SunwayLB: Enabling Extreme-Scale Lattice Boltzmann Method Based Computing Fluid Dynamics Simulations on Advanced Heterogeneous Supercomputers

Zhao Liu, *Associate Member, IEEE*, Xuesen Chu, Xiaojing Lv, Hongsong Meng, Hanyue Liu, Guanghui Zhu, Haohuan Fu, *Senior Member, IEEE*, and Guangwen Yang

***Abstract*—The Lattice Boltzmann Method (LBM) is a class of Computational Fluid Dynamics methods which models the fluid as fictive particles. In this paper, we report our work on SunwayLB, which enables LBM based solutions aiming for industrial applications using advanced heterogeneous systems such as the Sunway supercomputers. We propose several techniques to boost the simulation speed and improve the scalability of SunwayLB, including a customized multi-level domain decomposition and data sharing scheme, a carefully orchestrated strategy to fuse kernels with different performance constraints for a more balanced workload, and optimization strategies for assembly code. Based on these optimization schemes, we manage to scale SunwayLB on three advanced supercomputers: Sunway TaihuLight, the new Sunway Supercomputer and a GPU cluster. On Sunway TaihuLight, our largest simulation involves up to 5.6 trillion lattice cells, achieving 11,245 billion cell updates per second (GLUPS), 77% memory bandwidth utilization and a sustained performance of 4.7 PFlops. We further improve the memory bandwidth utilization and computational efficiency using the unique features of a new generation of Sunway supercomputer. On the new Sunway Supercomputer, the largest simulation contains over 4.2 trillion lattice cells, resulting in 6,583 GLUPS, 81% memory bandwidth utilization and a sustained performance of 2.76 PFlops. To evaluate the portability of our code, we also adapt our code to a GPU cluster with tailored optimization techniques, resulting in 191x speedup and 83.8% memory bandwidth utilization. We demonstrate a series of computational experiments for extreme-large scale fluid flow, as examples of real-world applications, to check the validity and performance of our work. The results show that our implementation is competent to be a highly scalable and efficient solution for large-scale CFD problems on heterogeneous systems.**

***Index Terms*—Sunway supercomputers, heterogeneous (hybrid) systems, numerical algorithms and problems, lattice boltzmann method.**

Manuscript received 23 February 2023; revised 8 October 2023; accepted 7 December 2023. Date of publication 18 December 2023; date of current version 5 January 2024. This work was supported in part by the Jiangsu Innovation Capacity Building Program under Grant BM2022028, in part by the Laoshan Laboratory Science and Technology Innovation Program under Grant LSKJ202202204, and in part by the National Natural Science Foundation of China under Grant T2125006. An earlier version of this paper was presented at the IEEE International Parallel and Distributed Processing Symposium in 2019, pp. 557–566 [doi: 10.1109/IPDPS.2019.00065]. Recommended for acceptance by Amanda Randles. *(Corresponding authors: Xuesen Chu; Guangwen Yang.)*

Zhao Liu and Guangwen Yang are with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China, and also with the National Supercomputing Center in Wuxi, Wuxi 214000, China (e-mail: liuz18@mails.tsinghua.edu.cn; ygw@tsinghua.edu.cn).

Xuesen Chu is with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China, also with China Ship Scientific Research Center, Wuxi 214082, China, and also with the Taihu Laboratory of DeepSea Technological Science, Wuxi 214082, China (e-mail: chuxs@cssrc.com.cn).

Xiaojing Lv is with the Department of Software Engineering and Technology, China Ship Scientific Research Center, Wuxi 214082, China (e-mail: jing3704@126.com).

Hongsong Meng and Hanyue Liu are with the Department of Parallel Optimization, National Supercomputing Center in Wuxi, Wuxi 214000, China (e-mail: 1061851143@qq.com; liuhanyue15@lzu.edu.cn).

Guanghui Zhu is with the Institute of Electronics and Information Technology, Zhengzhou University of Technology, Zhengzhou 450044, China, and also with the Department of Government and Enterprise Business, Huirong Electronic Systems Engineering Company Ltd., Shanghai 200001, China (e-mail: jn_zgh@126.com).

Haohuan Fu is with the Department of Earth System Science, Tsinghua University, Beijing 100084, China (e-mail: haohuan@tsinghua.edu.cn).

Digital Object Identifier 10.1109/TPDS.2023.3343706

I. Introduction

COMPUTATIONAL Fluid Dynamics (CFD) has become a crucial tool in various industries, including aerospace, automobile, and machinery, to aid engineering design. In contrast to experimental methods, CFD simulation consumes fewer resources and provides more detailed results. Direct Numerical Simulation (DNS) is recognized as a highly effective method for obtaining high-fidelity solutions of turbulence without further modeling such as Reynolds-Averaged Navier-Stokes (RANS) closure. DNS can provide complete and accurate knowledge of all points of the flow and all times of the simulation period, producing highly reliable and detailed non-empirical results for researchers. However, the computational cost of DNS is high, even at relatively low Reynolds numbers. RANS is the most widely used method for turbulent flow simulation due to its computational efficiency, but the simulation results are affected by turbulence models, and a universal model suitable for all situations is yet to be found. The Large Eddy Simulation (LES) approach can be considered a compromise between DNS and RANS, reducing computational cost by ignoring the smallest length scales of turbulent flows.

Solving real-world industrial problems with DNS or LES cases can take months or years, even when run on supercomputers due to their high computational costs. Over the course

of the last two decades, the Lattice Boltzmann Method (LBM) has garnered significant interest within the research community, primarily owing to its aptitude for conducting DNS or LES in the realm of intricate fluid dynamics simulations [15]. LBM models the fluid as fictive particles that undergo collision and propagation over a discrete lattice mesh, making it suitable for dealing with complex boundaries, incorporating microscopic interactions, and parallelization of the algorithm. The particulate nature and spatial locality of the collision and propagation operators make LBM a promising method for complex fluid simulations. [10], [11].

Various computing devices and accelerators are available in modern supercomputing systems. Previous research has shown that the LBM can be parallelized on heterogeneous supercomputing systems with special-purpose accelerators, such as general-purpose graphics processing units (GPGPUs), Xeon Phi processors, and FPGAs [9], [24], [25]. The objective of this work is to find a practical solution for industrial applications and solve extreme-scale computing problems within days, utilizing the world's most powerful supercomputers. We mainly focus on Sunway supercomputers, which are equipped with "on-chip heterogeneous" Shenwei processors that offer powerful floating-point computing capabilities.

In this paper, we report our work on SunwayLB (Sunway Lattice Boltzmann code), an efficient and scalable LBM implementation targeting the Sunway supercomputers, which are the among the state-of-the-art supercomputers according to the Top500 list. A lot of efforts have been made to scale applications in different research domains to the Sunway supercomputers [28], [29], [30], [31], [32], [33], [34]. However, an LBM implementation that can take full advantage of the scalability of Sunway supercomputers has not yet been developed. To assess the feasibility of using Sunway supercomputers for LBM and perform large-scale simulations of complex geometries and scenarios, we design and build SunwayLB to exploit the computational potential of millions of cores for computational fluid dynamics simulations. Our major contributions include:

- We have created a comprehensive software framework designed for large-scale industrial applications, which includes various modules such as a mesh generator, a pre-processing module, an LBM solver, a post-processing module, and other features like domain partitioning, parallel I/O, and visualization interfaces. The goal of this framework is to provide a complete solution for large-scale simulations and facilitate the use of LBM in industrial applications.
- We propose several techniques to achieve the high performance and scalability on Sunway TaihuLight supercomputer, including: a customized multi-level parallelization approach, a carefully orchestrated strategy to fuse multiple kernels with different performance constrains and assembly code optimization strategies such as manual loop unroll and instructions reordering to exploit the computational potential of the many-core processors.
- We further improve the memory bandwidth utilization and computational efficiency using the unique features of a new generation of Sunway supercomputer, achieving even higher memory bandwidth utilization.
- To evaluate the portability and performance, we adapt SunwayLB to a GPU cluster with different optimization techniques, including utilization of the pinned memory, pre-computation of high-overhead operations, efficient communication strategy using NVIDIA Collective Communication Library (NCCL), etc.

After all the optimization strategies are implemented, we are able to utilize Sunway supercomputers and GPU clusters for ultra-high-resolution simulations. We perform direct numerical simulations containing up to 5.6 trillion lattice cells with linear parallel efficiency, 77% of total memory bandwidth utilization, a sustained performance of 4.7 PFlops and 11245 billion lattice cell updates per second (GLUPS). A series of computational experiments for extreme-large scale fluid flow are performed to validate the performance and adaptability of SunwayLB, including simulations of wind flow in complex urban areas which are widely needed by wind power industries to demonstrate the practical value of SunwayLB. With the capability to perform ultra-high-resolution simulation and excellent simulation speed powered by the Sunway TaihuLight, we are able to simulate more complex geometries and phenomena, exposing the details of the simulations, and most importantly, solving real-world problems within days, thus providing a practical DNS/LES solution for industrial applications [28].

II. Related Work

The Lattice Boltzmann Method (LBM) originated from the Lattice Cellular Automata (LCA) proposed by McNamara and Zanetti [1]. LBM adopted the individual particles concepts and collision-streaming operation from LCA but replaced the Boolean value with an averaged directionally discrete distribution function. Qian et al. [2] introduced a major simplification by replacing the collision matrix with a single relaxation time, called the LBGK model. Lallemand and Luo demonstrated that LBM can be derived from the continuous Boltzmann equation and can be regarded as a special discretized form of the Boltzmann equation from the Chapman-Enskog expansion [3], [4].

