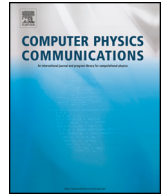




ELSEVIER

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

Computational Physics

Implementation of a direct-addressing based lattice Boltzmann GPU solver for multiphase flow in porous media [☆]Guang Yang ^a, Yu Chen ^b, Simeng Chen ^b, Moran Wang ^{a,*}^a Department of Engineering Mechanics, Tsinghua University, Beijing 100084, China^b Department of Mechanics and Aerospace Engineering, Southern University of Science and Technology, Shenzhen 518055, China

ARTICLE INFO

Article history:

Received 13 October 2022

Received in revised form 18 May 2023

Accepted 13 June 2023

Available online 16 June 2023

Keywords:

Lattice Boltzmann

Multiphase flow

GPU computing

Direct addressing

ABSTRACT

GPU accelerated lattice Boltzmann (LB) simulations of multiphase flow in porous media have become a powerful tool to study fluid displacement process in porous media. For porous structures with a very low porosity, indirect-addressing memory access methods are preferred due to significantly reduction of the memory footprint despite that those methods are more difficult to implement and the resulting performance may be more sensible to the computing architectures, such as cache hierarchy and size. The direct-addressing methods are straightforward to implement and are able to archive high throughput when the porosity is large, while the methods become less efficient when the structures are too sparse. Nevertheless, the direct-addressing methods combined with multi-block grid technique are promising for many applications. In this work, we present a hybrid-way to tackle multiphase flow simulations in porous media, where the direct-addressing method is employed for the main LB evolution while the indirect-addressing method is employed for the complex boundary conditions. The CSF-based LB color-gradient multiphase model and the geometry-based wetting model are employed to increase accuracy and stability. Thanks to the utilization of AA-pattern streaming scheme, non-slip boundary condition of arbitrary orientation can be enforced without extra cost. We perform comprehensive analysis of the computational performance of the present solver on NVIDIA GPUs. For typical sandstones, our results show that the present implementation is able to achieve over 1.5 speedup compared with other direct-addressing schemes on V100 and A100, respectively and, particularly, the computational performance of the boundary kernels is greatly increased thanks to the increased L2 cache size of the latest GPUs.

© 2023 Elsevier B.V. All rights reserved.

1. Introduction

Multiphase flows in porous media are ubiquitous and significant in energy and environment industries, such as enhanced oil recovery (EOR) [1] and carbon capture and storage (CCS) [2]. Lattice Boltzmann method (LBM) is one of the most popular methods to simulate multiphase flows in porous media due to its ability to handle complex geometry and interfacial dynamics and achieve high computational efficiency on modern computing architectures [3–6]. In order to correctly study the 3D multiphase flow mechanism in porous media and meet the requirement of representative elementary volume (REV) for upscaling, it is crucial to develop high performance multiphase LBM solvers [7,8].

General purpose graphics process unit (GPGPU) computing is a promising approach to improve LBM performance. Compared to

traditional CPUs, GPUs prioritize high throughput by using thousands of simple but cheap computing cores resulting much higher raw computing power and memory bandwidth [9], which makes them suitable for certain numerical schemes. LBM is an explicit method that requires extremely high memory bandwidth. As a result, LBM is particularly successful on GPUs. In fact, LBM is one of the earliest algorithms to be ported to GPUs [10,11]. Optimizations of LBM algorithm concerning memory layout [12–15], access pattern [16–18], tiling technique [12,19] and register usage [12,20] can be found in literatures.

For sparse geometry in porous media, many researchers suggest using the indirect addressing methods, also known as the pore-list scheme for porous media [6,14,15,19]. In indirect addressing methods, only the information of fluid nodes is stored by a pore-list. Since only fluid data and neighboring information are stored, this approach reduces memory usage and potentially reduces memory IO when the porous structure has a low porosity. However, as indirect addressing induces inefficient and uncoalesced memory

[☆] The review of this paper was arranged by Prof. David W. Walker.

* Corresponding author.

E-mail address: mrwang@mail.tsinghua.edu.cn (M. Wang).

transactions, careful and complicated tiling is required to achieve good performance [19].

On the other hand, the direct addressing method, also known as the pore-matrix scheme, allocates memory for both fluid and solid nodes while an additional mask array (matrix) is used to identify node type [6]. As a result, by choosing the Structures-of-Arrays (SoA) type data structure, this approach can maximize vectorization and archive very high throughput on non-conventional processors such as GPU or other many-core processors [12,13,21]. In addition, combining the direct addressing array blocks with the multi-block technology, one can significantly reduce memory consumption in practical applications [22–24].

In order to avoid branch divergence in the direct-addressing method caused by complex geometry of porous media, some researchers suggested an explicit masking approach [12,21,25] instead of traditional if-else based branching approach to meet the requirement of vectorization for SIMD (Single Instruction, Multiple Data) computing. In this approach, both the solid and fluid nodes are computed, but only the fluid nodes are updated while the solid nodes are explicitly masked. However, this approach is proposed in early studies on SIMD architectures [25,26] for dense cubic geometries [12], which may not be suitable for modern GPU SIMT (Single Instruction, Multiple Threads) architecture and porous media. Additionally, structural effect on the performance of direct addressing method is not well studied, especially on recent GPUs.

Apart from sparse geometry, complex boundary condition is another key point in pore-scale multiphase flow simulations. While the LBM is particularly efficient to implement the non-slip boundary condition via the so-called bounce-back scheme, additional boundary treatments, i.e., the wetting boundary condition, are required in the multiphase LBM algorithm which may not be as efficient as the bounce-back scheme on GPUs [21,27–30]. The fictitious-density-based wetting model [31] simply specifies a const and fictitious fluid density on the solid nodes with no extra cost and desired contact angle forms naturally via direct fluid-solid interactions. However, recent studies have revealed that geometry-based wetting models perform better than the fictitious-density model, especially at large viscosity ratio [21,30,32]. Nevertheless, the geometry-based wetting model requires extrapolation on the boundary [21,33], resulting in low efficiency compared to the fictitious-density wetting model. For example, on a typical rock sample, boundary nodes that account for 8% of the total nodes may consume more than 25% of the total computing time.

In this work, we present a novel implementation of direct addressing-based multiphase LBM algorithm for flow in porous media. Combining AA pattern [16] and if-else branching based method, the present method outperforms other direct addressing methods for porous media without losing the simplicity and intuitiveness of the direct addressing methods. A detailed benchmark on both artificial and sandstone-like porous media is conducted. Our result not only shows if-else branching can be used as an optimization technique but also provides new insight on the performance sweet point between direct addressing and indirect addressing methods. As for complex boundary conditions, we analyze the kernel performance on recent NVIDIA GPUs via Nsight Compute. Improvement on L2 cache greatly improves memory access of complex boundary condition.

To tackle multiphase flow simulations in porous media, the hybrid-strategy is presented, where the direct-addressing method is employed for the main LB evolution while the indirect-addressing method is employed for the complex boundary conditions. The CSF-based LB color-gradient multiphase model and geometry-based wetting model are employed to increase simulation accuracy and stability (Section 2). Implementation details of the GPU code are given in Section 3. A detailed benchmark on both artificial and sandstone-like porous media is conducted (Section 4.2 and Sec-

tion 4.3). We then employ the multiphase flow solver to study the inertial effects in relative permeability measurement of a real sandstone (Section 4.4). A discussion and our conclusions are presented in Section 5.

2. Multiphase LBM Model

2.1. CSF based Color Gradient Model

In this work, we employ the continuum-surface-force (CSF) scheme [34–36] based color gradient (CG) LB multiphase model [21,27]. The CG-CSF model is popular for immiscible multiphase flow in porous media due to its ability to independently adjust surface tension, viscosity, and interfacial thickness [21,28,37]. In this model, two immiscible fluids are represented by two sets of colored distribution functions f_i^b and f_i^r , where superscripts denote different fluids, b for blue and r for red. The subscript i relates to discretized velocity \mathbf{e}_i in D3Q19 lattice model [38,39]. The bulk fluid distribution function is defined by $f_i = f_i^b + f_i^r$. Transportation of both fluids is described by the following lattice Boltzmann equation:

$$f_i^k(\mathbf{x} + \mathbf{e}_i \delta t, t + \delta t) = f_i^k(\mathbf{x}, t) + \Omega_i^{2,k} [\Omega_i^1 + \bar{F}_i], \quad k = r, b, \quad (1)$$

where superscript k denotes either fluid b or fluid r and Ω_i^1 , $\Omega_i^{2,k}$ are collision operators for viscous effect, and fluid separation, respectively. \bar{F}_i is the source term that relates to body force. Corresponding macroscopic variables can be obtained via [40],

$$\rho^k = \sum_i f_i^k \rho^k \mathbf{u}^k = \sum_i f_i^k \mathbf{e}_i + \frac{\rho^k F}{2} \quad (2)$$

$$\rho = \sum_k \rho^k, \quad \rho \mathbf{u} = \sum_k \rho^k \mathbf{u}^k \quad (3)$$

As CSF model mainly considers fluids with identical density [41, 42], collision operator Ω_i^1 can be calculated collectively via the bulk distribution functions [27,43], i.e.,

$$\Omega_i^1 = -\frac{1}{\lambda} (f_i - f_i^{eq}), \quad (4)$$

where λ is relaxation time and f_i^{eq} is equilibrium distribution function. However, as Eq. (1) suggests, the recoloring operator $\Omega_i^{2,k}$ responsible for fluid separation is color dependent which can be found in [37,44].

