LLNL-CONF-670030

LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

Massively Parallel Models of the Human Circulatory System

A. Randles, E. W. Draeger, T. Oppelstrup, W. Krauss, J. Gunnels

April 27, 2015

Supercomputing 2015
Austin, TX, United States
November 15, 2015 through November 20, 2015

Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.

Massively Parallel Models of the Human Circulatory System

Amanda Randles
Lawrence Livermore National Laboratory
Livermore, CA 94550
Duke University
Durham, NC, USA
amanda.randles@duke.edu

Erik W. Draeger
Lawrence Livermore National Laboratory
7000 East Ave.
Livermore, CA 94550
draeger1@llnl.gov

Tomas Oppelstrup
Lawrence Livermore National Laboratory
7000 East Ave.
Livermore, CA 94550
oppelstrup2@llnl.gov

Liam Krauss
Lawrence Livermore National Laboratory
7000 East Ave.
Livermore, CA 94550
liam1@llnl.gov

John A. Gunnels
IBM T.J. Watson Research Center
1101 Kitchawan Road
Yorktown Heights, NY 10598
gunnels@us.ibm.com

ABSTRACT

The potential impact of blood flow simulations on the diagnosis and treatment of patients suffering from vascular disease is tremendous. Empowering models of the full arterial tree can provide insight into diseases such as arterial hypertension and enables the study of the influence of local factors on global hemodynamics. We present a new, highly scalable implementation of the lattice Boltzmann method which addresses key challenges such as multiscale coupling, limited memory capacity and bandwidth, and robust load balancing in complex geometries. We demonstrate the strong scaling of a three-dimensional, high-resolution simulation of hemodynamics in the systemic arterial tree on 1,572,864 cores of Blue Gene/Q. Faster calculation of flow in full arterial networks enables unprecedented risk stratification on a per-patient basis. In pursuit of this goal, we have introduced computational advances that significantly reduce time-to-solution for biofluidic simulations.

Keywords

Category: time-to-solution

1. INTRODUCTION: OVERVIEW OF THE PROBLEM AND ITS IMPORTANCE

One of the long-established challenges facing the medical community is not only the prevention of major cardiovascular and cerebrovascular events such as myocardial infarction and stroke, but also the identification of those individuals most at risk. Despite advances in medical imaging and early screening methods, 50% of men and 64% of women who die suddenly of cardiovascular disease have no previously detected symptoms [4]. It is now possible to use medical imaging to design computer simulations using the specific geometry and dynamics of a person's circulatory system. Over the last several decades, researchers have established the use of these image-based hemodynamic simulations as a means to gain insight into the localization and progression of vascular disease [16, 25, 36]. Such patient-specific hemodynamic models can provide key insights into abnormalities that may be associated with vascular disease risk factors. The computational demands of these simulations have historically restricted their size and scope, but advances in parallel algorithms and computer hardware[26, 3, 12, 10] have extended the reach of such simulations to much larger regions of the circulatory system. Results stemming from these capabilities have enabled at-risk patient identification and surgical planning previously not possible through computational or experimental studies for a range of diseases including, but not limited to: congenital heart defects [24, 18], cerebral aneurysm [11], aortic aneurysm [6, 42], and coronary artery atherosclerosis [8, 20].

While there has been a great deal of progress made towards enabling the study of flow in small segments of the circulatory system, computational demand has limited the advancement of systemic circulatory modeling. Systemic arterial flow simulations could greatly improve the ability of clinicians to both identify and treat at-risk patients. For example, models could assist in the early identification of patients suffering from peripheral artery disease (PAD), a significant manifestation of systemic atherosclerosis in which narrowed arteries reduce flow to the extremities. The Framingham Heart Study showed that PAD is associated with a two- to four-fold increase in risk of mortality from cardiovascular disease and a loss of 10 years in life expectancy [33]. Early identification and intervention can reduce the likelihood of IC, the clinical manifestation of PAD, thereby improving cardiovascular morbidity and overall mortality [21].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
SC '15, November 15-20, 2015, Austin, TX, USA
Âľ 2015 ACM. ISBN 978-1-4503-3723-6/15/11âĂę$15.00
DOI: http://dx.doi.org/10.1145/2807591.2807676

One proven diagnostic used to identify the severity of PAD and predict the likelihood of IC is the ankle-brachial index (ABI), defined as the ratio of the systolic blood pressure measured at the ankle to that in the arm[40]. Measurement of the ABI has been shown to be a promising technique for improving the accuracy of cardiovascular risk prediction [23, 22, 31, 9, 7]. Systemic arterial simulations can improve risk stratification through calculation of ABI for a range of physiological conditions not easily replicated in the physician's office, as well as predicting the impact of different interventions on critical measurements such as the ABI. Image-based simulations would provide further insight into the impact of both disease states and potential treatment plans.

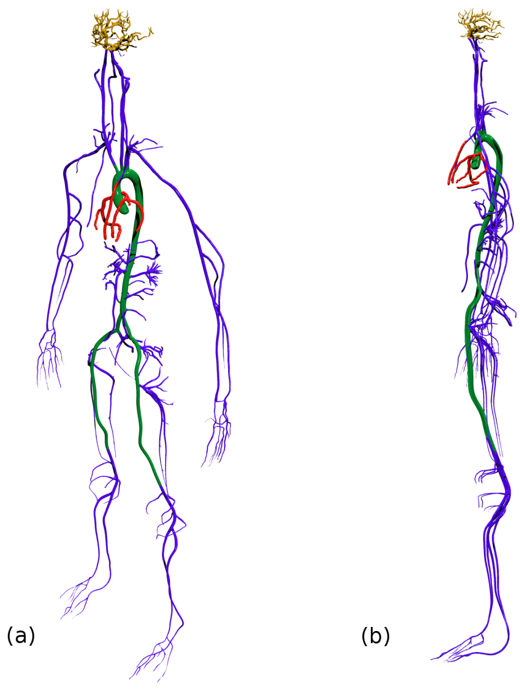

Figure 1: System arterial geometry. (a) Frontal view. (b) Side view. The vessels shown in blue demonstrate the arterial tree modeled in this work including all arteries with diameters greater than 1mm. The gold indicates the region of the body modeled in [12], the red corresponds to [26, 3, 10], and the green represents the region modeled in [30].

Furthermore, risk indicators such as ABI need to be understood for a range of physiological circumstances (exercise, rest, at altitude, etc.), co-existing conditions (e.g. anemia or polycythemia), and over long time durations. The ability to quantify a patient's specific risk state over time remains an outstanding issue. **In order to address such challenges and facilitate future clinical translation, it is imperative that we not only enable system-level simulation, but also drastically reduce the time-to-solution so as to be much faster than real time.**

However, achieving such large-scale simulations presents an immense computational challenge due to the geometric complexity of the system, memory requirements associated with high-resolution grids, and load balancing issues associated with the processor core counts required. Simply storing the data for the number of grid points required to model the systemic arterial network requires memory on the scale of that possessed by leadership-class supercomputers. In order to tackle these challenges, we extend the design and parallel efficiency of our C/C++ software, HARVEY [27], a computational fluid dynamics code based on the Lattice Boltzmann Method (LBM). Efficiently utilizing such supercomputer systems requires that work be assigned to over one million processes while only computing and storing local data. To address this, we present a lightweight load balancing algorithm to address the sparseness of vascular domains. Furthermore, we introduce a load balance cost function in order to analyze the computational load per unit of work. We also demonstrate a strong correlation between our estimated cost and measured cost-per-process.

