Available online at www.sciencedirect.com
ScienceDirect
Procedia Computer Science 1 (2012) 1075–1082
Procedia Computer Science
www.elsevier.com/locate/procedia

International Conference on Computational Science, ICCS 2010

Lattice Boltzmann fluid-dynamics on the QPACE supercomputer

L. Biferale[a], F. Mantovani[b,c,*], M. Pivanti[b], M. Sbragaglia[a], A. Scagliarini[a], S. F. Schifano[d], F. Toschi[e], R. Tripiccione[b]

[a] *Department of Physics and INFN, University of Rome "Tor Vergata", via della Ricerca Scientifica 1, 00133 Rome, Italy*
[b] *Dipartimento di Fisica, Università di Ferrara and INFN - Sezione di Ferrara, I-44100 Ferrara, Italy*
[c] *Deutsches Elektronen-Synchrotron (DESY), 15738 Zeuthen, Germany*
[d] *Dipartimento di Matematica, Università di Ferrara and INFN - Sezione di Ferrara, I-44100 Ferrara, Italy*
[e] *Department of Physics and Department of Mathematics and Computer Science, Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands; and International Collaboration for Turbulence Research*

Abstract

In this paper we present an implementation for the QPACE supercomputer of a Lattice Boltzmann model of a fluid-dynamics flow in 2 dimensions. QPACE is a massively parallel application-driven system powered by the Cell processor. We review the structure of the model, describe in details its implementation on QPACE and finally present performance data and preliminary physics results.

Keywords: Fluid-dynamics, Lattice Boltzmann Model, CBE processor, QPACE supercomputer

1. Overview

Fluid-dynamics is studied today with the critical support of numerical techniques, that allow to compute solutions for the highly non-linear equations of motion in regimes and geometries interesting in physics or engineering contexts. Over the years, many different numerical approaches – implicit and explicit, in direct space or Fourier space – have been theoretically developed and implemented on several massively parallel computers.

The Lattice Boltzmann method (LBM) is a very flexible approach able to encode (in its many versions) many different fluid equations (e.g., multiphase, multicomponent and thermal fluids) and to consider complex geometries and boundary conditions. LBM builds on the fact that the details of the interaction among the fluid components at the microscopic level are irrelevant in defining the structure of the equations of motion at the macroscopic level, but only modulate the values of their parameters. The key idea is that of creating on the computer simple synthetic dynamics of fictitious particles (the "populations") that evolve explicitly in time and, appropriately averaged, provide the correct values of the macroscopic quantities of the flow; see [1] for a complete introduction.

From the computational point of view, LBM is "local" (i.e. it does not involve computation of pressure or other non local fields, communications are only amongst nearest neighbor nodes), so it is easy to parallelize. This has already been done in several cases, e.g. for irregular geometries [2] and even for the Cell processor [3], with focus on biomedical applications. In this paper, we report on a high-efficiency implementation of LBM for QPACE, a massively

*Corresponding author: filippo.mantovani@fe.infn.it
1877-0509
doi:10.1016/j.procs.2010.04.119

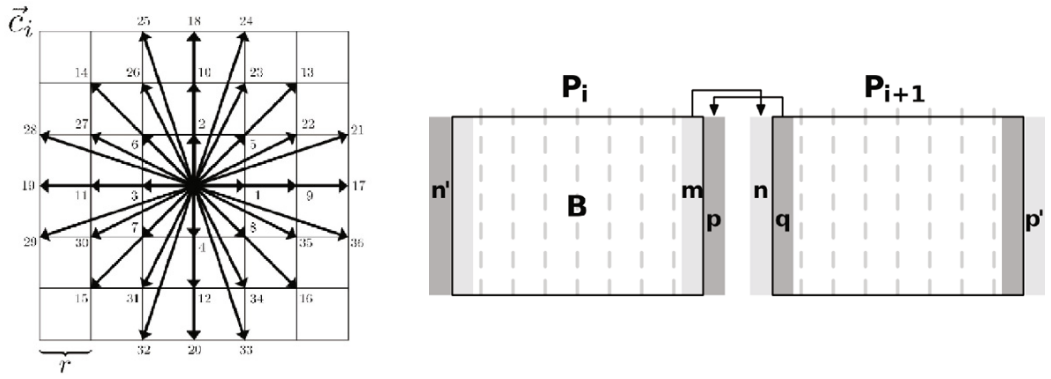

Figure 1: Left: Velocity scheme for the D2Q37 LBM model (see [4] for more details). Right: Graphical representation of the data organization on the processing nodes. Two neighbor nodes (P_i and P_{i+1}) are shown. Populations handled by each processor are stored in area B. Vertical strips n and p (and n' and p') store a copy of the first 3 columns of data updated by the neighbor processor. Vertical dashed lines show data splitting within each processor: each SPU updates one strip of B.

