Journal of Molecular Graphics and Modelling 38 (2012) 243–247

Contents lists available at SciVerse ScienceDirect

Journal of Molecular Graphics and Modelling

journal homepage: www.elsevier.com/locate/JMGM

Implementation of the force decomposition machine for molecular dynamics simulations

Urban Borštnik[a], Benjamin T. Miller[b], Bernard R. Brooks[b], Dušanka Janežič[a,c,*]

[a] National Institute of Chemistry, Hajdrihova 19, SI-1000 Ljubljana, Slovenia
[b] National Heart, Lung, and Blood Institute, National Institutes of Health, 5635 Fishers Ln, Rockville, MD 20892-9314, United States
[c] University of Primorska, FAMNIT, Glagoljaška 8, SI-6000 Koper, Slovenia

ARTICLE INFO

Article history:
Accepted 26 June 2012
Available online 7 July 2012

Keywords:
Personal computer clusters
Beowulf clusters
Network topologies
Interconnection architectures

ABSTRACT

We present the design and implementation of the force decomposition machine (FDM), a cluster of personal computers (PCs) that is tailored to running molecular dynamics (MD) simulations using the distributed diagonal force decomposition (DDFD) parallelization method. The cluster interconnect architecture is optimized for the communication pattern of the DDFD method. Our implementation of the FDM relies on standard commodity components even for networking. Although the cluster is meant for DDFD MD simulations, it remains general enough for other parallel computations. An analysis of several MD simulation runs on both the FDM and a standard PC cluster demonstrates that the FDM's interconnect architecture provides a greater performance compared to a more general cluster interconnect.

1. Introduction

Increasingly longer molecular dynamics (MD) simulations [1,2] are used to study even very large molecular systems [3,4]. Due to their computationally demanding nature, reducing the MD simulation time is an important goal. Simulation time can be reduced by advances in MD methodology that reduce the computational effort [5,6] or by increasing the computational speed by employing parallel computation [7]. In parallel MD simulations, the computation is distributed among many processors, which may decrease the simulation time [7].

For effective parallelization, the parallel computer must offer very fast communication among processors. Contemporary parallel computers are interconnected nodes containing multiple multicore processors. Since programs must usually exchange data, the communication time may constitute significant overhead, which decreases the parallel efficiency [8]. The focus on parallelization in this article is on communication among nodes and not individual processors because shared-memory approaches offer efficient intra-node parallelization. One goal of parallelization is therefore to decrease the communication time; this can be achieved in software or hardware. Parallel methods vary in the amount of data that they transfer. In general, for algorithms that are not latency-bound, decreasing the data volume decreases the communication time. For a given problem, using a different parallel method may reduce the data transfer requirements, yielding greater parallel efficiency. Another, often complementary, approach to increasing parallel efficiency is to decrease the communication time by using faster communication hardware [9–11] or even specialized hardware for the problem domain, such as the MD-GRAPE chip and computer [12–14] or the Anton machine [11,15,16].

Increasing the coupling between the parallelization method and the underlying parallel computer is an effective way to increase parallel efficiency [17]. Modern clusters of PCs are flexible enough to allow adapting the parallel computer to the parallel method being utilized [18–20].

In this article we introduce the force decomposition machine (FDM) for efficient parallel simulations. The FDM is designed for the communication patterns of the distributed diagonal force decomposition (DDFD) method [21], a new parallelization method for MD simulation. The goals of the FDM and the implemented architecture are presented in Section 2.2. This section also details the expandability options of the FDM as well as its suitability for general-purpose calculations. The FDM is evaluated, and the performance characteristics of the FDM and the DDFD method are analyzed, in Section 4.

2. Distributed diagonal force decomposition method

The DDFD method [21], which is implemented in the CHARMM program [2], exhibits a characteristic data exchange pattern where nodes communicate with a limited set of other nodes. A set of nodes exchanges set-local data independently from the other sets.

* Corresponding author.
E-mail addresses: urban@cmm.ki.si (U. Borštnik), btmiller@nhlbi.nih.gov (B.T. Miller), brb@nih.gov (B.R. Brooks), dusa@cmm.ki.si (D. Janežič).

