

# Topology-Aware Parallel Molecular Dynamics Simulation Algorithm

Hideaki Kikuchi<sup>a</sup>, Bijaya B. Karki<sup>a,b</sup> and Subhash Saini<sup>c</sup>  
karki@csc.lsu.edu

<sup>a</sup>Department of Computer Science, <sup>b</sup>Department of Geology and Geophysics, Louisiana State University,  
Baton Rouge, LA 70801, USA

<sup>c</sup>NASA Systems Division, NASA Ames Research Center

## Abstract

We have developed the topology-aware parallel molecular dynamics (TAPMD) algorithm, in which the processors are rearranged automatically according to resource topology so as to minimize the cost required for the simulation. It is demonstrated that TAPMD can reduce the communication time to less than half compared to the time in the worst case on a distributed PC clusters. This improvement is expected to be more significant when the communication time is dominating over the total wall-clock time and when the resource topology consists of more types of the clusters. Additional tests involving several clusters having different types of connections as well as different types of processors are under progress.

**Keywords:** Distributed and parallel computing, topology awareness, parallel molecular dynamics

## 1. Introduction

Distributed and parallel computing could revolutionize scientific computational research [1]. The availability of useful middleware/software, such as Globus (<http://www.globus.org/>), MPICH-G2 (<http://www3.niu.edu/mpi/>), etc., and inexpensive PC clusters enable us to construct a distributed infrastructure at each research-group level. It is not only increasing the computational capacity but also increasing the diversity of the research through collaboration with expertise of different background. The resulting infrastructure creates a complicated resource topology, which

involves many types of processors with different speeds and a wide variety of intra- and inter-cluster connection links. In the Grid and/or distributed computing environment, the resource manager monitors the status of the resources throughout the entire infrastructure and provides the available resources to the user. Many algorithms have been developed by researchers to achieve high efficiency and scalability of the scientific and engineering parallel programs on supercomputers and/or massively parallel machines, in which each processor is tightly coupled [2-8]. The programs developed in such environments, however, often fail to scale on any distributed computing infrastructure due to the communication bottleneck; only a few applications have been reported their performance and scalabilities on the such resources [9,10]. Furthermore, the arrangement of resources, given by the resource manager in such applications is independent of the resource topology. The lack of the resource information in the applications causes an inefficient use of the resource thereby causing substantial performance degradation. To demonstrate the effectiveness of the resource topology awareness in the simulation, we chose the molecular dynamics (MD) technique, which is one of the most widely used techniques in scientific computing. Over the years, the MD simulations have been so widespread that essentially all types of materials from metals, to semiconductors, to ceramics, to minerals, to biomolecular systems are simulated at atomic scales on a routine basis. One can now simulate very complex, large-size ( $>10^6$  m, or  $>10^{12}$  atoms) and long time ( $>10^5$ sec) phenomena with the MD technique on massively parallel computers [11-13].

This paper describes how the topology-aware

parallel molecular dynamics method improves the performance and efficiency for a given set of resources, compared to the conventional parallel molecular dynamics. In the Section 2, we have described the parallel MD and topology-aware parallel MD programs with case studies. Their implementation and preliminary benchmark results are discussed in the Section 3. The conclusions are given in the Section 4.

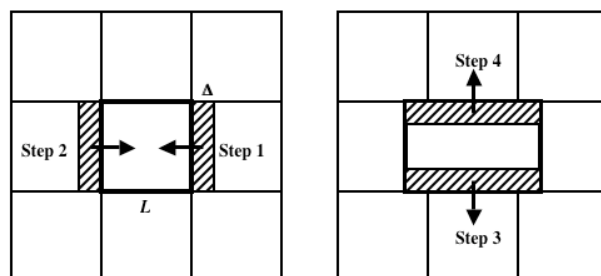
## 2. Topology-Aware Parallel Molecular Dynamics Algorithm

We have developed a topology-aware parallel molecular dynamics (TAPMD) program that rearranges the computing resources to the physical subsystems so as to minimize the communication cost in the simulation.

**Parallel Molecular Dynamics:** The molecular dynamics (MD) obtains the phase-space trajectories of the system (positions and velocities of all atoms at all time) [14, 15]. Atomic force laws for describing how atoms interact with each other is mathematically encoded in the interatomic potential energy,  $E_{\text{MD}}(\mathbf{r}^N)$ , which is a function of the positions of all  $N$  atoms,  $\mathbf{r}^N = \{\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_N\}$ , in the system. For parallelization, we use spatial decomposition [16], in which total volume of the system is divided into  $P$  subsystems of equal volume and each subsystem is assigned to a processor in an array of  $P$  processors. To calculate the force on an atom in the subsystem, the coordinates of the atoms in the boundaries of neighbor subsystems are “cached” from the corresponding processors. After updating the coordinates, some atoms may have moved out of its subsystem, and are “migrated” to the proper processors.