In terms of related research, a significant amount of work has been conducted on homogeneous supercomputers. Jelinek et al. [8] have proposed a Lattice Boltzmann cellular automaton model for simulating 2D dendritic growth on Kraken and Talon supercomputers, achieving a simulation speed of 133 GLUPS on Kraken. Several open-source CFD software based on LBM are available, such as OpenLB, HemeLB, and waLBerla. For example, Krause et al. [14] have reported on the simulation of human lung expiration using OpenLB, which was scaled up to 256 processes. Groen et al. [12] have presented the performance results for the cell-structured HemeLB framework on vascular geometries with 20 million lattice cells on HeCTOR and SuperMUC supercomputers, achieving a performance of 29.5 GLUPS and 68.8 GLUPS, respectively. Götz et al. [18] have used waLBerla for direct numerical simulation of particulate flows with up to 150 billion lattice cells on a Blue Gene system, sustaining 188 GLUPS and 84 TFlops. Godenschwager et al. [11] have

employed the waLBerla framework for blood flow simulations on the coronary artery tree, performing up to 837 GLUPS with 450 billion cells on SuperMUC and up to 1930 GLUPS with 790 billion cells on JUQUEEN, respectively. Schornbaum and Rüde [10] have achieved the largest LBM simulation to date on homogeneous supercomputers by pushing the simulation scale to 886 billion on JUQUEEN, achieving a simulation speed of 889 GLUPS.

In addition to CPU-based platforms, GPUs have been used for LBM implementations for over a decade. Tölke et al. [26] first attempted to implement LBM with the D3Q13 discretization scheme on a single GPU, while Bailey et al. [22] demonstrated that GPUs can outperform CPUs in LBM implementation using the D3Q19 discretization scheme. The use of supercomputers equipped with GPUs has become popular for large-scale simulations due to the increased computation power of GPUs. Feichtinger et al. [7] achieved a speed of 245 GLUPS on the waLBerla framework using more than 1000 GPUs of the Tsubame 2.0 cluster, while Riesinger et al. [9] developed a scalable LBM implementation approach for a series of supercomputers with GPUs, achieving 2605 GLUPS using 24576 CPU cores and 2048 GPUs with a mesh size of 7 billion and utilizing 69% of the total memory bandwidth. Meanwhile, the heterogeneous architeture also poses huge challenges in terms of memory footprint and data motion between CPU and GPU. Valero-Lara et al. [38], [39] proposed a strategy using ghost cells to minimize the memory requirements and to avoid race conditions among CUDA blocks. Gounley et al. [37] futher suggested a moment representation algorithm which significantly reduces the amount of data required for updating lattice cells, thus alleviating the data motion problem and improving the simulation performance on CPU-GPU systems.

The Sunway supercomputers adopt a different design philosophy from the hardware platforms mentioned above by using 'on-chip heterogeneous processors. We manage to scale our LBM code SunwayLB to more than 10 million cores of Sunway TaihuLight supercomputer, achieving a sustained performance of 4.7 PFlops [36]. In the context of this research endeavor, we have introduced novel conceptual extensions to our prior work. To elaborate, we have undertaken significant enhancements to our original parallelization and optimization strategies, thereby enabling us to more effectively harness the memory bandwidth and computational capabilities of a next-generation Sunway supercomputer, succeeding the Sunway TaihuLight. Furthermore, in our pursuit of evaluating both portability and performance, we have adapted SunwayLB for deployment on a GPU cluster, employing a diverse set of optimization techniques. These advancements empower our code to be seamlessly utilized across varying heterogeneous architectures.

III. The Sunway Supercomputers and Software Design Challenges

A. Systems Overview

The series of Sunway supercomputers have a history of over 25 years in China. Among them the Sunway TaihuLight is the most well-known one for being the first supercomputer surpassing the peak performance of 100 PFlops in 2016 [35]. In order

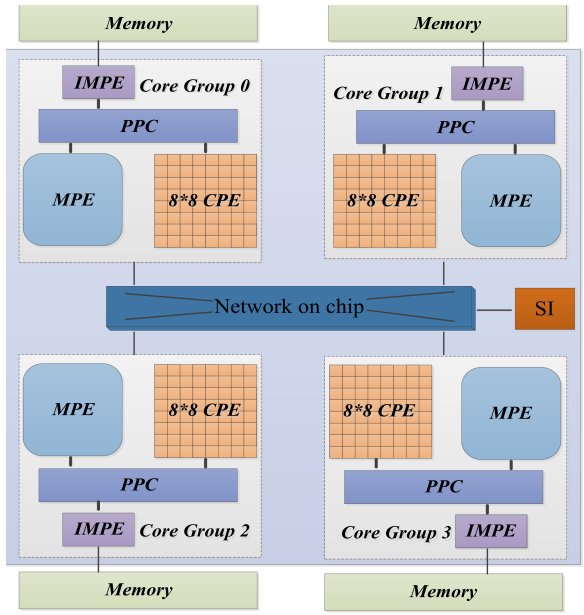

Fig. 1. Architecture of SW26010.

to meet the increasing computational demand of simulations with extreamely complex processes and high resolution, a new generation of Sunway supercomputer with a peak performance of over 1.5 EFlops was published in 2021 [33]. A considerable number of software code have been tuned and optimized on Sunway TaihuLight and the new Sunway supercomputer [28], [29], [30], [31], [32], [33], [34].

The Sunway TaihuLight is a 40-rack supercomputing system with 40960 260-core SW26010 processors, providing a peak performance of 125PFlops and a sustained LINPACK performance of 93PFlops, while the new Sunway supercomputer consists of 107520 390-core SW26010-Pro CPUs with a peak performance of over 1.5 EFlops. The differences between SW26010 and SW26010-Pro processors are discussed in Section III-B. Both systems adopt a unique design for the network topology: supernode. One supernode is composed of 256 processors, and all processors within a supernode are fully connected by a customized network switch board, enabling highly efficient all-to-all communications. The network topology for connecting all the supernodes is a fat tree using a proprietary high-speed interconnect network as depicted in Fig. 2(b).

The software systems are developed to adapt SW26010 and SW26010-Pro processors, including customized 64-bit Linux OS kernel, global file system, compliers, job management system, etc. C/C++ and Fortran are supported by the Sunway compiling systems as well as mainstream parallel programming standards such as MPI and OpenACC. A customized SACA (Sunway Accelerate Computing Architecture) library provides interfaces of manipulating the CPEs and support for dynamic loading [28], [35].

B. The SW26010 and SW26010-Pro Processors

The SW26010 processor is designed by Shanghai High Performance IC Design Center. It is a many-core processor with 260 heterogeneous cores providing a peak performance of 3.06

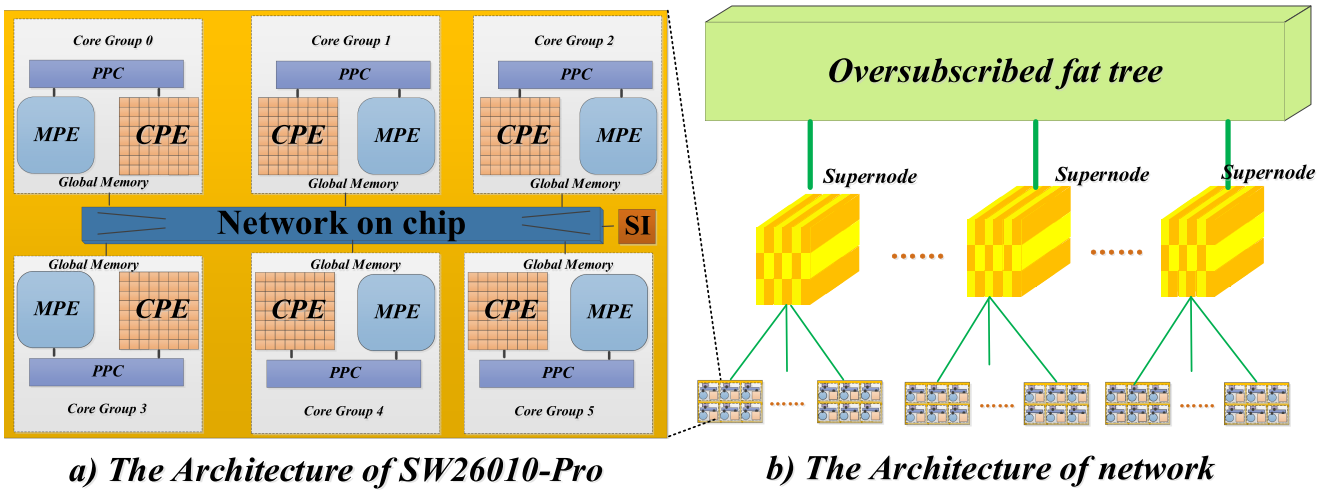

Fig. 2. (a) Architecture of SW26010-Pro processor. The SW26010-Pro processor is equipped with 390 heterogeneous cores divided into 6 CGs. Each CG has 1 MPE and 64 CPEs organized as 8 by 8 mesh. The major computing power is provided by the CPE mesh. (b) Architecture of system network. A total of 256 processors form a supernode. All processors within a supernode are fully connected by a customized network switch board, enabling highly efficient all-to-all communications. The network topology for connecting all the supernodes is a fat tree using a proprietary high-speed interconnect network.

TFlops and a performance-to-power ratio of 10 GFlops/Watt. As in Fig. 1 shows, the 260 cores are divided into four core-groups (CG).

Each CG is composed of one management processing element (MPE), one intelligent memory processing element (IMPE) and 64 computing processing elements (CPEs) organized as 8 by 8 mesh. MPEs, IMPEs and CPEs are designed for different goals. The IMPE has single instruction fetch unit and multiple instruction buffer, mainly targeting at memory access operations. The MPE is a complete 64-bit RISC core with a frequency of 1.45 GHz, mainly targeting to handle the flow control of a program, I/O and communication functions. While CPEs adopt a more simplified microarchitecture to maximize the aggregated computing power. Each CPE has 16 KB L1 instruction cache and 64 KB local data memory (LDM) which can be configured as user-controlled fast buffer. A performance model based on the three-level (REG-LDM-MEM) memory hierarchy was proposed by Fang et al [16]. The CPE can either directly access the global memory with a limited bandwidth of 8GB/s, or through a REG-LDM-MEM memory hierarchy to obtain much higher bandwidth. Each CPE has two pipelines to process instruction decoding and execution. A carefully orchestrated instruction reordering scheme can alleviate the dependences of instruction sequence, thus potentially improve the instruction execution efficiency. Inside each CPE cluster, a mesh network with 8 row communication buses and 8 column communication buses enables fast data communications and sharing at the level of registers, which allows efficient data reuse and communication among CPEs [28], [35].

