Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 267 (2025) 52–61

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the Third EuroHPC user day

Multi-GPU porting of a phase-change cascaded lattice Boltzmann method for three-dimensional pool boiling simulations

Alessandro Gabbana[a,b,*], Linlin Fei[c], Xander M. de Wit[d], Ziqi Wang[d], Daniel Livescu[a], Federico Toschi[d,e]

[a]*Computational Physics and Methods Group (CCS-2), Los Alamos National Laboratory, Los Alamos, 87545 New Mexico, USA*
[b]*Center for Nonlinear Studies (CNLS), Los Alamos National Laboratory, Los Alamos, 87545 New Mexico, USA*
[c]*Key Laboratory of Thermo-Fluid Science and Engineering of Ministry of Education, School of Energy and Power Engineering, Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China*
[d]*Department of Applied Physics and Science Education, Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands*
[e]*CNR-IAC, I-00185 Rome, Italy*

Abstract

The Lattice Boltzmann method (LBM) has proven effective in simulating phase-change phenomena, such as melting, solidification, evaporation, and boiling. In this work, we develop a highly parallelized multi-GPU implementation of LBM for three-dimensional pool boiling simulations. The code is based on the OpenACC programming model, which enables the code to be deployed efficiently on multi-core CPUs, GPUs, and potentially other accelerators, without the need for architecture-specific rewrites. To support large-scale simulations, the domain is decomposed and distributed across multiple compute nodes using MPI. We demonstrate that the code exhibits excellent scaling properties, with ideal strong-scaling running with up to 256 GPUs on the MareNostrum5 cluster.

© 2025 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY 4.0 license (https://creativecommons.org/licenses/by/4.0)
Peer-review under responsibility of the scientific committee of the Proceedings of the Third EuroHPC user day

Keywords: Pool Boiling; Lattice Boltzmann Method; OpenACC; Multi-GPU;

1. Introduction

Boiling is a highly efficient heat transfer mechanism with widespread applications, from everyday uses such as cooking to critical industrial systems such as nuclear reactors, heat exchangers, and electronic cooling devices. Despite its ubiquity, the boiling process remains a subject of intense research because of its inherently multiscale and multiphysics nature, which involves complex interactions between fluid dynamics, phase change, and heat transfer [1].

Among the different boiling configurations, pool boiling, where buoyancy is the sole driving force for fluid motion, offers a relatively simple setup to study the fundamental mechanisms of bubble nucleation, growth, departure, and

* Corresponding author.
E-mail address: agabbana@lanl.gov

1877-0509 © 2025 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY 4.0 license (https://creativecommons.org/licenses/by/4.0)
Peer-review under responsibility of the scientific committee of the Proceedings of the Third EuroHPC user day
10.1016/j.procs.2025.08.232

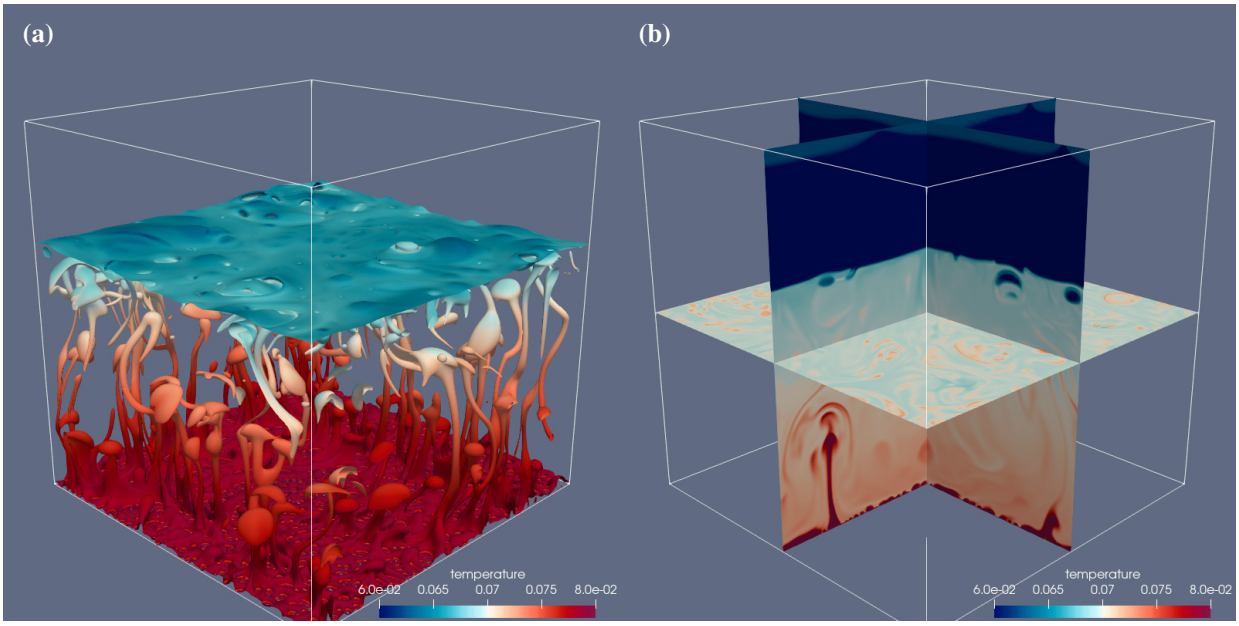

Fig. 1. Snapshot from a pool boiling simulation on a 1024^3 domain, in the nucleate boiling regime. (a) Iso-surface representation of the liquid-vapor interface, colored by the local temperature field. (b) Cross-sectional slices of the temperature field.

coalescence. Understanding these phenomena is essential to improve heat transfer efficiency and predict performance limits, such as the critical heat flux (CHF), in engineering systems. However, accurately capturing these dynamics requires high-fidelity numerical simulations that resolve both the microscale vapor-liquid interfaces and the macroscale thermal convection.

