International Journal of Modern Physics C
Vol. 15, No. 1 (2004) 193–201

World Scientific
www.worldscientific.com

LARGE-SCALE MOLECULAR-DYNAMICS SIMULATION OF 19 BILLION PARTICLES

KAI KADAU
Theoretical Division, Los Alamos National Laboratory
MS B262, Los Alamos, New Mexico 87545, USA
kkadau@lanl.gov

TIMOTHY C. GERMANN
Applied Physics Division, Los Alamos National Laboratory
MS F699, Los Alamos, New Mexico 87545, USA
tcg@lanl.gov

PETER S. LOMDAHL
Theoretical Division, Los Alamos National Laboratory
MS B262, Los Alamos, New Mexico 87545, USA
pxl@lanl.gov

Received 8 August 2003
Revised 10 August 2003

We have performed parallel large-scale molecular-dynamics simulations on the QSC-machine at Los Alamos. The good scalability of the SPaSM code is demonstrated together with its capability of efficient data analysis for enormous system sizes up to 19 000 416 964 particles. Furthermore, we introduce a newly-developed graphics package that renders in a very efficient parallel way a huge number of spheres necessary for the visualization of atomistic simulations. These abilities pave the way for future atomistic large-scale simulations of physical problems with system sizes on the μ-scale.

Keywords: Large-scale; molecular-dynamics simulations; data analysis; sphere rendering.

1. Introduction

In the early 90's Lomdahl, Beazley, and coworkers did pioneering work on large-scale molecular-dynamics (MD) simulations[1] with up to 131 million particles on a variety of parallel platforms, including the Connection Machine 5.[2–5] Ever since, large-scale MD simulations have been used intensively to get insight into atomic processes on the picosecond time scale for various solid–state material science issues.[6–13] It has been demonstrated that on today's largest parallel supercomputers MD simulations with one billion atoms are feasible,[12,13] and in 2000 Roth *et al.* ran some five billion atoms for five integration time steps on a Cray T3E using 512 processors.[14]

However, such ultra-large-scale simulations present an analysis problem due to the enormous datasets which are produced.[4,14] For example, the memory needed to save one configuration of one billion atoms in single precision is more than 15 GByte, which makes it problematic and inconvenient to save all the raw data and analyze the data in a postprocessing fashion, even on modern platforms with TBytes of hard-disc capacity.[a] Rather, an on-the-fly analysis during the simulation run is desired.

The MD code SPaSM (Scalable Parallel Short-range Molecular dynamics) has multiple tools to analyze the simulation results during the course of the simulation.[4,5] The methods implemented include generation of pictures of configurations (particle as well as cell-based), local and global quantities like stress tensor elements, temperature, energy, displacement, centrosymmetry parameter,[15] etc. It also allows for calculating histograms and correlation functions for the whole system as well as for subsets of the system.

Here, we demonstrate the capability of the SPaSM library to run and analyze systems containing up to ≈ 20 billion atoms on Los Alamos' QSC-machine.[16] Furthermore, we introduce a new parallel sphere rendering capability that we developed in order to generate high resolution pictures of configurations containing millions to billions of particles.

2. The SPaSM Code

The SPaSM code[2–5] consists of a library that manages the essential steps common for every large-scale MD simulation performed on parallel machines such as memory management, communication, graphics, etc. Outside this core is a user-code which allows for different specifics like potentials, analysis tools, boundary conditions, etc. Both parts are written in ANSI C, with the Message Passing Interface (MPI) used by the library for the communication between processing nodes (PN).[17] The actual simulation is run using the scripting language Python[18] to call the C-subroutines by utilizing the Simplified Wrapper and Interface Generator (SWIG).[19] The approach taken assumes a finite range of interaction with a cut-off distance r_{cut}, to achieve linear scaling in the number of particles.

For simplicity and brevity we limit the discussion here to the 2D case with a square geometry. However, the SPaSM library is able to run rectangular 2D and 3D geometries. A spatial decomposition of the simulation space assigns particles in a square to each PN. In Fig. 1 such a decomposition is shown for a 9 PN case. The space assigned to each PN is further subdivided into squares with an edge-length r_{cell} slightly larger than the interaction cut-off r_{cut} of the interaction potential. Since the interaction range is slightly smaller than r_{cell}, the interaction has to be

[a]It is assumed that the configuration consists of the (X, Y, Z)-particle position and a scalar like the potential energy. Each of these four numbers usually takes four bytes in single precision. However, for a restart of the system the position and velocity vectors are usually needed in double precision, which makes the memory consumption about 45 GByte for one billion particles.

