Modeling Parallel Molecular Simulations on Amazon EC2

Xiangqiang Xu Gabriel Dunham Xinghui Zhao
Washington State University
{xiangqiang.xu, gabriel.dunham, x.zhao}@wsu.edu

David Chiu
University of Puget Sound
dchiu@pugetsound.edu

Jie Xu
University of Illinois at Chicago
jiexu@uic.edu

***Abstract*—Cloud computing has been widely used by computational scientists and engineers as a means to run large-scale simulations while circumventing capital investment of hardware. However, a challenging problem is to accurately estimate how much cloud resources that a specific computation requires, in order to execute computations in a cost-effective way. In this paper, we use a real-world molecular dynamics (MD) simulation as a motivating scenario and present our work in modeling parallel execution of such a computation on the cloud. Our model estimates the workload of an MD simulation at fine-grained detail, and based on that estimate, calculates the time required to run the simulation. The accuracy of the model has been evaluated using various types of MD simulations on different scales.**

I. Introduction

The scientific computing community has long relied on the presence of high-performance systems, *e.g.*, clusters and supercomputers, and today's scientific applications enjoy the plethora of parallel and distributed frameworks, including MPI, Map-Reduce variants [1], workflow management systems [2] built on top of them. The emergence of the cloud has been timely. Given the presence of public clouds, such as Amazon EC2 [3], users can obtain immediate results without requiring an initial capital investment of costly computational equipment.

As a result, there is now abundant interest in exploiting cloud resources to carry out large-scale scientific computations. One such example is Molecular Dynamics (MD) simulation [4], [5], which aims to model the complex behavior of liquid molecules and biomolecules in nanometer-scale environments. MD applications simulate the movements of molecules based on Newton's laws of motion [6], using the forces and distances between molecules and their masses.

A better understanding of the interactions and properties of these small atoms could lead to a great impact on various areas, such as the Human Genome Project, gene chips, personalized molecular diagnostics, and DNA sequencing. However, it is extremely difficult to perform actual experiments with liquid molecules and biomolecules at the nanometer scale. These experiments often involve expensive ($\approx$ million-dollar range) large-scale equipment, such as a scanning-electron microscope (SEM), a transmission-electron microscope (TEM), so they are not always accessible to researchers. Therefore, molecular dynamics (MD) simulation has become a powerful tool to investigate molecules at the nanooscale. While simulation can reduce some costs of doing science, to obtain precise results, MD simulations often require massive amounts of CPU cycles, pushing the need for execution on parallel and distributed systems.

Taking advantage of the computational resources on public clouds, scientists can execute their MD applications without having to own high-end computational resources. However, here is often still a need to estimate these costs. This challenge is more pronounced when the scientist has certain constraints, such as accuracy, performance, and/or budget limitations. From a user's point of view, being able to predict the behavior of the cloud to estimate the performance and cost of their simulations is critical.

In this paper, we propose modeling the performance of a popular parallel molecular dynamics simulator on clouds. Specifically, we have modeled MD applications running in LAMMPS (Large-scale Atomic/Molecular Massively Parallel Simulator) [7] on the Amazon EC2 cloud. LAMMPS can simulate a wide range of molecular systems, typically from a few particles up to millions or billions of particles. The output of LAMMPS includes locations, trajectories, forces and energy of each particles in the 3D space. Practically, the scale of a LAMMPS simulation is limited by computing power, time, and cost. Our performance model takes a LAMMPS configuration file as input and predicts with high accuracy the execution time on Amazon EC2 clouds, as a function of N, the number of nodes used to run the simulation.

The remainder of this paper is organized as follows. In Section II, we describe some background information about the MD simulation software we use in our work, LAMMPS (Large-scale Atomic/Molecular Massively Parallel Simulator). In Section III, we present details of our performance model. To evaluate our model, experiments have been carried out on Amazon EC2 clouds using various types of simulations, and the experimental results are presented in Section IV. We conclude and discuss future works in Section V.

II. Background

LAMMPS (Large-scale Atomic/Molecular Massively Parallel Simulator) [7] is a widely-used software simulator for molecular dynamics. Using LAMMPS, scientists can simulate a variety of systems, including solid-state materials (*e.g.*, metals and semiconductors), soft matter (*e.g.*, biomolecules and polymers), and coarse-grained or mesoscopic systems. The simulation can be executed at different scales, such as stomic, meso, and continuum scales.

