This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCBB.2017.2713362, IEEE/ACM Transactions on Computational Biology and Bioinformatics



# A CPU/MIC Collaborated Parallel Framework for GROMACS on Tianhe-2 Supercomputer

+Shaoliang Peng*, +Shunyun Yang, +Wenhe Su, +Xiaoyu Zhang, +Tenglilang Zhang
School of Computer Science,
National University of Defense Technology
Changsha, China
E-mail:pengshaoliang@nudt.edu.cn
*Corresponding author +Equal Contributors

Weiguo Liu
School of Computer Science,
Shandong University,
Jinan, China
E-mail:weiguo.liu@sdu.edu.cn

Xingming Zhao
Department of Computer Science,
Tongji University,
Shanghai, China
E-mail:xm_zhao@tongji.edu.cn

***Abstract*—Molecular Dynamics (MD) is the simulation of the dynamic behavior of atoms and molecules. As the most popular software for molecular dynamics, GROMACS cannot work on large-scale data because of limit computing resources. In this paper, we propose a CPU and Intel® Xeon Phi Many Integrated Core (MIC) collaborated parallel framework to accelerate GROMACS using the offload mode on a MIC coprocessor, with which the performance of GROMACS is improved significantly, especially with the utility of Tianhe-2 supercomputer. Furthermore, we optimize GROMACS so that it can run on both the CPU and MIC at the same time. In addition, we accelerate multi-node GROMACS so that it can be used in practice. Benchmarking on real data, our accelerated GROMACS performs very well and reduces computation time significantly.**
**Source code: https://github.com/tianhe2/gromacs-mic**

***Index Terms*—GROMACS, Molecular Dynamics Simulation, High Performance Computing, Parallel Framework**

## I. Introduction

With the rapid advance of the pharmaceutical industry, there is a continuous demand for molecular dynamics, which is a key procedure in pharmaceutical research. GROMACS [1] is a versatile package for performing molecular dynamics, and is increasingly popular with pharmaceutical researchers for its convenience and practicality. In China, more than 20% of the computing resources in supercomputer centers are used for GROMACS simulations. Although GROMACS is highly efficient in its use of graphics processing units (GPUs) and SIMD [2], it does not meet the rapidly increasing demand of molecular dynamics, owing to the limits of the algorithms and models. It always takes millions of iterations to achieve a simulation. Furthermore, the interaction between thousands of atoms needs to be calculated at each step. Such a huge amount of computation means that computing resources can be a bottleneck in this field. As a consequence, it usually takes a few months to accomplish a simulation with GROMACS, which is an intolerable situation.

The current version GROMACS has already added SIMD to support Many Integrated Core (MIC) architectures. The SIMD data width of the core calculation is the only part that has been modified, and it is sufficient. Some methods of acceleration, such as storage modes and data alignment, are also quite useful. Although these modifications generally have little influence on efficiency, for software such as GROMACS, whose calculation amount is extremely large, a little optimization can be very helpful and valuable. If we are able to shorten the running time of each step by even tens of milliseconds, the incremental performance obtained by the optimization will be quite notable.

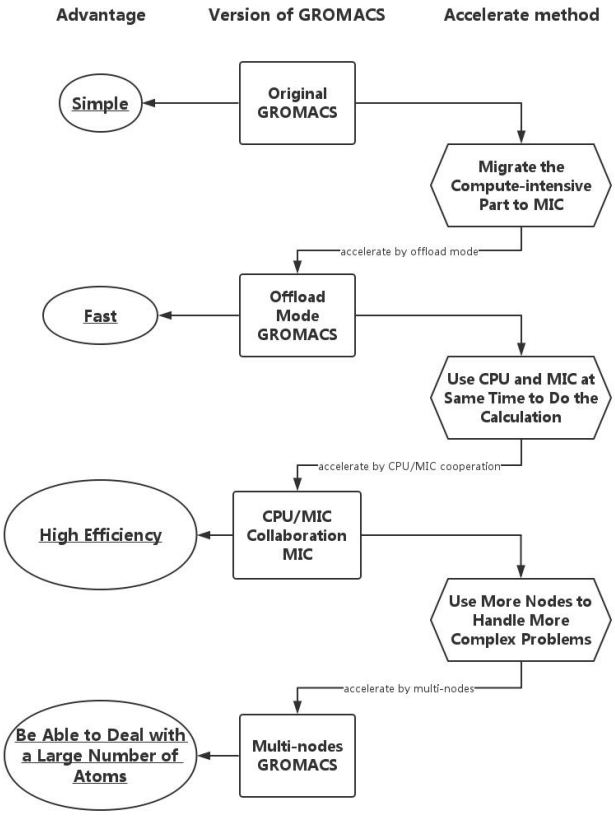


**Figure 1** A CPU/MIC collaborated parallel framework to accelerate GROMACS in multi-steps.

In this paper, we first characterize molecular dynamics simulation, GROMACS, MIC, and offload mode. We then introduce a method to accelerate GROMACS, which is carried out on a supercomputer with MIC in offload mode. Our method

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCBB.2017.2713362, IEEE/ACM Transactions on Computational Biology and Bioinformatics



is divided into three steps. First, based on the process of GROMACS, we carry out the calculation of the non-bonded force on the MIC, which can reduce the load on the CPU. Next, we propose dividing the kernel computing tasks at the atomic level, and utilize the new data flow to upgrade the software, which can make room for further acceleration and lead to faster software processing. In addition, the synergistic utilization of the CPU and MIC makes full use of the computing resources on the Tianhe-2 supercomputer. We balance the task load by experiment. At last, we apply new method on muti-node to make GROMACS more powerful, which means it can deal with larger atom systems. The relationship of three steps of acceleration is shown in Figure 1.

