Collaborating CPUs and MICs for Large-Scale LBM Multiphase Flow Simulations

Chuanfu Xu, Xi Wang, Dali Li, Yonggang Che, Zhenghua Wang

Collaborating CPUs and MICs for Large-scale LBM Multiphase Flow Simulations *

Chuanfu Xu, Xi Wang, Dali Li, Yonggang Che, and Zhenghua Wang

College of Computer Science, National University of Defense Technology, Changsha 410073, P.R.China

Abstract. This paper highlights the use of the OpenMP4.5 accelerator programming model to collaborate CPUs and Intel Many Integrated Cores (MIC) co-processors for large-scale LBM multiphase flow simulationson the Tianhe-2 supercomputer. To enhance the collaborative efficiency among intra-node CPUs and co-processors, we propose a flexible load balance model with heterogeneous domain decomposition for CPU-MIC task allocation, as well as asynchronous offloading to overlap operations of CPUs and multiple MICs. Tests for 3D multi-phase (liquid and gases) problem (about 100 Billion lattices) simulating drop impact with gravity effect using D3Q19 Lattice Boltzmann discretization and Shan-Chen BGK single relaxation time collision model are presented, achieving a weak parallel efficiency of above 80% in going from 128 to 2048 compute nodes.

Keywords: Heterogeneous parallel computing · Lattice Boltzmann methods · Many-core processor · OpenMP4.5 accelerator programming model.

1 Introduction

Lattice Boltzmann Methods (LBM) regard fluids as Newtonian fluids from a microscopic perspective, divide flow field into small lattices (mass points), and simulate fluid evolution dynamics through collision models (lattices collision and streaming) [1]. Currently, LBM has been increasingly used for real-world flow problems with complex geometries and various boundary conditions. Large-scale LBM simulations with increasing resolution and extending temporal range require massive high performance computing resources. It is therefore essential and practical to port LBM codes onto modern supercomputers, often featuring many-core accelerators/coprocessors (GPU, Intel MIC, or specialized ones).These heterogeneous processors can dramatically enhance the overall performance of HPC systems with remarkably low total cost of ownership and power consumption, but the development and optimization of large-scale applications are also becoming exceptionally difficult. Accelerator programming models such as OpenMP4.X [2], OpenACC and Intel Offload aim to provide performant and productive heterogeneous computing through simple compiler directives. Among them, OpenMP4.X is especially attractive since it incorporates accelerator programming with

* Supported by NSFC under Grant No. 61772542.

traditional shared memory multithreading into a unified high-level model, and supports major languages (C++, C and Fortran...) and devices (CPU, GPU, MIC, ARM, DSP...).

In this paper, we parallelize an LBM code *openlbmflow* and highlight the use of OpenMP4.5 for large-scale CPU-MIC collaboration on the Tianhe-2 supercomputer [3]. A load balance model with heterogeneous domain decomposition is proposed for CPU-MIC task allocation. We use asynchronous offloading to minimize the cost of halo exchanges and significantly overlap CPU-MIC computation/communication. Our collaborative approach achieves a speedup of up to 5.0X compared to the CPU-only approach. Tests for 3D multi-phase (liquid and gases) problem (about 100 Billion lattices) simulating drop impact with gravity effect using D3Q19 Lattice Boltzmann discretization and Shan-Chen BGK single relaxation time collision model are presented, achieving a weak scaling efficiency of above 80% in going from 128 to 2048 compute nodes.

```
#pragma omp declare target
    //declare variables and functions on MICs
#pragma omp end declare target
    //initialization on CPUs
for (n=0;n<mic_num;n++) //for multiple MICs
{
   //pre-allocate and initialize variables on MICs
   #pragma omp target device(mic_num) data map(alloc...)
   #pragma omp target device(mic_num) data map(to...)
}
for (iter=1;iter<max_iter;iter++)//time-marching loop
{
   for (n=0;n<mic_num;n++) //for multiple MICs
    {
       //gather boundary lattices into the Inbuffer (CPU)
       #pragma omp target device(mic_num) map(to...) map(from...) nowait
       //H2D the Inbuffer
       //scatter the Inbuffer and update halo lattices (MIC)
       //MIC caltulation
       //gather boundary lattices into the Outbuffer (MIC)
       //D2H the Outbuffer
    }
    //caltulation on CPUs
    #pragma omp taskwait //synchronization
    for (n=0;n<mic_num;n++) //for multiple MICs
    {
       //scatter the Outbuffer and update halo lattices (CPU)
     }
     //boundary conditions
     //MPI communication
}
```

Fig. 1. Code skeleton for CPU-MIC collaboration with asynchronous offloading and overlapping of CPU-MIC computation/communication using OpenMP directives.