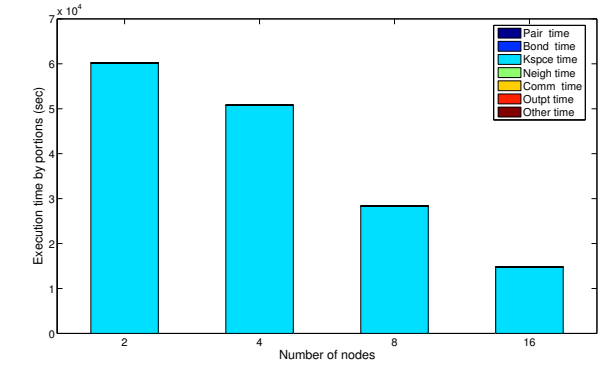
We chose LAMMPS in our work in part because of its popularity, and more importantly, due to its flexibility in supporting parallel/distributed execution. LAMMPS can run applications on single processors, or in parallel on more advanced

hardware, *e.g.*, a high-performance cluster. LAMMPS parallelizes MD applications using message-passing techniques (MPI) [8] by spatially decomposing the simulation domain [9]. In parallel execution, the simulation domain is partitioned into a number of small 3D sub-domains, and each of which is assigned to a processor for execution. Each sub-domain considers its neighbor sub-domains as *ghost* atoms and store their information at its own processor. Using this spatial-decomposition technique, LAMMPS can easily parallelize an MD simulation and execute it in parallel. This also makes it easier to run these simulations using resources on the clouds.

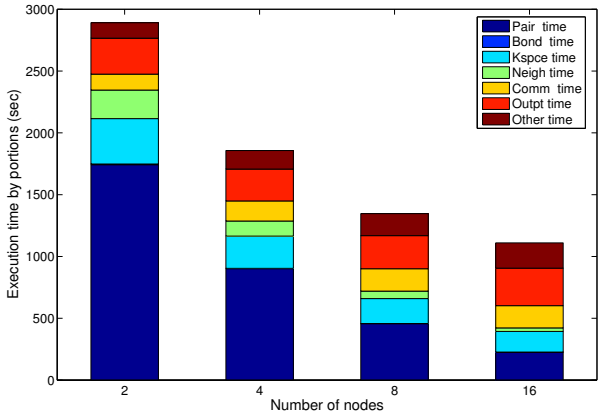
Another desirable feature in LAMMPS is that it records fine-grained performance information during the execution and provides it at the end of the execution in a log file. Table I shows a sample LAMMPS output for a MD simulation which involves $24,048$ atoms and runs on 16 processors.

LAMMPS Output	
Loop time of 1109.77 on 16 procs for 10000 steps with 24048 atoms	
CPU Time	Pair time (%) = 226.106 (20.3742) Bond time (%) = 0.776199 (0.0699424) Kspce time (%) = 166.729 (15.0238) Neigh time (%) = 28.8686 (2.60131) Comm time (%) = 179.684 (16.1912) Outpt time (%) = 303.231 (27.3238) Other time (%) = 204.372 (18.4157) FFT time (% of Kspce) = 70.3567 (42.1981) FFT Gflps 3d (1d only) = 1.07363 9.16506
Atoms Per Processor	Nlocal: 1503 ave 1524 max 1489 min Histogram: 2 0 5 0 5 1 1 1 0 1 Nghost: 9973.75 ave 10048 max 9919 min Histogram: 3 2 0 4 2 1 0 1 1 2 Neighs: 530928 ave 543321 max 516004 min Histogram: 3 1 1 2 0 0 1 3 2 3
Aggregated Statistics	Total # of neighbors = 8494854 Ave neighs/atom = 353.246 Ave special neighs/atom = 2.34032 Neighbor list builds = 1020 Dangerous builds = 0

TABLE I: LAMMPS Sample Output



(a) Uniform Simulation (atoms: 512; timesteps: 5k)



(b) Peptide Simulation (atoms: 24048; timesteps: 10k)

Fig. 1: CPU Time Breakdown of Example MD Simulations: Uniform and Peptide

As shown in Table I, LAMMPS gives the total execution time (loop time) and three sets of information about the execution. The first section gives the breakdown of the CPU runtime (in seconds) into major categories. The second section lists the number of owned atoms ($Nlocal$), ghost atoms ($Nghost$), and pair-wise neighbors stored per processor. The max and min values is the spread of these values across processors over a 10-bin histogram. The total number of histogram counts is equal to the number of processors. The last section lists aggregate statistics for pair-wise neighbors and special neighbors of which LAMMPS keeps track. The number of times that neighbor lists is rebuilt during the run is given, as well as the number of potentially "dangerous" rebuilds, which are the rebuilds triggered by atom movements. They are considered dangerous because it is likely that force interactions are missed by atoms moving beyond the neighbor skin distance before a rebuild takes place.