In CSF model, surface tension effect is introduced via a body force that relates to interfacial curvature and local color gradient [36],

$$\mathbf{F} = \frac{1}{2} \sigma \kappa \mathbf{C} = \frac{1}{2} \sigma \kappa \nabla \phi, \quad (5)$$

where σ is surface tension, κ is interfacial curvature, \mathbf{C} is the color gradient which is calculated via isotropic gradient operator in [45], and $\phi = \frac{\rho_r - \rho_b}{\rho_r + \rho_b}$ is the order parameter, where $\phi = 1$ and $\phi = -1$ indicates pure fluid r and pure fluid b region, respectively. With color gradient \mathbf{C} , interfacial curvature can be calculated via

$$\kappa = [(\mathbf{I} - \mathbf{nn}) \cdot \nabla] \cdot \mathbf{n}, \quad (6)$$

where \mathbf{n} denotes the normal direction of color gradient \mathbf{C} ,

$$\mathbf{n} = \frac{|\nabla \phi|}{\nabla \phi}. \quad (7)$$

In order to achieve better stability and accuracy, the LB multiple-relaxation-rate (MRT) framework [46,47] is employed. LB equation with the MRT collision operator can be expressed as

$$f_i^k(\mathbf{x} + \mathbf{e}_i \delta t, t + \delta t) = f_i^k(\mathbf{x}, t) + \Omega_i^{2,k} \left[-\mathbf{M}^{-1} \mathbf{S} (\mathbf{M} \mathbf{f}(\mathbf{x}, t) - \mathbf{m}^{eq}) + \delta t \mathbf{M}^{-1} \left(\mathbf{I} - \frac{\mathbf{S}}{2} \right) \hat{\mathbf{F}} \right], \quad (8)$$

where \mathbf{f} is the aforementioned bulk distribution functions in vector form, and \mathbf{m}^{eq} stands for the equilibrium moment. Additionally, \mathbf{M} is the transformation matrix to moment space and $\hat{\mathbf{F}} = \mathbf{M} \mathbf{f}$ accounts for the forcing term in moment space. With the D3Q19 lattice, the equilibrium moments \mathbf{m}^{eq} and forcing moments $\hat{\mathbf{F}}$ are given in [38,48]. And \mathbf{S} is the diagonal matrix composed of relaxation parameters which are chosen following [49].

2.2. Wetting Model

We employ the geometry-based wetting model proposed in [30], which directly solves the interfacial direction via

$$\mathbf{n}_{\pm} = \left(\cos \pm \theta - \frac{\sin \pm \theta \cos \theta'}{\sin \theta'} \right) \mathbf{n}_s + \frac{\sin \pm \theta}{\sin \theta'} \mathbf{n}^*, \quad (9)$$

$$\theta' = \arccos(\mathbf{n}_s \cdot \mathbf{n}^*), \quad (10)$$

where θ denotes contact angle, \mathbf{n}_s is the unit normal vector of boundary and \mathbf{n}^* is the estimated interfacial direction via Eq. (7). Then \mathbf{n}^* is altered to one of \mathbf{n}_+ and \mathbf{n}_- which has a shorter Euclidean distance to \mathbf{n}^* , with color gradient \mathbf{C} modified correspondingly. For complex geometry, \mathbf{n}_s can be calculated following [27]. As a result, the wetting model is introduced via the interfacial force according to Eq. (5) and Eq. (7).

Apart from modification of the interface normal vectors, in order to minimize spurious current at three-phase contact line region, two extrapolation steps are required in geometry-based wetting model, i.e. the extrapolation of ϕ and the extrapolation of the interface normal vectors \mathbf{n} in Eq. (5) [21,33]. We adopt the weighted-average method to perform the extrapolation steps [33]. Assuming that ξ is the parameter to be extrapolated, its value is known at all fluid nodes while its value at solid boundary nodes is required for subsequent calculations. The value at solid boundary node is evaluated via a weighted average of all surrounding fluid nodes, which is given by

$$\xi_s(\mathbf{x}) = \frac{\sum_{\alpha_l} \omega_{\alpha_l} \xi(\mathbf{x} + \mathbf{e}_{\alpha_l} \delta t)}{\sum_{\alpha_l} \omega_{\alpha_l}}, \quad (11)$$

where the subscript α_l denotes the direction of lattice velocity where $\mathbf{x} + \mathbf{e}_{\alpha_l} \delta t$ is a fluid node.

Although geometry-based wetting model successfully avoids the unphysical film along the solid boundaries and improves accuracy [21,27,30,33], the complexity of this model raises concerns about its computational efficiency, especially on modern heterogeneous architectures, such as GPUs. Compared with the fictitious-density wetting model that simply assigns fictitious densities at the solid boundary nodes, the geometry-based wettability model is more complex and computationally intensive. For a typical pore-structure of real sandstones, over 20% performance loss could be introduced from the geometry-based wetting model [21]. Poor performance may limit the application of the geometry-based wetting model in pore-scale simulations, which have a larger surface-area-to-volume-ratio, i.e., more boundaries nodes within the same volume compared to typical external flow problems. Therefore, it is of great importance to study the computational performance of the wetting boundary scheme in depth.

3. Implementation

3.1. Branch-based approach and Mask-based approach

Although the architecture of modern GPUs varies by vendor and generation, all GPUs can be classified as SIMD/SIMT processors [9,50]. For NVIDIA's GPUs, parallel computing is achieved via SIMT i.e., a warp of threads with size of 32 executes the same instruction spontaneously [9]. Divergent branches are executed in a serial manner within warps, resulting in performance penalties. Therefore, it is a consensus to remove conditional branches in the main evolution kernels of a LBM GPU solver [6,12,21,25,51]. One possible solution is to use an explicit mask to avoid conditional branches [12,21,25], which rewrites the branch-based approach into a mask-based approach as Algorithm 1 and 2 demonstrate,

Algorithm 1: Branch-based approach.

```

1: if (cell_type == FLUID)
2:   x=a;
3: else
4:   x=b;
5: endif

```

Algorithm 2: Mask-based approach.

```

1: is_fluid = (cell_type == FLUID);
2: x = a * is_fluid + b * (!is_fluid)

```

For LBM solvers based on the direct addressing memory access schemes, the most significant branch divergence occurs on fluid-solid mask. Generally, the collision step only occurs at fluid nodes while the streaming step can occur on both fluid and solid nodes depending on the memory access pattern and subsequent streaming scheme. While there are many streaming schemes such as pull, push [13], AA pattern [16], esoteric twist [52] and shift-swap streaming [26], most of the streaming schemes can be classified into two categories, the two-grid scheme and the one-grid schemes [16,26]. Streaming schemes that share the same memory pattern have the same memory load/store, yielding similar performance [53]. In the next section, we select AA pattern and pull scheme as typical one-grid scheme and two grid scheme, respectively. Based on the memory access pattern, a detailed insight on both the branch-based and the mask-based approaches is provided.

3.2. AA pattern and Pull Scheme LBM

Originally, the AA pattern was proposed to reduce memory usage, since the AA pattern scheme only takes about half the memory usage of the two-grid schemes. Performance improvement has also been reported in literatures [16,53]. In the AA pattern scheme, LBM timesteps are divided into odd and even step. The odd step consists of a pull-streaming step, a collision step followed by a push-streaming step to the same node but opposite location, while the even step only consists of a collision step. Through aforementioned process, distribution functions are swapped with their opposing direction at each step. Therefore, the full-way bounce-back nonslip scheme can be achieved automatically at solid boundary nodes [16]. In contrast, the full-way bounce-back scheme has to be carried out separately as a special collision step on the solid nodes in the pull scheme. The following comparison only considers the full-way bounce-back scheme which does not require the orientation of the solid surface [54], and the pull-type streaming scheme which outperforms the push-type streaming scheme [12,13]. Therefore, a general LBM kernel can be expressed as,

Algorithm 3: AA pattern, Branch-based.

```

1: Odd step:
2: for each node in domain do
3:   if (cell_type[node] == fluid)
4:     ftemp = LoadPullStream();
5:     ftemp = Collision(ftemp);
6:     PushStreamOppSave(ftemp);
7:   end if
8: end for
9: Even step:
10: for each node in domain do
11:   if (cell_type[node] == fluid)
12:     ftemp = LoadOpp();
13:     ftemp = Collision(ftemp);
14:     Save(ftemp);
15:   end if
16: end for

```

Algorithm 4: AA pattern, Mask-based.

```

1: Odd step:
2: for each node in domain do
3:   is_fluid = cell_type[node];
4:   ftemp = LoadPullStream();
5:   ftemp_new = Collision(ftemp);
6:   ftemp_new = ftemp_new * is_fluid + ftemp * (!isfluid);
7:   PushStreamOppSave(ftemp_new);
8: end for
9: Even step:
10: for each node in domain do
11:   is_fluid = cell_type[node];
12:   ftemp = LoadOpp();
13:   ftemp_new = Collision(ftemp);
14:   ftemp_new = ftemp_new * is_fluid + ftemp * (!isfluid);
15:   Save(ftemp);
16: end for

```

Algorithm 5: Pull scheme, Branch-based.

```

1: for each node in domain do
2:   ftemp = LoadPullStream();
3:   if (cell_type[node] == fluid)
4:     ftemp = Collision(ftemp);
5:   else
6:     ftemp = Oppose(ftemp);
7:   end if
8:   Save(ftemp);
9: end for

```

Algorithm 6: Pull scheme, Mask-based.

```

1: for each node in domain do
2:   ftemp = LoadPullStream();
3:   is_fluid = cell_type[node];
4:   ftemp_c = Collision(ftemp);
5:   ftemp_s = Oppose(ftemp);
6:   ftemp_new = ftemp_c * is_fluid + ftemp_s * (!isfluid);
7:   Save(ftemp_new);
8: end for

```

Consider a uniform computation grid on a porous medium with a porosity of ε . Assuming the total number of the grid nodes is N , we can estimate the total memory load/store for each scheme. For AA pattern and branch-based scheme, the total memory load/store is $2qN\varepsilon$, and $2qN$ for all other schemes, where q denotes number of distribution functions as in DmQn, since the fullway bounce-back can be achieved automatically in AA pattern and branch-based scheme. For multiphase flow, memory load/store is roughly 2 times that for single-phase accordingly. This analysis shows that AA pattern and branch-based scheme has smaller memory load/store compared to other schemes. Consequently, AA pattern

and branch-based scheme outperforms other direct addressing-based schemes since LBM is a typical memory bandwidth bound algorithm [6,16,53].

