Memory-Access Optimization of Parallel Molecular Dynamics Simulation via Dynamic Data Reordering

Manaschai Kunaseth, Ken-ichi Nomura, Hikmet Dursun, Rajiv K. Kalia, Aiichiro Nakano, and Priya Vashishta

University of Southern California, Los Angeles, CA 90089, USA
{kunaseth,knomura,hdursun,rkalia,anakano,priyav}@usc.edu

Abstract. Dynamic irregular applications such as molecular dynamics (MD) simulation often suffer considerable performance deterioration during execution. To address this problem, an optimal data-reordering schedule has been developed for runtime memory-access optimization of MD simulations on parallel computers. Analysis of the memory-access penalty during MD simulations shows that the performance improvement from computation and data reordering degrades gradually as data translation lookaside buffer misses increase. We have also found correlations between the performance degradation with physical properties such as the simulated temperature, as well as with computational parameters such as the spatial-decomposition granularity. Based on a performance model and pre-profiling of data fragmentation behaviors, we have developed an optimal runtime data-reordering schedule, thereby archiving speedup of 1.35, 1.36 and 1.28, respectively, for MD simulations of silica at temperatures 300 K, 3,000 K and 6,000 K.

Keywords: Data reordering, memory-access optimization, data fragmentation, performance degradation, molecular dynamics.

1 Introduction

Molecular dynamics (MD) simulation is widely used to study material properties at the atomistic level [2,7,9,10]. One of the major problems on improving performance of MD simulations is to maintain data locality. Since the memory-access pattern in MD simulation is highly non-uniform and unpredictable, the locality optimization problem is challenging.

To address the locality issue, a commonly used method is data reordering, which organizes data of irregular memory-access patterns in memory according to a certain locality metric [3,4,11,13]. However, a further challenge arises from the dynamic, irregular nature of MD computations. Mellor-Crummey *et al.* showed that data reordering and computation restructuring enhance data locality, resulting in the reduction of cache and TLB misses and accordingly considerable performance improvement of MD simulations [6]. The study also suggested the necessity of repeated runtime reorderings, for which the remaining

C. Kaklamanis et al. (Eds.): Euro-Par 2012, LNCS 7484, pp. 781–792, 2012.

problem is how often the reordering should be performed in order to achieve the optimal overall performance. In addition, as the data reordering and computation restructuring incur computational overhead, such reordering cost should be considered to find the optimal reordering frequency.

Runtime data behaviors of MD simulations are closely related to both physical properties of the system being simulated (*e.g.* temperature, diffusion rate, and atomic configuration) and computational parameters (*e.g.* spatial decomposition granularity in parallel MD). Therefore, understanding how these quantities affect the deterioration of data locality is a prerequisite for designing the optimal runtime data-reordering schedule.

To address these challenges, we first introduce a *data fragmentation ratio* metric that quantifies the locality of atom data arrays during MD simulation. Then, we perform an analysis on how MD simulations with different physical/computational characteristics (*e.g.* temperature, diffusion rate, and granularity) impact the data fragmentation ratio and performance deterioration of the system. Based on the data fragmentation analysis, we finally design and evaluate an memory optimization scheme that features an optimal runtime data-reordering schedule during MD simulation.

2 Parallel Molecular Dynamics

Molecular dynamics simulation follows the phase-space trajectories of an N-atom system, where force fields describing the atomic force laws between atoms are spatial derivatives of a potential energy function $E(\mathbf{r}^N)(\mathbf{r}^N = \{\mathbf{r}^1, \mathbf{r}^2, ..., \mathbf{r}^N\}$ is the positions of all atoms). Positions and velocities of all atoms are updated at each MD step by numerically integrating coupled ordinary differential equations. The dominant computation of MD simulation is the evaluation of $E(\mathbf{r}^N)$, which, in the program considered in this paper, consists of two-body $E_2(\mathbf{r}_i, \mathbf{r}_j)$ and three-body $E_3(\mathbf{r}_i, \mathbf{r}_j, \mathbf{r}_k)$ terms [7].

Figure 1(a) shows a schematic of the computational kernel of MD, which employs a linked-list cell method to compute interatomic interactions in $O(N)$ time. Periodic boundary condition is applied to the system in three Cartesian dimensions. Here, a simulation domain is divided into small cubical cells, and the linked-list data structure is used to organize atomic data (*e.g.* coordinates, velocities and atom type) in each cell. Traversing through the linked list, one retrieves the information of all atoms belonging to a cell, and thereby computes interatomic interactions. Thus, the dimensions of the cells are usually determined by the cutoff radius r_c of the interatomic interaction.