To address this challenge, numerical methods like the lattice Boltzmann method (LBM) have emerged as promising tools due to their algorithmic simplicity, scalability, and ability to naturally handle complex boundary conditions. In particular, the pseudopotential multiphase LBM, enhanced by thermal coupling via finite-difference solvers and non-ideal equations of state, has proven effective in simulating phase-change phenomena, including nucleate boiling and transitions to film boiling. The model developed by Fei et al. [2], which extends the hybrid LBM to fully three-dimensional domains using the cascaded collision operator for improved numerical stability, has successfully reproduced the entire boiling curve and boiling regimes with hundreds of interacting bubbles.

Despite these advances, most existing numerical studies on boiling remain limited to two-dimensional simulations or coarse three-dimensional models, which significantly restrict their applicability to real-world scenarios. Furthermore, the computational cost of resolving a fully 3D boiling process at realistic scales has, until recently, remained prohibitive [3].

In this work, we take a major step forward by developing a highly parallelized multi-GPU implementation of LBM for three-dimensional pool boiling simulations. By leveraging massively parallel GPU architectures, our code enables simulations involving millions of grid points, offering the possibility of simulations with unprecedented fidelity and throughput, making it possible to quantitatively study bubble nucleation, bringing us closer to quantitatively matching experimental observations and validating theoretical models. A qualitative example is shown in Figure 1, where we provide the snapshot from a simulation in the nucleate boiling regime on a 1024^3 domain.

The remainder of this paper is organized as follows. In Section 2 we provide a short overview on the numerical method. Section 3 outlines the key aspects of porting the LBM code to multi-GPU clusters. Performance results and scalability analysis are presented in Section 4. Finally, Section 5 offers concluding remarks and discusses possible future developments.

2. Numerical Method

In this section we provide details on the numerical methods employed for the simulation of pool boiling. We couple two methods, with a hybrid cascaded lattice Boltzmann model (CLBM) addressing the liquid-vapor phase-change process, and a finite difference method for the evolution of the temperature field, and with the two components coupled via a non-ideal equation of state [2].

2.1. Solution of the flow field

The lattice Boltzmann method (LBM) has proven to be a highly efficient and flexible mesoscopic approach for simulating a wide variety of fluid flows [4]. Unlike traditional Navier-Stokes solvers which explicitly discretize the macroscopic equations, LBM is a mesoscopic approach that models the evolution of particle distribution functions over a discrete lattice. At its core, LBM solves a discretized form of the Boltzmann equation:

$$f_i(\mathbf{x} + \mathbf{e}_i\delta t, t + \delta t) - f_i(\mathbf{x}, t) = \Omega(f_i) + F_i, \tag{1}$$

where f_i denotes the particle distribution function along the i-th discrete velocity direction $\mathbf{e}_i$, Ω is the collision operator, F_i is a source term accounting for internal and external forces, and δt is the time-step size. The left-hand side represents the streaming step, which moves information from one lattice node to its neighbors, while the right-hand side captures local changes in the particle distributions due to collisions and forcing effects.

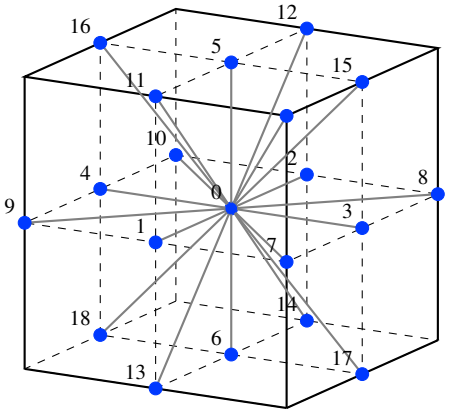

Fig. 2. D3Q19 velocity stencil used for the discretization of the velocity space in the lattice Boltzmann method.

We use the D3Q19 model (see Fig. 2) for the discretization of the continuum velocity space, employing a set of 19 discrete velocities, defined as:

$$\mathbf{e}_i = [|\, e_{ix} \,\rangle, |\, e_{iy} \,\rangle, |\, e_{iz} \,\rangle], \quad i = 0, \ldots, 18,$$

with the specific components given by:

$$\begin{aligned} |\, e_{ix} \,\rangle &= [0, 1, -1, 0, 0, 0, 0, 1, -1, 1, -1, 1, -1, 1, -1, 0, 0, 0, 0]^\top, \\ |\, e_{iy} \,\rangle &= [0, 0, 0, 1, -1, 0, 0, 1, 1, -1, -1, 0, 0, 0, 0, 1, -1, 1, -1]^\top, \\ |\, e_{iz} \,\rangle &= [0, 0, 0, 0, 0, 1, -1, 0, 0, 0, 0, 1, 1, -1, -1, 1, 1, -1, -1]^\top. \end{aligned} \tag{2}$$

Here, $|\cdot\rangle$ denotes a 19-dimensional column vector. The lattice speed is defined as $c = \delta x/\delta t = 1$, and the lattice sound speed is $c_s = 1/\sqrt{3}$, with $\delta x = \delta t = 1$.