parallel system powered by IBM Cell processors: the challenge is twofold; i) adapting a complex numerical algorithm to the architecture of the Cell and ii) partitioning the computation on a large set of processing elements.

Our paper is structured as follows: in section 2 we briefly review the structure of our LBM algorithm, while in section 3 we recall the QPACE architecture. Section 4 describes our implementation and section 5 reviews our performance results. Section 6 reports on preliminary physics results, and is followed by our concluding remarks.

2. The Lattice Boltzmann approach

Here we introduce the system studied in terms of its mathematical formulation along with a brief description of the computational method employed. More details, along with validations can be found in [4].

The Thermal-Kinetic description of a compressible gas/fluid of variable density, ρ, local velocity $\boldsymbol{u}$, internal energy, $\mathcal{K}$ and subject to a local body force density, $\boldsymbol{g}$, is given by the following equations: $\partial_t\rho + \partial_i(\rho u_i) = 0$, $\partial_t(\rho u_k) + \partial_i(P_{ik}) = \rho g_k$, $\partial_t\mathcal{K} + \frac{1}{2}\partial_i q_i = \rho g_i u_i$ where P_{ik} and q_i are the momentum and energy fluxes.

It has been shown that it is possible to recover exactly these equations, starting from a continuum Boltzmann Equations and introducing a suitable shift of the velocity and temperature fields entering in the local equilibrium [5]. The lattice counterpart of the continuum description can be obtained through the usual lattice Boltzmann discretization ($f_l(x,t)$ are the fields associated to the populations):

$$f_l(\boldsymbol{x} + \boldsymbol{c}_l\Delta t, t + \Delta t) - f_l(\boldsymbol{x}, t) = -\frac{\Delta t}{\tau}\left(f_l(\boldsymbol{x}, t) - f_l^{(eq)}\right)$$

where the equilibrium is expressed in terms of hydrodynamical fields on the lattice, $f_l^{(eq)}(\boldsymbol{x}, \rho, \bar{\boldsymbol{u}}, \bar{T})$, and the subscript l runs over the discrete set of velocities, $\boldsymbol{c}_l$ (see fig. 1). The macroscopic fields are defined in terms of the lattice Boltzmann populations: $\rho = \sum_l f_l$, $\rho\boldsymbol{u} = \sum_l \boldsymbol{c}_l f_l$, $D\rho T = \sum_l |\boldsymbol{c}_l - \boldsymbol{u}|^2 f_l$ (for simplicity and clarity of exposition, here we ignore a shift that has to be applied to the fields entering in the Boltzmann equilibrium, see [5] for details).

Lattice discretization induce non trivial corrections terms in the macroscopic evolution of averaged hydrodynamical quantities. In particular both momentum and temperature must be renormalized by discretization effects in order to recover the correct thermal kinetic description out of the discretized LBM variables: the first correction to momentum is given by the pre and post-collisional average [11, 12]: $\boldsymbol{u}^{(H)} = \boldsymbol{u} + \frac{\Delta t}{2}\boldsymbol{g}$ and the first non-trivial, correction to the temperature field by [5]: $T^{(H)} = T + \frac{(\Delta t)^2 g^2}{4D}$. ($D$ is the dimensionaly of the system). Using these "renormalized" hydrodynamical fields it is possible to recover, through a Taylor expansions in Δt, the thermo-hydrodynamical equations [5, 4]:

$$D_t\rho = -\rho\partial_i u_i^{(H)} \tag{1}$$

$$\rho D_t u_i^{(H)} = -\partial_i p - \rho g \delta_{i,3} + \nu \partial_{jj} u_i^{(H)} \quad (2)$$

$$\rho c_v D_t T^{(H)} + p \partial_i u_i^{(H)} = k \partial_{ii} T^{(H)} \quad (3)$$

where we have introduced the material derivative, $D_t = \partial_t + u_j^{(H)} \partial_j$, and we have neglected viscous dissipation in the heat equation (usually small). Moreover, c_v is the specific heat at constant volume for an ideal gas $p = \rho T^{(H)}$, and ν and k are the transport coefficients.

