

Computational Physics

Scalable, cloud-based simulations of blood flow and targeted drug delivery in retinal capillaries



Lucas Amoudruz ^{a,1}, Sergey Litvinov ^{a,1}, Riccardo Murri ^b, Volker Eyrich ^c,
Jens Zudrop ^b, Costas Bekas ^b, Petros Koumoutsakos ^{a,*}

^a Computational Science and Engineering Laboratory, School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, 02138, United States

^b Citadel Securities, 200 S. Biscayne Boulevard, Miami, FL, 33131, United States

^c Google Mountain View, CA, United States

ARTICLE INFO

The review of this paper was arranged by Prof. Andrew Hazel

Keywords:

Cloud computing
Mesoscale simulations
Biological flows

ABSTRACT

We investigate the capabilities of cloud computing for large-scale, tightly-coupled simulations of biological fluids in complex geometries, traditionally performed in supercomputing centers. We demonstrate scalable and efficient simulations in the public cloud. We perform meso-scale simulations of blood flow in image-reconstructed capillaries, and examine targeted drug delivery by artificial bacterial flagella (ABFs). The simulations deploy dissipative particle dynamics (DPD) with two software frameworks, Mirheo(developed by our team) and LAMMPS. Mirheoexhibits remarkable weak scalability for up to 512 GPUs. Similarly, LAMMPS demonstrated excellent weak scalability for pure solvent as well as for blood suspensions and ABFs in reconstructed retinal capillaries. In particular, LAMMPS maintained weak scaling above 90% on the cloud for up to 2000 cores. Our findings demonstrate that cloud computing can support tightly coupled, large-scale scientific simulations with competitive performance.

1. Introduction

Computing has become the third pillar of scientific inquiry and engineering design, along with theory and experimentation. We distinguish two major computing modalities: large-scale simulations, where all computational nodes are dedicated to solving one particular problem, and high-throughput, which involves many smaller simulations run independently across different compute nodes. In the former case, communication within and between nodes is essential, whereas in the latter case, communication between nodes is reduced. High-throughput computing has become indispensable for optimization and uncertainty quantification in areas ranging from manufacturing to automotive design.

To date, the vast majority of computing, in particular for large-scale simulations, has relied on dedicated computational infrastructure such as local clusters, university computing centers, and national supercomputing facilities. These types of infrastructure offer a unique combination of benefits in terms of accessibility, speed, cost, and service. However, they often require long-term investments, which can impact short-term cost-efficiency, while access to them can be restricted or may not be readily available on demand.

In recent years, cloud computing has emerged as a disruptive trend by offering a flexible, scalable and potentially cost-efficient alternative to traditional high performance computing (HPC). Researchers initially evaluated cloud computing for relatively simple tasks, such as single-node computations [1], or small-scale parallel runs [2]. In early benchmarks, Iosup et al. [3] and Zhai et al. [4] found that commercial cloud services (Amazon EC2, GoGrid, ElasticHosts, and Mosso) performed poorly compared to dedicated HPC systems, largely due to limited interconnect speed and high latency. These constraints were considered critical for tightly coupled simulations, where frequent communication between nodes is required.

However, this is not a major bottleneck for high-throughput workloads [5–7], and cloud computing has proven highly effective in this scenario [8]. Furthermore, recent effort has shown progress in enabling tightly coupled simulations on the cloud. This includes microbenchmarks [9], tightly coupled simulations on four nodes [10] and computational fluid dynamics at small to moderate scale [11–13].

Despite this progress, large-scale simulations of complex biological systems, such as blood flow at the microscale, remain dependent on traditional HPC infrastructure. These computing platforms offer

* Corresponding author.

E-mail address: petros@seas.harvard.edu (P. Koumoutsakos).

¹ Equal contributions.

high-bandwidth and low-latency interconnects but come with limitations: restricted and competitive access, limited scalability beyond allocated quotas, high operational overheads, and a lack of on-demand elasticity. In contrast, modern cloud infrastructure offers elastic compute resources, user-based pricing, and global availability, potentially transforming current approaches in scientific computing.

In fact, other fields have already embraced cloud computing for tightly-coupled tasks. For instance, weather forecasting models have been successfully migrated to the cloud, offering faster deployment cycles and reduced infrastructure costs [14,15]. Similarly, industries such as automotive and aerospace have adopted cloud platforms to run commercial computational fluid dynamics codes [16–18], citing improved flexibility and cost-efficiency as key drivers.

A previous study has demonstrated that the lattice-Boltzmann solver Palabos can scale efficiently for blood-flow simulations on public clouds [19,20]. Yet, to our knowledge, no study has evaluated whether large-scale tightly coupled microscale blood simulations, using particle-based methods like dissipative particle dynamics (DPD), can be efficiently executed on public cloud infrastructure.

This work addresses this gap by investigating whether recent advances in cloud computing, such as enhanced networking, faster storage, and cost-optimized compute tiers, are sufficient to support demanding simulations of blood at sub-micron resolution. These simulations are critical for biomedical applications, including targeted drug delivery, precision microsurgery, and microsensing. The complexity of these simulations arises from the need to accurately predict the deformations of red blood cells (RBCs) and their interaction with blood plasma.

We consider two software packages for this purpose. Both programs represent blood as a set of RBC membranes suspended in plasma, an incompressible viscous Newtonian fluid. The membranes are discretized into a triangle mesh, while the plasma is represented by the DPD method [21]. The first program that we consider is LAMMPS [22], a widely used molecular dynamics (MD) package parallelized across nodes with MPI. Here, LAMMPS was extended to support the addition of RBCs membrane model described in ref. [23], as well as artificial bacterial flagella (ABFs) [24,25].

LAMMPS is a widely used software package for particle simulations, known for its user-friendly design and relatively good performance [22]. It features an advanced spatial decomposition algorithm tailored for simulations on distributed memory hardware. Its proven scalability on leading supercomputers further establishes its reputation for reliability and high performance in particle simulations. LAMMPS has been designed to optimize its critical performance kernels for diverse hardware environments, including graphics processing units (GPUs) and multi-threaded CPUs. LAMMPS achieves dynamic load balancing using the recursive coordinate bisectioning (RCB) algorithm [26]. This algorithm plays a critical role in optimally distributing computational workloads, thus enhancing overall system efficiency by maintaining a balanced resource allocation. The second program considered in this study is Mirheo [27], a high-performance package for microfluidics simulations on multi-GPU architectures. Mirheo includes a detailed and accurate model of blood that was extensively validated against experimental data in [28]. Mirheo is an extension of uDeviceX [29], a finalist for the Gordon Bell Prize in 2015, and was extensively optimized for the Nvidia P100 GPUs on the Cray XC50 Piz-Daint supercomputer at CSCS. It demonstrated unprecedented time-to-solution and excellent weak scalability up to thousands of GPUs for blood simulations, as well as pure DPD simulations on supercomputers.

Contributions.

1. We demonstrate that cloud computing can democratize access to tightly coupled large-scale scientific simulations.
2. We show that cloud-based simulations can achieve competitive performance compared to traditional supercomputers, both with CPU and GPU programs.

3. We introduce improvements to data structures and communication patterns in the widely used software LAMMPS, significantly reducing all-to-all communication and message sizes between ranks. These enhancements enable excellent weak and strong scalability across thousands of cores.

