Multiscale simulation of cardiovascular flows on the IBM Blue Gene/P: full heart-circulation system at near red-blood cell resolution

Amanda Peters*, Simone Melchionna*†, Efthimios Kaxiras*†, Jonas Lätt†, Joy Sircar*,
Massimo Bernaschi‡, Mauro Bisson‡, Sauro Succi‡*§
* Department of Physics and School of Engineering and Applied Sciences,
Harvard University, Cambridge, MA, USA
†Institute of Material Sciences and Engineering
École Polytechnique Fédérale de Lausanne, Switzerland
‡Istituto Applicazioni Calcolo,
Consiglio Nazionale delle Ricerche, Rome, Italy
§ Freiburg Institute for Advanced Studies,
Freiburg , Germany

***Abstract*—We present the first large-scale simulation of blood flow in the coronary artieries and other vessels supplying blood to the heart muscle, with a realistic description of human arterial geometry at spatial resolutions from centimeters down to 10 microns (near the size of red blood cells). This multiscale simulation resolves the fluid into a billion volume units, embedded in a bounding space of 300 billion voxels, coupled with the concurrent motion of 300 million red blood cells, which interact with one another and with the surrounding fluid. The level of detail is sufficient to describe phenomena of potential physiological and clinical significance, such as the development of atherosclerotic plaques. The simulation achieves excellent scalability on up to $294,912$ Blue Gene/P computational cores.**

I. Introduction

Accurate and reliable modeling of blood flows in the human cardiovascular system has the potential to improve understanding of cardiovascular diseases, which are the most common cause of death in Western countries. But building a detailed, realistic model of hemodynamics is a formidable computational challenge. The simulation must combine the motion of the fluids, the intricate geometry of the blood vessels, continual changes in flow and pressure driven by the heartbeat, and finally the behavior of red and white blood cells and other suspended bodies, such as platelets and lipids.

Large-scale hemodynamic simulations have made substantial progress in recent years [1], [2], [3], [4], [5], but until now the coupling of fluid dynamics with the motion of blood cells and other suspended bodies in vessels with realistic shapes and sizes has remained beyond reach. Typically, coupled fluid-particle simulations are confined to microscale vessels, such as capillaries and venules, bearing no notion of the global geometry of the problem [6], [7], [8]. Yet the global geometry has significant effects on local circulation patterns, most notably the shear stress at arterial walls. Wall shear stress is a recognized trigger for the complex biomechanical events that can lead to atherosclerotic pathologies. Currently it is not possible to measure this shear stress in vivo. These simulations are a necessary precursor to the development of accurate and reliable hemodynamic simulations that can predict this critical parameter in the progression of cardiovascular disease.

In this work we present the first multiscale simulation of cardiovascular flows in realistic human arterial geometries derived from Computed Tomography Angiography (CTA) data. The simulation covers the entire heart circulation system, the network of arteries and arterioles that supply blood to the heart muscle, with a spatial resolution extending from 5 cm down to $10\,\mu$m.

The simulations involve up to a billion fluid nodes, embedded in a bounding space of about a three hundred billion voxels, with 10-300 million suspended bodies. They are performed with the multiphysics code MUPHY (MUlti PHYsics/multiscale), which couples Lattice Boltzmann methods for the fluid flow and a Molecular Dynamics treatment of the suspended bodies [9], [10]. The simulation achieves an aggregate performance in excess of 60 teraflops, with a parallel efficiency of more than 60 percent on a full $294,912$-processor Blue Gene/P configuration.

Our work presents a number of unique features, both at the level of high-performance computing technology and in terms of physical/computational modeling. The extremely complicated conditions that are implicit to irregular geometries require that the workload be evenly distributed across the pool of as many as $294,912$ computational nodes of the Blue Gene/P supercomputer. The formidable graph-partitioning problem, even at the mere level of the fluid computation, cannot be overestimated. On top of this, our multiphysics/scale application adds the further constraint of keeping a good workload balance also across the Molecular Dynamics (MD) sector of the simulation. To the best of our knowledge, the latter issue has never been tackled before in any MD simulation. Indeed, even top-ranking (Gordon-Bell winning) multi-billion MD simulations

SC10 November 2010, New Orleans, Louisiana, USA 978-1- 4244-7558-2/10/$26.00

are invariably performed in idealized geometries, cubes or regular boxes [12]. Similarly, multi-billion node simulations of, say, biofluid turbulence are indeed available, but only in the same ideal geometries mentioned above. Complex and large-scale geometries, such as the one considered here, are rare in the literature [13]. By leveraging large-scale parallel architectures, this work demonstrates a simulation of cardiovascular flow of unprecedented scale in a geometry from real patient data and with blood flow at physiological hematocrit values for the length of a full heartbeat. In this paper, we introduce our treatment of red blood cells as extended structure (i.e. not as point sources), our handling of highly irregular geometries via topology driven graph partitioning, and an efficient MD load balancing scheme.

II. Multiscale hemodynamics

Our approach is based on efficient and accurate algorithms capable of handling the requirements of the diverse computational entities and associated scales. The numerical framework is handled by the software *MUPHY* developed by our group in recent years [9]. The approach is genuinely multiphysics, as it combines different levels of the description of matter, continuum hydrokinetic fluids for the dynamics of blood plasma and individual particles for the representation of red blood cells and other minority suspended species. The method is also multiscale, since fluid and particles are advanced concurrently and the exchange of information is computed on-the-fly. On general grounds, kinetic theory provides the conceptual framework to bridge micro and macroscales, and the Lattice Boltzmann (LB) method is extremely well suited for the numerical solution. LB is a minimal form of the kinetic Boltzmann equation, based on the collective dynamics of fictitious particles, representing a local ensemble of molecules moving on the nodes of a regular lattice [20]. The dynamics of such particles reproduces hydrodynamics in the continuum limit, when the molecular mean free path is much shorter than typical macroscopic scales. The LB method is a low-Mach, weakly-compressible fluid solver and presents several advantages for the practical implementation in complex geometries over Navier-Stokes solvers.

First, in the kinetic LB formalism, information always travels along the straight lines defined by a set of discrete speeds $\{\mathbf{c}_p\}$. This represents a major advantage as compared with the hydrodynamic representation, whereby any physical quantity, including the flow field itself, is transported by the local flow velocity $\mathbf{u}(\mathbf{x},t)$, a highly heterogeneous vector field, especially in the presence of complex real-world geometries such as those involved in cardiovascular hemodynamics. The very same feature also reflects into a much handier treatment of complex boundary conditions (intersections with straight lines). This comes at the expense of numerical accuracy, which usually degrades to first order, due to the staircase representation of arbitrarily shaped boundaries. This weakness is, however, strongly mitigated by two compensating effects: first, the shear stress is available *locally*, as a linear combination of the discrete populations sitting at each given cell. This relieves LB from the burden of computing spatial derivatives of the velocity field at the boundaries, which is an accuracy-threatening procedure also in Navier-Stokes solvers. Second, while it is true that staircase boundaries compromise second-order accuracy, it is also true that a favorable prefactor (due again to straight trajectories) secures significant error reduction by increasing resolution. In practice, wall shear stress is found to converge to acceptable levels of accuracy at grid resolutions below 20 microns. Another favorable feature of LB is that, like the stress tensor, fluid pressure is available locally, with no need of resorting to any expensive Poisson solver. However, the most compelling asset of LB rests with its outstanding amenability to parallel computing, *even in complex geometries*. The present work represents a major testimonial to this asset: here we prove excellent scalability on up to $\sim 300K$ cores on a real-world complex geometry directly derived from medical data.

