Accuracy of the lattice-Boltzmann method using the Cell processor

M. J. Harvey*
Information and Communications Technologies, Imperial College London, South Kensington, London SW7 2AZ, United Kingdom

G. De Fabritiis† and G. Giupponi‡
Computational Biochemistry and Biophysics Laboratory (GRIB-IMIM), Universitat Pompeu Fabra, Barcelona Biomedical Research Park (PRBB), C/Doctor Aiguader 88, 08003 Barcelona, Spain

(Received 19 February 2008; published 6 November 2008)

Accelerator processors like the new Cell processor are extending the traditional platforms for scientific computation, allowing orders of magnitude more floating-point operations per second (flops) compared to standard central processing units. However, they currently lack double-precision support and support for some IEEE 754 capabilities. In this work, we develop a lattice-Boltzmann (LB) code to run on the Cell processor and test the accuracy of this lattice method on this platform. We run tests for different flow topologies, boundary conditions, and Reynolds numbers in the range Re=6–350. In one case, simulation results show a reduced mass and momentum conservation compared to an equivalent double-precision LB implementation. All other cases demonstrate the utility of the Cell processor for fluid dynamics simulations. Benchmarks on two Cell-based platforms are performed, the Sony Playstation3 and the QS20/QS21 IBM blade, obtaining a speed-up factor of 7 and 21, respectively, compared to the original PC version of the code, and a conservative sustained performance of 28 gigaflops per single Cell processor. Our results suggest that choice of IEEE 754 rounding mode is possibly as important as double-precision support for this specific scientific application.

DOI: 10.1103/PhysRevE.78.056702 PACS number(s): 47.11.−j, 02.70.−c

I. INTRODUCTION

Lattice-Boltzmann (LB) modeling schemes [1,2] have emerged in the last several years as powerful methods for gaining scientific insights on a variety of complex systems. Different phenomena such as microfluidics [3], spinodal decomposition [4], rheology of amphiphilic mixtures [5], colloidal suspensions [6], and flow in porous media [7] have been profitably investigated using the LB algorithm. In fact, the discrete character of the model can be an advantage over continuum fluid dynamics models when studying interfacial phenomena, e.g., bubble pinching [8]. The LB method has also been used in multiscale models coupling molecular dynamics (MD) and hydrodynamics to study coarse-grained biomolecular systems, where the solvent dynamics is solved on a lattice [6,9–11], while the coarse-grained macromolecule dynamics is solved by molecular dynamics. While this approach permits a speed-up by two orders of magnitude for the calculation of the solvent dynamics compared to MD, it remains insufficient to study big macromolecular systems due to the computational cost of the LB domain.

Improving the performance of the LB method is therefore crucial to expand the scientific insight afforded by simulations. Hitherto, performance increase has usually come from faster versions of standard processors. However, already heat dissipation problems and memory access limitations make it very difficult to deliver the same performance gain as during the last decade. A major shift in processor design has therefore begun as a response to these problems, and recent processors are rapidly changing toward multicore architectures. For high-performance computing, however, more innovative solutions, which can be referred to as many-core architectures, have been recently implemented. The Cell processor [12], designed and developed jointly by Sony-Toshiba-IBM, is a heterogeneous multicore processor, containing eight specialized vector cores known as synergistic processing elements (SPEs) controlled by a general purpose PowerPC core. Similarly, modern graphics processing units (GPUs) have a multicore, highly multithreaded architecture and may be used for general purpose computation [13,14]. Both Cell and GPU devices achieve very high rates of floating-point arithmetic operations relative to general purpose central processing units (CPUs). However, this performance is currently limited to operations on single-precision (32-bit) data, with double-precision operations performed many times more slowly. Furthermore, for the case of the Cell processor, some aspects of the IEEE 754 standard for binary floating-point arithmetic [15] are not implemented, most significantly signaled NaN (“not-a-number”) quantities, and a limited choice of rounding modes are available. In addition, in order to achieve high performance on these devices, it is currently necessary to explicitly design (or refactor) code to accommodate the architectural complexity.