4. We show that cloud computing enables complex and tightly coupled simulations at scale. In this study, we focus on the example of sub-micron resolution blood flow simulations, but we emphasize that this methodology extends to many scientific applications.

2. Methods

Blood is modeled as a suspension of RBCs in plasma. Each RBC consists of a viscoelastic membrane that encloses the cytosol, a fluid six times more viscous than plasma. The ABF is treated as a rigid body with a magnetic moment, enabling its orientation and movement to be controlled by external magnetic fields. Fluids inside and outside the RBC are modeled using the DPD method. The RBC membranes are discretized as particles that form a triangle mesh, while the ABF consists of frozen DPD particles moving as a rigid object.

2.1. Dissipative particle dynamics

Fluids are discretized into DPD particles that have positions \mathbf{r}_i , velocities \mathbf{v}_i , and mass m , $i = 1, 2, \dots, N$. The particles evolve according to Newton's laws of motion and interact with neighboring particles within a cut-off radius r_c . These interactions are governed by pairwise forces, which consist of three distinct terms [21,30],

$$\mathbf{F}_{ij} = aw(r_{ij})\mathbf{e}_{ij} - \gamma(\mathbf{e}_{ij} \cdot \mathbf{v}_{ij})w_D(r_{ij})\mathbf{e}_{ij} + \sigma\xi_{ij}w_R(r_{ij})\mathbf{e}_{ij},$$

where $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$, $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$, $r_{ij} = \|\mathbf{r}_{ij}\|$ and $\mathbf{e}_{ij} = \mathbf{r}_{ij}/r_{ij}$. The coefficients a , γ , and σ are the conservative, dissipative, and random force magnitudes, respectively. Furthermore, we use the standard DPD conservative kernel

$$w(r) = \begin{cases} 1 - r/r_c, & r < r_c, \\ 0, & \text{otherwise.} \end{cases}$$

We set $w_D = w^{1/4}$ to obtain a high solvent viscosity [31]. w_R satisfies the fluctuation-dissipation theorem, $\sigma^2 = 2\gamma k_B T$ and $w_D = w_R^2$ [30], where $k_B T$ is the temperature of the system in energy units. The DPD method allows for the incorporation of complex wall boundaries. Walls are modeled as a layer of frozen DPD particles that interact with the other particles of the simulations via DPD forces. Furthermore, solvent particles are bounced-back from the surface of the walls.

2.2. Red blood cell membranes

We discretize the RBC membrane into a triangulated mesh, composed of N_v vertices and N_s edges [32]. The elasticity of the spectrin cytoskeleton is decomposed into two terms: a shear energy modeled by in-plane elastic forces between vertices that share an edge and a stretch energy modeled by a local area potential. The presence of the lipid bilayer is modeled through three different terms: (i) resistance to bending, incorporated through an energy potential whose magnitude depends on the angle between neighboring triangles, (ii) viscosity, modeled through viscous dissipation on the springs, and (iii) membrane incompressibility, represented by a global area penalization. Finally, the incompressibility of the enclosed hemoglobin is represented by a volume penalization, as the solvent is not strictly modeled as an incompressible fluid. The total potential energy of the RBC membrane is therefore composed of four terms:

$$U = U_s + U_b + U_A + U_V.$$

U_s accounts for the energy of the elastic spectrin network of the RBC membrane, modeled by an attractive worm-like chain potential and a

repulsive potential such that a non-zero equilibrium spring length can be obtained,

$$U_s = \sum_{j=1}^{N_s} \left[\frac{k_s l_m (3x_j^2 - 2x_j^3)}{4(1-x_j)} + \frac{k_{p_j}}{l_j} \right],$$

where k_s is a spring constant, l_j is the length of spring j , $x_j = l_j/l_m$, l_m is the maximum spring extension, and k_{p_j} is computed such that the total spring force on each spring is zero at equilibrium ($l_j = l_{0_j}$). The bending energy term, U_b , represents the bending resistance of the lipid bilayer and is modeled as

$$U_b = k_b \sum_{j=1}^{N_s} [1 - \cos \theta_j],$$

where k_b is a bending coefficient and θ_j is the angle between two adjacent triangles. U_A and U_V represent the area and volume conservation constraints, respectively,

$$U_A = \frac{k_a(A - A_0)^2}{2A_0} + \sum_{j=1}^{N_t} \frac{k_d(A_j - A_{0_j})^2}{2A_{0_j}}, \quad (1)$$

$$U_V = \frac{k_v(V - V_0)^2}{2V_0}, \quad (2)$$

where A_j and A_{0_j} are the current and initial area of triangle j , A and A_0 are the current and initial total membrane area, V and V_0 are the current and initial volume enclosed by the membrane, and k_a , k_d , and k_v are coefficients for the global area, local area, and volume, respectively.

The viscous dissipation of the membrane is modeled by adding a dissipative force term to the springs. We use the membrane viscosity formulation presented in Fedosov et al. [32] and set the non-central part of the force to zero ($\gamma^T = 0$), as this term does not conserve angular momentum. In this case, the dissipative force on a spring that connects vertices i and j is

$$\mathbf{F}_{m,ij}^D = -\gamma^C (\mathbf{v}_{ij} \cdot \mathbf{e}_{ij}) \mathbf{e}_{ij}$$

where \mathbf{e}_{ij} is the unit vector along the membrane vertex centers, $\mathbf{e}_{ij} = \mathbf{r}_{ij}/\|\mathbf{r}_{ij}\|$, and $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$, with \mathbf{r}_i being the positional vector of vertex i .

The parameters of the model were obtained by combining multiple datasets obtained from experiments [33,34].

2.3. Artificial bacterial flagella

The ABF is represented as a rigid body with a magnetic moment \mathbf{m} . The ABF is discretized with a set of frozen particles and a surface, similar to the representation of walls. The magnetic moment, frozen particles and surface of the ABF are constant in the frame of reference of the body, and the velocity of the particles follow that of the rigid body. DPD particles are bounced-back from the surface of the ABF and their change in velocity is converted into the ABF momentum. Furthermore, the DPD particles interact with the frozen particles via DPD interactions. The sum of these interactions are accumulated into a force and torque,

$$\mathbf{F}_{DPD} = \sum_{i=1}^N \mathbf{f}_i, \quad \mathbf{T}_{DPD} = \sum_{i=1}^N (\mathbf{r}_i - \mathbf{R}) \times \mathbf{f}_i,$$

where \mathbf{r}_i are the positions of the frozen particles, \mathbf{R} is the center of mass of the ABF and, \mathbf{f}_i the forces acting on these particles. The position of the center of mass, the orientation, and the linear and angular momentum of the ABF are then evolved according to Newton's law of motion. In addition, the ABF is subject to the magnetic torque $\mathbf{T}_{magn} = \mathbf{m} \times \mathbf{B}$, where \mathbf{B} is the external magnetic field.

3. Implementation and parallelization strategies

The methods described in the previous section require three types of objects: single particles that are the discretization unit of fluids; membranes, composed of a fixed amount of particles connected through a

triangle mesh; rigid objects, modeling ABFs, composed of particles that are connected to the frame of reference of the rigid object. When parallelizing such systems, both LAMMPS and Mirheo take a domain decomposition approach, where each MPI rank is assigned to a rectangular cuboid region, called subdomain, of the simulation space. Objects that belong to a subdomain are stored in the memory of the associated rank. In the case of fluid particles, the notion of "belonging" to a subdomain is well defined and is handled similarly in both LAMMPS and Mirheo. In contrast, membrane particles might be assigned to different ranks, depending on the data structure.

