2019 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)

swMD: Performance Optimizations for Molecular Dynamics Simulation on Sunway Taihulight

Kun Li*†, Shigang Li∓, Bei Wang‡, Yifeng Chen‡, Yunquan Zhang*
*State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing
†School of Computer and Control Engineering, University of Chinese Academy of Sciences, Beijing
∓Department of Computer Science, ETH Zurich
‡HCST Key Lab at School of EECS, Peking University, Beijing
Email:{likungw,shigangli.cs}@gmail.com,zyq@ict.ac.cn

***Abstract*—Molecular dynamics is an extensively utilized computational tool for solids, liquids and molecules simulation. Currently, much research on molecular dynamics simulation focuses on simplifying forces or parallelizing tasks to reduce the overheads of forces computation. However, the molecular dynamics simulation still remains challenging since the communication and neighbor list construction are time-consuming in the existing algorithm. In this paper, we propose a swMD optimization strategy including a new communication mode called ghost communication to reduce superfluous communication overheads and an innovative neighbor list algorithm to improve the construction efficiency of it. Moreover, we accelerate computation by utilizing many-core resources on Sunway Taihulight and present an auto-tuning Producer-Consumer pairing algorithm to make neighbor list construction happen in fast register communication. Compared to traditional methods, swMD optimization strategy obtains a maximal 82.2% and an average of 79.4% performance improvement. We also evaluate the scalability up to 266,240 cores and the results demonstrate the high efficiency of swMD optimization strategy on communication, computation and neighbor list construction respectively.**

***Index Terms*—Ghost communication, Molecular dynamics, Neighbor list, Register communication, Sunway architecture**

I. Introduction

Molecular simulation techniques including molecular dynamics (MD) and Monte Carlo (MC) methods are widely utilized to study the molecular system in different fields [21], [29]. Since the limitation of hardware and computing techniques, no more than a few thousand particles are enabled in molecular simulation over nanosecond time in the past. Currently, advanced computing techniques and high-performance supercomputers have provided essentials for large-scale simulation and up to millisecond time scales on clusters [12]. Conventionally, the MD algorithm mainly includes two basic steps: computation of the intermolecular forces and establishment of neighbor lists for all particles [21].

In the physical system, the potential energy of particles is often described as a sum of N-body interactions for all particle pairs [29]. Generally, the interaction among particles will decrease rapidly as the distance increases [30], and a cutoff distance r_{cut} is introduced in molecular dynamics simulation. The interaction that actually works between particle pairs are defined as short-range forces within the cutoff distance. Based on the above theory, a common choice of forces computation is to compute the total potential energy for a certain particle utilizing its newest neighbors. When simulated on modern supercomputers, all particles data are distributed on different processes, and a few neighbors of the boundary particles are not involved in local process. Thus, it is necessary to exchange particles with neighboring processes. However, since the simulation performance will be limited greatly by the massive inter-process communication, it is necessary to reduce the redundant communication in large-scale molecular dynamics simulation.

When essential particle data are gathered to local process, the following step is to construct neighbor lists [28]. A central particle refers to the particle which is constructing its own neighbor list. Since the interactions beyond r_{cut} are neglected, the neighbors for a central particle i will produce a sphere space, where i and r_{cut} are center and radius respectively [30]. Currently the neighbor lists are built by two traditional algorithms: Verlet table algorithm [25] and cell linked list algorithm [18]. However, the expensive distance computation in these algorithms dominates the overall neighbor list construction time, making the performance of molecular dynamics simulation far from satisfactory.

To reduce the communication overheads, we propose a new communication mode called ghost communication to replace conventional total-exchange communication. Ghost communication analyzes the data dependencies, divides the local particle data into fine-grained sectors and transfers them instead of the whole local data to reduce the communication redundancy. We further accelerate computation part by utilizing many-core architectures, and the experimental result shows that communication time is reduced a lot.

As for neighbor lists construction, fast neighbor list algorithm is designed to remove redundant distance calculation by adopting bitwise operations and subtractions. On modern processors, the bitwise operations and subtractions are performed much faster than the multiplications and square root operations [3], [8]. The periodic boundary check is also taken into account in fast neighbor list algorithm. Moreover, the

This work was supported by National Natural Science Foundation of China under Grant No. 61502450, Grant No. 61432018, and Grant No. 61521092; National Key R&D Program of China under Grant No. 2017YFB0202302, Grant No. 2016YFB0200800, and Grant No. 2016YFE0100300.

978-1-7281-4328-6/19/$31.00 ©2019 IEEE
DOI 10.1109/ISPA-BDCloud-SustainCom-SocialCom48970.2019.00078

storage requirements are released a lot in this way and more particle pairs could be loaded to CPEs on Sunway architecture.

In order to fully use many-core resources, we further accelerate the computation and neighbor list construction on Sunway Taihulight. Based on the fast neighbor list algorithm and the pairing method [26], we achieve an auto-tuning Producer-Consumer pairing algorithm, where the two stages of the algorithm are held in the paired Producer and Consumer respectively. By handling the result produced by first stage immediately through register communication, Producer-Consumer pairing algorithm reduces the synchronization time caused by load imbalance notably and avoids frequent global memory accesses.

All optimization methods are packed into swMD strategies and it was evaluated on Sunway processors. For strong scalability, 131,072 particles are simulated on up to 266,240 cores to evaluate the performance of swMD and for weak scalability, 2,048 particles are simulated per core group. Compared with the traditional molecular dynamics simulation algorithm, our basic optimization strategies employing ghost communication, many-core computation and fast neighbor list algorithm achieve a 73.3% performance improvement on average. When Producer-Consumer pairing algorithm is utilized by register communication, our swMD method outperforms the traditional one by as much as 79.4% and a maximal improvement 82.2%.