In this study, we employ a central-moment-based LBM, which offers enhanced numerical stability and the possibility to independently tune the relaxation rate of the different transport parameters. In CLBM the post-collision state is defined as

$$f_i^* = \Omega(f_i) = f_i^{\mathrm{eq}} + \left(1 - \mathbf{M^{-1}N^{-1}SNM}\right)\left(f_i - f_i^{\mathrm{eq}}\right) \tag{3}$$

where f_i^{eq} is the (discrete) local equilibrium distribution, $\mathbf{M}$ is the transformation matrix mapping the particle distributions to moment space, $\mathbf{N}$ is the transformation matrix shifting from raw moments to central moments, and $\mathbf{S}$ is the relaxation matrix, dictating the relaxation rate of the different moments. Therefore, in the CLBM framework, the collision operator is defined in terms of a set of raw and central moments of the particle distribution functions f_i, respectively

$$k_{mnp} = \left\langle f_i \, e_{ix}^m e_{iy}^n e_{iz}^p \right\rangle, \quad \tilde{k}_{mnp} = \left\langle f_i \, (e_{ix} - u_x)^m (e_{iy} - u_y)^n (e_{iz} - u_z)^p \right\rangle, \tag{4}$$

where m, n, p are non-negative integers and u_x, u_y, u_z are the components of the macroscopic velocity vector $\mathbf{u}$.

Following the work of Fei et al. [5], we consider the following non-orthogonal set of central moments:

$$\left|\tilde{T}_i\right\rangle = [\tilde{k}_{000}, \tilde{k}_{100}, \tilde{k}_{010}, \tilde{k}_{001}, \tilde{k}_{110}, \tilde{k}_{101}, \tilde{k}_{011}, \tilde{k}_{200}, \tilde{k}_{020}, \tilde{k}_{002}, \tilde{k}_{120}, \tilde{k}_{102}, \tilde{k}_{210}, \tilde{k}_{201}, \tilde{k}_{012}, \tilde{k}_{021}, \tilde{k}_{220}, \tilde{k}_{202}, \tilde{k}_{022}]^\top, \tag{5}$$

with

$$|T_i\rangle = \mathbf{M}\,|f_i\rangle = [T_0, T_1, ..., T_{18}]^\top, \quad \left|\tilde{T}_i\right\rangle = \mathbf{N}\,|T_i\rangle = \mathbf{NM}\,|f_i\rangle = [\tilde{T}_0, \tilde{T}_1, ..., \tilde{T}_{18}]^\top. \tag{6}$$

This choice leads to a sparse transformation matrix [5], which allows for enhanced computational efficiency.

The collision process can then be expressed as the independent relaxation of each central moment towards its equilibrium counterpart as:

$$\left|\tilde{T}_i^*\right\rangle = \left|\tilde{T}_i\right\rangle - \mathbf{S}\left(\left|\tilde{T}_i\right\rangle - \left|\tilde{T}_i^{eq}\right\rangle\right) + (\mathbf{I} - \mathbf{S}/2)\delta t\,|C_i\rangle\,, \tag{7}$$

where $|C_i\rangle$ represents the forcing terms in moment space.

The choice for the set of moments in Eq. 5 implies the following block-diagonal form for the relaxation matrix:

$$\mathbf{S} = \text{diag}\left\{\text{s}_0, \text{s}_1, \text{s}_1, \text{s}_1, \text{s}_\nu, \text{s}_\nu, \text{s}_\nu, \begin{bmatrix} s_+ & s_- & s_- \\ s_- & s_+ & s_- \\ s_- & s_- & s_+ \end{bmatrix}, \text{s}_3, \text{s}_3, \text{s}_3, \text{s}_3, \text{s}_3, \text{s}_3, \text{s}_3, \text{s}_4, \text{s}_4, \text{s}_4\right\}, \tag{8}$$

with $s_+ = (s_{2b} + 2s_2)/3$, $s_- = (s_{2b} - s_2)/3$, enabling independent control of normal stress components and bulk viscosity [5].

The equilibrium central moments are matched to those of the continuous Maxwell-Boltzmann distribution:

$$\left|\tilde{T}_i^{eq}\right\rangle = [\rho, 0, 0, 0, 0, 0, 0, \rho c_s^2, \rho c_s^2, \rho c_s^2, 0, 0, 0, 0, 0, 0, \rho c_s^4, \rho c_s^4, \rho c_s^4]^\top. \tag{9}$$

The forcing term in moment space is:

$$|C_i\rangle = [0, F_x, F_y, F_z, 0, 0, 0, 0, 0, 0, F_x c_s^2, F_x c_s^2, F_y c_s^2, F_z c_s^2, F_y c_s^2, F_z c_s^2, 0, 0, 0]^\top. \tag{10}$$