In Mirheo, particles of the same membrane are all stored on the rank associated with the subdomain that contains the center of mass of the membrane. This choice results in a straightforward implementation of the membrane force computation, as all particles are local to this rank and thus do not require to be communicated. However, this approach leads to additional communication related to interactions between membranes and other objects. We note that the communication overlaps with computation as shown in Ref. [27], resulting in an excellent weak scalability. Similarly, in Mirheo, all particles in a single rigid object are stored on the same rank. Compared to splitting particles between ranks, this approach simplifies the implementation of the time-stepping procedure of rigid objects.

In LAMMPS, particles are distributed across subdomains according to their position, regardless of the objects they are associated with. This results in membrane or ABFs particles that can be split into multiple ranks. Thus, additional data structures were introduced to maintain the connectivity of these particles. The standard approach provided by LAMMPS, consisting of explicitly storing pairs, angles, and dihedral, would be sufficient to compute membrane forces. However, these data structures are wasteful in terms of memory as the membranes all have the same connectivity. In addition, these structures make the computation of the area and volume of membranes, required for the computation of the membrane forces, cumbersome and poorly scalable if not carefully implemented. In this section, we outline the key implementation details that were essential to retaining scalability in LAMMPS. We remark that these implementation details aim to reduce the communication cost between nodes, and are thus beneficial for both traditional HPC and cloud platforms.

3.1. Implementation of red blood cell membranes

Due to the implementation details of the domain decomposition algorithm, particles of the same membrane may be distributed among several processors. The computation of the membrane forces thus requires communication between ranks. We distinguish two types of communications. First, particles are communicated to neighboring ranks to compute the forces on bonds, triangles, and dihedrals that are local to the membrane, e.g. shear and bending forces. We reuse the implementation in LAMMPS that exchanges particles that are within a cutoff radius from each rank. An important remark is that the connectivity of all membranes is the same, and thus we can easily reconstruct triangles and bonds by communicating the index of the membrane particles, with no extra communication required for the connectivity. Using the LAMMPS approach of storing all angles and bond indices explicitly would result in a much larger memory footprint and would be redundant in this case. Second, the penalization terms (Eqs. (1) and (2)) require the total area and volume of each cell. These quantities are computed in two steps. First, each rank computes the partial areas and signed volumes associated with each triangle, from the local particles and those coming from neighboring ranks within a prescribed cutoff. Second, we communicate the partial sums of the areas and volumes with neighboring ranks and sum up the results in the process, resulting in the total area and volume of the cells on each rank. These communication patterns do not involve any all-to-all collective calls, resulting in better scaling over many ranks. This communication pattern assumes that membranes are not larger than one subdomain length, i.e. membranes are stored in at most

two consecutive ranks along each direction. In practice, subdomains are larger than the size of one membrane, and thus this assumption is not restrictive. An alternative approach relies on all-to-all communications, summing up partial volumes and areas of all membranes of the system. We will demonstrate in [Section 6](#) that this alternative approach shows poor scaling compared to the method we have described above.

3.2. Implementation of artificial bacterial flagella

The implementation of ABFs follows the approach of rigid fix in LAMMPS. This approach attaches a rigid object index to each particle in the domain and a shared state for each object (center of mass, orientation, linear, and angular velocities) stored on every rank of the simulation. This approach allows for simulating rigid objects that are larger than the subdomain associated with one rank. However, the implementation relies on all-to-all communication patterns that do not scale well with the number of objects and the number of ranks in the simulations when the rigid object is much smaller than the simulation domain. Since the ABF has a size of the order of one single RBC, only a few ranks need to update this ABF. We modified the implementation of the rigid fix in LAMMPS so that only those ranks are involved in the communication, improving the scaling efficiency of the program (see [Section 6](#)). The set of ranks involved in the update of the ABF changes as the ABF moves within the domain. We update this set of ranks at a given frequency that is much lower than the time-stepping frequency but high enough to keep the number of ranks in this set relatively low. This approach substantially reduces the number of all-to-all communications, and thus the scalability of the program is better than the original approach used in LAMMPS, as we will see in [Section 6](#).

4. Description of the compute environments

We use two platforms in this study: Google Cloud Platform (GCP) and Piz Daint at the CSCS supercomputing center.

The cloud benchmarks are run on an SLURM cluster running on GCP. The GPU benchmarks are performed on a partition built out of n1-highmem-16 nodes, each equipped with 4 Nvidia T4 GPUs. The LAMMPS benchmarks use c2-standard-60 nodes, each equipped with a 60-core Intel® Xeon® CPU, running at a base frequency of 3.10 GHz. The cluster was built using Google's "HPC Toolkit" [35], with the addition of a GCP Filestore instance for persistent data storage. The software was installed using SPACK package managers [36] and includes MPICH, CUDA, cmake and HDF5 libraries.

The Piz Daint supercomputer has two partitions. The GPU partition is a Cray X50 system with Intel® Xeon® E5-2690 v3 CPUs, running at a base frequency of 2.60 GHz (12 cores, 64 GB RAM) and NVIDIA® Tesla® P100 16 GB. The multicore partition is a Cray X40 system with two Intel® Xeon® E5-2695 v4 running at a base frequency of 2.10 GHz (2 × 18 cores, 64/128 GB RAM). The interconnect configuration is based on Aries routing and communications ASIC, with Dragonfly network topology.

5. Performance of Mirheo on the cloud

We perform scaling experiments on the cloud with tightly coupled simulations using Mirheo. [Fig. 1](#) shows the weak scaling of pure bulk DPD solvent with a number density $n_d = 10r_c^{-3}$, where $r_c = 1$ is the cut-off radius of the DPD interactions. The weak scaling experiment was repeated for subdomain volumes of $(96r_c)^3$ and $(128r_c)^3$ per GPU (about 8.8 and 20.1 million particles per GPU, respectively). Mirheo has a weak scaling efficiency above 80 % for a subdomain volume $(128r_c)^3$ per GPU up to 512 GPUs (128 nodes). In comparison, simulations on Piz Daint exhibit a weak scaling efficiency above 98 % up to 729 GPUs, due to the better bandwidth between nodes.