In this paper we consider a 2D LBM algorithm, that uses 37 population fields (a so called D2Q37 model), moving as shown in fig. 1 (left). At each time step populations move up to 3 lattice points away; this has an impact on the parallelization strategy.

3. The QPACE supercomputer

QPACE – described in details elsewhere [6, 7, 8] – is a novel massively parallel computer, primarily developed for Lattice QCD simulations. It turns out (see later for all details) that this machine almost ideally fits the needs of LBM algorithms.

QPACE is powered by IBM PowerXCell 8i processors (an enhanced version of the well known Cell processor). Processing nodes have one PowerXCell 8i processor; they are interconnected by a custom application-optimized 3-dimensional torus network. Each node has a peak performance of $\sim$ 100 Gflops (double precision) and a private memory bank of 4 Gbyte; the peak bandwidth between processor and memory is 25 Gbyte/sec, while each of the 6 bi-directional communication links of the torus fabric has a bandwidth of $\sim$ 900 Mbyte/sec. QPACE machines are assembled in blocks of 32 processing elements (informally known as "backplanes"); large systems have 1024 closely interconnected processing nodes each, providing a peak performance of the order of $\sim$ 100 TFlops.

QPACE nodes are independent systems, running Linux. A large application on QPACE has two hierarchically structured levels of parallelism:

- The inner level stays inside each processor on the node, containing one control processor (the so-called PPU) and 8 independent floating-point optimized computing cores (so called SPUs). The workload belonging to each node has to be divided in a balanced way between the cores, which are able to exchange data on an on-chip high bandwidth bus and share access to just one external memory bank.

- The outer level of parallelism has to do with mapping the application on the nodes connected by the torus network. Multiple instances of the application program start independently on all nodes; they exchange data by passing messages over the network. From this point of view, the programming style appropriate for QPACE is conceptually similar to the well-known and widely used MPI model. A major difference is however that the network performs data `send` and `receive` operations only between adjacent nodes in the torus, a limitation with no significant impact for LBM applications, as shown later on. In exchange for this restricted set of allowed communication patterns, the torus network has a low communication latency of the order of $\simeq 3\mu$sec. From the programmer point of view, this means that there is no real need to gather large amount of data and send them to a destination node as a single message, since even small messages are transfered with high effective bandwidth (bandwidth of approximately 50% (85%) of peak have been measured for messages of 512 (2048) bytes).

Applications achieve performance on QPACE if they are mapped in a balanced way at both levels of parallelism, and if data exchange and computation can be overlapped in time, so parallelization overheads are limited. In the next section we describe how we reach this goals for our LBM algorithm.

4. Porting LBM on QPACE

Grid-like interconnection structures are perfectly fit for massively parallel implementations of the LBM algorithm. In fact the physical lattice can be regularly tiled onto the available processing elements and the corresponding communication pattern only involve data exchanges between neighbor nodes in the grid; successful attempts in this direction on massively parallel machines are almost 20 year old [9].

In our 2D simulation, we divide our physical lattice of size $L_x \times L_y$ in equally sized vertical strips, each strip containing $(L_x/N_p) \times L_y$ physical points (N_p is the number of processing elements). Processors handling adjoining strips in the physical lattice exchange data associated to the right and left borders of the lattice. This arrangement does not minimize data traffic among processing elements, but we do not need more clever partitioning since the overheads are small (see later for details). We use two appropriate communication channels on each processing element of the QPACE torus and establish an 1D path that orderly links the processors handling successive strips in the physical lattice. The toroidal structure makes it possible to enforce periodic boundary conditions in the x direction.

On each processing element, we prepare a data structure for the population data that includes two vertical borders – each 3-point wide, corresponding to the largest distance a population can move at each time step – at the right and left edges of the local lattice. The parallelization strategy is straightforward: at the beginning of each iteration, we copy the population values of the outermost 3 vertical lines on each side of the lattice onto the borders of the neighboring processors; after this step is made, each processor works independently, updating its physical points and using data from the outer borders as needed (see fig. 1, right).