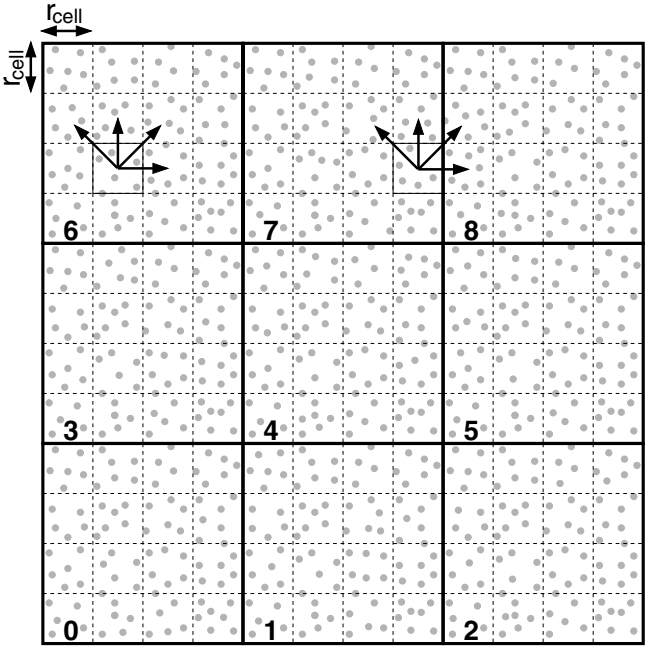

Fig. 1. Schematic of a nine PN (0–8) run with square geometry: A spatial decomposition of the simulation space assigns particles in a rectangle to each PN. The space assigned to each PN is further subdivided into (square) cells with an edge-length r_{cell} slightly larger than the interaction cut-off r_{cut} of the interaction potential. The interaction is calculated only for particles in the same cell and for the eight neighboring cells following an interaction path for each cell that only visits four neighbor cells (see PN 6). If a neighbor cell is located on a different PN (see PN 7), synchronous message passing is applied.

calculated only between atoms in the same cell and the eight neighboring cells. This procedure is done by an interaction pathway, where each cell only visits four neighbor cells as depicted in Fig. 1. Since every cell is performing on that pathway the "missing" interactions of the four nonvisited neighbor cells are picked up by the pathways of these neighbor cells. If a neighboring cell is assigned to a different PN, synchronous message passing is applied. An "interaction path" is used to minimize the number of messages required in 2D and 3D.[2,3]

After all the forces $\mathbf{f}_i$ have been accumulated on each particle i with mass m_i, the new positions $\mathbf{r}_i$, according to Newton's equations of motion $\mathbf{f}_i = m_i \ddot{\mathbf{r}}_i$, are calculated. As particles may move to different cells or processors, a redistribution of particles between processor boundaries is done by synchronous message passing. After the redistribution of particles the system is ready for the next iteration.

3. Large-Scale Simulations

We performed MD on the QSC-machine, which is the "little brother" of Los Alamos' Q-machine with the same architecture but less computing nodes (CN).[16] The QSC

is a cluster consisting of 256 CN. Each HP/Compaq ES-45 CN consists of four Alpha EV6 processors at 1.25 GHz running TRU64 and has 16 GByte RAM. The interconnection between CN is performed by a Quadrics interconnect in a Fat Tree topology.

Demonstration runs were performed for an initially ideal periodic cubic chunk of a face-centered cubic lattice interacting via a splined Lennard-Jones system ($r_{\rm cut} = 1.548$) using reduced units.[20] The initial temperature was set to $T = 2.6$, which is almost four times higher than the melting temperature, forcing the system to undergo quickly a solid–liquid phase transition, thus having lots of movements of particles between cells and PN. The system was integrated for 50–100 time steps with a step $dt = 0.01$ using the leapfrog version of the velocity-Verlet algorithm.[1] After every 10 steps the local potential energy (see Fig. 3), radial distribution functions, and various other properties have been calculated. As in previous ultra-large-scale MD demonstration runs,[14] calculations are performed in single precision.

