Towards blood flow in the virtual human: Efficient self-coupling of HemeLB

Journal Article

Author(s):
McCullough, Jon W.S.; Richardson, Robin A.; Patronis, Alex; Halver, Rene; Marshall, R.; Ruefenacht, Martin; Wylie, Brian J.N.; Odaker, Thomas; Wiedemann, Markus; Lloyd, Bryn; Neufeld, Esra; Sutmann, Godehard; Skjellum, Anthony; Kranzlmüller, Dieter; Coveney, Peter V.

Publication date:
2021-02-06

Permanent link:
https://doi.org/10.3929/ethz-b-000465832

Originally published in:
Interface Focus 11(1), https://doi.org/10.1098/rsfs.2019.0119

INTERFACE FOCUS

royalsocietypublishing.org/journal/rsfs

Research

Cite this article: McCullough JWS *et al*. 2021 Towards blood flow in the virtual human: efficient self-coupling of HemeLB. *Interface Focus* **11**: 20190119.
http://dx.doi.org/10.1098/rsfs.2019.0119

Accepted: 23 September 2020

One contribution of 11 to a theme issue 'Computational biomedicine. Part II: organs and systems'.

Subject Areas:
biomedical engineering, medical physics, computational biology

Keywords:
high-performance computing, blood flow modelling, virtual human, lattice Boltzmann method

Author for correspondence:
P. V. Coveney
e-mail: p.v.coveney@ucl.ac.uk

Electronic supplementary material is available online at https://doi.org/10.6084/m9.figshare.c.5202157.

THE ROYAL SOCIETY PUBLISHING

Towards blood flow in the virtual human: efficient self-coupling of HemeLB

J. W. S. McCullough[1], R. A. Richardson[1], A. Patronis[1,2], R. Halver[2], R. Marshall[3], M. Ruefenacht[3], B. J. N. Wylie[2], T. Odaker[4], M. Wiedemann[4], B. Lloyd[5], E. Neufeld[5], G. Sutmann[2,6], A. Skjellum[3], D. Kranzlmüller[4] and P. V. Coveney[1,7]

[1]Centre for Computational Science, Department of Chemistry, University College London, London, UK
[2]Jülich Supercomputing Centre, Forschungszentrum Jülich, Jülich, Germany
[3]SimCenter, University of Tennessee at Chattanooga, Chattanooga, TN, USA
[4]Leibniz Supercomputing Centre, Leibniz-Rechenzentrum (LRZ), Garching, Germany
[5]Foundation for Research on Information Technologies in Society (IT'IS), Zurich, Switzerland
[6]ICAMS, Ruhr-University Bochum, Bochum, Germany
[7]Informatics Institute, University of Amsterdam, Amsterdam, The Netherlands

JWSM, 0000-0002-9606-0408; RAR, 0000-0002-9984-2720; MW, 0000-0002-3366-0537; PVC, 0000-0002-8787-7256

Many scientific and medical researchers are working towards the creation of a virtual human—a personalized digital copy of an individual—that will assist in a patient's diagnosis, treatment and recovery. The complex nature of living systems means that the development of this remains a major challenge. We describe progress in enabling the HemeLB lattice Boltzmann code to simulate 3D macroscopic blood flow on a full human scale. Significant developments in memory management and load balancing allow near linear scaling performance of the code on hundreds of thousands of computer cores. Integral to the construction of a virtual human, we also outline the implementation of a self-coupling strategy for HemeLB. This allows simultaneous simulation of arterial and venous vascular trees based on human-specific geometries.

1. Introduction

The human body is comprised of several complex and interacting subsystems that, in concert, determine its full operation. Each of these depend on mechanisms that span multiple spatial and temporal scales, from sub-cellular processes to directly observable macroscopic properties. The behaviour of these systems is influenced by individual factors such as age, gender, genetics, environment and medical history. All of these must be considered when a patient presents to a clinician for treatment. The development of a virtual human—a digital replica of an individual and their physiological processes—will assist these decisions by allowing multiple courses of treatments to be considered and the optimal one enacted [1–4].

The transport of blood around the body is integral to physiological function. Vessels transporting blood to and from the heart connect tissue, organs and muscle, providing the oxygen and nutrients needed for their operation. This fundamental nature of the vasculature makes it a pivotal component in the development of a virtual human and is the focus of the present work. The extensive computational and data requirements of modelling a full virtual human will require the resources of next-generation exascale supercomputers. Taking full advantage of these necessitates developing efficient simulation codes and strategies for communication between them on the largest current machines.

Many previous studies of large sections of the human vasculature use a 1D solver to capture the blood flow in some or all of the vessels [5–8]. While this can be an efficient approach for simulating large, complex networks, it makes many assumptions about the flow behaviour within a vessel. Modelling 3D

flow is more computationally demanding but allows high-fidelity analysis of flow within all vessels. Coupled 1D–3D models offer a compromise but still do not resolve all features. Full 3D modelling permits local flow features to be identified that are not possible in lower-dimensional models, such as wall shear stress distribution throughout the surface of arteries. The use of a 3D model also permits exact simulation of an individual's vasculature. 1D models generally assume that vessels have a circular cross-section that may vary between neighbouring nodes and over time. Even with patient-specific dimensions, these structural assumptions will lead to homogeneity of solutions between individuals. The use of a 3D model allows simulation results to be exactly constructed for a specific geometry without such assumption.

HemeLB [9,10] is an open-source 3D, lattice Boltzmann based, fluid dynamics solver for the study of blood flow in complex geometries. It has been developed in C++ for, currently, CPU cores only. Unlike many other, more general, open-source lattice Boltzmann codes (e.g. Palabos, TCLB, OpenLB and waLBerla [11–14]), HemeLB has been specifically optimized to enable efficient simulation of the sparse geometries that are characteristic of the vascular networks. Since first being published in 2008 [15], HemeLB has been used to study a number of different aspects of the cardiovasculature, including cerebral flow, retinal flow, vascular remodelling and magnetic drug targeting [16–18].

The development of a virtual human is an ongoing process that will require many computational and algorithmic developments to accurately and efficiently capture physiological behaviour. The purpose of the present paper is to discuss some recent advancements in HemeLB that bring us closer to being able to conduct high-fidelity simulations of the full human vascular tree. In this paper we present proof-of-concept studies that demonstrate the capability of these changes to run a large, 3D flow model on realistic geometries of human-scale arterial and venous trees. These serve as a stepping stone to highlight both the existing capability and the areas which need ongoing development and improvement. The results at this stage are not intended to provide quantitative validation of full-human-scale blood flow. This is a target for future work.

The paper is structured into three further sections. Section 2 discusses the computational advancements that have been made within HemeLB to permit the modelling of full human vasculatures. In particular, we discuss the incorporation of next-generation Message Passing Interface (MPI) developments for data communication, the large-scale performance characteristics of HemeLB and the self-coupling of HemeLB. Section 3 describes the progress that has been made in using the self-coupling of HemeLB to simulate the flow through the arteries and veins of a specific human vasculature. We present results for proof-of-concept models, including an illustrative test case and coupled vascular trees of human legs, important steps towards simulation of high-fidelity full-human models on exascale machines. The paper concludes in §4 with a discussion of pathways to continue with the current work towards the modelling of a full virtual human.

2. Computational advancements

HemeLB solves 3D fluid flow using the lattice Boltzmann method. This approach to solving the Navier–Stokes equations is well known to possess excellent parallel computational efficiency, particularly for bulk flow, and is readily adaptable to complex geometries. The current work uses a D3Q19 lattice with a single relaxation time collision operator with pressure and velocity boundary conditions implemented as appropriate. For a more technical description of the core HemeLB implementation we refer the reader to our previous publications [15,16,19]. Here, HemeLB assumes vessel walls to be rigid and characterizes the blood as a Newtonian fluid, although other rheological models are available [16,20]. Although previous HemeLB studies have successfully captured blood flow with these assumptions [16,20], we recognize that as we progress towards the development of a virtual human the physics captured by HemeLB will need ongoing evaluation. For example it is known that the compliance of vessels has an impact on flow behaviour for both arterial and venous networks and can be a factor in cardiovascular disease.

In order to successfully model a full virtual human, many physiological and biological processes need to be captured over sufficiently long time scales. In the context of blood flow this pertains to the execution of several cardiac cycles. HemeLB focuses on the macroscopic behaviour of vascular transport and requires coupling to other codes to model further system interactions. For example, a simulation of the human heart using Alya Red [21], another high-performance computing code, may also be included and coupled to the boundaries of the arterial/venous tree. High-fidelity resolution of the vasculature of a full human requires extremely large quantities of geometry data (billions of data points). Processing such quantities of data and conducting simulations for physically meaningful time scales demands the resources of cutting-edge supercomputers. Here we discuss recent developments of HemeLB that enable this.