The major contributions in this paper includes:

- The ghost communication mode is designed by dividing data into fine-grained sectors to eliminate the communication redundancy and improve the utilization of transfer data.
- A fast neighbor list algorithm is proposed to accelerate the neighbor lists construction by adopting two subtractions and several bitwise operations.
- Based on the SW26010 architecture, we achieve a significant acceleration for the computation and neighbor lists construction by employing many-core resources.
- An auto-tuning Producer-Consumer pairing algorithm is designed to further reduce the synchronization time caused by load imbalance and to avoid frequent global memory accesses through fast register communication.

II. Background

Since silicon material is crucial to information industries and has diverse potentials in thin-film transistors, photovoltaic resistors and biomedical sensors, it is commonly selected as candidate material for molecular dynamics simulation [5]. Taking into account the physical properties of crystalline silicon, the Tersoff III potential [24] is employed to calculate thermodynamic and dynamic processes in this paper.

A. Communication for Molecular Dynamics Simulation

Large-scale molecular dynamics simulation constitutes a popular applications in various fields. However, as the simulation size increases, the particle data are distributed on different processes. In order to update the particle information on local process, the communication between processes is inevitable. Considering the role of short-range forces, the local process only needs to communicate with neighboring processes. Conventionally, the local process is treated as the center, and the traditional total-exchange communication occurs when the particles in local process needs to update. Similarly, the particles on local process are also transferred to other neighboring processes without precise judgements for particle updates on them.

B. Neighbor List Algorithms

The traditional neighbor list algorithms primarily include Verlet table algorithm and cell linked algorithm.

The basic idea of Verlet table algorithm is to build and maintain a list of neighbors for each particle. Firstly, we introduce an extension distance r_{skin} [30]. Since a sphere space is created by r_{cut}, r_{skin} will continue to produce a skin surrounding the sphere. For a central particle i, another particle j is counted into neighbor list when the distance r_{ij} between them is less than $r_{cut} + r_{skin}$ [30]. In this way, the neighbor list contains less redundant particles, however, it is built for every particle and the construction of it scales dramatically with increasing simulation size [25].

In cell linked list algorithm, the simulation domain is divided into several cells, and the cells edge is generally no less than r_{cut}. The procedure of neighbor list construction is efficient since the atom indices are organized as a linked list. For a central particle i in cell m, all surrounding cells (9 cells for 2D simulation and 27 cells for 3D simulation) are required to be checked [30]. Compared to Verlet table algorithm, the cell linked list algorithm is more suitable for a relatively large simulation system [30].

The traditional algorithms have been intensively studied and applied in many molecular dynamics simulations [13]. However, these algorithms universally use the Euclidean Metric [4] to compute the interatomic distance. Apparently, the distance calculation involves some time-consuming multiplications and square root operations, and the cost dominates most of the time for neighbor list construction.

Recently, significant research has been devoted to developing a variety of molecular dynamics simulation packages, including GROMACS [11], LAMMPS [19], [23] and NAMD [15], [22]. A combined solution for Verlet table algorithm and cell linked list algorithm is utilized in these packages. The cell length is determined according to the shortest cutoff radius, r_{cut}, and the search area is restricted in the largest cutoff radius that is equal to $r_{cut} + r_{skin}$. Though redundant search for neighbors is avoided, the distance judgements are still computationally intensive and a mass of cores are used to get a reasonable performance in the simulation of colloidal systems [12]. Fen-Zi [9] or HALMD [7], [10] are exclusively designed for GPUs and limited feature set is implemented in these codes. HOOMD-blue [1], [2] is a more recent addition and a rich feature set is provided for general-purpose molecular dynamics simulations. However, the codes are designed for NVIDIA GPUs exclusively [10]. These popular molecular dynamics packages generally use a combination of Verlet table

algorithm and cell linked list algorithm on serial or vector machines [20]. Moreover, the neighbor lists are typically implemented using a brute force distance check for particle pairs, which compares the square of distance directly. Even though the comparison for squares in the combined solution only needs 3 additions, 3 subtractions and 4 multiplications, the overhead is still hard to be ignored in large-scale simulation.

C. The SW26010 Many-Core Processor

The Sunway Taihulight supercomputer (125 Pflops peak performance, composed of 40,960 processors) is mainly assembled by SW26010 many-core processor [16]. Each processor is made up of four core groups (CGs), and each CG consists of 65 cores. A CG contains one management processing element (MPE) and 64 Computing Processor Element (CPEs). The computation on MPE could be accelerated by CPEs, and they are organized as an 8×8 mesh [16]. The MPE contains a 64 KB L1 cache and a unified 256 KB L2 cache. Each CPE contains a 16 KB L1 instruction cache and a 64 KB local store (user-controlled scratch pad memory, SPM). The processor connects to others by a system interface.

The SW26010 processor is characterized by fast register communication among the 8×8 CPEs. The latency of fast register communication on CPEs is at most 11 cycles. Because of the restriction of hardware and other conditions, the fast register communication is only supported in the same column or row among the CPE mesh [16], [26].