Similarly, we study the weak scaling of a suspension of RBCs at hematocrit $Ht = 30\%$ on the same platforms ([Fig. 1](#)). The weak scaling ef-

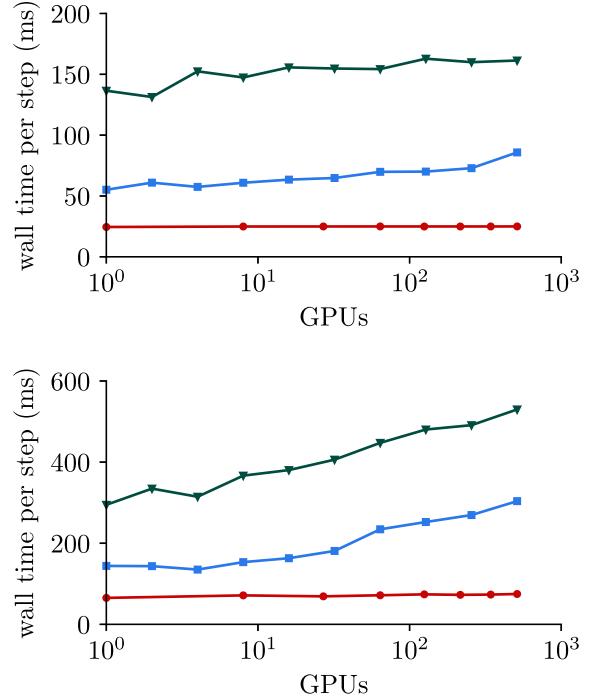


Fig. 1. Weak scaling of Mirheo on the cloud with 4 T4 GPUs per node and on Piz Daint with one P100 GPU per node. —●—: Piz daint, $(96r_c)^3$ volume per GPU. —■—: GCP, $(96r_c)^3$ volume per GPU. —▲—: GCP, $(128r_c)^3$ volume per GPU. Top: DPD particles only. Bottom: Bulk blood.

ficiency is lower than that obtained from pure DPD bulk simulations, but remains above 50 % despite the complexity of the simulations. The degradation of the weak scaling efficiency compared to the previous case originates from extra communication due to the bounce back of the particles against the membrane of the RBCs. This operation cannot be overlapped with any computation. In addition, the size of the messages is larger due to the exchange of membrane particles between neighboring ranks.

[Fig. 2](#) shows the strong scaling performance of Mirheo on the cloud for a suspension of DPD particles and the more complex scenario of blood suspension at $Ht = 30\%$. In the pure DPD case, the speedup is almost ideal from 8 to 64 GPUs, and degrades for more than 128 GPUs. In the case with blood cells, the speedup is close to ideal in the same range as the DPD case only for the larger domain size of $(384r_c)^3$, while the smaller domain size degrades above 32 GPUs. As expected, the speed-up is closer to ideal for larger domain sizes, since there is more work per GPU. The case including blood cells shows worse strong scaling than the pure DPD case, caused by the higher amount of communication required to exchange membrane information, the larger number of messages due to the different types of solvent particles (cytosol and plasma), and the additional communication due to bounce-back on membranes.

6. Performance of LAMMPS on the cloud

We examine the scaling capabilities of LAMMPS on the C2-standard partition. [Fig. 3](#) shows the weak scaling efficiency of LAMMPS for pure DPD dynamics and for a 30 % hematocrit blood suspension. The simulations of pure DPD solvent are set in a periodic domain, with a number density of $n_d = 10r_c^{-3}$, where r_c is the cut-off radius of the DPD interactions. Each node consists of 30 physical cores per subdomain of volume $(128r_c)^3$. The suspension of RBCs has a hematocrit of 30 %. The RBC membranes are composed of 642 vertices and their size corresponds to an equivalent radius of $R_A = 3r_c$, with $R_A = \sqrt{A_0/(4\pi)}$. A_0 is the

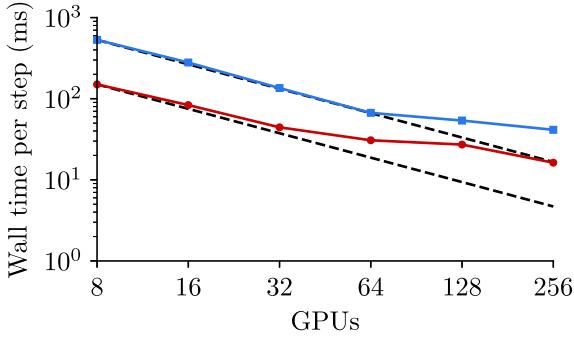


Fig. 2. Strong scaling of Mirheon on the cloud with 4 T4 GPUs per node. —●— : $(256r_c)^3$ volume, —■— : $(384r_c)^3$ volume, - - - : ideal. Top: DPD particles only. Bottom: Bulk blood.

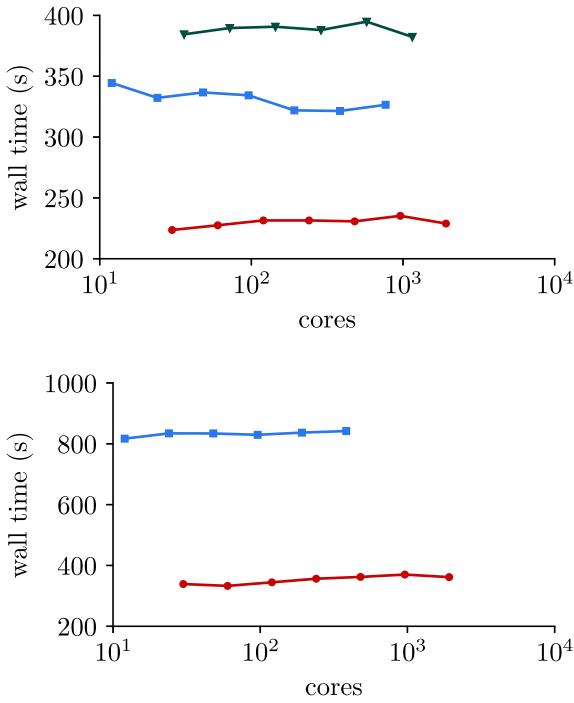


Fig. 3. Weak scaling performance of LAMMPS. Top: DPD only, $(128r_c)^3$ volume per node, —●— : GCP c2-standard-60 partition, 30 cores per node, —■— : Piz Daint GPU-partition 12 cores per node, —▲— : Piz Daint multi-core partition, 36 cores per node. Bottom: Blood suspension with 30 % hematocrit, $(128r_c)^3$ volume per node, —●— : GCP c2-standard-60 partition, 30 cores per node, —■— : Piz Daint GPU-partition 12 cores per node.

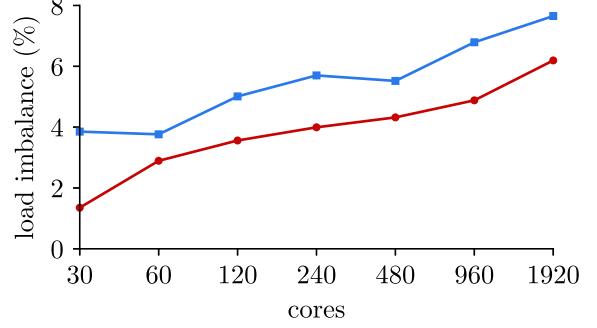


Fig. 4. Weak scaling of LAMMPS on the C2 partition: load imbalance for a volume of $(128r_c)^3$ per node. —●— : Pure DPD particles, —■— : Blood suspension with 30 % hematocrit.