After the collision step, the post-collision distribution functions are reconstructed by:

$$\left|f_i^*\right\rangle = \mathbf{M}^{-1}\mathbf{N}^{-1}\left|\tilde{T}_i^*\right\rangle,$$

and propagated via the streaming step:

$$f_i(\mathbf{x} + \mathbf{e}_i\delta t, t + \delta t) = f_i^*(\mathbf{x}, t). \tag{11}$$

The macroscopic variables are recovered as:

$$\rho = \sum_i f_i, \quad \rho\mathbf{u} = \sum_i f_i \mathbf{e}_i + \frac{\delta t}{2}\mathbf{F}. \tag{12}$$

The kinematic viscosity ν and bulk viscosity ξ are related to the relaxation parameters via:

$$\nu = \left(\frac{1}{s_2} - \frac{1}{2}\right)c_s^2\delta t, \quad \xi = \frac{2}{3}\left(\frac{1}{s_{2b}} - \frac{1}{2}\right)c_s^2\delta t. \tag{13}$$

In order to incorporate phase-change phenomena, the pseudopotential model is adopted [6, 7], with the interaction force:

$$\mathbf{F}_{\text{int}} = -G\psi(\mathbf{x}) \sum_i w(|\mathbf{e}_i|^2)\psi(\mathbf{x} + \mathbf{e}_i\delta t)\mathbf{e}_i. \tag{14}$$

The pseudopotential ψ is defined by:

$$\psi = \sqrt{\frac{2(p_{\text{EOS}} - \rho c_s^2)}{Gc^2}}, \tag{15}$$

where $G = -1$ ensures the positivity of the expression under the square root.

To address thermodynamic inconsistency, Li et al. [8, 9] proposed modifying the forcing term:

$$|C_i\rangle = [0, F_x, F_y, F_z, 0, 0, 0, \eta, \eta, \eta, F_x c_s^2, F_x c_s^2, F_y c_s^2, F_z c_s^2, F_y c_s^2, F_z c_s^2, 0, 0, 0]^\top, \tag{16}$$

where:

$$\eta = \frac{2\sigma |\mathbf{F}_{\text{int}}|^2}{\psi^2 (s_e^{-1} - 0.5)\delta t}, \tag{17}$$

with σ used for tuning η. By applying a Chapman-Enskog analysis, the following macroscopic equations are recovered in the low-Mach limit:

$$\begin{aligned} &\partial_t \rho + \boldsymbol{\nabla} \cdot (\rho \mathbf{u}) = 0, \\ &\partial_t (\rho \mathbf{u}) + \boldsymbol{\nabla} \cdot (\rho \mathbf{u}\mathbf{u}) = -\boldsymbol{\nabla}(\rho c_s^2) + \boldsymbol{\nabla} \cdot \left[\rho \nu (\boldsymbol{\nabla}\mathbf{u} + (\boldsymbol{\nabla}\mathbf{u})^\top) - \tfrac{2}{3}\rho\nu(\boldsymbol{\nabla}\cdot\mathbf{u})\mathbf{I}\right] + \boldsymbol{\nabla}\left[\rho\xi(\boldsymbol{\nabla}\cdot\mathbf{u})\right] + \mathbf{F} - 2G^2c^4\sigma\boldsymbol{\nabla}\cdot\left(|\boldsymbol{\nabla}\psi|^2\mathbf{I}\right) \end{aligned} \tag{18}$$

where the final term arises from the thermodynamic correction via η. Note that for $\sigma = 0$, the standard Navier-Stokes equations are recovered.

2.2. Solution of the Temperature Field

The temperature evolution in the liquid-vapor phase-change process is governed by the following equation [10]:

$$\frac{\partial T}{\partial t} = -\mathbf{u} \cdot \boldsymbol{\nabla} T + \boldsymbol{\nabla} \cdot (\lambda \boldsymbol{\nabla} T) - \frac{T}{\rho c_v} \left(\frac{\partial p_{\text{EOS}}}{\partial T}\right)_\rho \boldsymbol{\nabla} \cdot \mathbf{u}, \tag{19}$$

where T is the temperature, $\mathbf{u}$ is the fluid velocity, λ is the thermal conductivity, ρ is the density, and c_v is the specific heat at constant volume. The term involving $\partial p_{\text{EOS}}/\partial T$ accounts for the thermodynamic coupling due to the non-ideal equation of state.

To numerically solve Eq. 19, spatial derivatives must be discretized appropriately. In this work, a central difference scheme based on lattice directions is employed to approximate both the gradient and Laplacian operators [11]. The discretization is given as:

$$\begin{aligned} \frac{\partial \phi}{\partial x_\alpha} &= \frac{1}{c_s^2 \delta t} \sum_i \omega\left(|\mathbf{e}_i|^2\right) e_{i\alpha}\, \phi\left(\mathbf{x} + \mathbf{e}_i \delta t\right), \\ \frac{\partial^2 \phi}{\partial x_\alpha \partial x_\alpha} &= \frac{2}{c_s^2 \delta t} \sum_i \omega\left(|\mathbf{e}_i|^2\right) \left[\phi\left(\mathbf{x} + \mathbf{e}_i \delta t\right) - \phi(\mathbf{x})\right], \end{aligned} \tag{20}$$

where $\alpha \in \{x, y, z\}$ denotes the spatial coordinate direction, ϕ is a scalar physical quantity (e.g., temperature), $\mathbf{e}_i$ are the discrete lattice velocities, and ω is a weight function associated with the magnitude of $\mathbf{e}_i$.