**Load balancing:** In practical MD simulations on a Grid environment, there are two types of load imbalances caused by the irregular atomic distribution and heterogeneous resource topology. We use a computational-space-decomposition approach, which involves a curvilinear coordinate transformation, for the load balancing [17, 18]. The optimal system coordinate is determined to minimize the cost for the load imbalance and communication (see Equation 1).

**3D Spatial Decomposition:** We use a 3D spatial decomposition algorithm in our parallel MD code [19]. The physical system is partitioned into subsystems. The processors are logically arranged according to the topology of the physical subsystems in simple manner, i.e., the atoms located in a particular subsystem are assigned to the corresponding processor. In six-way message passing scheme, each processor, which has twenty-six neighbors, only needs to send/receive data to/from the six face-sharing neighbors. Copies to the other twenty non-face-sharing neighbors are done by a message forwarding mechanism. Figure 1 is a schematic view of four steps in message passing. In the first two steps, the central node receives the atoms in the boundary regions (shaded areas) from the right/left neighbor nodes. Also, the atoms in the boundary regions within the central node are sent to the left/right neighbor nodes. In effect, the physical subsystem corresponding to the central node gets extended. In the next two steps, the boundary atoms in the extended subsystem (the central node) are sent to the lower/upper neighbor nodes while the central node is receiving the boundary atoms from the upper/lower neighbor nodes at same time. Finally, in the last two steps (not shown in Figure 1), the resulting boundary atoms from the previous four steps are sent/received to/from the front/back neighbor nodes by the central node. The communication in this message passing scheme is highly anisotropic: The size of the data to be transferred to the proper neighbor nodes in the step 1 and 2 is the smallest; the size in the step 3 and 4 is larger than that in the step 1 and 2 but smaller than that in the step 5 and 6. With the spatial decomposition algorithm (with a finite cutoff imposed), the computational cost scales as  $(N/P)$  while the communication cost scale in proportion to  $(N/P)^{2/3}$  for an  $N$ -atom system.



**Figure 1.** A schematic view of the first four steps in the six-way message-passing scheme.

**Topology-aware parallel MD program:** In a common parallel MD program, the way the processors are assigned to different subsystems is independent of how a given set of resources are physically distributed. However, such a simple choice often fails to utilize the resources efficiently, especially in the Grid and/or distributed environment, due to their heterogeneous nature. In the MD program, the communications required to send/receive atoms across the processors have to be synchronized at each MD step. Then, a cost function of the program can be written by following expression:

$$\Phi = \max(\alpha_i) + \max(\varphi_i) \text{ for } i = 0, 1, \dots, P-1 \quad (1)$$

Here  $P$  is the total number of processors under consideration. The first term represents the computational cost function for each processor:  $\alpha_i$  is the weight representing computational cost of the  $i$ -th processor, which is proportional to the ratio between the number of atoms assigned to the processor,  $N_i$  and the speed of the processor,  $G_i$ :

$$\alpha_i \propto \frac{N_i}{G_i}. \text{ And } \varphi_i = \sum_{s=0}^5 \beta_{s,i} \text{ represents the}$$

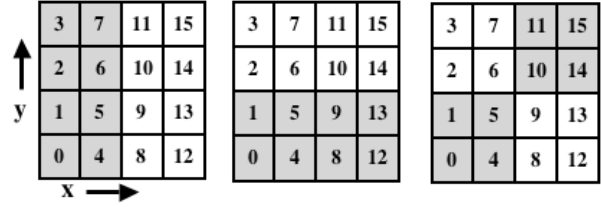
communication cost function on each processor:  $\beta_{s,i}$  is the weight for the communication cost of the  $i$ -th processor along the  $s$ -direction, which is proportional to the ratio between the amount of the data to be transferred to the proper neighbor processor along the  $s$ -direction,  $D_{s,i}$  and the bandwidth between the  $i$ -th processor and the neighbor processors along the  $s$ -direction,  $B_{s,i}$ :

$$\beta_{s,i} \propto \frac{D_{s,i}}{B_{s,i}}. \text{ To obtain better performance using the}$$

same resource, the mapping between the processors and the subsystems has to be generated so as to minimize  $\Phi$  at the beginning of the MD simulation.