A drawback of the lattice Boltzmann method is that it stores flow information in a memory-intensive manner. HemeLB must initialize and distribute this data within the limits of the available computer. On supercomputers, both of these tasks are non-trivial, meaning that code efficiency is an ever-present development concern. In this section we discuss improvements that have been developed for HemeLB to address these issues and present scaling results to demonstrate their efficacy. Additionally, we outline a framework for the self-coupling of HemeLB that complements these development goals.

2.1. Reduction of data communication within MPI

One significant challenge in modelling the full human vasculature is generating accurate geometries on which to conduct simulations. Based on studies presented in §3, at least 10^8 lattice sites are required in order to simulate flow in the smallest imaged vessels within the same model as large vessels such as the aorta. Efficiently constructing geometries of this magnitude is a significant computational challenge. Attempting to perform this task within the limitations of many supercomputers can cause memory or time restrictions to be exceeded. A particular problem that can be encountered is exceeding the limit on the number of elements that can be communicated within a single MPI operation. We refer to this restriction as the BigCount problem.

BigCount is based on the premise that, in the early years of 32-bit computing, the need for an application to track more than a billion items would be rare or too far into the future

to be practical. In computing environments, a single byte using 8 binary bits can express 2^8 (or 256) unique values, such as the set of integers from 0 to 255, or from −128 to 127. If a multi-process application needs to send an array of 1000 items from one process to another, the sending process would need more than 1 byte to express to the receiving process a count of how many items to expect. Thus, the generic integer type `int` on a 32-bit system was commonly allotted a width of 4 bytes, even as 64-bit hardware and operating systems were entering the market. To use more or less than 4 bytes for an integer type, developers using languages like C must often specify explicitly with keywords such as `long` or `double` for the former, or `char` for the latter. Earlier MPI standards defined counts and displacements to use the generic integer type, which imposed the 32-bit limit on their associated variables.

The first steps towards solving BigCount have already been taken by the introduction of a new datatype called `MPI_Count` into the MPI-3 standard. `MPI_Count` can support up to 128-bit integers, though its support in MPI-3 is limited to a few functions [22]. As described in §2.1.3 below, the MPI-4 standard (currently in draft form and to debut in 2021) [23] will include a solution to BigCount for commonly used functions that impact this research. However, its release in production MPI implementations was too far into the future when this research was developed. Thus, the approaches described next were implemented. Note that work such as this helped motivate the standardization of these operations. Displacements and offsets, values that allow arbitrary access to memory locations in relation to a starting address, are also vulnerable to the 32-bit limit. The `MPI_Offset` type was added to the current standard, designed to store explicit offsets that may be larger than can be expressed with a 32-bit integer. The `MPI_Aint` type is also in the current standard, and is meant to store memory addresses larger than 4 bytes.

A number of transitional measures have been developed to mitigate the BigCount problem until it is solved by the MPI-4 standard. For example, derived datatypes have been used to express and package large counts into contiguous arrays with the introduction of generalized requests [24]. Some work has also been done to try to reduce the overhead by offloading the work to graphical processing units [25]. Recent discussions among members of the MPI Forum about handling BigCount have focused on possible solutions, including function pointers, adding '`_x`' variants to standard functions [26], and using `MPI_Offset` or `MPI_Aint` to handle large displacement values [27]. At a user level, other options for addressing BigCount involve using extra software libraries or making modifications to the existing code. BigMPI is a library that introduces a set of MPI-related functions which accept large counts as `MPI_Count` types that map to MPI-3 calls [28]. BigMPI focuses more on native types over derived types, as its purpose is to support counts that exceed the system-defined maximum (`INT_MAX`) for commonly used functions such as send and receive types, most of the collective operations, and remote memory access functions. BigMPI converts input stored with large types (and associated data) into a series of smaller supported types (and contiguous data) by way of the MPI standard '`_v`' variants like `scatterv` for collectives.

HemeLB would be able to take advantage of exascale machines today, but is hindered by limitations in the input/output (I/O) component of MPI. At scale, HemeLB has the opportunity to use a large number of data items that exceed the 32-bit limit when calling `MPI_File_read`. To work around this current limitation, the data is broken into 32-bit safe segments and read in sequence. Here we describe two methods for achieving this: Method 1, which is functionally identical to application code used in a function called `read_x` (provided in §1 of the electronic supplementary material), and Method 2, an implementation-specific workaround which bypasses the MPI layer and directly uses an underlying I/O module. Schematic outlines are presented in figure 1.

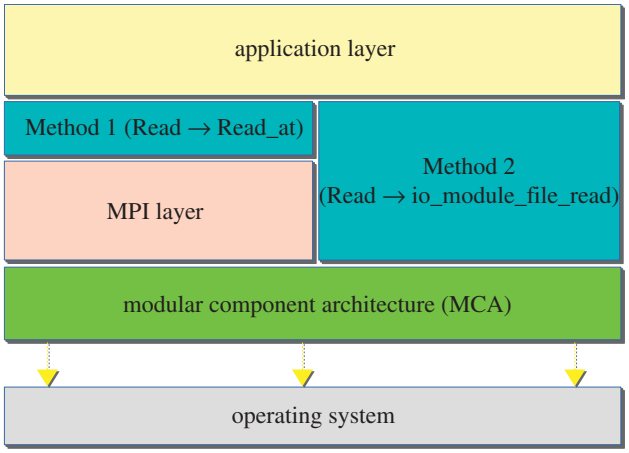

Figure 1. Two methods for working around the 32-bit limitation of MPI communications by breaking large data into segments. Method 1 modifies the existing MPI calls to permit `MPI_Count` types. Method 2 bypasses the I/O section to allow usage of the `MPI_Count` type.

2.1.1. Method 1: Chunking at the MPI API level

The first method emulates a user space method at the application programming interface (API) level that breaks file reading operations into chunks with manageable count values. We implemented it as a proposed MPI function with the signature below.

Listing 1 : MPI file read that accepts the MPI_Count datatype

```
1 MPI_File_read_x (
2     MPI_File fh, void *buf, MPI_Count count,
3     MPI_Datatype datatype, MPI_Status *status);
```

As shown in figure 1, the new function resides on top of the standard `MPI_File_read`, and accepts a `MPI_Count` type for the item count. The function will make a series of calls to the standard `MPI_File_read_at`, transparent to the user. We expect this function to be more convenient for users who regularly deal with large data counts, but cannot modify their MPI implementation. In most cases, we expect this method to perform marginally faster than `read_x()`, since it makes fewer internal MPI calls compared to the user implementation.

2.1.2. Method 2: I/O module bypass of the MPI API

This method is specific to the OpenMPI implementation of MPI, but it could possibly be adapted to other variants. What makes OpenMPI particularly suitable for Method 2 is its structure, which includes the Modular Component Architecture (MCA). By default, OpenMPI uses its base I/O module, OMPIO, to read files. While 32-bit count values are accepted, many of the OMPIO functions could support larger values.