1093-3263/$ – see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jmgm.2012.06.015

The DDFD method is based on the force decomposition method [22,7]. It is related to the Under Triangle Force Block Decomposition method [23], with the difference that we do not assign nodes to the diagonal block products, but instead distribute the force calculations in these block products to other nodes. The benefit that this crucial difference brings is that all nodes have the same type of work and it enables load balancing, so we have therefore decided to name the new method the distributed diagonal force decomposition method. In the DDFD method, the N atoms of the simulated system are divided into B blocks (disjoint sets). The atoms are randomly assigned to blocks to provide an acceptable initial load balance, though dynamic load balancing also occurs during the simulation [21]. There are B^2 block products formed by pairs of blocks. Each block product includes the pairwise interactions among atoms of the two blocks as illustrated in Fig. 1a. Assuming that the blocks all have the same size N/B, every block product contains N^2/B^2 interactions. Since an interaction between a pair of atoms gives rise to two equal but opposite forces, only half of the total interactions must be calculated: the forces in the block products above the diagonal are equal but opposite to the forces in those below the diagonal. The block products on the diagonal also contains the mirrored forces. The three types of block products are outlined in Fig. 1a. We can therefore concentrate on the interactions in block products below the diagonal (white areas in Fig. 1a) and the lower half of the block products on the diagonal (red areas in Fig. 1a).

A total of $B(B-1)/2$ nodes are assigned to the block products below the diagonal. No node is assigned to the diagonal block products; instead, the interactions from the diagonal block products are distributed to the other nodes in the row or column, as depicted in Fig. 1b. A different node is responsible for calculating the interactions in every unique block product and for this it must have the atomic coordinates for atoms in both blocks. Because it has the data for both of these two blocks, the node can calculate any of the interactions in the diagonal block products. To distribute the computational load among all nodes, an equal part of these interactions are distributed to the $B-1$ nodes that share this data block, yielding the final distribution as seen in Fig. 1c.

In the DDFD method, nodes communicate only with other nodes sharing the same blocks. A node with blocks B_i and B_j communicates with $2(B-2)$ other nodes: the $B-2$ nodes also holding data for B_i in a block product $B_i \times B_k$ as well as the $B-2$ nodes with B_j in a block product $B_l \times B_j$. In Fig. 1c, node 2 would communicate with nodes 1 and 4, belonging to block 1, and with nodes 3 and 6, belonging to block 3. Communication is therefore intra-block.

Each time step of an MD simulation includes the calculation and summation of forces acting upon the atoms and the broadcast of new coordinates after integration. Since the atomic blocks are disjoint sets, a standard global collective operation is replaced by intra-block (block-wise) collective operations. The intrablock summation and broadcast intra-block operations are independent among the blocks. Since each node belongs to two blocks, it participates in two intra-block collective operations. The scope of a node in performing intra-block collective operations is illustrated in Fig. 2.

2.1. Data transfer characteristics

Since the intra-block collective sum and broadcast operations in the different blocks are independent, they can be performed concurrently. The communication time for the entire global sum or broadcast, i.e., for all of the intra-block collective operations, can be as low as the time required to perform one such operation, if a parallel computer's interconnection architecture and communication library allow concurrent transfers between two different nodes. On more restricted interconnections, e.g., switched Ethernet, the time