The MD program considered in this paper employs spatial decomposition at an outermost level of hierarchical parallelization [8]. Here, the physical system is partitioned into subsystems, and atoms residing in different subsystems are assigned to different compute nodes. When the atomic coordinates are updated according to the time-integration algorithm, some resident atoms may have moved out of the subsystem boundary, and such atoms are migrated to proper nodes.

MD is an archetype of irregular memory-access pattern applications, and this paper addresses two distinct sources of irregularity: (1) physically induced disorder such as atom diffusion or flow; and (2) algorithmically induced disorder such as atom migration between spatial decomposition domains in parallel MD.

3 Data Fragmentation in Parallel Molecular Dynamics

High memory-access penalty is a major obstacle of irregular memory accessing applications, which needs to be mitigated. Our MD application performs data and computation orderings by organizing atom data contiguously in memory in the same order as the linked lists of the cells access them. The reordering algorithm performed in $O(N)$ is shown in Fig. 1(b). Though the spatial locality after data reordering is retained for a while, data get disordered as simulation progresses due to atom movement among cells. How the benefit from data reordering deteriorates as simulation evolves at runtime essentially affects the performance of MD simulations. To quantify the level of fragmentation, we first define a fragmentation measurement ratio metric.

Let $C(i, j, k)$ be a linked-list cell with cell indices i, j, and k in the x, y, and z directions, respectively. Each linked-list cell contains indices of atoms whose coordinates are within their cell dimensions. Before data reordering, atom data are likely scattered in memory as illustrated in Fig. 1(c), and data reordering improves the data locality as shown in Fig. 1(d), where the atom data of the same physical cell volume reside continuously. Fragmentation in the atom data array could occur as follows. Suppose that the a-th atom in linked-list cell C moves to another cell C'. Consequently, the memory block of C' becomes partially fragmented because the a-th atom is not contiguous in memory space with other atoms in C' such that their distances in the memory space exceed the page size. Therefore, any atom moving out of its original linked-list cell introduces fragmentation in memory, which likely causes data translation lookaside buffer (DTLB) misses. To quantify the degree of fragmentation, we thus define a *data fragmentation ratio* as $f = N_{fragment}/N$, where $N_{fragment}$ is the number of atoms whose positions have moved out of the originally ordered cells and N is the total number of atoms. The data fragmentation ratio is between 0 (*i.e.*, all atoms in all cells reside continuously in memory—fully ordered state, see Fig. 1(d)) and 1 (*i.e.*, no atom resides in the original cell—fully disordered state, see Fig. 1(c)). The data fragmentation ratio will be used to quantify fragmentation extensively throughout this paper. Note that the page size used in all experiments is 4 KB.

Data fragmentation in memory is dictated by the dynamics of atoms, and thus understanding the factors that control the atom dynamics would provide an insight on how to prevent the fragmentation. One of the physical factors that are directly related to the dynamics is the temperature, since high temperature drives atoms to move faster and more freely. Among the computational factors, migration of atom data from one node to another across a subsystem boundary in parallel MD also causes fragmentation. The granularity of spatial decomposition