In addition, we evaluate our method using a series of tests that are carried out on the supercomputer to compare the performance of the updated and unmodified programs.

This paper is organized as follows: Section II introduces the necessary background of molecular dynamics simulation, GROMACS and MIC. In Sections III, IV, and V, we introduce our methods which are about accelerating GROMACS. Section VI explains our evaluation methods and presents the results of our experiments. We conclude the paper in Section VIII.

## II. Background and related work

### A. Molecular Dynamics Simulation

Molecular dynamics simulation is a method pioneered by Alder and Wainwright [3] in 1957. This extremely powerful technique involves solving the classical many-body problem in contexts relevant to the study of matter at the atomic level. Because there is no alternative approach capable of handling such a broad range of problems at the required level of detail, molecular dynamics methods have proved themselves indispensable in both pure and applied research. As a method of simulation, molecular dynamics solve micro-scale problems that traditional methods cannot explain. This simulation method obtains the macroscopic properties of a material by calculating the trajectory of microscopic molecules, thus enabling scientists to explain the macroscopic properties of material at a molecular level and modify macro properties using molecular prediction. Therefore, molecular dynamics simulation is a bridge between macro properties and micro motion. Nowadays, thousands of studies and articles relevant to molecular dynamics are published every year. In recent years, with the rapid developments in the energy, materials, mechanical and electrical industries, the large-scale computation of molecular dynamics simulations is widely demanded, and computation speed has become a bottleneck of development for many industries.

### B. GROMACS

GROMACS is a versatile package for performing molecular dynamics, i.e., simulating the Newtonian equations of motion for systems with hundreds to millions of particles. It is primarily designed for biochemical molecules like proteins, lipids, and nucleic acids that have many complicated bonded interactions, but because GROMACS is extremely fast at calculating non-bonded interactions (which usually dominate simulations) many groups are also using it for research on non-biological systems, e.g.,polymers. GROMACS supports all the usual algorithms expected by a modern molecular dynamics implementation [5].

The process of a GROMACS simulation can be divided into the following three stages:

(1) Initialization of the coordinates of the atomic system, topological structure of the files, balancing parameters, and external parameters.

(2) Main simulation process, which includes the following steps. Minimization of the energy of the system before the equilibrium process is balanced to avoid errors resulting in an unreasonable structure. The system is heated, the initial velocities are set up, and initial balance is achieved. Then, by means of the molecular dynamics simulation, the trajectory of each atom, looking for the minimum potential energy of the system, is calculated.

In the simulation process, the user can observe the state of the system at any time using the visualization software.

(3) After the simulation analysis, GROMACS produces a series of documents for further analysis.

### C. Intel MIC

Intel MIC Architecture is a co-processorcomputer architecture developed by Intel, incorporating earlier work on the Larrabee many-core architecture, Teraflops Research Chip multicore chip research project, and Intel Single-chip Cloud Computer multicore microprocessor[6]. Intel Xeon Phi coprocessors provide up to 61 cores, 244 threads, and 1.2 tera FLOPS of performance, and they come in a variety of configurations to address diverse hardware, software, workload, performance, and efficiency requirements [7]. The offload mode of MIC is very efficient. This mode could make the computer calculate the logic module on the CPU and calculate the computing module on the MIC, making full use of computing resources.

### D. Related Work

GROMACS has a highly developed source code, heavily linked modules, and deeply nested structures, which makes the offload mode hard to achieve. Furthermore, the molecular dynamics are difficult to parallelized because of the data dependencies. So far, GROMACS has been optimized with OPENMP, MPI, SIMD, and CUDA, achieving notable acceleration. As a result, it is now hard but also significant to optimize GROMACS, even just a small amount. Recently, Plotnikov [11] ran the GROMACS in *native* mode on a CPU and MIC with MPI. Although this could be a good approach to utilizing both the CPU and MIC at the same time, there are still some shortcomings remaining. First, we need to offload the binary file to each MIC, which is a large-scale task that takes many computing resources. Second, when we run GROMACS with large-scale atoms and molecules, the communication will become a bottleneck. For the reasons given above, the native mode is not yet a practical solution. Plotnikovonly tested the native mode using three MICs and two CPUs.

Compared to our previously presented work [5], we introduce the following new contributions in this paper:

(1) We have solved problems such as write-collision and communication overhead, which are caused by using the offload mode.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCBB.2017.2713362, IEEE/ACM Transactions on Computational Biology and Bioinformatics



(2) We have designed a new method to calculate the non-bonded force using both CPUs and MICs on a single node.
(3) We have implemented our method on multiple computation nodes so that we can do a larger scale simulation and solve more complex problems.

## III. ACCELERATION OF GROMACS USING OFFLOAD MODE

Molecular dynamics is an iterative process. In general, the system arranges millions of iterations, and every step only takes a small amount of time, sometimes even half a millisecond. In addition, we can only carry out the offload mode during one step, because the system has a strong data dependency, which means that the next step cannot start until the last one is finished. It is a challenge to arrange the offload mode in such a little time. In molecular dynamics, there are many calculations to perform in each step, and the most compute-intensive part is the non-bonded force calculation, which calculates the non-bonded forces between atoms. To calculate the force, we need to compute the interaction between almost every pair of atoms, which is at most $n \times n$ times of calculation, where $n$ is always the number of atoms. This number can range from thousands to millions. Luckily, some interactions between atoms are negligible, but the remaining number of non-negligible interactions is still very large.