Before entering into the details of our implementation, let us derive approximate upper bounds on the performance that we may expect on QPACE. If we are able to fully overlap in time i) data processing, ii) access to the memory bank of each processor and iii) data exchange with the neighbor processors (and neglecting any control overhead) we compute the new populations at each lattice site in a time $T \geq max\,(W/F, I/B_M, E/B_N)$, where W is the workload for each lattice point and F the performance of the processor; I is the amount of data moved between processor and memory for each point and B_M is the memory bandwidth; finally, E is the information exchanged with the neighbor processors, and B_N is the interconnection bandwidth. For QPACE, we have $F \simeq 100$Gflops, $B \simeq 25$Gbyte/sec, $B_N \simeq 0.9$Gbyte/sec. For each lattice point, our algorithm needs ~ 7850 double precision floating-point operations; we read from memory 37 incoming populations and store 37 outgoing populations. We have to exchange data with neighbor processors only for the right and left vertical borders, so $E = 37 \cdot 8 \cdot 6 \cdot N_p/L_x$. Our equation reads (the time unit is ns):

$$T \geq max\,(78.5, 23.7, 1973 \times (N_p/L_x)) \tag{4}$$

This equation shows that: i) full performance might be expected for this algorithm on QPACE, and ii) parallelization overheads can be hidden in principle as long as the third term in the equation is smaller than the first one, that is for $L_x/N_p > 25$. In practice this nice upper limit is difficult to reach, for three main reasons: overlap of the various operations is difficult to achieve in a real life code, actual memory bandwidth is much lower that its peak value for sparse access patterns and – for practical reasons - we move data from/to memory more often than required in principle. We now describe in detail our implementation, that follows the lines of some earlier exploratory attempts [10].

The data structure for the dynamical variables is an array of structures called `pop`. Each pop represents a site of the 2D physics lattice and has 48 double words (384 bytes): 37 floating point numbers for the population components and 11 more values associated to the macroscopic physical values and to some padding in order to keep data alignment on 128 byte boundaries, as required by the Cell architecture for DMA transfers.

The LBM kernel evolves the system for one time step. It uses 4 main routines;

- `comm` copies onto the border locations the fresh data generated by the previous iteration (see fig. 1 (right)). This is the only routine involving internode communication.
- `stream` gathers – for each lattice site – all populations that will "collide" among each other in order to evolve into the new set of populations at the next time step. This process involves several small accesses at non contiguous memory locations.
- `bc` sets the boundary conditions at the top and bottom walls of the cell, computing the values of the populations lying on a thin layer of sites close to the wall. This routine is floating point intensive but involves less than 0.1% of all sites, so its computational impact is negligible.
- `coll` performs the mathematical steps associated to the collision of the populations gathered by `stream` and computes the new set of populations.

Routines `stream` and `coll` move data from main memory to processor and vice versa two times: merging them together would halve memory traffic and should increase performance; however we kept them as independent routines since attempts at optimizing at the same time many memory accesses and heavy computation proved to be a "chaotic" process.

The general organization of the algorithm described above has been carefully adapted to the details of the Cell architecture. `coll` is executed by each SPU on independent subsets of the lattice. We operate concurrently on pairs of adjacent sites so we exploit the vector (SIMD) datapath on the SPU. We accurately overlap data transfer and calculation using double buffering technique. `stream` is also executed by all SPUs, even if it does not perform compute intensive tasks. By having the SPUs post independent memory access requests, we partially hide the large latencies associated to the sparse addressing patterns. Performance is remarkably better than obtained if we execute the routine on the PPU, but still much lower than peak. Finally, the SPUs also handle communications, as required by the QPACE network architecture that only supports SPU-to-SPU transfers.

5. Performance analysis

In December 2009 we used QPACE to perform the first large scale physics campaign since the machine was commissioned. Our main focus of interest has been a high-resolution study of the properties of the Rayleigh-Taylor instability. A short summary of preliminary physics results is given in the next section.