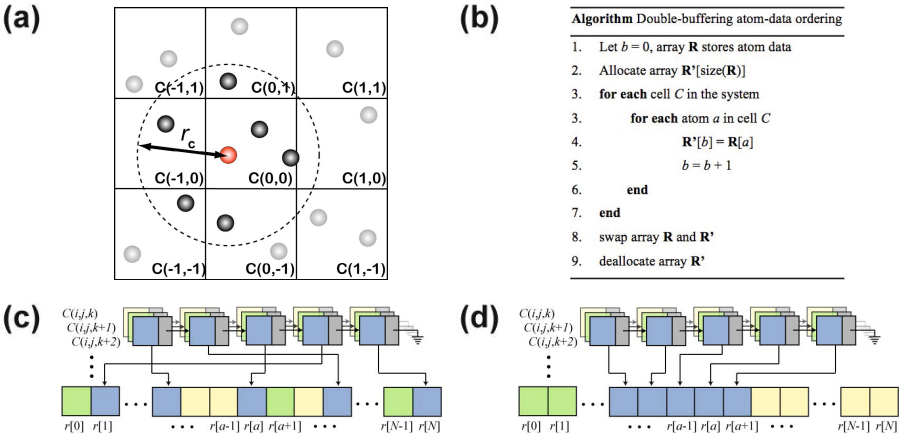

Fig. 1. (a) 2D schematic of the linked-list cell method. The center cell $C(0,0)$ is surrounded by eight neighbor cells. The cell dimensions are often chosen to be the cutoff radius (represented by the two-heads arrow) of interatomic interaction. Only force exerted from atoms within the cutoff radius are computed. (b) Pseudocode of the data reordering algorithm. (c) Schematic of memory layout for atom data in a disordered state and (d) a fully ordered state, where $C(i,j,k)$ is the linked list for the cell with indices i, j, and k in the x, y, and z directions, respectively, $r[a]$ is the data associated with the a-th atom.

(*i.e.* the number of atoms per compute node) is related to the subsystem surface-to-volume ratio, and hence is likely to control the degree of fragmentation via the amount of atom migrations. In the following subsections, we measure and analyze the effects of temperature and granularity on the data fragmentation.

3.1 Temperature Induced Fragmentation

In this subsection, we study the effect of temperature on the data fragmentation during MD simulations on two systems: Silica material [1] and combustion of aluminum nanoparticles [12]. The silica simulations involving 98,304 atoms and 8,000 linked-list cells ($20 \times 20 \times 20$ in x, y, and z directions, respectively) are performed for 3,000 MD time steps using a time discretization unit of 2 femtoseconds. The simulations are performed on a dual quadcore Intel Xeon E5410 2.33 GHz (Harpertown) at the High Performance Computing and Communication (USC-HPCC) facility of the University of Southern California. Temperature is a major physical factor that enhances the diffusion of atoms and hence the disordering of data arrays. To examine the effect of atomic diffusions on the data fragmentation ratio, we thus perform a set of simulations with three different initial temperatures, 300, 3,000, and 6,000 Kelvin (K), starting from the same initial atomic configuration—an amorphous structure with a uniform density across the system. Each dataset represents a distinct phase of silica—solid

(300 K), highly viscous liquid (3,000 K) [5], and low-viscosity liquid (6,000 K) (note that the melting temperature of silica is 1,800 K). Here, data ordering is performed only once at the beginning of the simulation, so that the atom data are fully ordered (Fig. 1(d)) initially, and subsequently we measure the data fragmentation ratio as a function of MD steps without further data reordering.

In order to study the influence of temperature on data fragmentation, Fig. 2(a) plots the data fragmentation ratio as a function of MD steps. Simulations at higher temperatures exhibit larger fragmentation ratios. The fragmentation of 300 K dataset fluctuates in the first 500 steps, then rises to 0.15 after 1,000 steps, and remains nearly constant throughout the simulation thereafter. At 3,000 K, the fragmentation ratio quickly increases to 0.24 in the first 200 steps, and then increases linearly with time (0.033 per 1,000 steps), reaching 0.36 after 3,000 MD steps. In contrast, the fragmentation ratio at 6,000 K rapidly rises to 0.73 in the first 500 steps, then continues to increase with a slightly slower rate until the data gets almost fully disordered at 0.93 after 3,000 MD steps.

In order to understand the physical origin of the correlation between the fragmentation ratio and temperature in Fig. 2(a), we measure the mean squared displacement (MSD) of atoms in each dataset as a function of time. MSD is used to identify the phase (*e.g.* solid vs. liquid) of the material and to calculate the diffusion rate of atoms in the system, which may be used to quantify the dynamics of atoms during the simulation. Figure 2(b) shows the MSD of 300 K, 3,000 K, and 6,000 K datasets over 6 picoseconds (3,000 MD steps). The result shows that at 300 K, silica remains in solid phase (MSD remains constant). In contrast, 3,000 K silica is melted and becomes a highly viscous liquid with a small diffusion coefficient of 2.31×10^{-5} cm^2/s (the diffusion coefficient is obtained from the linear slope of the MSD curve). Similarly, MSD of 6,000 K dataset shows that the system is completely melted with a much larger diffusion coefficient of 4.17×10^{-4} cm^2/s. Because atoms are not diffusing in 300 K dataset, 83% of the atoms ($f = 0.17$) remain in the same linked-list cell throughout the simulation.