SW26010-Pro processors deliver massive threads and data parallelism to achieve high performance for scientific and engineering simulations. The SW26010-Pro processor is equipped with 390 heterogeneous cores providing 14.03 TFlops of compute at either FP64 or FP32 precision and 55.3 TFlops at BF16 or FP16 precision. As Fig. 2(a) shows, the 390 cores are divided into 6 CGs. Each CG has one independent memory controller (MC) connecting to 16 GB DDR4 memory. The aggregated memory bandwidth of SW26010-Pro reaches 307.2 GB/s (51.2GB/s for each CG). The MPE is a complete 64-bit RISC core with a frequency of 2.1 GHz, mainly targeting to handle the flow control of a program, I/O and communication functions, while CPEs operate at a frequency of 2.25 GHz maximize the aggregated computing power.

There are two major differences between SW26010 and SW26010-Pro. First, the LDM capacity of SW26010-Pro has been increased to 256 KB per CPE and the vector width has been doubled to 512-bits compared with that of SW26010. Second, data sharing among CPEs of SW26010-Pro is supported by a low-latency communication mechanism called Remote Memory Access (RMA) instead of register communication. RMA allows not only row/column broadcast, but also one-sided communication between CPEs. These updates and new features provide possibility for further improvement of SunwayLB.

C. *The Major Challenges for Implementing LBM on Sunway Supercomputers*

The challenges are the severe memory constraint from both the hardware and the software algorithm. The aggregated memory bandwidth of SW26010-Pro processor is 307.2 GB/s while the performance in double-precision reaches 14.03 TFlops, leading to a B/F (Bytes per FLOP) of 0.022. which is much lower than other state-of-the-art computing systems, posing huge challenges for scientific applications requiring high memory bandwidth to achieve high performance. The LBM is a typically memory-bound algorithm due to its stencil-like computation pattern. There are two phases in one LBM time step: collision (also called relaxation) and propagation (also called streaming). To exploit the full potential of all the computing cores within the SW26010-Pro processor, the CPE cluster have to access data

from the REG-LDM-MEM memory hierarchy, which means all related data need to be transferred from main memory to LDM through DMA. The discontinuous memory accesses will prevent the program from achieving high DMA bandwidth utilization. Therefore, a carefully orchestrated domain decomposition and memory access scheme is required to effect utilize the memory bandwidth.

IV. Algorithm Design and Innovation Realized

A. The Lattice Boltzmann Method and Basic Implementation

Common solutions of CFD problems are based on solving governing partial differential equations, e. g. Euler or Navier-Stokes equations. The computational domain is then discretized with finite volumes, finite differences or finite elements methods so that an algebraic system of equations can be derived and solved numerically.

While the LBM is a mesoscopic method which models the fluid with fictive particles, it originates from the cellular and lattice gas automata. We do not assign the physical values of the fluid to the lattice cells directly but use $\vec{x}$ to denote the position of particles in space, $\vec{u}$ to denote the velocity of particles, and t as the time parameter, so that the density distribution function f $(\vec{x}, \vec{u}, \mathrm{t})$ is discretized in space, velocity and time for each lattice cell. As a result, the computational domain is regularly divided into cells of the Cartesian grid in 2D squares or 3D cubes, and the updates of each cell only require data from the current and neighboring cells instead of the distant cells because the virtual particles move at most to the adjacent cells.

In this paper, the LBGK model introduced by Qian et al. [2] is adopted. For velocities of each cell, a floating-point number representing the fraction of particles moving with this velocity is stored and governed by (1).

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

Here, $f_i(\vec{x}, t)$denotes velocity distribution function (DF) in i direction at $\vec{x}$ position and t time, $f_i(\vec{x}+\vec{c}_i\Delta t)$ indicates DF in i direction of the neighborhood cell, Δt is the time step size, $\vec{c}_i$ represents the lattice velocity vector in direction, τ refers to relaxation time, corresponding to the lattice viscosity ν by $\nu = (2\tau - 1)/6$, and $f_i^{(eq)}$ denotes the equilibrium distribution function in direction.

One common way of classifying the different methods by lattice is the DnQm scheme. Here "*Dn*" stands for "*n* dimensions," while "*Qm*" stands for "*m* speeds." D2Q9 scheme is most frequent used for 2D model. For 3D model, several cubic lattice models such as D3Q13, D3Q15, D3Q19, and D3Q27 are proposed. In this paper, D3Q19 model is adopted, which is a regular 3D lattice with 19 populations, as shown in Fig. 3.

The iteration of LBM program is achieved by alternating the collision and propagation steps. For each time step, all lattice cells in the domain are traversed and the distribution functions of each cell are updated using the data of neighboring cells at the previous time step. The update step consists of two operations,

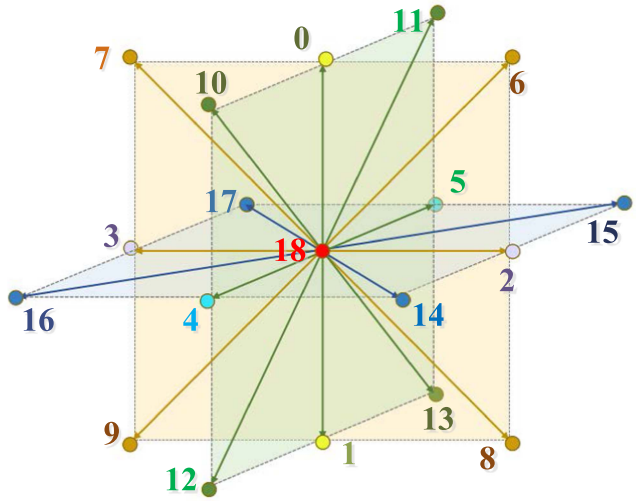

Fig. 3. D3Q19 discretization scheme.

propagation and collision. In the propagation operation, the data needed for updating a lattice cell are received from the neighboring cells, while in the collision operation the values of distribution function are computed according to the LBGK model.

In our current research, we have chosen to employ the pull approach, as introduced by [40] and discussed in [38], which has proven to be highly efficient. It operates on a single-loop strategy where each lattice node can be independently calculated by executing a complete time step of LBM. Essentially, the pull approach combines both collision and streaming operations within a single loop that iterates over the entire computational domain. This integration serves to enhance temporal locality while eliminating the need for synchronization between these operations. Additionally, it offers advantages in terms of memory utilization compared to alternative approaches, which is beneficial to the Sunway manycore processor considering the limited capacity of LDM and memory bandwidth.

The way of storing all the particle populations in memory is crucial for achieving high performance and memory bandwidth. One intuitive way is to use a data structure to store particle populations and other related data for each cell resembling array of structures (AoS). As the D3Q19 model shows, when updating one cell, 19 particle populations from its neighbors will be needed, which are not continuous in memory if we adopt AoS, resulting in large amount of random memory accesses and frequent DMA startups. The overhead of DMA of this strategy is a severe limitation to the overall performance. Therefore, we apply a structure of arrays (SoA) pattern to store all the particle populations continuously in memory [28].

We have adopted a two-level parallel architecture, combining MPI with Athread, to implement our solver. The MPI processes are managed by MPEs to oversee coarse-grained domain decomposition, process-level communication, and I/O operations. Athread, a specialized lightweight thread library designed specifically for Sunway Supercomputers, is utilized to enable programmers to achieve finer-grained optimization. In order

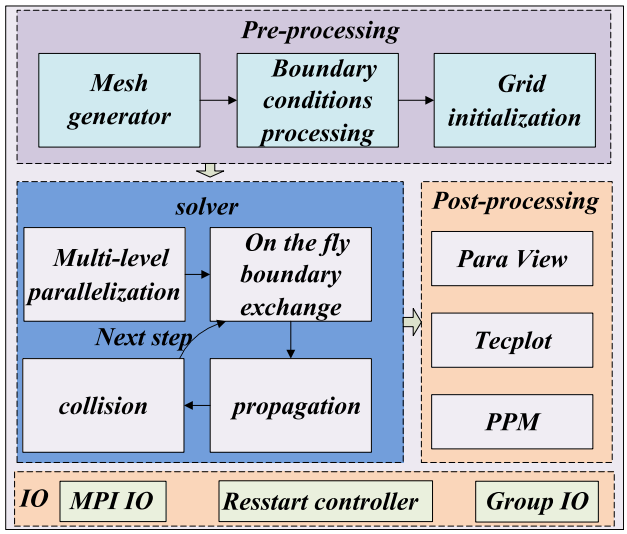

Fig. 4. Holistic framework of SunwayLB.

to make proper use the heterogeneous architecture, we offload kernels to CPEs using Athread interfaces. The Athread library also provides other important functions such as synchronization among CPEs, DMA operators and register communication, etc. This hybrid approach allows us to effectively balance between coarse and fine-grained parallelism.

B. *The Holistic Framework of SunwayLB*

To provide a holistic solution for CFD simulation, we build the SunwayLB framework to provide a holistic solution for CFD simulation as Fig. 4 shows, including: the pre-processing module, the LBM solver, the post-processing module and the I/O layer support. The pre-processing module includes three key features: mesh generation, boundary conditions processing and grid initialization. The mesh generator supports three different kinds of input: geometries from CAD tools with stl format, terrain files from GIS software and the outline described directly inside SunwayLB. The solver is an LBM implementation based on D3Q19 discretization scheme, which has been carefully redesigned and refactored to exploit the computational potential of Sunway TaihuLight. The key functions of the solver include multi-level parallelization scheme, on-the-fly boundary exchange, propagation and collision operators.

As for the post-processing module, several kinds of post processing interfaces are supported by our framework, providing proper formats of data, data analysis and visualization tools such as ParaView and Tecplot, as well as image files in the format of PPM generated by a built-in post-processing function. As an important component of the framework, the I/O layer provides support for pre-processing and post-processing modules with several options such as group I/O and MPI I/O, with addition of a checkpoint and restart controller which enables fast recover from system-level or hardware fault. With the unified software framework of SunwayLB, we are able to perform long-term and stable simulations for complex geometries and phenomena [28].

C. *The Multi-Level Parallelization Approach on Sunway TaihuLight*

The way of storing all the particle populations in memory is crucial for achieving high performance and memory bandwidth. One intuitive way is to use a data structure to store particle populations and other related data for each cell resembling array of structures (AoS). As the D3Q19 model shows, when updating one cell, 19 particle populations from its neighbors will be needed, which are not continuous in memory if we adopt AoS, resulting in large amount of random memory accesses and frequent DMA startups. The overhead of DMA of this strategy is a severe limitation to the overall performance. Therefore, we apply a structure of arrays (SoA) pattern to store all the particle populations continuously in memory.