For simplicity, we use a one-dimensional block configuration [55]. Each block contains (128, 1, 1) threads and each thread is responsible for computation of one node. For a structure of (M_x, M_y, M_z) , total of $(\lceil M_x/128 \rceil, M_y, M_z)$ blocks are used, where $\lceil \cdot \rceil$ represents ceiling function.

3.3. Boundary Implementation

In this work, we employ a boundary-list approach [6,21] for the wetting boundary condition implementation. Two helper structs called **SolidBoundaryInfo** and **FluidBoundaryInfo** are defined for solid boundary nodes and fluid boundary nodes respectively in order to track position and necessary information of boundary nodes. For a solid boundary node, connectivity information to fluid nodes nearby is needed as shown in Eq. (11). Since the maximum number of fluid nodes nearby is determined by the chosen extrapolation scheme, we can reserve a fixed space to store indices of neighboring fluid nodes for each boundary node, which is 18 for D3Q19 lattice. We also store total number of neighbor fluid nodes and their lattice weight for convenience. For a fluid boundary node, we track the local normal vector and contact angle as Eq. (9) indicates. Then two list are constructed in column-major order which is also employed in the construction of the multidimensional array on the host before LBM computation. As **SolidBoundaryInfo** and **FluidBoundaryInfo** are *POD (Plain Old Data)* [56], corresponding boundary lists can be transferred from host to device easily by **cudaMemcpy** API.

Struct SolidBoundaryInfo

```

int x, y, z; //position info
int neighbor_num; //number of neighbor fluids
int neighbor_list [18]; //neighbor fluid list
double lattice_weight; //sum of lattice weight of neighbor fluid nodes

```

Struct FluidBoundaryInfo

```

int x, y, z; //position info
double theta; //local contact angle
double nx,ny,nz; //local normal direction vector

```

The boundary algorithm and related color gradient and curvature computation are divided into five GPU kernels as Fig. 1 illustrates. The separation of kernels mainly depends on the non-local operation in algorithms such as gradient and extrapolation which allows to minimize the memory load/store on GDRAM while maximize the temporal locality [9].

4. Results and Discussion

4.1. Testbed and Profiling Tool

We adopt NVIDIA Tesla V100 and A100 GPUs as our testbed. Based on the latest Volta (2017) and Ampere (2020) architectures, Tesla V100 and A100 are top-of-the-line GPUs designed for data centers in tasks such as scientific computing, AI, data analytics [57,58]. Both of our GPUs are PCIe version and operated with ECC off. Detailed specification of Tesla V100 and A100 is concluded in Table 1.

Nsight System and Nsight Compute is utilized to profile the entire application. Nsight System [59] and Nsight Compute [60] are official profiling and analytical tools provided by NVIDIA in order to profile and optimize GPU applications. Nsight System provides

Table 1
Specification of Tesla V100 [57] and A100 [58].

GPU Model	Peak FP64	Memory Bandwidth	Memory Size	SM Number	L1 Cache + Share Memory per SM	Register per SM	L2 Cache per Device	Launch Year
Tesla V100	7.8 TFLOPs	900 GB/s	16 GB	80	128 KB	65536	6 MB	2017
A100	9.7 TFLOPs	1555 GB/s	40 GB	108	196 KB	65536	40MB	2020

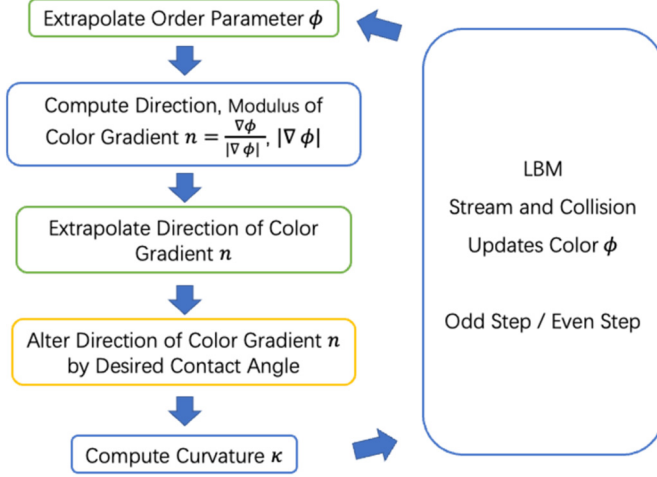


Fig. 1. Flow chart of the wetting boundary algorithm. The green and yellow box indicates the GPU kernels perform on all solid boundary nodes and fluid boundary nodes, respectively, whereas the blue one denotes the GPU kernel performs on all nodes in computational domain. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

a system level information on application performance [59]. We use Nsight System to measure overall kernel performance since it is fast and has low overhead. On the other hand, Nsight Compute provides a much more detailed metrics of kernels, such as compute and memory throughput (Speed of Light, SOL), occupancy, SM utilization, stall reasons, etc., though it takes more time profiling [60]. Therefore, Nsight Compute is used to study and analyze discrete kernels in depth.

4.2. Performance of the Streaming and Collision Kernels

In order to thoroughly evaluate our implementation, we first benchmark our multiphase LBM implementation on artificial structures. According to our block and thread configuration in Section 3.2, artificial porous media are composed of sticks that parallels to x -axis with length of 32 as Fig. 2 illustrates, which guarantees threads within a warp have identical node type. Therefore, there are no branch divergences or uncoalesced memory accesses [9]. The position of sticks on yz -plane is randomly distributed. For the purpose of excluding the influence of other kernels, we only present the performance of the LBM streaming and collision kernels in this section.

The performance of each direct addressing method on artificial porous media with different porosity is demonstrated in Fig. 3. The size of each porous medium is 256^3 . The performance of LBM implementation can be measured by MLUPS (Mega Lattice Update Per Second) and MFLUPS (Mega Fluid Lattice Update Per Second) which represents lattice update speed concerning all nodes and fluid nodes, respectively. As predicted in Section 3.2, distribution functions on all nodes, regardless of node types, are loaded and stored during the streaming-collision process in direct-addressing methods apart from AA pattern with branch approach method. For this reason, the speed in MLUPS is bounded by memory bandwidth independent of porosity. The performance in MFLUPS is proportional to porosity for these three methods. In contrast, AA pattern

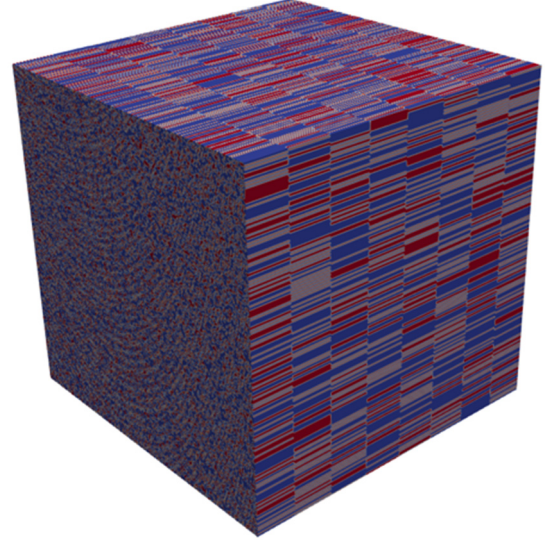


Fig. 2. Illustration of an artificial porous medium constructed for our test. Blue and red color represents solid nodes and fluid nodes, respectively. The size of the porous medium in the figure is 256^3 with a porosity of 40%.

with branch approach method only updates fluid nodes resulting in a performance increase compared with other methods. About 80% percent peak performance can be achieved when porosity $\phi = 30\%$ on V100 and $\phi = 50\%$ on A100.

However, the performance of AA pattern with branch approach method decreases as porosity decreases compared with peak performance achieved when porosity $\phi = 100\%$. This performance decrease can be attributed to a variety of reasons. Although branch divergences within warps are completely removed in artificial geometry, the number of fluid warps is different in different blocks, which causes load imbalance. On the other hand, the number of fluid nodes decreases as the porosity decreases. As a result, data size is insufficient to fully utilize GPU memory bandwidth at low porosity. This also explains why the proposed method outperforms on V100 than A100 at low porosity in Fig. 3 since V100 has a lower latency than A100 at small data size [61].

To further investigate the effect of data size, we evaluate the proposed method on a series of artificial porous media vary in porosity and domain size. We choose structures with porosity ranging from 15% to 45% to represent transitional region between memory bandwidth bound and latency bound. The benchmark result is presented in Fig. 4. As the maximum domain size is limited by the GPU memory in the benchmark, the problem size we consider is a good representation of simulation tasks on modern GPUs. On A100, performance increases about 25% at 384^3 compared with 128^3 , which indicates that performance is limited by data size at low porosity and small structure size. However, load imbalance has a significant influence when porosity $\phi \leq 25\%$, since achieved performance is much smaller than peak performance. In comparison, approximately 80% of the peak performance can be realized when porosity $\phi \geq 35\%$ on V100. This is because the memory bandwidth of V100 is only 57% of that of A100, which is more likely to induce memory bandwidth bound. The 25% of performance increase can be observed when porosity $\phi = 15\%$, which indicates that perfor-