The force contains two parts, the Coulomb force and the Vander Waal's force. At each step, GROMACS calculates the two forces billions of times as quickly as possible. The code to calculate these forces is included in two files, the outer code and the inner code. The outer code iterates each atom, and the inner code finds all the atoms that interact with this atom. Hence, at each step, the outer code is called $n$ times, and the inner code is called more frequently. The efficiency of GROMACS seriously depends on these two files, which are called kernel codes. GROMACS has many different kernel codes, and it transfers different codes according to the background to achieve the best performance.

At this stage of the optimization work, there are two key issues that can significantly affect the efficiency. The first problem we need to solve is the data transmission between the CPU and MIC. Based on the algorithm, when we calculate the non-bonded force, the essential input data are the coordinates and the charge of each atom. The size of these data is O(n). The essential output is the force of each atom, whose size is also O(n). However, in an implementation, these input and output data is insufficient. Some other information is also needed, such as atom types, atomic offsets, and some other parameters and control variables. These data also needs to be transmitted between CPU and MIC. The relationship and the data flow are shown in Figure 2.

Currently, when we need to exchange data between the CPU and MIC, we can only transfer the data that are stored in memory continuously, in a single step. That is to say, the discrete data in memory cannot be transmitted directly. Unfortunately, when we calculate the non-bonded force between atoms, a large amount of discrete data needs to be transmitted, resulting in multiple threads. For example, every thread has an output array of the non-bonded force which is a dynamic array, and is not continuous in memory. We can only use loops to achieve segment transmission, and the number of cycles equals to the number of threads. In this way, too many

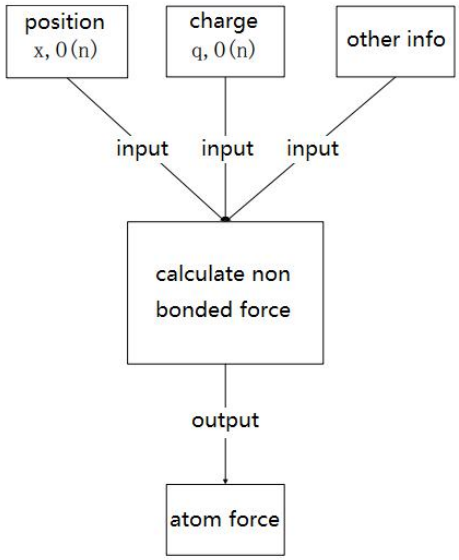


**Figure 2** Calculation of the non-bonded force

Instances of data transmission and logical operations occur, which leads to a large amount of time consumption to prepare and close the transmission between CPU and MIC. As a result, the extra overhead is quite large, and the transmission is very inefficient.

In order to reduce the communication overhead, it is necessary to trim the memory segment of the data. Our approach is to reconstruct the data array that needs to be transferred by allocating a larger amount of memory and then gathering the data. Although the overall size does not change, the data are continuous in memory now, and we do not need to utilize a loop anymore. We can hence transmit between the CPU and MIC just once.

In this way, we greatly reduce the time needed for communication, and we lessen the overhead caused by the repetition work, which means we overcome the disadvantage of using offload mode. However, such adjustments will also produce some additional overheads, including the time to allocate space and the extra time for memory copy. The latter can have a particularly serious impact on program's performance.

The other issue is write-collision, which can affect the efficiency when we use OpenMP to accelerate GROMACS on MIC for non-bonded force calculations between atoms. Write-collision occurs when many results are written back to the same address of the memory, including the force, energy and other information.

In the non-bonded force calculation, because of the interaction of two atoms, it is not only necessary to write the position of the current atom, but also the position of the atoms which the current one is interacting with. In this way, the workload can be reduced by half. However, there is no way to ensure data locality. The problem is that the same atom's position in the non-bonded force array can be written at the same time in different threads because of parallel computing. A

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCBB.2017.2713362, IEEE/ACM Transactions on Computational Biology and Bioinformatics



write-collision is then generated, which can reduce the efficiency of the code implementation.

The ideal method is to sort the atom pairs so that we can allow each thread to have access to a segment of the non-bonded force array to avoid write-collisions among different threads. However, during the calculation, the interactions between atoms are very complex, and it is quite hard to avoid write-collision by rearranging the atom pair order. Hence, although this solution is reasonable, it is quite difficult to implement. When studying acceleration methods for GROMACS, we found that GROMACS has a certain way to solve this problem: it allocates one output structure for each thread, and every structure has a complete copy of the non-bonded force. During the thread execution process, each thread then writes the data back to its own structure. GROMACS then deals with all the structures after all the threads have completed their computation.

## IV. Acceleration of GROMACS with Collaboration of CPU/MIC

Because the offload mode accelerated version of GROMACS does not achieve sufficient performance, we go a step further. Based on the offload mode of the MIC, we design and implement a new reasonable method, which allows the CPU to collaborate with the MIC on the Tianhe-2 super computer to accelerate GROMACS. The new method utilizes a new data flow to reduce the extra overhead caused by offload mode. We also propose a new load-balancing task allocation to make full use of Tianhe-2's computing resources. In this section, we will introduce the main principles and implementation of our method.