D. Challenges of Constructing Neighbor Lists on SW26010

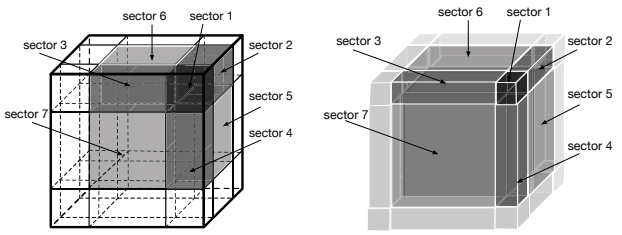
Even though vast research has been performed on Sunway architecture to fully leverage the hardware advantages of SW26010 [6], [14], [27], it is still challenging to utilize these characteristics adequately for constructing neighbor lists on it.

First, the fast neighbor list algorithm contains a large quantity of bitwise operations to relieve the overheads brought by traditional algorithms. However, the SW26010 processor provides few bitwise vector instructions. Moreover, in register communication, the CPE can not distinguish the data source in Receive Buffer, and it is not guaranteed the sequence of receiving data when multiple CPEs are sending to the same one simultaneously. Furthermore, the proportion of Producer and Consumer processors also needs determining exactly, otherwise some CPEs are idle when loads are imbalanced and the efficiency for constructing neighbor lists decreases severely.

III. Optimization

A. Ghost Communication for molecular dynamics simulation

The key point of ghost communication is to eliminate redundant particle data transfers. Fig.1 presents the data transfers of ghost communication. The particle data are divided into three kinds of data sectors [17] which are calculated in advance on all processes: *Point Sectors*, *Edge Sectors* and *Surface Sectors*.



(a) *Send Sectors* (b) *Recv Sectors*

Fig. 1. The *Send Sectors* and *Recv Sectors* in ghost communication. The left part displays the 26 *Send Sectors* of local process (outer balck hollow cube) and the right part shows the local process (inner black solid cube) receives 26 sectors from neighboring processes.

a) Point Sectors: The particle data distributed on eight vertex region are defined as *Point Sectors*. As shown in Fig.1, the local simulation space is described as a outer black hollow cube and an inner black solid cube respectively. The small dark grey cube indicated with "sector 1" is a *Send Point Sector* of local process in Fig.1(a), and the sector in same position of Fig.1(b) is a *Recv Point Sector*. Since every local simulation space contains eight vertex regions, the total number of *Point Sectors* is eight.

b) Edge Sectors: The particles in edge regions of local process are aggregated into *Edge Sectors* in terms of the cutoff distance. As shown in Fig.1, the three dull grey *Edge Sectors* adjacent to the dark grey Point Sector are *Send Sectors* or *Recv Sectors* respectively, and they are marked with sector 2, sector 3 and sector 4. The local process contains 8 vertexes, and 3 edges adjacent to a vertex. Since one edge is shared by 2 vertexes, there are 12 $(8 \times 3 \div 2)$ *Send Edge Sectors* or *Recv Edge Sectors* in local process.

c) Surface Sectors: The light grey sectors marked with sector 5, sector 6 and sector 7 are *Surface Sectors* in Fig.1. Generally, the thickness of *Surface Sectors* are equal to the cutoff distance. Since every 3 surface linked to a vertex and every surface is shared by 4 vertexes, the local process contains 6 $(8 \times 3 \div 4)$ *Send Surface Sectors* and also receives 6 *Surface Sectors* from surrounding processes.

Ghost communication removes the data dependence by analyzing the effect ranges of cutoff distance, and then divides local data into more fine-grained sectors to transfer instead of transferring the whole data in local. According to our experiment, ghost communication suits problems with lots of particles on each cores. Detailed results can be found in Sec.IV.

B. Fast Neighbor List Algorithm

In this section, we discuss the general fast neighbor list algorithm that can improve the construction efficiency significantly.

Firstly, the storage requirements for particle position are released since only an integer value of *int* type is utilized to contain all data of three dimensions. For a central particle P_0, an *int* type T_0 is used. Secondly, the highest 2 bits of T_0 are set to 0, and the remaining bits are equally divided into 3

segments. For each segment, the highest bit is reserved and also set to 0. The following 9 bits of each segment are used to represent the coordinate value of each dimension for P_0. For a candidate neighbor particle P_1, the *int* value is defined as T_1. Then the following formula is utilized in fast neighbor list algorithm:

$$D_0 = [(T_0 \mid Mask_0) - T_1] \& Mask_1 \tag{1}$$

Equation (1) involves 1 subtraction and 2 bitwise operations. The cutoff radius is defined as R. $Mask_0$ and $Mask_1$ are two integer constants. For $Mask_0$, three reserved bits are set to 1 and other bits are all set to 0. For $Mask_1$, the lower $\lceil log_2(R+1) \rceil$ bits and reserved bits of each segment are set to 0, and other bits of it are all set to 1. Using Equation (1), we get the value D_0. Then we exchange the positions of T_0 and T_1 in Equation (1), and D_1 is generated. We will use D_0 and D_1 for further judgment.

a) Bitwise AND operation:

$$Result_0 = D_0 \& D_1 \tag{2}$$

A bitwise *AND* operation are performed between D_0 and D_1 using Equation (2), and we obtain $Result_0$. The algorithm continues if $Result_0$ is zero. Otherwise, the algorithm returns the result that P_1 is not a neighbor of P_0.

b) Left Shift and bitwise AND operations:

$$Result_1 = (D_0 \ll 1) \& D_1 \tag{3}$$

It is not guaranteed that P_1 is the neighbor of P_0 when $Result_0$ is equal to 0. We further perform left shift and bitwise *AND* operations on D_0, as shown in Equation (3). If the result $Result_1$ is equal to zero, the algorithm continues.

c) Right Shift and bitwise AND operations:

$$Result_2 = (D_0 \gg 1) \& D_1 \tag{4}$$

Similarly, right shift and bitwise *AND* operations are carried out on D_0 in Equation (4). P_1 is a neighbor of P_0 when $Result_2$ is zero. Otherwise, P_1 is not a neighbor of P_0. We can notice that P_1 is a neighbor of P_0 only if $Result_0$, $Result_1$ and $Result_2$ are all equal to zero.