In LB the basic quantity is $f_p(\mathbf{x},t)$, representing the probability of finding, at mesh location $\mathbf{x}$ and at time t, a "fluid particle" traveling with discrete speed $\mathbf{c}_p$. "Fluid particles" represent the collective motion of a group of physical particles (often referred to as populations). We employ the common three-dimensional 19-speed cubic lattice (D3Q19) with mesh spacing Δx, where the discrete velocities $\mathbf{c}_p$ connect mesh points to first and second neighbors. The fluid populations are advanced in a timestep Δt through the following evolution equation

$$f_p(\mathbf{x}+\mathbf{c}_p\Delta t, t+\Delta t) = \\ f_p(\mathbf{x},t) - \omega\Delta t(f_p - f_p^{eq})(\mathbf{x},t) + \Delta f_p(\mathbf{x},t) \tag{1}$$

The right hand side of Eq. (2) represents the effect of fluid-fluid molecular collisions, through a relaxation towards a local equilibrium, $f_p^{eq} = w_p\rho\left[1+\frac{\mathbf{u}\cdot\mathbf{c}_p}{c_s^2}+\frac{\mathbf{uu}:(\mathbf{c}_p\mathbf{c}_p-c_s^2\mathbf{I})}{2c_s^4}\right]$, a second-order expansion in the fluid velocity of a local Maxwellian with density ρ and speed $\mathbf{u}$. In addition, $c_s = 1/\sqrt{3}$ is the speed of sound, w_p is a set of weights normalized to unity, and $\mathbf{I}$ is the unit tensor in Cartesian space. The relaxation frequency ω controls the kinematic viscosity of the fluid, $\nu = c_s^2\Delta t\left(\frac{1}{\omega}-\frac{1}{2}\right)$. The kinetic moments of the discrete populations f_p provide the local mass density $\rho(\mathbf{x},t) = \sum_p f_p(\mathbf{x},t)$ and mass current $\rho\mathbf{u}(\mathbf{x},t) = \sum_p \mathbf{c}_p f_p(\mathbf{x},t)$. Finally, the last term in eq. (2) represents the coupling between fluid and suspended bodies. This is given by

$$\Delta f_p(\mathbf{x},t) = -w_p\Delta t\left[\frac{\mathbf{G}\cdot\mathbf{c}_p}{c_s^2} + \frac{(\mathbf{G}\cdot\mathbf{c}_p)(\mathbf{u}\cdot\mathbf{c}_p) - c_s^2\mathbf{G}\cdot\mathbf{u}}{c_s^4}\right] \tag{2}$$

where $\mathbf{G}$ is a forcing term containing the translational and rotational exchange of momentum induced by N moving red blood cells at position $\{\mathbf{R}\}$. The forcing term is smeared over a region made of 32 mesh points around each RBC and with ellipsoidal shape. The drag force acting on particles is modelled as

$$\mathbf{F}_i^D(\mathbf{R}_i) = -\gamma^T(\mathbf{V}_i - \tilde{\mathbf{u}}) \tag{3}$$

and the torque is

$$\mathbf{T}_i^D(\mathbf{R}_i) = -\gamma^R \left(\mathbf{\Omega}_i - \tilde{\mathbf{\Omega}} \right) \tag{4}$$

with $\{\mathbf{V}_i\}$ and $\{\mathbf{\Omega}_i\}$ being the RBC velocities and angular velocities, and with $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{\Omega}}$ the fluid velocity and vorticity fields, smeared over the same ellipsoidal region occupied by a RBC. This smearing is achieved through an envelope function similar to the one used in the Immersed Boundary method [14], which takes into account the finite extent of the particles by means of a smooth interaction. The constants γ^T and γ^R are translational and rotational coupling coefficients of RBCs represented as oblate ellipsoids in a hydrodynamic environment.

Due to the finite extent representation of an RBC, the hydrodynamic size of RBC is smaller than the smearing region covered by the particle, with the RBC effective size depending on the strength of the coupling coefficients γ^R and γ^T. By varying these coefficients, and matching the hydrodynamic volume with the GB exclusion volume, the effective extension of a RBC covers ~ 1 lattice cell. This corresponds to a hematocrit level of 1%. To reach a more physiological level of $30 - 45\%$ about 300 million RBCs are required.

Here a compromise between physical fidelity and computational efficiency must be taken. Indeed, recent studies [15] indicate that the minimum number of degrees of freedom required for a quantitative description of RBC dynamics in a fluid flow, including deformability, is of the order of hundreds to thousands. This is far too much for a viable fluid-particle coupling at large-scales. As a result, an intermediate strategy has been developed, whereby RBCs are treated as rigid ellipsoidal bodies (six degrees of freedom) interacting with each other through custom potentials, and with the surrounding fluid (the blood plasma) via tensorial mobility coefficients, accounting for the anisotropic drag experienced by the RBC along and across the local fluid direction of motion. Such an intermediate strategy permits to capture the essential features of the complex behavior of the RBCs and their impact on the macroscopic blood rheology, at a very moderate computational cost. The represented behavior accounts not only for the translational and rotational motion of the RBCs, but also for their mutual interaction, reproducing aggregation patterns of RBCs and their impact on the overall behavior of the blood flow.

From an algorithmic point of view, this approach scales linearly with the number of RBCs, thanks to the fact that the solvent-mediated RBC-RBC interactions are entirely local and explicit. This stands in marked contrast with consolidated approaches, based on Brownian dynamics, which rely upon a non-local Green function representation of the Oseen tensor and consequently can only attain $N \log N$ scaling with the number of RBCs by resorting to highly sophisticated procedures. The strategy described here, which is entirely new and still unpublished [16], makes therefore a particularly efficient use of the invested computational resources. In particular, as typical of LB applications, it provides a very economical algorithmic representation of fairly complex physical phenomena. In the present implementation, we neglect additional torques arising from coupling with the elongational component of the flow pattern and tank treading of the RBCs.