The most difficult problem is how to reduce extra resource consumption. It is well-known that the consumption caused by the data copy is completely unnecessary. Another issue is to reduce needless communication caused by the discrete storage of non-bonded force arrays of different threads. Hence, in order to improve the efficiency, we have to find a way to completely eliminate needless data copying, which means we have to solve the problem of discrete storage of non-bonded force arrays. However, in the current data flow, arrays are copied to each thread and stored in segregated structures. Hence, if the data flow cannot be changed, we cannot notably improve the efficiency. Overall, it would be pointless to utilize offload to accelerate GROMACS because of the large amount of extra consumption caused by the current CPU data flow.

However, GROMACS achieves good performance on a GPU, and the method used by the GPU is quite similar to the offload mode of MIC, which both needs to first transmit data, then does the calculations on the co-processors and return the result to the CPU for subsequent operations. We believe that there must be some method to solve the problem of discrete storage on a MIC. In numerous versions of the GROMACS kernel code, there are some sections of codes that are reserved for software developers who need to facilitate debugging work by using a CPU for a GPU simulation. At last, we found what we need the code utilizes the GPU's data flow in a CPU environment and simulates the calculation of the GPU without the CUDA function library. It is a good solution to the problem mentioned above. However, because of the limited number of threads on the CPU, the resulting performance is not very good. Luckily, this problem can be perfectly solved by the MIC, which has many cores. Hence, the solution is relatively simple: We modify the method introduced in the last section using the new data flow to reduce the extra overhead and communication consumption.

When we use both the CPU and MIC in offload mode to run GROMACS, we need to divide the tasks. This is a good way to lessen the workload on each computing unit(one MIC can be seen as one unit and two CPUs can be taken as one unit) and also a way to increase the efficiency. In the kernel calculation of the non-bonded force, we calculation the interaction between every two atoms and then record the result.The details is shown in Algorithm 1.

**Algorithm 1:** Kernel calculation of the non-bonded force

```
for(atom i){// for every atom in the atomic system
   for(atom j){/* select one atom j to calculate the
non-bonded force betweeni and j*/
      tf=cal_non_bonded(i,j);
      f[j]-=tf;
      f[i]+=tf;
   }
}
```

There are many other control variables in the practical implementation of this process. Algorithm 1 is just a simplified model. In order to balance the task division, we open the outer loop in the code above. That is, we segregate atom i into several parts, and each part is assigned to a different computing unit. Using this task division, it is quite easy to adjust the workload onto different units by changing the number of times that the for loop runs. In addition, atoms, represented by i, are all completely independent, except with respect to write-collision, so this kind of task division does not produce any extra overhead, and we do not need to repeat the calculation. This is in full compliance with the requirements of task division.

The problem of write-collision remains. The solution is quite similar to the one we introduced before. We allocate a large contiguous array in each computing unit and cut it into many segments. Each thread has access to its own segment. When the calculations are complete, we run a process on each computing unit to make a single non-bonded force array which is much shorter. We then download the array to the CPU.

## V. Acceleration of GROMACS using Multi-nodes

After we complete the acceleration on a single node, the method begins to appear more practical. This approach appears with a good performance and it can be shown that it can meet the requirements of calculation in tests with a small number of atoms. However, we need to also accelerate the process across nodes not only to satisfy the calculation requirements for large numbers of atoms, but also for better extendibility. In this section, we introduce the main method, which we designed and implement for GROMACS across nodes with offload mode. In addition, we also present the challenges encountered in the implementation and their corresponding solutions.

When we perform a cross-node calculation, the task-partition in GROMACS cuts an atomic system into grids. Every computing node calculates the atom's non-bonded forces that

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCBB.2017.2713362, IEEE/ACM Transactions on Computational Biology and Bioinformatics



are in its range. In this way, we can reduce that number of atoms on each node so that we can reduce the workload on each node to promote the performance. However, there are forces between different grids. In the practical calculation, the atoms need to calculate the force within a grid; at the same time, the force across the grids also needs to be calculated. These two kinds of force are called the local non-bonded force and non-local non-bonded force. Hence, when computing the force across nodes, the main challenge is that we need to do the calculation twice, local ones and non-local ones. GROMACS has dynamic load balancing and atom migration, which means that the number and type of atoms in every single grid changes all the time, so that different nodes need to communicate with each other by MPI to exchange the atom information. This process produces more communication overhead and we cannot reuse the spatial data. Hence, when we apply GROMACS across nodes, we should be careful with communication and data control. Furthermore, we need to transmit the data, local non-bonded force, and non-local non-bonded force, to the MIC in the same simulation step. The temporal relation of transmission across nodes is much more complex than before, where we just needed to upload the data to a single node. Although the number of atoms of each node is relatively reduced, the total communication overhead is not lessened simultaneously; sometimes it is even increased. In addition, the task division of GROMACS across nodes can be a great trouble and the acceleration performance will degrade when the number of nodes increases in theory. The problems above are very difficult to solve, which leads to higher requirements for us to accelerate GROMACS across nodes with offload mode.

Our solution is to design different transmission modes for different kinds of data. A part of the data that we use when calculating the non-bonded force is static, which does not change with the process of molecular dynamic simulation. This type of data can be used by both local and non-local non-bonded force calculations, so they need to be uploaded only once, at the beginning of the process. This type of data includes a structure of the atom information. The second type of data has different content for local and non-local non-bonded force calculations; correspondingly, it is stored in different positions but has the same size. When we do different calculations, we only need to move the pointer to a different memory address to transmit the data for computing. Because the size of each data array is the same, there is no need to allocate memory again. This type of data includes the atom pair information, atom position, atom charge, and other information. The third kind of data, whose size is related to atom number, varies with the process of simulation, and represents the local and non-local non-bonded force. Hence, the space it takes must be adjusted, which means that the space cannot be reused. The strategy we apply here is to make segregative room for both local and non-local non-bonded force, calling the pointer to do the data transmission. During the simulation, we need to reapply space for the atoms, which migrate across grids, with GROMACS's dynamic load balancing. Finally, the last kind of data does not exist in the original GROMACS. This part of the data, which are intermediate variables, are mainly used to achieve optimization strategies, including for some arrays of force. Its feature is that it is not only related with the atom number, which means they need reallocation in the simulation process, but its memories are completely controlled by text.