In this work, we evaluate the scalability and performance of HARVEY in a 3-dimensional vascular geometry consisting of all arteries greater than 1 mm in diameter. We demonstrate strong scaling to 1.57 million cores of the LLNL supercomputer, Sequoia, an IBM Blue Gene/Q system. We show that HARVEY can successfully model complex geometries of an unprecedented scale in an efficient manner.

This work presents major advancements to patient-specific hemodynamic modeling through the following novel contributions:

- First 3-dimensional high-resolution simulation of hemodynamics in the systemic arterial tree at cellular resolution
- Two new lightweight load balance algorithms for large-scale CFD
- A $2x$ improvement in work-per-second over current state-of-the-art
- A novel load balance cost function
- Single node optimization strategies for grid-based stencil applications
- Data structure optimizations reducing our achieved time-to-solution by 82%

The remainder of this paper is organized as follows. We provide an overview of related work in the area of large-scale hemodynamics in Section 2 which includes discussion of the challenges in modeling the systemic arterial network. In Section 3, we describe the lattice Boltzmann method. We present our load balance algorithms and cost model in Section 4 and evaluate their accuracy and overall application performance in Section 5. Conclusions are found in Section 6.

2. CURRENT STATE OF THE ART FOR SCIENCE AND PERFORMANCE

The development of realistic image-based models of hemodynamics in the human vasculature has been a topic of intense focus over the last decade. Similarly, advances in high performance computing have fostered the increase in scalability of computational fluid dynamics algorithms. As Table 1

Geometry	**Resolution**	**Suspended Bodies**	**Award Status**	**Citation**
Periodic box		200 million RBCs	2010 Gordon Bell Winner	[29]
Coronary arteries	O(10μm)	300 million RBCs	2010 Gordon Bell Finalist	[26]
Coronary arteries	O(10μm)	450 million RBCs	2011 Gordon Bell Finalist	[3]
Cerebral vasculature	O(1nm)	RBCs and platelets	2011 Gordon Bell Finalist	[12]
Coronary arteries	O(1μm)	fluid only		[10]
Aortofemoral	O(10μm)	fluid only		[30]

Table 1: Large-scale hemodynamics simulations. We provide an overview of the region of the circulatory encapsulated, number of suspended bodies, and award status in landmark computational hemodynamic studies.

shows, these efforts have resulted in high impact research pushing the limits of what is computationally possible year after year. Blood flow calculations have been the subject of three recent Gordon Bell Finalist papers, one of which won the prize.

Even with such progress, each of these codes is limited to modeling small regions of the circulatory system. The largest high-resolution simulation of a confined area is by Godenschwager *et al* using the LBM to model coronary arteries at 1.276 μm resolution [10]. Works looking at larger regions of the body typically employ a one-dimensional or lump parameter model (c.f. [34], [38], [1], [32]). Xiao *et al* presented the first 3-dimensional model of unsteady flow in the large primary arteries of the human circulatory system. This research served as a feasibility study for utilizing a 3D framework, but the resolution was too low to demonstrate grid independence [41]. For the macroscopic quantities of interest in these simulations such as pressure and shear stress, a resolution of 20 μm or finer is needed for convergence. We recently demonstrated efficient simulation of the aortofemoral region at 10 μm resolution using 1.57 million cores of the IBM Blue Gene/Q Sequoia Supercomputer at Lawrence Livermore National Laboratory [30].

In this work, we present high-resolution unsteady flow dynamics in a geometry consisting of all arteries above 1 mm in diameter, extracted from computed tomography (CT) images. The segmentation and volumetric mesh construction of the arterial geometry was performed by Simpleware Ltd. Exeter, UK. For a 9 μm resolution simulation, the systemic model requires an overall bounding box of 68909 x 25107 x 188584 grid points containing 509.0 billion fluid nodes and 4.5 billion wall, inlet, and outlet nodes. The results presented demonstrate the largest region modeled to date at a resolution from which conclusions about key risk factors and diagnostic tests such as the ABI can be drawn. Moreover, as red blood cells range in size from 8-10 μm in diameter, the systemic arterial hemodynamics are being modeled at the cellular scale.

3. LATTICE BOLTZMANN

In this work, we use the LBM introduced by both the teams of McNamera and Zanetti [19] and Higuera and Jimenez [15]. LBM is an alternative to the classical Navier-Stokes equation for computational fluid dynamics. The LBM comes from kinetic theory and is a minimal form of the Boltzmann equation. The simulation domain is discretized into a regular Cartesian grid and macroscopic flow properties are derived from the collective dynamics of fictitious particles that represent a local ensemble of molecules moving between the grid points. An explicit time-stepping scheme that is particularly well-suited for massively parallel simulations is used: the stencil is formed by local neighbors of each computational node, so in each time step information is only exchanged between neighboring nodes (c.f. [5, 39, 28]). For more details regarding the LBM, see [35].

The governing equation describes the evolution of the distribution function denoted by $f_i(\vec{x}, \vec{c}_i, t)$, describing the probability of finding a particle at grid point $\vec{x}$, at time t, with discrete velocity $\vec{c}_i$. In this work, we use the 19-speed cubic stencil, D3Q19, with the Bhatnager-Gross-Krook (BGK) collision formulation using a single relaxation time. The grid spacing is defined by Δx, where discrete velocities connect grid points to first and second neighbors on the 19-point stencil. The fluid populations are advanced in a timestep Δt through:

$$f_i(\vec{x}+\vec{c}_i\Delta t, t+\Delta t) = f_i(\vec{x}, t) - \omega\Delta t[f_i(\vec{x}, t) - f_i^{eq}(\vec{x}, t)] \quad (1)$$

The local equilibrium, $f_i^{eq}(\vec{x}, t)$, is the result of a second-order expansion in the fluid velocity of a local Maxwellian with speed $\vec{u}$ and is defined by:

$$f_i^{eq} = w_i \rho \left[1 + \frac{\vec{c}_i \cdot \vec{u}}{c_s^2} + \frac{1}{2}\left(\frac{(\vec{c}_i \cdot \vec{u})^2}{(c_s^2)^2} - \frac{u^2}{c_s^2}\right)\right] \quad (2)$$

where ρ denotes the density, $\vec{u}$ the average fluid speed, $c_s = 1/\sqrt{3}$ the speed of sound in the lattice, and w_i the weights attributed to each discretized velocity as determined by the lattice structure. Due to the use of explicit time-stepping, LBM requires small time-steps that scale with Δx^2. In the case of the 20 μm simulations discussed in this work, approximately 1 million time-steps are required to simulate one heartbeat.