1) Domain Decomposition at the MPI Level and On-the-Fly Halo Exchange Scheme: In order to map the LBM algorithm to millions of heterogeneous cores, first we need to deal with the domain decomposition for MPEs at the MPI level. As the x or y dimension usually has less than 1000 elements, the 1D decomposition scheme cannot expose enough parallelism for 160000 MPEs. In contrast, the 3D decomposition scheme will bring much more complicated communications among different cells, thus resulting in more overhead of communications. We divide the entire domain into cuboid shaped subdomains with equal sizes, which is a 2D decomposition scheme for xy-direction. Each cuboid shaped subdomain contains the full z axis. Thus, to expose enough parallelism and reduce the communication complexity, we derived a 2D domain decomposition for the MPI level shown in Fig. 5(1) and an efficient scheme of on-the-fly halo exchange shown in Fig. 6. As each MPI process needs to communicate with up to 8 neighbors to update local lattice cells, our original implementation is to offload the whole computation domain to CPE cluster after halo exchange is performed by MPE, which, however, does not make the most of the parallelism mechanism of MPE and CPE cluster. Thus, we decompose the computation domain of each process into the halo and inner parts to separate halo update and propagation-collision part. By utilizing non-block communications, the MPE is in charge of performing communications for halo exchange and a small part of computation task while the CPE cluster carries out collaborative computation for the inner domain simultaneously. Thus, we are able to implement a pipelining between the MPE and CPE cluster and hide almost all communication overhead by using the on-the-fly halo exchange scheme. By employing the effective domain decomposition and on-the-fly halo exchange schemes, the parallel efficiency of our code can reach 94% when scaling up to 10.4 million cores as our weak scaling experiment shows in the following section. In addition, the on-the-fly halo exchange scheme can improve the overall performance by approximately 15%.

2) Data Blocking and Sharing at the Level of CPEs: As the CPE has a three-level memory hierarchy, we follow the REG-LDM-MEM performance model to optimize the kernels of SunwayLB. Therefore, all data have to be copied into the 64KB LDM of each CPE through DMA. One CPE cluster is composed of 64 CPEs, so the total LDM size of one CPE

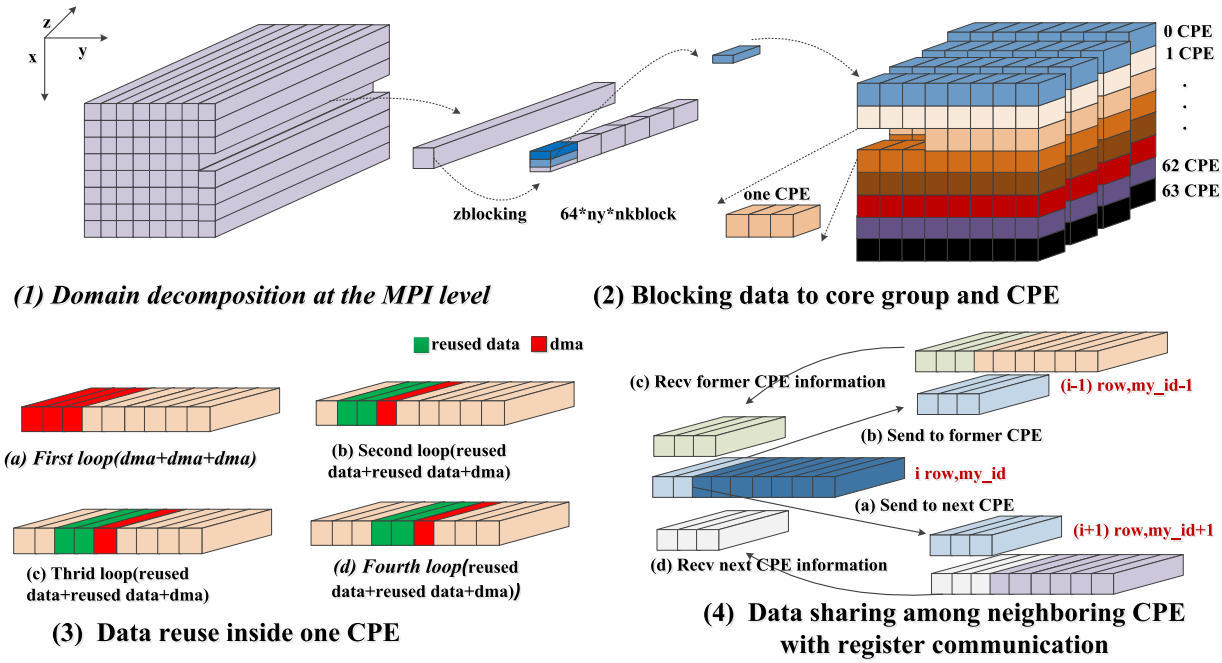

Fig. 5. Multi-level parallelization approach and data sharing scheme. (1) Domain decomposition at the level of MPI; (2) Blocking data to core group and CPE. As the data are too large to fit in a core group's LDM, we first block the data along z axis, then divide them into 64 parts for 64 CPE; (3) Data reuse inside one CPE. Inside one CPE, we update the status of particles along the x axis. For the first time through the loop, all data are transferred to LDM through DMA, while for the following loops we keep most of the data inside the LDM using Athread interfaces and only copy a small part of necessary data into LDM to continue to update the status of particles, so we can save substantial time for transferring data from memory to LDM; (4) Data sharing among neighboring CPE with register communication. The particle update also needs data of y axis, we employ register communication to share data among neighboring CPEs along the y axis instead of copying data from memory because register communication is more efficient.

cluster is 4MB, which is far less than the size of data in one MPI process. Thus, a further blocking plan for CPE clusters is necessary. In order to maximize the DMA bandwidth and reduce the times of calling the DMA interface, we need to copy as much continuous block size as possible into the LDM. As the data is consecutive along the z axis, we block data as $64*3*70$ in x-y-z axes respectively for each CPE cluster, so that each CPE owns a $3*70$ block in the z-x plane. Due to the stencil-like features of propagation kernel, each CPE thread needs neighboring data blocks to perform local propagation. Fig. 5(3) shows our data reuse strategy along the x axis. In the meanwhile, as for the y direction, we observe an opportunity of data sharing among the neighboring CPEs as shown in Fig. 5(4). Comparing with accessing main memory, a more efficient way is to employ the register communication mechanism to share data among the CPEs. The fine-grained blocking scheme enables efficient data communication and sharing at the level of CPE. By implementing the data blocking and sharing scheme on the CPE cluster, we are able to map the most time-consuming kernels to the CPE cluster and benefit from the aggregated memory bandwidth and computational power of the whole core group, providing more than 75 times speedup.

3) Fusing Kernels With Different Performance Constraints: The propagation and collision steps are the most time-consuming kernels of LBM code. In each time step, 19 particle populations need to be loaded from and stored to neighboring cells for propagation. Thus, in our implementation, a total amount of 380 bytes including write allocate cache need to be fetched to LDM and written back to main memory to update one fluid cell and hardly any floating-point operations are conducted. In our largest tests, each core group contains 35 million cells, resulting in a total of 12 GB data transferred between main memory and LDM for one time step. Consequently, the propagation kernel is heavily memory-bound. While the collision kernel is computed for each lattice cell after the propagation step, performing most of the floating-point mathematical operations. Comparing with the propagation kernel, in the collision kernel each cell only uses density data associated with itself, which makes it fully parallelizable.

We observe different performance constraints for propagation and collision kernels. Thus, with the asynchronous memory access ability provided by IMPE and the DMA operations with low latency which can be launched by each CPE, it is possible to map the kernels to different elements of the core group and execute the kernels simultaneously so as to overlap the computation and DMA. To achieve that, we choose the ping-pong buffering or A-B pattern scheme [22] for memory layout, which means two copies of particle populations are stored in memory that can be read and written by multiple kernels in turn, as Fig. 7 shows. With the A-B pattern memory layout, we are able to fuse the propagation and collision into the same loop and map the time-consuming operations to different elements, thus overlapping the collision and propagation kernels.

Furthermore, as we employ a multi-level parallelization approach (Section VI-B), a total of 12 and 2 DMA operations for data transfer between main memory and LDM in one time step

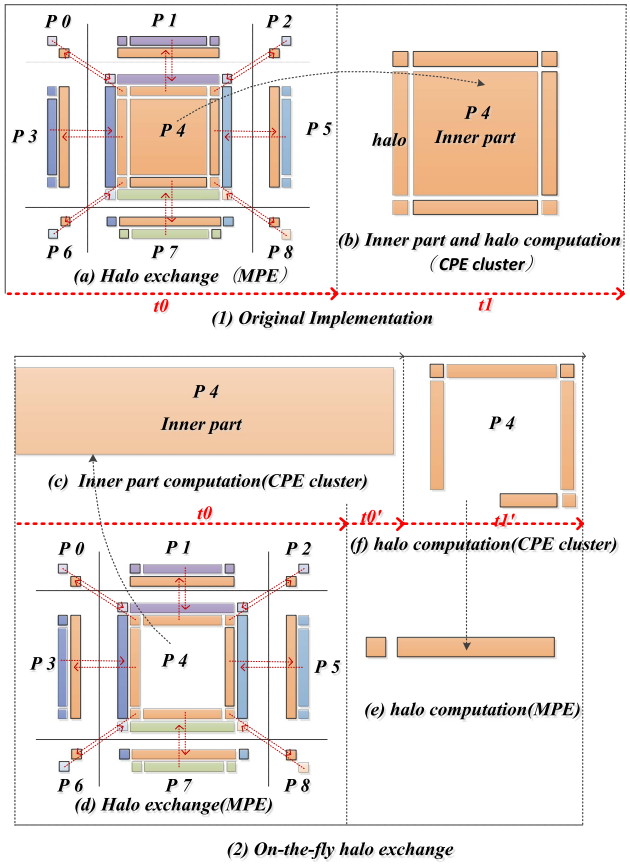

Fig. 6. (1) In the original implementation, halo exchange and kernel computation are executed by MPE and CPE cluster in sequence. (2) We propose an on-the-fly halo exchange scheme to pipeline the MPE and CPE cluster and hide almost all communication overhead. MPE can offload the inner part to the CPE cluster and perform halo exchange simultaneously, and then finish the halo computation with CPE cluster together.