For the case of molecular dynamics, production codes for biomolecular simulations already exist for the Cell processor (CELLMD [16]) and for GPUs [17,18]. In the case of CELLMD, a sustained performance of 30 gigaflops can be achieved by using the Cell processor. Such a performance increase required a substantial change in algorithms, which had to be adapted to the Cell hardware architecture. Techniques such as multithreading, low-level memory management, and vectorized code are necessary to exploit the new design of this innovative architecture. Nonetheless, we stress here that increase in performance for MD simulations directly translates

*m.j.harvey@imperial.ac.uk
†gianni.defabritiis@upf.edu
‡giovanni.giupponi@upf.edu

into new application possibilities. As an example, we have performed the calculation of the potential of mean force for ion permeation across a transmembrane pore using CELLMD and a distributed infrastructure (PS3GRID.NET [19]) capable of hundreds of nanoseconds per day of computation [20].

The aim of this paper is to exploit the Cell processor for the case of lattice-Boltzmann simulations. We show a lattice-Boltzmann implementation on the Cell processor which correctly reproduces fluid dynamics theoretical results, even if this architecture presents at the moment some numerical precision limitations such as single precision and reduced IEEE compliance. The paper is structured as follows. In the next section we provide details of the Cell broadband engine architecture and the LB algorithm. Section III shows results for our implementation of LB algorithm on the Cell processor. Also, we report benchmarks on two different platforms that feature the Cell processor, the Playstation3 (IBM QS20/QS21 blade server), showing a speed-up of 7 (21) times. In the final section, we summarize our findings, estimate the performance of coming second-generation products of these new architectures, and discuss the scientific feedback that such platforms are likely to bring.

II. SIMULATION METHODS

A. The Cell processor

The Cell broadband engine (CBE) [12] processor comprises one PowerPC processing element (PPE) which runs the operating system and acts as a standard processor and eight independent synergistic processing elements. Each of the cores (PPE and SPEs) features a single-instruction multiple-data (SIMD) vector unit for single-precision floating-point operations. The PPE is not designed for high performance, for instance it features only 512 kbytes of internal cache compared to the several megabytes of modern processors. It is therefore crucial to use the SPEs on the computing-intensive parts of the code. Most importantly, each SPE does not access memory directly but only via asynchronous direct memory accesses which copy data from the main memory into a 256 kbyte local store memory. The local store memory is accessed by the SPE with no intermediate caching at very high speed (a latency of just a few clock cycles). This memory hierarchy is probably one of the key points for obtaining very high sustained performance on the Cell processor compared to standard processors. The Cell processor can be programmed as a multicore chip using standard ANSI C and relying on the libraries from the IBM system development kit (SDK) to handle communication, synchronization, and SIMD computation. It is also important to clarify that an existing application would run on the PPE core of the Cell processor through simple recompilation, but would not benefit from any SPE acceleration.

Currently, double precision is supported but with performance penalties, while the next release of the Cell processor (PowerXCell 8i [21]) supports double-precision calculations at a rate approaching half that of the current Cell in single precision. The Cell processor is available at present in the IBM blades (QS20/QS21) with two Cell processors per blade giving access to 16 SPEs transparently, and also in the Sony Playstation3 (PS3). The PlayStation3 can be regarded as a commodity machine as it is easy to convert a PS3 into a Cell workstation by installing GNU/Linux as the operating system.

B. The lattice-Boltzmann method

The lattice-Boltzmann method is given by the set of equations [22]

$$f_i(\mathbf{x}+\mathbf{c}_i,t+1)-f_i(\mathbf{x},t)=-\frac{1}{\tau}[f_i(\mathbf{x},t)-f_i^{\mathrm{eq}}(\mathbf{x},t)], \quad (1)$$

where $f_i(\mathbf{x},t)$ is the single-particle distribution function, indicating the probability of the fluid to have velocity $\mathbf{c}_i$ at site $\mathbf{x}$ at time step t. The index i runs over a discrete set that depends on the dimensionality and type of the chosen lattice, in our case 2-dimensional, 9-velocity vectors (D2Q9), i.e., $i=1,\ldots,9$. We choose a customary single relaxation time τ and the Bhatnagar-Gross-Krook form [23] for the collision operator, the right-hand side of Eq. (1).