In order to conserve the heat flux, particularly across the liquid-vapor interface where sharp gradients may occur, the diffusion term $\boldsymbol{\nabla} \cdot (\lambda \boldsymbol{\nabla} T)$ is discretized using a finite volume method [12]. This approach provides better accuracy and stability when handling discontinuities and steep gradients, which are common in phase-change phenomena.

For temporal discretization, the right-hand side of Eq. 19 is denoted as $K(T)$, representing all spatially discretized contributions to the temperature evolution. The time integration is carried out using a fourth-order Runge-Kutta (RK4) method:

$$\begin{aligned} T^{t+\delta t} &= T^t + \tfrac{\delta t}{6}(h_1 + h_2 + h_3 + h_4), \\ h_1 &= K(T^t), \\ h_2 &= K\left(T^t + \tfrac{\delta t}{2} h_1\right), \\ h_3 &= K\left(T^t + \tfrac{\delta t}{2} h_2\right), \\ h_4 &= K\left(T^t + \delta t\, h_3\right). \end{aligned} \tag{21}$$

2.3. Non-ideal equation of state

The pseudopotential CLBM described in Section 2.1 is coupled with the finite-difference temperature solver introduced in Section 2.2 through a non-ideal equation of state. Specifically, temperature variations affect the local thermodynamic pressure p_{EOS}, which in turn modifies the interparticle interaction force in the pseudopotential CLBM via Eqs. 14 and 15.

To model the non-ideal thermodynamic behavior of the fluid, various equations of state—such as the van der Waals, Carnahan-Starling, or Peng-Robinson EOS—can be incorporated into the framework through the square-root form of the pseudopotential defined in Eq. 15. Following a previous study [2], the Peng-Robinson EOS is employed:

$$p_{\text{EOS}} = \frac{\rho RT}{1 - b\rho} - \frac{a\varphi(T)\rho^2}{1 + 2b\rho - b^2\rho^2}, \tag{22}$$

where the temperature-dependent function $\varphi(T)$ is defined as:

$$\varphi(T) = \left[1 + \left(0.37464 + 1.54226\varpi - 0.26992\varpi^2\right)\left(1 - \sqrt{T/T_c}\right)\right]^2,$$

and the parameters $a = 2/49$, $b = 2/21$ have been established following the analysis in Refs. [13, 9], with $\varpi = 0.344$ and $R = 1$. The corresponding critical temperature and pressure are computed as $T_c = 0.07292$ and $p_c = 0.05957$, respectively.

Furthermore, in the simulation of pool boiling phenomena, the motion of vapor bubbles and surrounding liquid is significantly influenced by buoyancy forces arising from density differences. The buoyancy force is modeled as:

$$\mathbf{F}_b = -(\rho - \bar{\rho})g\hat{\mathbf{k}}, \tag{23}$$

where $\bar{\rho}$ is the average density in the computational domain, g is the gravitational acceleration, and $\hat{\mathbf{k}}$ is the unit vector in the vertical (gravity) direction.

As a result, the total body force acting on the system is given by the sum of the interparticle interaction force and the buoyancy force:

$$\mathbf{F} = \mathbf{F}_{\text{int}} + \mathbf{F}_b.$$

This coupling strategy ensures that both temperature-induced pressure variations and gravitational effects are consistently incorporated into the fluid dynamics, enabling accurate simulation of liquid-vapor phase-change processes such as boiling and condensation.

3. Implementation

To achieve simultaneously high performance and portability across heterogeneous computing architectures, our LBM code is implemented using the OpenACC programming model. OpenACC enables incremental parallelization of existing C/C++ and Fortran codebases and supports a broad range of accelerators and CPUs. The primary goal of this implementation is to ensure that the same codebase can be deployed efficiently on NVIDIA GPUs, multi-core CPUs, and potentially other accelerators, without requiring architecture-specific modifications.

The core compute routines, such as streaming, collision, macroscopic moment computation, boundary conditions and the Runge-Kutta solver for the temperature fields, are offloaded to the accelerator using OpenACC directives. Loop-level parallelism is expressed through the kernels construct, allowing the compiler to generate optimized code for the target architecture.

To support parallel updates of all grid points during both streaming and collision operations, we store two copies of the lattice for the particle distribution functions f_i, following the so-called A-A pattern. The lattice data is transferred to the accelerator memory at the beginning of the time-stepping loop, and an OpenACC data region is used to minimize communication between the host and the device, restricting data transfers to essential operations such as disk writes and MPI communications.

The code adopts a Structure of Arrays (SoA) data layout, which aligns with memory coalescing requirements on GPUs and supports efficient vectorization on CPUs. In this layout, the particle distribution functions are stored with

spatial indices contiguously in memory. This organization enhances memory bandwidth utilization, which is a key performance factor in LBM algorithms due to their memory-bound nature. Although more sophisticated data structures for LBM have been proposed, offering performance benefits on architectures with large cache hierarchies [14], they often introduce additional complexity. For the sake of simplicity and maintainability, we opt not to adopt them in the current implementation.