We used for our simulation up to seven QPACE backplanes (e.g., a total of 224 processing elements). After tuning the relevant physics parameter with some preliminary runs, we extensively simulated two lattices, that we label *type A* and *type B*. Type A lattices have a size of 4096×6000; they are followed for $2.6 \cdot 10^5$ time steps. Type B lattices are smaller, 2048×3600; we evolve them for a longer time span of 10^6 time steps. For each of the two lattices, we performed 25 independent runs with different initial configurations. Each simulation was run on a set of 32 interconnected processing elements, several independent runs being in progress on different backplanes at the same time. Each run takes approximately 32 hours on type A lattices and 44 hours on type B lattices. All in all approximately 61000 Cell-CPU hours have been used in about 15 days wall-clock time (average up-time $\sim$ 75%). We have dumped configurations of key physics variables (ρ, u, v, T) on the whole lattices at regular time intervals, for later off-line analysis. Altogether our simulation data-base is $\sim$ 1.5TB.

Table 1: Performance figures for LBM on QPACE as a function of the number of processors. For each lattice size and number of processors we list the time spent in the 4 kernel routines and the total time (T) for one time step (time units are ms, except for the last column). In the last column, we list (in ns) T multiplied by the number of processors and divided by the number of lattice sites; this figure – a constant for perfect scaling – fluctuates by less than 10%.

	`comm`	`stream`	`bc`	`coll`	T	$T \times N_p/(L_x L_y)$
2048x3600@32p	11.6	37.2	0.8	71.8	121.4	527
2048x3600@16p	11.7	81.0	1.7	143.6	238.0	516
2048x3600@08p	11.6	148.9	3.3	287.3	451.1	489
4096x6000@32p	19.4	135.0	1.7	239.3	395.4	515
4096x6000@16p	19.4	248.0	3.3	478.8	749.5	488
4096x6000@08p	19.4	476.1	6.7	957.4	1459.6	475
4096x16000@32p	51.4	360.2	1.7	637.9	1051.2	513

Performance figures are gathered in table 1 where we list the time spent by the program in the four key routines of the program at each time step, for type A and type B lattices and for a larger lattice that we plan to study in the near future. In order to measure the scaling properties of our implementation, we list data for different numbers of processors.

Some comments are in order:

- the most time consuming part of the code is routine `coll`, encompassing most of the floating point part of the computation.

- a non negligible fraction of the program is spent in the streaming part of the code. Stream does not perform any "useful" processing: it loads and stores data words from memory with a very irregular addressing pattern. The memory interface is not very efficient in this case: the measured memory bandwidth is $\sim$ 3.5Gbyte/sec, about a factor 7 lower than peak. This is the single most severe bottleneck of the computation.
- as the lattice is split in vertical tiles among the processors, the amount of data moved by neighbor processors does not depend on the number of processor for a given size of the lattice, so the time used by `comm` is independent of the number of processors. In all practical cases, the overhead is not larger than $\sim 5\%$.
- Establishing the correct boundary conditions (routine `bc`) has a fully negligible impact on performance.
- The last column in table 1 lists values of $T \times N_p/(L_x L_y)$, that is the total time spent by the program for each site of the lattice, independently of the number of processors. In a perfect scaling case this number should remain constant: we have fluctuation $\leq 10\%$ showing very good scalability for physically interesting configurations.
- If we compare the measured performance (previous point) with the theoretically derived peak (eq. (4)) we find a sustained performance in the $14 \cdots 17\%$ range: our implementation has reached a reasonable level of performance for a real-life production code. Further improvements – mainly merging stream and collision, to avoid unnecessary memory access – may allow to reach a performance level of $\sim 20\%$.

6. Overview of physics results

In this section we present preliminary results on the Rayleigh-Taylor instability and the turbulent flow in the mixing region, based on the simulations described above. The Rayleigh-Taylor (RT) instability is a hydro-dynamic unstable configuration associated to the superposition of a heavy fluid above a lighter one in a constant acceleration field. It plays an important role in several areas such as inertial-confinement fusion, supernovae explosions and many others [13]. The RT instability – studied for decades – is still an open problem. In particular, it is crucial to control the initial and late evolution of the mixing layer between the two miscible fluids; the small-scale turbulent fluctuations, their anisotropic/isotropic ratio; their dependency on the initial perturbation spectrum or on the physical dimensions of the embedding space. In many cases, especially concerning astrophysical and nuclear applications, the two fluids evolve with strong compressible and/or stratification effects, a situation which is difficult to investigate either theoretically or numerically. All this key questions calls for high spatial resolution [14], that can be tackled with QPACE.