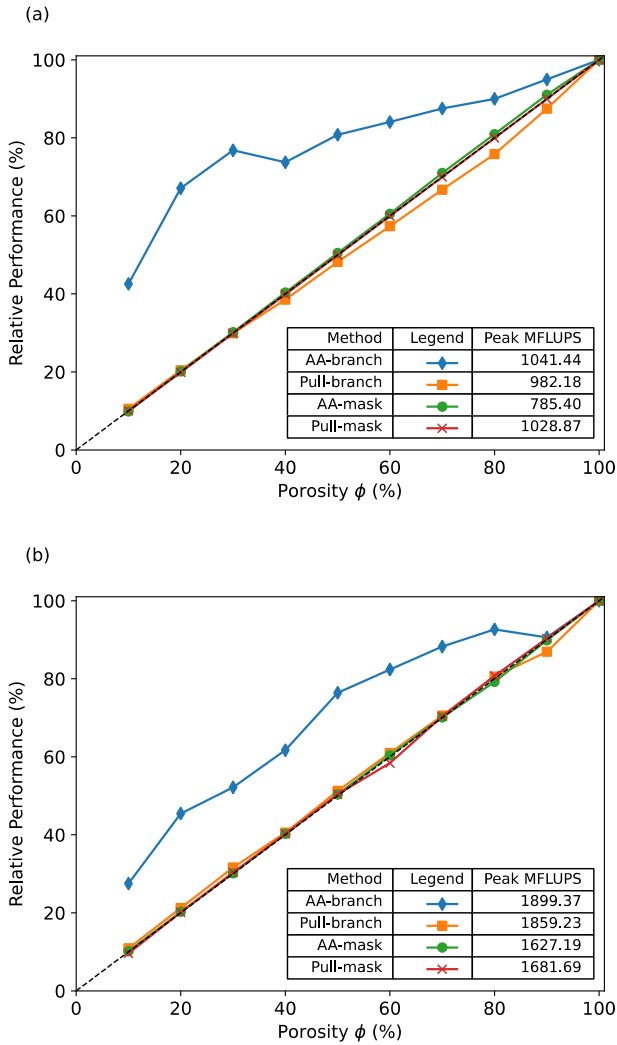


Fig. 3. Relative performance in MFLUPS of each direct addressing method on an artificial porous medium as a function of porosity ϕ : Tesla V100 (a) and A100 (b). Relative performance denotes performance relative to peak performance of each method at porosity $\phi = 100\%$.

performance is limited by data size as well on V100. In summary, structure size is the limiting factor on performance before hitting memory bandwidth bound for structure with relatively large porosity. The present method reaches approximately 80% of the peak performances for sufficiently large structure. On the other hand, load imbalance has a significant impact when porosity $\phi \leq 25\%$ which limits the further improvement on performance.

Nevertheless, the artificial structures are not representative because branch divergences and uncoalesced memory accesses are ignored. The following benchmarks are conducted on sandstone-like porous structures with different grain size and porosity generated by QSGS (Quartet Structure Generation Set) method [62]. QSGS employs a random growth process controlled by a set of parameters, which allows to generate morphological features resembling many natural porous media [62–64]. We conduct 4 grain size ranging from 4 voxels to 24 voxels and 5 porosities ranging from 15% to 35% similar to the properties of real sandstone. Ten structures of size 256^3 are generated for each grain size and porosity to minimize errors induced by the randomness of structures. A typical structure generated by QSGS and grain size of generated structures are illustrated in Fig. 5 and Fig. 6, respectively. Although QSGS cannot precisely control the grain size due to the random growth process, our results show that, except for small porosity

and large grain size, the grain size of the resulting structure is close to the specified value with sufficiently small deviation.

As Fig. 7 demonstrates, apart from AA-branch method, performance of other direct addressing methods decreases linearly with porosity which is the same as artificial structures, while the AA-branch method outperforms other methods. The present method achieves average 60% and 40% of the peak performance while approximately 80% and 60% for the best scenario on V100 and A100, respectively, for the porosity and grain size considered. This relative performance is comparable to many indirect addressing methods for single-phase LBM reported in literatures [14,15,19]. Although indirect addressing methods still have some advantage in performance and memory consumption, the present method inherits the simplicity and intuitiveness of the direct addressing methods, which takes less effort to implement.

Our results show that the computational performance increases as grain size increases. This is because on Nvidia GPU, memory access is conducted via 32-, 64-, 128-byte memory transactions. Memory access instructions of the threads in a warp are coalesced depending on the data size requested and distribution of memory address [9]. For porous media with large grain size, fluid nodes are more likely to be continuous, resulting in coalesced memory access. On contrary, fluid nodes in porous media with small grain size are less likely to be continuous, which induces uncoalesced memory access and low effective memory throughput. This also explains why this phenomenon is more significant on V100 than on A100. For porous media with small grain size, many discrete, uncoalesced memory transactions fulfill GPU's memory bandwidth, resulting in memory bandwidth bound as Group 1 in Fig. 7 (a) demonstrates. The improvement in porosity, i.e., data size to process doesn't contribute to performance, which is typically memory bandwidth bound. In comparison, memory bandwidth bound is unlikely to appear on A100 due to larger memory bandwidth.

To further understand influence of grain size on memory access, we performed a detailed analysis of 4 natural structures with the same porosity but different grain size generated by QSGS via Nsight Compute. In Nsight Compute, memory transactions are measured in sectors which represents 32 bytes since minimum transaction on GPU is 32 bytes [60]. A discrete request smaller than 32 bytes, for example a 4-byte float will be executed in a 32-byte sector, resulting in 28 bytes waste, also known as excessive

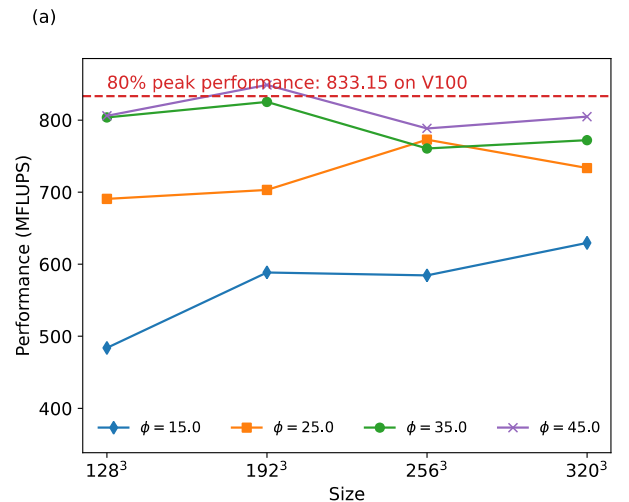


Fig. 4. (a, b) Performance in MFLUPS of AA-branch approach as function of domain size and porosity; (c, d) Performance in MFLUPS of AA-branch approach as function of number of fluid nodes and porosity. Tesla V100 (a, c) and A100 (b, d). Data in (c, d) is reorganized from (a, b) to provide a better display of the actually problem size. The red dotted line denotes 80% of the peak performance on corresponding device. The maximum domain size is limited by the GPU memory.

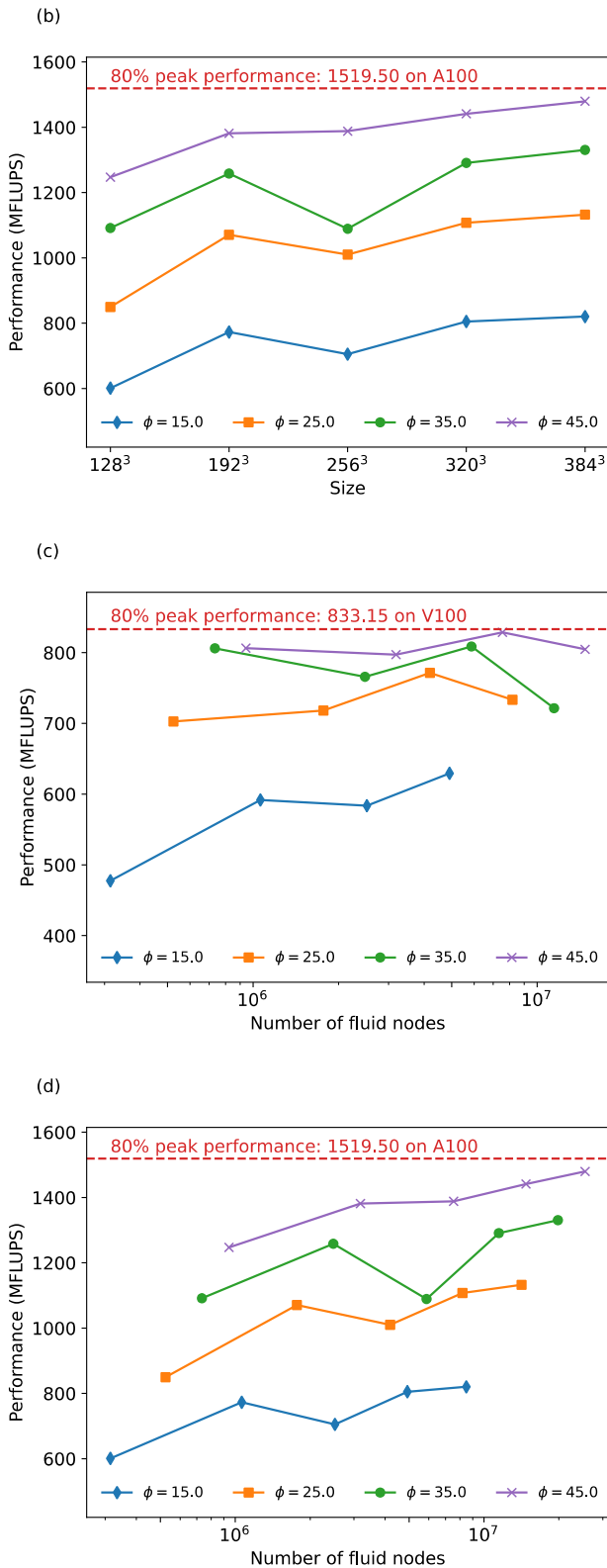


Fig. 4. (continued)

sectors. Total and excessive sectors of memory load and store are provided by source counter in Nsight Compute [60]. One sample with 30% porosity in each group is selected and analyzed as well as a corresponding artificial structure, as shown in Fig. 8. Identical results can be obtained on A100 and V100. Our results show that the 4 natural porous media samples all have similar effective

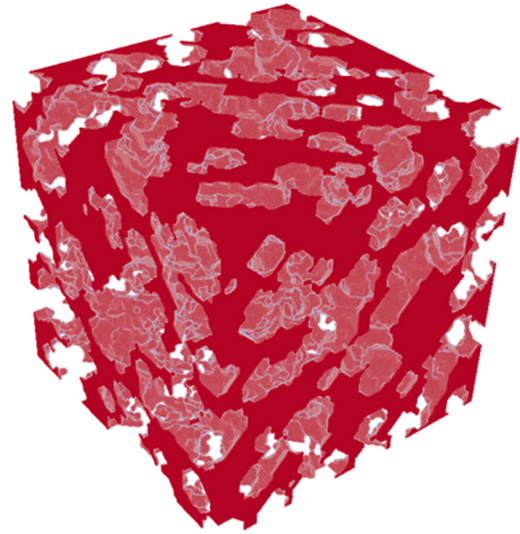


Fig. 5. Illustration of sandstone-like porous medium generated by QSGS. Red represents pores and grains are transparent. The size of the porous medium in the figure is 256^3 and porosity is 25%.