We implement the Zou-He boundary conditions [43], in which a pulsating velocity is imposed at the inlet through a *plug profile* at the entrance to the vessel and a constant pressure is imposed at the outlets. While the inlet condition does not assert the familiar parabolic profile that drops to zero close to the wall, it does allow a total flow to be imposed at a set value. In a short distance past the inlet, the parabolic profile is recovered. This method uses information streamed from the bulk fluid points alongside a completion scheme for the unknown particle populations whose neighbors are outside the fluid domain. This method can be executed with second-order accuracy [17]. In this paper, the modification introduced by Hecht and Harting [14] in which the velocity conditions are specified on-site is used, thus removing the constraint that all points of a given inlet or outlet must be aligned on a plane that is perpendicular to one of the three main axes. Furthermore, this addition allows the boundary conditions to be applied locally. A no-slip boundary condition is imposed at the walls via the full bounce-back method.

4. METHODOLOGY

One of the biggest challenges of modeling the full human arterial system lies in its sheer size and scale. For the arterial geometry shown in Fig. 1, only 0.15% of the total grid points are fluid points. While keeping the entire bounding box in memory would retain the regular Cartesian structure of the underlying grid, the majority of data points would be unused during the simulation. At 20 μm resolution, an array storing only the node type (as a 1-byte char) of each point on the grid would consume nearly 30 TB. It is therefore critical that local data sizes be kept as small as possible and that data duplication is avoided at all costs.

4.1 Data Structure

In the current implementation, each processor owns all fluid and boundary nodes contained within a non-overlapping rectangular bounding box. Nodes needed from neighboring tasks are identified during initialization and lists of local points to be sent to other tasks are stored. Indirect addressing is used to loop over all local fluid points as efficiently as possible and requires minimal storage for the lattice Boltzmann distribution functions. Additional memory is used to improve the speed of stencil calculations by storing local indices of boundary points (walls, inlets or outlets) and offsets for streaming rather than computing them on the fly during each iteration. We found that these optimizations resulted in a decrease in time-to-solution of over 82% when compared to the timing at 131,072 tasks using indirect addressing only.

4.2 Cost function

Optimizing the load balance is a critical component for efficient utilization of modern supercomputers. On systems such as Sequoia, an imbalance can result in hundreds of thousands of cores sitting idle. To develop an efficient load balance algorithm, it is necessary to analyze the computational load per unit of work, in this case one fluid lattice update. The compute time in the simulation loop for a task is governed by the number of active grid cells as well as the number of source, sink, and boundary points. The volume of the simulation box assigned to the specific task may also influence the run time. To create a performance model, we gathered data on simulation loop time, number of fluid points (n_{fluid}), number of wall points (n_{wall}), number of inlet points (n_{in}), number of outlet points (n_{out}), and task bounding box volume (V) for each task in several simulations. We then proceeded to fit a function $C(n_{\text{fluid}}, n_{\text{wall}}, n_{\text{in}}, n_{\text{out}}, V)$ of the following form to the data:

$$C = a \cdot n_{\text{fluid}} + b \cdot n_{\text{wall}} + c \cdot n_{\text{in}} + d \cdot n_{\text{out}} + e \cdot V + \gamma,$$

where a, b, c, d, γ are parameters to be fit. The results were:

$$\begin{aligned} a &= \ \ 1.47 \cdot 10^{-4} \\ b &= -2.73 \cdot 10^{-6} \\ c &= \ \ 4.63 \cdot 10^{-5} \\ d &= \ \ 4.15 \cdot 10^{-5} \\ e &= \ \ 2.88 \cdot 10^{-9} \\ \gamma &= \ \ 8.18 \cdot 10^{-2} \end{aligned}$$

The accuracy of the model can be approximated by the maximum relative underestimation compared to data, $\max_{\text{tasks}}(\text{compute time})/C - 1$. For the parameters above, with 4096 tasks and 4M fluid points in the input set, this maximum is about 0.23, which indicates that a load balancer using this performance model should achieve a maximum imbalance of about 23%. The median and mean of the relative underestimation are both very close to zero.

The average number of fluid points per unit volume is about 3%, so the volume term (e) is insignificant. The major contributions stem from the number of fluid points (a) and the constant term (γ). If we use only those parameters and fit the data to a simpler performance model C^* with:

$$C^* = a^* \cdot n_{\text{fluid}} + \gamma^*,$$

we get $a^* \approx 1.50 \cdot 10^{-4}$ and $\gamma^* \approx 7.45 \cdot 10^{-2}$. This performs as well as the more detailed model above. We obtain a maximum relative underestimation of the compute time of about 0.22, and the median and mean are again very close to zero. Fig. 2 shows scatter plots over the accuracy of this simplified performance model. We conclude that load balancing based on the number of fluid points in a rank should allow excellent scaling.

4.3 Load Balance Algorithms

Load balance becomes an increasingly significant factor in determining the run time of parallel applications as core count increases. An inadequate work distribution can prevent efficient scaling and cause some compute nodes to sit idle or run out of memory. Furthermore, a load balancer that scales poorly to large machines will lead to a noteworthy amount of compute time spent re-distributing work units rather than advancing the simulation. To address this challenge, we developed two novel load balance algorithms. The first uses a gap-aware structured grid decomposition mapped onto a 3D process grid that is efficient and produces work that maps well onto torus architectures. The second is a recursive bisection load balancing scheme that is memory lean, fast, and highly scalable.

4.3.1 Grid Load Balance Algorithm

In order to balance the demands of performance against memory limitations, we have devised a lightweight load balance algorithm that distributes work in stages. Tasks are mapped onto a three-dimensional process grid to simplify communication and limit local data sizes during set up. Each step is carried out iteratively until the maximum estimated workload on any task is as small as possible. To limit memory consumption, fluid points are identified in one dimensional strips, by first identifying which points in the