To support large-scale simulations, the computational domain is decomposed and distributed across multiple compute nodes using MPI. For simplicity, we adopt a one-dimensional domain decomposition, where each MPI rank handles a subdomain. Halo regions (ghost layers) are exchanged at each time step to maintain data consistency across subdomain boundaries. The code is designed for multi-node execution, with one MPI process assigned to each GPU when running on GPU clusters.

Inter-process communication is implemented using non-blocking MPI routines, enabling the overlap of computation and communication. Specifically, the interior nodes of each subdomain are updated while halo data is exchanged asynchronously. This overlap is explicitly managed by partitioning the domain into interior and halo regions and launching separate OpenACC kernels for each subdomain. When possible, the kernels are executed on different CUDA streams to further reduce potential overheads introduced by small kernels reserved to the preparation of buffers used in MPI communications and kernels operating on the boundary layers.

We leverage a CUDA-aware MPI library, which allows communication buffers to reference device memory directly. This enables efficient pipelining of data movement between host and device memory, reducing overhead and improving scalability, and also enables seamless use of NVLink for direct GPU-to-GPU data transfers between devices on the same node.

4. Results

In this section we provide a brief overview of the performances of the code.

All results are obtained using a realistic physical configuration that replicates the pool boiling setup illustrated in Fig. 1. We perform three-dimensional simulations where the bottom plate is maintained at a fixed temperature, with a small perturbation applied at $t = 0$ to trigger bubble nucleation. At the top boundary, we impose a constant pressure condition along with a Neumann boundary condition for velocity, enforcing that the normal derivative of the velocity field is set to zero. Periodic boundary conditions are applied in the remaining directions. The initial condition consists of a domain half-filled with liquid and half with vapor. The flow field is solved using the D3Q19 LBM (cf. Fig. 2), while the temperature field is evolved using a fourth-order Runge-Kutta method.

We begin by analyzing the performance obtained on a single GPU. As a representative case, we consider the performance of the two main computational kernels: streaming and collide. The streaming kernel is responsible for moving pseudo-particles between neighboring grid points and is thus entirely memory-bound. On a single NVIDIA A100 GPU, this kernel achieves a bandwidth of 1245.31 GB/s, which corresponds to approximately 62% of the theoretical peak. In contrast, the collide kernel is the most compute-intensive component of the code. On the same A100 GPU, it reaches 2745.43 GFLOP/s, about 28% of the theoretical peak performance, comparing well against results of previous LBM implementations from the literature [15, 16]. These figures nearly double when running on a single H100 GPU at the MareNostrum 5 supercomputer (BSC), achieving 2659.13 GB/s for the streaming kernel and 5616.93 GFLOP/s for the collide kernel. These figures show that the code allows to extract a significant fraction of the performance peak on some of the latest GPU cards.

Moving now to the evaluation of the multi-GPU implementation, Fig. 3a reports on the strong scalability and corresponding parallel efficiency achieved on the MareNostrum 5 cluster.

We consider three representative domain sizes: a small domain of size $1536 \times 512 \times 512$, a medium domain of $3072 \times 1024 \times 1024$, and a large domain of $6144 \times 2048 \times 2048$. In the strong scalability test, the domain is partitioned along the x-dimension as L_x/p, where p is the number of GPUs. We evaluate performance for $p = 4, 8, \ldots, 512$ GPUs.

We compare two code variants: one where communication and computation are overlapped, and another where they are executed synchronously. The plot clearly shows the benefit of overlapping, which brings performance close to ideal scaling up to 256 GPUs. At $p = 512$, execution times of the two variants converge, suggesting that the execution time becomes dominated by the cost of MPI communications. Since this overhead is roughly constant