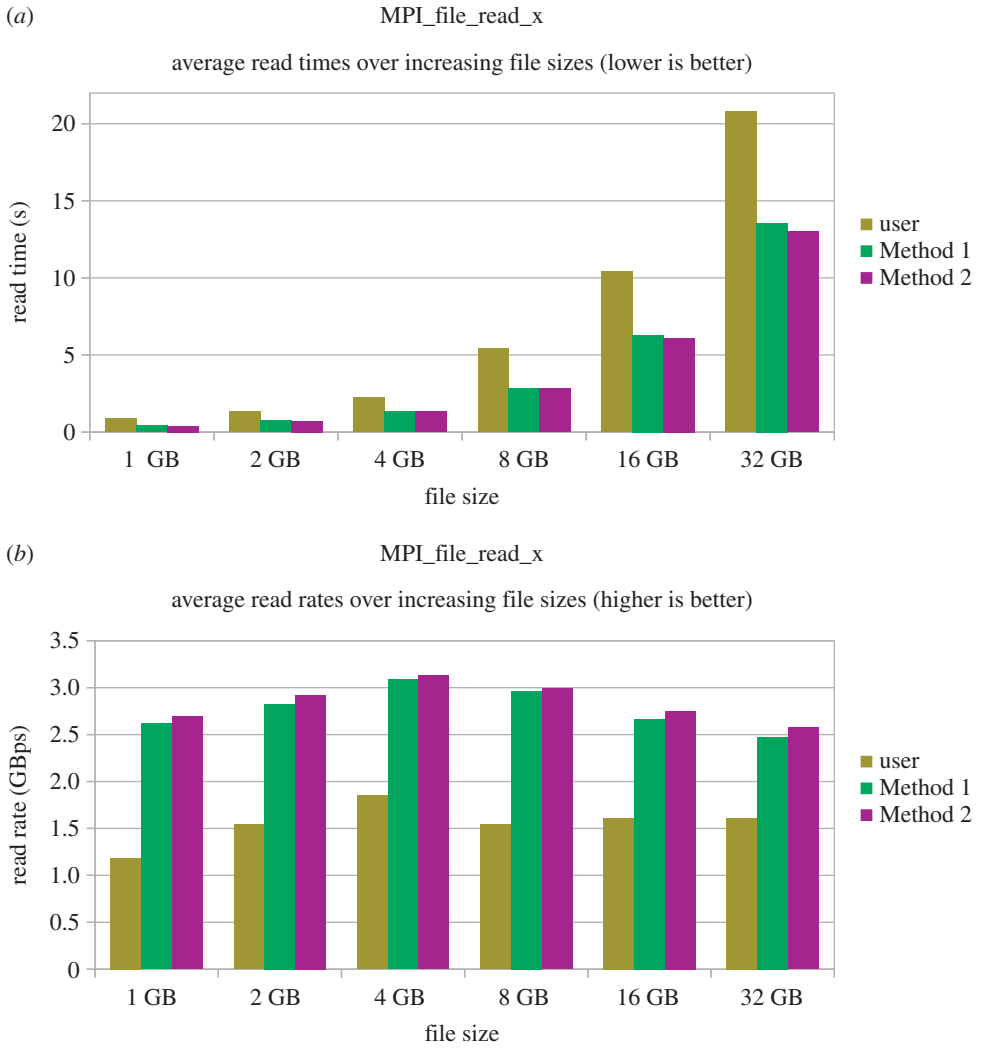

Figure 2. Comparison of the mean read time and read rate observed by user space file chunking versus two different BigCount workarounds in `MPI_File_read_x` when reading files of various sizes. (*a*) Mean read times; (*b*) mean read rates.

We modified the function signatures through the hierarchy of OMPIO functions to accept the `MPI_Count` type, which works as long as the `INT_MAX` value for the system's C library is large enough to support 64-bit integers.

To evaluate the performance of the two methods, we measured the run times of a user space chunking method that avoids large counts against the API layer chunking (Method 1) and the I/O module bypass (Method 2) by reading a series of randomly generated files ranging from 1 to 32 GB in size, and the average time spent in `MPI_File_read_x` over multiple runs.

Figure 2*a* shows the time spent in calls to `MPI_File_read_x` as the file sizes increase, in which both methods are observed to be clearly faster than the user space workaround. This advantage becomes more apparent when read rates are compared (figure 2*b*). Performance improvements realized with these techniques will enable HemeLB to read large datasets more efficiently.

Performance differences aside, the decision to use one method over the other is a practical matter. Both methods require adding `MPI_File_read_x` to the standard, though Method 1 leaves the rest of the MPI implementation untouched. Method 2 is specific to OpenMPI and requires slight modification to the underlying I/O module. In either method, the user only needs to ensure the type for the count is `MPI_Count`.

2.1.3. Standardization in MPI-4

After the completion of the research described here, the MPI Forum adopted the revised model of MPI transfers [23]. This large-count model supports numbers of items to be transferred that are `MPI_COUNT` in size (normally 63-bit signed integers on 64-bit architectures) versus 2^{31} in MPI-3.1, plus it supports all the corollary changes needed to make these work for large transfer support of point-to-point, collective, one-sided, datatype and I/O operations. A second API is provided in both C and Fortran that supports these new, BigCount modes.

2.2. Extreme scale performance

HemeLB executions have been audited on different HPC computer systems by the EU Centre of Excellence Performance Optimisation and Productivity (POP) (https://www.pop-coe.eu/). These performance assessments, based on measurements taken with the highly scalable open-source Scalasca/Score-P toolset [29], found very good computation and communication efficiencies, while identifying memory consumption and load balance as issues to improve.

HemeLB has been previously demonstrated to scale exceptionally well up to 100 000 cores [18]. We refer the reader to Groen *et al.* [9] for a comparison of HemeLB's

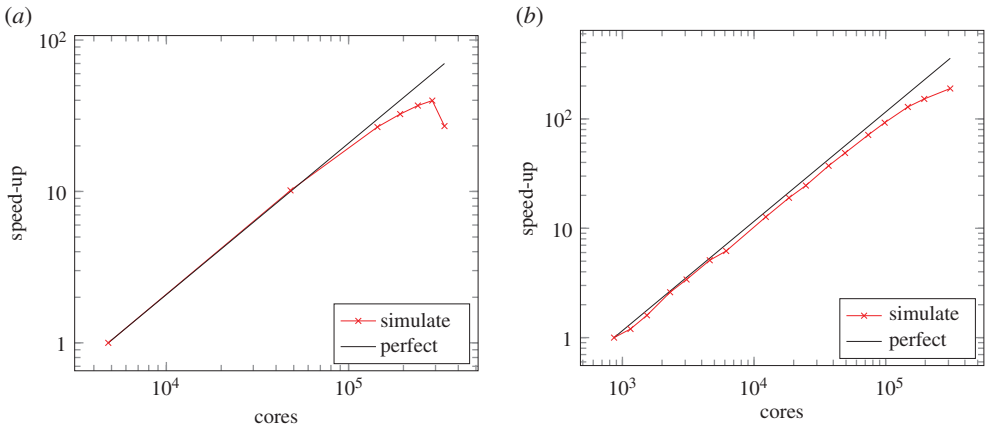

Figure 3. Scaling of HemeLB simulation time for 5000 lattice update steps of the coW-6.4um.gmy dataset on Blue Waters and SuperMUC-NG. Efficient strong scaling is seen on both systems to very large core counts. (*a*) Speed-up relative to 300 nodes (16 cores/node) on Blue Waters; (*b*) speed-up relative to 18 nodes (48 cores/node) on SuperMUC-NG.

performance relative to other lattice Boltzmann codes. Furthermore, the code was recently found to scale with good efficiency on 288 000 AMD 6276 Bulldozer-based Interlagos processor cores of 18 000 Cray XE nodes of NCSA Blue Waters (figure 3*a*). The breakdown in performance at the largest scales on this machine is due to an unfavourable surface-to-volume (communication-to-computation) ratio of partitions (subdomains). Larger models, where computation dominates communication, are required to sustain computational scaling in this regime. In this paper we articulate more recent development on the SuperMUC-NG machine that is based on newer processors with more cores.

The SuperMUC-NG supercomputer [30] at the Leibniz-Rechenzentrum (Germany) comprises 6480 compute nodes with dual 24-core Intel Xeon Platinum 8174 @ 3.10 GHz ('Skylake') processors. The 144 'fat' compute nodes each have 768 GB memory, compared to only 96 GB memory for the remaining 6336 'thin' compute nodes bundled into eight domains (known as 'islands'). The internal interconnect is an Intel OmniPath network, with a fat-tree topology within islands and 1 : 4 pruned connection between islands. A high-performance 50 TB parallel file system is provided by IBM Spectrum Scale (GPFS), with SUSE Linux Enterprise Server (SLES) 12 SP3 operating system.

HemeLB was built with the Intel 19.0.4.243 compiler and MPI library. It was configured to use the MPI Shared Memory model within each compute node to reduce memory requirements when loading the initial lattice data. For scalability testing, a circle of Willis geometry dataset of 21.15 GiB was used (similar to that used in [18] but with a lattice spacing of approximately 6.4 µm). This domain is split into a total of 1 138 236 832 blocks (1376 × 1087 × 761), of which 20 740 240 are non-empty. These non-empty blocks contain a total of 10 154 448 502 lattice sites used for simulation. Excluding the initialization phase and data output, the simulation time for 5000 update steps of blood flow was recorded to establish a strong scaling benchmark on SuperMUC-NG.

The fat compute nodes were used for executions with 864–6144 MPI processes (18–128 compute nodes); thin nodes were used for larger runs. In all cases 48 MPI processes (one per core) were executed on each compute node. Generally, only a single execution was done at each scale, during regular operation of SuperMUC-NG, with run-to-run variation in the simulation time found to be small (potentially due to variation in allocation of compute nodes and communication interaction with other jobs).

Simulation speed-up relative to the smallest execution configuration (18 compute nodes based on memory requirements) for different numbers of compute nodes is plotted in figure 3*b*, along with a comparison to perfect linear scaling. At half machine scale (147 456 cores), a speed-up factor of 128.7 was obtained for a 170.7 factor increase in compute nodes. This corresponds to a scaling efficiency of 75%. At full machine scale (309 696 cores) the speed-up factor increased to 189.8 reducing simulation time to 83 s.

The Scalasca/Score-P assessment of the HemeLB results on SuperMUC-NG showed that, overall, very good computational scaling above 87% is sustained. Efficient non-blocking communication between neighbouring lattice blocks maintains excellent communication efficiency above 97%. The most significant inefficiency at all scales tested is due to the load balance, generally around 80% but dropping to 72% in some larger execution configurations. While this is still fairly good, it presents the largest opportunity for performance improvement and motivated the inclusion of an improved load-balancing framework into the HemeLB code. In §2.2.1, we present initial results for the inclusion of the ALL (A Load-balancing Library) framework.