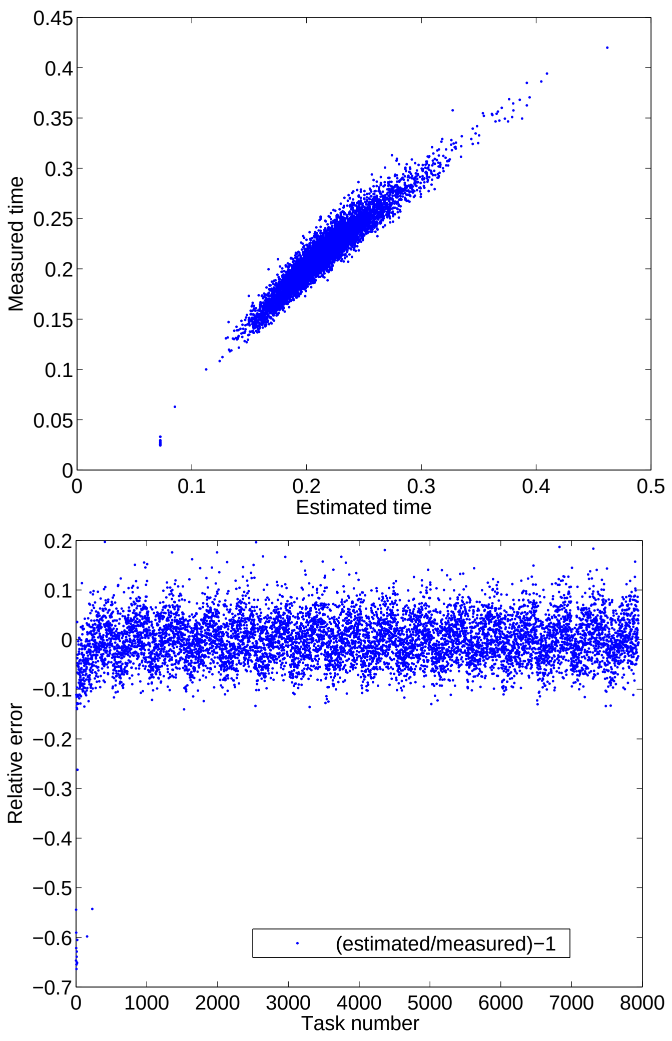

Figure 2: Accuracy of simplified performance model, $C^* = a^* \cdot n_{\mathbf{fluid}} + \gamma^*$. Upper: Measured vs. estimated time. Lower: Relative error in estimated time.

strip border surface mesh triangles and then using angle-weighted pseudonormals[2] to determine which points are on the interior of the surface. The space between interior points is then filled with fluid nodes, which are assigned to processors in that direction of the process grid.

A more detailed description of the algorithm follows:

1. Distribute xy-planes of grid across process planes
2. Compute interior grid points from surface mesh.
3. Estimate work of each xy-plane.
4. Reassign ownership of xy-planes, recompute interior grid points.
5. Estimate work of local xy-planes as a function of y.
6. Assign y-strips of grid points to y-strips of tasks.
7. Distribute strips across tasks in x-direction.

From the measurements described in Section 4.2, it is reasonable to assume that work is proportional to the number of fluid nodes owned by a given task. Although the computational time in an indirect-addressing loop is insensitive to the bounding box volume, communication and, more importantly, memory can be impacted if bounding boxes become too large. We thus explicitly forbid bounding boxes from spanning more than a few exterior points, so that tasks do not end up owning points on multiple branches in the same plane.

4.3.2 Recursive Bisection Load Balance Algorithm

The basic recursion premise is as follows: Assume that the domain is geometrically confined to a brick shaped region, and the computational work for this domain is to be divided among P tasks. A cut parallel to one of its sides is made through the brick, dividing the region into two parts (see Fig. 3). The cores are divided into two groups of roughly equal size and half of the brick is assigned to each group. The optimization problem lies in assigning equal work per task, i.e. solving $N2 * C(S1) = N1 * C(S2)$, where C is the cost function, $N1$ and $N2$ are the number of cores in each group, and $S1$ and $S2$ are the two parts into which the computational domain was cut. After the cut is made, and core groups are assigned, the load balancer is called again for each of the two groups. The subdivision of a task group into two is done so that the two sub-groups are of as equal size as possible. All subsequent steps are done in parallel, and each task group recursively solves its load balancing problem for its work part independently of the other task group. The load balancing is complete when a task group consists of only one task, after $\mathcal{O}(\log(P))$ steps.

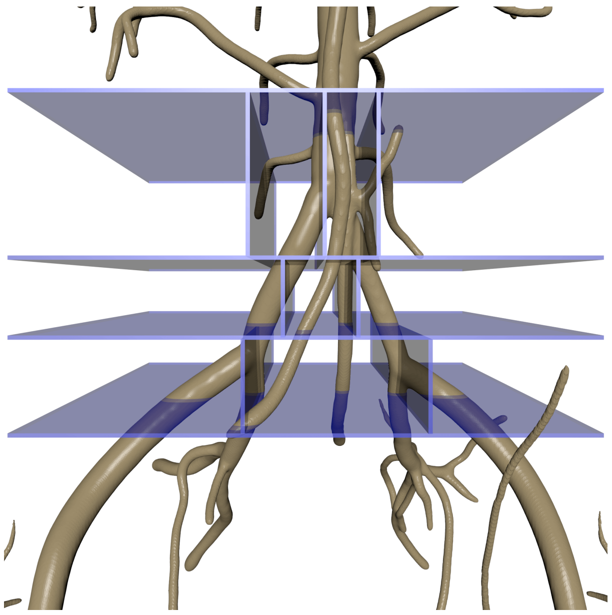

Figure 3: Graphic representation of the recursive bisection load balance algorithm. Cut planes are determined by a histogram of the cost function, and define new subdomains in which load balance is again applied along the longest axial dimension.

Any cost function, $C(x)$, of one variable that is non-decreasing and which describes the work in part $S1$ for a cut x can be used. A simple example is $C(x)$ = the number of grid points to the left of the cut. This choice would ensure that each task gets the same number of grid points. For this work, we used a cost function consisting of a weighted combination of the different node types plus a term proportional to the local bounding box volume.

The work partitioning algorithm is as follows:

1. Compute histogram of cost function in direction of cut dimension.
2. Reduce histogram to obtain total work in task group's box.
3. Determine which bin divides total work into almost equal halves.

This process can also be done recursively to refine the cut point to a given accuracy. We used 32 bins, and 5 iterations of the above algorithm. This achieves a cutting plane with the fidelity of a single precision floating point number. Eleven iterations would yield a data resolution of a double precision floating point number. Assuming N simulation data is roughly evenly distributed over P tasks, the cost of the above algorithm is $O(N/Plog_b(1/\epsilon))$, where b is the number of bins, and ϵ the required precision of the cutting plane position relative to the length of the simulation box along the cutting axis.

Once the cutting plane has been determined, a reduction operation is used to determine whether the data is sufficiently balanced and ensure that a data exchange will not cause any tasks to run out of memory. If necessary, data is redistributed (leveled) so that each task has the same amount of work assigned. Each task then divides its data into two sets, one containing the data to remain on the task subgroup to which this task belongs, and one with the rest of the data. After this, each task picks a companion task in the other task group, and exchanges data with that task using point-to-point messaging. At this point, the load balancer is called recursively for each task subgroup until all subgroups consist of a single task.

4.4 Single Node Optimization

The most computationally intense routine in our circulatory model calculates both collision and equilibrium relaxation. It is both heavily exercised and representative of a large number of codes with similar characteristics regarding both massive multithreading and utilization of SIMD floating-point engines.

The code was modified to both take advantage of system architectural features and avoid potential performance pitfalls for the computation of three primary components: task distribution, density and momentum vector, and collision and equilibrium relaxation.