We chose two representative MD applications and executed them under varying hardware settings, *i.e.*, number of nodes. The first example is an MD simulation in which the atoms are *uniformly distributed* in the simulation domain. Figure 1(a) illustrates the CPU time breakdown of the execution of this simulation, when the number of processors (nodes) are 2, 4, 8, and 16, respectively. In these runs a significant amount of the execution time is taken by the *Kspace* category, which is for computing long-range coulombic interactions. Other categories only contribute a small amount of execution time.

The second example is an MD simulation with a different atom distribution pattern: *peptide simulation*. This simulation models the interactions of peptide molecules in a water environment. The dynamics of peptides in water plays a critical role in determining protein structures and functions. This information will provide profound implications in clinical research for curing various diseases. Because there are different types of molecules and the distribution of these molecules is non-uniform, the CPU-time breakdown shows a much different pattern (Figure 1(b)).

For peptide simulation, a main contributor of the execution time is *pair time*, which is for calculating non-bonded or pair-wise forces between atoms. Other contributing categories include *Kspace time*, *neighbor time, communication time* and *output time*. The neighbor category represents the time used to build and store neighbor lists. The communication category represents the time that it takes to perform inter-processor communication, typically containing ghost atom data. This usually involves MPI message exchanges with six neighboring

processors in the 3D logical grid of processors mapped to the simulation box. The output category represents the time that it takes to generate output from the simulation. More detailed information about these categories can be found in LAMMPS Developer Guide [10]. The CPU breakdown of the execution time provided by LAMMPS is critical for our work. It enables fine-grained modeling on computation and communication, as described in Section III.

III. Performance Model

A number of frameworks and software packages have been developed for supporting cloud-based MD executions, such as AceCloud [11], CycleCloud [12], and GROMACS 4.5 [13]. However, estimating the performance of large-scale simulations remains a challenge. Being able to predict the performance is desirable, as scientists may need that information to estimate their costs, time constraints, and the resource requirements. In this section, we describe the performance model we developed for MD simulations and the evaluation of the model on the Amazon EC2 cloud.

Table II shows the notations that are used in our performance model. a denotes the number of atoms in the simulation, representing the size of the computation. N is the number of nodes on which the simulation executes. r_s is the extended cutoff distance that is defined in LAMMPS. It is a constant for a specific input. t is the time steps, *i.e.*, iterations to run the simulation. The work of the computation (W_{comp}) and work of communication (W_{comm}) are the workload for the computation component and communication component of the simulation, respectively. R_{comp} and R_{comm} are the rate of computation and communication, which are obtained from the baseline execution and used in performance prediction.

Parameters	Descriptions
a	number of atoms or particles
N	number of nodes
r_s	extended cutoff distance
t	time steps
W_{comp}	work of computation
W_{comm}	work of communication
R_{comp}	rate of computation
R_{comm}	rate of communication

TABLE II: Model Parameters

As the first step towards performance prediction, we model the workload of an MD simulation for computation and communication. Based on the spatial-decomposition algorithm [9] used in LAMMPS, we derive the workload for computation and communication in a single time step, shown in equation 1. The first term represents the workload for constructing lists of interacting pairs for the sub-domain, and workload for calculating the atom positions which will be sent to other neighbor sub-domains. The second term represents the workload of updating atom positions in the sub-domain. Note that for each rectangular 3D sub-domain, there are 6 neighbor sub-domains.

$$W_{comp} = 2\left(\frac{a}{2N} + 6r_s\left[\frac{a}{N}\right]^{2/3}\right) + \frac{a}{N} \quad (1)$$

Similarly, we derive the workload for communication, as shown in equation 2. This represents the workload for exchanging atom positions across all 6 boundaries of the sub-domain.

$$W_{comm} = 6r_s\left[\frac{a}{N}\right]^{2/3} \quad (2)$$

For a given hardware configuration and a specific simulation, the two workload parameters W_{comp} and W_{comm} are fixed. We then use a baseline execution to derive R_{comp} and R_{comm} as follows,

$$R_{comp} = \frac{W_{comp}}{(T_{exec} - T_{comm})} \quad (3)$$

$$R_{comm} = \frac{W_{comm}}{T_{comm}} \quad (4)$$

where W_{comp} and W_{comm} are calculated using equation 1 and 2 for the baseline execution, T_{exec} is the total execution time of the simulation, and T_{comm} is the communication portion of the execution time, which can be obtained from the CPU breakdown time of the baseline execution.