With 792 compute nodes bundled within island domains, and islands connected via an additional switch, it is notable that no significant simulation performance advantage was observed when inter-domain switches were avoided or reduced. Small numbers of failed compute nodes throughout SuperMUC-NG can therefore conveniently be avoided, allowing full flexibility in allocating partitions.

The HemeLB version used on SuperMUC-NG incorporated optimizations which were essential to be able to set up and run this size of simulation. In particular, these included the use of more memory-efficient data structures and MPI shared memory model for each compute node.

Hoekstra *et al.* [31] observe that the lattice Boltzmann method possesses an algorithmic structure that will enable it to continue its scaling performance on larger supercomputers in the transition to exascale platforms. In part, this is due to the fact that, unlike some algorithms, the lattice Boltzmann

method does not possess a hard limit on scalability that inhibits performance at large scale. This feature assists in enabling HemeLB to study extremely large flow problems.

2.2.1. Load balancing

When conducting a simulation over hundreds or thousands of compute cores, its parallel efficiency depends strongly on how evenly the workload is distributed. In its default form, HemeLB conducts a basic decomposition that can, when partitioning certain geometries, result in an unbalanced workload between cores. In order to better distribute the HemeLB workload, ongoing investigations have been conducted with specific load-balancing libraries. A combination of Zoltan [32,33] and ParMETIS [34] was found to require substantially longer walltimes and more memory due to its extensive preprocessing. Currently, the ALL package, developed by the EU E-CAM Centre of Excellence (https://www.e-cam2020.eu/), is being tested with HemeLB.

The original load-balancing approach successively assigns workload to cores until a nominally even distribution is obtained. Concave geometric domains may force this algorithm to assign non-contiguous regions to the same core, leading to unfavourable communication overhead. Initial studies with ALL have used the package's orthogonal recursive bisection scheme combined with a histogram method. This combination guarantees good load balancing for single connected and convex domains. When compared to the native HemeLB approach, we find both approaches give a balanced workload distribution of $\text{workload}_{\text{ave}}/\text{workload}_{\text{max}} \approx 80\%$ with only a slight dependence on the number of cores used. Inspection shows that the work distribution, based on the prescribed weight of the blocks, is close to being ideally balanced (greater than 98%), which can be observed for the majority of cores. However, the overall performance is limited by a few outliers, which consume about 10–20% more time. Initial analysis shows that this is not due to communication overhead and also varies between runs with the same process allocation. A possible reason for this behaviour may be found in memory access patterns where memory cannot be allocated contiguously and read/write operations get imbalanced as a result of concurrent memory allocation of processes on the nodes. A more detailed analysis of these results is currently being conducted and this will, ultimately, be used to improve the load balancing of HemeLB on exascale machines. Preliminary results from these studies are presented in §2 of the electronic supplementary material.

2.3. Self-coupling of HemeLB

In order to create a realistic simulation of a virtual human, HemeLB must be able to communicate with both other HemeLB instances and other biophysical modelling codes. In the first instance, coupling HemeLB to itself will establish a framework for how this can be conducted. While motivated by simulating coupled arterial and venous vasculatures, it also provides a setting to enable simulation of regions at different resolutions. The self-coupled version of HemeLB constructs a common computational universe in which multiple HemeLB instances are instantiated. Each instance retains its own internal communications for executing the simulation on the cores assigned to it. The master ranks of each instance exchange boundary condition states at coupled interfaces to generate interaction between HemeLB worlds.

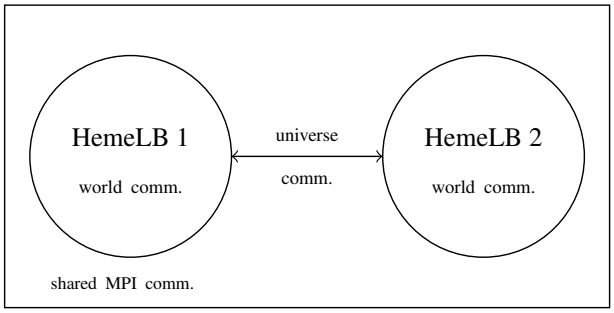

Figure 4. Schematic layout of the communications required to couple HemeLB with itself. All communications are conducted using standard MPI calls. This strategy can be extended to permit coupling of further HemeLB worlds.

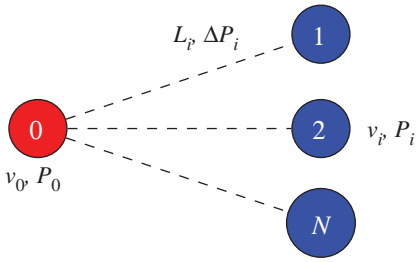

Figure 5. A one-to-many coupling is generated to link an arterial outlet (red) with venous (blue) inlets. Each boundary site has an associated velocity, v and pressure, P. The distance between each vein and artery, L_i, is measured and the associated pressure drop, ΔP_i, is defined.

This strategy is laid out schematically in figure 4. To minimize the data needing to be passed between coupled HemeLB instances, only the average value of macroscopic properties is passed between coupled boundaries. The coupling of the HemeLB instances is carried out by constructing a mapping between the outlets of the first case and the inlets of the second case. In the case of arterial and venous trees there are typically many more venous inlets than arterial outlets, resulting in a number of one-to-many interfaces. Between each of the inlet–outlet pairings, a scaling factor of velocity and pressure is assigned to represent the change in flow behaviour in the intervening vasculature between the two geometries and used to construct local boundary conditions.

3. Arterial–venous coupling

The human circulatory system consists of vessels ranging from the centimetre scale such as the aorta down to micrometre sized capillaries. These are classified (e.g. pulmonary/systemic, arteries/veins) based on their transport direction and location. Within a full human model, the resolution of the smallest vessels and capillaries is often limited. To complete the connection of coupled arterial–venous flows, these vessels are approximated as a sub-scale feature that can be represented with a pressure and velocity drop between vessels on either side. Using this approach, the self-coupling of HemeLB will enable simultaneous blood flow simulations of arterial and venous networks. If found to be necessary, future work may involve developing and implementing more sophisticated coupling strategies to represent the sub-scale material,

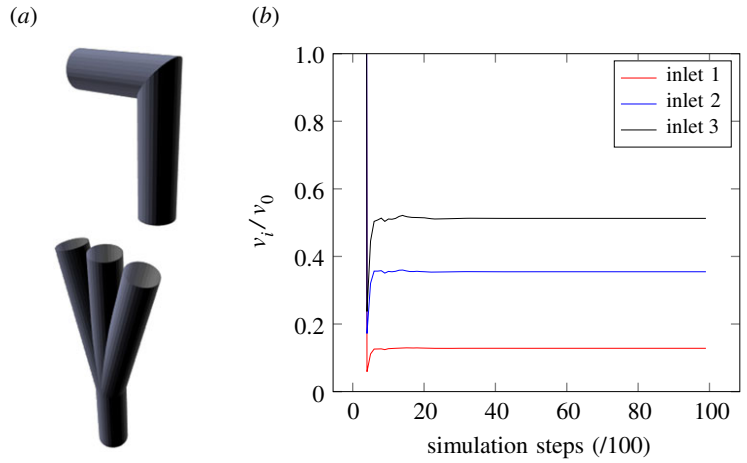

Figure 6. Schematic layout and velocity ratio results of the 1 : 3 test domain for demonstrating performance of the self-coupling of HemeLB; arteries are the upper section of the layout. (*a*) Test layout; (*b*) velocity ratios.

such as through incorporation of porous media models of capillary beds.

To simulate coupled vascular networks, a strategy for determining the pressure and velocity scaling factors between coupled boundary locations was developed to ensure that mass conservation is maintained. For a given vessel size there are, generally, many more veins than arteries. In the vascular networks simulated in this study (described later in this section) there are 13 times more vein inlets than arterial outlets. The first stage in defining a coupling map between arteries and veins involves identifying which inlets and outlets of the independent geometries are linked. A naive initial approach is based on the relative proximity of boundary locations. A mapping algorithm was devised to ensure that each arterial outlet is coupled to at least one vein. For each artery, the list of venous inlets is inspected and the closest is assigned to that vessel, then removed from the list. Once complete, the remaining veins are assigned to their closest artery. This generates a $1:N$ coupling map for each arterial outlet, as shown in figure 5. The distance between each coupled pair is stored for later use.