The timings for runs on a different number of PN for a varying number of atoms is shown in Fig. 2. The time used per integration step on a fixed number of processors shows a good linear increase with the number of atoms for system sizes up to some 20 billion. However, we see a slight increase of the time used if the number of atoms per PN is fixed and the number of PN is increased. This has to do with communication bottlenecks: in the (Fat) Tree connection topology, the

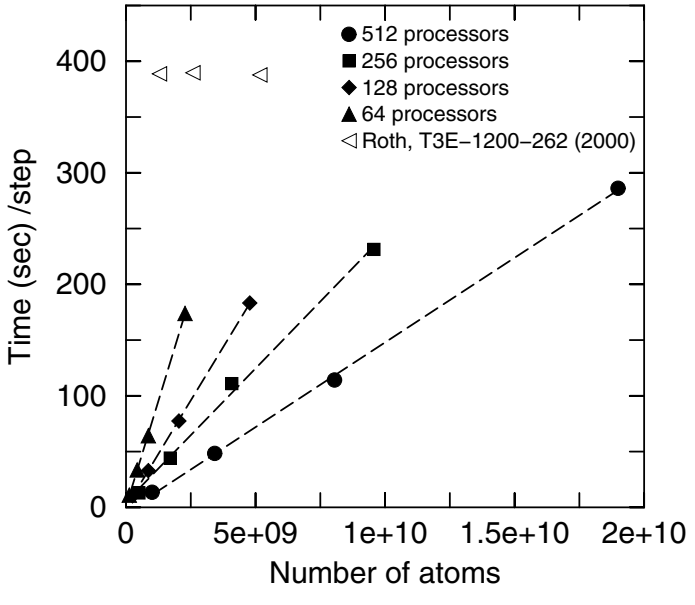

Fig. 2. Timings for runs on a different number of PN for varying number of atoms. Time used per integration step on a fixed number of processors increases well linearly with the number of atoms for system sizes up to some 20 billion. A slight increase of the time used can be seen, if the number of atoms per PN is fixed and the number of PN is increased. The data is compared to earlier demonstration runs by Roth *et al.* using 128, 256, 512 PN on a T3E (left to right).[14]

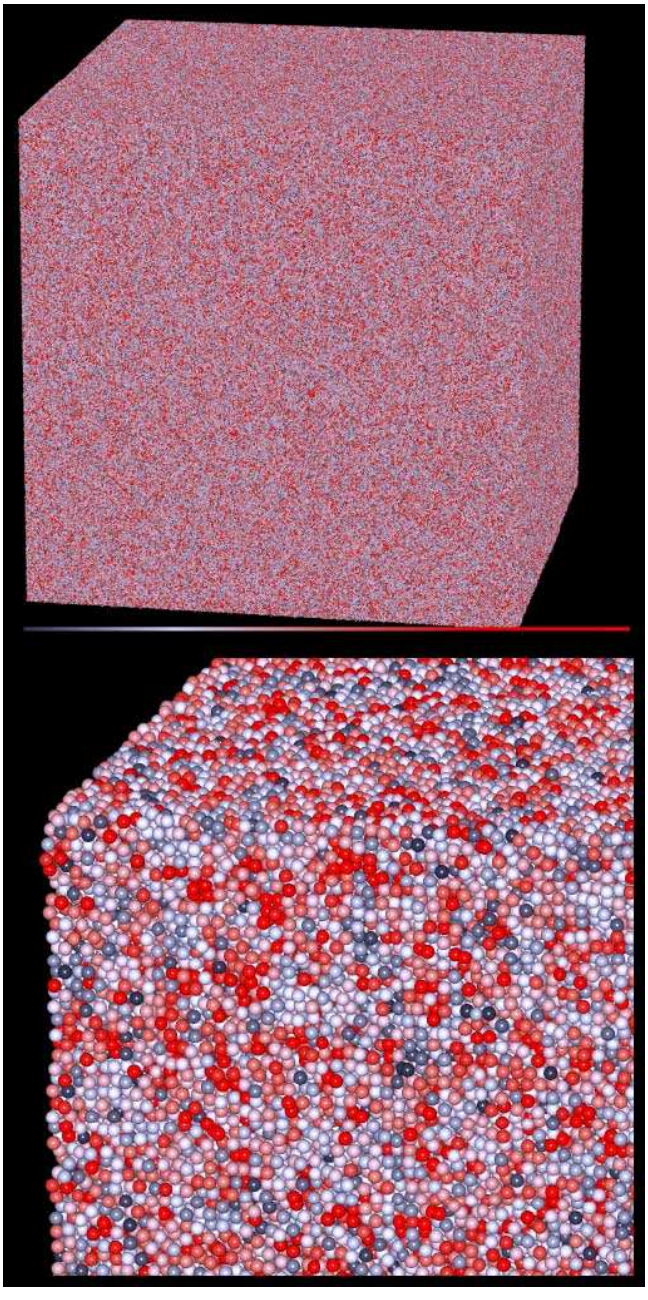

Fig. 3. ≈ 37 million particles rendered on four PN with a resolution of 5000 pixel by 5000 pixel (top). The bottom shows a close-up of the same picture file. The grayscale represents the potential energies of the atoms from -6 to -2 (grayscale version of the original color picture).

connections close to the root of the tree get heavier and heavier load if the global communication[b] increases, thus congestion near the root is possible. This can result in a delay of some messages if the global communication on the system increases. For other topologies like a 3D torus realized on the T3E, the communication topology is more geometric and the MD simulation geometry can be exactly mapped to the connection topology, which saves some communication time.