Mechanical hard core forces prevent contacts between RBCs. The RBC-RBC interactions are pairwise and modelled via the Gay-Berne potential [24], reading

$$u_{ij}^{GB}(q_{ij}) = 4\epsilon(q_{ij}) \times \left[\left(\frac{\sigma_0}{R_{ij} - \sigma(q_{ij}) + \sigma_0} \right)^{12} - \left(\frac{\sigma_0}{R_{ij} - \sigma(q_{ij}) + \sigma_0} \right)^{6} \right] \tag{5}$$

where $q_{ij} \equiv (R_{ij}, \hat{\mathbf{u}}_i, \hat{\mathbf{u}}_j)$ and with R_{ij} being the relative distance, $\hat{\mathbf{u}}_i$ and $\hat{\mathbf{u}}_j$ are the principal directions of the i-th and j-th ellipsoids, with $\epsilon(q_{ij})$ and $\sigma(q_{ij})$ being functions with lengthy expressions reported in ref. [24].

The potential u_{ij}^{GB} is set to zero beyond a orientation-dependent cut-off given by the condition

$$\left(\frac{\sigma_0}{R_{ij} - \sigma(q_{ij}) + \sigma_0} \right)^{6} > 0 \tag{6}$$

to retain the repulsive component of the potential only. The rigid body dynamics of the suspended bodies is propagated in time via a second-order accurate timestepping algorithm [30], properly modified to handle fluid-particle forces and torques.

III. Geometry acquisition and mesh-generation

The global geometry of the problem used for the present simulations is obtained from CTA scans of the coronary arterial system of a real patient. Data acquisition was performed by a 320×0.5 mm CTA scanner (Toshiba) and subsequently segmented into a stack of two-dimensional contours at a nominal resolution of 0.5 mm. The slice contours, each consisting of 256 points, are oversampled along the axial distance down to a slice-to-slice separation of $12.5\,\mu$m and further smoothed out by appropriate interpolators. The resulting multi-branched geometrical structure is finally mapped into the Cartesian LB lattice, ready for the simulation. Full details can be found in [10].

IV. Initial and Boundary conditions

Fluid boundary conditions are set up as follows. At the inlet, a uniform flow profile with prescribed velocity is imposed, and at the outlet ports a zero pressure difference from the inlet is maintained. The flow-pressure inflow/outflow conditions are implemented via the Zou-He method to set up the LB populations in the proper way [21]. At rigid walls, a standard mid-way bounce-back rule is applied to impose no-slip flow conditions.

The fluid flow is initialized with zero speed and constant density across the entire domain. Particles are seeded at random positions and orientations, and with null linear and angular velocity. In flow conditions, RBC that exit from the outlet ports are reinjected in the inlet port in order to maintain a constant total hematocrit. The injected RBC have velocity given by the imposed inlet velocity, random orientation and

zero angular velocity. The RBCs are repelled by the wall via a GB pairwise potential acting between a RBC ellipsoid and a spherical particle positioned on a wall mesh node.

V. Code Features

The *MUPHY* (MUlti PHYsics/multiscale) code is written in Fortran 90 and uses MPI for the parallelization. To handle in a flexible and efficient way any complex geometry, *MUPHY* makes use of an indirect addressing scheme that has been described along with other main features of the code in [9]. We showed in the same paper that the penalty introduced by the indirect addressing scheme for the cases of regular geometries is very limited ($\sim 5\%$ of the execution time) and, as a matter of fact, we used the code also for problems, such as bio-polymer translocation in which the geometry is trivial (a regular box). Originally developed for the IBM Blue Gene/L system [25] *MUPHY* has been recently ported to heterogeneous clusters of CPUs and Graphics Processing Units (GPU), using the CUDA software environment, showing excellent results [26]. For the present work we employ the latest generation of the IBM Blue Gene system whose main features may be summarized as follows:

- quad SMP processor chip per node with 2GB of memory per node.
- system-on-a-chip design with superscalar 850 MHz PowerPC 440 cores;
- a large number of cores (scalable up to at least 294,912);
- three-dimensional torus interconnect with auxiliary networks for global communications, I/O, and management;
- lightweight, Unix-like OS per node for minimum system overhead.

The first versions of the LB component of the code used the "fusion" of the collision and streaming steps in a single loop. This technique, by now standard in all high-performance LB codes, significantly reduces data traffic between main memory and cache/registers of the processor, since there is only one read and one store of all LB populations at each time step. However most implementations of the fused update resort to a "double buffer" to store the LB populations. The double buffer avoids the mixing of old and new data during the non-local streaming step that would be a source of inconsistency but, obviously, doubles the memory required for the LB populations. A recent enhancement to *MUPHY* is the implementation of the "single buffer" mechanism for the Lattice Boltzmann update through an adaptation of the so-called swap algorithm [27]. In this algorithm, the particle populations are rearranged after collision. This results in a memory layout in which, as a given population is copied to a neighboring cell during the streaming step, it simply exchanges its memory location with a neighbor-node population. Thus, the copy operations in the streaming step are replaced by exchange operations, or variable swaps, as suggested by the name of the algorithm. Given that no information is lost during the swap operation, the algorithm handles the streaming phase without the need for temporary buffers. This point becomes clear by looking at Fig. 1, where four nodes of a one-dimensional Lattice Boltzmann mesh, with just two populations per node, perform a single global collision-streaming cycle, carrying them from a discrete time step t to $t + \Delta t$. Right after the mesh populations have collided, the algorithm rearranges the data locally by exchanging the two populations (in a higher-dimensional case, all local populations are exchanged with the population corresponding to the opposite direction), as indicated by the keyword "swap" on the figure. Then, as soon as the neighboring node also reaches its post-collision state, a non-local exchange operation, indicated by the keyword "exchange", is performed in lieu of the streaming step.

In a typical execution of a *MUPHY* program, memory is used mainly for the storage of the particle populations and the connectivity list. Thus the swap algorithm reduces the overall memory needs by one-third by avoiding a duplication of memory for the particle populations. Furthermore, this approach leads to a sensible performance improvement, because the program becomes more cache efficient as it holds the populations and connectivity matrix in a smaller memory space.

The geometry gathered from the CTA data that was used in the runs reported in section VI is highly irregular, as shown in Figure 4, and its partitioning among the available processors represents a major challenge in itself. Several domain decomposition strategies for irregular lattices already exist, as described in [28]. In particular, state-of-the-art techniques like those represented by multilevel $k-$way partitioning schemes can be used for irregular geometries. However, when either the size of the mesh or the number of partitions increases to critical values (in our case, the figures are ~ 1 billion nodes for the mesh and $\sim$ 300,000 partitions) most of the widely used tools simply fail, meaning that they are unable to manage the problem (that is much worse, of course, than producing a sub-optimal solution). For instance, the well-known tool for graph-partitioning *METIS* [22], even in its parallel version, requires the allocation on each processor of a block of memory equal in size to the square of the number of partitions. On the other hand, preliminary tests showed that a naive partitioning based only on a balanced number of mesh nodes on each processor produced a very poor load balancing.