To represent the sub-scale material between the macroscopic arteries and veins, a pressure drop between each inlet–outlet pair is assigned to a random value between 0.3 and 0.7. This range represents physiologically expected values [35] and permits variability of structure in the hypothetical capillary network it represents. This correlates directly to the pressure scale factor, $\Delta P_i/P_0$, assigned to that pairing. The velocity scale factor, v_i/v_0, is determined by assuming that for each inlet–outlet pair the pressure drop, ΔP_i, is proportional to the product of flow rate, q_i, and length, L_i,

$$\Delta P_i = P_0 - P_i = k_i q_i L_i. \tag{3.1}$$

By comparison of units, it can be noted that k_i is inversely proportional to the area of the inlet, A_i. The scaling factor for velocity is then determined by combining equation (3.1) for each inlet i with mass conservation of the system (i.e. $q_0 = \sum_{i=1}^{N} q_i$) to determine the scaling of velocity, v_i, for each inlet–outlet pair:

$$\frac{v_i}{v_0} = \frac{A_0}{A_i}\frac{\Delta P_i L_1 A_i}{\Delta P_1 L_i A_1}\left(\sum_{j=1}^{N}\frac{\Delta P_j L_1 A_j}{\Delta P_1 L_j A_1}\right)^{-1}. \tag{3.2}$$

Note that a subscript of zero indicates values relating to the arterial outlet.

These factors are used to construct the velocity boundary conditions on the opposite geometry. In the 'forward' (i.e. arterial-to-venous) direction, the scale factors are applied to the average velocity at the arterial outlet to construct boundary condition values to be applied at the coupled venous inlets. In the 'reverse' direction (i.e. venous-to-arterial), the boundary force values at the arterial outlets are determined by taking the average of the average velocities at the inlets scaled by the appropriate scaling factor. To represent the volume of fluid existing in the capillaries between any given coupled inlet and outlet, an explicit force is applied to the outlet side based on the dynamic pressure at that location using the approach of Guo *et al.* [36]. The coupling strategy alternates between the 'forward' and 'reverse' directions each time boundary information is swapped.

As a demonstration of the performance of the coupling strategy, we show the recorded velocity ratios at coupled inlets and outlets in a simplified 1 : 3 geometry. This is illustrated in figure 6*a*. In this setting the imposed pressure scaling factors were $\Delta P_i/P_0 = 0.30$, 0.52, 0.69 for the $i = 1$, 2, 3 inlets; the corresponding velocity factors were $v_i/v_0 = 0.13$, 0.36, 0.52 with all boundaries of equal size. Figure 6*b* demonstrates that these ratios are achieved between the appropriate inlets when steady state is reached (again, after approximately 3000 simulation steps). These results verify the boundary condition implementation by demonstrating that the desired results are achieved.

3.1. Obtaining human-scale input data

We used the recently created computational anatomical model Yoon-sun [37], which was segmented from high-resolution ($0.1 \times 0.1 \times 0.2$ mm) colour photographs of cross-sections from a frozen female cadaver obtained in the Visible Korean project [38]. Yoon-sun, available as part of the Virtual Population (ViP) library [39,40], was segmented at $0.2 \times 0.2 \times 1.0$ mm resolution, separating more than 1100 tissues, and provides unprecedented details in the peripheral nervous system, muscles and arterial–venous system (figure 7). Yoon-sun was created following standardized quality assurance guidelines developed to ensure consistent segmentations across the ViP library. The finest segmented vessels have a diameter in the range of 1–2 pixels (0.2–0.4 mm).

The surface model of the arteries and veins was clipped at the vessel tips to improve the ability of HemeLB to apply

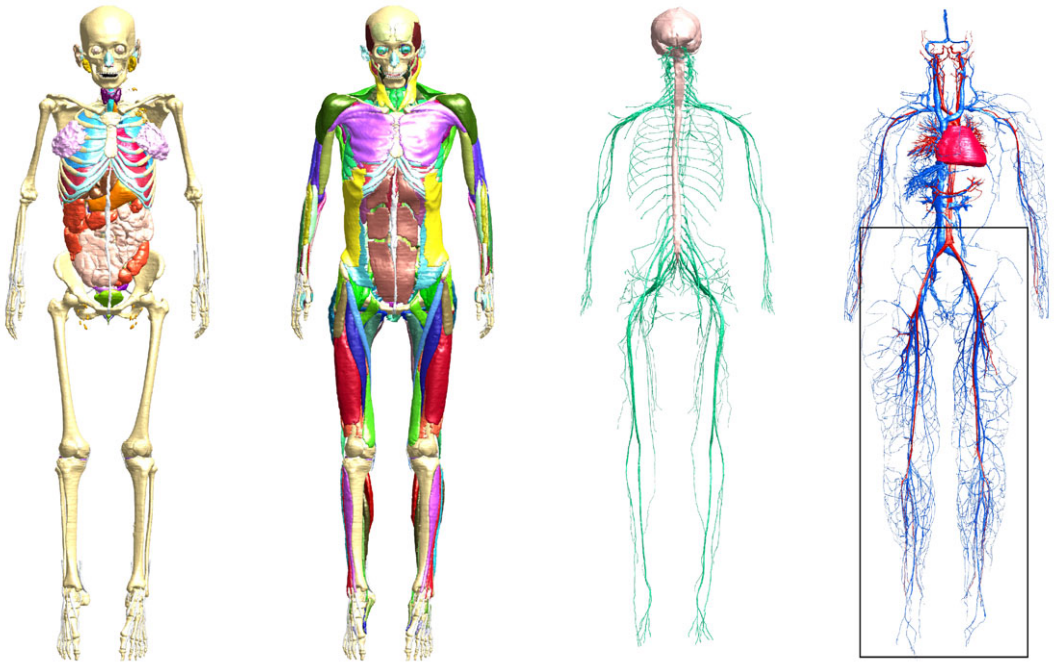

Figure 7. The female ViP model Yoon-sun with unprecedented detail of the (from left to right) skeleton and internal organs, muscles, peripheral nerves and vasculature [38–40]. Colours represent different structures in each case. The section of the leg vessels highlighted by the box (and shown in figure 8) was used for initial tests of human-scale vasculature simulations.

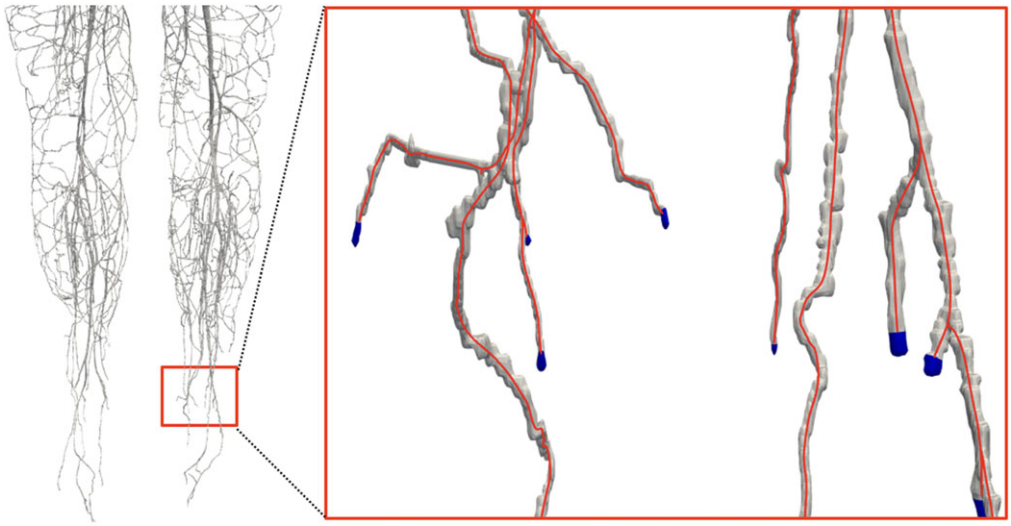

Figure 8. Illustration of vessel surface pre-processing for HemeLB, which detects locations for boundary conditions at open edges of the vessel surface. Centrelines (red lines) are generated from the triangulated surfaces of arterial and venous trees. The vessel tips (depicted in blue) are clipped by a cutting algorithm.

boundary conditions to the geometry. To handle the large number of vessel tips (more than 1500), this was automated through the following algorithm: (i) surfaces of the vasculature were extracted using the Marching Cubes algorithm [41]; (ii) centrelines were extracted from this triangulated surface using a skeletonization technique [42,43]; and, finally, (iii) the vasculature model was clipped by cutting the surface mesh at the end points of the centrelines (perpendicular to the line orientation). This algorithm is implemented using in-house C++ building-blocks, except for the skeletonization which uses the CGAL library [43]. To limit the cut to the corresponding tip surface (i.e. to avoid cutting with an infinite plane), a traversal algorithm was implemented, which starts the cut at the vertex closest to the tip location. Figure 8 shows an example from the arterial tree.