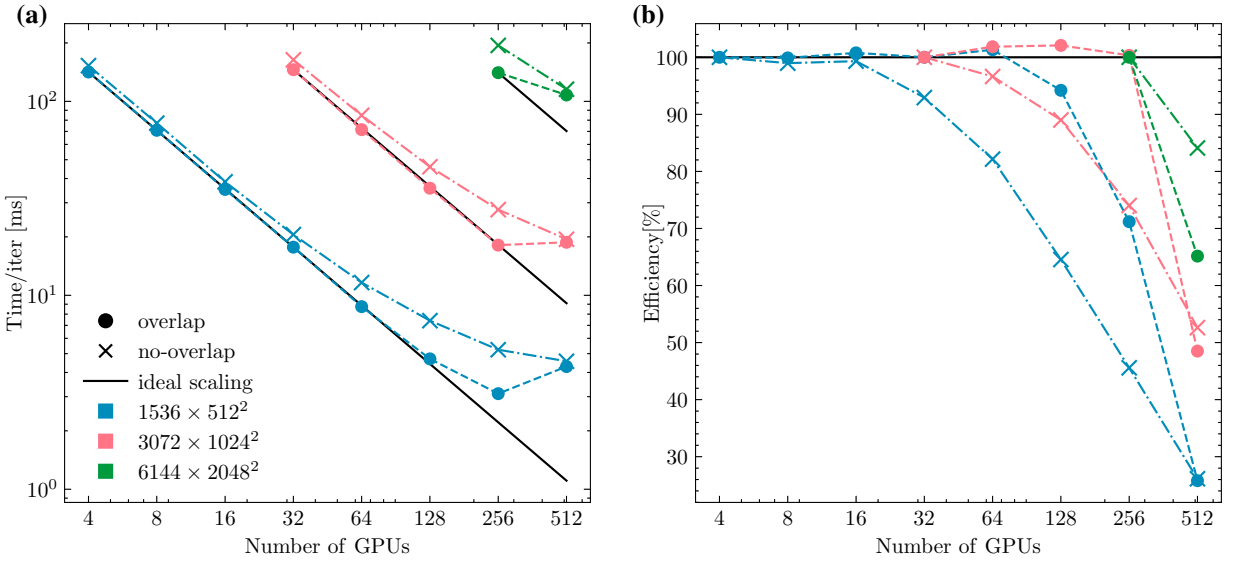

Fig. 3. (a) Strong scalability results for three domain sizes on the MareNostrum 5 cluster, using up to 512 GPUs. Two code variants are compared: with and without overlapping of communication and computation. (b) Parallel efficiency corresponding to the strong scaling test, with baseline GPU counts of 4, 32, and 256 for the small, medium, and large domains, respectively.

with respect to p under 1D partitioning, scalability plateaus. Higher-dimensional domain decompositions could help alleviate this bottleneck, albeit at the cost of added code complexity.

In Fig. 3b, we show the corresponding parallel efficiency, defined as:

$$E(p) = \frac{T(b)}{T(p)} \frac{p}{b} \tag{24}$$

where $T(p)$ is the execution time using p GPUs, and b is the minimum number of GPUs required to fit a given problem size in memory (4 for the small domain, 32 for the medium, 256 for the large).

Remarkably, for the medium-size domain, efficiency remains close to 100% up to 256 GPUs. Minor super-linear speedups can be observed and are attributed to runtime kernel scheduling on CUDA streams. These effects are difficult to predict *a priori* and depend on available GPU resources. When the workload per GPU decreases (at higher p), the potential for concurrent kernel execution increases, which may yield slight performance boosts.

5. Conclusions and future work

We have presented a multi-GPU implementation of a Lattice Boltzmann Method (LBM) code designed for large-scale simulations of pool boiling.

The code exhibits excellent strong scaling, reaching near-ideal performance up to 256 GPUs on representative three-dimensional domains. This opens up the possibility of performing high-resolution numerical studies of the complex dynamics of boiling at scales that were previously computationally prohibitive.

Extended details on the implementation, validation strategy, and performance analysis will be provided in an extended version of this work. Moreover we plan to use the code for studying in detail the transition between boiling regimes as a function of wall superheat, the spatiotemporal evolution and distribution of bubble footprint areas, and the influence of geometrical and thermophysical properties on boiling curves. Work along these lines is in progress.

Acknowledgments

We acknowledge HPC-Europe computational facilities and allocation on the MareNostrum 5 supercomputer (BSC, Spain) through grant EHPC-EXT-2023E02-035, and the Karolina cluster (IT4I, Czech Republic) through grant

EHPC-DEV-2023D04-032. A.G. gratefully acknowledges the support of the U.S. Department of Energy through the LANL/LDRD Program under project number 20240740PRD1 and the Center for Non-Linear Studies for this work.

Appendix A. Scaling performance Data

For completeness, in Table A.1 we report the data presented in Fig. 3.

Table A.1. Scaling performance: time per iteration [ms], speedup, and parallel efficiency for different problem sizes and GPU counts

nGPUs	nNodes	No overlap			With overlap		
		Time [ms]	Speedup	Efficiency [%]	Time [ms]	Speedup	Efficiency [%]
Small problem size: $1536 \times 512 \times 512$							
4	1	152.87	1.00	100.0	141.69	1.00	100.0
8	2	77.23	1.98	99.0	70.92	2.00	100.0
16	4	38.48	3.97	99.3	35.15	4.03	100.8
32	8	20.56	7.44	92.9	17.71	8.00	100.0
64	16	11.63	13.14	82.1	8.74	16.21	101.3
128	32	7.40	20.66	64.6	4.70	30.15	94.0
256	64	5.24	29.17	45.6	3.11	45.56	71.2
512	128	4.57	33.44	26.1	4.29	33.04	25.8
Medium problem size: $3072 \times 1024 \times 1024$							
32	8	163.74	1.00	100.0	145.61	1.00	100.0
64	16	84.72	1.93	96.7	71.48	2.04	102.0
128	32	45.99	3.56	89.0	35.66	4.08	102.0
256	64	27.65	5.92	74.0	18.14	8.03	100.4
512	128	19.45	8.42	52.6	18.76	7.76	48.5
Large problem size: $6144 \times 2048 \times 2048$							
256	64	194.34	1.00	100.0	140.57	1.00	100.0
512	128	115.54	1.68	84.0	107.87	1.30	65.0

Appendix B. EuroHPC Supercomputers

In this appendix, we provide detailed hardware specifications for the supercomputing systems employed in this study.

Karolina Cluster – Accelerated Compute Nodes

The accelerated partition of the Karolina Cluster, hosted at IT4Innovations National Supercomputing Center (Czech Republic), comprises 72 GPU-accelerated nodes equipped with NVIDIA A100 GPUs. This system was primarily utilized for early-stage benchmarking and development of the code.