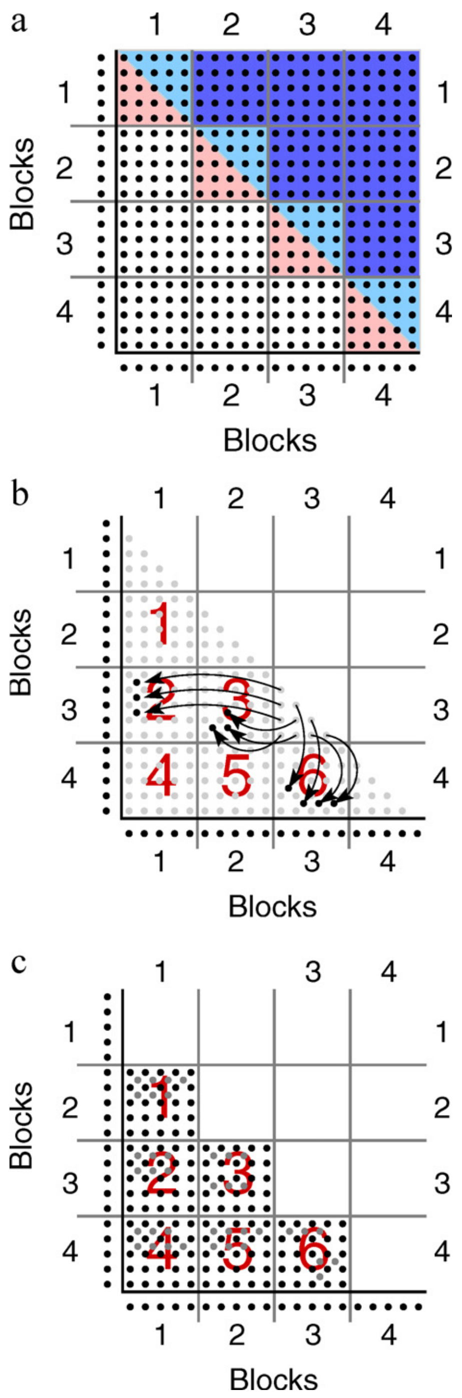

Fig. 1. Distributed diagonal force decomposition overview. (a) 20 atoms (the block dots to the left and below the black lines) are divided into 4 blocks separated by gray. The interactions among atoms are accordingly divided into 16 block products, which are classified into three regions, highlighted here with different backgrounds: fully above the diagonal (dark blue), on the diagonal (light blue and light red), and fully below the diagonal (white). Since the forces above and below the diagonal are mirrored, those above the diagonal are not calculated. (b) Nodes are assigned to the blocks fully below the diagonal (in this case nodes 2, 4, and 5). The interactions from the diagonal block product (i.e., the interactions among atoms within the same block, highlighted with light red in subfigure (a) are distributed to the nodes sharing the same block. The distribution of only one diagonal block product, that is of block 3 with itself, is shown. (c) The final distribution is shown after the distribution of the diagonal blocks is performed for all block products. A node is assigned to each off-diagonal block product to calculate the interactions present in that block product.

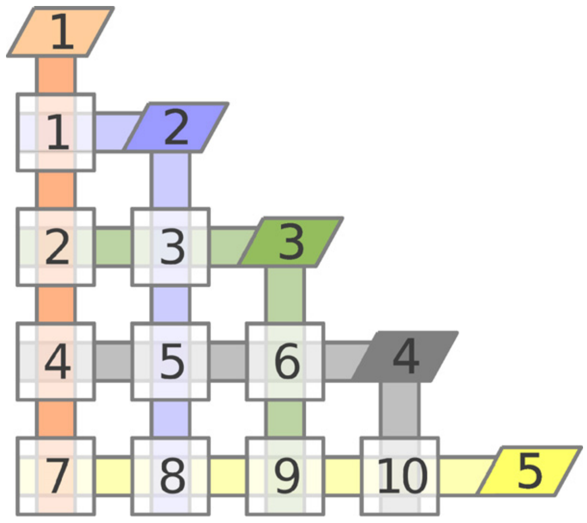

Fig. 2. A 5-block FDM. It has 10 nodes (represented with squares) connected to 5 switches (represented with parallelograms). Every one of the 5 blocks is assigned one of the 5 switches, to which 4 nodes are connected. For example, nodes 1, 2, 4, and 7, which belong to the first block, are connected to switch 1 (colored orange). Every node is connected to two switches; e.g., node 2, which calculates interactions among the atoms in blocks 1 and 3, is connected to the two appropriate switches, switches 1 and 3. A node communicates only with the other nodes in its two blocks: node 2, for example, communicates only with the 6 other nodes sharing its two blocks. It communicates with nodes 1, 4, and 7, sharing block 1, using switch 1 (orange colors). It communicates with nodes 3, 6, and 9, sharing block 3, using switch 3 (green colors). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

required for the global operation is at least twice the time required for a single intra-block collective operation.