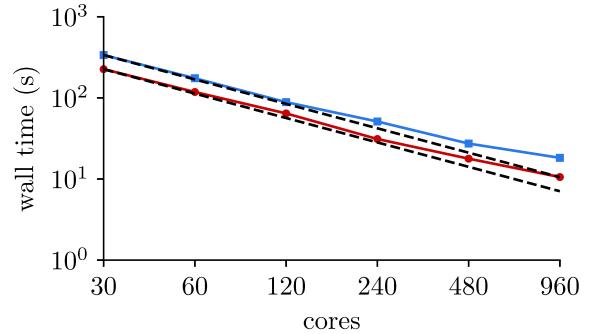


Fig. 5. Strong scaling of LAMMPS on the C2 partition with $(128r_c)^3$ volume. —●— : DPD particles only. —■— : Bulk blood at 30 % hematocrit. - - - : ideal.

membrane area. Each simulation is repeated five times, and the minimum wall time over 100 steps is recorded. The scaling efficiency remains above 95 % for pure solvent and above 90 % for the more complex case of a blood suspension, up to 1920 cores (64 nodes). LAMMPS has a similar weak scaling efficiency on two partitions of the Piz Daint supercomputer, although the CPU frequencies are lower and thus the time to solution is higher on this platform. Fig. 4 shows the load imbalance for both pure DPD dynamics and blood suspension on GCP. The load imbalance was measured by enabling the *timer full sync* option of LAMMPS, which synchronizes all ranks after each phase of the simulation time steps, by performing a collective barrier call. The synchronization time was measured and normalized by the simulation time, forming the load imbalance. We note that this quantity is an upper bound of the load imbalance, as this also includes the time to perform the collective barrier calls. Each simulation was run five times and the minimal load imbalance was reported. The load imbalance remains below 8 % for all simulations performed and is, as expected, lower for the pure DPD solvent than for the blood suspension. Furthermore, the load imbalance deteriorates as the number of cores increases.

Finally, Fig. 5 shows the strong scaling performance of LAMMPS on the C2 partition. The simulation consists of the same cases used during the weak-scaling study, in a domain of fixed volume $(128r_c)^3$. The simulations show excellent strong scaling as the speed-up remains close to ideal up to 480 cores (16 nodes), for both cases considered in this section. Beyond 480 cores, the strong scaling efficiency decreases. This is likely due to the load imbalance, as seen in Fig. 4, and increased communication latency, which becomes more significant compared to the computation load as the number of cores increases. To further improve strong scaling efficiency, strategies include hybrid MPI/OpenMP approach to reduce the number of messages and

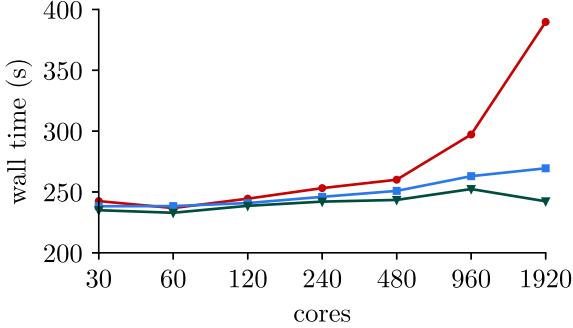


Fig. 6. Weak scalability of a $(128r_c)^3$ blood simulation (similar to Fig. 3) with different data structures to store the RBC membrane connectivity. —●— : Default LAMMPSapproach, —■— : local reconstruction of connectivity with collective communication, —▼— : local reconstruction of connectivity with neighboring communication.

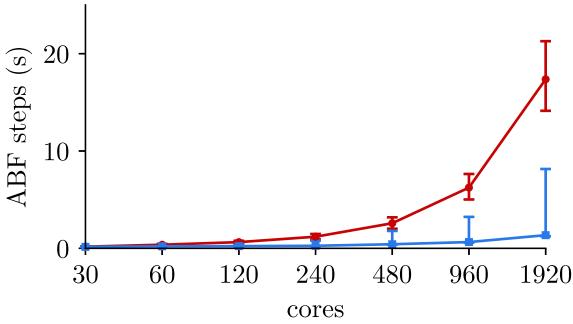


Fig. 7. Time spent evolving the ABF against number of cores while increasing the domain size proportionally, using *full sync* timer option in LAMMPS. The curve denotes the average time per core, and the error bars show the minimal and maximal time among all cores. —●— : Default LAMMPSimplementation, —■— : Sub-communicator implementation.

asynchronous communication to better overlap communication with computation.

Fig. 6 demonstrates the benefits of not explicitly storing the RBCs connectivity for each membrane. Using LAMMPSstructures (bonds, angles and dihedrals) leads to larger messages and a larger memory footprint, which is unnecessary as all RBC membranes have the same connectivity. Instead, our approach reconstructs the connectivity of the triangle mesh on each rank at every iteration from the particle global indices without communicating them. This approach decreases the message sizes between ranks and the memory footprint of the simulations. We tested the different implementations on a blood suspension with $Ht = 30\%$ and a volume per node (30 cores) of $(128r_c)^3$, with membranes of effective radius $R_A = 3r_c$ and 642 vertices per membrane. It is clear from Fig. 6 that the new approach scales much better than using the LAMMPSstructures.

The calculation of membrane forces requires one to calculate the volume and area of each membrane at each time step, as shown by Eqs. (1) and (2). Due to the domain partitioning, RBCs may be split between multiple ranks, therefore the initial implementation used collectives to calculate the area and volume of cells. Since RBCs have a finite size, usually only a few ranks are involved in the computation of each RBC area and volume. Thus, we exchange the partial volume and areas of the membranes only with the ranks containing parts of the membranes. This approach further improves the weak scaling of the blood suspension simulations compared to using an all-to-all communication pattern, as shown on Fig. 6.

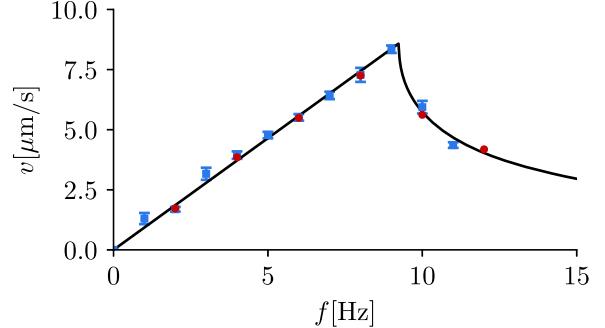


Fig. 8. Swimming speed of an ABF in a viscous fluid against the rotation frequency of the external magnetic field. ● : DPD simulations; ■ : experimental measurements from Mhanna et al. [37]; — : fit to the experimental data with the ODE model explained in refs. [38,39].

Table 1

Comparison of strong scaling efficiency between Palabos (from AWS benchmarks [20]) and LAMMPS (this work) for fluid and blood simulations, against the number of instances N .

N	Fluid		Blood	
	Palabos	LAMMPS	Palabos	LAMMPS
1	—	—	1.00	1.00
2	1.00	1.00	0.89	0.97
4	0.96	0.92	0.74	0.95
8	0.90	0.95	0.57	0.83
16	0.81	0.83	—	0.77
32	0.74	0.70	—	0.58
64	0.69	—	—	—
128	0.57	—	—	—

Fig. 7 shows the time spent updating a single ABF in a pure DPD solvent over 1000 steps. The volume of the domain increases proportionally to the number of cores, and we use a volume of $(32r_c)^3$ per node (30 cores). The default approach used in LAMMPSuses collective calls, resulting in poor scalability. We improved the communication pattern by restricting collective communication calls to only the ranks in the vicinity of the ABF particles (see Section 3.2). This approach results in much better scalability than the original LAMMPSapproach. The measurements were performed with the synchronized and nonsynchronized timer options in LAMMPS, and no major differences were found between the two, suggesting that the load imbalance between ranks in earlier stages does not influence these results. We note that the maximum time spent per rank for this stage of the simulation is not negligible, as this timing takes into account the computation part of the update. However, the minimum and average time across all ranks remain very low with the improved communication pattern, even when the number of ranks is large.