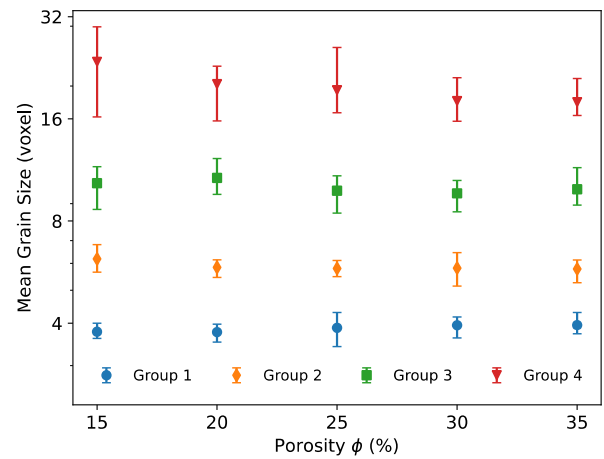


Fig. 6. Mean grain size of sandstone-like porous media generated by QSGS as function of porosity. Each group corresponds to a desired mean grain size.

load/store, which is slightly larger than artificial structure. In comparison, the excessive load/store increases as the grain size become smaller, which indicates worse situation in memory access.

4.3. Performance of the Complex Boundary Condition

The study on complex boundary condition is conducted on a natural porous structure generated via QSGS. The porosity of the structure is 30% and mean grain size is 10.9 voxels. The surface to volume ratio of this structure is 0.075, which indicates that boundary nodes account for 7.5% of the total voxels. We profile the entire application on both V100 and A100, as shown in Fig. 9.

Our results demonstrate that wetting boundary accounts for approximately 25% of the total computation time on V100 while this value decreases to 12% on A100. The computation time of V100 is 1.45 times that of A100, which is consistent with hardware specification since A100 improves memory bandwidth and computation power by 1.73 and 1.35 as to V100 [57,58], respectively. However, the computational performance of the wetting boundary condition improves 3.3 times on A100 compared to V100, which cannot be explained simply by the improvement of DRAM bandwidth and floating-point computing power. Here, we use Nsight Compute [60] to conduct a kernel-level study to investigate the performance of

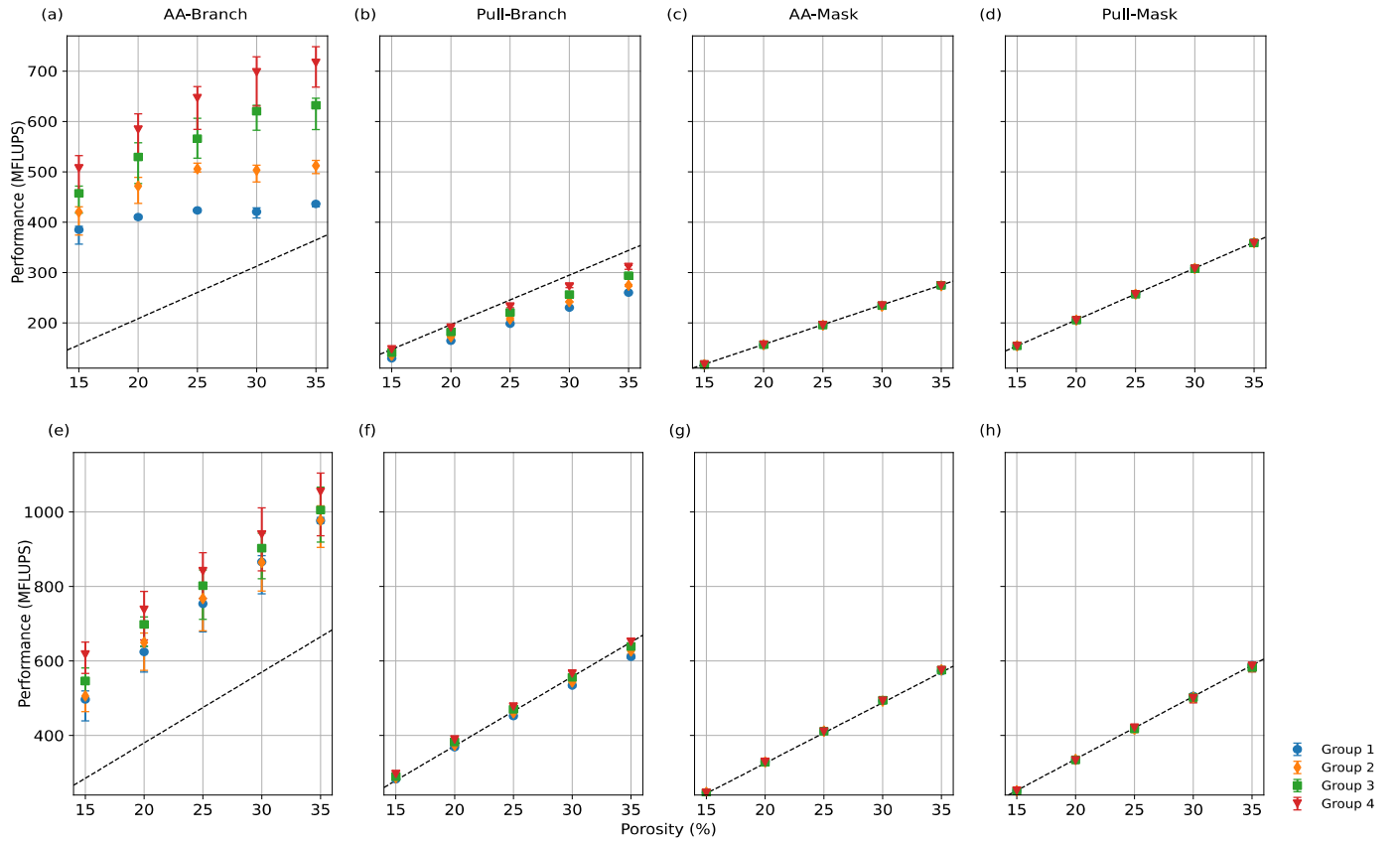


Fig. 7. Performance in MFLUPS of direct addressing methods as function of porosity and grain size, from left to right AA-branch, pull-branch, AA-mask, pull-mask. (a) - (d) Tesla V100, (e) - (h) A100. Each group corresponds to a mean grain size. Group 1 has the smallest grain size while group 4 has the largest grain size. Detailed information of each group can be found in Fig. 6. The dotted line denotes the predicted performance of each method according to the peak performance via the linear relationship as the function of porosity. The peak performances of each method are displayed in Fig. 3.

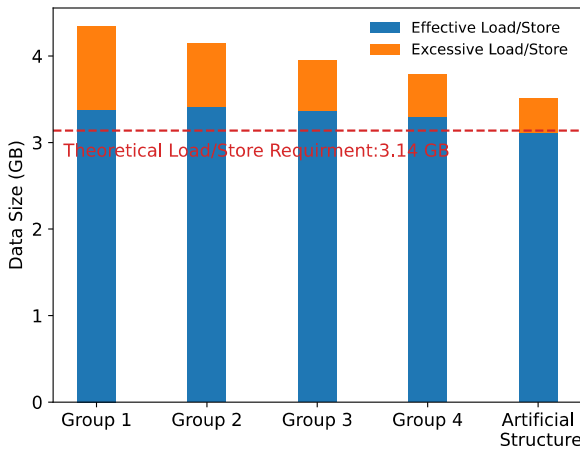


Fig. 8. Effective load/store and excessive load/store data size measured via Nsight Compute [60] of porous media with different mean grain sizes. Each group corresponds to a mean grain size. Group 1 has the smallest grain size while group 4 has the largest grain size. Detailed information of each group can be found in Fig. 6.

the boundary algorithms. We focus on the step of extrapolating normal vector \mathbf{n} as this kernel accounts for more than 50% of the wetting boundary computation. The results can be found in Fig. 10.

According to Fig. 10(a), **Stall Long Scoreboard** and **Stall LG Throttle**, which represent warps stalled waiting for L1 load/store operations and L1 load/store operations queue [60], are much higher on V100 than that A100. Our results indicate that warps spend approximately 5 times being stalled on V100 as to A100 due to L1 read and write requests. This is because the boundary list

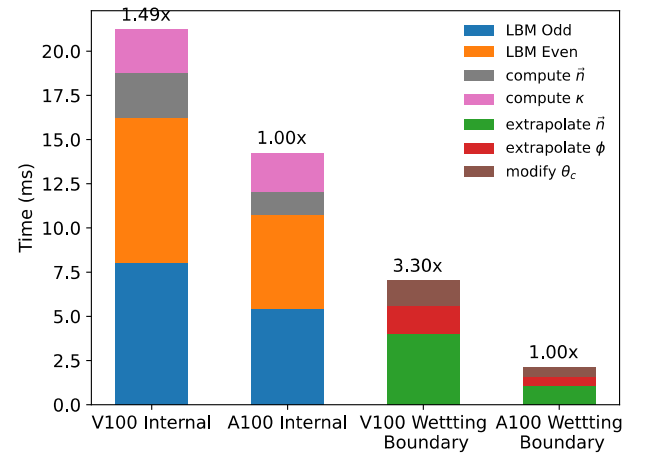


Fig. 9. Profile results of each kernel in CG-CSF LBM with geometry-based wetting model on V100 and A100.

approach introduces indirect addressing which induces high memory IO burden. In direct addressing, sequential memory accessed pattern can be achieved since memory access is regular, while indirect addressing forms a more random and scattered memory IO pattern. As a result, detailed and complicated tiling is required to achieve good performance for indirect addressing [15,19]. However, on more recent GPU A100, the larger L2 cache effectively reduces memory IO burden according to Fig. 10. Without any adjustment on memory order, a larger L2 Cache hit rate and throughput reduces warps stall to 1/5 of that on V100 and improve more than 3 times the overall performance of boundary kernels. Our profiling