We have implemented the intra-block collective sum and broadcast operations by manually programming message sending. The collective operations of standard MPI 2 message-passing libraries are blocking, even for a subset of nodes, which limits their ability to be used concurrently [24–26]. We therefore chose to implement the collective operations by interleaving the intra-block transfers on each node. The message passing transfers for an intra-block collective operation are based on the recursive doubling technique [27,28]. To profit from interconnections capable of concurrent multi-point communication, we implemented the interleaving using non-blocking message passing, but we retain strict ordering to maintain data consistency.

The non-collective data transfers that occur in every MD simulation time step are also implemented as intra-block transfers. To properly treat interactions among three or more atoms, a node may need atomic coordinates for atoms that are not members of any of its two blocks; in addition, the calculated force must be summed to the atoms in the correct block. In our implementation of the DDFD method, these data transfers are also aware of the DDFD topology so that two nodes communicate only if they share the same block. Similarly, we have implemented other truly global collective communication routines so that they transfer the data in a block-wise manner in two major steps.

2.2. *Network architecture of the force decomposition machine*

The FDM was conceived as a special PC cluster topology that would complement the distributed diagonal force decomposition (DDFD) method for the calculation of parallel MD simulations.

The following guidelines were applied for the design of the FDM:

- the machine should rely on commodity components [9],
- there should be a noticeable performance gain with modest hardware additions to a standard cluster,
- the cluster should remain flexible, and
- the machine should be general enough to remain useful for parallel simulations not using the DDFD method.

By limiting the design to commodity clusters, we ensure that the FDM can be built from readily available consumer components. In addition, using standard PC components and widely used networking equipment allows greater flexibility when choosing components and allows for easier expandability. Since many types of calculations are performed on parallel computers, requiring the FDM to allow general parallel computation is also important.

3. Topology of the force decomposition machine

In general, for any switching interconnect technology, several switches with fewer ports are cheaper than a bigger switch with an equivalent number of ports. The network architecture of the FDM therefore uses a number of smaller switches to connect subsets of nodes. Any technology could be used, such as Ethernet (1 or 10 Gbps) or Infiniband. One switch is used for each of the B blocks and the $B-1$ nodes of the block are connected to the switch. Every node is connected to two switches and therefore needs two network interfaces. For example, these interfaces may be combined on a single Ethernet card. It must be stressed that in any case the two interfaces should be capable of independent full-bandwidth operation, including from the bus to the card. As is described in the data transfer characteristics of intra-block collective operations, our implementation of the DDFD method uses non-blocking communication. Two independent network interfaces thus allow the DDFD method to take full advantage of the interconnection network.

The topology of the FDM complements the data transfer characteristics of the distributed diagonal force decomposition method, in which communication is block-based. All of the data transfers needed for an intra-block collective operation pass solely through the block's assigned switch. The topology of a 5-block, 10-node FDM is illustrated in Fig. 2. This figure also highlights the switches and nodes that take part in intra-block collective communication.

In this topology, there are only a specific number of nodes that can be used, depending on the target number of atomic blocks. For B blocks, the number of nodes P in the FDM is $B(B-1)/2$. The smallest practical number of nodes is 3 (when 3 blocks are used). Other possible numbers of nodes are 6, 10, 15,. . .. Particularly interesting are FDMs with 10 and 36 nodes (5 and 9 blocks, respectively) because in these the number of nodes in a block is a power of two. Having a square number of nodes in a block simplifies the intra-block collective communication and allows for useful implementations using multi-CPU computers.

The basic FDM uses only the B switches with $B-1$ connections; however, it is useful for a FDM to also use an additional larger common switch (or several bridged switches), to which all of the P nodes are connected. The purpose of this switch is to provide default connectivity from external computers and to ease the administration of the computers. Its presence or performance does not affect the performance of MD simulations using the DDFD method.