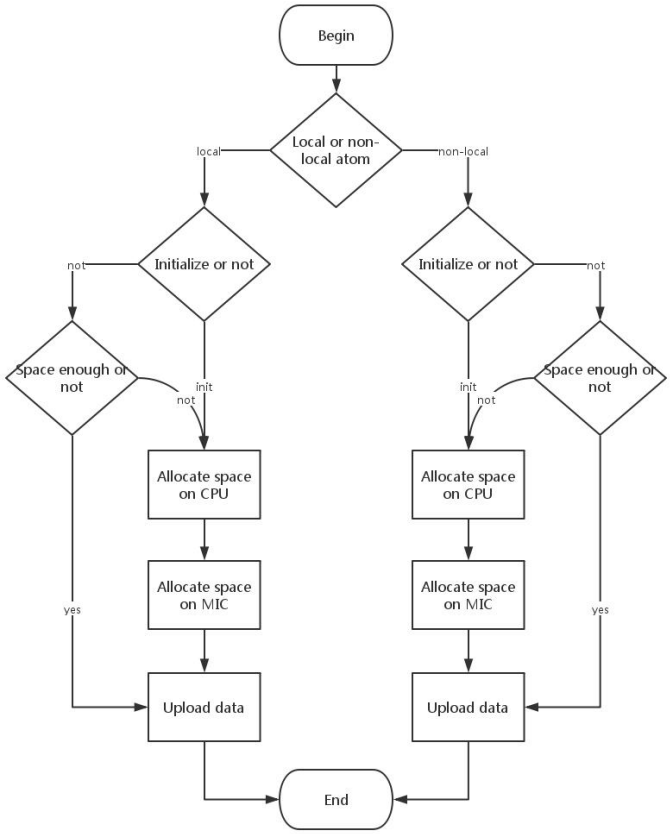


**Figure 3** The fourth transmission mode with additional allocations

Here we will only show the fourth data transmission which can be regard as the most typical one and it is the most complicated. We summarize it in Figure 3.

The fourth transmission mode (Figure 3)is needed for additional allocations on both CPU and MIC side.

## VI. PERFORMANCE EVALUATION

In this section, we introduce the experiments and evaluation of our method. In addition, we present some performance results and data.

### *A. Platform*

Our experimental platform is the Tianhe-2 supercomputer in Guangzhou Center. Tianhe2 has 16,000 computing nodes, and each node consists of two Xeon E5 12 core CPU processors with three Xeon Phi57 core MIC coprocessors. Each node has 64GB CPU memory and 8GB MIC coprocessor memory, so each node's memory is 88GB in total. Other specifications of the CPU and MIC are listed in Table 1.

**Table 1:** Experimental equipment specifications

| | CPU | MIC |
|---|---|---|
| Clock Frequency(GHz) | 2.20 | 1.10 |
| VPU width(bits) | 256 | 512 |
| L1/L2/L3 Cache(KB) | 32/256/30720 | 32/512 |
| Memory Size(GB) | 64 | 8 |

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCBB.2017.2713362, IEEE/ACM Transactions on Computational Biology and Bioinformatics



### B. Evaluation of Offload Mode on a Single MIC

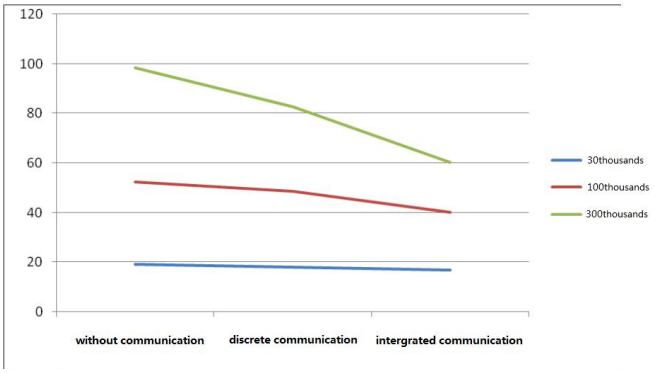


**Figure 4** Results with and without communication acceleration

As mentioned before, integrated communication is a method that gathers the segregated data, which is for multiple threads, into a large and continuous array. In this way, we can complete data transmission once to reduce the time of communication, thereby reducing the overhead.

On the Tianhe2 computing platform, we set up test simulations of 200 steps on a single node with two CPUs (24 cores in total) and a single MIC card. The test includes 30,000, 100,000, and 300,000 atomic systems correspondingly with and without integrated communication. The test results are shown in Figure 4.

Figure 4, the y-axis is the time that tasks take, in seconds, and the x-axis represents the different versions of GROMACS. "Without communication" refers to the core code running on a CPU without any acceleration, which is the original version of GROMACS, is the slowest. The other two optimized versions use the MIC to calculate the non-bonded force. The communication overhead is quite large, but the efficiency is greatly improved compared with the original version. It can be seen that because of the communication integration and optimization, GROMACS obtains a notably better performance in all atomic systems. In addition, the integrated communication version is the best overall.