First, there is the task distribution determination. This is a fairly simple section of code, as we attempt to evenly distribute tasks (by count) to threads. One potential risk is employing a methodology such as farming out Ceiling(tasks/threads) until no tasks are left to distribute. This works well if there are large numbers of tasks or a small number of threads,

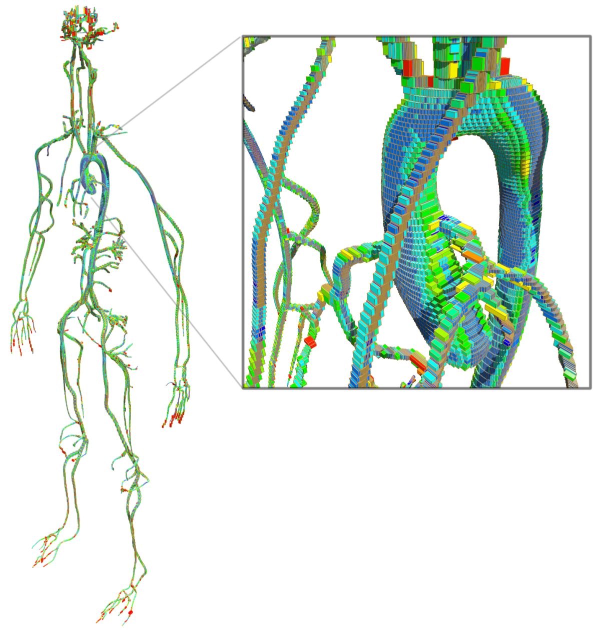
Figure 4: Image of the bounding boxes computed by the grid load balance algorithm. The color corresponds to the bounding box volume, from green (smallest) to red (largest).

but can greatly impact performance in the strong scaling limit. Of smaller concern is the fact that the master thread has more work to do. Thus, it is usually prudent to assign thread 0 the lightest load possible and advisable to progress from lower to higher task count as work is assigned to increasing thread ids.

Next, there is the computation of the density and vector function. In order to make effective use of SIMD operations for this section of the code, we copied the data from the discrete velocity direction and degeneracy structures containing the increment values to an aligned array. Further, we did this so that the 3-component velocities and associated degeneracies for a given stencil were adjacent; a good match for our 4-way SIMD registers and operators. If this data is in the L1 cache upon entry to the routine, it can be fed to the SIMD FMAs and the operation can continue at almost the peak rate of the system (there are some non-SIMD components). However, if the data is not in the L1 cache, it should be efficiently prefetched, likely from the L2 cache, yielding between 33% of peak if the bulk of the data resides in the main memory and $>$ 80% of peak if it resides in the L2 cache for the SIMDized components.

The code completes by computing the actual collision and relaxation values. As the operations used for this computation require the values to be, in a sense, transposed from their orientation in the previous step, we have many options. The simplest path, is to transpose the data explicitly. There are a few ways to do this, but the differences (related to what will result insofar as outstanding load distributions are concerned) are only important when the data is too large to fit in the L1D cache and that is not the case for our stencils. However, since the L1D cache is write-through, the advantage from an explicit transpose in this case is somewhat reduced. The more efficient path and one we will explore for the 19-point stencil (and, perhaps, for the higher-order 39-point stencil, though this is made more difficult as there are more points than SIMD registers in our system in this case) is to permute the vectors while they are still in registers. While the permutes consume instruction cycles on the floating-point unit, we believe that this path or a hybrid approach wherein some data is permuted in registers while other data is permuted via a copy to a tiny array, thus evening out the load between the two executions units, will be a performance win in the strong scaling extreme.

5. PERFORMANCE RESULTS

In this section, we present strong scaling measurements on the full 1.5 million core LLNL Sequoia Blue Gene/Q machine. We discuss the relevant hardware details, analyze the impact of our kernel optimizations, and present strong scaling results for the full systemic arterial network.

5.1 The Blue Gene/Q Compute Node Architecture

While the Blue Gene/Q compute node contains many features, such as hardware support for speculative multithreading [13], here we provide a summary concentrating on the architectural features that were critical to achieving high performance for our application.

The Blue Gene/Q compute node includes 16 user-space compute cores, based upon the PowerPC A2 processor. The core is designed for high-performance computing and high-performance analytics capability. This purpose-based design encompasses the entire hardware gamut, from the novel SIMD architecture, through the programmable L1 cache prefetch units, and the large, multi-versioned L2 cache, to the integrated networking unit.

Because of the demands of traditional high-performance computing, it was important that the Blue Gene/Q cores have high floating-point compute capacity. To that end, the QPX floating-point engine [13], with the ability to execute a 4-way SIMD fused-multiply-add (FMA) instruction every cycle, was designed. The QPX unit is not only an improvement, in terms of raw capability, over the previous Blue Gene/L and Blue Gene/P FPUs [37], but, in order to provide a method to address the fact that data is often not aligned or not stored in the appropriate format for simple vector operations, the QPX unit also provides comprehensive permute and duplicate register-to-register operations. At a frequency of 1.6 GHz, each Blue Gene/Q core is capable of 12.8 GFLOPS, yielding a peak performance of 204.8 GFLOPS per node.

Floating-point capacity is not of great utility if one cannot feed the compute engines useful data at the appropriate rate. This is especially true for an energy-conserving, in-order architecture such as the BG/Q compute chip. While out-of-order execution units are generally considered to be far more forgiving of stalls, as they can execute any of a number of possible instructions, on Blue Gene/Q this issue was addressed through a different, holistic set of measures. The first is the 4-way symmetric multi-threading (SMT) design of the A2. In any given cycle the two functional units (floating-point and integer/load-store) can execute concurrently, as long as there are two threads executing on the core. With hardware SMT, an eight-entry (shared) outstanding load queue, and independent register files for all threads, high latency tolerance is achievable for many applications.

Hardware caches can be used to both reduce latency and increase bandwidth available in compute-intensive portions of the code. Blue Gene/Q cores each contain a 16KB L1 data cache and a 32-entry prefetch buffer that can be used to manage prefetching of up to 16 streams of data. The next layer of cache is the large (32 MB), high-bandwidth (in excess of 8 bytes per core per clock cycle), L2 cache.

System construction was a pervasive concern in the Blue Gene/Q design and the network and messaging unit on each node of the system reflects this. There are several networks on the system, but the network most heavily used to communicate data in scientific codes is the five-dimensional (5D) torus. Through this network, each Blue Gene/Q code can simultaneously send and receive data at 40 GB/s aggregate through 10 chip-to-chip links.

5.2 SIMD and Threading Performance Analysis

The results of our single node optimizations can be seen in Fig. 5. It shows the performance results for four optimization stages of the LBM kernel on up to 16384 MPI tasks of the IBM Blue Gene/Q. These smaller scale studies were conducted to analyze the reduction of time-to-solution for a reduced vessel geometry in order to optimally tune the code for the large-scale simulations shown in Fig. 6.

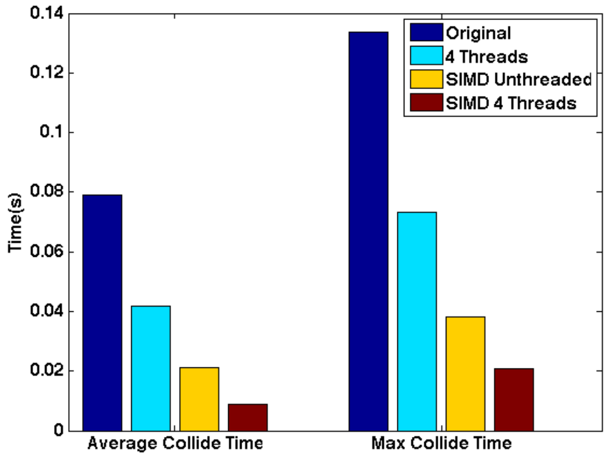

Figure 5: Performance of the optimized collide kernel on 16,384 MPI Tasks of the IBM Blue Gene/Q.