In the limit of low Mach numbers, the LB equations correspond to a solution of the Navier-Stokes equation for isothermal, quasi-incompressible fluid flow. Its implementation can efficiently exploit parallel computers, as the dynamics at a point requires only information about quantities at nearest neighbor lattice sites. The local equilibrium distribution f_i^{eq} plays a fundamental role in the dynamics of the system as shown by Eq. (1). The local equilibrium distribution $f_i^{\mathrm{eq}}(\mathbf{x},t)$ is derived by imposing certain restrictions on the microscopic processes, such as explicit mass and total momentum conservation

$$f_i^{\mathrm{eq}}=\zeta_i f\left(1+\frac{3}{c_s^2}\mathbf{c}_i\cdot\mathbf{u}+\frac{4.5}{c_s^4}(\mathbf{c}_i\cdot\mathbf{u})^2-\frac{1.5}{c_s^2}u^2\right), \quad (2)$$

where $\mathbf{u}=\mathbf{u}(\mathbf{x},t)$ is the macroscopic bulk velocity of the fluid, defined as $f(\mathbf{x},t)\mathbf{u}\equiv\Sigma_i f_i(\mathbf{x},t)\mathbf{c}_i$, ζ_i are the coefficients resulting from the velocity space discretization, and c_s is the speed of sound.

We consider here a set of LB method implementations.

(1) A standard C++ implementation ($\mathrm{LB}_{\mathrm{PC}}$) of the LB method in two dimensions. No particular optimization techniques, such as unified streaming collision, have been employed in this code.

(2) A vectorized version of the code for the PPE single-instruction multiple-data instructions which is referred to as $\mathrm{LB}_{\mathrm{PPE}}$. $\mathrm{LB}_{\mathrm{PPE}}$ uses the ALTIVEC vector instructions available on the Power4 processor (the PPE in the Cell processor). A vectorial rearrangement of structure data plus some low-level programming as required by the Cell architecture were necessary for this, but no changes in the algorithm and computing techniques have been made.

(3) The Cell-fully-enabled version of the code, $\mathrm{LB}_{\mathrm{SPE}}$, which uses SIMD instructions on the SPE and a multithreaded parallelization. There is a very close correspondence between ALTIVEC vector primitives and those for the Cell SPE, making conversion of algorithms from PPE to SPE relatively straightforward. Source code of the vectorial

propagation and collision subroutines for both LB_{SPE} and LB_{PPE} is available [30].

For all codes we set lattice site $\sigma=1$ and the integration time step $dt=1$. All quantities in the rest of the paper are calculated in lattice units.

III. RESULTS

The use of single precision for the lattice-Boltzmann method is interesting because the LB method usually presents large memory footprints. Also, the simple update rules in Eq. (1) can be easily vectorized using SIMD instructions to update four sites at a time, providing a further computational advantage. We have checked different flows (boundary conditions and channel topologies) and regimes (Reynolds numbers Re). The LB algorithm conserves mass and momentum by construction, so we test how different platforms and precisions can affect this physical property. In addition, Eq. (1) is known to produce numerical instabilities and spurious oscillations in low-viscosity regimes. Accordingly, we also establish whether single-precision simulations deviate from double-precision results at small fluid viscosities (higher Re). We simulate Poiseuille flow and the relaxation of a sinusoidal wave. Analytical solutions are known for these flows, and we compare LB_{SPE} and LB_{PC} results to demonstrate the ability of LB_{SPE} to describe accurately the fluid dynamics of the system. Our reference PC runs GNU/Linux 2.6.22 Fedora Core 6 and is equipped with 2 gigabytes of random-access memory (RAM) and Intel Core 2 clocked at 2.4 GHz with 4096 kbyte cache size.

A. LB method on Cell: Precision

1. Relaxation of random velocity field

We set up a simulation starting from a randomized velocity field. The initial values f_i for each site are randomly picked out of a distribution centered on the equilibrium values as in Eq. (2), with $m=1.0$, $u_x=0.1$, and $u_y=0.0$ and with a range of $\pm10\%$ from equilibrium values. We run this system on a 128×128 lattice with periodic boundary conditions. After fewer than 2000 time steps the fluid velocity field relaxes to the constant values of $u_x=0.1$, $u_y=0.0$.