Figure 2 illustrates three distinguishable cases for the resource topology consisting of two clusters, A and B, each with 8 processors. A total of 16 processors thus need to be allocated by a resource manager. For the sake of simplicity, we partition the physical system into  $4 \times 4 \times 1$  subsystems. The number in the Figure 2 denotes the index of the subsystem varying from 0 to 15. The subsystems in the shaded and white areas are assigned to the processors in the clusters A and B, respectively. Typically, the connection speed between the processors within the cluster is much higher than

the speed between the clusters, i.e., the intra-cluster bandwidth is much larger than the inter-cluster bandwidth. The first case (left in Figure 2) has three face-sharing neighbors inside the same cluster along both  $x$ - and  $y$ -directions but has only one face-sharing neighbor across the two clusters along the  $x$ -direction. The second case (center in Figure 2) has again three intra-cluster face-sharing neighbors along the  $x$ -,  $y$  directions and one inter-cluster face-sharing neighbor along the  $y$ -direction. However, the third case (right in Figure 2) has two intra-cluster and two inter-cluster face-sharing neighbors along both  $x$ - and  $y$ -directions. In the six-way message passing scheme (see Figure 1), the data transfers across the boundary along the  $y$ -direction is larger than that along the  $x$ -direction but smaller than that along the  $z$ -direction.



**Figure 2.** Topology of the resources and the physical system: the subsystems in the shaded and white areas are allocated to the processors in the cluster A and B, respectively.

**Table 1.** Weights for different communication directions and bandwidths.

	x-direction	y-direction	z-direction
1 Gbps	0.01	0.012	0.045
100 bps	0.1	0.12	0.45
10 Mbps	1.0	1.2	4.5

We have used the following weights for the test runs in this study:  $\alpha_i = 5$ , assuming that the speed among all allocated processors is the same. The weights  $\beta_{s,i}$  shown in the Table 1 were obtained from several runs and they are consistent with the communication times obtained from the test runs (see the Section 3). The connection bandwidth between the cluster A and B is 10 Mbps, and the bandwidth within the both clusters is 1 Gbps. Then,

the calculated cost function values are 6.03, 6.23 and 7.22 for the first, second and third case, respectively. These numbers imply that the topology case (left in Figure 2) is the best choice among the three selections. This is, in fact, coincident with the topology obtained from our topology-aware MD scheme.

Now we present a relatively more complicated test case by considering a system of three clusters. As in the previous two-cluster case, the program requests a total of 16 processors. Also the physical system is partitioned to the 4x4x1 subsystems. However, in this case, the resource manager provides four processors from cluster A (light-shaded areas), three processors from cluster B (dark-shaded areas) and the other nine processors from cluster C (white area). Again assume that CPU speed is the same among all processors. The intra-cluster bandwidth of all three clusters is 1 Gbps, the inter-cluster bandwidths are 100 Mbps between the clusters A and B, 10 Mbps between the clusters A and C, and 10 Mbps between the clusters B and C. This mimics a real world situation in which the cluster A and B are located at the same institute but not the cluster C.

7.24	8.41	5.04	6.03	7.02	7.22	6.23	7.22
6.12	7.31	6.03	6.03	7.02	6.03	4.04	6.03
6.12	6.12	6.03	6.03	7.02	6.22	6.23	7.22
6.12	7.31	6.03	6.03	5.22	7.51	7.42	7.51
7.84	6.03	7.42	6.03	7.02	7.22	6.23	6.03
7.82	7.02	8.21	7.02	7.02	7.22	5.04	6.03
7.82	7.2	7.02	7.02	6.12	7.31	7.22	6.03
7.82	7.02	8.21	7.02	6.12	6.22	8.41	7.02

**Figure 3.** The value of the cost function  $\Phi$  on each processor for four given topologies using three clusters. The physical subsystems in the light- and dark-shaded areas are assigned to the processors in the cluster A and B, respectively. The other subsystems (white areas) are distributed among the processors in the cluster C.

The values of the cost function  $\Phi$  on each processor obtained by using the Table 1 and  $\alpha_i = 5$  are given in the cells in the Figure 3. The cost function gives 8.41 for the first case (upper left); 7.51 for the second case (upper right); 8.21 for the third case (lower left) and 8.41 for the fourth case (lower right). The best choice among the four selections is the second case (upper right in Figure 3). To find the best topology for the MD simulation is now non-trivial. The difference between the best and the worst cases become more significant when the time required for the communication is comparable to the total wall-clock time. Topology-aware parallel MD, which automatically generates the best mapping between the processors and the subsystems, is necessary in such case.