We wish to emphasize that the graph-partitioning strategy is entirely topology-driven, i.e. it proceeds based on the input provided by the local connectivity supplied by the Lattice Boltzmann grid (18 neighbors, uniformly across the entire computational domain) with no information on the global geometry of the problem.

Finally, we found a very effective solution by using *PT-SCOTCH*, the parallel version of the *SCOTCH* graph/mesh partitioning tool [19]. One of the interesting features of *SCOTCH* is that its running time is linear in the number of edges of the source graph, and logarithmic in the number of vertices of the target graph for mapping computations. Moreover, a test carried out on a smaller case ($\sim 20,000,000$ mesh nodes partitioned among $1,024$ processors) showed that *SCOTCH* produces a partitioning scheme superior to *METIS*, that is, with a better load balancing taking into account both the number

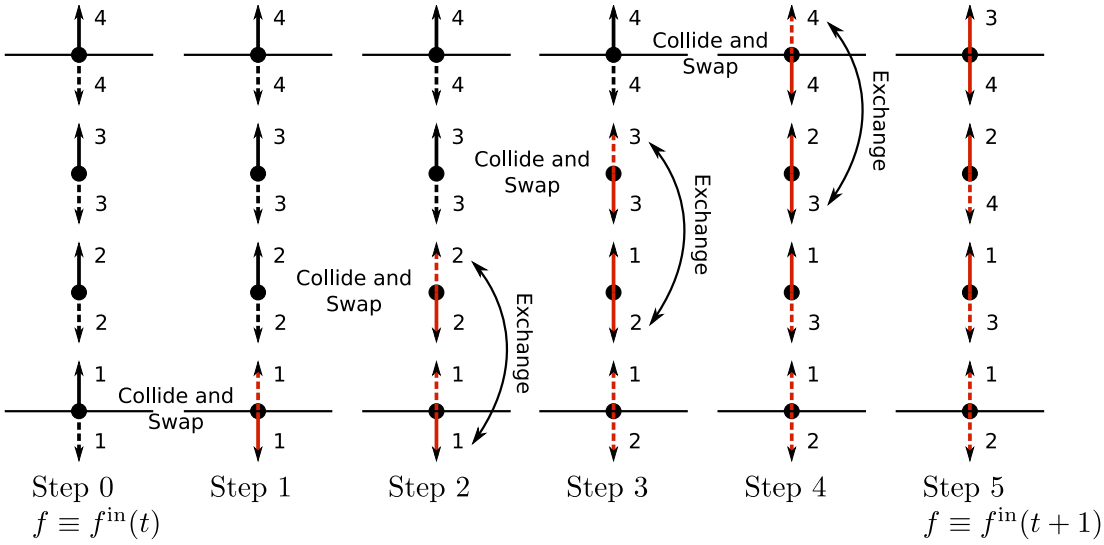

Figure 1. Schematic representation of a single collision-streaming cycle on four cells of a one-dimensional Lattice Boltzmann simulation with two populations per cell (detailed explanations are found in the text). The two populations on each cell are distinguished by the use of a solid line for the first and a dashed line for the second. The numbers next to the populations label the cell on which the populations were located at the initial time step t. Red denotes a population that has reached the post-collisional state.

of mesh nodes per processor and the total communication among the processors. Unfortunately, *PT-SCOTCH* runs out of memory on our real test case that produces a graph with almost one billion vertices and ~ 18 billion edges. Our solution has been to use a *pruned* graph which represents the connectivity along the six main directions only $(+x, -x, +y, -y, +z, -z)$. This reduces the number of edges in the graph by 66% (by eliminating the 12 edges: $+x+y, +x-y$, *etc*). We confirm that, in a smaller test case, the resulting partition is, for all practical purposes, very similar to the partition produced for the whole graph. By using the pruned graph we managed to create the required partition ($294,912$ domains) for the large test case on a cluster using 128 Intel cores (Xeon E5520 @ 2.27 Ghz) with a total of 256 GB of memory.

It is interesting to look at the distribution of the tasks with respect to the number of other tasks with which they are required to exchange data, following the partitioning scheme produced by *SCOTCH*. The result is reported in Figure 2, which shows that, on average, each task exchanges data with other 15 tasks.

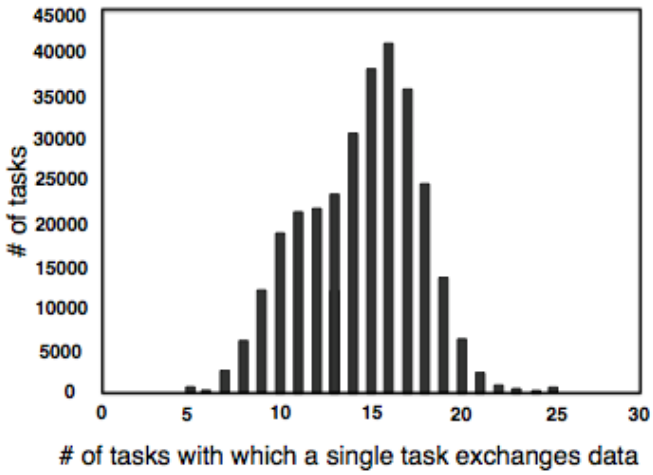

Figure 2. The distribution of the 294,912 tasks with respect to the number of tasks with which they are required to communicate

This workload distribution indicated that despite the highly complex geometry, the final workload partitioning ends up relatively close to the initial topological input, which we expect indeed as a heuristic measure of good balance. Visual inspection of the computational domains, shows that this is realized through a fairly sophisticated and highly varied morphology of the computational domains, often taking highly counterintuitive shapes in the vicinity of geometrical complexities. This "morphological richness", which stands in stark contrast with elementary partitionings in idealized geometries (cubes, slabs and similar), conveys an intuitive flavor for the complexity of the partitioning task and also hints at some form of homeo-morphism between dynamics and geometry which surely deserves a separate investigation for the future.

We create the communication pattern by the following "run-time" pre-processing procedure. Mesh nodes are assigned to tasks according to the partition created as described above. Each mesh node is also labeled, in the input file, with a *tag* that identifies it as belonging to a specific subregion of the computational domain (*e.g.*, fluid, wall, inlet, outlet). After the assignment of the nodes to the tasks, the *pre-processing* phase begins. Basically, each task asks which tasks own the nodes to be accessed during the subsequent phases of simulation, for instance for the streaming part of the LB algorithm and for the Molecular Dynamics. Such information is exchanged by using MPI collective communication primitives, so that each

task knows the neighboring peers for send/receive operations. Information about the size of data to be sent/received is exchanged as well.