However, the distance between P_0 and P_1 can still be larger than the cutoff distance even if $Result_0$, $Result_1$ and $Result_2$ are all equal to zero. Thus, we only obtain an approximate neighbor list by now. If high precision is required in simulation, the particles will be further checked by Euclidean distance calculation. The overhead for the precise neighbor list construction is very low since the approximate neighbor list contains few redundant neighbors.

C. Many-core Acceleration on Sunway architecture

Since we adopt the ghost communication, abundant communication redundancy is eliminated. However, the ghost region computation inevitably introduces new overheads. The major task for ghost region computation is to determine the destination process for every particle in local according to the cutoff distance, and all particles are checked in sequence on MPE. When the essential particles are fetched from surrounding processors by ghost communication, they need to start constructing neighbor lists in sequence. It is a serial process on each processor and abundant many-core resources are wasted. In order to further tap the potential of CPEs, we accelerate the computation part and construction of fast neighbor lists on many-core processors.

The key point of many-core acceleration is to divide the simulation task on local processor into fine-grained subtasks and assign them on different CPEs. It is worth noting that data dependences should be eliminated among subtasks. From these considerations, we distribute ghost region and interaction forces computation from MPE to available CPEs, and then every CPE constructs neighbor lists for a part of central particles separately. Apparently, the many-core resource provides more computation ability to the simulation task and the experimental results also prove a significant performance improvement in Sec.IV.

However, though fast neighbor list algorithm accelerated by CPEs makes a dramatical breakthough on performance, new problems caused by employing many-core resources arise: load imbalance and frequent memory accesses. When tasks are divided unequally on CPEs, the whole computation time of processor is determined by the CPE whose task burden is the heaviest. Thus it is usually the case that some CPEs are busy for computation while others are idle. In addition, when precise neighbor judgement is required, every CPE needs memory accesses to MPE for writing information to neighbor list once a new neighbor particle is determined by computation. This interrupt also makes cache accesses discrete inevitably and the potential benefits of spatial locality can not be obtained easily.

D. Auto-tuning Producer-Consumer Pairing Algorithm

In this subsection, we propose an auto-tuning Producer-Consumer pairing algorithm based on pairing method [26] to use fast register communication for further improving the performance of parallel fast neighbor list algorithm.

Generally, the molecular dynamics simulation will perform thousands of iterations for a given simulation case, and the Producer-Consumer pairing algorithm contains an auto-tuning mechanism to adjust the proportion for Producers and Consumers dynamically according to the runtime every iteration. In this case, Producers and Consumers are defined as the left 32 cores and right 32 cores in the whole CPE mesh respectively. But note that arbitary available proportion is allowed for Producers and Consumers since the auto-tuning mechanism exists.

Fig.2 shows the producer-consumer pairing algorithm on SW26010 processor [26]. Each Producer contains **Cache_c**, **Cache_n** and **Buffer_1**. The former one and the middle one are used to cache and reuse central particle data and candidate particle data respectively, and the latter one is to store the three results generated by fast neighbor list algorithm for further judgements. Similarly, each Consumer also contains **Cache_c**, **Cache_n** and **Buffer_2**. The former one and the middle one

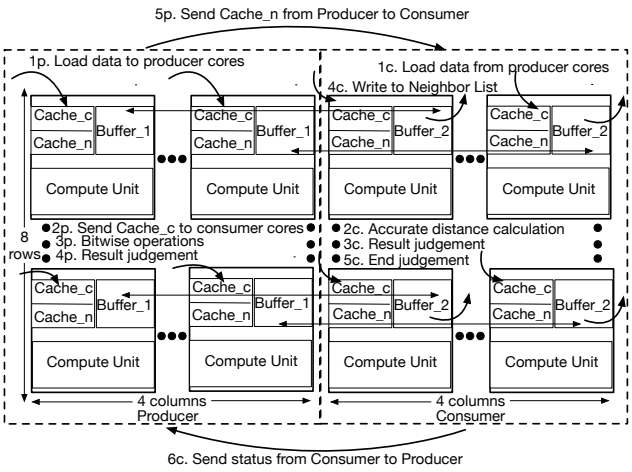


Fig. 2. The Auto-tuning Producer-Consumer Pairing Algorithm on SW26010 Processor.

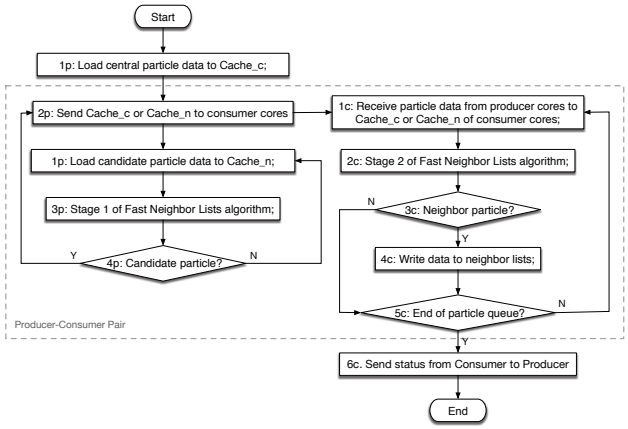


Fig. 3. The neighbor list construction process for a central particle on SW26010 processor.

are used to cache and reuse the particle data received from Producer by register communication, and the latter one is to store the result produced by precise neighbor list computation.