### 3. Implementation and Performance Measurements

The proposed topology-aware mapping generator between the processors and the physical subsystems can be easily encoded in the parallel MD program. It is highly modular and needs to be called at the beginning of the program. We have implemented the topology-aware parallel molecular dynamics algorithm within a single-program multiple-data programming paradigm, in which all the processors execute the same program on different datasets using the MPI and Fortran 90.

In the TAPMD program, all the communication tasks are associated with a single MPI communicator and all the processes are assigned to unique (processor) IDs. The IDs are determined so as to minimize the cost function  $\Phi$ , while the ID in normal parallel MD program is usually set to identical to rank given in the MPI. All the information required for the topology-aware mapping process, such as processor speed, connection bandwidth, number of allocated processors inside the  $i$ -th cluster  $P_i$ , etc., are obtained from a database according to the hostname. The hostnames are obtained by calling `MPI_Get_processor_name` in the program. The database, which has extensive information about the available resources, can be created manually before the test runs.

Exploring all possible mapping topologies could be extremely exhaustive so it could introduce an unwanted huge overhead in the simulation especially when the simulation requires a large number of processors. Therefore, in the mapping process, we first construct objects such that each processor has to share at least one face with a neighboring processor from the same cluster, i.e., the number of processors in the object is equal to the number of processors in the cluster. Then objects are constructed with the clusters that have the second largest number of processors. Finally, the processors in the cluster that has the largest number of processors are mapped to the remaining subsystems. With these constraints, all possible mapping topologies can be explored in a more efficient way to minimize the cost function.

We have implemented the TAPMD code on two PC clusters. Each cluster has eight processors with the same CPU speed (Pentium Xeon 550 MHz). The clusters are interconnected via 10 Mbps, while the processors within the cluster were connected through 1 Gbps. Three topologies mentioned in the previous section (Figure 2) are used for the preliminary test runs. We have used a Lennard-Jones potential. The length of the partitioned subsystem and the width of the boundary region are set at  $L = 27.5\sigma$  and  $\Delta = 5\sigma$ , respectively (see Figure 1), where  $\sigma$  is one of the Lennard-Jones parameters. The dynamic management of distributed linked cell list makes the program scale linear. The entire system consists of 262,144 atoms. The three topologies used in this study were manually configured through the machinefile.

The Table 2 shows the measured values for the wall-clock, CPU and communication times per 20 MD steps for the three cases (see Figure 2) and one case using the topology-aware parallel MD program. The CPU times are the same within the reported uncertainties for all four cases. The communication time in the second case is larger than that in the first case, but smaller than that in the third case as we discussed in the previous section. The ratios of the communication time to the wall-clock time are 0.12, 0.16 and 0.24 for the first, second and third cases, respectively. The cost of the communication required for sending/receiving the data across the neighboring nodes in the third case is two times more expensive

than the cost in the first case. The best topology obtained directly from topology-aware parallel MD is, in fact, identical to that of the first case; the wall-clock, CPU and communication times are the same between the first case and TAPMD within the reported uncertainties.

**Table 2.** Wall-clock, CPU and communication times (in seconds) per 20 MD steps for three cases and TAPMD using two clusters.

	Wall-clock	CPU	Comm.
First	109.01±1.55	94.07±0.69	13.36±0.15
Second	113.05±0.52	93.78±0.65	17.95±0.15
Third	125.21±0.76	93.60±0.70	30.62±0.25
TAPMD	109.23±0.86	94.37±0.62	13.38±0.14

## 4. Conclusions

We have demonstrated the performance improvement by introducing the topology-aware parallel MD simulation over the conventional parallel MD simulation on a distributed PC clusters. The proposed TAPMD method automatically chooses the best topology so as to minimize the performance degradation due to the communication bottleneck. Performance measurements have shown that the total communication time in the simulation was reduced to less than half of that of the worst case. The improvement using TAPMD becomes more significant when the communication time is dominating over the total wall-clock time. It enables us to study long time phenomena, such as dislocation dynamics and protein folding by increasing the number of processors  $P$  while the total number of atoms  $N$  is kept same to reduce the computational time over the distributed computing resources. Larger tests involving more many clusters should consider not only different types of connections but also the different types of processors.

## Acknowledgements

This work was supported by the NASA (NN A05CP84) and NSF Career (EAR 0347204) grants.