By comparison with data from 2000 by Roth *et al.*,[14] and assuming a similar performance of the MD codes, we see an improvement with respect to MD calculations of approximately a factor of five for an individual PN, which is in accordance with Moore's law postulating that computer speed doubles every 1.5 years. The amount of communication time between the PN as compared to the total iteration time varies from ≈10% for 37 million atoms per PN to ≈25% for 2 million atoms per PN. This fraction is larger than previously measured on the Connection Machine 5,[2,3] suggesting that communication speed is not developing as fast as processor speed, which is certainly the case with the current trend towards commodity-based cluster computing.

4. Sphere Rendering

We developed MD_RENDER,[21] a parallel scalable object (sphere, cylinder, wall) rendering capability in order to visualize results of large-scale atomistic simulations with arbitrarily high resolution and a 24-bit color-depth. The capability was realized by implementing an in-house developed graphics library in ANSI C, using MPI for the communication between PN. Since no (hardware-oriented) graphics library is needed, the software is portable and easy to implement on various architectures including large parallel platforms.

The approach allows for multiple point light sources and the color-vector $\mathbf{I}$ (as represented in the Red-Green-Blue (RGB) color model) for each surface element of an object is calculated according to the following basic expression for the four light components (emission, ambient, diffusive, and specular)[22]:

$$\mathbf{I} = \underbrace{\mathbf{M}_e}_{\text{emission}} + \underbrace{\mathbf{M}_a\mathbf{S}_a}_{\text{ambient}} + \underbrace{\mathbf{M}_d\mathbf{S}_d\mathbf{NL}}_{\text{diffusive}} + \underbrace{\mathbf{M}_s\mathbf{S}_s(\mathbf{NH})^{4M_{\text{shi}}}}_{\text{specular}} . \tag{1}$$

The emission component is the self-emission of the material and independent of any light source. The ambient component is simply the product of the ambient material $\mathbf{M}_a$ and the ambient light $\mathbf{S}_a$. The diffusive component is the product of the diffusive material vector $\mathbf{M}_d$ and the diffusive light $\mathbf{S}_d$, multiplied by the scalar-product between the surface-normal vector $\mathbf{N}$ and the light-vector $\mathbf{L}$ ($\mathbf{L}$ is the connection between the point on the surface of the object and the position of the light source). The specular part is the product of the specular material vector $\mathbf{M}_s$

[b]In our case, this means the synchronous MPI_send-and-receive() commands is necessary to transport particle information from one to another PN.

and the specular light $\mathbf{S}_s$, multiplied by the $(4M_{\mathrm{shi}})$th-power of the scalar-product between **N** and the half-vector **H**. Only this component depends on the position of the viewer, thus giving valuable information while rotating the scene. The half-vector **H** is half of the sum of **L** and the connection vector between the point on the surface of the object and the position of the viewer.[c] The shininess-parameter M_{shi} determines how focussed the specular reflection on the material is. For more than one light-source the diffusive and specular contributions will be added for each additional light-source.

Graphic libraries like OpenGL are basically implementing the same color model and are designed for general usage, such that objects like spheres are approximated by basic objects like triangles. This procedure is very time consuming and here we are taking another approach: simple objects like spheres, cylinders, and walls are treated as the basic elements and their surface gets directly transformed onto the two-dimensional picture. Only for each pixel on the picture, the needed vectors and color values of Eq. (1) are calculated and stored in three one-dimensional arrays, representing the RGB color of the pixels of the picture (per pixel shading). An additional one-dimensional array is used to store depth information of each pixel. In order to speed up the rendering process even further, the objects to render are sorted according to their depth with respect to the viewer, such that the closest objects are rendered first. This procedure makes it likely that the visible objects of the scene get rendered first. The color of a pixel of an object is only calculated according to Eq. (1) if it is visible with respect to the objects already rendered.

The parallelization is simply done by assigning the huge number of objects to different PN. Each PN renders the picture only with the objects assigned to it. After every PN has rendered its partial picture, the depth information is used to merge the partial pictures into one which can be written in a pixel-based picture file format. The parallelization has almost 100% efficiency since no communication during the rendering process is needed.