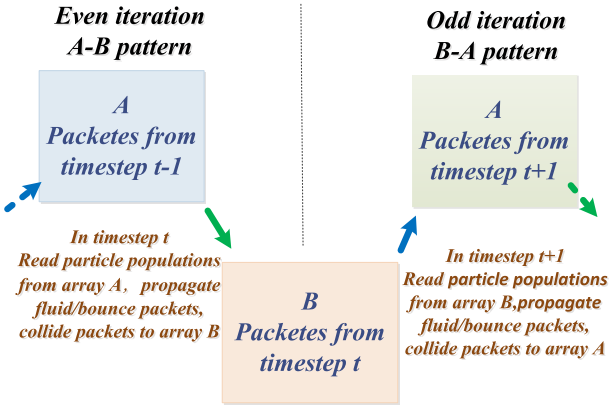

Fig. 7. Execution patter of A-B memory layout.

have to be initiated for propagation and collision respectively. With the strategy of fusion, we can reuse data between kernels and reduce 4 DMA operations in one time step, thus providing further performance improvement. The strategy of kernel fusion brings around 30% performance boost.

4) Assembly Code Level Optimizations: To achieve the maximum performance, we rewrite the kernels with assembly language using manual loop unroll and instruction scheduling techniques to enable highly efficient utilization of the pipelines and 256-bit vectorization instructions of CPE.

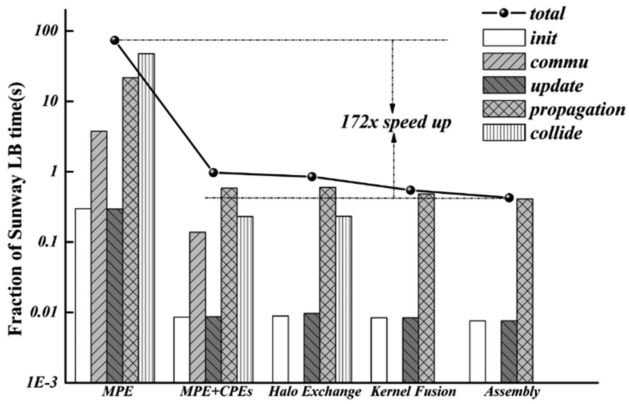

Fig. 8. Speedup of major kernels of SunwayLB as we perform our optimization strategies on Sunway TaihuLight. The columns represent the different kernels in one time step. The collision part is totally hidden because of our kernel fusion strategy. The elapsed time for one single time step reduces from 73.6 s to 0.426 s, so we finally achieve 172x speedup. These results are based on the largest direct numerical simulation for external flow of cylinder with Re = 3900.

We present results of all the optimizations in Fig. 8. With the redesign and optimization schemes implemented, we are able to achieve a 172 times speedup for one time step.

D. Parallelization and Optimization Strategies on the New Sunway System

As mentioned in Section III, the new Sunway supercomputer is the successor of Sunway TaihuLight with more computation power and new features, which give us an opportunity to further improve the performance of SunwayLB.

The resemblance between the two systems allows us to adopt similar kernel fusion and domain decomposition strategies as introduced in Section IV-C. We mainly focus on exploiting communication performance and memory bandwidth.

1) Communication Scheme: We have designed a more efficient communication scheme to perform data exchange among adjacent subdomains. Due to the particulate nature of LBM, subdomains assigned to each CG need to exchange data such as probability densities with neighboring subdomains. The memory capacity and LDM capacity of SW26010-Pro has been significantly increased compared with SW26010, as a result, the communication scheme introduced in Section IV-C is not applicable on the new Sunway supercomputer. In order to store and update the data from the adjacent subdomains, we put a single layer of cells which we call halo cells around the existing subdomains as shown in Fig. 9(1). The cells are classified as inner cells, which can be updated with cells within the subdomain, and boundary cells, which are at the outermost of the subdomain and need data of halo cells to perform the propagation and collision operations.

In our implementation, each MPI process is required to establish communication with up to eight neighboring nodes in order to update the local lattice cells. A straightforward

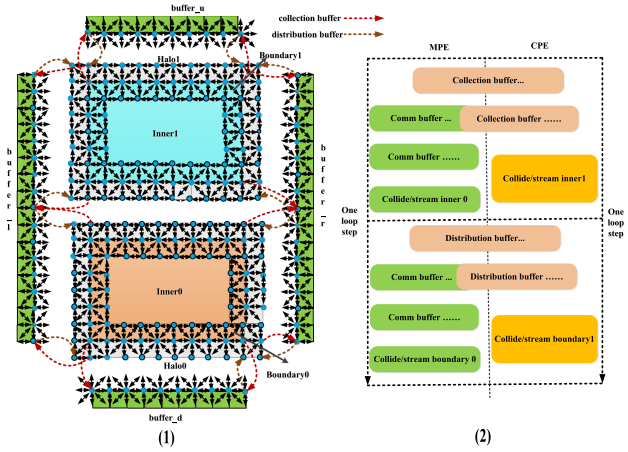

Fig. 9. (1) Halo exchange scheme. The inner cells assigned to MPE and CPE cluster are marked as Inner0 colored in brown and Inner1 colored in blue respectively. The boundary cells are a layer of cells around the inner cells. The halo cells are colored in white. The green cells are the communication buffer. (2) A highly efficient MPE-CPE collaboration scheme. The MPE is in charge of performing communications for halo exchange and a small part of computation task while the CPE cluster launches DMA to transfer data from memory to its LDM and carries out collaborative computation for the inner domain simultaneously. Most of the communication overhead is overlapped with the computation of inner cells.

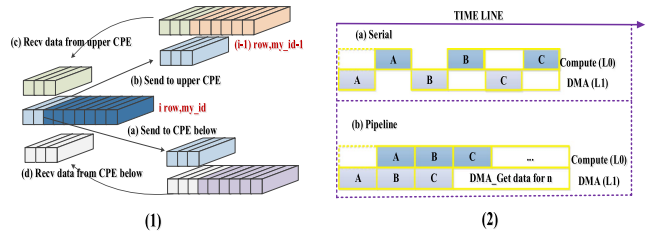

Fig. 10. (1) Data sharing among neighboring CPE using RMA. The particle update needs data of y axis, we employ RMA to share data among neighboring CPEs along the y axis instead of copying data from memory because RMA is more efficient. (2) Pipelining scheme. L0 and L1 are the two pipelines of CPE. A, B and C represent different segments of code. The original execution order is performing DMA to load data from memory to LDM and then starting computation as depicted in the upper subfigure. We carefully reorder the program stream and employ asynchronous DMA to prefetch data for following computation to realize pipelining.

approach involves initiating a halo exchange operation through MPE. Once the halo cells have been updated with data from adjacent sub-domains, we proceed to offload data to the CPE mesh sequentially and carry out propagation/collision operations. However, this implementation fails to fully exploit the parallelism inherent in the heterogeneous architecture.

To address this limitation, we have adopted a collaborative processing scheme, as illustrated in Fig. 9(2), which effectively balances the computational workload between MPE and CPEs. Leveraging non-blocking communication techniques, the MPE is responsible for managing communications related to halo exchange and a smaller portion of computational tasks. Meanwhile, the CPE cluster initiates DMA operations to efficiently transfer data from memory to its Local Data Memory (LDM) and conducts collaborative computations for the inner domain in parallel. This approach allows us to overlap most of the communication overhead with the computation of inner cells.

The computation for boundary cells is also distributed between MPE and CPEs concurrently following the completion of communication and computation for inner cells. Through the implementation of this balanced multi-level domain decomposition and halo exchange scheme, we have achieved linear parallel efficiency, as demonstrated by our weak scaling experiment, which involved scaling up to 3.9 million cores, as elaborated upon in the subsequent section.

2) Data Sharing and Pipeline Optimization: Within the CPE cluster of SW26010-Pro, a mesh network and a low-latency communication mechanism referred to as Remote Memory Access (RMA) have been integrated to facilitate rapid data communication and sharing among CPEs. RMA offers support for various data transfer modes, encompassing peer-to-peer (P2P) data transfers between two CPEs, as well as data broadcasting to specific rows or columns within the CPE mesh.

As previously mentioned, CPEs are constrained by the limited Local Data Memory (LDM) capacity and can only process a small portion of the domain at any given time. When addressing boundary cells within this small segment, multiple DMA operations are necessitated to transfer data from neighboring cells stored in the main memory to the LDM of the CPEs, which can be a time-consuming process. An opportunity arises to optimize data sharing among adjacent CPEs, as depicted in Fig. 10(1). Capitalizing on the robust bandwidth offered by the CPE mesh network, we leverage RMA to facilitate the sharing of boundary data between neighboring CPEs. This approach proves significantly more efficient than accessing the main memory through DMA. Moreover, we employ the non-blocking P2P RMA interface, enabling the concurrent execution of data transfers and computations.

To further enhance performance, we make the most of the dual pipelines within the CPEs. Each CPE encompasses two operation execution pipelines, denoted as L0 and L1. L0 primarily manages scalar and vector computing operations, spanning both floating-point and integer types, while L1 is responsible for load/store operations and RMA functions. We meticulously optimize program sequencing, as illustrated in Fig. 10(2), to fully pipeline L0 for floating-point computations and L1 for data transfers.

E. Optimization Techniques Realized on a GPU Cluster

As GPUs have become a common component of today's top supercomputers, we also port SunwayLB to a GPU cluster. Each node of the GPU cluster contains 2 Intel Xeon 6248R CPUs and 8 NVIDIA GEFORCE RTX 3090 GPUs.