Using R_{comp} and R_{comm}, we can predict the execution time of a new simulation in a different hardware setting, as follows,

$$\begin{aligned} T_{predict} &= t \times \left(\frac{W_{comp}}{R_{comp}} + \frac{N \times W_{comm}}{R_{comm}}\right) \\ &= t \times \left(\frac{\frac{2a}{N} + 12r_s[\frac{a}{N}]^{2/3}}{R_{comp}} + \frac{N \times 6r_s[\frac{a}{N}]^{2/3}}{R_{comm}}\right) \end{aligned} \quad (5)$$

Equation 5 gives the predicted execution time of a simulation which is running on N nodes for t time steps.

IV. Evaluation

We have evaluated our performance model using different types of MD simulations, running on Amazon EC2 clouds at different scales. The Amazon nodes we use to run our experiments are all `m1.large`, which have 2 cores (4 ECUs), 7.5GB RAM, and 850 GB disk.

To evaluate the accuracy of our model, we run both uniform and peptide simulations on `m1.large` nodes using different settings ($N = 2, 4, 8, 16$), and compare the actual execution time with the predicted execution time $T_{predict}$ derived using equation 5. Figure 2 shows the experimental results for uniform simulations of different sizes with 5000 time steps. We then run the case with 16384 atoms for different time steps, and the results are shown in Figure 3. For larger simulations (in size or in time steps), the model tends to over-estimate the execution time for the 2-node case, but for most of the experiments, it can accurately predict execution time.

We have also evaluated our model using a peptide simulation with 24048 atoms non-uniformly distributed. The results illustrate that our performance model is accurate for this non-uniform simulation, as shown in Figure 4.

The performance model can be used by scientists who would like to run their simulations on the clouds. Before they use the model, they need to run a small simulation on the computational nodes, in order to establish a baseline. After that they can use the model to predict the performance of their simulation under different hardware settings, and estimate the cost of such executions. Being able to accurately predict the resource requirements and performance of a simulation

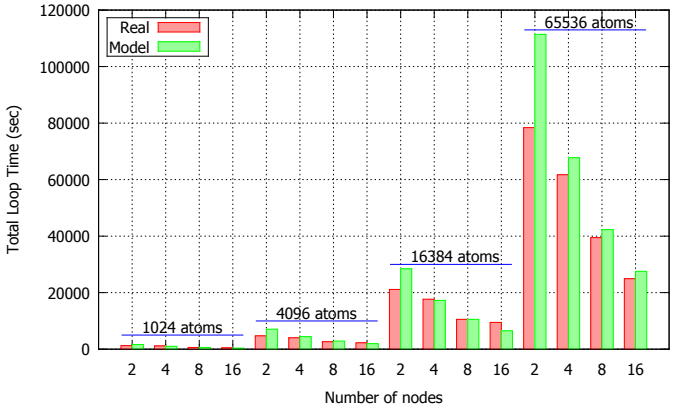


Fig. 2: Uniform Simulation (t = 5000)

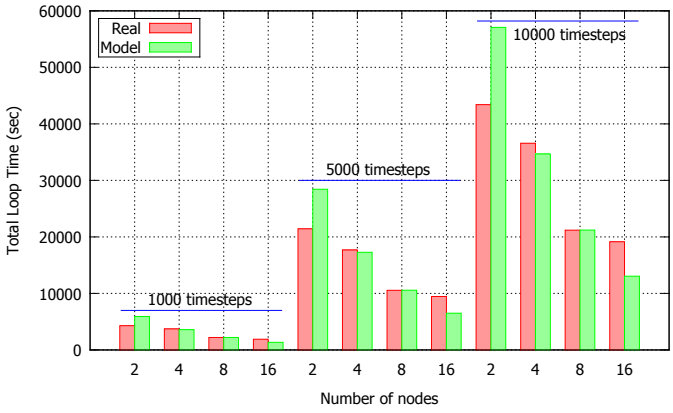


Fig. 3: Uniform Simulation (a = 16384)

before its execution is critical, especially when the size of the simulation scales.