As an example we rendered the ≈37 million particles assigned to one PN of the 19 000 416 964 particle run after 50 integration steps (Fig. 3) on four QSC PN. It took less than two hours to render the scene with a resolution of 15 000 pixel by 15 000 pixel. The picture then was reduced to 5000 pixel by 5000 pixel resolution by averaging the color values of nine pixels. This procedure is a simple anti-aliasing and results in a "smoother" appearance of the scene. As a result, a high-resolution picture with a 24-bit color depth is available for further analysis by zooming into the scene. Figures rendered by a prototype version of MD_RENDER can be found in Ref. 10. This algorithm scales well with the number of objects. For packed scenes where not every object is visible, the algorithm scales even better. We have also implemented a capability which can create shadows of spheres on other objects,

[c]Note that the vectors **N**, **L**, and **H** are unit-vectors. Furthermore, if the scalar-product between such two vectors becomes negative, it is set to zero.

which for some circumstances helps understanding the scene better. However, this procedure does not scale linearly with the number of particles.

5. Conclusions/Outlook

We have demonstrated the ability to run and analyze atomistic simulations with up to some 20 billion atoms with the SPaSM library on the QSC machine. To the best of our knowledge this is the largest molecular-dynamics simulation reported to date. According to the good linear scaling of the simulation time with the number of particles, it would be possible to run 100 billion atoms on a large partition of the current Q-machine. This potential enables future simulation of a physical problem on the μ-scale which has been so far only been theoretical investigated by continuum methods.

The flexibility of the SPaSM library can be exploited in other ways, for instance by opening up the possibility of large-scale agent-based simulations. Smaller-scale agent-based models for the spread of epidemics[23] can now scale up to encompass the entire world population of $\approx$6.3 billion people.

Acknowledgments

K. Kadau thanks Johannes Roth for many discussions on large-scale computing. Support by the DOE through LDRD-20020053DR and the ASCI program, QSC access through the Institutional Computing Program is gratefully acknowledged. US Department of Energy contract W-7405-Eng-36.

References

1. D. C. Rapaport, *The Art of Molecular Dynamics Simulation* (Cambridge University Press, Cambridge, 1995).
2. P. S. Lomdahl, P. Tamayo, N. Grønbech-Jensen and D. M. Beazley, *Proc. Supercomputing 93*, ed. G. S. Ansell (IEEE Computer Society Press, Los Alamitos, CA, 1993), p. 520.
3. D. M. Beazley and P. S. Lomdahl, *Parallel Comput.* **20**, 173 (1994).
4. D. M. Beazley and P. S. Lomdahl, *Comput. Phys.* **11**, 230 (1997).
5. http://bifrost.lanl.gov/MD/MD.html.
6. B. L. Holian and P. S. Lomdahl, *Science* **280**, 2085 (1998).
7. S. J. Zhou, D. L. Preston, P. S. Lomdahl and D. M. Beazley, *Science* **279**, 1525 (1998).
8. P. Gumbsch and H. Gao, *Science* **283**, 965 (1999).
9. M. Marder, *Comput. Sci. Eng.* **1**, 48 (1999).
10. T. C. Germann, B. L. Holian, P. S. Lomdahl and R. Ravelo, *Phys. Rev. Lett.* **84**, 5351 (2000).
11. K. Kadau, T. C. Germann, P. S. Lomdahl and B. L. Holian, *Science* **296**, 1681 (2002).
12. F. F. Abraham, R. Walkup, H. Gao, M. Duchaineau, T. D. De La Rubia and M. Seager, *Proc. Natl. Acad. Sci. USA* **99**, 5777 (2002).
13. F. F. Abraham, R. Walkup, H. Gao, M. Duchaineau, T. D. De La Rubia and M. Seager, *Proc. Natl. Acad. Sci. USA* **99**, 5783 (2002).
14. J. Roth, F. Gähler and H.-R. Trebin, *Int. J. Mod. Phys. C* **11**, 317 (2000).

15. C. L. Kelchner, S. J. Plimpton and J. C. Hamilton, *Phys. Rev. B* **58**, 11085 (1998).
16. http://public.lanl.gov/consult/qsc.
17. http://www.unix-mcs.anl.gov/mpi.
18. D. M. Beazley, *Python Essential Reference* (New Riders, Indianapolis, 2001).
19. http://www.swig.org.
20. B. L. Holian *et al.*, *Phys. Rev. A* **43**, 2655 (1991).
21. http://www.thp.Uni-Duisburg.DE/∼kai/index_4.html.
22. T. Möller and E. Haines, *Real-Time Rendering* (A K Peters Ltd., Natick MA, 1999).
23. M. E. Holloran, I. M. Longini, Jr., A. Nizam and Y. Yang, *Science* **298**, 1428 (2002).