Each Producer is matched with one Consumer, thus the **Cache_c** and **Cache_n** on one Producer are exactly needed by paired Consumer. Producers will send the corresponding particle data to their paired Consumers by fast Row register communication to fulfill the precise neighbor list computation. These sending operations are marked with the "1c" and "5p" in Fig.2. As Fig.2 displays, the steps performed on Producers and Consumers are attached with "p" and "c" on step number respectively.

Next we introduce the fast neighbor list construction process for a central particle utilizing auto-tuning Producer-Consumer pairing algorithm, as exhibited in Fig.3. The numbers 1p to 4p and 1c to 6c in Fig.3 are the corresponding operations in Fig.2.

a) Step 1: The Producers fetch the current central particle data from MPE to **Cache_c**, which is indicated by 1p operation in Fig.3. The **Cache_c** need transferring to the corresponding caches in paired Consumers only when this central particle is a new one to construct neighbor list.

b) Step 2: The Producers continue to load candidate particle data to **Cache_n**. In the whole neighbor list construction for a certain central particle, a new candidate particle is transferred to the **Cache_n** every time while the central particle information only caches one time.

c) Step 3: Once a new candidate particle is cached in Producers, the fast neighbor list algorithm is performed on **Cache_c** and **Cache_n**. Then three results are generated as mentioned in Sec.III-B, and they are stored in **Buffer_1** waiting for further judgements.

d) Step 4: For Producers, the candidate particle data will be sent to paired Consumers directly and then empty **Cache_n** and replenish it with next candidate particle if three results are all equal to 0. Otherwise, the Producers load the next candidate particle immediately which is corresponding to 4p in Fig.3. When the judgements for three results have completed, a whole procedure for Producers ends and these steps will loop till all candidate particle have been checked. At that time, both **Cache_c** and **Cache_n** will be emptied and a new round of neighbor list construction for next central particle starts.

For Consumers, the **Cache_n** is filled with the data receiving from Producers and then a precise neighbor list computation is carried out on **Cache_c** and **Cache_n**. The result is stored in Buffer_2 on Consumers. When the result proves a neighbor, the Consumer will record it and write to neighbor lists in idle time and then continues to 5c in Fig.3, or it will judge whether the candidate particle is an end of queue or not directly. If the answer is "No", consumers empty the **Cache_n**, receive a new candidate particle and loop these steps. While, if the answer is "Yes", the Consumer will empty **Cache_c** and **Cache_n**, and send status information to paired Producer.

In fact, the Producers and Consumers will work together and run separately except 1p and 6c operations in Fig.3. Through decoupling the computation process of neighbor list and dividing different functions on CPEs precisely, all CPEs work with its own designated task. In addition, since all precise computation for neighbor list are accomplished on Consumers, Producers tap the potential from spatial locality and avoid frequent memory accesses. Based on Producer-Consumer pairing algorithm, we achieve to construct neighbor list more efficiently on SW26010 processors.

IV. Experimental Results

To compare our proposed swMD optimization strategy with traditional algorithm, we evaluate them in our application that is implemented as the silicon atoms model on Sunway Taihulight supercomputer. Taking into account the heat and stress changes of silicon, the Tersoff III potential [24] is utilized for energies calculation. The codes are compiled by SWCC Compilers (Version 5.421-sw-500), which is a customized compiler

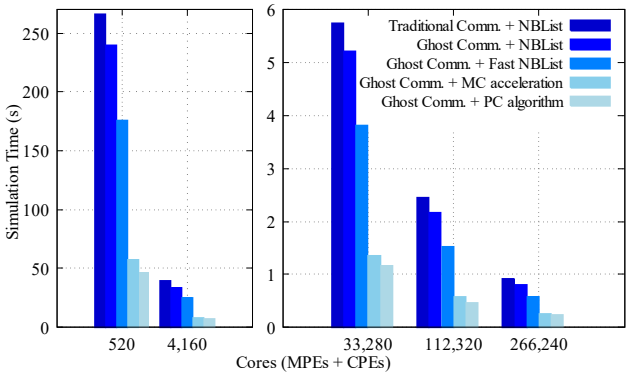


Fig. 4. Total runtime comparisons for utilizing various optimization strategies.

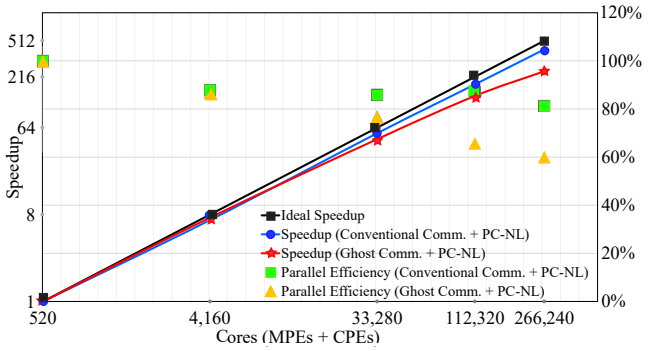


Fig. 5. Performance comparisons: Strong scaling

for Sunway architecture. Then the performance for progressive optimizations, scalability, and performance for single module are all evaluated. For convenience, the comm. is short for communication; the NBList is short for traditional neighbor list; the MC acceleration means computation and fast neighbor list construction optimized with many-core acceleration; PC algorithm refers to auto-tuning Producer-Consumer pairing algorithm applied on the swMD version of MC acceleration.