For a practical Ethernet setup each node is connected to general switches for administration with a unique IP address such as 10.0.0.x (with a /16 subnet) where x is the node number. In addition each node will then be connected to two smaller (and preferably faster) switches. In keeping with good networking practice each interface on a node has a different IP address. Each node will therefore have three distinct IP addresses. Because switches correspond to blocks we can number the networks (hence IP subnets) correspondingly, starting with 1: 10.1.0.0/16, 10.2.0.0.0/16,. . .. The interfaces on the nodes are then assigned an address corresponding the switch (network) they are assigned to. For example, with reference to Fig. 2, node 2 is connected to switches 1 and 3. It would therefore have the IP address 10.1.0.2 on the interface connected to switch 1 and IP address 10.3.0.2 on the interface connected to switch 3. Routing must then be set up: all nodes sharing the same block (switch) should have routes to their block-neighbors listed to go through the common switch with the gateway being the interface of the neighbors on the common switch. For example, node 2 would define routes to nodes 1, 4, and 7 through the interface connected to switch 1, 10.1.0.2. The gateways for nodes 1, 4, and 7 should be listed as 10.1.0.1, 10.1.0.4, and 10.1.0.7, respectively. Similarly the routes to nodes 3, 6, and 9 should be routed through the interface connected to switch 3, 10.3.0.2, and gateways 10.3.0.3, 10.3.0.6, and 10.3.0.9, respectively.

4. Results and discussion

For an evaluation of the force decomposition machine, the performance of MD simulations using the DDFD and standard RD (replicated data) methods was compared on a 55-node FDM to a standard PC cluster consisting of 64 nodes. The nodes in both cases were the same. Each had two Intel single-core Xeon 3.06 GHz processors with 2 GB of RAM and connected via gigabit Ethernet. Only one processor per node was used for the application in these benchmarks. For the 11-block FDM each node had an Intel 55 82546 GB dual-port PCI-X network card was connected to two of the 11 different switches.

Short molecular dynamics simulations of two molecular systems were used to compare the different methods and clusters. The smaller system (system A) consists of a solvated carboxymyoglobin protein with 14,026 atoms in total and the larger system (system B) consists of a solvated HIV-1 protease enzyme [29] with 54,212 atoms in total. For both systems 1000 steps of molecular dynamics simulation were run, preceeded by a 5-step setup under CHARMM's default electrostatic and van der Waals conditions. The purpose of including the small 5-step setup was to test all possible communication patterns besides the intra-block communication, including an inter-block broadcast to all nodes. Simulations of both systems used switching nonbonded interactions starting at 10 Å with interaction and list cutoffs at 12 Å and 14 Å, respectively. Neither system used periodic boundary conditions and only pairwise treatment of electrostatic interactions were used.

Molecular dynamics simulations of the two molecular systems using the RD and DDFD methods were used to compare the performance of the standard cluster and the force decomposition machine. For this comparison, a ring topology was used for all RD runs with CHARMM version c32a2. The DDFD runs used block-based DDFD transfers added to the c32a2 CHARMM version, and ring communication for collective communication involving non-atomic data. Speedups for both systems on both clusters are given in Fig. 3a and communication time for these simulations is shown in Fig. 3b.

The results clearly indicate that the FDM, running the DDFD code, performed better than the standard cluster with the replicated data code for any number of nodes. The higher speedup seems to be largely due to the improvement in communication time. In most cases communication time provided by the FDM is faster than the standard cluster, which uses a single networking switch. There is a slight inversion in the speedup for the DDFD when going from 10 to 15 nodes for system A and when going from 28 to 36 nodes for system B. These drops correspond to jumps in the communication time at these transitions. Both of the inversions occur at a transition when the number of processors in a block blocks crosses