We also evaluate the overall performance of our new program. The platform is again the Tianhe2, with two CPUs and a single MIC, and the simulation is run for 200 steps.The performance is shown in Figure 5.

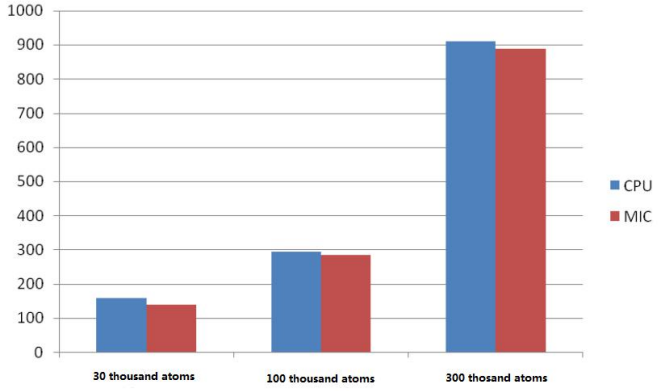


**Figure 5** Overall system performance of off-load mode

In Figure 5, the y-axis represents the running time in seconds, and the x-axis represents atom numbers in the systems. We can see that the calculation performance of a single MIC is essentially the same as the performance of both CPUs, because the MIC's frequency is slower than that of the CPU and adds too much extra cost, although our methods, integrated communication and solving the write-collision, have achieved reasonable performance. We will present further results in the next section.

### C. Evaluation of CPU&MIC collaboration

In order to accelerate the GROMACS, we need to solve two more problems. The first one is to lessen the additional time cost, data copy overhead and communication overhead. The method uses a new data flow, which we have introduced before, to decrease the extra cost. The other one is aim to make a perfect task partition. We need to divide the task into different computing units for the CPU and MIC, to reduce the workload on each unit and then improve the performance. To clarity, GROMACS with SIMD acceleration is referred to as the "original version," which obtains a practical performance when it runs on the Tianhe2.

To solve the first problem, we use a new data flow to speed up GROMACS, which can effectively decrease the time cost of extra communication and data transportation. In addition, it can increase the ratio of calculation. We performed 200 steps in different atomic systems in this test, and the results are shown in Figure 6.

Figure 6 shows that the new data flow for GROMACS only performs the worst, because it runs in serial. The new data flow with MIC parallel acceleration has the best performance. In addition, we can see the calculation time cost of the parallel new data flow is much less than that in the original version, which shows a good acceleration improvement.

To obtain even better performance, we need to use the CPU and MIC at the same time to do the calculations. Task allocation among CPUs and MICs is quite similar to that among MICs. In addition, the calculation on the CPU will not bring any extra overhead because there is no need to transmit the data and there is no write-collision either.

Because of the different computing abilities of the CPUs and MICs, in actual calculations, we need to change the workload on the CPU and MIC according to task type. We set 200 steps, respectively to 30,000, 100,000, and 300,000 atoms of the system, to do the test for all task allocation conditions. The performance and results are shown in Figure 7.

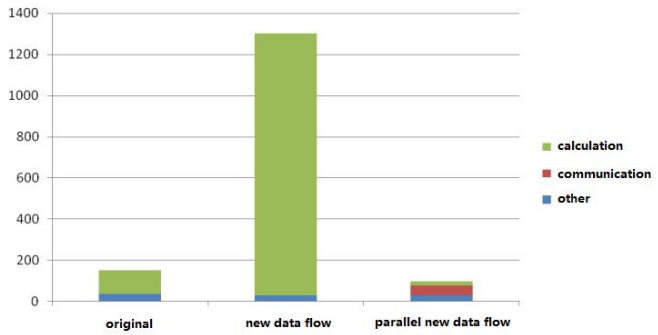


**Figure 6** Results for the new data flow

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCBB.2017.2713362, IEEE/ACM Transactions on Computational Biology and Bioinformatics



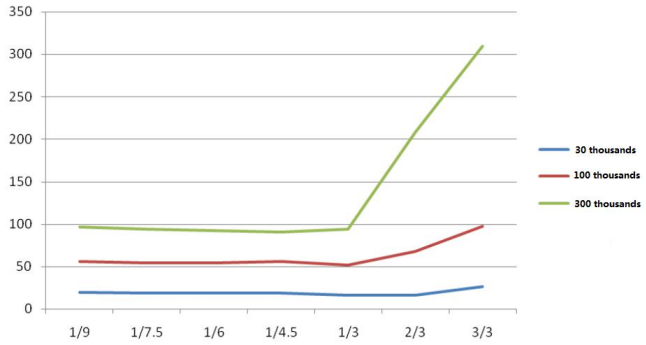


**Figure 7** Performance of different task load allocation on the CPUs and MICs

The x-axis is the ratio of workload on CPU and MIC, the y-axis is the time to complete the task, in seconds. As can be seen in the figure, no matter what kind of atomic system is used, the efficiency changes along with the changing of the ratio of workload between the CPUs and MICs. Moreover, one of the highest efficiency is achieved. In addition, it can be seen that the peak of different atomic systems are different, which is because the ratio of the calculations differs among the three atomic systems.

Through the use of the new data flow and multi-unit optimization, GROMACS can achieve good acceleration with the offload mode. We performed the test with a 50,000 steps simulation on a single node that has two CPUs and three MICs. The test results are shown in Figure 8.