A. Performance Comparisons for Various Optimizations

Firstly, we evaluate the optimization effects when various optimization strategies are applied on the simulation in sequence. We test the molecular dynamics simulation with 131,072 particles using different number of processors as shown in Fig.4. The benchmark utilizes traditional neighbor list algorithm and communication mode. Ghost communication improves performance significantly on benchmark algorithm by eliminating redundant data transfers. Then Fast NBList, MC acceleration and PC algorithm are employed successively.

It can be seen evidently that the performance improves gradually when these optimization algorithms are used. The PC algorithm obtains a 99.1% performance improvement than traditional algorithm on average, and the precise investigations are illustrated in Sec.IV-D.

Compared with the traditional algorithm, our basic optimization strategies by employing ghost communication and MC acceleration achieve a 38.6% performance improvement on average. The swMD method utilizing PC algorithm outperforms the traditional one by as much as 52.0% and a maximal improvement 57.8%. Moreover, the total runtime decreases with the increase of processors, which indicates a good scalability for the swMD optimization strategy.

B. Strong Scaling

For strong scaling evaluation, we fix the simulation size and use the same 131,072 particles as Sec.IV-A which are random initialized. As mentioned above, the significant effects of different optimization algorithms on total runtime has been exhibited in Fig.4. For further discussion on strong scaling, we compare the ideal speedup with swMD rigorously in Fig.5. The Producer CPEs are responsible for F-NL part and the Consumer CPEs are in charge of precise neighbor computation.

We can see that the swMD optimization strategy has a good strong scalability. Parallel efficiency for swMD gradually decreases as cores increase, which is mainly caused by the communication overheads. Scaling from 520 cores to 266,240 cores, a 246.6x speedup (60.9% parallel efficiency) is obtained for swMD optimization strategy.

In our investigations, when computing resources scales, the superior performance of traditional comm is in contrast to our expectant wisdom of using ghost comm with PC algorithm to achieve the best performance. The main reason is that the benefits brought about by only exchanging essential sectors are counteracted by computing the exchanging sectors when the simulation task is not heavy on each processor in strong scaling. Moreover, the parallel efficiencies of swMD are inferior to the traditional algorithm when 33,280 or more cores are used. The main reason is that the baseline of swMD exploits the performance of swMD optimization with sufficient computation task. With the increase of cores, the computation workload on each processor is gradually reduced in strong scaling. Therefore, the parallel efficiencies decrease with a comparison to strong baseline.

C. Weak Scaling

Fig.6 shows the weak scaling test for our swMD, and parallel efficiency value is specified for each bar. We compare the performance of traditional method and swMD respectively. As we increase the number of cores from 520 (including 8 MPEs and 512 CPEs), the problem size increases from 1.6×10^4 particles to 8.4×10^6 particles. Our swMD scales up to 266,240 cores with total 8.4×10^6 particles by a 86.8% parallel efficiency.

We can see that the communication time and computation time dominate nearly all overheads in the process of simulation, and the proportions of them remain almost constant on different number of cores. Since the communication contention

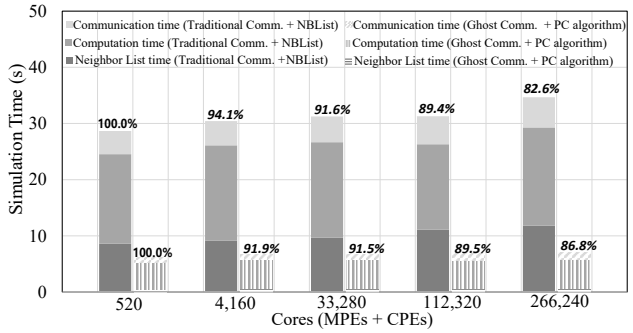


Fig. 6. Performance comparisons: Weak scaling. Parallel efficiency value is specified for each bar.

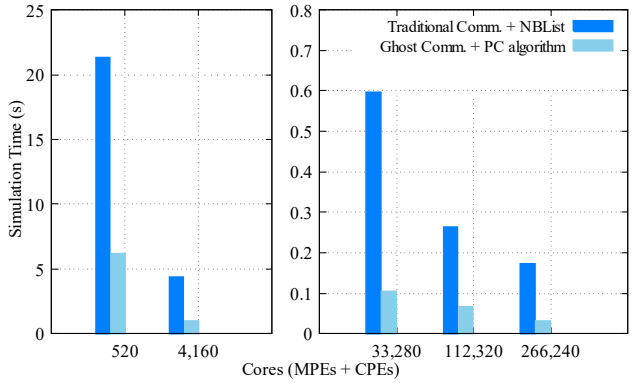


Fig. 7. Time comparisons of communication module.

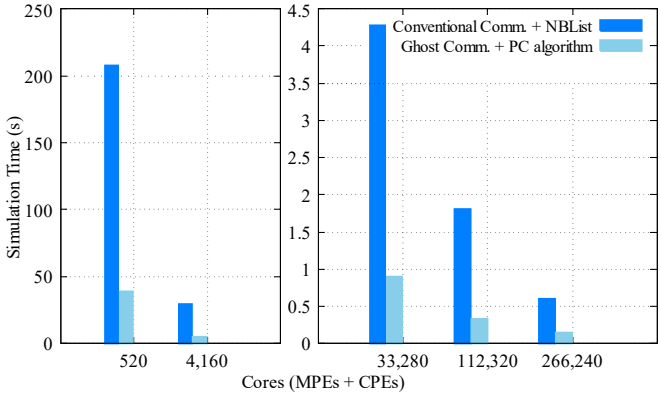


Fig. 8. Time comparisons of computation module.