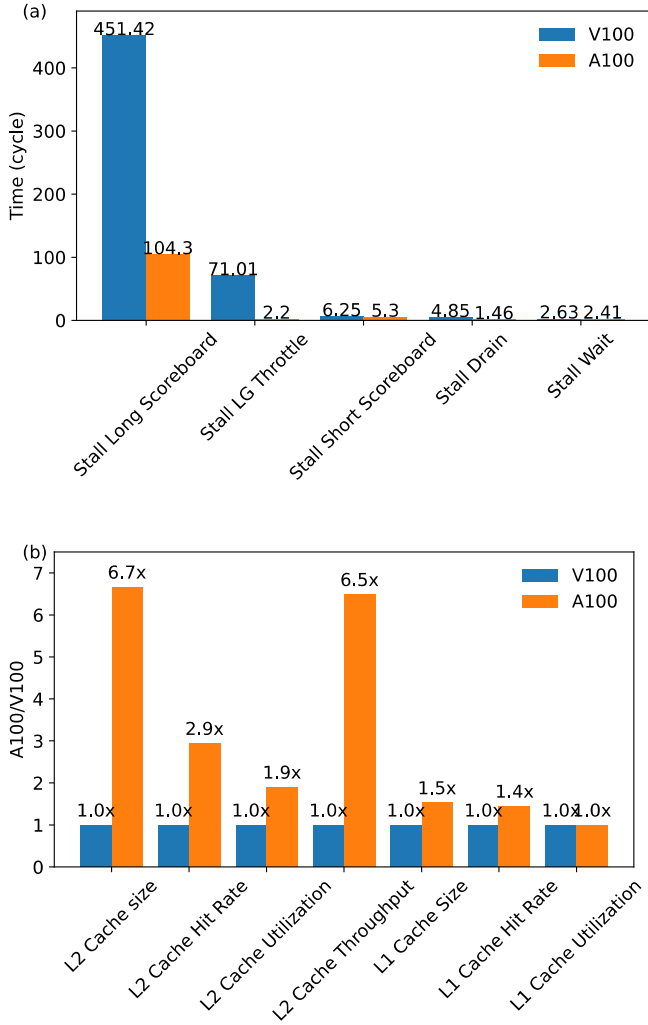


Fig. 10. (a) Top 5 stall reasons on GPU, lower the better; (b) Cache usage on GPU (data normalized by V100), higher the better. Explanation of each stall reason can be found in [60].

results demonstrate that geometry-based wetting model for multiphase flow in porous media can be implemented without the requirement of complicated tiling techniques while achieving decent performance. The present implementation can be benefited from recent advances in GPU architecture, especially larger L2 cache.

4.4. Overall Performance

Based on the discussion above, the overall performance of the current implementation is demonstrated in Fig. 11. The performance of other direct-addressing schemes is similar, therefore, AA mask is selected as a comparison. To make our discussion manageable, we select sandstone-like porous structures of 6 sets of different porosity and mean grain size with a wide range to evaluate the performance. There are 5 different porous structures in each set. The results presented are the average value of each set, in order to eliminate random errors caused by the generated structures. Additionally, since the previous results in section 4.2 suggest that structures of 256^3 may not fully utilize the GPU bandwidth, therefore, in this section the domain size of each structure is 320^3 and 448^3 on V100 and A100, respectively. As Fig. 11 shows, the present implementation achieves higher performance with larger porosity and grain size on both V100 and A100. On the one hand, the grain size affects memory access pattern as the aforementioned discussion. On the other hand, the larger grain size suggests

fewer boundary nodes which also improves the performance. The speedup over 1.5 on both V100 and A100. Since the bandwidth of the V100 is smaller, the bandwidth bottleneck is more severe. Our implementation can effectively improve the utilization of bandwidth and thus achieve a greater speedup compared with A100. However, thanks to the larger memory bandwidth and cache size on A100, the performance is roughly two times of that on V100.

We also perform a relative permeability calculation as a real-world application to demonstrate the capability of the present implementation. The inertia effect in drainage displacement has been studied via pore-scale direct numerical simulations in [21,49]. However, such studies haven't been conducted for the relative permeability. In experiments, the alteration of inertia effect is mainly achieved by increasing velocity [65–67]. Nevertheless, the viscous force is altered as well. In other words, the Reynold's number Re and capillary number Ca cannot be changed independently, which makes it difficult to distinguish the inertia effect and capillary effect. Therefore, the inertia effect on the relative permeability is a perfect example to demonstrate the capability of our multiphase flow solver.

Our rock sample is selected from Daqing oil-field in China, which is a typical sandstone. The digital rock structure is obtained by first scanning the rock sample then reconstructed via a typical image processing procedure [7]. For simplicity, the structure size is scaled to 400^3 which allows to conduct the simulation on a single A100 GPU. The porosity of the sample is 24.5% and the permeability is 2557 mD. As Fig. 12 shows, the sample is mirrored to ensure periodic boundary condition at inlet and outlet and an external force F is applied as driven force, which is widely used in the calculation of relative permeability in order to avoid saturation gradient and capillary pressure gradient [8,68,69]. The capillary number $Ca = Qv/A\sigma \sim 1 \times 10^{-5}$ and Reynold's number $Re = \frac{Qd}{Av} \sim 1 \times 10^{-5}$ to 1×10^{-3} where Q is the flow rate and A , v , d denotes intersection area, kinetic viscosity and characteristic pore size, respectively. At given saturation S , the relative permeability can be obtained via

$$k_{r,w} = \frac{Q_w(S_w)F}{QF_w}, \quad k_{r,nw}(S_w) = \frac{Q_{nw}(S_w)F}{QF_w}, \quad (12)$$

where k_r represents relative permeability and w, nw represents wetting and non-wetting phase, respectively. The calculated relative permeability is shown in Fig. 12(b). The relative permeability decreases as the Re increases, which is consistent with the “negative inertia” effect observed in experiments [65–67] when interfacial tension is high. This could be explained by the Ohnesorge number $Oh = v/\sqrt{\sigma d} = \sqrt{Ca}/Re$. According to Oh , increase of the Re is effectively decreasing Ca , resulting capillary force dominant flows and smaller relative permeability [70,71]. However, the detailed flow mechanisms require further studies. On this particular sample, we achieve 487 MFLUPS and 1.4 speedup compared to AA-mask-based approach, considering 8.7% of this sample are boundary nodes and the average pore-size is merely 3.7 pixels.

5. Conclusions

In this work, we present a hybrid implementation of multiphase Lattice Boltzmann Method concerning both sparse geometry and complex boundary condition in porous media. Structural effect on performance is considered for direct addressing methods. Compared with other direct addressing method, AA pattern and if-else branch method proposed in this work reduces memory load/store for porous media resulting in higher performance. Over 80% of the peak performance achieved at complete dense geometry can be observed for artificial porous media with porosity over 50% and sufficient large data size on both Tesla V100 and A100.

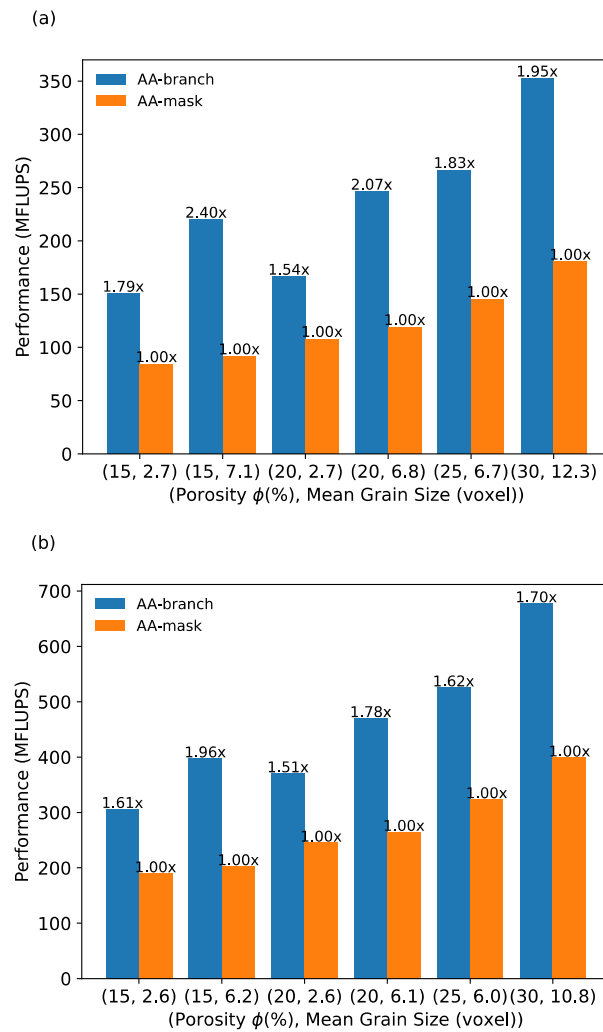


Fig. 11. Overall performance comparison between AA-branch approach and AA-mask approach on V100 (a) and A100(b) with different porous structures.

As for structure with low porosity, performance has a tendency to latency bound due to insufficient data size and load unbalance. Performance is also affected by grain size in sandstone structures which represents porous media in real world. An average of 60% and 40% of the peak performance is achieved on V100 and A100 respectively for sandstone structures we considered. Performance loss is mainly caused by sparse memory requests in porous media, resulting in uncoalesced memory transactions. We conducted performance analysis for complex boundary conditions as well. Stall reasons and cache utilizations are measured via Nsight Compute [60]. Our results demonstrate that larger L2 cache on A100 improves memory access for indirect addressing and extrapolation algorithms in boundary treatment compared to V100, resulting 3.3 to 8.4 times speed up depending on the specific boundary condition algorithms. As for the overall performance, our results show that the present implementation is capable of speedup over 1.5 compared with other direct-addressing schemes on V100 and A100, respectively, with only minor modification of the code.