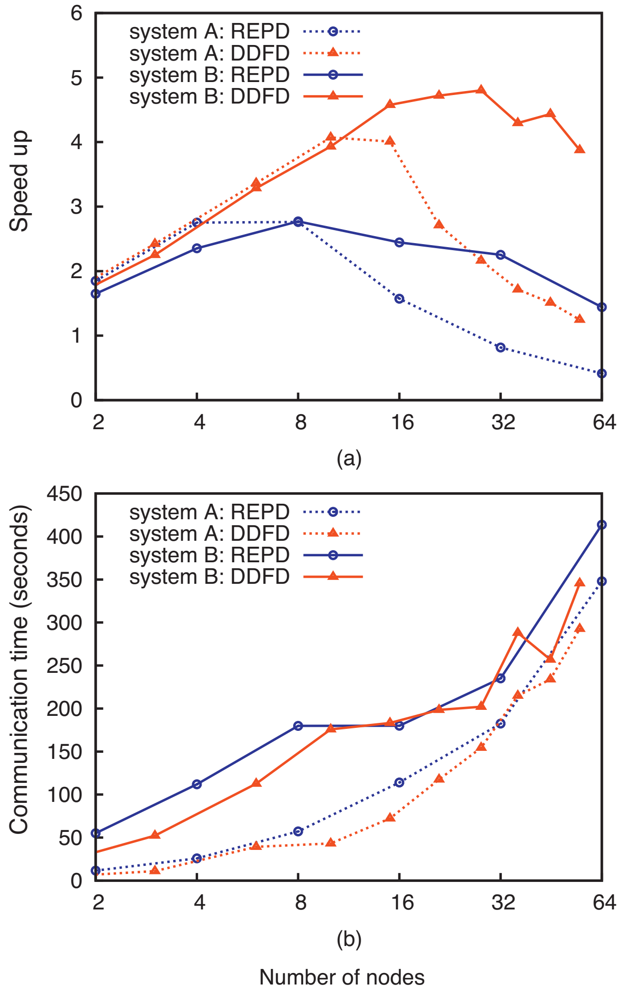

Fig. 3. Speedups (a) and communication times (b) of the replicated data method on the standard cluster (replicated data-REPD, blue lines with circles) and the distributed diagonal force decomposition method on the force decomposition machine (DDFD, red lines with triangles) codes for systems A (dashed lines) and B (solid lines). Communication time was the average per process time as reported by CHARMM. The speedup for *N* nodes was determined by dividing the wall time for one node by the wall time for *N* nodes. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

a power of two. This transition is relevant because the intra-block communication uses the recursive doubling technique as explained in Section 2.1. The number of processors in a block goes from 4 to 5 in the case of system A and from 7 to 8 in the case of system B.

Based on the data, the FDM fully meets the design goals specified earlier. Being composed of standard PCs and widely available networking equipment, the cluster remains expandable and flexible. The additional networking equipment needed to build the FDM topology provides a notable performance gain. The FDM is still usable for other parallel computation: if a large and fast common switch is used, it is still a general PC cluster. In addition, every block can act as an independent cluster since each block has its own networking switch. Smaller parallel computations can be run on the set of nodes that belong to a single block, separating their communication via their block's switch from that of other computations on the cluster.

Despite these positive results, there are limits to scaling which we wished to overcome. Upon further investigation, it was discovered that CHARMM's ring communication topology, which was used to generate the timings of the DDFD method on the FDM, was less efficient than its internal hypercube topology. This difference explains the very poor scaling seen in the FDM in Fig. 3 and which is why we chose to also use the ring topology in the comparison to the RD on the standard cluster.

5. Conclusions

We have described the force decomposition machine (FDM) and showed its implementation as a cluster of personal computers using Ethernet switching technology. The FDM's architecture is optimized toward the newly introduced DDFD method for parallel molecular dynamics simulations. Designing a computer topology to reflect the communication patterns of a parallel program is an effective way to decrease communication time when running the program and thereby increasing the performance.

The FDM uses standard, widely available components. It is easily expandable to a certain limit and offers several options to plan for quickly recovering from failed PCs in the cluster. Despite its commodity nature, the FDM's novel architecture provides increased parallel performance compared to the prevalent switched network topology.