3.2. Visualization of very large datasets

Additionally, we are developing a visualization method for the analysis of data and communication of results produced by HemeLB. Previous work with HemeLB datasets—developed in tandem with the 'Virtual Humans' IMAX movie [44]—focused on simulating movement of particles based on a much smaller dataset (about 160 million data sites/time step) than is required for our virtual human-scale simulation. This earlier work, using sparsely populated octrees, was used for real-time visualization and data processing for cinematic renderings [44]. However, this approach is no longer viable due to the size and complexity of datasets planned to be generated with the self-coupled HemeLB code. Instead, we aim to provide precomputed flow data to visualize flow in complex, large-scale datasets. While this

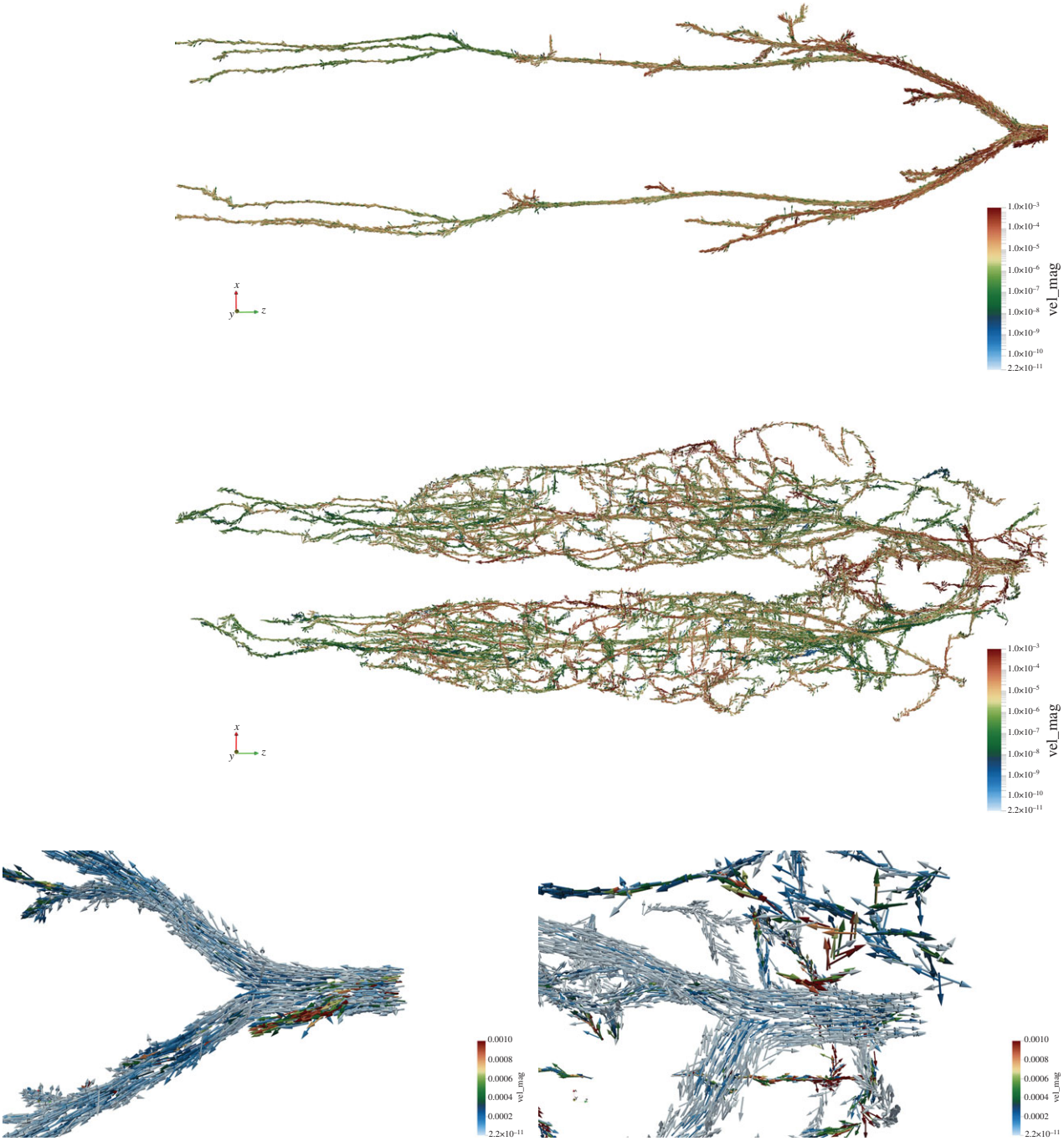

Figure 9. Field of velocity vectors (coloured by magnitude on a logarithmic scale) through the arterial (upper row) and venous (centre row) domains. The lower row (arterial left and venous right) is a close-up view of the iliac bifurcation of the aorta at the top of the simulated geometry (right-hand side of full domain images). All figures have been capped at the arterial venous velocity to illustrate the progression of flow through the domain at this point of the simulation. The full domain of the venous tree contains approximately 700 000 vectors. These figures demonstrate the scale of 3D vascular models which we can currently simulate with HemeLB.

limits versatility of the visualization, it allows for real-time exploration of the flow data that can be directed and altered to investigate different scenarios.

3.3. Human-scale blood flow results

We simulate blood flow from just above the iliac bifurcation of the Yoon-sun geometry, as indicated in figure 7. In this configuration, the arterial tree has a single inlet and 38 outlets, while the venous tree has 494 inlets and one outlet. For this case study, the domain was resolved with a lattice spacing of 75 µm (total of 434 579 134 lattice sites across the arterial and venous geometries). For more detailed simulations, a resolution of perhaps 25 µm would be necessary to fully resolve the finest vessels. Models of this scale would exceed 10^9 lattice sites. Velocity conditions were applied at the inlets to domains and pressure boundary conditions applied to the outlets. The maximum velocity at the inlet to the arterial tree was set at a constant 0.001 m s^{-1}, smaller than physically expected at that location but necessary to simulate stable flow for this resolution model. This simulation is intended to provide a proof-of-concept of the self-coupling strategy outlined in this paper on human-scale vascular geometries rather than a rigorous quantitative evaluation of its performance. The flow conditions chosen are adequate for the current demonstration purpose. Full validation of physiologically accurate flow will be the subject of future work. The parameters applied to the coupled boundary locations were determined by the self-

coupling strategy described in this paper. The simulations used to generate this data were run for 1 000 000 steps, corresponding to 100 s of physical time. This took approximately 13 h on 10 080 cores of SuperMUC-NG. The resulting velocity distribution (figure 9) illustrates how the flow development is occurring throughout the full geometry and around the iliac bifurcation. The flow field is, qualitatively, as expected throughout the domain. One issue with conducting simulations on this scale and geometry is the time taken for flow to develop throughout the domain. This highlights the need to carefully consider the initialization state of large domains so as to rapidly develop physiologically realistic states that minimize simulation time.

To provide further qualitative validation of the simulation, we examine the flow velocity experienced in vessels compared to their size. Within the systemic circulatory system, the mean velocity is maximal upon leaving the heart and reduces as it travels through to smaller vessels. Within the venous system, the velocity increases again on returning to the heart but at a more gradual rate. Schematic illustrations of this are provided in chapter 20.2 of [35]. Figure 10 illustrates the distribution of velocity magnitude as a function of vessel area. Note that the plot is organized from left-to-right based on general progression through the circulatory system; arteries are plotted with a 'negative' area. These data points were obtained by inserting a measurement plane at approximately one-quarter, one-half and three-quarters of the way along the length of the geometry. These were separated into cross-sections of the various vessels and mean velocity calculated. The area was estimated by taking a convex hull around the lattice points associated with each vessel. The anticipated velocity reduction in the arterial vessels is clearly observed. The results for the venous tree also generally demonstrate the correct trend—an increase in velocity with vessel area but at a much slower rate of change than that seen in the arteries. A small number of higher-velocity data points are still present in the venous tree, particularly for smaller vessels. These may arise due to an inappropriate coupling being generated by the naive and automated initial strategy. This indicates that the coupling of complex vascular networks may require a more subtle strategy than the solely distance-based approach outlined here as an initial proof-of-concept. It is also possible that these high-velocity data points are driving instabilities that resulted in a low inlet velocity being required in this simulation. The assumption of rigid walls may also impact venous behaviour more strongly than the arterial tree. Addressing these will be a focus for future work.