V. Conclusion

Molecular dynamics (MD) simulation is a widely used tool for modeling the properties and interactions of liquids, solids, and molecules. Recently, there has been increasing interest in running MD simulations using shared resources on the cloud. In this paper, we model the computation and communication components of an MD simulation separately to predict its execution time on Amazon EC2. Experimental results show that our performance model is accurate and reliable for various types of MD simulations. For future work, we will develop resource allocation algorithms for MD simulations, by taking into consideration both performance and budget constraints. The performance model presented in this paper is a critical step towards this direction.

Acknowledgment

The authors would like to thank the generous support of an *Amazon Web Services Research Grant*.

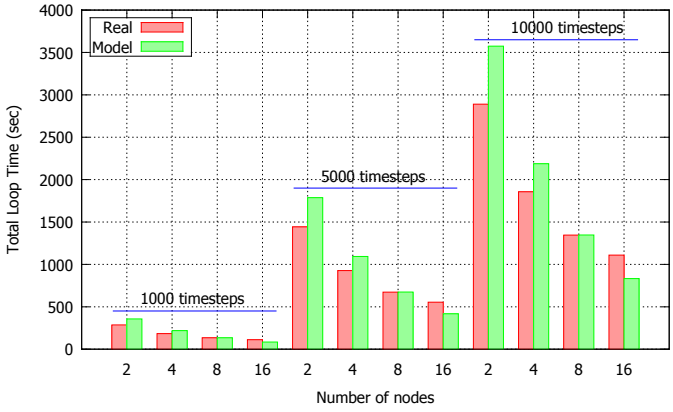


Fig. 4: Peptide Simulation (a= 24048)

References

[1] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," in *Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation - Volume 6*, ser. OSDI'04. Berkeley, CA, USA: USENIX Association, 2004, pp. 10–10.

[2] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, ser. HotCloud'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 10–10.

[3] "Amazon elastic compute cloud, http://aws.amazon.com/ec2."

[4] F. F. Abraham, "Computational Statistical Mechanics Methodology, Applications and Supercomputing," *Advances in Physics*, vol. 35, no. 1, pp. 1–111, 1986.

[5] J.-P. Ebejer, S. Fulle, G. M. Morris, and P. W. Finn, "The Emerging Role of Cloud Computing in Molecular Modelling," *Journal of Molecular Graphics and Modelling*, vol. 44, pp. 177–187, 2013.

[6] W. F. van Gunsteren and H. J. C. Berendsen, "Computer Simulation of Molecular Dynamics: Methodology, Applications, and Perspectives in Chemistry," *Angewandte Chemie International Edition in English*, vol. 29, no. 9, pp. 992–1023, 1990.

[7] S. Plimpton, P. Crozier, and A. Thompson, "LAMMPS: Large-Scale Atomic/Molecular Massively Parallel Simulator," *Sandia National Laboratories*, vol. 18, 2007.

[8] W. Gropp, E. Lusk, N. Doss, and A. Skjellum, "A High-Performance, Portable Implementation of the MPI Message Passing Interface Standard," *Parallel computing*, vol. 22, no. 6, pp. 789–828, 1996.

[9] S. Plimpton, "Fast Parallel Algorithms for Short Range Molecular Dynamics," *Journal of Computational Physics*, pp. 1–19, 1 March 1995.

[10] "Sandia Corporation: LAMMPS Developer Guide," pp. 4–5, 23 Aug 2011.

[11] M. Harvey and G. De Fabritiis, "AceCloud: Molecular Dynamics Simulations in the Cloud," *Journal of chemical information and modeling*.

[12] "CycleCloud - Accelerating Molecular Dynamics with Large Scale HPC." [Online]. Available: http://cyclecomputing.com/

[13] S. Pronk, S. Páll, R. Schulz, P. Larsson, P. Bjelkmar, R. Apostolov, M. R. Shirts, J. C. Smith, P. M. Kasson, D. van der Spoel, *et al.*, "GROMACS 4.5: A High-Throughput and Highly Parallel Open Source Molecular Simulation Toolkit," *Bioinformatics*, p. btt055, 2013.