We first refactor all time-consuming functions of our code with CUDA (Compute Unified Device Architecture) API, which gives direct access to GPU's computational elements. To improve the bandwidth utilization, we take advantage of pinned memory. If a source or destination of a cudaMemcpy() in the host memory is not allocated in pinned memory, it needs to be first copied to a pinned memory, which causes an extra overhead. Therefore, we can avoid this extra step and extra overhead by utilizing pinned memory directly. We use the cudaMallocHost()

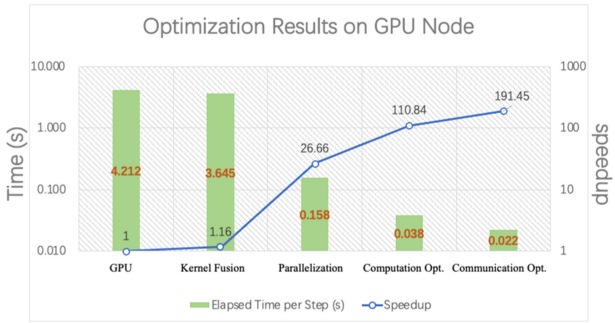

Fig. 11. Optimization results on GPU node. The columns represent the elapsed time per time step in seconds with different optimization techniques realized. "CPU" stands for the basic MPI version run on one CPU socket. "Kernel Fusion" means propagation and collision are fused into one kernel. "Parallelization" refers to performing domain decomposition and offloading major kernels to the GPUs with utilization of pinned memory. "Computation Opt." stands for reducing the computation overhead. "Communication Opt." refers to the communication optimization scheme with NCCL interfaces.

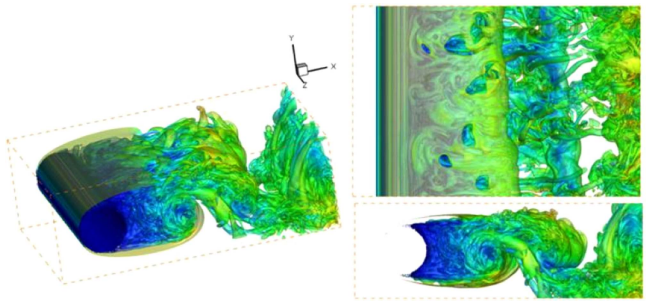

Fig. 12. Direct numerical simulation of instantaneous isosurface of Q-Criterion for flow past circular cylinder scenario at Re = 3900r following computation to realize pipelining.

for data copy to GPU memory, which much is faster since we allocate the host memory source in pinned memory and no extra copy is needed. After we manage to port all kernels to the GPU device and make use of the pinned memory, the GPU resources are effectively utilized, providing a speedup of 200x over the CPU version (1 CPU core + 1 GPU vs. 1 CPU core).

AS the D3Q19 model assigns 19 probability densities f_i ($\vec{x}$, t), i = 1 …,19 to the center of each cell, the data structure contains a Q dimension representing the 19 directions of probability. We block data in z, y, x dimension and move the loop of Q dimension into the kernel function to achieve the best utilization of CUDA thread block, which is the multiples of 32. Furthermore, we have undertaken a series of optimizations geared towards curtailing both computational overhead and the frequency of floating-point divisions. This is particularly essential due to the absence of dedicated hardware instructions for division in GPUs. To address this, we have identified recurring operations within the kernels, such as square and division, and preemptively computed their results. This enables the reuse of these precomputed values throughout the loop, eliminating the need for redundant calculations and thereby reducing the overall computational overhead.

As we adopt MPI+CUDA to implement high-performance multi-GPU parallelization, in order to update the boundary cells of adjacent subdomains, firstly the data need to be transferred from the GPU memory to main memory, then sent to the another MPI process, and transferred to the memory of corresponding GPU at last. We observe an opportunity to reduce the overhead of data movement and communication. We replace the MPI functions with NVIDIA Collective Communication Library (NCCL) functions for intra-node data exchange. NCCL implements multi-GPU communication primitives optimized for NVIDIA GPUs to achieve high bandwidth and low latency, and most importantly the data is transferred between GPUs directly so we avoid writing back to main memory.

We also illustrate the performance improvement on GPU nodes in Fig. 11. Each GPU node computes a subdomain. We use the basic MPI version run on one CPU socket as the baseline. The results show that we effectively exploit the excellent computation capacity and memory bandwidth of GPUs by implementing tailored optimization techniques, achieving 191x speedup and 83.8% memory bandwidth utilization.

V. Results

In this section, we illustrate the computational performance of SunwayLB by simulating direct numerical simulation and a number of computational experiments for extreme-large scale fluid flow, as examples of real-world applications. The performance in terms of Flops is measured by a performance monitor on Sunway TaihuLight called PERF. Another more popular way to measure the performance of LBM software is in GLUPS or MLUPS, meaning "billion lattice cell updates per second" and "million lattice cell updates per second" respectively, which can be calculated as (2), where P is the performance measured in LUPS, M is the total number of the lattice cells and t_s is the elapsed time for a single time step.

$$P = M/t_s \tag{2}$$

A. Performance Results

1) Direct Numerical Simulation for External Flow of Cylinder With Re = 3900: External flow around cylinder is one of most popular benchmarks for checking the CFD solver's capability of capturing turbulent phenomena. We have conducted a series of experiments based on this benchmark with Re = 3900 to demonstrate the weak scaling and strong scaling results. For the largest simulation, we achieve an unprecedented high simulation speed of 11245 GLUPS and 4.7 PFlops with a mesh size of 40000 × 40000 × 3500, corresponding to 5.6 trillion lattice cells in total. The mesh size of our simulation is two times larger than the previous largest scale DNS conducted by Lee et al. [13], [23] on Mira.

As Fig. 12 shows, our DNS results can depict both large vortex structures and complex small vortex tube structures.

2) Scaling Results on Sunway TaihuLight: The weak scaling results tested on Sunway TaihuLight are demonstrated in Fig. 13. For weak scaling tests, each CG contains a block size of 500 by 700 by 100, and scale gradually to 160000 MPI processes, resulting in 5.6 trillion cells for the largest run, achieving 11245 GLUPS and a sustained performance 4.7 PFlops.

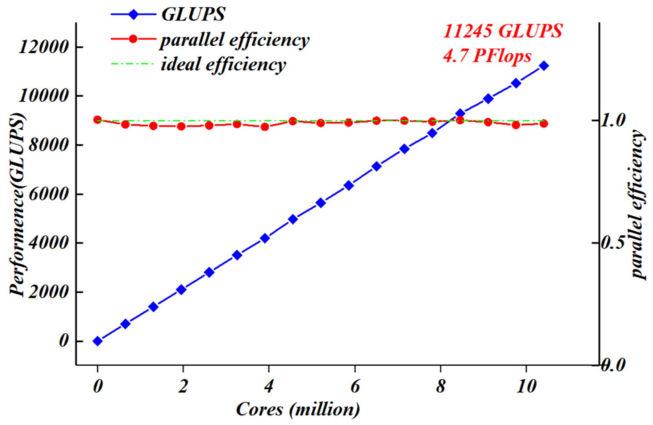

Fig. 13. Performance and parallel efficiency of weak scaling, scaling from 65 cores (1 CG) to 10400000 cores (16000 CGs).

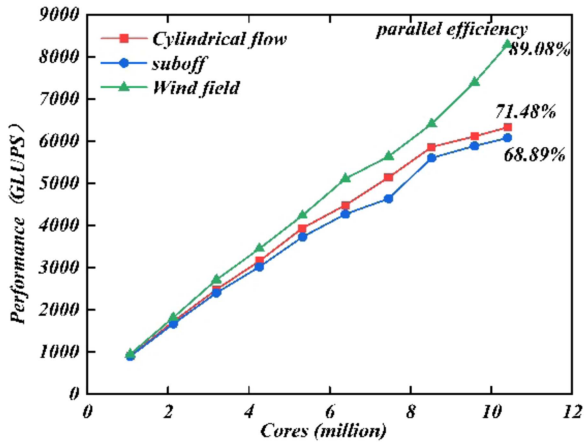

Fig. 14. Strong scaling results of three different simulations, including external flow around cylinder, DARPA Suboff and wind flow simulation in complex urban areas, scaling from 1064960 cores to 10400000 cores.

Due to the memory-bound nature of LBM as shown in the previous sections, we can calculate the upper bound of the overall performance according to the roofline model [17]. In our implementation, a total amount of 380 bytes needs to be transferred between the main memory and LBM to update one each lattice cell. Thus, with a maximum DMA bandwidth of 32 GB/s for one core group, an upper bound of performance of one core group can be estimated as follows:

$$32\frac{GB}{s} \div 380\frac{B}{LUP} = 90.4\text{MLUPS}$$

Hence, the upper bound of performance for 160000 core groups is 14464 GLUPS. According to the measured performance of 11245 GLUPS, we have reached 77% of theoretical maximum performance. As LBM is memory bound, the memory bandwidth utilization is equivalent to the ratio of theoretical maximum performance that we can achieve, which can be calculated as follows:

$$\frac{11245 * 10^9 LUP/S * 380B/LUP}{32 * 1024^3 B/s * 160000} = 77\%$$

Thus, we manage to achieve better memory bandwidth than the state-of-the-art LBM simulation performed on JUQUEEN and Piz Daint (67.4% and 69% respectively).

We choose a case with mesh size of 10000∗10000∗5000 to perform strong scaling tests. Fig. 14 shows that our implementation achieved 71.48% parallel efficiency when scaling to 10400000 cores. The results of other two experiments are also revealed in Fig. 14.

3) Scaling Results on the New Sunway Supercomputer: We utilize 10000 SW26010-Pro processors for weak and strong scaling tests. We conduct simulations of flow around a circular cylinder to evaluate the scalability and performance. Each CG contains a block size of 1000∗700∗100, and scale from 6000 MPI processes gradually to 60000 MPI processes (60000 CGs, 3900000 cores), resulting in 4.2 trillion cells for the largest run, achieving 6583 GLUPS, 81.4% memory bandwidth utilization and a sustained performance 2.76 PFlops. The results are presented in Fig. 15.