The accuracy and stability of single relaxation time lattice Boltzmann simulations are governed by two equations. The first, $\nu = (1/3)(\tau - 1/2)(\Delta x^2/\Delta t)$, links fluid viscosity, ν, with the grid spacing, Δx, time step, Δt, and relaxation time, τ. The second, $Ma = (\sqrt{3} v_{\max} \Delta t)/\Delta x < 0.1$, stipulates the valid computational Mach number for a simulation. Typically it is desired that $\tau \in (0.5, 1]$ and preferably closer to one; and the Mach number to be as small as possible. These are competing objectives. Rough calculations suggest that for physiologically realistic flows (e.g. 1 m s^{-1} in the aorta) a resolution of 25 µm and a time step of 1 µs represent the cusp of achievable simulation parameters. Randles *et al.* [45] have generated an arterial geometry with over 10^{11} sites at 10 µm, indicating that this is achievable. These values represent a reason for using large supercomputers to conduct these simulations. Matching of Reynolds numbers with

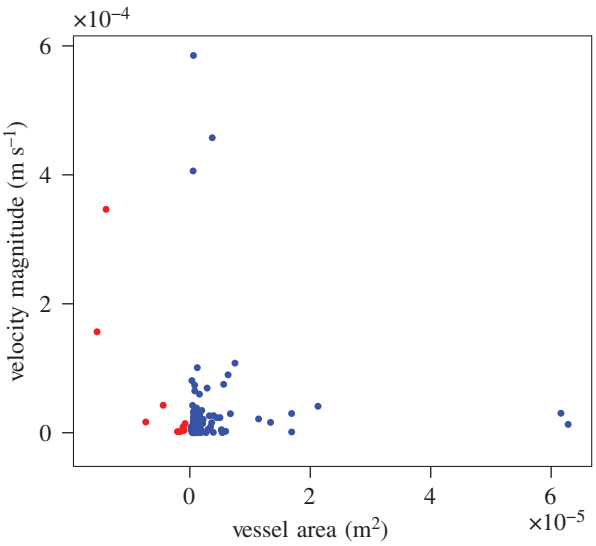

Figure 10. Plot of flow velocity magnitude against vessel area after 1 000 000 simulation steps for the coupled arterial (red) and venous (blue) networks of the region indicated in figure 7. Planes of vessels were extracted at three locations along the length of the domain. The 'negative' region for the arterial vessels is an artefact designed to keep the arterial and venous points separated on the graph.

increased fluid viscosity could assist in further relaxing these requirements.

Once communication between the coupled instances of HemeLB has been optimized, the full performance profile of the coupling strategy can be assessed. This will require a sufficiently large pair of geometries to study this adequately, perhaps requiring at least 10^9 fluid sites each. As demonstrated in §2.2, HemeLB has excellent strong scaling performance as a single instance, although the coupling of multiple instances reduces this behaviour due to additional communication overhead. We plan to assess this behaviour of self-coupled HemeLB in a future publication.

4. Conclusion

Significant scientific and medical research effort is being directed towards enabling the simulation of a virtual human. The ultimate delivery of this goal will allow for the prescription of medical treatments personalized for every patient. The virtual human would also allow healthy individuals to discover how they can improve their current lifestyle choices. The performance of supercomputers is now reaching a level at which the creation of a virtual human is feasible. This paper brings together cutting-edge technologies and algorithms for building and simulating virtual humans on the latest high-performance computing platforms. Firstly, we described software implementations that permit strong scaling performance for complex domains with $\mathcal{O}(10^{10})$ lattice sites on $\mathcal{O}(10^5)$ cores. In particular, this has involved modifying MPI communication behaviour to allow for larger data communication. Achieving performance on this scale also requires streamlining of internal HemeLB data structures and improved load-balancing techniques. We have outlined a self-coupling strategy to allow coupled simulations of arterial and venous flows (a critical feature for modelling blood flow within a virtual human). Here we demonstrate that the coupling strategy

is able to communicate information between several hundred linked boundary locations and is able to reconstruct flow at this scale.

In this paper, we have demonstrated that the self-coupling of HemeLB is able to simulate blood flow in human-scale geometries that capture the expected dynamic flow features. In future developments of this work, we plan to progress towards creating a virtual human by extending the current simulation domain to the full human and continuing to expand the physics modelled by HemeLB. In particular, we will look at methods for improving the coupling behaviour between domains and relaxing HemeLB's structural and fluid assumptions. In future work, we plan to validate the coupling model presented here by close comparison with physiological data. This will allow us to apply the model to large-scale physiological and clinical situations. Visualization remains an essential element in comprehension and communication of the large-scale data generated by simulations of human-scale domains. We seek to further develop techniques for the visualization of human-scale blood flow in order to better enable understanding of the virtual human. The advent of exascale computers will greatly facilitate these endeavours. As many future exascale machines are planned to incorporate the use of accelerators, we are also developing a GPU-enabled version of HemeLB to further capitalize on the performance capabilities of these new machines.

Data accessibility. HemeLB code for single simulations can be found at https://github.com/UCL-CCS/HemePure and coupled simulations at https://github.com/UCL-CCS/HemePure_SelfCoupled. The circle of Willis geometry used for scaling tests was provided by Figueroa [46]. The full human geometry datasets are available through the IT'IS [37].

Authors' contributions. J.W.S.M. and P.V.C. provided the scientific direction and conceptualization of the manuscript. All authors contributed to the overall methodology of the work presented here. Particular contributions to software and development are as follows: (i) HemeLB coupling and scientific applications: J.W.S.M., R.A.R., A.P.; (ii) MPI: M.R., R.M., A.S.; (iii) code scalability and performance: B.J.N.W., A.P.; (iv) load balancing: A.P., R.H., G.S.; (v) virtual human vasculature: B.L., E.N.; and (vi) visualisation: T.O., M.W. Authors J.W.S.M. and P.V.C. prepared the initial draft and all authors contributed to their specific sections and to the critical revision of the overall manuscript. J.W.S.M. managed the data presented in this paper. P.V.C. and D.K. supervised the work conducted here, including directing the main efforts for acquisition of funding and supercomputing allocations. P.V.C. provided overall project direction and management. All authors gave final approval for publication and agree to be held accountable for the work performed therein.

Competing interests. We declare we have no competing interests.

Funding. We acknowledge funding support from European Commission Centres of Excellence CompBioMed (grant nos 675451 and 823712), E-CAM (grant no. 676531) and POP (grant nos 676553 and 824080). Support from the Research Councils UK Engineering and Physical Sciences Research Council under the project 'UK Consortium on Mesoscale Engineering Sciences (UKCOMES)' (grant no. EP/R029598/1) is gratefully acknowledged. We acknowledge funding support from Research Councils UK Medical Research Council (MRC) for a Medical Bioinformatics grant (no. MR/L016311/1), and special funding from the UCL Provost. This work was also performed with partial support from the National Science Foundation under grant nos 1562306, 1713749, 1822191, 1821431 and 1918987. We acknowledge Innosuisse (CTI 25290.1 PFLS-LS) for funding the development of the anatomical model used in this work through the NEUROMAN project.

Acknowledgements. The authors gratefully acknowledge the Gauss Centre for Supercomputing e.V. (www.gauss-centre.eu) for funding this project by providing computing time on the GCS Supercomputer SuperMUC-NG at Leibniz Supercomputing Centre (www.lrz.de). This work used the ARCHER UK National Supercomputing Service (www.archer.ac.uk).

References

1. Kohl P, Noble D. 2009 Systems biology and the virtual physiological human. *Mol. Syst. Biol.* **5**, 292. (doi:10.1038/msb.2009.51)
2. Hunter P *et al.* 2010 A vision and strategy for the virtual physiological human in 2010 and beyond. *Phil. Trans. R. Soc. A* **368**, 2595–2614. (doi:10.1098/rsta.2010.0048)
3. Hunter P *et al.* 2013 A vision and strategy for the virtual physiological human: 2012 update. *Interface Focus* **3**, 20130004. (doi:10.1098/rsfs.2013.0004)
4. Chase JG, Desaive T, Preiser J-C. 2016 Virtual patients and virtual cohorts: a new way to think about the design and implementation of personalized ICU treatments. In *Annual Update in Intensive Care and Emergency Medicine* (ed. JL Vincent), pp. 435–448. Cham, Switzerland: Springer. (doi:10.1007/978-3-319-27349-5_35)
5. Sheng C, Sarwal SN, Watts KC, Marble AE. 1995 Computational simulation of blood flow in human systemic circulation incorporating an external force field. *Med. Biol. Eng. Comput.* **33**, 8–17. (doi:10.1007/BF02522938)
6. Olufsen MS. 1999 Structured tree outflow condition for blood flow in larger systemic arteries. *Am. J. Physiol. Heart Circ. Physiol.* **276**, H257–H268. (doi:10.1152/ajpheart.1999.276.1.H257)
7. Qureshi MU, Vaughan GDA, Sainsbury C, Johnson M, Peskin CS, Olufsen MS, Hill NA. 2014 Numerical simulation of blood flow and pressure drop in the pulmonary arterial and venous circulation. *Biomech. Modeling Mechanobiol.* **13**, 1137–1154. (doi:10.1007/s10237-014-0563-y)
8. Audebert C, Bucur P, Bekheit M, Vibert E, Vignon-Clementel IE, Gerbeau J. 2017 Kinetic scheme for arterial and venous blood flow, and application to partial hepatectomy modeling. *Comput. Methods Appl. Mech. Eng.* **314**, 102–125. Special Issue on Biological Systems Dedicated to William S. Klug. (doi:10.1016/j.cma.2016.07.009)
9. Groen D, Hetherington J, Carver HB, Nash RW, Bernabeu MO, Coveney PV. 2013 Analysing and modelling the performance of the HemeLB lattice-Boltzmann simulation environment. *J. Comput. Sci.* **4**, 412–422. (doi:10.1016/j.jocs.2013.03.002)
10. HemeLB, 2019. www.hemelb.org.
11. Palabos, 2019. See https://palabos.unige.ch/.
12. TCLB Reference Manual, 2019. See https://tclb-docs.netlify.com/.
13. Krause MJ. 2019 OpenLB – Open Source Lattice Boltzmann Code. www.openlb.net/.
14. Rüde U. 2019 waLBerla. www.walberla.net/index.html.
15. Mazzeo MD, Coveney PV. 2008 HemeLB: a high performance parallel lattice-Boltzmann code for large scale fluid flow in complex geometries. *Comput. Phys. Commun.* **178**, 894–914. (doi:10.1016/j.cpc.2008.02.013)
16. Groen D, Richardson RA, Coy R, Schiller UD, Chandrashekar H, Robertson F, Coveney PV. 2018 Validation of patient-specific cerebral blood flow simulation using transcranial Doppler measurements. *Front. Physiol.* **9**, 721. (doi:10.3389/fphys.2018.00721)
17. Bernabeu MO *et al.* 2014 Computer simulations reveal complex distribution of haemodynamic forces in a mouse retina model of angiogenesis. *J. R. Soc. Interface* **11**, 20140543. (doi:10.1098/rsif.2014.0543)
18. Patronis A, Richardson RA, Schmieschek S, Wylie BJN, Nash RW, Coveney PV. 2018 Modeling patient-specific magnetic drug targeting within the