In this case, simulations of a human aorta were conducted at a 20 μm resolution. As discussed in Section 4.4, we focused our single node optimizations on the routine accounting for the collision and equilibrium relaxation. For consistency, only time spent in this kernel was taken into account to study the performance impact of the optimizations. As expected, the original unoptimized kernel is the slowest, followed by the threaded, the unthreaded SIMD version, and finally the threaded SIMD kernel. The SIMD threaded kernel outperformed the original implementation by 89% and without SIMD by 79%, respectively. It also improved by 84% and 71% in terms of the average and maximum time spent in the collide kernel, respectively.

5.3 Strong Scaling to 1,572,864 Tasks

Figure 6 shows the strong scaling of the full arterial geometry at 20 μm resolution run on 8,192 to 98,304 nodes of the Sequoia Blue Gene/Q machine. We observed a speedup of 5.2x over a 12x increase in node count, corresponding to a parallel efficiency of 43%. Load imbalance ranged from 41% to 162% with the grid balance algorithm and from 57% to 193% with the bisection algorithm, indicating that in both cases the performance model needs improvement once work is being distributed across more than a million tasks. The improved performance translates to an unprecedented time-to-solution for large-scale blood flow simulations. The time spent in each iteration is shown in Table 2. The fastest time per iteration was less than 0.2 sec, a significant decrease in time-to-solution over the previous state-of-the-art for a large, arterial geometry at high-resolution.

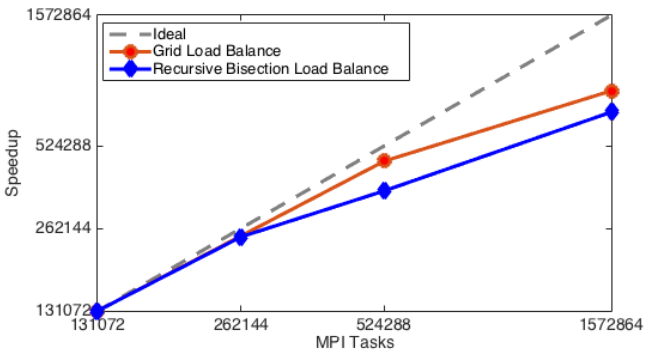

Figure 6: Strong scaling of 20μm resolution systemic arterial geometry for each load balance algorithm.

MPI Tasks	Iteration time (s)
262,144	0.46
524,288	0.31
1,572,864	0.17

Table 2: Time-to-solution of systemic arterial geometry at 20 μm resolution on up to 1,572,864 cores of the IBM Blue Gene/Q supercomputer, using the grid load balance algorithm described in Section 4.3.

Figure 7 shows the weak scaling and load imbalance of the full arterial geometry. In order to reach a resolution of 9 μm on the full machine, it was necessary to implement a very lightweight initialization routine in which all surface mesh and fluid data was fully distributed at all times and interior points computed from single-bit xor operations to avoid exceeding the total memory of any given task while the initial data was distributed across tasks. This highly distributed approach was only compatible with the bisection load balancer, but allowed us to achieve unprecedented resolution for a system of this size. This low-memory footprint positions us well to incorporate multiphysics models such as deformable suspended bodies or move to even higher resolutions in the future, as larger systems are deployed.

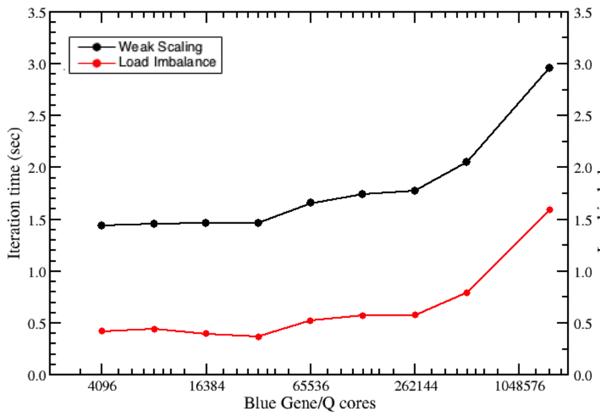

Figure 7: Weak scaling (black) and load imbalance (red) of systemic arterial geometry using the bisection load balancer. The grid resolution was adjusted to keep the average number of fluid nodes per core as constant as possible, from 65.7 μm and 1.3 billion fluid nodes on 4,096 Blue Gene/Q cores up to 9 μm and 509.0 billion fluid nodes on 1,527,864 cores.

Load imbalance, which we define as the difference between the average time and the maximum time spent in the iteration loop normalized by the average iteration time, was a significant obstacle to strong scaling for both load balance algorithms on the full 1.57 million core Sequoia machine. Fig. 8 shows both the load imbalance and communication costs at scale for the 20 μm arterial geometry using the grid load balance algorithm. The communication costs remain fairly constant for both the average and maximum times spent in communication. For these systems, it is load imbalance and not relative communication costs that inhibit strong scaling. The deviation seen in Fig. 6 from ideal scaling is in fact due almost entirely to load imbalance, specifically the number of fluid points involved in each task's stream and collide routines. Because the load balance algorithms try to equalize only the number of fluid nodes owned by each task, increasingly large discrepancies in the actual work load arise at large scale. To improve load balance at these scales, we will need a cost model that takes into account the costs of work supplied by neighboring fluid points, e.g. by including a surface area term in addition to a volume term in our work function.

The best performance metric for the LBM is *million fluid lattice updates per second* (MFLUP/s), which allows direct measurement of the amount of work completed in the simulation by counting only fluid nodes that are actually processed by the compute kernel. Such a metric is especially important when considering the sparse geometries compos-

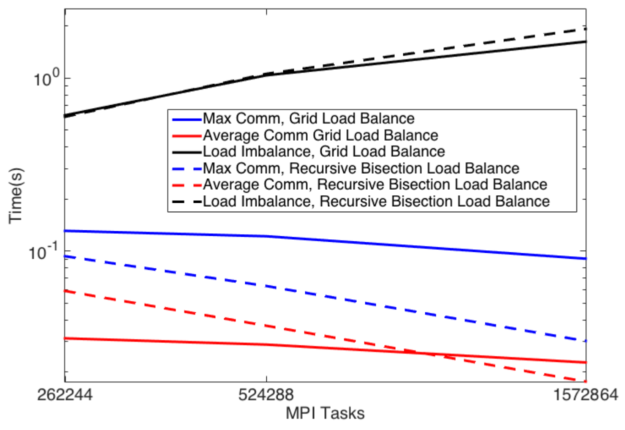

Figure 8: Communication and load imbalance for 20 μm resolution systemic arterial geometry using the grid load balancer.