All point-to-point communication operations make use of the same scheme: the *receive* operations are always posted in advance by using corresponding non-blocking MPI primitives, then the *send* operations are carried out using either blocking or non-blocking primitives, depending on the parallel platform in use (unfortunately, as it is well known, few platforms allow real overlapping between communication and computation). Then, each task waits for the completion of its receive operations, using the MPI *wait* primitives. The latter operation, in the case of non-blocking *send* operations, is to wait for their completion. The choice between blocking and non-blocking *send* can be done at run time. The evaluation of global quantities (*e.g.,* the momentum along the x, y, z directions) is carried out by using MPI collective *reduction* primitives.

Molecular Dynamics with a highly irregular domain decomposition is a major challenge in itself. In most parallel Molecular Dynamics applications the geometry of the spatial domain is a regular bounding box with Cartesian decompositions defined in such a way that each task has (approximately) the same number of particles and minimal communicating regions. In our case, this strategy would generate two separate domain decompositions: one for the LB (defined by the graph-based partitioning method previously described) and another for the MD part of the simulation. As a consequence, the exchange of momentum between particles and fluid would become a non-local operation with a very high cost due to the long-range point-to-point communications imposed on the underlying hardware/software platform. For the IBM Blue Gene such communications are explicitly discouraged. We decided to resort to a domain decomposition strategy where the MD parallel domains coincide with the decomposition of the LB mesh. In this way, each computational task performs both the LB and MD calculations and the interactions of the particles with the fluid are quasi-local.

The underlying LB mesh serves the purpose of identifying particles that belong to the domain via a test of membership: a particle with position $\mathbf{R}$ belongs to the domain if the vector of nearest integers [round(R_x), round(R_y), round(R_z)] coincides with a mesh point of the domain. In addition, we exploit the load balancing of the mesh partitioning into the MD component, given that an even number of RBCs is expected to populate the domains. For the MD part of the code, a novel parallelization strategy suitable for the irregular geometry of the LB domains has been developed. Our solution relies on the notion of cells, parallelepipeds with linear sizes greater or equal to the interaction cutoff, that cover the whole irregular domain. This representation allows the processors to *i)* perform an efficient search of both interdomain and intradomain pairs of particles and *ii)* to reduce data transfers by exchanging a limited superset of the particles actually involved in interdomain pairs and particles moving across domains.

The cells are grouped into three sets (*internal*, *frontier* and *external* cells) that verify the following properties:

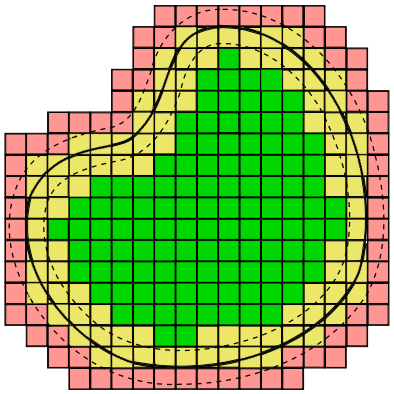

Figure 3. Decomposition of a 2D domain in *external* cells (red), *frontier* cells (yellow) and *internal* cells (green). The dashed line represents the region within a cutoff distance from the domain (solid line). The domain frontier has a staircase shape, but in this figure it is shown as a smooth curve for f simplicity.

1) Every point of the domain is within either an *internal* or a *frontier* cell;
2) *Internal* cells contain only points of the domain at distance greater than the cutoff distance from the domain boundary;
3) *Frontier* cells contain all the points of the domain at distance less than or equal to the cutoff distance from the domain boundary;
4) *External* cells contain only points outside the domain;
5) All external points at distance less than or equal to the cutoff distance from the domain boundary lie within either an *external* or a *frontier* cell.

Figure 3 shows an example of such decomposition applied to a simplified two dimension domain.

The decomposition into cells helps in handling MD for irregular domains in the following way. At the beginning of each iteration, each processor searches for the particles inside its domain that could interact with particles located inside neighboring domains. Property 3 guarantees that the particles are only contained inside *frontier* cells. All particles located in the *frontier* cells are exchanged with neighboring processors so that only a limited superset of the particles that could interact with the outer region is transferred. On the receiving side, only the particles that could interact with the inner region are considered. Given property 5, the received particles to be retained lie inside either *external* or *frontier* cells. After particles involved in interdomain pairs are exchanged, forces can be computed and particles positions updated.

Next, RBC migration among processors is handled. All particles are binned inside the cells they moved into, and those that left the domain are exchanged with neighboring processors. Departing particles are found by selecting those that moved to *external* cells (property 4) and *frontier* cells. To discriminate RBCs inside the frontier cells that moved to other domains, we make use of the underlying mesh by means of the membership test previously described. In this way, each processor sends exactly the particles that left its domain

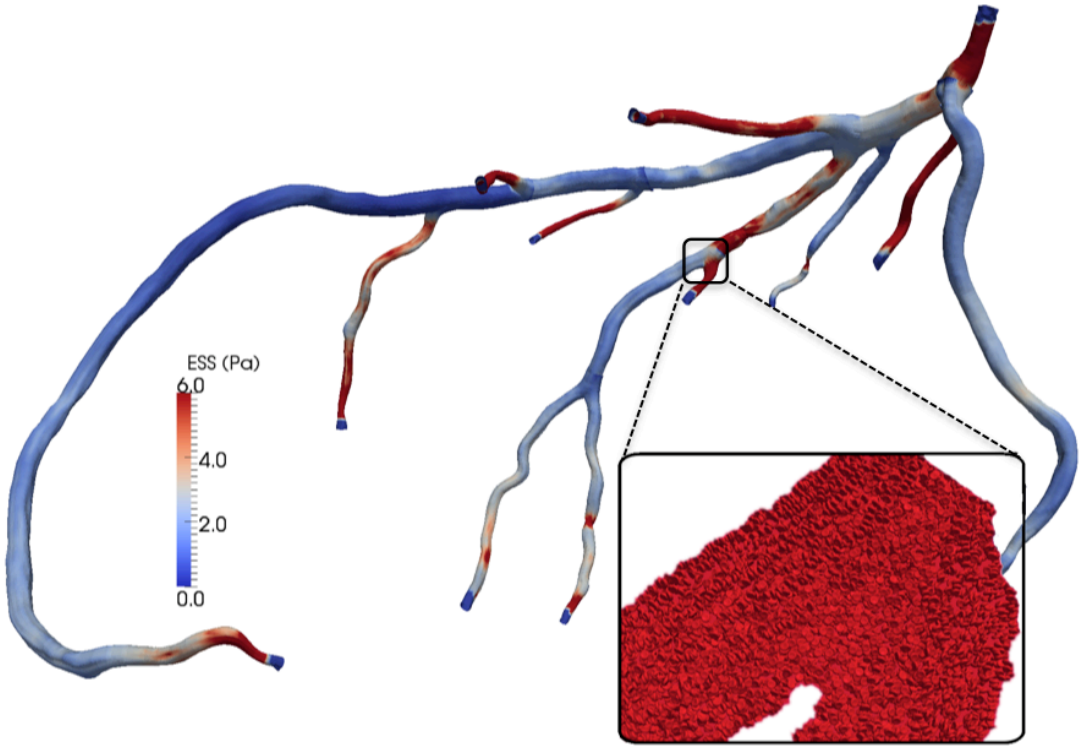

Figure 4. The geometry of the 12.5μm resolution test case, derived from a CTA scan of human coronary arteries. The inset shows a detail of the geometry with red blood cells visible. Note: the red color in the inset is meant simply to highlight the presence of RBCs and is not an indicator of ESS. The Endothelial Shear Stress (ESS) is the field derived from the simulations that encodes the atherosclerotic risk map and is represented as a color map on the arterial walls.

to all neighboring domains. On the receiving side, incoming particles are selected among the pool of all transferred ones.