## References

- [1] I. Foster and C. Kesselman. *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, San Francisco, 2003.
- [2] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73:325-348 (1987).
- [3] A. Toukmaji, D. Paul, and J. Board, Jr. Distributed particle-mesh Ewald: a parallel Ewald summation method, In *Proceedings of Parallel and Distributed Processing Techniques and Applications* (1996)
- [4] M. E. Tuckerman, D. A. Yarne, S. O. Samuelson, A. L. Hughes, and G. J. Martyna. Exploiting multiple levels of parallelism in molecular dynamics based calculations via modern techniques and software paradigms on distributed memory computers. *Computer Physics Communications*, 128:333-376 (2000).
- [5] F. F. Abraham, J. Q. Broughton, N. Bernstein, and E. Kaxiras. Spanning the length scales in dynamic simulation. *Computational Physics*, 12:538-546 (1998).
- [6] A. Nakano, M. E. Bachlechner, R. K. Kalia, E. Lidorikis, P. Vashishta, G. Z. Voyiadjis, T. J. Campbell, S. Ogata, and F. Shimojo. Multiscale simulation of nanosystems. *Computing in Science and Engineering*, 3(4):56-66 (2001).
- [7] A. Nakano, R. K. Kalia, P. Vashishta, T. J. Campbell, S. Ogata, F. Shimojo, and S. Saini. Scalable atomistic simulation algorithms for materials research. In *Proceedings of Supercomputing 2001*, ACM, New York, 2001.
- [8] H. Kikuchi, R. K. Kalia, A. Nakano, P. Vashishta, F. Shimojo, and S. Saini. Scalability of a low-cost multi-teraflop Linux cluster for high-end classical atomistic and quantum mechanical simulations, In *Proceedings of International Parallel and Distributed Processing Symposium 2003*.
- [9] G. Allen, T. Dramlitsch, I. Foster, N. T. Karonis, M. Ripeanu, E. Seidel, B. Toonen. Supporting efficient execution in heterogeneous distributed computing environments with Cactus and Globus. In *Proceedings of Supercomputing 2001*, ACM, New York, 2001.
- [10] H. Kikuchi, R. K. Kalia, A. Nakano, P. Vashishta, H. Iyetomi, S. Ogata, T. Kouno, F. Shimojo, K. Tsuruta, and S. Saini, Collaborative simulation grid: multiscale quantum-mechanical/classical atomistic simulations on distributed PC clusters in the US and Japan. In *Proceedings of Supercomputing 2002*.
- [11] A. Nakano, M.E. Bachlechner, T, J, Campbell, R. K. Rajiv, A. Omeltchenko, K. Tsuruta, P. Vashishta, S. Ogata, I. Ebbsjo, and A. Madhukar, Atomistic simulation of nanostructured materials, *IEEE Computational Science & Engineering*, 5(4):68-78 (1998).
- [12] V. S. Pande, I. Baker, J. Chapman, S. P. Elmer, S. Khaliq, S. M. Larson, Y. M. Rhee, M. R. Shirts, C. D. Snow, E. J. Sorin, and B. Zagrovic, Atomistic Protein Folding simulations on the submillisecond time scale using worldwide distributed computing, *Biopolymers*, 68:91-109 (2003).
- [13] V. Yenaguntla, B. Karki, and H. Kikuchi, A parallel molecular dynamics algorithm for polycrystalline minerals, In *The 2005 Int'l Conf. on Modeling, Simulation and Visualization Methods*, 201-207 (2005).
- [14] S. J. Plimpton. Fast parallel algorithm for short-range molecular dynamics, *Journal of Computational Physics*, 117:1-19 (1995).
- [15] J. C. Phillips, G. Zheng, S. Kumar, and L. V. Kalé, NAMD: Biomolecular simulations on thousands of processors, In *Proceedings of Supercomputing 2002*.
- [16] A. Nakano, R. K. Kalia, and P. Vashishta. Multiresolution molecular dynamics algorithm for realistic materials modeling on parallel computers. *Computer Physics Communications*, 83:197-214 (1994).
- [17] A. Nakano. Multiresolution load balancing in curved space: The wavelet representation. *Concurrency: Practice and Experience*, 11:343-353 (1999).
- [18] A. Nakano and T. J. Campbell. An adaptive curvilinear-coordinate approach to dynamic load balancing of parallel multi-resolution molecular dynamics, *Parallel Computing*, 23:1461-1478 (1997).
- [19] D. C. Rapaport, Multi-million particle molecular dynamics II. design considerations for distributed processing, *Computer Phys. Commun.*, 62:217 (1991).