Geometry	MFLUP/s	Citation
Coronary arteries	$1.14 \cdot 10^5$	[26]
Coronary arteries	$7.19 \cdot 10^4$	[3]
Coronary arteries	$1.29 \cdot 10^6$	[10]
Aortofemoral	$1.28 \cdot 10^5$	[30]
Systemic arterial	$2.99 \cdot 10^6$	Presented here

Table 3: Demonstrated improvement over state-of-the art achieved MFLUP/s.

ing the human vasculature. Table 3 shows estimated number of MFLUP/s for seminal papers leveraging the LBM for large-scale hemodynamic simulations. The improved time-to-solution mentioned above leads to a performance of $2.99 * 10^6$ an MFLUP/s for the full arterial geometry simulated at a 20μm resolution. **We note that in the more complex geometry of the systemic arterial network, we were able to achieve twice the MFLUP/s performance as previous work modeling flow in the coronary arteries** [10]. .

6. IMPLICATIONS FOR FUTURE SYSTEMS AND APPLICATIONS

A property of vascular geometries is the small fraction of space the actual simulated region covered in the overall bounding box. Moreover, this sparse geometry represents an increasingly complex domain as we move to larger and larger regions of the human vasculature. In order to identify risk factors for vascular disease and improve upon current diagnostic tests such as the ABI, we need to not only be able to model the systemic arterial network, but do so at high-resolution, under a wide range of physiological conditions, and for long time scales. This underscores the necessity to focus on the reduction in time-to-solution in order to enable such simulations to lead to future clinical impact. Furthermore, the data structures and optimization techniques introduced here have a wide applicability to other stencil-based codes addressing sparse geometries such as encountered in models of subsurface flow, biosecurity (e.g. the subway problem), and microfluidics.

As both core counts and hardware heterogeneity continue to increase on the path to exascale, codes will have to be able to not only support massive parallelism but also make efficient use of a wide range of hardware-specific kernels. Our implementation demonstrates our ability to do both. However, the increasing variety in node architecture and the increasing power of accelerators will require more flexibility in terms of how work is distributed across the machine. Implementing a hierarchical blocked data structure along with more flexible and robust load balance algorithms will likely be needed before we can take full advantage of the next generation of supercomputing hardware.

By substantially improving the parallel scalability and reducing the time-to-solution, we enable the simulation of image-based hemodynamics on an unprecedented geometric resolution, the systemic arterial system at a resolution corresponding to the size of a biological cell, and a timescale that allows for a new level of risk stratification on a per-patient basis. The benefit is two-fold: (1) As some circulatory phenomena manifest over relatively long time scales, reducing the time-to-solution of these simulations facilitates the study of groundbreaking lengths, and (2) The results of diagnostic tests such as the ABI are strongly influenced by conditions such as exercise, rest, temperature, altitude, or co-existing conditions like anemia. The ability to model several hundred cardiac cycles in a shortened wall clock time makes it feasible to run a range of simulations at full-scale, thus providing physicians with a more complete picture of a patient's vascular risk. This paper, has demonstrated a key step in advancing patient-specific hemodynamics by introducing the first high-resolution simulation of the systemic arterial network at a groundbreaking time-to-solution.

7. ACKNOWLEDGMENTS

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. Research reported in this publication was supported by the Office Of The Director, National Institutes Of Health of the National Institutes of Health under Award Number DP5OD019876. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health. The authors would like to thank Livermore Computing for their outstanding support and assistance, Bob Walkup at IBM for his guidance in memory optimization and Dr. Jane Leopold at the Brigham and Women's Hospital for discussion and guidance regarding the medical application.

8. REFERENCES

[1] J. Alastruey, A. W. Khir, K. S. Matthys, P. Segers, S. J. Sherwin, P. R. Verdonck, K. H. Parker, and J. Peiró. Pulse wave propagation in a model human arterial network: Assessment of 1-d visco-elastic simulations against *in vitro*measurements. *Journal of Biomechanics*, 44(12):2250–2258, 2011.

[2] J. Baerentzen and H. Aanaes. Signed distance computation using the angle weighted pseudonormal. *IEEE Transactions on Visualization and Computer Graphics*, 11(3):243–253, 2005.

[3] M. Bernaschi, M. Bisson, T. Endo, S. Matsuoka, M. Fatica, and S. Melchionna. Petaflop biofluidics

simulations on a two million-core system. In *Proceedings of the 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, page 4. ACM, 2011.

[4] J. Berry, W. Borden, D. Bravata, S. Dai, E. Ford, et al. Heart disease and stroke statistics?2012 update. *Circulation*, 125:e2–e220, 2012.

[5] J. Carter, M. Soe, L. Oliker, Y. Tsuda, G. Vahala, L. Vahala, and A. Macnab. Magnetohydrodynamic turbulence simulations on the earth simulator using the lattice Boltzmann method. In *Proceedings of the 2005 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '05. IEEE Computer Society, 2005.

[6] J. Cebral, M. Castro, J. Burgess, R. Pergolizzi, M. Sheridan, and C. Putman. Characterization of cerebral aneurysms for assessing risk of rupture by using patient-specific computational hemodynamics models. *American Journal of Neuroradiology*, 26(10):2550–2559, 2005.

[7] A. B. I. Collaboration et al. Ankle brachial index combined with Framingham risk score to predict cardiovascular events and mortality: a meta-analysis. *JAMA: the Journal of the American Medical Association*, 300(2):197, 2008.

[8] J. S. Coogan, J. D. Humphrey, and C. A. Figueroa. Computational simulations of hemodynamic changes within thoracic, coronary, and cerebral arteries following early wall remodeling in response to distal aortic coarctation. *Biomechanics and modeling in mechanobiology*, 12(1):79–93, 2013.

[9] A. V. Doobay and S. S. Anand. Sensitivity and specificity of the ankle–brachial index to predict future cardiovascular outcomes a systematic review. *Arteriosclerosis, Thrombosis, and Vascular Biology*, 25(7):1463–1469, 2005.

[10] C. Godenschwager, F. Schornbaum, M. Bauer, H. Köstler, and U. Rüde. A framework for hybrid parallel flow simulations with a trillion cells in complex geometries. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, page 35. ACM, 2013.

[11] L. Grinberg, D. Fedosov, and G. Karniadakis. Parallel multiscale simulations of a brain aneurysm. *Journal of Computational Physics*, 2012.

[12] L. Grinberg, V. Morozov, D. Fedosov, J. Insley, M. Papka, K. Kumaran, and G. Karniadakis. A new computational paradigm in multiscale simulations: Application to brain blood flow. In *Proceedings of the 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–12. IEEE, 2011.

[13] R. Haring, M. Ohmacht, T. W. Fox, M. K. Gschwind, D. L. Satterfield, K. Sugavanam, P. W. Coteus, P. Heidelberger, M. A. Blumrich, R. W. Wisniewski, et al. The IBM Blue Gene/Q compute chip. *Micro, IEEE*, 32(2):48–60, 2012.

[14] M. Hecht and J. Harting. Implementation of on-site velocity boundary conditions for D3Q19 lattice Boltzmann simulations. *Journal of Statistical Mechanics: Theory and Experiment*, 2010(01):P01018, 2010.

