. Selection of simulation tasks in WaLBerla. The implementation has also been carried out within several cooperations and by master students. Form top left to lower right, we have included free-surface flows [1], free-surface flows with floating objects [2], flows through porous media, clotting processes in blood vessels [3], particulate flows for several million volumetric particles [4] on up to 8192 cores, and a fluctuating Lattice-Boltzmann [5] for nanoscale flows where thermal fluctuations are important.

With this prototype it has already been possible to solve various complex simulation tasks on massively parallel systems. Some of them can be seen in Fig. 1.

Currently, a new prototype is under development extending WaLBerla 1.0 to the needs of novel architectures with accelerators, such as GPUs, to new simulation tasks, and the increasing number of users. In addition to WaLBerla 1.0, e.g. the new prototype WaLBerla 2.0 has refined the design goals:

- *Understandability and usability*: Further improvement of the software quality by forcing a modular implementation using dynamic shared libraries and applying common C++ design patterns.
- *Portability*: Enable heterogeneous computations involving e.g. GPUs and CPUs by supporting CUDA or OpenCL kernels.
- *Maintainability and expandability*: Extension of the framework by adaptive grid-refinement.
- *Efficency and scalability*: Implementation of load balancing strategies.

Heterogeneous computations on CPUs and GPUs are already supported by WaLBerla 2.0 and will be discussed in detail. The designs for load balancing and adaptivity are currently under development.

Our results described in this article are structured as follows: the Lattice-Boltzmann method is described in Section 2, followed by the introduction of the software designs of WaLBerla 1.0 and 2.0 in Sections 3 and 4. In Section 5 the software quality of the framework will be discussed with respect to understandability, usability, reliability, portability, maintainability, extensibility, efficiency, and scalability. The article is concluded in Section 6.

2. The Lattice-Boltzmann method

The Lattice-Boltzmann method is one possible approach to solve computational fluid dynamics problems numerically. Computationally, the LBM is based on a uniform grid of cubic cells that are updated in each time step using an information exchange with nearest neighbor cells only. Structurally, this is equivalent to an explicit time stepping for a finite difference scheme, or also a cellular automaton. Different from conventional computational fluid dynamics, the LBM uses a set of particle distribution functions (PDF) in each cell. A PDF is defined as the expected value of particles in a volume located at the lattice position x_i with the lattice velocity $e_{\alpha,i}$. For the LBM the lattice velocities $e_{\alpha,i}$ determine the finite difference stencil, where α represents an entry in the stencil. In 3D and with the so-called D3Q19 model [16] a 19 point stencil is used resulting in 19 PDFs in each cell. Hence, in each time step 19 PDFs have to be advected for each cell to the neighboring cells, which is followed by the application of a collision operator. Discretized in time and space, and with the single relaxation time collision operator [16] the LBM is given by:

$$f_\alpha(x_i + e_{\alpha,i}\delta t, t + \delta t) - f_\alpha(x_i, t) = -\frac{\delta t}{\tau}[f_\alpha(x_i, t) - f_\alpha^{(eq)}(\rho(x_i, t), u_i(x_i, t))], \quad (1)$$

where t is the time and δt is the length of one discrete time step. The relaxation time τ can be determined from the lattice viscosity ν (Eq. 2). Further, $f_\alpha^{(eq)}(\rho, u_i)$ is the equilibrium distribution depending on the macroscopic velocity u_i and the macroscopic density ρ. For the isothermal case it is given by the Maxwell–Boltzmann distribution function discretized for low Mach numbers. The macroscopic quantities of interest (ρ, p, u_i) are determined from the moments of the distribution functions:

$$\rho u_i = \sum_{\alpha=0}^{18} e_{\alpha,i} \cdot f_\alpha \qquad \rho = \sum_{\alpha=0}^{18} f_\alpha,$$

$$p = c_s^2 \rho,$$

$$\nu = \left(\tau - \frac{1}{2}\right) c_s^2 \, \delta t. \quad (2)$$

In multiphysics applications, the LBM must be coupled to other models. These include additional field equations, e.g. for temperature, energy or electrostatic fields that can be discretized by finite differences, finite volumes, or finite elements. WaLBerla provides