In Figure 8, the y-axis is the program's running time. As the number of atoms increases, increasing the amount of calculation, the acceleration effect is more significant. The 300,000-atomic system obtains the best speed-up compared to the original GROMACS. In addition, for all atomic systems, the CPU/MIC collaborative computing always has better efficiency.

### *D. Evaluation of Multi-node GROMACS*

Compared to the offload GROMACS running on a single node, which we evaluated in the last section, multi-node GROMACS has more computing resources and correspondingly better efficiency; however, the acceleration effect cannot be as good as for a single node. Hence, we set 2,000 steps and 100,000 atoms to test the performance of multi-node GROMACS. The results are presented in Figure 9.

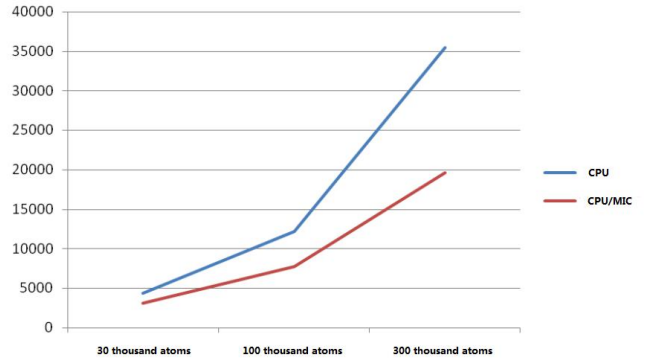


**Figure 8** Performance for a 50000-step simulation on a single node

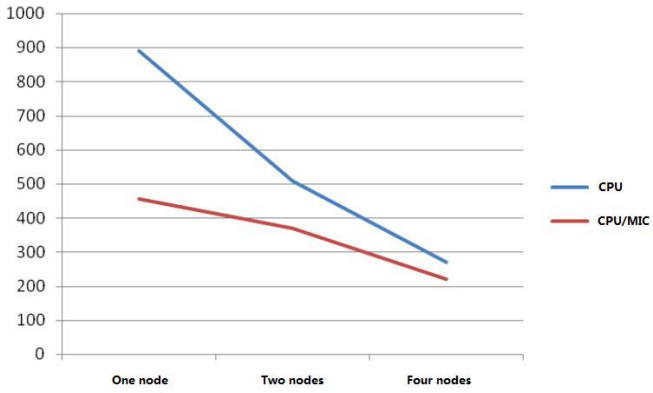


**Figure 9** Performance of multi-node GROMACS

The y-axis represents the time, in seconds, and the x-axis represents the number of nodes. As the number of nodes increases, the speed of both the original version and accelerated version are significantly improved. However, the extra overhead of the accelerated version is also increased, decreasing the calculation ratio, so that the speed-up decreases gradually. Overall, compared to the original GROMACS, the offload mode still achieves a significant acceleration effect.

## VII. Conclusion

Molecular dynamics simulation is a widely applied method in materials science, life science, medicine, and other fields. It utilizes Newton's classical mechanics to calculate the trajectory of microscopic particles and simulate the movement of each atom in the system through an iterative process, so that we can obtain the macroscopic properties of the atomic system. GROMACS is one of the most commonly used molecular dynamics simulation software packages. It is popular because it is easy to use and powerful. Currently, GROMACS has been through dozen times of acceleration and optimization. The current acceleration is quite good, including the OPENMP&MPI and GPU&SIMD acceleration. However, after the optimization, GROMACS may sometimes be limited by insufficient computing resources, because the number of iterations are often quite large, resulting in a long calculation time. In addition, we found that there is no efficient version of GROMACS running on MICs in offload mode, so we propose a new method to accelerate GROMACS this way.

Given a full understanding of the current GROMACS acceleration methods, we presented a method that implements GROMACS on MICs in offload mode in Section III. We proposed to transplant the calculation of non-bonded forces to the MIC, and utilize a large array to gather the discrete data. We tested our method and found that it achieved very good performance.

Overall, our method uses three main steps to complete the acceleration for GROMACS on MIC with offload mode, and we ensured the acceleration effect and practicability by experiments carried out on the Tianhe2. We also analyzed varieties of challenges presented by the molecular dynamics simulation algorithm and provided practical solutions. In conclusion, we can say that our new framework and method

greatly improve the efficiency of GROMACS and we can guarantee the extendibility of our method.

## ACKNOWLEDGMENT

We would like to appreciate Herman Berendsen's group, department of Biophysical Chemistry of Groningen University and other developers all over the world for providing the source code of GROMACS, and Prof. Hualiang Jiang, Weiliang Zhu, and Jin'an Wang from Shanghai Institute of Materia Medica, Chinese Academy of Sciences (SIMM) for providing related test data and explaining GROMACS. And we would also like to appreciate Prof. Shengzhong Feng, Zhibin Yu, and Dr. Yanjie Wei from Shenzhen Institutes of Advanced Technology (SIAT) for providing advice of algorithm and optimization. This work is supported by NSFC Grant 61272056, U1435222, and 1133005.