The final component of our multiscale methodology involves the fluid-particle coupling. Each suspended RBC experiences hydrodynamic forces and torques arising from the fluid macroscopic velocity and vorticity, smeared over a domain made of $4 \times 4 \times 4$ mesh points. This non-local operation requires a communication step such that each processor owning a given particle exchanges the hydrodynamic quantities with the surrounding processors. The same type of information is exchanged to build the forces acting on the fluid and arising from the suspended RBCs.

VI. Results

The performance of the Lattice Boltzmann component of *MUPHY* on a single core is in line with other LB kernels highly tuned for the Blue Gene/P platform[29]. From this viewpoint, we wish to note that: *i)* the algorithm for the update of the LB populations has an unfavorable ratio between number of floating point operations and number of memory accesses; *ii)* unlike other applications which can heavily draw upon consolidated computational kernels (*e.g.,* matrix operations or FFTs), no optimized libraries are available to perform the basic LB operations; and *iii)* it is not possibile to exploit the SIMD-like operations of the PowerPC 440 processor, since these require stride-one access whereas the LB method has a "scattered" data access pattern due to the streaming phase.

We focus on the total runtime for the simulation, as well as on the breakdown between computation and communication. To this end, we ran a simulation at $12.5\,\mu$m resolution, corresponding to about 1 billion lattice sites for the fluid flow. All simulations were run over 200 time-steps. The measurements were performed on the Jülich Blue Gene/P with $294,912$ cores, 144 TB of total memory and a theoretical peak performance of about 1 Petaflops. All runs were made in VN mode.

With a mesh having 1 billion fluid nodes within a bounding box having a total of almost 300 billion nodes, our reference hematocrit level corresponds to 10 million RBCs, as we run on the 72 racks system. More recently, we obtained results utilizing the same mesh but with 100 and 300 million RBCs on the 40 rack Blue Gene/P system at Argonne National Laboratory. These results provided a fundamental check of the reliability of the code at physiologic levels of hematocrit.

The successful completion of the simulations at each number of RBCs proves the feasibility and robustness of the method up to physiological levels. The joint usage of linkcell algorithms to compute pairwise interactions together with the linear method to access the indirect addresses of the mesh for the RBC-fluid exchange of hydrodynamic forces proved the linearity of the multiscale methodology with the problem size on the 40 racks installation. Through the combination of the

fine mesh and inclusion of red blood cells, we were able to observe the ESS in the real patient over the course of several hundred time steps as shown in Figure 4.

A. *Strong Scaling*

This scaling analysis is performed by increasing the number of processors at a fixed problem size, in an effort to analyze the impact of the number of computational cores on the total simulation time. In Table I and Figure 5, we show the elapsed time per time-step, as well as the breakup for the LB and MD components separately. A few comments are in order. First, the elapsed time decreases significantly with the number of cores, with a speed-up of 43.5 between the $4,096$- versus $294,912$- core configurations (see Fig. 6), corresponding to a parallel efficiency in excess of $60\,\%$. This result is particularly significant given that the average number of mesh points per computational core becomes pretty low (*i.e.,* $\sim 3,300$) on the full configuration of $294,912$ cores. Particularly efficient are the data concerning the LB component, showing a speed-up of 54.1 and efficiency of $75\,\%$. Second, we notice that up to $147,456$ cores the MD and LB sections remain in a fairly satisfactory balance with each other across the whole range of cores, thereby highlighting the excellent quality of the workload partitioning. Third, the MD component shows saturation above $147,456$ cores. This is not unexpected, since at this number of cores and with 10 million RBCs each domain contains an average number of 60 RBCs, a critically small number regarding the calculations of Gay-Berne forces, the time-consuming MD component. Below the threshold of $147,456$ cores, the constant ratio between the LB and MD workloads underscores the good response of the MD component in dealing with a handful of particles per domain. It is likely that, with a significant increase in the number of RBCs, the MD component would show a further speed-up up to $294,912$ cores.

Table I
BREAKDOWN OF THE ELAPSED RUNTIME

Cores	LB	MD	LB+MD
4,096	0.4761	0.04633	0.5224
16,384	0.1191	0.01610	0.1352
147,456	0.0151	0.00419	0.0193
294,912	0.0088	0.00419	0.0130

To further analyze the parallel performance of our simulation, we next inspected the breakdown of the communication time across the pool of cores. Details of the communication performance were obtained using the MPI Profiler [31]. The communication times decrease significantly as the number of cores increases, roughly by 24% when going from $16,384$ to $294,912$ cores. The time for communication by the master core remains basically the same, with just a minor 5% increase. Table II reports the MPI function communication summary, where the Send/Irecv calls represent by and large the most time-consuming communication routines. The bandwidth for the (blocking) MPI_send corresponds to roughly 27 MB/sec

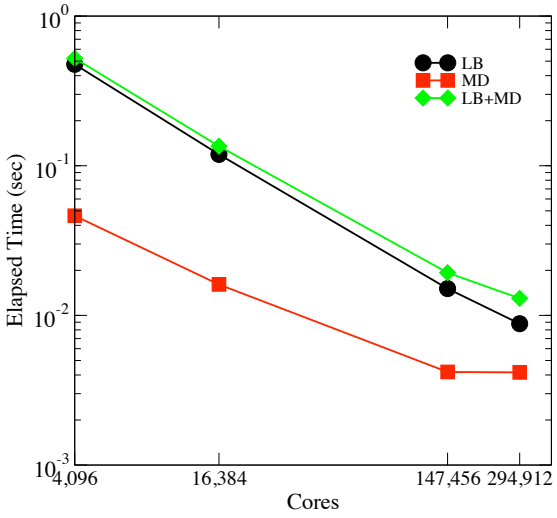

Figure 5. Log-log plot of the elapsed time for the LB component (circles), the MD component (squares) and for the full simulation (diamonds) versus the number of cores, for the system composed by 1 billion fluid nodes and 10 million RBCs.