Only atoms close to the cell boundaries move back and forth between the original and the neighbor cells due to thermal vibration. However, atoms in 3,000 K dataset are melted, and their slow diffusion from their original cells causes the gradually increasing fragmentation ratio. For 6,000 K system, atoms diffuse approximately 18 times faster than 3,000 K system, resulting in the rapid rise of fragmentation ratio. Only 3.2% of the atoms remain in the same cell after 3,000 steps. These results clearly show that diffusion is a major physical factor that contributes to data fragmentation in MD simulations.

Although the silica experiment shows that the temperature significantly affects data fragmentation, we also found that its effect on the fragmentation ratio is sensitive to the material and phenomenon being simulated. To demonstrate this point, the second experiment simulates flash heating of an aluminum nanoparticle surrounded by oxygen environment using a reactive interatomic potential [12], which involves 15,101,533 atoms on 1,024 processors of dual core AMD Opteron 270 2.0 GHz (Italy) at the USC-HPCC facility. Similar to the first simulation, the atom data arrays are ordered only once at the beginning

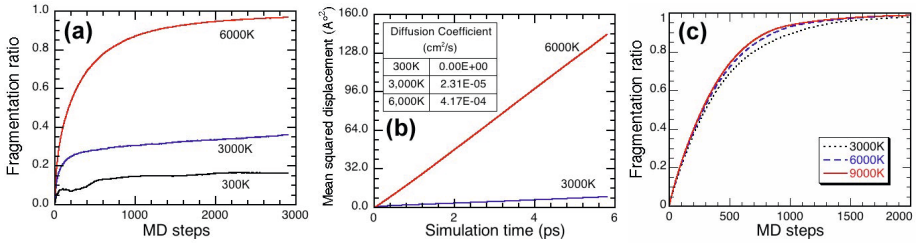

Fig. 2. (a) Time variation of data fragmentation ratio over 3,000 MD steps. Three datasets of different initial temperatures (300 K, 3,000 K, and 6,000 K) are plotted. (b) Mean squared displacement (MSD) of 300 K, 3,000 K, and 6,000 K datasets. The MSD of 300 K dataset remains constant at $0.23\,\text{Å}^{-2}$ (too small to be seen in the figure). The inset shows the diffusion coefficients at the three temperatures. (c) Data fragmentation ratio during aluminum combustion simulation over 2,000 MD steps. The dataset involves 15 million atoms on 1,024 processors.

of the simulation. Then, the fragmentation ratio of 3,000 K, 6,000 K and 9,000 K datasets are measured at each step. Figure 2(c) shows the data fragmentation ratio as a function of MD steps. We see that the fragmentation ratios of all datasets rapidly rise to above 90% after 1,000 MD steps, and continue to increase to over 98% after 2,000 MD steps. Since the aluminum nanoparticles are at considerably high temperatures (far above the melting temperature of aluminum $\sim$ 930 K) and are surrounded by oxygen atoms in the gas phase, the atoms are highly reactive and move very rapidly. This accelerates the fragmentation to proceed very quickly regardless of the temperature. In such applications, data reordering is indispensable and is required to be performed more often in order to maintain good spatial locality. These results indicate that fragmentation is highly system dependent, so that data reordering needs to be performed dynamically at runtime, adapting to the systems behaviors.

3.2 Granularity Induced Fragmentation

Parallel MD using spatial decomposition introduces computational artifacts such as spatial subsystem (or domain) boundaries, which also contribute to the fragmentation of atom data arrays. In this subsection, we study the influence of granularity (*i.e.* the number of atoms per compute node) on data fragmentation. Atoms near a domain-boundary surface tend to migrate among domains regardless of physical simulation conditions. This inter-domain atomic migration triggers rearrangements of data arrays, including the deletion of the migrated atom data from the original array, compression of data array after the removal of the migrated atoms, and their appending to the array in the destination domain. These newly migrated atoms in the destination domain cause data fragmentation, since they do not reside continuously in memory.

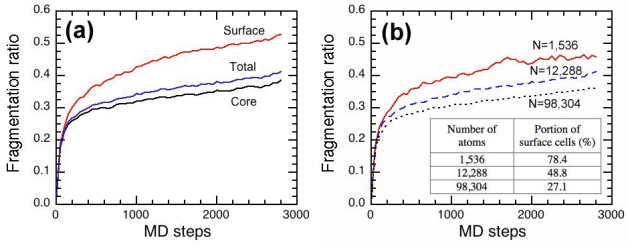

Fig. 3. (a) Time variation of the data fragmentation ratio of 12,288-atom silica at 3,000 K over 3,000 MD steps, for surface cells, core cells, and both combined. (b) Time variation of the data fragmentation ratio of 3,000 K silica varying granularities over 3,000 MD steps. Inset table shows computational parameters of silica datasets.