2 CPU-MIC collaboration and performance results

openlbmflow is an LBM code written in C that can simulate both 2D/3D single-phase or multi-phase flow problems with periodic and/or bounce-back boundary conditions. It mainly consists of three phases: initialization, time iteration, and post-processing. During the initialization phase, the geometry of the flow field, flow density and the distribution function are initialized. The time iteration

phase includes three important procedures: inter-particle force calculation (as well as velocity and density), collision and streaming. In the post-processing phase, simulation results are collected and saved according to a user-specified iteration interval.

We decompose the original computational domain along the three dimensions evenly into many blocks and distribute them among MPI processes. On each compute node, each block is divided into 4 sub-blocks with one calculated by CPUs and the other three offloaded to the three coprocessors. Fig.1 illustrates the intra-node collaborative programming approach. Before time-marching loops, we use `omp declare target` directive to declare variables or functions which are both available on CPU and MIC (line 1-3). We use `omp target data` directive with `map` clause to pre-allocate device memory and perform initialization of global flow variables and data transfer buffers on each MIC (line 5-10). We design a unified *In/Out-buffer* for PCI-e data transfer among intra-node CPUs and coprocessors. In each iteration, boundary lattices on CPUs are gathered into the *Inbuffer*, and transferred to different MICs using `map` clause with array section syntax (line 15-17). Before MIC calculation, we scatter boundary lattices from the *Inbuffer* and update halo lattices on MICs (line 18). After MIC calculation, boundary lattices on MICs will be gathered into the *Outbuffer* and transferred back to CPUs (line 20-21). We use OpenMP `nowait` to asynchronously dispatch kernels on MIC and overlap CPU-MIC computation/communication. We synchronize CPU-MIC computation using the `taskwait` directive to ensure that both sides have finished their computations before updating halo lattices on CPUs and MPI communications. We use a parameter r to represent the workload ratio on CPU side and r can be configured by profiling *openlbmflow*'s sustainable performance on both sides.

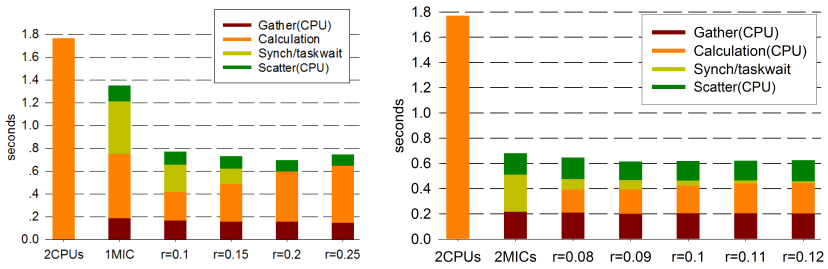

Fig. 2. Performance of CPU+1MIC (left) and 2MICs (right) with problem size $256 \times 256 \times 256$.

We use icc 17.0.1 from Intel composer 2017.1.132 in out tests. Our heterogeneous code was compiled in double precision with option "-qopenmp -O3 -fno-alias -restrict -xAVX". MPICH2-GLEX was used for MPI communications.

Fig.2(left) demonstrates the performance of CPU+1MIC with overlapping of both CPU/MIC computation and PCI-e data transfer. We decompose the costs into CPU gather/scater, CPU calculation and CPU-MIC synchronization. Due to overlapping, the synchronization cost decreases with increasing workloads on CPUs, and disappears when $r = 0.2$, indicating a perfect overlapping. Afterwards further increasing r will improve the cost of CPU calculation and degrade the overall performance. The maximum speedup was improved to about 2.5 due to the enhanced overlapping. For CPU+2MICs (Fig.2(right)), the maximum speedup is about 2.88 ($r = 0.09$), only about 15.2% enhancement compared to the CPU+1MIC simulation. This is mainly due to a relatively small total workload, and the collaborative overhead exceeds more than half of the whole execution time.

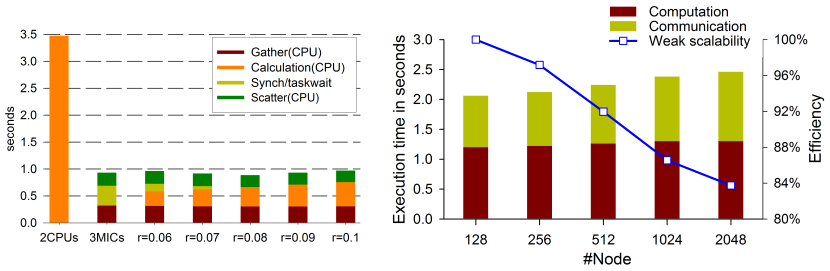

Fig. 3. Performance of CPU+3MICs with problem size of $512 \times 256 \times 256$ (left) and large-scale weak scalability on CPU+MIC nodes (right).