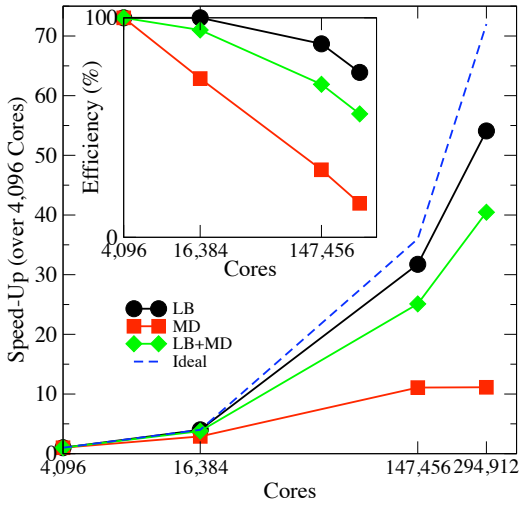

Figure 6. Semilog plot of the speed-up for the LB (circles) and MD (squares) components, for the full simulation (diamonds), and for the ideal regime (dashed line) versus the number of cores. Data are for the system of 1 billion fluid nodes and 10 million RBCs.

and is satisfactory in view of the highly non-trivial communication pattern.

Table II
COMMUNICATION BREAKDOWN FOR THE RUN WITH $294,912$ CORES.

MPI routine	Calls	Avg. bytes	Time(sec)
MPI_Send	10251	1187.7	0.452
MPI_Irecv	10302	17148.0	0.016
MPI_Waitall	603	0.0	0.222

B. Hardware Performance Monitoring

Finally, we conducted a performance analysis using the hardware performance monitoring library (HPM) on Blue Gene/P. HPM tracks 256 performance counters that measure events ranging from integer and floating-point operations to cache and memory accesses. The hardware counters can be set to measure the performance for either cores 0 and 1 or cores 2 and 3 during a single execution. The tool reports Flops as a weighted sum of various floating-point operations. More information is given in [31]. For 72 racks of Blue Gene/P in VN mode ($294,912$ cores), we measured 64 TeraFlops, as shown in Figure. 7. In view of the intrinsic Flop-limitations of the Lattice Boltzmann algorithm discussed previously, and taking into account the coupling between LB and MD components, this appears to be a fairly satisfactory overall performance. Just to convey the flavor of the practical impact of this application, the above performance corresponds to simulating a full heartbeat at microsecond resolution in only a few hours time on the 72-rack Blue Gene/P system. Using 40 racks, we were able to successfully complete the simulation of a full heartbeat in just under six hours.

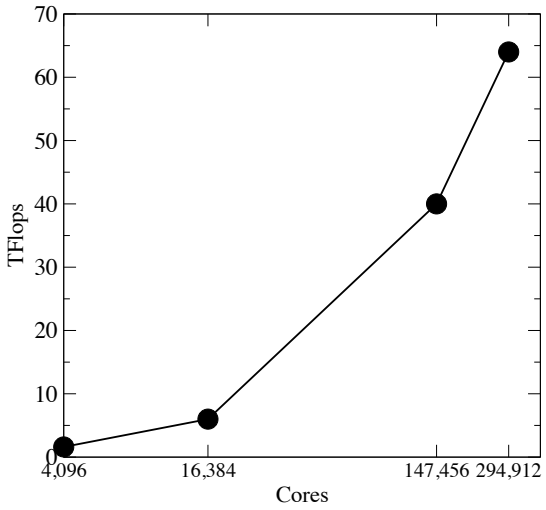

Figure 7. Aggregate performance (Floating Point Operations Per Second) as a function of the number of cores.

VII. CONCLUSIONS

Summarizing, we have presented the first large-scale simulation ever of the entire heart-circulation cardiovascular system, with a realistic representation of the complex human arterial geometry at the spatial resolution of red-blood cells: from centimeters all the way down to microns in a single multiscale simulation. This simulation, involving one-billion fluid nodes, embedded in a bounding space of three hundred billion voxels and coupled with the concurrent motion of ten million red-blood cells, achieves over 60 Teraflops performance on the full $294,912$ Blue Gene/P processor configuration, with a parallel efficiency in excess of 60 percent, performing about 100 billion lattice updates per seconds. Using the same arterial system and simulation parameters it has been possible to elevate the hematocrit level to physiological levels of 300 million RBCs.

The above achievement results from the development of several unique features, in terms of both high-performance computing technology and of physical/computational modeling, namely i) the solution of the formidable graph-partitioning problem prompted by the need of evenly distributing the workload associated with the complex arterial geometry, across as many as $294,912$ Blue Gene/P cores; ii) the innovative communication techniques required to secure a balanced workload between fluid-dynamics and Molecular Dynamics in geometries of real-life complexity; and iii) the innovative modeling techniques required to manage the self-consistent fluid-particle interactions in complex geometries. As to (ii), we are not aware of any previous implementation dealing with non-ideal geometries.

This work represents major progress in the predictive capabilities of computer simulation for *real-life* cardiovascular applications. The scientific and societal impact of the extensions of such activity cannot be underestimated.

Currently, no combination of computational models, however sophisticated, can provide a comprehensive and all-embracing description of all complex phenomena which underlie the dynamical behaviour of the entire human cardiovascular system, including a realistic description of the arterial tissues, wall compliance, RBCs deformability, to name but a few. However, this does not prevent the possibility to gain completely new insights on specific cardiovascular phenomena of major clinical relevance. For instance, as far as long-term atherogenesis is concerned, neither wall compliance, nor RBCs deformability, are credited for playing a lead role. On the other hand, even in large arteries, the finite extent of the RBCs, is likely to exert a major effect on the near-wall circulation patterns, hence the local wall shear stress distribution. This is the kind of effect that the present simulations are expected to shed new light on, once appropriate hardware resources are available.

VIII. ACKNOWLEDGMENTS

We wish to thank C.L. Feldman, A.U. Coskun, F.J. Rybicki, and A.G. Whitmore for help in preparing the anatomic data and for numerous discussions. We wish to thank the staff

at the CI Lab at Harvard University, at the École Polytechnique Fédérale de Lausanne, at the Jülich Supercomputing Center and at the Argonne National Labs for precious support and computer time at the respective Blue Gene installations. We would also like to thank Brian Hayes, Rosalind Reid, and Edward Randles for their assistance. This research used resources of the Argonne Leadership Computing Facility at Argonne National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02-06CH11357.

REFERENCES

[1] D.A. Vorp, D.A. Steinman, C.R. Ethier, "Computational Modeling of Arterial Biomechanics", *Comput. Sci. Eng.*, vol. 3, 2001, pp. 51-64.

[2] A. Quarteroni, A. Veneziani, P. Zunino, "Mathematical and numerical modeling of solute dynamics in blood flow and arterial walls", *SIAM J. Num. Analysis*, vol. 39, no. 5, 2002, pp. 1488-1511.