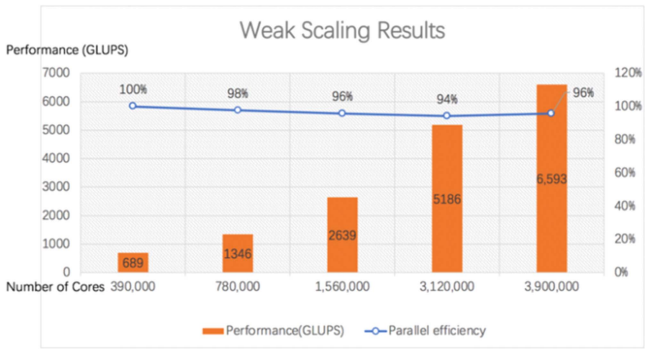

Fig. 15. Performance and parallel efficiency of weak scaling, starting from 390000 cores (6000 core group) to 3900000 cores (60000 core groups) on the new Sunway Supercomputer.

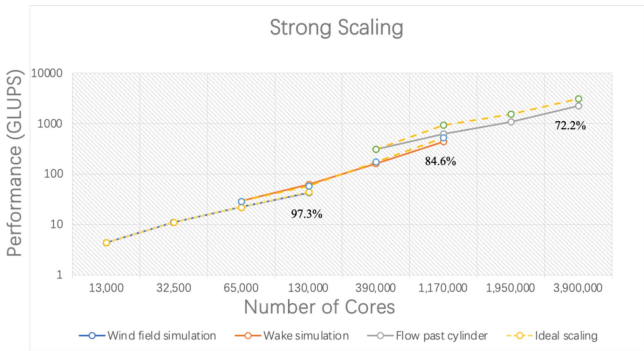

Fig. 16. Strong scaling results of three different simulations on the new Sunway Supercomputer, including wind field simulation, wake simulation and flow past cylinder simulation, scaling from 13000 cores to 130000 cores, 65000 cores to 1170000 cores and 390000 cores to 3900000 cores respectively.

We adopt three different cases for strong scaling tests: wind field simulation, wake simulation and flow past cylinder simulation. The total numbers of lattice cells are fixed as 4000∗4000∗1000, 200000∗1000∗1500, and 10000∗7000∗5000 respectively. Fig. 16 shows the strong scaling results of three

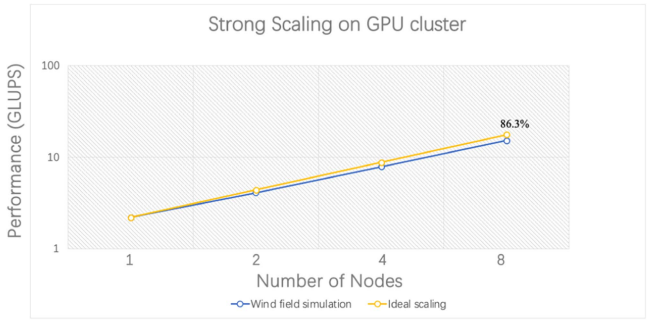

Fig. 17. Strong scaling results on the GPU cluster. We perform an experimental wind field simulation scaling from one node to eight nodes.

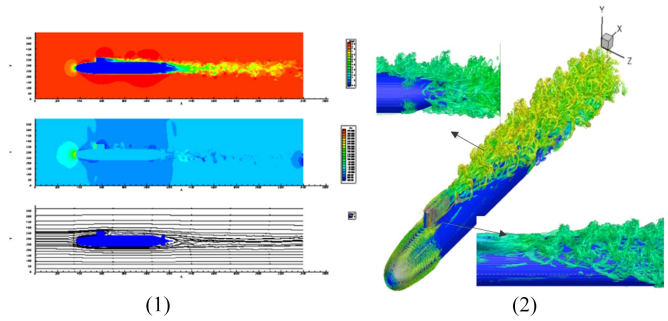

Fig. 18. (1) Instantaneous contour of velocity field, pressure field and streamlines with flow past Suboff. (2) Instantaneous isosurface of Q-Criterion with flow past Suboff.

cases. The largest run of flow past cylinder simulation achieves 72.2% parallel efficiency when scaling to 3900000 cores.

4) Scaling Results on the GPU Cluster: We conduct an experimental wind field simulation on 8-node GPU cluster with 8 GPUs on each node. The total number of lattice cells is 1400∗2800∗100. As shown in Fig. 17, we use one node as the basis and scale to 8 nodes with 64 GPUs in total, achieving a strong scaling efficiency of 86.3%.

B. Simulation of DARPA Suboff

To show the capability of solving engineering problem by SunwayLB, we simulate the submarine model of Suboff provided by DARPA to capture the turbulent flow surround the submarine model. It could be used to analyze the resistance, maneuverability, and the noise generated by the submarine [27]. We have scaled the DARPA Suboff benchmark to the TaihuLight system and the new Sunway supercomputer, achieving parallel efficiency of 68.89% and 84.6% as demonstrated in Figs. 14 and 16 respectively. The simulation results are shown in Fig. 18.

C. Simulation of Wind Flow in Complex Urban Areas

Understanding the details of the wind flow in urban areas offers us a possible way to improve wind energy use in turbulent urban wind environment to meet the energy requirement in these areas [20], [21]. The previous largest LBM based LES wind simulation for urban areas is conducted by Onodera et al. [19]

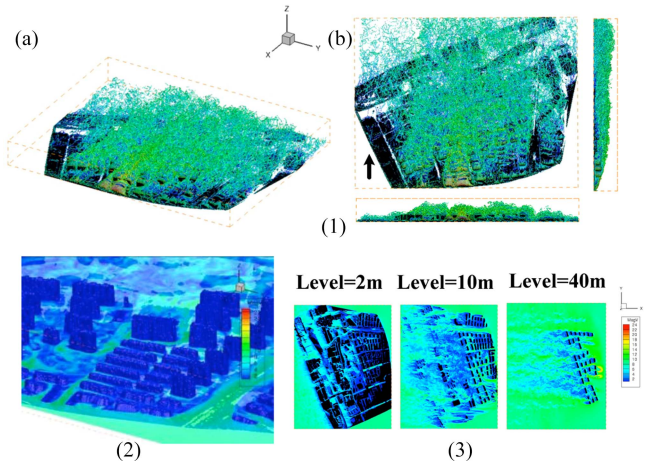

Fig. 19. Velocity of the wind at inlet is set to 8m/s and the highest building is about 80m. (1) Instantaneous isosurface of Q-Criterion with flow over urban area. It could be observed through the Q-Criterion isosurface that the affected region reaches a vertical height of 160m. (2) Details of velocity field in the simulated area. The flow surrounding the building could be identified clearly. (3) Instantaneous contour of velocity field at different height from ground.

on 4032 GPUs of TSUBAME 2.0 with a sustained performance of around 149 TFlops. The simulation area is 10km × 10km, with a 10080 × 10240 × 512 (53 billion cells in total) mesh and a horizontal resolution of 1 meter. We choose an area located in north of Shanghai city with a size of 1km × 1km for our simulation. The mesh size is 11511 × 14744 × 1600 (271 billion cells in total), providing a horizontal resolution of 0.1 meter. We achieve an excellent parallel efficiency of 89%, more than 8000 GLUPS with 10.4 million cores and 0.054 second per time step. The simulation details are presented in Fig. 19.

VI. Conclusion

In this paper, we report our work on designing and building SunwayLB, a highly scalable Lattice Boltzmann Method framework for three representative heterogeneous systems: Sunway TaihuLight, the new Sunway supercomputer and a GPU cluster. In order to map the LBM algorithm to the millions of heterogeneous cores of Sunway supercomputers and alleviate the memory constraint from both the hardware and the LBM algorithm, we apply several optimization schemes including a customized multi-level domain decomposition and data sharing scheme, a carefully orchestrated strategy to fuse kernels with different performance constraints for a more balanced workload, and assembly code optimization strategies such as vectorization and instructions reordering. To evaluate the portability and performance of SunwayLB, we also implement a GPU version based on SunwayLB with different optimization techniques, including utilization of the pinned memory, pre-computation of high-overhead operations, efficient communication strategy using NCCL, etc.

A series of experiments covering different scientific domains are performed to illustrate the computational performance and simulation results. For the first time, direct numerical simulations contain up to 5.4 trillion lattice cells are conducted. We

achieve linear parallel efficiency and 77% memory bandwidth utilization when scaling up to more than 10.4 million cores on Sunway TaihuLight and a simulation speed of 11245 GLUPS, corresponding to 4.7 PFlops.

The results show that GPUs deliver high performance for LBM-based simulation due to its excellent memory bandwidth and computation capacity, while the Sunway supercomputers are suited for large-scale parallelization. With the excellent scalability and performance measured, we believe these techniques have great potential to be applied to other applications and heterogeneous architectures, especially the memory-bound applications. With the excellent simulation speed and capability to perform extreme-large scale simulations provided by SunwayLB, a practical LBM based DNS/LES solution for large scale industrial applications is becoming a reality.

REFERENCES

[1] G. R. McNamara and G. Zanetti, "Use of the Boltzmann equation to simulate lattice-gas automata," *Phys. Rev. Lett.*, vol. 61, no. 20, 1988, Art. no. 2332.

[2] Y. Qian, D. d'Humières, and P. Lallemand, "Lattice BGK models for Navier–Stokes equation," *EPL(EurophysicsLetters)*, vol. 17, no. 6, 1992, Art. no. 479.

[3] P. Lallemand and L. Luo, "Theory of the lattice Boltzmann method: Dispersion, dissipation, isotropy, Galilean invariance, and stability," *Phys. Rev. E*, vol. 61, no. 6, pp. 6546–6562, 2000.

[4] L. Luo, "Unified theory of lattice Boltzmann models for nonidealgases," *Phys. Rev. Lett.*, vol. 81, no. 8, pp. 1618–1621, 1998.

[5] J. Smagorinsky, "General circulation experiments with the primitive equations," *Mon. Wea. Rev.*, vol. 91, pp. 99–164, 1963.

[6] M. Hasert et al., "Complex fluid simulations with the parallel tree-based Lattice Boltzmann solver Musubi," *J. Comput. Sci.*, vol. 5, no. 5, pp. 784–794, 2014.

[7] C. Feichtinger, J. Habich, H. Köstler, U. Rüde, and T. Aoki, "Performance modeling and analysis of heterogeneous lattice boltzmann simulations on CPU–GPU clusters," *Parallel Comput.*, vol. 46, pp. 1–13, 2015.