In Fig. 1 we plot the fraction of total fluid mass [Fig. 1(a)] and momentum [Fig. 1(b)] lost against simulation time. A systematic deviation from mass-momentum conservation for the vectorized and Cell-enabled versions (LB_{PPE} and LB_{SPE}) of the original code is shown. When running the vector code on the PPE, mass and momentum are not perfectly conserved, with a very small, nearly undetectable increase during the course of simulation. This is the consequence of using single precision, as we recover the same qualitative behavior using LB_{PC} compiled in single precision, even if LB_{PC} can nevertheless use the extended 80-bit precision of the x86 floating-point unit (FPU). This FPU capability makes it possible for intermediate values in a calculation to be kept at higher precision provided there are sufficient processor registers free. On the SPE, such systematic deviation is more pronounced, with a clearly detectable loss of mass or momentum. In addition, we note that the smaller the fluid viscosity, the bigger is the loss. Mass and momentum conservation are exact when the same simulation of Fig. 1 is run with our PC reference code LB_{PC} in double precision over the simulation window, independently from the fluid viscosity. We explain the lower mass-momentum conservation of LB_{SPE} with the fact that the SPE FPUs of the current version of the Cell processor support single precision (32-bit) arithmetic and provide only a simple truncation (round-to-zero) rounding mode. In fact, we recover the same behavior as that of LB_{SPE} when the simulation of Fig. 1 is repeated with LB_{PC} setting the rounding mode to round to zero (not shown). As LB_{PPE} uses 32-bit precision, round-to-nearest mode, we conclude that the biasing introduced by rounding to zero dominates the numerical error of LB_{SPE}, obscuring any bias introduced by the fixed 32-bit precision. The ALTIVEC instruction set does not support the round-to-zero mode so a direct comparison between LB_{PPE} and LB_{SPE} is not possible.

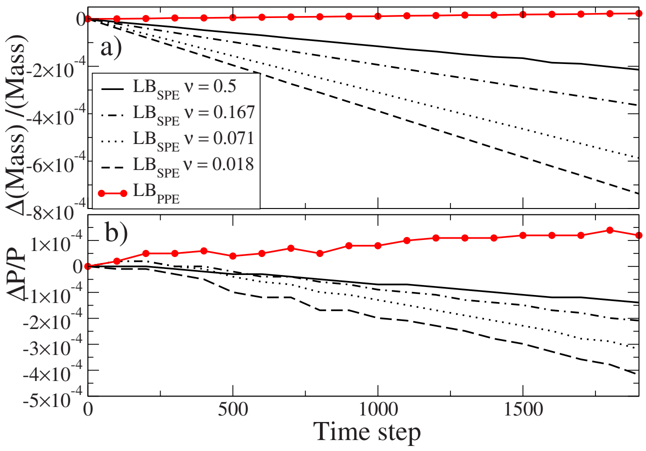

FIG. 1. (Color online) LB_{SPE} percentage of mass (a) and momentum (b) lost against time for a relaxing random velocity field using different fluid viscosities $\nu=0.018, 0.071, 0.167, 0.5$ on a 128×128 lattice. Small but systematic mass or momentum loss is obtained using LB_{SPE}. Mass or momentum loss increases as fluid viscosity decreases. Continuous line with points shows a negligible mass or momentum increase using LB_{PPE}.

We note that on the SPE the percentage of mass or momentum lost is very small ($\sim10^{-4}$). However, around 1% of the total mass or momentum can be lost in a few hundred thousands time steps, i.e., a typical production run. In the following, we show that this does not affect the predicted flow behavior of our test simulations, but we stress here the importance of a consistency check with the simulation results of double-precision standard architectures before starting production runs and scientific investigation exploiting more general LB setups and methods of a LB code on the Cell processor. Finally, the range of normalized single-precision numbers for the SPE is extended beyond the IEEE standard, but our results do not show any improvements due to this feature, as in fact was not seen in the case of biomolecular simulations [16]. The advent of the next generation Cell processor [24], capable of double-precision support, will significantly reduce present drawbacks. Nevertheless, in view of the results of LB_{SPE} we argue that, for this application, an appropriate choice of rounding mode is as important as the use of double precision.