To confirm the expected high fragmentation ratio at the domain boundaries as explained above, the data fragmentation ratios of the surface cells (*i.e.* the outermost layer cells that share at least one facets with the inter-domain surfaces) and core cells (*i.e.* non-surface cells deep inside each domain) are measured separately. Here, we consider a 12,288-atom silica dataset initially at 3,000 K temperature similar to the dataset used in the first experiment of section 3.1 but with a reduced domain size (their dimensions are reduced by half in all three directions, so that the domain volume is one-eighth of that in section 3.1). This dataset consists of 1,000 cells in total ($10 \times 10 \times 10$), of which 488 cells (48.8%) are surface cells. Figure 3(a) clearly shows that the data fragmentation ratio of the surface cells is larger than that of the core cells. In the first 100 MD steps, the fragmentation ratios of the two groups are almost identical. Then, the fragmentation ratio of the surface cells begins to increase at higher rate reaching 0.53 after 3,000 MD steps, whereas the fragmentation ratio of the core cells is only 0.39. Thus, the atom data of the surface cells is 14% more fragmented that that in the core cells, yielding a total fragmentation ratio of 0.42 for the entire system. This result confirms that computational artifacts such as domain boundary indeed induce additional fragmentation.

The domain-boundary induced fragmentation also implies that systems with smaller granularities will have more fragmentation due to their larger surface-to-volume ratios (*i.e.*, larger portions of linked-list cells are domain-boundary cells). To test this hypothesis, we perform MD simulations for 3,000 K silica system with three different granularities—98,304, 12,288, and 1,536 atoms, over 3,000 MD steps. Figure 3(b) shows that datasets with smaller granularities indeed have larger fragmentation ratios. After 3,000 MD steps, the data array of the smallest granularity (1,536 atoms) dataset is 9.7% more fragmented than that in the largest granularity (98,304 atoms) dataset. Also, the 12,288 atoms dataset is 5.2% more fragmented compare to the largest dataset. The figure also shows large fluctuation for the fragmentation ratio for $N = 1,536$ dataset, due to less statistics inherent for the small system size.

4 Performance Measurements

In this section, we first establish a correlation between the data fragmentation ratio and the performance of the program. In a fragmented dataset, atom data are likely to reside in different memory pages, which causes a large number of DTLB misses, when they are fetched. The reordering algorithm explained in section 3 clusters atom data that need to be fetched and computed in relatively proximate times (*e.g.* atoms in the same cells) in the same page, thereby reducing DTLB misses tremendously. However, as the simulation progresses, atoms that are once ordered possibly move out of their original cells. It is thus expected that the increase of DTLB misses caused by data fragmentation is a major source of performance degradation in MD.

To confirm the correlation between the number of DTLB misses and the data fragmentation ratio, we perform a test using the same datasets from the first experiment of section 3.1. In addition to the fragmentation ratio measured in section 3.1, we here measure the DTLB miss rate as a function of MD steps. We use the Intel VTune Performance Analyzer to monitor DTLB miss events during MD simulation on Intel Core i7 920 2.67 GHz (Nehalem) processor. We measure the number of DTLB misses after data ordering is performed, then normalize it by the original number of DTLB misses without data ordering. We find that the initial DTLB miss rates at all temperatures are approximately 0.07 ($\ll 1$) right after the reordering is performed (namely, the reordering reduces DTLB misses by 93%). The great improvement highlights a significant role of data reordering in reducing DTLB misses for irregular memory-access applications. We also monitor the number of DTLB misses as a function of MD time steps and observe distinct profiles at different temperatures (Fig. 4(a)). The DTLB misses ratio at 300 K and 3,000 K saturates at 0.08 and 0.13, respectively. In contrast, the 6,000 K simulation exhibits a continuous increase of the number of DTLB misses. The DTLB miss rate reaches 0.62 at 3,000 MD steps after the initial data reordering.