intracranial vasculature. *Front. Physiol.* **9**, 331. (doi:10.3389/fphys.2018.00331)

19. Nash RW, Carver HB, Bernabeu MO, Hetherington J, Groen D, Krüger T, Coveney PV. 2014 Choice of boundary condition for lattice-Boltzmann simulation of moderate-Reynolds-number flow in complex domains. *Phys. Rev. E* **89**, 023303. (doi:10.1103/PhysRevE.89.023303)
20. Bernabeu MO, Nash RW, Groen D, Carver HB, Hetherington J, Krüger T, Coveney PV. 2013 Impact of blood rheology on wall shear stress in a model of the middle cerebral artery. *Interface Focus* **3**, 20120094. (doi:10.1098/rsfs.2012.0094)
21. Vázquez M, Arís R, Aguado-Sierra J, Houzeaux G, Santiago A, López M, Córdoba P, Rivero M, Cajas JC. 2015 Alya Red CCM: HPC-based cardiac computational modelling. In *Selected topics of computational and experimental fluid mechanics* (eds J Klapp, G Ruíz Chavarría, A Medina Ovando, A López Villa, L Di G Sigalotti), pp. 189–207. Cham, Switzerland: Springer. (doi:10.1007/978-3-319-11487-3_11)
22. Message Passing Interface Forum. 2012 *MPI: a message passing interface standard*. www.mpi-forum.org/docs/mpi-3.1/mpi31-report.pdf.
23. Message Passing Interface Forum. 2020 *The MPI 4.0 (DRAFT) Standard*. www.mpi-forum.org/mpi-40/.
24. Latham R, Gropp W, Ross R, Thakur R. 2007 Extending the MPI-2 generalized request interface. In *Recent advances in parallel virtual machine and message passing interface. EuroPVM/MPI 2007* (eds F Cappello, T Herault, J Dongarra). Lecture Notes in Computer Science, vol. 4757, pp. 223–232. Berlin, Germany: Springer. (doi:10.1007/978-3-540-75416-9_33)
25. Shi R, Lu X, Potluri S, Hamidouche K, Zhang J, Panda DK. 2014 HAND: a hybrid approach to accelerate non-contiguous data movement using MPI datatypes on GPU clusters. In *2014 43rd Int. Conf. on Parallel Processing, Minneapolis, MN, 9–12 September*, pp. 221–230. IEEE. (doi:10.1109/ICPP.2014.31)
26. Ruefenacht M. 2019 BigCount—generic/overloading interfaces. https://github.com/mpi-forum/mpi-issues/issues/137.
27. Hammond J. 2018 Big MPI—large-count and displacement support–collective chapter. https://github.com/mpi-forum/mpi-issues/issues/80.
28. Hammond JR, Schäfer A, Latham R. 2014 To INT_MAX … and beyond! Exploring large-count support in MPI. In *2014 Workshop on Exascale MPI at Supercomputing Conference, New Orleans, LA, 17 November*, pp. 1–8. IEEE. (doi:10.1109/ExaMPI.2014.5)
29. Geimer M, Wolf F, Wylie BJN, Ábrahám E, Becker D, Mohr B. 2010 The Scalasca performance toolset architecture. *Concurrency Computat. Pract. Exper.* **22**, 702–719. (doi:10.1002/cpe.1556)
30. Leibniz Supercomputing Centre. 2019 SuperMUC-NG. See https://doku.lrz.de/display/PUBLIC/SuperMUC-NG.
31. Hoekstra AG, Chopard B, Coster D, Portegies Zwart S, Coveney PV. 2019 Multiscale computing for science and engineering in the era of exascale performance. *Phil. Trans. R. Soc. A* **377**, 20180144. (doi:10.1098/rsta.2018.0144)
32. Boman E, Devine K, Fisk L, Heaphy R, Hendrickson B, Leung V, Vaughan C, Catalyurek U, Bozdag D, Mitchell W. 1999 Zoltan home page. www.cs.sandia.gov/Zoltan.
33. Boman EG, Catalyurek UV, Chevalier C, Devine KD. 2012 The Zoltan and Isorropia parallel toolkits for combinatorial scientific computing: partitioning, ordering, and coloring. *Sci. Program.* **20**, 129–150. (doi:10.3233/SPR-2012-0342)
34. Karypis G, Kumar V. 2009 MeTis: Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 4.0. www.cs.umn.edu/metis.
35. OpenStax. 2016 *Anatomy & Physiology*. OpenStax CNX. See https://opentextbc.ca/anatomyandphysiology/.
36. Guo Z, Zheng C, Shi B. 2002 Discrete lattice effects on the forcing term in the lattice Boltzmann method. *Phys. Rev. E* **65**, 046308. (doi:10.1103/PhysRevE.65.046308)
37. IT'IS. 2019 Human Models—Yoon-sun. See https://itis.swiss/virtual-population/virtual-population/vip3/yoon-sun/.
38. Park JS, Chung MS, Hwang SB, Lee YS, Har D, Park HS. 2005 Visible Korean Human: improved serially sectioned images of the entire body. *IEEE Trans. Med. Imaging* **24**, 352–360. (doi:10.1109/TMI.2004.842454)
39. Christ A *et al.* 2010 The Virtual Family—development of surface-based anatomical models of two adults and two children for dosimetric simulations. *Phys. Med. Biol.* **55**, N23–N38. (doi:10.1088/0031-9155/55/2/N01)
40. Gosselin M *et al.* 2014 Development of a new generation of high-resolution anatomical models for medical device evaluation: the Virtual Population 3.0. *Phys. Med. Biol.* **59**, 5287–5303. (doi:10.1088/0031-9155/59/18/5287)
41. Lorensen WE, Cline HE. 1987 Marching Cubes: a high resolution 3D surface construction algorithm. In *ACM SIGGRAPH Computer Graphics*, vol. 21, pp. 163–169. New York, NY: ACM. (doi:10.1145/37401.37422)
42. Tagliasacchi A, Alhashim I, Olson M, Zhang H. 2012 Mean curvature skeletons. *Comput. Graphics Forum.* **31**, 1735–1744. (doi:10.1111/j.1467-8659.2012.03178.x)
43. Fabri A, Teillaud M. 2011 CGAL—the computational geometry algorithms library. In *10e colloque national en calcul des structures, Giens, France, May 2011*, p. 6. See https://hal.archives-ouvertes.fr/hal-00592685.
44. CompBioMed. 2018 CompBioMed Virtual Humans Film. www.youtube.com/watch?v=1FvRSJ9W734.
45. Randles A, Draeger EW, Bailey PE. 2015 Massively parallel simulations of hemodynamics in the primary large arteries of the human vasculature. *J. Comput. Sci.* **9**, 70–75. (doi:10.1016/j.jocs.2015.04.003). Computational Science at the Gates of Nature.
46. Figueroa CA. 2020 .stl file of Circle of Willis Benchmark geometric model for hemodynamic simulation software [Dataset]. University of Michigan Library, Deep Blue Data. (doi:10.7302/xx1r-zg70)