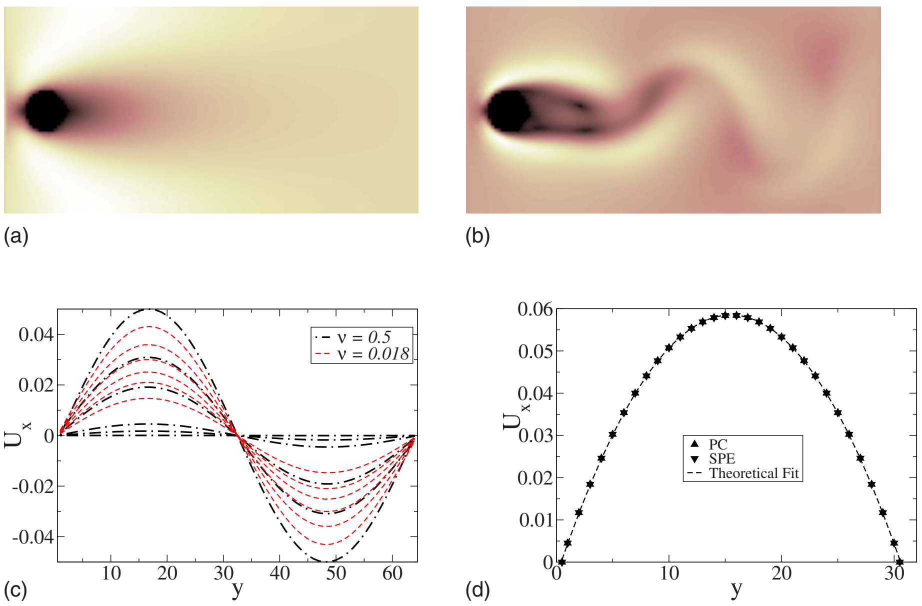

FIG. 2. (Color online) $\mathbf{u}(\mathbf{x})^2$ at $t=200\,000$ of flow past a cylinder. The slower the velocity, the darker the pixel. (a) The case of a viscous fluid (Re=12.8). (b) The beginning of vortex production (Re=355.5). (c) The velocity field profiles u_x at $L_y/2$ for two simulations of relaxing waves with $\nu=0.018$ (dashed lines) and $\nu=0.5$ at different simulation times. (d) The velocity profile u_x against y for a Poiseuille $\nu=0.5$ flow. Simulation results on the Cell processor and on a PC coincide. Also, the profiles obtained agree with the profile fit (dashed line) using Eq. (4).

2. Flow past cylinder

We run a simulation on a 128×64 lattice, placing a circular obstacle of radius 13, keeping $u_x=0.1$ constant, on the two first left lattice columns and randomly initializing the other lattice sites as above, but within 2% randomization from equilibrium values. We run 200 000-time-step simulations with fluid viscosities $\nu=0.018$ and 0.5 on a PC (single and double precision) and on the Cell processor. Standard bounce-back conditions have been used near the solid boundary. For such setup, we report exact conservation of mass and momentum, and we explain this as a consequence of fixing momentum at the boundary, which reduces the accuracy necessary to integrate Eq. (1) correctly. Figure 2 shows the squared velocity field $\mathbf{u}(\mathbf{x})^2$ at $t=200\,000$ obtained for the two values of fluid viscosity. Figure 2(a) shows the case of $\nu=0.5$, which corresponds to a Reynolds number $\mathrm{Re}=u_xL_y/\nu=12.8$. For such a low Re the flow is viscous, and the figure correctly captures this characteristic. Figure 2(b) shows the case of $\nu=0.018$ (Re=355.5). For such a high Reynolds number the flow presents a velocity field with vortex formation. No difference between the velocity fields of simulation runs on PC and Cell has been detected.

B. LB on Cell: Physical accuracy

1. Wave relaxation

The linearized solution for the velocity field for a small initial perturbation $\mathbf{u}=(u_0\sin(ky),0)$ is given by

$$u_x(t)=u_0\sin(ky)\exp(-\nu k^2 t). \tag{3}$$

We set up a 64×64 simulation box with this sinusoidal initial velocity profile with $u_0=0.1$ and $k=2\pi/L_y$. In Fig. 2(c), we plot velocity field u_x profiles at $x=L_y/2$ for two simulations with $\nu=0.018$ and 0.5 at different simulation times. For a more compelling inspection of the numerical accuracy of $\mathrm{LB}_{\mathrm{SPE}}$, we compare the effective (or numerical) shear viscosity ν_{eff} obtained by fitting the simulation results with Eq. (3) with the input value ν_{input}, plotting the percentage error $(\nu_{\mathrm{input}}-\nu_{\mathrm{eff}})/\nu_{\mathrm{input}}$ in Fig. 3(a). The input kinematic viscosity is calculated as $\nu_{\mathrm{input}}=(\tau-0.5)/c_s^2$. Figure 3(a) shows that the difference between input and effective viscosities is less than 0.02%. In addition (not shown), the difference between effective viscosities calculated using $\mathrm{LB}_{\mathrm{SPE}}$ and $\mathrm{LB}_{\mathrm{PC}}$ (double precision) is negligible.