Relative performance of the present implementation may be not as fast as that of indirect addressing methods, however, it is comparable with indirect addressing methods reported for single-phase LBM [14,15]. Due to the limited cache size and shared memory, for multiphase flow, the performance of the indirect-addressing methods may not be as well as single-phase flow, since more data are processed. Nevertheless, AA pattern and if-else branch are simple, intuitive and easy to implement compared with indirect addressing

methods. Considering the difference in performance and memory consumption between the direct and indirect addressing methods, the sweet point needs to be reconsidered. This work also emphasizes the significance of boundary treatment for complex flow in porous media, since boundary conditions in complex porous flow are more time consuming than ordinary structures and single-phase flow. With the development of computer architectures, many complicated boundary conditions can be implemented in an efficient way while reserving the advantages of physical accuracy.

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.

Data availability

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

Acknowledgements

The authors would like to thank Dr. Y. H. Huang from NVIDIA Corporation for providing useful discussion on Nsight Compute. This work is financially supported by the National Key R&D Program of China (No. 2019YFA0708704) and the NSF of China grant (No. 12272207).

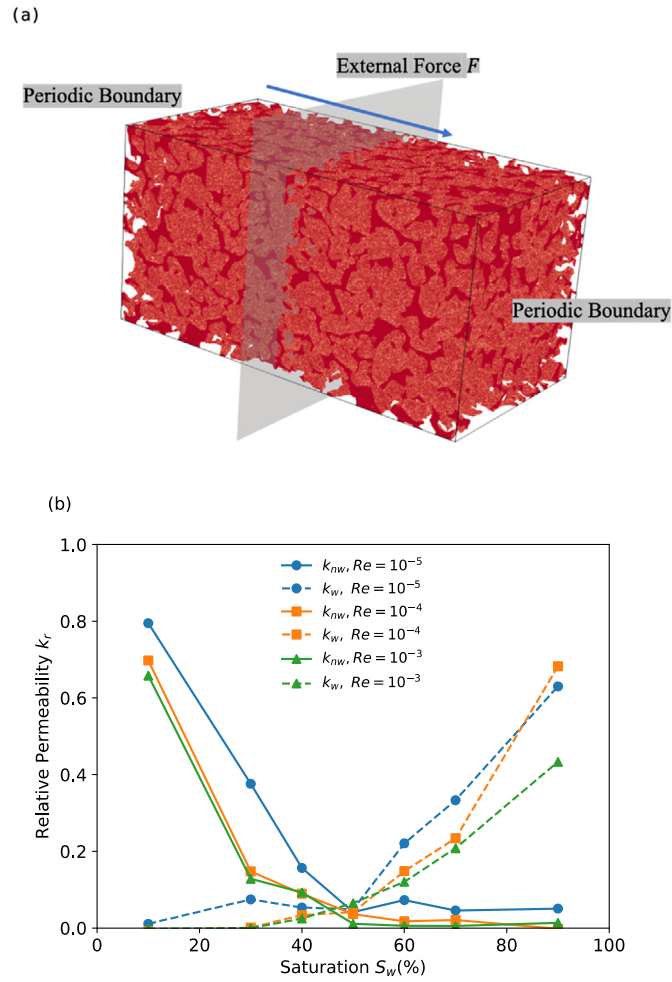


Fig. 12. (a) Simulation configuration for relative permeability computation. The sample is from Daqing oil field. (b) Relative permeability calculated by CG-CSF-LBM. The capillary number $Ca = 1 \times 10^{-3}$ for all simulations. The Re ranges from 1×10^{-5} to 1×10^{-3} .

Appendix A. Verification of L2 cache by destroying locality

Since modern caches exploit the locality of access [72], the contribution of L2 cache can be verified by using a cache-unfriendly pattern. In this work, we adopt random ordered boundary lists as a cache-unfriendly pattern compared to column-major order used in Section 3.3. Random ordered boundary lists can be generated by shuffling the aforementioned column-major order boundary lists. We adopt the `std::shuffle` provided in `C++ STL`. Boundary lists are

shuffled immediately after generation then copied to Device. Profiling results is demonstrated in Fig. A.1. The structure is the same as that used in Section 4.3.

Shuffled boundary lists perform worse 2.6 times and 2.9 times respectively than their baselines on both V100 and A100, which indicates a cache-unfriendly pattern decreases performance of indirect memory access. Our analysis in Section 4.3 on effect of L2 cache is confirmed. Again, stalling and cache utilization of extrapolation n of are presented in Fig. A.2, which supports our analysis as

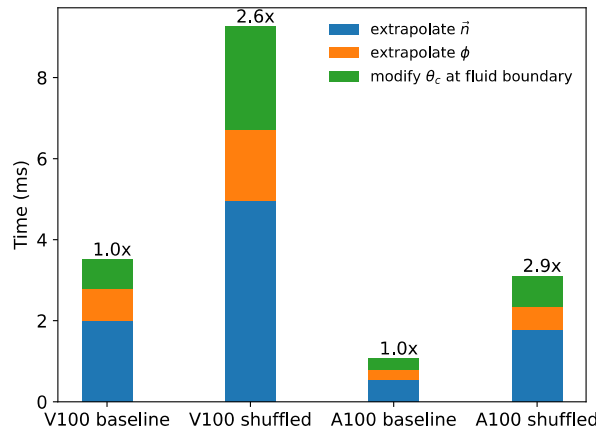


Fig. A.1. Profile results of each kernel in CG-CSF LBM with geometry-based wetting model on V100 and A100: shuffled boundary list vs baseline.

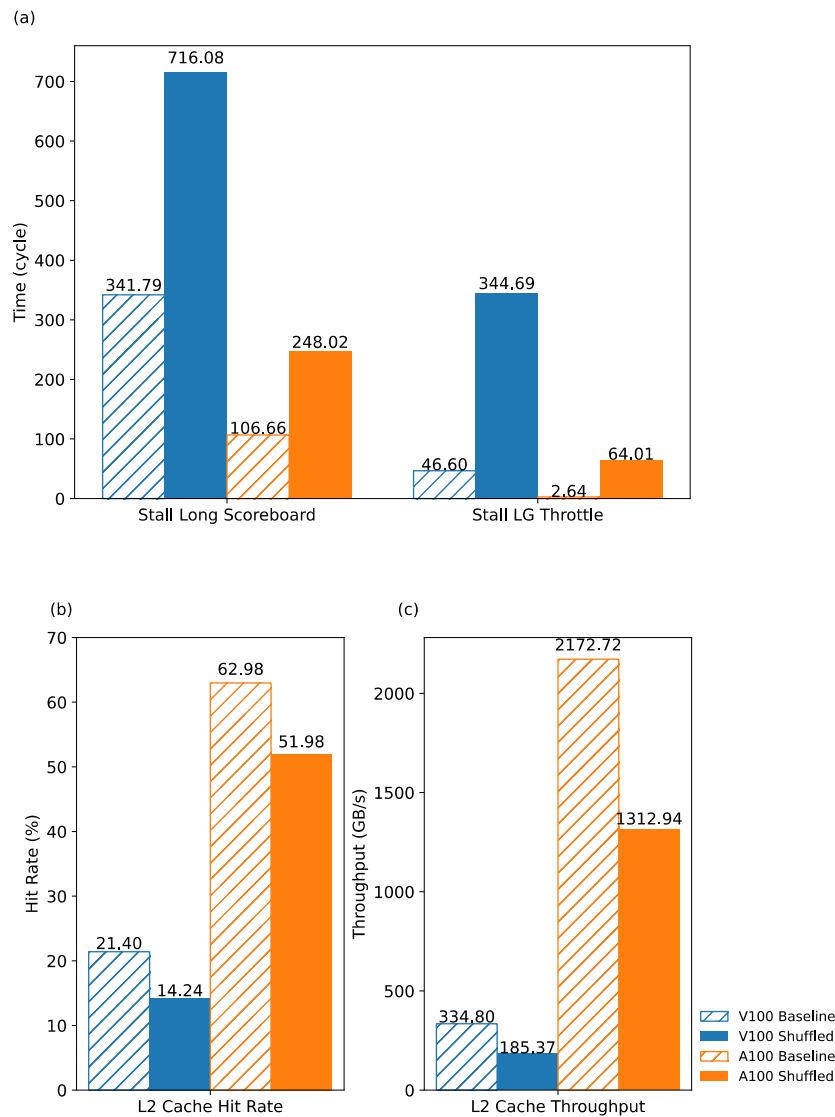


Fig. A.2. Comparison between shuffled boundary list and baseline on V100 and A100: main stall reasons (a) and L2 cache hit rate (b) and throughput (c).

well. Increase of memory load/store related stalling and decrease of L2 cache throughput and hit rate are illustrated.