Figure 4(b) shows the relation between the data fragmentation ratio and the DTLB misses rate at the three temperatures. The DTLB misses rate remains relatively small when the fragmentation ratio is small, while it rapidly increases when the fragmentation ratio increases above 0.8. This result clearly demonstrates a strong correlation between the DTLB miss rate and the data fragmentation ration. Namely, in MD simulations where atoms are moving extensively, the DTLB miss rate increases rapidly as its fragmentation ratio does.

To show that DTLB misses are the source of performance deterioration during simulation, we measure the running time of the same datasets as in the DTLB miss measurement. Figure 4(c) shows the running time of each MD step for 3,000 MD steps after ordering data at the first step. At 6,000 K, the running time gradually increases over time, and after 3,000 MD steps, the average running time per step increases by 8%. To study how much more performance degradation occurs after the initial data ordering, we extend the execution of 6,000 K dataset to 20,000 MD steps. The result shows an increase of the running time per step by 21%. This result indicates that without data reordering, the performance

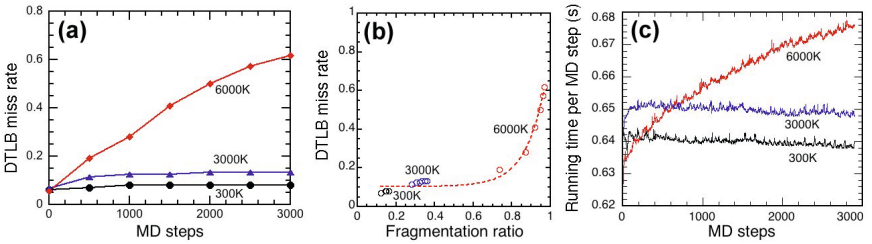

Fig. 4. (a) Time variation of the DTLB miss rate in 300, 3,000 and 6,000 K dataset over 3,000 MD steps. (b) Relation between the DTLB miss rate and the data fragmentation ratio in 300, 3,000 and 6,000 K datasets. (c) Running time per MD step as a function of MD steps at temperatures 300, 3,000, and 6,000 K for 98,304 silica atoms.

continues to degrade, and the running time per step continues to increase. On the other hand, the running time rapidly increases at the first 100 MD steps but then remains almost constant at 300 K and 3,000 K. These performance behaviors are akin to the fragmentation ratio and DTLB misses profiles in Figs. 2(a) and 4(a), respectively. These results thus confirm that data fragmentation in memory is indeed the source of performance deterioration during runtime through the increase of DTLB misses.

In Fig. 4(c), we also observe that 6,000 K simulation initially executes fastest compare to the other simulations (3.1% faster than 300 K and 2.5% faster than 3,000 K), which cannot be explained by the difference in the DTLB miss rate. One possible reason for this discrepancy is that higher temperature systems have sparser atomic distributions (*i.e.* lower atomic number densities), such that there are less number of atom interactions within the interaction range $r_{\mathbf{c}}$, and hence less number of floating-point operations. To test this hypothesis, we measure the atom-distance distribution for all datasets. The results show that the number of atom pairs within distance $r_{\mathbf{c}}$ of 6,000 K system is approximately 3.0% and 1.4% less than those of 300 K and 3,000 K systems, respectively. As a result, 6,000 K simulation has the least computational load resulting in the fastest running time at the beginning of the simulation, which however is rapidly offset by the increase in DTLB misses.

In addition to the atom ordering discussed above, the cell ordering in the memory space also affects the locality. To address this issue, we measure the performance of three different cell ordering methods: 1) sequential ordering; 2) Hilbert-curve ordering; and 3) Morton-curve ordering [6]. The performance measurements are performed with 300 K temperature silica systems with grain size ranging from 12,288 - 331,776 atoms with reordering period of 20 MD steps on Intel Core i7 920 2.67 GHz (Nehalem) processor. The performance comparisons of all ordering methods are shown in Table 1. The results do not exhibit significant improvement due to different ordering methods. For example, the running time of Hilbert curve ordering is at most 1.76% less than that of the sequential ordering at largest granularity (331,776 atoms), while the running time of Morton curve ordering is less than 1% different for all granularities. One possible

Table 1. Runtime result with different ordering methods for silica systems at temperature 300 K. Reordering costs shown are obtained from sequential ordering.

Number of atoms	System dimensions (cells)	Average running time per MD step (ms)			Reordering cost (ms)
		Sequential	Morton	Hilbert	
12,288	10×10×10	78.48	78.57	78.45	0.66±0.02
98,304	20×20×20	639.9	640.1	637.1	8.41±0.07
331,776	30x30x30	2125.7	2117.4	2088.8	28.21±0.10

reason of the insensitivity of the performance on the cell-ordering method is the small number of references across intra-node cell boundaries. For simplicity, we employ the sequential ordering in the following.