We remark that while LAMMPShas better weak scalability than Mirheo in the above tests, the time to solution of Mirheois much lower. On the pure DPD dynamics in a volume of $(128r_c)^3$ per T4 GPU, Mirheo exhibits a wall time of 130 ms per time step on one GPU while LAMMPStakes about 2.23 s per step on one node (17 times longer). Thus, the communication time represents a larger part of the simulations in Mirheothan in LAMMPS, possibly explaining the lower weak scaling efficiency in this specific case.

Finally, we compare the strong scaling efficiency of LAMMP-Swith that of Palabos [19]. Palabos was benchmarked on AWS EC2 Hpc7g instances with 64 cores per instance [20] for two representative cases: a pure fluid system, with 1 billion grid cells, and a blood flow case with 476 RBCs and 95 platelets. As shown in Table 1 Palabos and LAMMPSexhibit comparable strong scaling

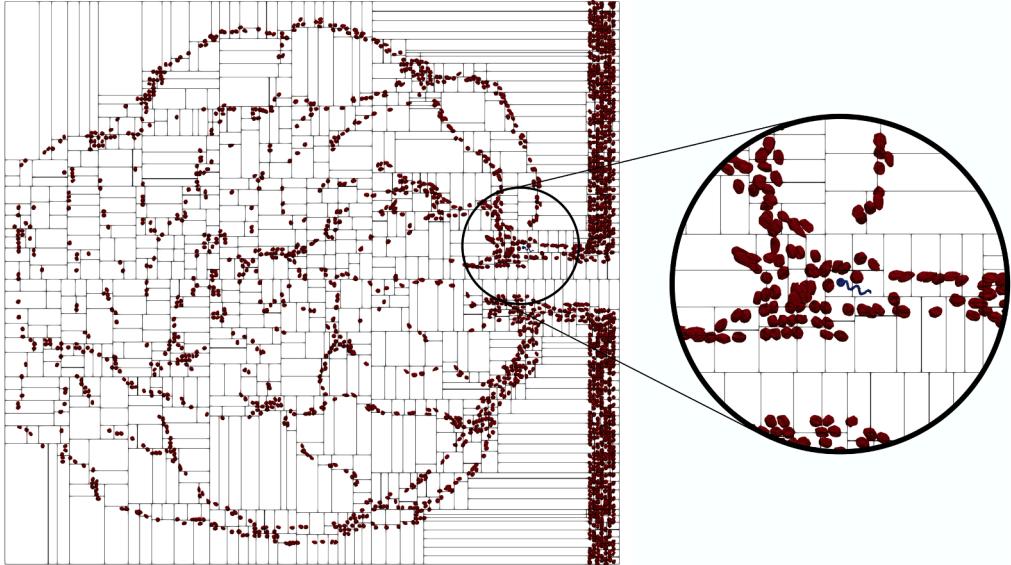


Fig. 9. Simulation of an ABF swimming through retinal capillaries. Capillary walls and solvent particles are not shown, for clarity. The rectangular boundaries represent the domain decomposition (one MPI rank per subdomain).

efficiency for the fluid case, despite Palabos being run at a larger problem size (1 billion grid cells versus 20 million particles in our simulations). In the more complex case of blood, where we implemented algorithmic improvements for the membrane communication patterns, LAMMPS achieved higher strong scaling efficiency. This improvement is also supported by the larger problem size considered in this work (7489 RBCs compared to 476 in the Palabos study), which provides more computational work per core and thus better scaling efficiency.

7. Cost efficiency

In this section, we compare the cost efficiency of the simulations benchmarked in Sections 5 and 6. Specifically, we evaluate the average number of simulation steps achieved per unit cost, as summarized in Table 2. In all cases, Piz Daint achieves a higher number of simulation steps per unit cost than GCP. However, the gap is relatively modest: GCP's cost efficiency is within a factor of 2 compared with Piz Daint.

It is important to note that the reported cost on Piz Daint (CHF 0.40 per node-hour) is heavily subsidized, as is typical for publicly funded supercomputing centers. This cost does not reflect the full operational expenses of the system, which include infrastructure, personnel, and energy costs. In contrast, the prices on GCP represent full commercial rates, including overheads and profit margins. From this perspective, the cost efficiency of cloud resources appears competitive, especially considering the flexibility, scalability, and accessibility they provide.

Table 2
Cost efficiency (in number of simulation steps per dollar) of the DPD and blood cases obtained with LAMMPS and Mirheon both the Piz Daint supercomputer and GCP.

Case	Piz Daint X50	GCP
Mirheo, DPD, $(96r_c)^3$	306 115	196 818
Mirheo, blood, $(96r_c)^3$	115 024	72 439
LAMMPS, DPD, $(128r_c)^3$	2177	1028
LAMMPS, blood, $(128r_c)^3$	918	678

8. ABF in a retinal capillary network

The benchmarks presented in the previous sections demonstrate the weak and strong scaling capabilities of LAMMPS on GCP for homogeneous, isotropic cases (periodic DPD fluids and blood suspensions). However, more realistic systems include complex geometries and large-scale simulations, possibly coupled with additional physics, and complex flows. In this section, we demonstrate that our approach generalizes to such cases by simulating the evolution of an ABF swimming in a retinal capillary network.

First, to validate the ABF model, we simulate a single swimmer in a viscous fluid with zero background velocity. The domain is periodic with large enough dimensions to avoid interactions between the ABF and its periodic images. A uniform, rotating magnetic field with constant frequency f propels the swimmer to an average velocity v . The swimming speed of ABFs in such conditions has been measured experimentally [37] and explained with an ODE model [38,39]. Fig. 8 demonstrates that the DPD model used in this work has an excellent agreement with both theory and experiments: below a critical frequency of the magnetic field, the swimming speed of the ABF increases linearly; above the critical frequency, however, the time-averaged speed of the ABF decreases due to the bounded magnetic torque [24,38,39].

We now extend these results to the case of an ABF in a retinal capillary network. The geometric representation of the capillaries is reproduced from a fundus image of a retinal capillary network [40]. The simulations are carried out with LAMMPS with a tiled domain decomposition to ensure load balancing between ranks. The ABF is guided by an external magnetic field, controlled by a reinforcement learning (RL) agent. The RL agent was trained as described in [25] from a reduced-order model of the simulations. The policy of the RL agent maps the center of mass of the swimmer to the desired swimming direction \mathbf{p} . We thus apply a magnetic field rotating in a plane perpendicular to the swimming direction,

$$\mathbf{B}(t) = R_x(\mathbf{p}) \begin{pmatrix} \cos \omega t \\ \sin \omega t \\ 0 \end{pmatrix},$$

where $R_x(\mathbf{p})$ is the rotation that transforms the vector \mathbf{e}_x into the swimming direction \mathbf{p} with axis of rotation $\mathbf{e}_x \times \mathbf{p}$, and \mathbf{e}_x is the principal axis of the non-rotated ABF. The magnetic field is adapted at every time step of the simulation.