The Lattice Boltzmann code we have developed, and its efficient implementation on QPACE make it possible to achieve state-of-the-art resolution for the Rayleigh-Taylor problem. Here, we focus on the large scale properties of the mixing layer, studying the spatio temporal evolution of a single component fluid when initially prepared on the hydrostatic unstable equilibrium, i.e. with a cold (higher density) uniform region in the top half and a hot (lower density) uniform region with in the bottom half of the cell. In our simulation the Atwood number (a dimensionless measure of the density difference in the two regions) is $A = 0.05$ (see left panel of fig. 2 for a color figure). We study the problem with a resolution up to 4096×6000 collocation points. This is the highest resolution ever reached for RT turbulence (previous important studies in this direction reached resolution up to 128×4096 in 2D [16] and 3072^3 in 3D [14]); it will allow us to assess with high accuracy both long term properties of the mixing layer evolution and small scale velocity and temperature properties. While small-scales fluctuations may be strongly different in 2D or 3D geometries, the large scale mixing layer growth is not supposed to change its qualitative evolution [15].

One of the key problem in Rayleigh-Taylor systems is to predict the growth rate of the mixing region. The latter is usually defined as an integral of the dimensionless temperature profile, $c(x,z) = (T_{max} - T(x,z))/(T_{max} - T_{min})$:

$$H(t) = \frac{1}{L_x} \int_0^{L_x} dx \int_{-L_z/2}^{L_z/2} dz M[c(x,z)] \tag{5}$$

where M is a mixing function with support only on the mixing region, for istance a tent map. Dimensionally, $H(t)$ is expected to grow quadratically, as the mixing region is accelerated by an almost constant density jump:

$$H(t) = \alpha A g t^2. \tag{6}$$

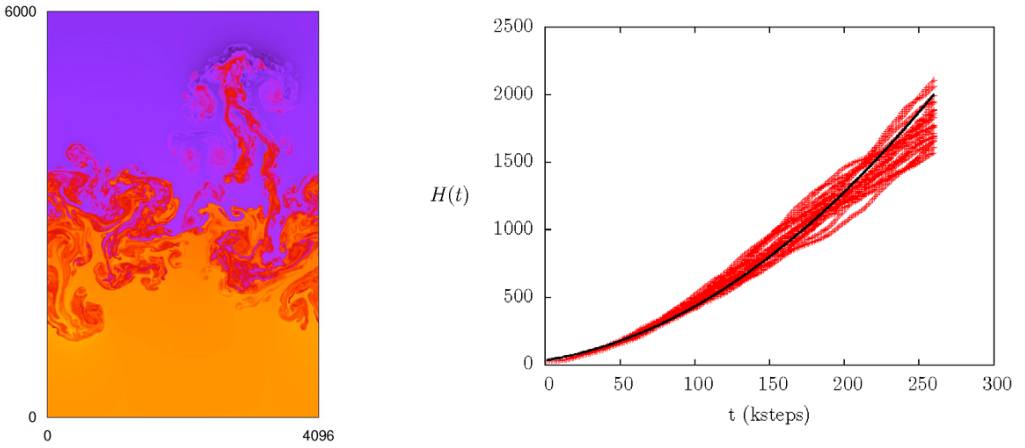

Figure 2: Left: Color coded temperature field of a 2D high resolution (4096×6000) Rayleigh-Taylor system (yellow for high temperatures, blue for low temperatures). The system is at an advanced stage in the evolution of the mixing layer with well developed (cold) descending spikes and (warm) ascending plumes. Right: Growth of the mixing layer as a function of time, $H(t)$, for 23 runs on a lattice of size 4096×6000. The black solid line is a best fit in the range $t \in [10000 : 260000]$ to the quadratic scaling formula proposed in [14], leading to the estimate $\alpha = 0.016 \pm 0.005$.