2. Poiseuille flow

We set up boundary conditions for flow in a pipe (Poiseuille flow) on a 64×32 lattice and run simulations for 200 000 time steps with ν ranging from 0.07 to 0.5. The no-slip, steady state velocity profile for Poiseuille flow reads [25]

$$u_x=-\frac{1}{2\eta}\frac{\partial p}{\partial x}y(y-L_y) \tag{4}$$

where $\partial p/\partial x$ is the gradient pressure, $\eta=\rho\nu$ the dynamic viscosity, L_y the channel width, and ν the fluid viscosity. We

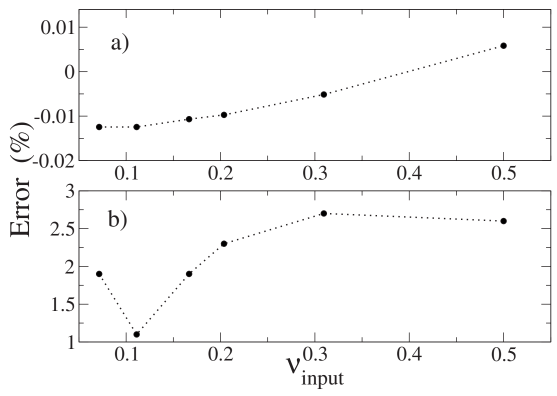

FIG. 3. (a) Percentage error $(\nu_{\mathrm{input}}-\nu_{\mathrm{eff}})/\nu_{\mathrm{input}}$ depending on input fluid viscosity ν_{input} when simulating wave relaxation in a 64×64 box. Input and effective viscosities differ by less than 0.02%. (b) Percentage error $(\nu_{\mathrm{input}}-\nu_{\mathrm{eff}})/\nu_{\mathrm{input}}$ depending on input fluid viscosity ν_{input} obtained by simulating Poiseuille flow on a 64×32 lattice. ν_{eff} agrees well with ν_{input}.

TABLE I. Mega lattice site updates per second (MLUPS). PS3 can at most use six SPEs at the same time, whereas the IBM QS20/QS21 can use 16 SPEs concurrently.

	PC	LB_{SPE} 1 SPE	LB_{SPE} 6 SPEs (PS3)	LB_{SPE} 16 SPEs	6 SPEs Ref. [26]	16 SPEs Ref. [26]
MLUPS	3.3	4.5	26.4	72.6	95	200

plot in Fig. 2(d) the velocity profiles obtained by running the $\nu=0.5$ simulation on LB_{SPE} and LB_{PC} (double precision). We note that results for the two different architectures coincide and the prescribed parabolic behavior is recovered. We also fit [dashed lines in Fig. 2(d)] the obtained velocity profiles to Eq. (4) in order to recover the numerical viscosity. The effective viscosities obtained by fitting simulation results for LB_{SPE} and LB_{PC} are equivalent. We compare the numerical viscosity obtained by simulating Poiseuille flow with the input viscosity in Fig. 3(b), where we plot the percentage error $(\nu_{\text{input}}-\nu_{\text{eff}})/\nu_{\text{input}}$. ν_{eff} agrees well with ν_{input}, as the percentage error is within 3% for all values of viscosity tested.

C. Performance

We run simulations on the different flow topologies studied above to benchmark the performance of LB_{SPE}, our implementation of a single-phase lattice-Boltzmann scheme on the Cell processor. The Cell processor currently features on the Sony PlayStation3 and the QS20/QS21 IBM blade server. The differences between the two architectures are substantial, as IBM blade servers comprise two Cell processor chips. The eight SPEs of each single processor can be used transparently by a program, allowing therefore 16 SPEs to be used at the same time for computation. The PS3, instead, currently by design grants access to only six of the eight SPEs, one being reserved for running a virtualization engine and the other one being disabled. These and other hardware differences are reflected in a very different price, making the IBM server the best (fastest) platform in terms of raw performance (gigaflops). However, when the performance per dollar is compared, the PS3 results in the best architecture (see below).