5 Reordering Frequency Optimization

To minimize the performance degradation due to data fragmentation, we propose to repeat data reordering periodically during MD simulation. Specifically, we reorder atom arrays after every N_{rp} MD steps (*i.e.*, the reordering period N_{rp} is the number of MD steps between successive reordering operations). Though such data reordering is beneficial and often improves overall application performance, ordering arrays itself introduces an additional computational cost. Therefore, we here develop a dynamic data-reordering schedule based on a performance model and the runtime measurement in section 4. To do so, we first introduce a model that accounts for the reordering overhead. Let t_{cost} be the data reordering cost (*i.e.* the time to reorder arrays), N_{total} is the total simulation steps, and $t(n)$ is the running time at step n after ordering. The total running time, τ, as a function of reordering period, N_{rp}, is then written as

$$\tau(N_{\mathrm{rp}}) = \left\lfloor \frac{N_{\mathrm{total}}}{N_{\mathrm{rp}}} \right\rfloor \left(\sum_{n=1}^{N_{\mathrm{rp}}} t(n) + t_{\mathrm{cost}} \right) + \sum_{n=1}^{\mathrm{mod}(N_{\mathrm{total}}, N_{\mathrm{rp}})} t(n) \qquad (1)$$

The optimal reordering period is then determined as the one that minimizes the total running time $N^*_{\mathrm{rp}} = \mathrm{argmin}(\tau(N_{\mathrm{rp}}))$. To find the ordering cost parameters in Eq. (1), we measure the reordering cost with different grain sizes. The results are summarized in Table 1.

Figure 5(a) shows the total running times for 3,000 MD steps as a function of the reordering period estimated from Eq. (1) with the measured reordering costs in Table 1. We obtain the optimal reordering period (which minimizes the total running time) as 69, 5, and 3 steps for 300 K, 3,000 K and 6,000 K datasets, respectively (see the arrows in Fig. 5(a)). With the optimally scheduled reordering thus determined, the overall performance is estimated to be improved by a factor of 1.35, 1.36 and 1.28 at 300 K, 3,000 K and 6,000 K, respectively.

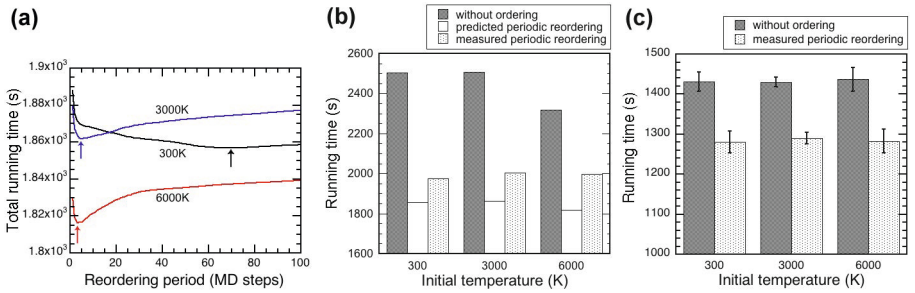

Fig. 5. (a) Total run time after 3,000 MD steps as a function of reordering period. The optimal period at 300 K, 3,000 K and 6,000 K are 69, 5 and 3 steps, respectively. (b) Comparison of total run time of silica MD over 3,000 steps achieved by model prediction and actual measurement of periodic reordering compared with that of the original code without ordering. (c) Total run time over 1,000 steps of parallel runs.

To verify this model prediction, we measure the running time of MD simulations that implement data reordering with the optimal reordering schedule. Here, MD simulations of silica containing 98,304 atoms at temperatures 300 K, 3,000 K, and 6,000 K are executed using periodic reordering at every 69, 5, and 3 steps, respectively. The results show that the measured speedups in actual executions are 1.27, 1.25, and 1.17, respectively, with 6.4%, 7.6%, and 10.0% error from the estimated values. The running time of the system without ordering, the optimized running time estimated from the model, and the measured running time with optimally scheduled reordering are compared in Fig. 5(b). To confirm that this performance benefit carries over to parallel runs, we performed a performance benchmark of silica MD with the same initial temperatures. The benchmark is executed on 192 cores of dual hexcore Intel Xeon X5650 2.66 GHz (Westmere) using 192,000 atoms per core (36 million atoms total). Figure 5(c) shows that speedups of 1.12, 1.11, and 1.12 are obtained from 300 K, 3,000 K, and 6,000 K runs, respectively. These figures shows a substantial effect of runtime data reordering on the performance of MD simulations. It should be noted that the state-of-the-art MD simulations are run up to 10^{12} time steps [2], for which the 10^3-step pre-profiling run for constructing the performance model, Eq. (1), can be amortized by updating the model every 10^6 steps.