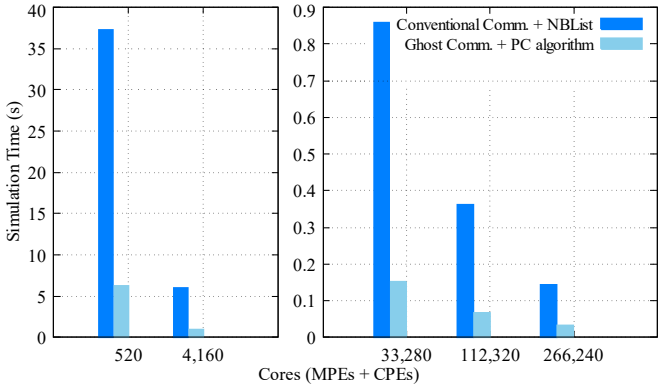


Fig. 9. Time comparisons of NBList module.

exists, the communication time is a little higher in traditional algorithm when more cores are used.

Both the computation time and communication time decreases obviously when employing swMD strategy. This is because a part of time is used to compute sectors for data exchanging thus the number of particles for communication reduces significantly.

As Fig.6 shows, the neighbor lists time decrease sharply and it only takes up a small proportion in the simulation when swMD is employed, which indicates that PC algorithm could construct it efficiently. Overall, our swMD method exhibits a good strong and weak scalability.

D. Performance Comparisons for Single Module

Since the communication, computation and neighbor list construction account for different proportions in the whole simulation, the optimization effect on a single module is not distinct. Therefore, we exhibit the effect of our proposed swMD optimization on every single module in this subsection. The configurations are same with the strong scaling experiments, and each set of bars in Fig.7 to Fig.9 displays the simulation effect before or after the swMD optimization clearly.

Fig.7 presents the communication time comparisons for the traditional algorithm and the swMD. Compared to the traditional algorithm, the swMD obtains an average speedup of 4.44x and this proves that our swMD optimization reduces communication time greatly. Moreover, nearly all benefits on decreasing communication time are introduced by ghost communication method, which is important in swMD optimization.

Fig.8 presents the computation time comparisons for different optimization methods. In previous investigations, because the swapping sectors need computing before communication, the corresponding computation time increases 56.8% on average when ghost communication mode is utilized on MPE. However, we can see that when many-core acceleration is employed, a large deal of computation time is saved and it has a 66.4% average performance improvement, which demonstrates the swMD could improve the computation efficiency definitely.

We also present the time comparisons results for neighbor list module in Fig.9. We increase the number of cores from 520 to 266,240 and keep the total number of particles fixed on 131,072. The result exhibits that swMD is efficient in improving the performance of neighbor lists construction. In addition, PC algorithm taht utilizes CPEs on SW26010 contributes a lot in swMD for neighbor list acceleration and

the optimization obtains a 5.13x speedup on average. When 520 cores are used, the computation workload is sufficient and a maximal 6.17x speedup is obtained.

V. Conclusion

Molecular dynamics simulation is crucially important in many scientific research. In this work, we propose a ghost communication method to eliminate redundant data transfers and an innovative algorithm to accelerate neighbor list construction in molecular dynamics simulation. These optimizations are general and can be applied on other architectures. Then, we utilize the many-core resources to optimize the computation and fast neighbor list algorithm specially on SW26010 processor. Furthermore, an auto-tuning Producer-Consumer pairing algorithm is presented to construct neighbor lists by fast register communication. The experimental results shows that our approach largely outperforms the traditional algorithms and has a good strong and weak scalability on SW26010 processors.

As future work, we plan to optimize the communication among CGs in SW26010 processor to further exploit the intra-node performance. We believed that our proposed optimization strategies could introduce insightful experience to algorithm design on other architectures.

References

[1] Joshua A Anderson and Sharon C Glotzer. The development and expansion of hoomd-blue through six years of gpu proliferation. *arXiv preprint arXiv:1308.5587*, 2013.

[2] Joshua A Anderson, Chris D Lorenz, and Alex Travesset. General purpose molecular dynamics simulations fully implemented on graphics processing units. *Journal of Computational Physics*, 227(10):5342–5359, 2008.

[3] ARM. ARM Cortex-A Series: Programmer's Guide for ARMv8-A, 2015.

[4] Leonard Mascot Blumenthal. *Theory and applications of distance geometry*. Chelsea New York, 1970.

[5] Akram I Boukai, Yuri Bunimovich, Jamil Tahir-Kheli, Jen-Kan Yu, William A Goddard Iii, and James R Heath. Silicon nanowires as efficient thermoelectric materials. In *Materials For Sustainable Energy: A Collection of Peer-Reviewed Research and Review Articles from Nature Publishing Group*, pages 116–119. World Scientific, 2011.

[6] Yuedan Chen, Kenli Li, Wangdong Yang, Guoqing Xiao, Xianghui Xie, and Tao Li. Performance-aware model for sparse matrix-matrix multiplication on the sunway taihulight supercomputer. *IEEE transactions on parallel and distributed systems*, 30(4):923–938, 2018.

[7] Peter H Colberg and Felix Höfling. Highly accelerated simulations of glassy dynamics using gpus: Caveats on limited floating-point precision. *Computer Physics Communications*, 182(5):1120–1129, 2011.