A quantitative unit of reference for LB simulation performance is the lattice site updates per second (LUPS), which measures the speed (as useful to the scientist end user) of a code running on an architecture. We report in Table I the number of LUPS obtained running 500 time steps of a 512×256 flow-past-cylinder simulation on a PS3 and on an IBM QS20/QS21 blade server and results obtained in Ref. [26]. We report that LB_{SPE} on a single SPE already runs considerably faster than the equivalent code on our reference PC. In fact, LB_{SPE} using a single SPE runs at $\sim$4.5 megaLUPS (MLUPS) on the Cell processor, without differences between PS3 and QS20/QS21 and $\sim$73 MLUPS on a dual-Cell IBM QS20/QS21 server, whereas a PC single-precision implementation reaches $\sim$3.3 MLUPS on our reference PC. We mention here that a LB code running at a few MLUPS can be regarded as an efficient single-phase LB code [27]. In terms of raw performance, LB_{SPE} on the Cell processor delivers sustained 28 gigaflops when run on eight SPEs. This estimate is obtained by counting only the floating-point operations in Eq. (1), which correspond to 772 flops per lattice site. We also recover here the same quantitative results when running our code only on the PPE and comparing the performance obtained with results obtained for one SPE. As already noted [16], the PPE runs decidedly more slowly (a factor of $\sim$2 more slowly compared to our reference PC), and this is due to the fact that PPE architecture is not optimized for number crunching, but rather for coordinating tasks and job scheduling [28]. Finally, a double-precision LB_{SPE} benchmark has not been carried out as executing code in double precision in the current Cell version is at least an order of magnitude slower than single precision [28].

Linear scaling is obtained when LB_{SPE} is run on the PS3 and IBM QS20/QS21 blade server. This is due to the fact that LB_{SPE} is clearly computation bound. In fact, each SPE uses one-tenth of the clock cycles fetching data compared to crunching numbers, as can be determined by detailed code profiling with IBM SDK tools. This permits us to obtain a speed-up factor of $\sim$7 for the PS3 and $\sim$21 for QS20/QS21 compared to the original, PC version of the LB code compiled using Intel C compiler (ICC) 9.1, as ICC provides some automatic vectorization and parallelization of the code. Stuermer *et al.* [26], starting from a heavily optimized LB code, obtained more than $\sim10\times10^7$ LUPS on a single Cell processor and up to $\sim20\times10^7$ on an IBM QS20/QS21 blade server, showing that careful memory layout and deep algorithmic optimization can add a further 3–4 times speed-up.

IV. CONCLUSIONS

In this paper, we have evaluated the use of the Sony-Toshiba-IBM Cell processor for fluid dynamics simulations using the lattice-Boltzmann method. We modified an already existing LB code, creating a program capable of running on the Cell processor. We simulated with LB_{SPE} different flows changing the topology, boundary conditions, and fluid viscosity such as flow past a circular object, relaxation of a transverse fluid velocity field, or a random velocity field and Poiseuille flow. Our results show that the single-precision and IEEE 754 rounding-mode limitations of the current Cell release can in general affect mass and momentum conservation properties. However, we demonstrated the correct description of different fluid regimes equivalent to a standard LB implementation covering a wide range of Reynolds numbers $Re=6-350$ and boundary conditions. We also performed benchmark simulations with LB_{SPE} on the two different platforms featuring the Cell processor, i.e., the Sony Playstation3 and the IBM QS20/QS21 blade server. The benchmarks are calculated in lattice site updates per second, therefore effectively estimating the potential scientific gain that can be achieved from the Cell processor. The benchmark results showed peak performances corresponding to that of an efficient LB code running in parallel on a small cluster of tens of PCs. Also, $\mathrm{LB}_{\mathrm{SPE}}$ on the Cell processor delivers a sustained raw performance of 28 gigaflops when run on eight SPEs. We also calculated the LUPS per dollar provided by such platforms, with the 50 000 LUPS/dollar for the PS3 outperforming not only the 3500 LUPS/dollar of an IBM QS20/QS21 server, but also a 17 000 LUPS/dollar estimate of quadcore performance, assuming that a quadcore machine costs \$1000 and delivers four times the performance of a single-core processor. We speculate that for the case of three-dimensional lattices performance might be limited by the RAM memory size, as the increased number of degrees of freedom per site would reduce the total number of lattice sites that can be stored in RAM. For the case of lattice-Boltzmann simulation on a PS3 (256 megabytes RAM size), a $128 \times 128 \times 128$ lattice would require the use of slow swap memory. This is not a limitation for IBM QS20/QS21 servers, which feature 2 gigabytes of RAM.

In our view, although support for double precision is highly desirable for scientific computation, of considerable importance is a greater level of IEEE compliance even for single-precision arithmetic. In any case, the Cell processor is already playing a primary role in next-generation high-performance computing, such as the first PetaFlops class supercomputer, the RoadRunner [29], that features 12 960 PowerXCell 8i processors.