## REFERENCES

[1] Gromacs. http://www.gromacs.org/About_Gromacs. 2015.
[2] GPU acceleration. http://www.GROMACS.org/GPU_acceleration. 2014
[3] Alder B J, Wainwright T E. Phase Transition for a Hard Sphere System [J]. Journal of Chemical Physics. 1957, 27 (5): 1208–1209.
[4](Intel) M P. GROMACS for Intel® Xeon Phi™ Coprocessor. https://software.intel.com/en-us/articles/GROMACS-for-intel-xeon-phi-coprocessor. 2014
[5] Wang H, Peng S, Zhu X, et al. A Method to Accelerate GROMACS in Offload Mode on Tianhe-2 Supercomputer [C]. In Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on. 2015: 781–784
[6] Wikipedia -Xeon phi
[7] www.intel.com
[8] Martyna G J, Tobias D J, Klein M L. Constant pressure molecular dynamics algorithms [J]. Journal of Chemical Physics. 1994, 101 (5): 4177–4189.
[9] Stillinger F H, Rahman A. Improved simulation of liquid water by molecular dynamics [J]. Chemical Physics. 1974, 60 (4): 1545–1557.
[10] NOSÉ S. A molecular dynamics method for simulations in the canonical ensemble [J]. Molecular Physics An International Journal at the Interface Between Chemistry Physics. 1984, 52 (2): 255–268.
[11] Plimpton S. Fast parallel algorithms for short-range molecular dynamics [C]. In Other Information: PBD: May 1993. 1993: 1–19.
[12] Spoel D V D, Lindahl E, Hess B, et al. GROMACS: Fast, flexible, and free [J]. Journal of Computational Chemistry. 2005, 26 (16): 1701ndash;1718.
[13] Spoel D V D, Hess B. GROMACS—the road ahead [J]. Wiley Interdisciplinary Reviews Computational Molecular Science. 2011, 1 (5): 710–715.
[14] David V D S, Van Maaren P J, Caleman C. GROMACS molecule liquid database [J]. Bioinformatics. 2012, volume 28 (5): 752–753.
[15] Van der Spoel D; Hess B; Lindahl E P S P S S R L P B P A R S M S J K P. GROMACS 4.5: a high-throughput and highly parallel open source molecular simulation toolkit. [J]. Bioinformatics. 2013, 29 (7): 845–854.
[16] Kutzner C, David V D S, Fechner M, et al. Speeding up parallel GROMACS on high-latency networks. [J]. Journal of Computational Chemistry. 2007, 28 (12): 2075–2084.
[17] Jeffers J, Reinders J. Intel Xeon Phi Coprocessor High Performance Programming [J]. Journal of Computer Science Technology. 2013.
[18] Misra S, Pamnany K, Aluru S. Parallel Mutual Information Based Construction of Whole-Genome Networks on the Intel (R) Xeon Phi (TM) Coprocessor [J]. IEEE Xplore. 2014: 1–1.
[19] S X X, C D R. Probing Single Molecule Dynamics [J]. Science. 1994, 265 (5170): 361–364.
[20] Sellis D, Vlachakis D, Vlassi M. Gromita: a fully integrated graphical user interface to gromacs 4. [J]. Bioinformatics Biology Insights. 2009, 3 (3): 99–102.
[21] Berendsen H J C, Spoel D V D, Vannen R. GROMACS: A message-passing parallel molecular dynamics implementation [J]. Computer Physics Communications. 1995, 91 (1): 43–56.
[22] Lindahl E, Hess B, Spoel D V D. GROMACS 3.0: a package for molecular simulation and trajectory analysis [J]. Journal of Molecular Modeling. 2001, 7 (8): 306–317.
[23] Hess B, Kutzner C, Spoel D V D, et al. GROMACS 4: Algorithms for Highly Efficient, Load-Balanced, and Scalable Molecular Simulation [J]. J.chem.theory Comput. 2008, 4 (3): 435–447
[24] Olivier S, Prins J, Derby J, et al. Porting the GROMACS Molecular Dynamics Code to the Cell Processor [C]. In Parallel and Distributed Processing Symposium, International. 2007: 370–370.
[25] Yan L, Guo L, Li X. Research in porting MD simulation software GROMACS on GPGPU based on CUDA [J]. Computers Applied Chemistry. 2010, 27 (12): 1603–1606.
[26] Dijk M V, Wassenaar T A, Bonvin A M J J. A Flexible, Grid-Enabled Web Portal for GROMACS Molecular Dynamics Simulations [J]. Journal of Chemical Theory Computation. 2012, 8 (10): 3463–3472.
[27] Lukat G, Krüger J, Sommer B. APL@Voro: A Voronoi-Based Membrane Analysis Tool for GROMACS Trajectories [J]. Journal of Chemical Information Modeling. 2013, 53 (11): 2908–2925.
[28] Tsuyoshi T, Tomoshi K, Shoji T. On easy implementation of a variant of the replica exchange with solute tempering in GROMACS. [J]. Journal of Computational Chemistry. 2011, 32 (7): 1228–1234.
[29] Poghosyan A H, Yeghiazaryan G A, Gharabekyan H H S A A. The GROMACS and NAMD software packages comparison [J]. Communications in Computational Physics. 2006, 1 (4): 736–743.
[30] Bussi G. Hamiltonian replica-exchange in GROMACS: a flexible implementation [J]. Molecular Physics An International Journal at the Interface Between Chemistry Physics. 2013, volume 112 (3): 379–384.
[31] Abraham, Mark J. Performance enhancements for GROMACS nonbonded interactions on BlueGene [J]. Journal of Computational Chemistry. 2011, 32 (9): 2041–2046.
[32] Surhone L M, Timpledon M T, Marseken S F, et al. Visual Molecular Dynamics [J]. Betascript Publishing. 2010.
[33] Nosé S, Klein M. Constant pressure molecular dynamics for molecular systems [J]. Molecular Physics An International Journal at the Interface Between Chemistry Physics. 1983, 50 (5): 1055–1076.