[3] L. Grinberg, T. Anor, E. Cheever, et al., "Simulation of the human intracranial arterial tree", *Phil. Trans. Royal Soc. A*, vol. 367, no. 1896, Jan. 2009, pp. 2371-2386.

[4] R. Ouared and B. Chopard, "Lattice Boltzmann Simulations of Blood Flow, Non-Newtonian Rheology and Clotting Processes", *J. Stat. Phys.* vol. 121, Oct. 2005, pp. 209-221.

[5] D.J.W. Evans, P.V. Lawford, J. Gunn, D. Walker, D.R. Hose, R.H. Smallwood, B. Chopard, M. Krafczyk, J. Bernsdorf, A. Hoekstra, "The application of multiscale modelling to the process of development and prevention of stenosis in a stented coronary artery", *Phil. Trans. R. Soc. A*, vol. 366, no. 1879, Sept. 2008, pp. 3343-3360.

[6] I.V. Pivkin, P. Richardson and G. Karniadakis, "Blood flow velocity effects on role of activation delay time on growth and form of platelet thrombi", *Proc.Natl.Acad.Sci.*, vol. 103, no. 46, Nov. 2006, pp. 17164-17169.

[7] J.L. Mc Whirter, H, Noguchi and G. Gompper, "Flow-induced clustering and alignment of vesicles and red blood cells in microcapillaries, *Proc.Natl.Acad.Sci.*, vol. 106, no. 15, April 2009, pp. 6039-6043.

[8] S.K. Veerapaneni, D. Gueyffier, G. Biros, and D. Zorin, "A numerical method for simulating the dynamics of 3D axisymmetric vesicles suspended in viscous flows", *J. Comp. Phys.*, vol. 228, Oct. 2009, pp. 7233-7249.

[9] M. Bernaschi, S. Melchionna, S. Succi et al. *MUPHY*: "A parallel MUlti PHYsics/scale code for high performance bio-fluidic simulations", *Comp. Phys. Comm.*, vol. 180, Sept. 2009, pp. 1495-1502.

[10] S. Melchionna, M. Bernaschi, S. Succi et al, "Hydrokinetic approach to large-scale cardiovascular flows", *Comp. Phys. Comm.*, vol. 181, 462, (2010).

[11] J. C. Phillips, G. Zheng, S. Kumar, and L. V. Kal. NAMD: Biomolecular Simulation on Thousands of Processors. Proc. IEEE/ACM SC2002 Conf. IEEE Press, 2002. Technical Paper 277.

[12] J. C. Phillips, G. Zheng, S. Kumar, and L. V. Kal. "NAMD: Biomolecular Simulation on Thousands of Processors". *Proc. IEEE/ACM SC2002 Conf.* IEEE Press, 2002. Technical Paper 277.

[13] LB simulations with coupled bodies, even deformable ones, are available in the literature, see for instance M. Dupin et al, Phys. Rev. E, 75, 6, (2007), but we are not aware of any massively parallel implementation of this method to tackle full heart-circulation problems. Parallel LBM applications to complex geometries have been performed by L. Axner et al, J. Comp. Phys., 227, 4895, (2008), but on much smaller sizes (order of ten million cells) and with no suspended particles. Computational fluid dyamics with multi-body dynamics are also available, see J. Gotz et al, Parallel Computing, 36, 142, (2010), but on a much smaller number of nodes (8192). Larger Lattice Boltzmann simulations, with up to 150 billions cells and 260 million suspended particles, have been recently presented (J. Gotz et al, Supercomputing 2010). However, these simulations only deal with idealized cartesian geometries.

[14] C. S. Peskin, "Accurate coarse-grained modeling of red blood cells", *Acta Numer.* 2002, vol. 479.

[15] I. Pivkin and G.E. Karniadakis, "Accurate coarse-grained modeling of red blood cells", *Phys. Rev. Letts.*, vol. 101, 2008, 118105.

[16] S. Melchionna, in preparation.

[17] S. Succi, The Lattice Boltzmann Equation for Fluid Dynamics and Beyond, Oxford University Press, USA, 2001

[18] Q. Zou and X. He, Phys. Fluids, 9, 1591 (1997).

[19] C. Chevalier and F. Pellegrini, "PT-Scotch: A tool for efficient parallel graph ordering." *Parallel Computing*,Jan. 2008, pp. 318-331.

[20] S. Succi, *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond*, Oxford University Press, USA, 2001.

[21] Q. Zou and X. He, "On pressure and velocity boundary conditions for the lattice Boltzmann BGK model," *Phys. Fluids*, vol. 9, 1997, pp. 1591.

[22] G Karypis, V Kumar, "METIS 3.0: Unstructured graph partitioning and sparse matrix ordering system". Dept. Computer Sci., Univ. Minnesota, Tech. Rep, 1997

[23] R. Benzi, S. Succi and M. Vergassola, "The Lattice Boltzmann equation: Theory and applications". *Phys. Rep*. vol. 222, 1992, pp. 145.

[24] J.G. Gay, B.J. Berne, J. Chem. Phys, 74, 3316 (1981)

[25] A. Gara et. al, "Overview of the Blue Gene/L system architecture". *IBM Journal of Research and Development*, vol. 49, issue 2, March 2005, pp. 195-212.

[26] M. Bernaschi, M. Fatica, S. Melchionna, S. Succi and E. Kaxiras, "A flexible high performance Lattice Boltzmann GPU code for the simulations of fluid flows in complex geometries", *Concurrency and Computation: Practice and Experience*, 2009, DOI: 10.1002/cpe.1466.

[27] K. Mattila, J. Hyväluoma, T. Rossi, M Aspnäs, J. Westerholm, "An efficient swap algorithm for the lattice Boltzmann method", *Comp. Phys. Comm.*, vol. 176, 2007, pp. 200.

[28] MD Mazzeo, PV Coveney PV, "HemeLB: A high performance parallel lattice-Boltzmann code for large scale fluid flow in complex geometries", *Comp. Phys. Comm.*, vol. 178, 2008, pp. 894.

[29] JR Clausen, DA Reasor Jr., CK Aidun, "Parallel performance of a lattice-Boltzmann/finite element cellular blood flow solver on the IBM Blue Gene/P architecture", *Comp. Phys. Comm.* (in press, 2010).

[30] A. Dullweber, B. Leimkuhler, R. McLachlan, "Symplectic splitting methods for rigid body molecular dynamics", *J. Chem. Phys.*, vol. 107, 1997, pp. 5840.

[31] G. Lakner, I.-H. Chung, G. Cong, S. Fadden, N. Goracke, D. Klepacki, J. Lien, C.Posiech, S.R. Seelam and H.-F. Wen. IBM System Blue Gene Solution: Performance Analysis Tools. *IBM Redpaper Publication*, Nov. 2008.