[15] F. Higuera, S. Succi, and R. Benzi. Lattice gas dynamics with enhanced collisions. *EPL (Europhysics Letters)*, 9(4):345–349, 1989.

[16] T. J. Hughes and J. Lubliner. On the one-dimensional theory of blood flow in the larger vessels. *Mathematical Biosciences*, 18(1):161–170, 1973.

[17] O. Malaspinas, B. Chopard, and J. Latt. General regularized boundary condition for multi-speed lattice Boltzmann models. *Computers & Fluids*, 49(1):29–35, 2011.

[18] A. L. Marsden, A. J. Bernstein, V. M. Reddy, S. C. Shadden, R. L. Spilker, F. P. Chan, C. A. Taylor, and J. A. Feinstein. Evaluation of a novel y-shaped extracardiac fontan baffle using computational fluid dynamics. *The Journal of Thoracic and Cardiovascular Surgery*, 137(2):394–403, 2009.

[19] G. McNamara and G. Zanetti. Use of the Boltzmann equation to simulate lattice-gas automata. *Physical Review Letters*, 61(20):2332–2335, 1988.

[20] S. Melchionna, M. Bernaschi, S. Succi, E. Kaxiras, F. J. Rybicki, D. Mitsouras, A. U. Coskun, and C. L. Feldman. Hydrokinetic approach to large-scale cardiovascular blood flow. *Computer Physics Communications*, 181(3):462–472, 2010.

[21] J. M. Murabito, R. B. D?Agostino, H. Silbershatz, and P. W. Wilson. Intermittent claudication a risk profile from the framingham heart study. *Circulation*, 96(1):44–49, 1997.

[22] J. M. Murabito, J. C. Evans, M. G. Larson, K. Nieto, D. Levy, and P. W. Wilson. The ankle-brachial index in the elderly and risk of stroke, coronary disease, and death: the framingham study. *Archives of Internal Medicine*, 163(16):1939–1942, 2003.

[23] C. M. Papamichael, J. P. Lekakis, K. S. Stamatelopoulos, T. G. Papaioannou, M. K. Alevizaki, A. T. Cimponeriu, J. E. Kanakakis, A. Papapanagiotou, A. T. Kalofoutis, and S. F. Stamatelopoulos. Ankle-brachial index as a predictor of the extent of coronary atherosclerosis and cardiovascular events in patients with coronary artery disease. *The American journal of cardiology*, 86(6):615–618, 2000.

[24] K. Pekkan, B. Whited, K. Kanter, S. Sharma, D. De Zelicourt, K. Sundareswaran, D. Frakes, J. Rossignac, and A. Yoganathan. Patient-specific surgical planning and hemodynamic computational fluid dynamics optimization through free-form haptic anatomy editing tool (surgem). *Medical & Biological Engineering & Computing*, 46(11):1139–1152, 2008.

[25] C. S. Peskin. Numerical analysis of blood flow in the heart. *Journal of Computational Physics*, 25(3):220–252, 1977.

[26] A. Peters, S. Melchionna, E. Kaxiras, J. Lätt, J. Sircar, M. Bernaschi, M. Bison, and S. Succi. Multiscale simulation of cardiovascular flows on the IBM Blue Gene/P: Full heart-circulation system at red-blood cell resolution. In *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '10, 2010.

[27] A. Peters Randles, V. Kale, J.R. Hammond, W. Gropp, and E. Kaxiras. Performance analysis of the lattice Boltzmann model beyond Navier-Stokes. In *Proceedings of the 27th IEEE International Parallel and Distributed Processing Symposium*, IPDPS '13, 2013.

[28] T. Pohl, F. Deserno, N. Thurey, U. Rude, P. Lammers, G. Wellein, and T. Zeiser. Performance evaluation of parallel large-scale lattice Boltzmann applications on three supercomputing architectures. In *Proceedings of the 2004 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '04. IEEE Computer Society, 2004.

[29] I. Rahimian, A.and Lashuk, S. Veerapaneni, A. Chandramowlishwaran, D. Malhotra, L. Moon, R. Sampath, A. Shringarpure, J. Vetter, R. Vuduc, et al. Petascale direct numerical simulation of blood flow on 200k cores and heterogeneous architectures. In *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–11. IEEE Computer Society, 2010.

[30] A. Randles, E. Draeger, and P. Bailey. Massively parallel simulations of hemodynamics in the primary large arteries of the human vasculature. In *Journal of Computational Science*, ICCS15, 2015. accepted.

[31] H. E. Resnick, R. S. Lindsay, M. M. McDermott, R. B. Devereux, K. L. Jones, R. R. Fabsitz, and B. V. Howard. Relationship of high and low ankle brachial index to all-cause and cardiovascular disease mortality the strong heart study. *Circulation*, 109(6):733–739, 2004.

[32] P. Reymond, F. Merenda, F. Perren, D. Rüfenacht, and N. Stergiopulos. Validation of a one-dimensional model of the systemic arterial tree. *American Journal of Physiology-Heart and Circulatory Physiology*, 297(1):H208–H222, 2009.

[33] G. D. Smith, M. Shipley, and G. Rose. Intermittent claudication, heart disease risk factors, and mortality. the Whitehall study. *Circulation*, 82(6):1925–1931, 1990.

[34] N. Stergiopulos, D. Young, and T. Rogge. Computer simulation of arterial flow with applications to arterial and aortic stenoses. *Journal of Biomechanics*, 25(12):1477–1488, 1992.

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

[36] C. Taylor, T. Hughes, and C. Zarins. Finite element modeling of blood flow in arteries. *Computer Methods in Applied Mechanics and Engineering*, 158(1):155–196, 1998.

[37] I. B. G. team. The IBM Blue Gene project. *IBM Journal of Research and Development*, 57(1/2):0, 2013.

[38] N. Westerhof, F. Bosman, C. J. De Vries, and A. Noordergraaf. Analog studies of the human systemic arterial tree. *Journal of Biomechanics*, 2(2):121–143, 1969.

[39] S. Williams, L. Oliker, J. Carter, and J. Shalf. Extracting ultra-scale lattice Boltzmann performance via hierarchical and distributed auto-tuning. In *Proceedings of the 2011 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '11, pages 1–10. IEEE Computer Society, 2011.

[40] A. J. Wood and W. R. Hiatt. Medical treatment of peripheral arterial disease and claudication. *New England Journal of Medicine*, 344(21):1608–1621, 2001.

[41] N. Xiao, J. Humphrey, and C. Figueroa. Multi-scale computational model of three-dimensional hemodynamics within a deformable full-body arterial network. *Journal of Computational Physics*, 244:22–40, 2013.

[42] G. Xiong and C. A. Taylor. Virtual stent grafting in personalized surgical planning for treatment of aortic aneurysms using image-based computational fluid dynamics. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2010*, pages 375–382. Springer, 2010.

[43] Q. Zou and X. He. On pressure and velocity boundary conditions for the lattice Boltzmann BGK model. *Physics of Fluids*, 9:1591, 1997.