[8] Agner Fog. Optimizing subroutines in assembly language: An optimization guide for x86 platforms, 2016.

[9] Narayan Ganesan, Michela Taufer, Brad Bauer, and Sandeep Patel. Fenzi: Gpu-enabled molecular dynamics simulations of large membrane regions based on the charmm force field and pme. In *2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum*, pages 472–480. IEEE, 2011.

[10] Jens Glaser, Trung Dac Nguyen, Joshua A Anderson, Pak Lui, Filippo Spiga, Jaime A Millan, David C Morse, and Sharon C Glotzer. Strong scaling of general-purpose molecular dynamics simulations on gpus. *Computer Physics Communications*, 192:97–107, 2015.

[11] Berk Hess, Carsten Kutzner, David Van Der Spoel, and Erik Lindahl. Gromacs 4: algorithms for highly efficient, load-balanced, and scalable molecular simulation. *Journal of chemical theory and computation*, 4(3):435–447, 2008.

[12] Michael P Howard, Joshua A Anderson, Arash Nikoubashman, Sharon C Glotzer, and Athanassios Z Panagiotopoulos. Efficient neighbor list calculation for molecular simulation of colloidal systems using graphics processing units. *Computer Physics Communications*, 203:45–52, 2016.

[13] Michael P. Howard, Joshua A. Anderson, Arash Nikoubashman, Sharon C. Glotzer, and Athanassios Z. Panagiotopoulos. Efficient neighbor list calculation for molecular simulation of colloidal systems using graphics processing units. *Computer Physics Communications*, 203:45 – 52, 2016.

[14] Lijuan Jiang, Chao Yang, Yulong Ao, Wanwang Yin, Wenjing Ma, Qiao Sun, Fangfang Liu, Rongfen Lin, and Peng Zhang. Towards highly efficient dgemm on the emerging sw26010 many-core processor. In *2017 46th International Conference on Parallel Processing (ICPP)*, pages 422–431. IEEE, 2017.

[15] Wei Jiang, David J Hardy, James C Phillips, Alexander D MacKerell Jr, Klaus Schulten, and Benoît Roux. High-performance scalable molecular dynamics simulations of a polarizable force field based on classical drude oscillators in namd. *The journal of physical chemistry letters*, 2(2):87–92, 2010.

[16] Kun Li, Shigang Li, Shan Huang, Yifeng Chen, and Yunquan Zhang. FastNBL: fast neighbor lists establishment for molecular dynamics simulation based on bitwise operations. *The Journal of Supercomputing*, pages 1–20, 2019.

[17] Shigang Li, Baodong Wu, Yunquan Zhang, Xianmeng Wang, Jianjiang Li, Changjun Hu, Jue Wang, Yangde Feng, and Ningming Nie. Massively scaling the metal microscopic damage simulation on sunway taihulight supercomputer. In *Proceedings of the 47th International Conference on Parallel Processing*, ICPP 2018, pages 47:1–47:11, New York, NY, USA, 2018. ACM.

[18] William Mattson and Betsy M Rice. Near-neighbor calculations using a modified cell-linked list method. *Computer Physics Communications*, 119(2-3):135–148, 1999.

[19] Steve Plimpton. Fast parallel algorithms for short-range molecular dynamics. *Journal of Computational Physics*, 117(1):1 – 19, 1995.

[20] Steve Plimpton. Fast parallel algorithms for short-range molecular dynamics. *Journal of computational physics*, 117(1):1–19, 1995.

[21] David Potter. Computational physics. 1973.

[22] John E Stone, James C Phillips, Peter L Freddolino, David J Hardy, Leonardo G Trabuco, and Klaus Schulten. Accelerating molecular modeling applications with graphics processors. *Journal of computational chemistry*, 28(16):2618–2640, 2007.

[23] Yu-Hang Tang and George Em Karniadakis. Accelerating dissipative particle dynamics simulations on gpus: Algorithms, numerics and applications. *Computer Physics Communications*, 185(11):2809 – 2822, 2014.

[24] Jerry Tersoff. Empirical interatomic potential for silicon with improved elastic properties. *Physical Review B*, 38(14):9902, 1988.

[25] Loup Verlet. Computer "experiments" on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules. *Physical review*, 159(1):98, 1967.

[26] Xinliang Wang, Wei Xue, Weifeng Liu, and Li Wu. swSpTRSV: a fast sparse triangular solve with sparse level tile layout on sunway architectures. In *Proceedings of the 23rd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 338–353. ACM, 2018.

[27] Guoqing Xiao, Kenli Li, Yuedan Chen, Wangquan He, Albert Zomaya, and Tao Li. Caspmv: a customized and accelerative spmv framework for the sunway taihulight. *IEEE Transactions on Parallel and Distributed Systems*, 2019.

[28] Helen Jun Xing, Muhammad Khawar R Khan, Rami H Alnatsheh, Ramesh Chandra Chirala, and Supratik Bhattacharjee. Method and apparatus for neighbor list updates, March 27 2012. US Patent 8,144,662.

[29] Zhenhua Yao, Jian-Sheng Wang, Gui-Rong Liu, and Min Cheng. Improved neighbor list algorithm in molecular simulations using cell decomposition and data sorting method. *Computer Physics Communications*, 161(1):27 – 35, 2004.

[30] Zhenhua Yao, Jian-Sheng Wang, Gui-Rong Liu, and Min Cheng. Improved neighbor list algorithm in molecular simulations using cell decomposition and data sorting method. *Computer physics communications*, 161(1-2):27–35, 2004.