In Fig.3(left), the maximum speedups are 3.93 ($r = 0.08$) and 4.81 ($r = 0.07$) for the problem set $512 \times 256 \times 256$ with CPU+3MICs. Because the sustainable performance of *openlbmflow* on a MIC outperforms much of that on two CPUs, only less than 10% of the whole workload is allocated to CPUs for collaborative simulations with multiple MICs. Due to the limited device memory capacity (8GB) on Xeon Phi 31S1P, the maximum problem size for each MIC is about $256 \times 256 \times 256$. As a result, we couldn't achieve ideal load balance in heterogeneous simulations. Fig.3(right) reports the weak scalability results for CPU+MIC collaborative simulations. Although large-scale heterogeneous simulations involve quite complicated interactions, efficiencies stay well above 80%. This is comparable to that of large-scale CPU-only simulations and demonstrates the effectiveness of the overlapping optimization.

3 Related work

Few researches about parallelizing scientific codes using the new OpenMP4.X accelerator programming model on heterogeneous supercomputers are report-

ed, but many researchers have shown the experiences of porting LBM codes onto GPUs or MICs using other programming models. Paper [4] ported a GPU-accelerated 2D LBM code onto Xeon Phi, and compared with previous implementations on state-of-the-art GPUs and CPUs. Paper [5] implemented a LBM program using the portable programming model OpenCL, and evaluated its performance on multi-core CPUs, NVIDIA GPUs as well as Intel Xeon Phi. In [6], researchers have also parallelized *openlbmflow* on the Tianhe-2 supercomputer and collaborate CPUs and MICs using Intel Offload programming model. The performance was preliminary evaluated in single precision. To summarize, current reports only involve simple LBM models on small MIC clusters. Paper [7] Collaborated CPU and GPU for large-scale high-order CFD simulations with complex grids on the TianHe-1A supercomputer. This is the first paper, to our best knowledge, reporting CPU-MIC collaborative LBM simulations using complex 3D multi-phase flow models with OpenMP4.5.

4 Conclusions

In this paper, we developed a CPU+MIC collaborative software *openlbmflow* for 3D Lattice Boltzmann multiphase flow simulations on the Tianhe-2 supercomputer based on the new OpenMP accelerator programming model. The software successfully simulated a 3D multi-phase (liquid and gases) problem (100 billion lattices) using D3Q19 and Shan-Chen BGK models on 2048 Tianhe-2 nodes, demonstrating a highly efficient and scalable CPU+MIC collaborative LBM simulation with a weak scaling efficiency of above 80%. For future work, besides fine tuning of the software, we are planning to port *openlbmflow* onto China's self-developed many-core processors/coprocessors based on the power-efficient high performance ARM architecture.

References

1. Succi, S., Benzi, R., et al: The lattice Boltzmann equation: A new tool for computational fluid-dynamics. Physica D: Nonlinear Phenomena **47**, 219–230 (1991)
2. Martineau, M., Price, J., et al: Pragmatic Performance Portability with OpenMP 4.x. In: 12th International Workshop on OpenMP, pp. 253–267 (2016)
3. Xiangke, L., Liquan, X., Canqun, Y.: MilkyWay-2 supercomputer: system and application. Front. Comput. Sci. **8**(3), 345–356 (2014)
4. Crimi, G., Mantovani, F., Pivanti, M., Schifano, S. F., Tripiccione, R.: Early experience on porting and running a Lattice Boltzmann code on the Xeon-Phi coprocessor. Procedia Computer Science **18**, 551–560 (2013)
5. McIntosh-Smith, S., Curran, D.: Evaluation of a performance portable lattice Boltzmann code using OpenCL. In: International Workshop on OpenCL, pp. 1–12 (2014)
6. Dali, L., Chuanfu, X., Yongxian, W., Zhifang, S., et al: Parallelizing and optimizing large-scale 3D multi-phase flow simulations on the Tianhe-2 supercomputer. Concurrency and Computation: Practice and Experience **28**, 1678–169 (2015)
7. Chuanfu, X., Xiaogang, D., Lilun, Z., et al: Collaborating CPU and GPU for large-scale high-order CFD simulations with complex grids on the TianHe-1A supercomputer. Journal of Computational Physics **278**, 275-C297 (2014).