The quadratic law is expected to onset at large times, i.e. when the initial unstable growth has reached a fully non-linear character. Moreover, linear subleading terms may be present, due to the influence of the initial condition [14, 4]. For this reason, it is particularly important to have large vertical resolution. In fig. 2 (right) we show a collection of data for the evolution of $H(t)$ on a set of 23 runs with different random perturbation of the unstable initial configuration. In the same plot, we also show the best fit to the mean evolution and an estimate of the α prefactor.

7. Conclusions and outlook

In this paper we have reported on the implementation of a massively parallel version of an LBM model of 2D fluid flows, carefully optimized for the QPACE architecture. Our code has a very good parallel scaling performance and runs on each processing element at a reasonable fraction of peak performance. We have used this code for extensive simulations of a 2D thermal flow, focusing on the properties of the Rayleigh-Taylor instability, with a spatial resolution and statistics much better than previous state-of-the-art.

From the computational point of view it will become increasingly interesting to compare our results with those that can be obtained with GPUs; work is in progress to tune our algorithms for these architectures. We are particularly interested in measuring performance on recently announced architectures that strongly improve for double precision computations.

From the point of view of physics, we have presented preliminary results on the rate of growth of the mixing layer. This study leaves a few important questions open. First, what happens in presence of strong compressible effects, i.e. at increasing Atwood number. Second, what are the consequences of strong stratification, where the front is stopped by adiabatic effects. We have started to collect data to address some of these problems, performing simulations with higher spatial resolution (lattice sizes of 4096×12000).

Acknowledgment

We warmly thank the QPACE development team for support during the implementation of our code and execution of the simulations. We furthermore acknowledge access to QPACE and eQPACE during the bring-up phase of these systems.

References

[1] S. Succi Lattice Boltzmann equation for fluid dynamics and beyond, *Oxford University Press*, (2001)
[2] M. Bernaschi et al., A flexible high-performance Lattice Boltzmann GPU code for the simulations of fluid flows in complex geometries. *Concurrency and Computation: Practice and Experience* **22**, (2009) pp. 1-14.
[3] M. Stürmera et al., Fluid flow simulation on the Cell Broadband Engine using the lattice Boltzmann method. *Computers & Mathematics with Applications* **58**, (2009) pp. 1062-1070.
[4] A. Scagliarini et al. Lattice Boltzmann Methods for thermal flows: continuum limit and applications to compressible Rayleigh-Taylor systems. *Phys. Fluids* submitted (2009)
[5] M. Sbragaglia et al. Lattice Boltzmann method with self-consistent thermo-hydrodynamic equilibria. *J. Fluid Mech.* **628**, (2009) 299
[6] F. Belletti et al., QCD on the Cell Broadband Engine, PoS(LAT2007) 039.
[7] G. Goldrian et al., Quantum Chromodynamics Parallel Computing on the Cell Broadband Engine, Computing in Science & Engineering, 10 (2008) 46-54.
[8] H. Baier et al. , QPACE – a QCD parallel computer based on cell processors, arXiv:0911.2174
[9] A. Bartoloni et al. LBE simulations of Rayleigh-Benard convection on the APE100 parallel processor, Int. J. Mod. Phys. C (1993) vol. 4 (5) pp. 993-1006
[10] F. Belletti, et al., Multiphase lattice Boltzmann on the Cell Broadband Engine, Il nuovo Cimento C 32 (2009) 53-56.
[11] J.M. Buick & C.A. Greated, Gravity in a lattice Boltzmann model. *Phys. Rev E* **61**,(2000) 5307
[12] Z. Guo, C. Zheng & B. Shi. Discrete lattice effects on the forcing term in the lattice Boltzmann method, *Phys. Rev. E* **65**, 046308 (2002)
[13] D.H. Sharp. An overview of Rayleigh-Taylor instability. *Physica D* **12**, 3 (1084)
[14] W.H. Cabot & A. W. Cook. Reynolds number effects on Rayleigh-Taylor instability with possible implications for type-Ia supernovae. *Nature* **2**, 562 (2006)
[15] M. Chertkov. Phenomenology of Rayleigh-Taylor Turbulence. *Phys. Rev. Lett.* **91** 115001. (2003)
[16] A. Celani, A. Mazzino and L. Vozella. Rayleigh-Taylor turbulence in 2d. Phys. Rev. Lett. 96, 134504 (2006)