ACKNOWLEDGMENTS

We gratefully acknowledge support from Barcelona Supercomputing Center (http://www.bsc.es) for giving access to IBM dual-Cell blades, and Acellera Ltd. (http://www.acellera.com). G.G. acknowledges the Aneurist project (http://www.aneurist.org) for financial support. G.D.F. acknowledges the Ramon y Cajal scheme. We also thank Ignacio Pagonabarraga, Sauro Succi, Jens Harting, and Massimo Bernaschi for a critical reading of the paper.

[1] S. Succi, *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond* (Oxford University Press, Oxford, 2001).
[2] R. Benzi, S. Succi, and M. Vergassola, Phys. Rep. **222**, 145 (1992).
[3] M. Sbragaglia, R. Benzi, L. Biferale, S. Succi, and F. Toschi, Phys. Rev. Lett. **97**, 204503 (2006).
[4] V. M. Kendon, J. C. Desplat, P. Bladon, and M. E. Cates, Phys. Rev. Lett. **83**, 576 (1999).
[5] G. Giupponi, J. Harting, and P. Coveney, Europhys. Lett. **73**, 533 (2006).
[6] A. J. C. Ladd, Phys. Rev. Lett. **70**, 1339 (1993).
[7] A. K. Gunstensen and D. H. Rothman, J. Geophys. Res. **98**, 6431 (1993).
[8] M. E. Cates, J. C. Desplat, P. Stansell, A. J. Wagner, K. Stratford, R. Adhikari, and I. Pagonabarraga, Philos. Trans. R. Soc. London, Ser. A **1833**, 1917 (2005).
[9] P. Ahlrichs and B. Dünweg, J. Chem. Phys. **111**, 8225 (1999).
[10] G. Giupponi, G. De Fabritiis, and P. Coveney, J. Chem. Phys. **126**, 154903 (2007).
[11] O. B. Usta, A. J. C. Ladd, and J. E. Butler, J. Chem. Phys. **122**, 094902 (2005).
[12] http://www.research.ibm.com/cell/
[13] Complete Unified Device Architecture, NVidia, http://developer.nvidia.com/object/cuda.html
[14] J. Tölke (unpublished).
[15] IEEE 754: IEEE Standard for Binary Floating-Point Arithmetic 754–1985. Institute of Electrical and Electronics Engineers, 1985.
[16] G. De Fabritiis, Comput. Phys. Commun. **176**, 660 (2007).
[17] J. A. van Meel, A. Arnold, D. Frenkel, S. F. Portegies Zwart, and R. G. Belleman, Mol. Simul. **34**, 259 (2008).
[18] M. J. Harvey, G. Giupponi, and G. De Fabritiis, AceMD: Accelerated Molecular Dynamics for the Microsecond Time Scale (to be published).
[19] http://www.ps3grid.net
[20] M. Harvey, G. Giupponi, J. Villa-Freixa, and G. De Fabritiis, *PS3GRID. NET: Building a distributed supercomputer using the PlayStation 3* (Distributed and Grid Computing—Science Made Transparent for Everyone. Principles, Applications and Supporting Communities, 2007) (Tektum Publishers, Berlin, 2008).
[21] http://www-03.ibm.com/technology/cell/pdf/PowerXCell_PB_7May2008_pub.pdf
[22] P. J. Higuera, S. Succi, and R. Benzi, Europhys. Lett. **9**, 345 (1989).
[23] P. L. Bhatnagar, E. P. Gross, and M. Krook, Phys. Rev. **94**, 511 (1954).
[24] http://www.fz-juelich.de/jsc/datapool/cell/Cell_Hardware_and_Software_Roadmap.pdf
[25] L. D. Landau and E. M. Lifshitz, *Fluid Mechanics* (Pergamon Press, Oxford, 1959).
[26] M. Stürmer, J. Götz, G. Richter, and U. Rüde, Lehrstuhl für Informatik 10 (Systemsimulation), Friedrich-Alexander-Universitat Erlangen-Nürnberg, Technical Report No. 07–9, 2007 (unpublished).
[27] G. Wellein, T. Zeiser, G. Hagger, and S. Donath, Comput. Fluids **35**, 910 (2006).
[28] http://www.ibm.com/developerworks/power/cell/
[29] http://www.lanl.gov/orgs/hpc/roadrunner/index.shtml
[30] http://www.multiscalelab.org/CellLB