We have presented the performance advantages of using the FDM in conjunction with the DDFD method, which is introduced in a preceding article [21], to a standard cluster of personal computers for molecular dynamics simulations. The FDM and DDFD are particularly advantageous for larger molecular systems and a greater number of nodes.

Acknowledgements

The authors would like to acknowledge the financial support of the Slovenian Research Agency under Grant No. P1-0002. This research was supported in part by the Intramural Research Program of the NIH, NHLBI. The authors would also like to thank Dr. Milan Hodošček for his valuable insights and Rishi P. Singh for assistance assembling the FDM.

References

[1] B.R. Brooks, R.E. Bruccoleri, B.D. Olafson, D.J. States, S. Swaminathan, M. Karplus, CHARMM: a program for macromolecular energy, minimization, and dynamics calculations, Journal of Computational Chemistry 4 (1983) 187–217.
[2] B.R. Brooks, C.L. Brooks, A.D. Mackerell, L. Nilsson, R. Petrella, B. Roux, Y. Won, G. Archontis, C. Bartels, S. Boresch, A. Caflisch, L. Caves, Q. Cui, A. Dinner, M. Feig, S. Fischer, J. Gao, M. Hodoščcek, W. Im, K. Kuczera, T. Lazaridis, J. Ma, V. Ovchinnikov, E. Paci, R.W. Pastor, C.B. Post, J. Pu, M. Schaefer, B. Tidor, R.M. Venable, H.L. Woodcock, X. Wu, W. Yang, D.M. York, M. Karplus, CHARMM: the biomolecular simulation program, Journal of Computational Chemistry 30 (2009) 1545–1614.
[3] P.L. Freddolino, A.S. Arkhipov, S.B. Larson, A. McPherson, K. Schulten, Molecular dynamics simulations of the complete satellite tobacco mosaic virus, Structure 14 (2006) 437–449.
[4] M.L. Klein, W. Shinoda, Large-scale molecular dynamics simulations of self-assembling systems, Science 321 (2008) 798–800.
[5] D. Frenkel, B. Smit, Understanding Molecular Simulation: From Algorithms to Applications, Academic Press, San Diego, 2002.
[6] D. Janežič, M. Praprotnik, F. Merzel, Molecular dynamics integration and molecular vibrational theory. I. New symplectic integrators, Journal of Chemical Physics 122 (2005), art. no. 174101.
[7] S.J. Plimpton, B.A. Hendrickson, A new parallel method for molecular-dynamics simulation of macromolecular systems, Journal of Computational Chemistry 17 (1996) 326–337.
[8] D.W. Heermann, A.N. Burkitt, Parallel Algorithms in Computational Science, Springer-Verlag, Berlin, 1991.
[9] T. Sterling, D.J. Becker, D. Savarese, Beowulf: a parallel workstation for scientific computation, in: Proceedings of the 24th International Conference on Parallel Processing, vol. 1, 1995, pp. 11–14.
[10] N.R. Adiga, M.A. Blumrich, D. Chen, P. Coteus, A. Gara, M.E. Giampapa, P. Heidelberger, S. Singh, B.D. Steinmacher-Burow, T. Takken, M. Tsao, P. Vranas, Blue Gene/L torus interconnection network, IBM Journal of Research and Development 49 (2005) 265–276.
[11] D.E. Shaw, M.M. Deneroff, R.O. Dror, J.S. Kuskin, R.H. Larson, J.K. Salmon, C. Young, B. Batson, K.J. Bowers, J.C. Chao, M.P. Eastwood, J. Gagliardo, J.P. Grossman, C.R. Ho, D.J. Ierardi, I. Kolossváry, J.L. Klepeis, T. Layman, C. McLeavey, M.A. Moraes, R. Mueller, E.C. Priest, Y. Shan, J. Spengler, M. Theobald, B. Towles, S.C. Wang, Anton, a special-purpose machine for molecular dynamics simulation, SIGARCH Computer Architecture News 35 (2007) 1–12.
[12] T. Narumi, R. Susukita, T. Ebisuzaki, G. McNiven, B. Elmegreen, Molecular dynamics machine: special-purpose computer for molecular dynamics simulations, Molecular Simulation 21 (1999) 401–415.