- **Total Nodes:** 72
- **Processor:** 2× AMD EPYC™ 7763 (2.45 GHz, 64 cores each) per node
- **CPU Cores per Node:** 128 (9,216 cores in total)
- **Main Memory:** 1,024 GB DDR4 3200 MT/s per node
- **GPU Configuration:** 8× NVIDIA A100 GPUs per node (320 GB HBM2 memory per node)
- **Peak Performance:** 361.27 TFLOPS
- **Interconnect:** 4× 200 Gb/s InfiniBand ports per node

MareNostrum 5 ACC (Accelerated Partition)

The accelerated partition of MareNostrum 5, operated by the Barcelona Supercomputing Center (Spain), consists of 1120 nodes based on the latest Intel Sapphire Rapids processors and NVIDIA Hopper GPUs. This system was used for large-scale performance evaluation and production runs.

- **Total Nodes:** 1,120
- **Processor:** 2× Intel Sapphire Rapids 8460Y+ (2.3 GHz, 40 cores each) per node
- **CPU Cores per Node:** 80
- **Main Memory:** 512 GB DDR5 per node
- **GPU Configuration:** 4× NVIDIA Hopper GPUs with 64 GB HBM2 memory each
- **Storage:** 480 GB NVMe (scratch space)
- **Interconnect:** 4× NDR200 ports (800 Gb/s per node)
- **Peak Performance:** 260 PFLOPS
- **Network Topology:** Fat-tree topology with full non-blocking 160-node islands and 2:1 contention between islands

References

[1] V. Dhir, Boiling heat transfer, Annual review of fluid mechanics 30 (1) (1998) 365–401. doi:10.1146/annurev.fluid.30.1.365.

[2] L. Fei, J. Yang, Y. Chen, H. Mo, K. H. Luo, Mesoscopic simulation of three-dimensional pool boiling based on a phase-change cascaded lattice Boltzmann method, Physics of Fluids 32 (10) (2020) 103312. doi:10.1063/5.0023639.

[3] H. Jiang, Y. Liu, H. Chu, A review of numerical investigation on pool boiling, Journal of Thermal Analysis and Calorimetry 148 (17) (2023) 8697–8745. doi:10.1007/s10973-023-12292-0.

[4] S. Succi, The Lattice Boltzmann Equation: For Complex States of Flowing Matter, OUP Oxford, 2018. doi:10.1093/oso/9780199592357.001.0001.

[5] L. Fei, K. H. Luo, Q. Li, Three-dimensional cascaded lattice boltzmann method: Improved implementation and consistent forcing scheme, Physical Review E 97 (5) (2018) 053309. doi:10.1103/PhysRevE.97.053309.

[6] X. Shan, H. Chen, Lattice boltzmann model for simulating flows with multiple phases and components, Physical Review E 47 (3) (1993) 1815. doi:10.1103/PhysRevE.47.1815.

[7] X. Shan, H. Chen, Simulation of nonideal gases and liquid-gas phase transitions by the lattice boltzmann equation, Physical Review E 49 (4) (1994) 2941. doi:10.1103/PhysRevE.49.2941.

[8] Q. Li, K. H. Luo, X. Li, Forcing scheme in pseudopotential lattice boltzmann model for multiphase flows, Physical Review E 86 (1) (2012) 016709. doi:10.1103/PhysRevE.86.016709.

[9] Q. Li, K. H. Luo, X. Li, Lattice boltzmann modeling of multiphase flows at large density ratio with an improved pseudopotential model, Physical Review E 87 (5) (2013) 053301. doi:10.1103/PhysRevE.87.053301.

[10] Q. Li, Q. Kang, M. M. Francois, Y. He, K. H. Luo, Lattice boltzmann modeling of boiling heat transfer: The boiling curve and the effects of wettability, International Journal of Heat and Mass Transfer 85 (2015) 787–796. doi:10.1016/j.ijheatmasstransfer.2015.01.136.

[11] T. Lee, C.-L. Lin, A stable discretization of the lattice boltzmann equation for simulation of incompressible two-phase flows at high density ratio, Journal of Computational Physics 206 (1) (2005) 16–47. doi:10.1016/j.jcp.2004.12.001.

[12] R. Huang, H. Wu, N. A. Adams, Lattice boltzmann model with self-tuning equation of state for multiphase flows, Physical Review E 99 (2) (2019) 023303. doi:10.1103/PhysRevE.99.023303.

[13] P. Yuan, L. Schaefer, Equations of state in a lattice boltzmann model, Physics of Fluids 18 (4) (2006) 042101. doi:10.1063/1.2187070.

[14] E. Calore, A. Gabbana, S. F. Schifano, R. Tripiccione, Optimization of lattice boltzmann simulations on heterogeneous computers, The International Journal of High Performance Computing Applications 33 (1) (2019) 124–139. doi:10.1177/1094342017703771.

[15] E. Calore, A. Gabbana, J. Kraus, E. Pellegrini, S. Schifano, R. Tripiccione, Massively parallel lattice–boltzmann codes on large gpu clusters, Parallel Computing 58 (12) (2016) 1–24. doi:10.1016/j.parco.2016.08.005.

[16] E. Calore, A. Gabbana, J. Kraus, S. F. Schifano, R. Tripiccione, Performance and portability of accelerated lattice Boltzmann applications with OpenACC, Concurrency and Computation: Practice and Experience 28 (12) (2016) 3485–3502. doi:10.1002/cpe.3862.