[8] B. Jelinek, M. Eshraghi, S. Felicelli, and J. F. Peters, "Large-scale parallel lattice Boltzmann–cellular automaton model of two-dimensional dendritic growth," *Comput. Phys. Commun.*, vol. 185, no. 3, pp. 939–947, 2014.

[9] C. Riesinger, A. Bakhtiari, M. Schreiber, P. Neumann, and H. J. Bungartz, "A holistic scalable implementation approach of the lattice boltzmann method for CPU/GPU heterogeneous clusters," *Computation*, vol. 5, no. 4, 2017, Art. no. 48.

[10] F. Schornbaum and U. Ru¨de, "Massively parallel algorithms for the lattice Boltzmann method on nonuniform grids," *SIAM J. Sci. Comput.*, vol. 38, no. 2, pp. C96–C126, 2016.

[11] C. Godenschwager, F. Schornbaum, M. Bauer, H. Köstler, and U. Rüde, "A framework for hybrid parallel flow simulations with a trillion cells in complex geometries," in *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.*, Nov. 2013, Art. no. 35.

[12] D. Groen, J. Hetherington, H. B. Carver, R. W. Nash, M. O. Bernabeu, and P. V. Coveney, "Analysing and modelling the performance of the HemeLB lattice-Boltzmann simulation environment," *J. Comput. Sci.*, vol. 4, no. 5, pp. 412–422, 2013.

[13] M. Lee, N. Malaya, and R. D. Moser, "Petascale direct numerical simulation of turbulent channel flow on up to 786k cores," in *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.*, Nov. 2013, Art. no. 61.

[14] M. Krause, T. Gengenbach, and V. Heuveline, "Hybrid parallel simulations of fluid flows in complex geometries: Application to the human lungs," in *Euro-Par 2010 Parallel Processing Workshops*, Berlin, Germany: Springer, 2011, pp. 209–216.

[15] J. Fietz, M. J. Krause, C. Schulz, P. Sanders, and V. Heuveline, "Optimized hybrid parallel lattice Boltzmann fluid flow simulations on complex geometries," in *Proc. Eur. Conf. Parallel Process.*, 2012, pp. 818–829.

[16] J. Fang, H. Fu, W. Zhao, B. Chen, W. Zheng, and G. Yang, "swDNN: A library for accelerating deep learning applications on sunway TaihuLight," in *Proc. IEEE Int. Conf. Parallel Distrib. Process. Symp.*, 2017, pp. 615–624.

[17] S. Williams, A. Waterman, and D. Patterson, "Roofline: An insightful visual performance model for multicore architectures," *Commun. ACM*, vol. 52, no. 4, pp. 65–76, 2009.

[18] J. Götz, K. Iglberger, M. Stürmer, and U. Rüde, "Direct numerical simulation of particulate flows on 294912 processor cores," in *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.*, 2015, pp. 1–11.

[19] N. Onodera, T. Aoki, T. Shimokawabe, and H. Kobayashi, "Large-scale LES wind simulation using lattice Boltzmann method for a 10 km$\times$ 10 km area in metropolitan Tokyo," *TSUBAME e-Sci. J. Glob. Sci. Inf. Comput. Center*, vol. 9, pp. 1–8, 2013.

[20] T. F. Ishugah, Y. Li, R. Z. Wang, and J. K. Kiplagat, "Advances in wind energy resource exploitation in urban environment: A review," *Renewable Sustain. Energy Rev.*, vol. 37, pp. 613–626, 2014.

[21] A. Gagliano, F. Nocera, F. Patania, and A. Capizzi, "Assessment of micro-wind turbines performance in the urban environments: An aided methodology through geographical information systems," *Int. J. Energy Environ. Eng.*, vol. 4, no. 1, 2013, Art. no. 43.

[22] P. Bailey, J. Myre, S. D. C. Walsh, D. J. Lilja, and M. O. Saar, "Accelerating lattice boltzmann fluid flow simulations using graphics processors," in *Proc. Int. Conf. Parallel Process.*, Vienna, Austria, 2009, pp. 550–557.

[23] M. Lee, R. Ulerich, N. Malaya, and R. D. Moser, "Experiences from leadership computing in simulations of turbulent fluid flows," *Comput. Sci. Eng.*, vol. 14, pp. 1521–9615, 2014.

[24] Y. Kono, K. Sano, and S. Yamamoto, "Scalability analysis of tightly-coupled FPGA-cluster for lattice boltzmann computation," in *Proc. 22nd Int. Conf. Field Program. Log. Appl.*, 2012, pp. 120–127.

[25] G. Crimi, F. Mantovani, M. Pivanti, S. F. Schifano, and R. Tripiccione, "Early experience on porting and running a Lattice Boltzmann code on the Xeon-Phi co-processor," *Procedia Comput. Sci.*, vol. 18, pp. 551–560, 2013.

[26] J. Tölke and M. Krafczyk, "TeraFLOP computing on a desktop PC with GPUs for 3D CFD," *Int. J. Comput. Fluid Dyn.*, vol. 22, pp. 443–456, 2008.

[27] N. Chase, "Simulations of the DARPA Suboff submarine including self-propulsion with the E1619 propeller," *Masters Abstr. Int.*, vol. 51, no. 1, 2012.

[28] Z. Liu et al., "Sunwaylb: Enabling extreme-scale lattice boltzmann method based computing fluid dynamics simulations on sunway taihulight," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, 2019, pp. 557–566.

[29] H. Fu et al., "Refactoring and optimizing the community atmosphere model (CAM) on the sunway taihulight supercomputer," in *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.*, 2016, pp. 969–980.

[30] H. Fu et al., "18.9-pflops nonlinear earthquake simulation on sunway taihulight: Enabling depiction of 18-hz and 8-meter scenarios," in *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.*, 2017, Art. no. 2.

[31] M. Iwasawa et al., "Global simulation of planetary rings on sunway taihulight," in *Proc. Int. Conf. Comput. Sci.*, 2018, pp. 483–495.

[32] T. Zhang et al., "SW_GROMACS: Accelerate GROMACS on sunway TaihuLight," in *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.*, 2019, pp. 1–14.

[33] Y. Liu et al., "Closing the 'quantum supremacy' gap: Achieving real-time simulation of a random quantum circuit using a new sunway supercomputer," in *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.*, 2021, pp. 1–12.

[34] W. Hu et al., "2.5 million-atom Ab initio electronic-structure simulation of complex metallic heterostructures with DGDFT," in *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.*, 2022, pp. 48–60.

[35] H. Fu et al., "The Sunway TaihuLight supercomputer: System and applications," *Sci. China Inf. Sci.*, vol. 59, pp. 1–16, 2016.

[36] Z. Liu et al., "Sunwaylb: Enabling extreme-scale lattice boltzmann method based computing fluid dynamics simulations on sunway taihulight," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, 2019, pp. 557–566.

[37] J. Gounley, M. Vardhan, E. W. Draeger, P. Valero-Lara, S. V. Moore, and A. Randles, "Propagation pattern for moment representation of the lattice Boltzmann method," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 3, pp. 642–653, Mar. 2022.

[38] P. Valero-Lara, "Leveraging the performance of LBM-HPC for large sizes on GPUs using ghost cells," in *Proc. Int. Conf. Algorithms Architectures Parallel Process.*, 2016, pp. 417–430.

[39] P. Valero-Lara, "Reducing memory requirements for large size LBM simulations on GPUs," *Concurrency Computation: Pract. Experience*, vol. 29, no. 24, 2017, Art. no. e4221.

[40] G. Wellein, T. Zeiser, G. Hager, and S. Donath, "On the single processor performance of simple lattice Boltzmann kernels," *Comput. Fluids*, vol. 35, no. 8-9, pp. 910–919, 2006.

Zhao Liu (Associate Member, IEEE) received the bachelor's degree from the Beijing Institute of Technology, in 2008, and the master's degree in computer technology and engineering from Shanghai Jiaotong University, in 2014. He is currently working toward the PhD degree with the Department of Computer Science and Technology, Tsinghua University. His main research interests include high performance computing and computer architecture.

Xuesen Chu received the bachelor's degree from Zhejiang University, in 2003, and the master's degree in fluid mechanics from China Ship Scientific Research Center, in 2006. He is currently working toward the PhD degree with the Department of Computer Science and Technology, Tsinghua University. His main research interests include high performance computing and computational fluid dynamics.

Xiaojing Lv received the bachelor's degree from the Harbin Institute of Technology, in 2012, and the master's degree in aerospace engineering from the Harbin Institute of Technology, in 2014. She is currently working toward the PhD degree with the China Ship Scientific Reseach Center. Her main research interests include high performance computing, and applications.

Hongsong Meng received the master's degree from Information Engineering University, China. She is currently a research staff with the Department of Parallel Optimization, National Supercomputing Center in Wuxi. Her main research interests include low-level tuning, elemental math libraries, weather/climate, and MD applications.

Hanyue Liu received the bachelor's degree from Lanzhou University, in 2019. He is currently a parallel software engineer with the Department of Parallel Optimization, National Supercomputing Center in Wuxi. His main research interests include high performance computing and applications.

Guanghui Zhu received the master's degrees in software engineering from the Beijing University of Technology, in 2010, and education technology from Henan University, in 2011. He is currently a research staff with the Department of Government and Enterprise Business, Huirong Electronic Systems Engineering company Ltd. His main research interests include Big Data analysis and visulisation.

Haohuan Fu (Senior Member, IEEE) received the PhD degree in computing from Imperial College, London. He is currently a professor with the Ministry of Education Key Laboratory for Earth System Modeling, and the Department of Earth System Science, Tsinghua University, China. He is also the deputy director with National Supercomputing Center, Wuxi, China. His research interests include high-performance computing in earth and environmental sciences, computer architectures, performance optimizations, and programming tools in parallel computing. He was the recipient of ACM Gordon Bell Prize in 2016 and 2017, and Most Significant Paper Award by FPL in 2015.

Guangwen Yang received the PhD degree in computer science from the Harbin Institute of Technology. He is currently a professor with the Department of Computer Science and Technology and Beijing National Research Center for Information Science and Technology, Tsinghua University, China. He is also the director with National Supercomputing Center, Wuxi, China. His research interests include parallel algorithms, cloud computing, machine learning, and the earth system model. He was the recipient of ACM Gordon Bell Prize in 2016 and 2017.