The largest capillary has a radius of $15\text{ }\mu\text{m}$. A force field is applied to every particle to mimic a pressure gradient. The magnitude of this force is set to achieve a mean inflow velocity of 1 mm s^{-1} . The hematocrit is set to 20 %. These numbers are typical for blood flow in capillaries [41]. The magnetic moment has a magnitude of $m = 1 \times 10^{-11}\text{ N m T}^{-1}$ and the external magnetic field has a magnitude of $B = 3\text{ mT}$ and a rotation frequency $f = 1\text{ kHz}$. The resolution of the DPD solvent corresponds to a number density of $10r_c^{-3}$, with $R_A = 4r_c$, resulting in approximately 90 million particles. We employ the C2 partition with 30 physical cores per node, on a total of 2490 cores (83 nodes). Fig. 9 shows a snapshot of the ABF swimming through the complex network of capillaries. We note that in this case the tiled domain decomposition is crucial to saving computational resources, as the geometry is relatively sparse.

9. Limitations

In this work, we have demonstrated that cloud platforms are well suited for large-scale, tightly coupled simulations. The two software frameworks tested show excellent weak and strong scalability, comparable to traditional supercomputers. However, these benchmarks were performed on a single rack, where the communication cost is minimal. Scaling up to simulations spanning multiple racks, network islands, or even datacenters would inevitably increase the communication latency by one or two orders of magnitude, while reducing bandwidth. In turn, the communication-to-computation ratio increases, leading to a drop in scaling efficiency. This is a general limitation of cloud infrastructures compared to specialized HPC clusters with low-latency interconnects such as InfiniBand. To mitigate these effects, hybrid parallelization strategies such as MPI/OpenMP can reduce the number of inter-node messages and help maintain efficiency by minimizing latency overhead. Additionally, careful resource placement and use of cloud-specific optimizations (e.g., placement policies) can further alleviate communication bottlenecks.

The simulations presented in this work are based on state-of-the-art models of blood at the microscale. Nonetheless, some idealizations were necessary. For instance, we have neglected the presence of white blood cells and platelets, which may affect the hydrodynamics in capillary networks. In the retinal microvasculature simulations, vessel walls were modeled as no-slip and no-through boundaries. However, *in vivo*, the endothelial glycocalyx layer and cellular permeability enable some degree of fluid and molecular exchange with surrounding tissues. These simplifications may influence the interaction of RBCs and ABFs with the vessel boundaries. While recent studies have started to incorporate such effects [42], their influence in the context of ABFs remains largely unexplored and requires further investigations.

10. Summary

We have demonstrated that the cloud is well suited for large-scale, complex, tightly coupled scientific simulations. We have tested the scalability of two programs on different cloud architectures. Mirheos shows excellent weak scalability up to 512 GPUs. Furthermore, excellent scalability was achieved with the widely used software LAMMPS for bulk DPD simulations. LAMMPS was further extended to simulate blood suspensions at physiological hematocrits and the swimming of ABFs within complex retinal capillaries. By reducing the size of point-to-point communication and minimizing collective communications, we were able to retain weak scaling efficiency above 90 % for up to 1920 cores. This was achieved by improving data-structures of the membrane connectivity, and by adapting communication patterns between nodes accordingly. We have shown that LAMMPS can simulate an ABF swimming in a complex network of capillaries that was reconstructed from fundus images of retinal capillaries, which paves the way for democratizing access to critical simulations for personalized medicine. The configuration of the cloud platform provides an environment similar to that of traditional

supercomputers. This methodology is thus reusable for any other large-scale scientific applications. These results highlight cloud computing as a viable platform that complements traditional HPC by providing accessible and competitive performance for complex, large-scale, tightly coupled simulations.

CRediT authorship contribution statement

Lucas Amoudruz: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Conceptualization; **Sergey Litvinov:** Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Conceptualization; **Riccardo Murri:** Writing – original draft, Software; **Volker Eyrich:** Writing – original draft, Software, Methodology, Conceptualization; **Jens Zudrop:** Methodology; **Costas Bekas:** Writing – original draft, Supervision, Methodology, Investigation, Conceptualization; **Petros Koumoutsakos:** Writing – original draft, Supervision, Methodology, Conceptualization.

Data availability

No data was used for the research described in the article.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

The authors thank Dan Speck, Matthew Cruger, and Mark Heil for helpful discussions and cloud support.

References

- [1] C. Vecchiola, S. Pandey, R. Buyya, High-performance cloud computing: a view of scientific applications, in: 2009 10th International Symposium on Pervasive Systems, Algorithms, and Networks, IEEE, 2009, pp. 4–16.
- [2] I. Sadooghi, J.H. Martin, T. Li, K. Brandstatter, K. Maheshwari, T.P.P. de Lacerda Ruivo, G. Garzoglio, S. Timm, Y. Zhao, I. Raicu, Understanding the performance and potential of cloud computing for scientific applications, *IEEE Trans. Cloud Comput.* 5 (2) (2015) 358–371.
- [3] A. Iosup, S. Ostermann, M.N. Yigitbasi, R. Prodan, T. Fahringer, D. Epema, Performance analysis of cloud computing services for many-tasks scientific computing, *IEEE Trans. Parallel Distrib. Syst.* 22 (6) (2011) 931–945.
- [4] Y. Zhai, M. Liu, J. Zhai, X. Ma, W. Chen, Cloud versus in-house cluster: evaluating Amazon cluster compute instances for running mpi applications, in: State of the Practice Reports, nill, 2011, pp. 1–10.
- [5] S.M. Martin, D. Wälchli, G. Arampatzis, A.E. Economides, P. Karnakov, P. Koumoutsakos, Korali: efficient and scalable software framework for Bayesian uncertainty quantification and stochastic optimization, *Comput. Methods Appl. Mech. Eng.* 389 (2022) 114264.
- [6] P.E. Hadjidoukas, P. Angelikopoulos, C. Papadimitriou, P. Koumoutsakos, Π4U: a high performance computing framework for Bayesian uncertainty quantification of complex models, *J. Comput. Phys.* 284 (2015) 1–21.
- [7] B.M. Adams, W.J. Bohnhoff, K.R. Dalbey, M.S. Ebeida, J.P. Eddy, M.S. Eldred, R.W. Hooper, P.D. Hough, K.T. Hu, J.D. Jakeman, et al., Dakota, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: version 6.13 user's manual, Technical Report, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2020.
- [8] C. Kutzner, C. Kniep, A. Cherian, L. Nordstrom, H. Grubmüller, B.L. de Groot, V. Gapsys, GROMACS in the cloud: a global supercomputer to speed up alchemical drug design, *J. Chem. Inf. Model.* 62 (7) (2022) 1691–1711.
- [9] A. Fernandez, Evaluation of the performance of tightly coupled parallel solvers and MPI communications in IaaS from the public cloud, *IEEE Trans. Cloud Comput.* 10 (4) (2021) 2613–2622.
- [10] V. Munhoz, M. Castro, L.G.C. Rego, Evaluating the parallel simulation of dynamics of electrons in molecules on AWS spot instances, in: Anais do XXIV Simpósio em Sistemas Computacionais de Alto Desempenho, SBC, 2023, pp. 205–216.
- [11] S.J.E. Taylor, A. Agnastotou, T. Kiss, G. Terstyanszky, P. Kacsuk, N. Fantini, D. Lakehal, J. Costes, Enabling cloud-based computational fluid dynamics with a platform-as-a-service solution, *IEEE Trans. Ind. Inf.* 15 (1) (2018) 85–94.
- [12] O. Mesnard, L.A. Barba, Reproducible workflow on a public cloud for computational fluid dynamics, *Comput. Sci. Eng.* 22 (1) (2019) 102–116.