6 Conclusions

We have developed an optimal data-reordering schedule for molecular dynamics simulations on parallel computers. Our analysis has identified physical/computational conditions such as a high temperature and a small granularity, which considerably accelerate data fragmentation in the memory space, thereby causing continuous performance degradation throughout the simulations

at runtime. Our profiling results have revealed that the degree of data fragmentation correlates with the number of DTLB misses and have identified the former as a major cause of performance decrease. Based on the data fragmentation analysis and a simple performance model, we have developed an optimal data-reordering schedule, thereby archiving a speedup of 1.36 for 3,000 K silica simulation. This paper has thus proposed a practical solution to a dynamic data-fragmentation problem that plagues many scientific and engineering applications. Future research could be focused on other physical properties such as pressure, local density, and non-uniform mechanical loadings (such as shear deformation). Also, performance degradation in the light of other performance metrics such as cache misses could be explored. This work was partially supported by DOE-BES/SciDAC and NSF-CDI/PetaApps.

References

1. Chen, Y.C., Nomura, K., Kalia, R.K., Nakano, A., Vashishta, P.: Void deformation and breakup in shearing silica glass. Phys. Rev. Lett. 103(3) (2009)
2. Dror, R.O., Jensen, M., Borhani, D.W., Shaw, D.E.: Molecular dynamics and computational methods exploring atomic resolution physiology on a femtosecond to millisecond timescale using molecular dynamics simulations. J. Gen. Physiol. 135, 555–562 (2010)
3. Han, H., Tseng, C.W.: Exploiting locality for irregular scientific codes. IEEE Trans. Par. Dist. Sys. 17(7), 606–618 (2006)
4. Hu, Y.C., Cox, A., Zwaenepoel, W.: Improving fine-grained irregular shared-memory benchmarks by data reordering. In: Supercomputing (2000)
5. Kushima, A., Lin, X., Li, J., Eapen, J., Mauro, J.C., Qian, X.F., Diep, P., Yip, S.: Computing the viscosity of supercooled liquids. J. Chem. Phys. 130(22), 224504 (2009)
6. Mellor-Crummey, J., Whalley, D., Kennedy, K.: Improving memory hierarchy performance for irregular applications using data and computation reorderings. Int'l J. Par. Prog. 29(3), 217–247 (2001)
7. Nomura, K., Dursun, H., Seymour, R., Wang, W., Kalia, R.K., Nakano, A., Vashishta, P., Shimojo, F., Yang, L.H.: A metascalable computing framework for large spatiotemporal-scale atomistic simulations. In: IPDPS (2009)
8. Peng, L., Kunaseth, M., Dursun, H., Nomura, K., Wang, W., Kalia, R.K., Nakano, A., Vashishta, P.: A scalable hierarchical parallelization framework for molecular dynamics simulation on multicore clusters. In: PDPTA (2009)
9. Phillips, J.C., Zheng, G., Kumar, S., Kale', L.V.: NAMD: Biomolecular simulations on thousands of processors. In: Supercomputing (2002)
10. Shaw, D.E.: A fast, scalable method for the parallel evaluation of distance-limited pairwise particle interactions. J. Comp. Chem. 26(13), 1318–1328 (2005)
11. Singh, J.P., Hennessy, J.L., Gupta, A.: Implications of hierarchical N-body methods for multiprocessor architectures. ACM Trans. Comput. Sys. 13(2), 141–202 (1995)
12. Wang, W.Q., Clark, R., Nakano, A., Kalia, R.K., Vashishta, P.: Fast reaction mechanism of a core-shell nanoparticle in oxygen. Appl. Phys. Lett. 95(26) (2009)
13. Yao, Z.H., Wang, H.S., Liu, G.R., Cheng, M.: Improved neighbor list algorithm in molecular simulations using cell decomposition and data sorting method. Comput. Phys. Commun. 161(1-2), 27–35 (2004)