References

- [1] S. Thomas, *Oil Gas Sci. Technol.* 63 (2007) 9–19.
- [2] L.K. Abidoeye, K.J. Khudaida, D.B. Das, *Crit. Rev. Environ. Sci. Technol.* 45 (2014) 1105–1147.
- [3] S. Chen, G.D. Doolen, *Annu. Rev. Fluid Mech.* 30 (1998) 329–364.
- [4] L. Chen, Q. Kang, Y. Mu, Y.-L. He, W.-Q. Tao, *Int. J. Heat Mass Transf.* 76 (2014) 210–236.
- [5] Q. Li, K.H. Luo, Q.J. Kang, Y.L. He, Q. Chen, Q. Liu, *Prog. Energy Combust. Sci.* 52 (2016) 62–105.
- [6] H. Liu, Q. Kang, C.R. Leonardi, S. Schmieschek, A. Narváez, B.D. Jones, J.R. Williams, A.J. Valocchi, J. Harting, *Comput. Geosci.* 20 (2015) 777–805.
- [7] T. Liu, X. Jin, M. Wang, *Energies* 11 (2018) 1798.
- [8] T. Liu, M. Wang, *Transp. Porous Media* 144 (2022) 111–132.
- [9] NVIDIA, *CUDA C Programming Guide*, 2022.
- [10] J. Tölke, M. Krafczyk, *Int. J. Comput. Fluid Dyn.* 22 (2008) 443–456.
- [11] W. Li, X. Wei, A. Kaufman, *Vis. Comput.* 19 (2003) 444–456.
- [12] N.-P. Tran, M. Lee, S. Hong, *Sci. Program.* 2017 (2017) 1–16.
- [13] M.J. Mawson, A.J. Revell, *Comput. Phys. Commun.* 185 (2014) 2566–2574.
- [14] G. Herschlag, S. Lee, J.S. Vetter, A. Randles, in: *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2018, pp. 825–834.
- [15] G. Herschlag, S. Lee, J.S. Vetter, A. Randles, *IEEE Trans. Parallel Distrib. Syst.* 32 (2021) 2400–2414.
- [16] P. Bailey, J. Myre, S.D. Walsh, D.J. Lilja, M.O. Saar, in: *2009 International Conference on Parallel Processing*, IEEE, 2009, pp. 550–557.
- [17] J. Myre, S.D.C. Walsh, D. Lilja, M.O. Saar, *Concurr. Comput., Pract. Exp.* 23 (2011) 332–350.
- [18] K. Mattila, T. Puurtinen, J. Hyväluoma, R. Surmas, M. Myllys, T. Turpeinen, F. Robertsén, J. Westerholm, J. Timonen, *J. Comput. Sci.* 12 (2016) 62–76.
- [19] T. Tomczak, R.G. Szafran, *Comput. Phys. Commun.* 235 (2019) 258–278.
- [20] C. Obrecht, F. Kuznik, B. Tourancheau, J.-J. Roux, *Comput. Math. Appl.* 61 (2011) 3628–3638.
- [21] Y. Chen, A.J. Valocchi, Q. Kang, H.S. Viswanathan, *Water Resour. Res.* 55 (2019) 11144–11165.
- [22] M.J. Krause, A. Kummerländer, S.J. Avis, H. Kusumaatmaja, D. Dapelo, F. Klemens, M. Gaedtker, N. Hafen, A. Mink, R. Trunk, J.E. Marquardt, M.-L. Maier, M. Haussmann, S. Simonis, *Comput. Math. Appl.* 81 (2021) 258–288.
- [23] J. Latt, O. Malaspinas, D. Kontaxakis, A. Parmigiani, D. Lagrava, F. Brogi, M.B. Belgacem, Y. Thorimbert, S. Leclaire, S. Li, F. Marson, J. Lemus, C. Kotsalos, R. Conradin, C. Coreixas, R. Petkantchinn, F. Raynaud, J. Beny, B. Chopard, *Comput. Math. Appl.* 81 (2021) 334–350.
- [24] M. Bauer, S. Eibl, C. Godenschwager, N. Kohl, M. Kuron, C. Rettinger, F. Schornbaum, C. Schwarzmeier, D. Thönnies, H. Köstler, U. Rüdiger, *Comput. Math. Appl.* 81 (2021) 478–501.
- [25] X. Ren, Y. Tang, G. Wang, T. Tang, X. Fang, in: *2010 International Conference on Data Storage and Data Engineering*, 2010, pp. 116–122.
- [26] M. Mohrhard, G. Thäter, J. Bludau, B. Horvat, M.J. Krause, *Comput. Fluids* 181 (2019) 1–7.
- [27] Z. Xu, H. Liu, A.J. Valocchi, *Water Resour. Res.* 53 (2017) 3770–3790.
- [28] S. Leclaire, A. Parmigiani, O. Malaspinas, B. Chopard, J. Latt, *Phys. Rev. E* 95 (2017) 033306.
- [29] J.E. McClure, J.F. Prins, C.T. Miller, *Comput. Phys. Commun.* 185 (2014) 1865–1874.

- [30] T. Akai, B. Bijeljic, M.J. Blunt, *Adv. Water Resour.* 116 (2018) 56–66.
- [31] M. Latva-Kokko, D.H. Rothman, *Phys. Rev. E, Stat. Nonlinear Soft Matter Phys.* 72 (2005) 046701.
- [32] H.J. Xu, Z.B. Xing, F.Q. Wang, Z.M. Cheng, *Chem. Eng. Sci.* 195 (2019) 462–483.
- [33] Y. Yu, H. Liu, Y. Zhang, D. Liang, *J. Mech. Eng. Sci.* 232 (2017) 416–430.
- [34] S.V. Lishchuk, C.M. Care, I. Halliday, *Phys. Rev. E, Stat. Nonlinear Soft Matter Phys.* 67 (2003) 036701.
- [35] I. Halliday, A.P. Hollis, C.M. Care, *Phys. Rev. E, Stat. Nonlinear Soft Matter Phys.* 76 (2007) 026708.
- [36] J.U. Brackbill, D.B. Kothe, C. Zemach, *J. Comput. Phys.* 100 (1992) 335–354.
- [37] M. Latva-Kokko, D.H. Rothman, *Phys. Rev. E, Stat. Nonlinear Soft Matter Phys.* 71 (2005) 056702.
- [38] T. Krüger, H. Kusumaatmaja, A. Kuzmin, O. Shardt, G. Silva, E.M. Viggien, *The Lattice Boltzmann Method*, Springer, Switzerland, 2017.
- [39] Y.-H. Qian, D. d’Humières, P. Lallemand, *Europhys. Lett.* 17 (1992) 479.
- [40] Z. Guo, C. Zheng, B. Shi, *Phys. Rev. E, Stat. Nonlinear Soft Matter Phys.* 65 (2002) 046308.
- [41] S. Leclaire, N. Pellerin, M. Reggio, J.-Y. Trépanier, *Int. J. Multiph. Flow* 57 (2013) 159–168.
- [42] H. Huang, J.-J. Huang, X.-Y. Lu, M.C. Sukop, *Int. J. Mod. Phys. C* 24 (2013).
- [43] H. Huang, J.-J. Huang, X.-Y. Lu, *Comput. Fluids* 93 (2014) 164–172.
- [44] S. Leclaire, M. Reggio, J.-Y. Trépanier, *Appl. Math. Model.* 36 (2012) 2237–2252.
- [45] S. Leclaire, M. Reggio, J.-Y. Trépanier, *Comput. Fluids* 48 (2011) 98–112.
- [46] D. D’Humières, I. Ginzburg, M. Krafczyk, P. Lallemand, L.S. Luo, *Philos. Trans. R. Soc. A, Math. Phys. Eng. Sci.* 360 (2002) 437–451.
- [47] P. Lallemand, L.S. Luo, *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Topics* 61 (2000) 6546.
- [48] Z. Guo, C. Shu, *Lattice Boltzmann Method and Its Application in Engineering*, World Scientific, 2013.
- [49] Y. Chen, Y. Li, A.J. Valocchi, K.T. Christensen, *J. Contam. Hydrol.* 212 (2018) 14–27.
- [50] AMD, *AMD CDNA™ 2 architecture whitepaper*, 2022.
- [51] T. Krüger, H. Kusumaatmaja, A. Kuzmin, O. Shardt, G. Silva, E.M. Viggien, in: *The Lattice Boltzmann Method: Principles and Practice*, Springer International Publishing, Cham, 2017, pp. 533–652.
- [52] M. Geier, M. Schönherr, *Computation* 5 (2017).
- [53] M. Wittmann, T. Zeiser, G. Hager, G. Wellein, *Comput. Math. Appl.* 65 (2013) 924–935.
- [54] T. Krüger, H. Kusumaatmaja, A. Kuzmin, O. Shardt, G. Silva, E.M. Viggien, in: *The Lattice Boltzmann Method: Principles and Practice*, Springer International Publishing, Cham, 2017, pp. 153–230.
- [55] C. Obrecht, F. Kuznik, B. Tourancheau, J.-J. Roux, *Parallel Comput.* 39 (2013) 259–270.
- [56] cpplibrary.com, C++ named requirements: PODType, 2022.
- [57] NVIDIA, *NVIDIA Tesla V100 GPU Architecture, The World’s Most Advanced Data Center GPU*, 2017.
- [58] NVIDIA, *NVIDIA A100 Tensor Core GPU Architecture, Unprecedented Acceleration at Every Scale*, 2020.
- [59] NVIDIA, *Nsight System User Guide*, 2022.
- [60] NVIDIA, *Nsight Compute User Manual*, 2022.
- [61] H. Anzt, Y.M. Tsai, A. Abdelfattah, T. Cojean, J. Dongarra, in: *2020 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)*, 2020, pp. 26–38.
- [62] M. Wang, J. Wang, N. Pan, S. Chen, *Phys. Rev. E, Stat. Nonlinear Soft Matter Phys.* 75 (2007) 036702.
- [63] L. Germanou, M.T. Ho, Y. Zhang, L. Wu, *J. Nat. Gas Sci. Eng.* 60 (2018) 271–283.
- [64] L. Chen, Q. Kang, Z. Dai, H.S. Viswanathan, W. Tao, *Fuel* 160 (2015) 346–356.
- [65] G.D. Henderson, A. Danesh, D.H. Tehrani, B. Al-Kharusi, in: *SPE Annual Technical Conference and Exhibition*, 2000.
- [66] R. Mott, A. Cable, M. Spearing, in: *SPE Annual Technical Conference and Exhibition*, 2000.
- [67] M. Jamiolahmady, M. Sohrabi, S. Ireland, *SPE J.* 15 (2009) 208–222.
- [68] F. Jiang, T. Tsuji, *Water Resour. Res.* 53 (2017) 11–32.
- [69] H. Li, C. Pan, C.T. Miller, *Phys. Rev. E, Stat. Nonlinear Soft Matter Phys.* 72 (2005) 026705.
- [70] C.D. Tsakiroglou, M.A. Theodoropoulou, V. Karoutsos, *AIChE J.* 49 (2003) 2472–2486.
- [71] R.T. Armstrong, J.E. McClure, M.A. Berrill, M. Rucker, S. Schluter, S. Berg, *Phys. Rev. E* 94 (2016) 043113.
- [72] D.A. Patterson, J.L. Hennessy, in: *Computer Organization & Design, The Hardware/Software Interface: RISC-V Edition*, Morgan Kaufmann, Cambridge, Massachusetts, 2018.