[13] T. Narumi, Special-purpose Computer for Molecular Dynamics Simulations, Doctor's thesis, University of Tokyo, 1998.
[14] T. Narumi, A. Kawai, T. Koishi, An 8.61 Tflop/s molecular dynamics simulation for NaCl with a special-purpose computer: MDM, in: Proceedings of the SuperComputing, ACM, Denver, 2001.
[15] J.S. Kuskin, C. Young, J.P. Grossman, B. Batson, M.M. Deneroff, R.O. Dror, D.E. Shaw, Incorporating flexibility in Anton, a specialized machine for molecular dynamics simulation, in: IEEE 14th International Symposium on High Performance Computer Architecture, 2008. HPCA, IEEE, 2008, pp. 343–354.
[16] J.P. Grossman, C. Young, J.A. Bank, K. Mackenzie, D.J. Ierardi, J.K. Salmon, R.O. Dror, D.E. Shaw, Simulation and embedded software development for Anton, a parallel machine with heterogeneous multicore ASICs, in: Proceedings of the 6th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, ACM, 2008.
[17] U. Borštnik, M. Hodošček, D. Janežič, Improving the performance of molecular dynamics simulations on parallel clusters, Journal of Chemical Information and Computer Science 44 (2004) 359–364.
[18] C. Reschke, T. Sterling, D. Ridge, D. Savarese, D.J. Becker, P. Merkey, A design study of alternative network topologies for the Beowulf parallel workstation, in: Proceedings of the Fifth IEEE International Symposium on High Performance Distributed Computing, IEEE, 1996, pp. 626–636.
[19] U. Borštnik, D. Janežič, Symplectic molecular dynamics simulations on specially designed parallel computers, Journal of Chemical Information and Modeling 45 (2005) 1600–1604.
[20] R. Trobec, U. Borštnik, D. Janežič, Communication performance of d-meshes in molecular dynamics simulation, Journal of Mathematical Chemistry 45 (2009) 503–512.
[21] U. Borštnik, B.T. Miller, B.R. Brooks, D. Janežič, The distributed diagonal force decomposition method for parallelizing molecular dynamics simulations, Journal of Computational Chemistry 32 (2011) 3005–3013.
[22] S.J. Plimpton, Fast parallel algorithms for short-range molecular dynamics, Journal of Chemical Physics 117 (1995) 1–19.
[23] J.W. Shu, B. Wang, M. Chen, J.Z. Wang, W.M. Zheng, Optimization techniques for parallel force-decomposition algorithm in molecular dynamic simulations, Computer Physics Communications 154 (2003) 121–130.
[24] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, J. Dongarra, MPI: The Complete Reference, MIT Press, Cambridge, MA, 1995.
[25] W. Gropp, S. Huss-Lederman, A. Lumsdaine, E. Lusk, B. Nitzberg, W. Saphir, M. Snir, MPI – The Complete Reference, MIT Press, Cambridge, MA, 1998.
[26] T. Hoefler, A. Lumsdaine, W. Rehm, Implementation and performance analysis of non-blocking collective operations for MPI, in: Proceedings of the 2007 International Conference on High Performance Computing, Networking, Storage and Analysis, SC07, IEEE Computer Society/ACM, 2007.
[27] R.A. van de Geijn, LAPACK Working Note 29 On Global Combine Operations, Technical Report CS-91-129, University of Tennessee, 1991.
[28] B.R. Brooks, M. Hodoščcek, Parallelization of CHARMm for MIMD machines, Chemical Design Automation News 7 (1992) 16–22.
[29] K.P. Eurenius, D.C. Chatfield, B.R. Brooks, M. Hodoščcek, Enzyme mechanisms with hybrid quantum and molecular mechanical potentials. I. Theoretical considerations, International Journal of Quantum Chemistry 60 (1996) 1189–1200.