- [13] P. Zaspel, M. Griebel, Massively parallel fluid simulations on Amazon's HPC cloud, in: 2011 First International Symposium on Network Cloud Computing and Applications, IEEE, 2011, pp. 73–78.
- [14] D. Siuta, G. West, H. Modzelewski, R. Schigas, R. Stull, Viability of cloud computing for real-time numerical weather prediction, *Weather Forecast.* 31 (6) (2016) 1985–1996.
- [15] J.G. Powers, K.K. Werner, D.O. Gill, Y.-L. Lin, R.S. Schumacher, Cloud computing efforts for the weather research and forecasting model, *Bull. Am. Meteorol. Soc.* 102 (6) (2021) E1261–E1274.
- [16] M. Kumar, Performance analysis of computational fluid dynamics workloads on HPC in the cloud, in: 2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT), IEEE, 2023, pp. 1–5.
- [17] T. Dancheva, U. Alonso, M. Barton, Cloud benchmarking and performance analysis of an HPC application in Amazon EC2, *Cluster Comput.* 27 (2) (2024) 2273–2290.
- [18] G.K. Boiger, D. Sharman, B. Siyahhan, V. Lienhard, M. Boldrini, D. Drew, A massive simultaneous cloud computing platform for OpenFOAM, in: 9th OpenFOAM Conference, Online, 19–20 October 2021, ZHAW Zürcher Hochschule für Angewandte Wissenschaften, 2021. <https://doi.org/10.21256/zhaw-24032>
- [19] C. Kotsalos, J. Latt, B. Chopard, Palabos-npFEM: software for the simulation of cellular blood flow (digital blood), *J. Open Res. Softw.* 9 (1) (2021) 16.
- [20] J. Tang, Lattice Boltzmann simulation with Palabos on AWS using Graviton-based Amazon EC2 Hpc7g instances, 2023, (AWS HPC Blog). Accessed: 2025-09-30, <https://aws.amazon.com/blogs/hpc/lattice-boltzmann-simulation-with-palabos-on-aws-using-graviton-based-amazon-ec2-hpc7g-instances>.
- [21] P.J. Hoogerbrugge, J. Koelman, Simulating microscopic hydrodynamic phenomena with dissipative particle dynamics, *EPL (Europhys. Lett.)* 19 (3) (1992) 155.
- [22] A.P. Thompson, H.M. Aktulga, R. Berger, D.S. Bolintineanu, W.M. Brown, P.S. Crozier, P.J. in't Veld, A. Kohlmeyer, S.G. Moore, T.D. Nguyen, et al., LAMMPS a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales, *Comput. Phys. Commun.* 271 (2022) 108171.
- [23] D.A. Fedosov, B. Caswell, G.E. Karniadakis, A multiscale red blood cell model with accurate mechanics, rheology, and dynamics, *Biophys. J.* 98 (10) (2010) 2215–2225.
- [24] L. Amoudruz, Simulations and Control of Artificial Microswimmers in Blood, Ph.D. thesis, ETH Zurich, 2022.
- [25] L. Amoudruz, S. Litvinov, P. Koumoutsakos, Optimal navigation of magnetic artificial microswimmers in blood capillaries with deep reinforcement learning, *Phys. Fluids* 37 (7) (2025) 071703. <https://doi.org/10.1063/5.0274623>
- [26] Bokhari, A partitioning strategy for nonuniform problems on multiprocessors, *IEEE Trans. Comput.* 100 (5) (1987) 570–580.
- [27] D. Alexeev, L. Amoudruz, S. Litvinov, P. Koumoutsakos, Mirheo: high-performance mesoscale simulations for microfluidics, *Comput. Phys. Commun.* 254 (2020) 107298.
- [28] L. Amoudruz, A. Economides, G. Arampatzis, P. Koumoutsakos, The stress-free state of human erythrocytes: data-driven inference of a transferable RBC model, *Biophys. J.* (2023). <https://doi.org/10.1016/j.bpj.2023.03.019>
- [29] D. Rossinelli, Y.-H. Tang, K. Lykov, D. Alexeev, M. Bernaschi, P. Hadjidoukas, M. Bisson, W. Joubert, C. Conti, G. Karniadakis, et al., The in-silico lab-on-a-chip: petascale and high-throughput simulations of microfluidics at cell resolution, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2015, pp. 1–12.
- [30] P. Espanol, P. Warren, Statistical mechanics of dissipative particle dynamics, *EPL (Europhys. Lett.)* 30 (4) (1995) 191.
- [31] X. Fan, N. Phan-Thien, S. Chen, X. Wu, T. Yong Ng, Simulating flow of DNA suspension using dissipative particle dynamics, *Phys. Fluids* 18 (6) (2006) 063102.
- [32] D.A. Fedosov, B. Caswell, G.E. Karniadakis, A multiscale red blood cell model with accurate mechanics, rheology, and dynamics, *Biophys. J.* 98 (10) (2010) 2215–2225.
- [33] A. Economides, G. Arampatzis, D. Alexeev, S. Litvinov, L. Amoudruz, L. Kulakova, C. Papadimitriou, P. Koumoutsakos, Hierarchical Bayesian uncertainty quantification for a model of the red blood cell, *Phys. Rev. Appl.* 15 (3) (2021) 034062.
- [34] L. Amoudruz, P. Koumoutsakos, Independent control and path planning of microswimmers with a uniform magnetic field, *Adv. Intell. Syst.* 4 (3) (2022) 2100183.
- [35] Google, Cloud HPC Toolkit Documentation, 2023. Accessed: 2024-04-02, <https://cloud.google.com/hpc-toolkit/docs>.
- [36] T. Gamblin, M. LeGendre, M.R. Collette, G.L. Lee, A. Moody, B.R. De Supinski, S. Futral, The SPACK package manager: bringing order to HPC software chaos, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2015, pp. 1–12.
- [37] R. Mhanna, F. Qiu, L. Zhang, Y. Ding, K. Sugihara, M. Zenobi-Wong, B.J. Nelson, Artificial bacterial flagella for remote-controlled targeted single-cell drug delivery, *Small* 10 (10) (2014) 1953–1957.
- [38] D. Schamel, M. Pfeifer, J.G. Gibbs, B. Miksch, A.G. Mark, P. Fischer, Chiral colloidal molecules and observation of the propeller effect, *J. Am. Chem. Soc.* 135 (33) (2013) 12353–12359.
- [39] P.J. Vach, N. Brun, M. Bennet, L. Bertinetti, M. Widrat, J. Baumgartner, S. Klumpp, P. Fratzl, D. Faivre, Selecting for function: solution synthesis of magnetic nanopropellers, *Nano Lett.* 13 (11) (2013) 5373–5378.
- [40] P. Ghassemi, J. Wang, A.J. Melchiorri, J.C. Ramella-Roman, S.A. Mathews, J.C. Coburn, B.S. Sorg, Y. Chen, T. Joshua Pfeifer, Rapid prototyping of biomimetic vascular phantoms for hyperspectral reflectance imaging, *J. Biomed. Opt.* 20 (12) (2015) 121312–121312.
- [41] Y.-C. Fung, Biomechanics: circulation, *Shock* 9 (2) (1998) 155.
- [42] X.Z. Jiang, K.H. Luo, Y. Ventikos, Understanding the role of endothelial glycocalyx in mechanotransduction via computational simulation: a mini review, *Front. Cell Dev. Biol.* 9 (2021) 732815.