Computational Materials Science 144 (2018) 322–330

Contents lists available at ScienceDirect

Computational Materials Science

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

Editor's Choice

A scalable parallel framework for microstructure analysis of large-scale molecular dynamics simulations data

Guoqing Wu [*], Haifeng Song, Deye Lin [*]

Institute of Applied Physics and Computational Mathematics, Fenghao Dong Road No. 2, Haidian District, Beijing, China
CAEP Software Center for High Performance Numerical Simulation, Huayuan Road No. 6, Haidian District, Beijing, China

A R T I C L E   I N F O

Article history:
Received 12 September 2017
Received in revised form 13 December 2017
Accepted 17 December 2017

Keywords:
Molecular dynamics
Microstructure analysis
Distributed-memory
Large scale

A B S T R A C T

Molecular dynamics (MD) simulation is a fundamental tool in computational materials science. With the development of parallel supercomputers, researchers can access the detail atomic responses of materials with MD simulations at unprecedented scales of physical and time. However, the size of generated output datasets is also growing rapidly, which poses a serious challenge for traditional data analysis methods. Therefore, parallel analysis methods to support faster and more scalable manipulation of atomic data are desperately needed. In this paper, we present a scalable parallel framework to meet the requirements. It allows users to implement a parallel analysis program using a simple interface, make use of the existing sequential analysis codes, and carry out distributed-memory post-simulation data analysis. We have integrated three popular microstructure characterization methods (lattice structure identification, Voronoi analysis, and Wigner-Seitz defect analysis) based on this framework. Performance evaluations run on massively parallel process computers with 10^9 atoms on up to 1024 processor cores demonstrate the scalability and efficiency of the proposed framework. The proposed framework is helping accelerate large-scale MD data analysis to a new level.

1. Introduction

Molecular dynamics simulations are widely used to study the dynamic behavior of multiple-particles systems and have been successfully applied in many fields of material science research [1–4]. With parallel supercomputers popularly deployed around the world, researchers are able to conduct these simulations at unprecedented scale and performance. For example, using modern software [5,6] and hardware [7] techniques, MD codes can run efficiently on petascale computers simulating important physical processes with billions of atoms [8]. Accompanied with these large-scale simulations, output datasets (i.e., atomic configurations or trajectories) on the order of gigabytes or terabytes are produced—a scale that is several orders of magnitude larger than what can be deal with on an average desktop computer. In order to use these vast and information-rich datasets to gain scientific insights, advanced computational analysis, visualization and interpretation technologies are urgently required.

While great efforts have gone into the design of MD simulation algorithms and tools, the analysis technology of output datasets is unable to keep pace with the rapid development of scalability and performance of MD simulations, and is usually relegated to shared-memory processing. Within the last decade, a considerable amount of atomistic analysis software packages are released. Stukowski designed a 3D visual analysis tool named OVITO [9] for post-processing atomistic data of MD and Monte Carlo simulations. OVITO integrates several microstructure analysis functions such as atomic strain analysis, common neighbor analysis, cluster analysis, dislocation analysis, and Wigner-Seitz defect analysis [10]. Li developed an efficient atomistic configure viewer named AtomEye [11]. It provides many customizable functions to identify and accentuate certain structural characters of MD datasets, and it is a general tool to survey the microstructures evolutions of materials under specific loading conditions. Naveen et al. introduced MDAnalysis [12], a framework for structural and temporal analysis of MD trajectories. Robert et al. released a library named MDTraj [13] which is comparable to MDAnalysis. The code supports a wide range of data formats and provides many trajectory analysis capabilities including root mean squared deviation, DSSP secondary structure assignment and the extraction of common order parameters. Both MDAnalysis and MDTraj are Python-based libraries, facilitating rapid code development by using the extensive packages in the scientific Python ecosystem. Several other software packages exist (e.g., CHARMM [14], VMD [15], and GROMACS

* Corresponding authors.
E-mail addresses: wu_guoqing@iapcm.ac.cn (G. Wu), lin_deye@iapcm.ac.cn (D. Lin).

https://doi.org/10.1016/j.commatsci.2017.12.048
0927-0256/© 2017 Elsevier B.V. All rights reserved.

[16]) that enable users to analyze MD datasets. However, all tools mentioned above are designed for doing atomistic post-processing in a shared-memory environment and assume that the entire dataset to be analyzed could fit into the main memory of computers. Efficient and effective though they are for analyzing a dataset with million scale atoms, these analysis tools lack the necessary scalability and performance to handle very larger MD systems. The widening gap between high-performance MD simulations and data analysis has mounted to such a point that novel techniques need to be developed to tackle the big data problem. Otherwise, it would hamper scientists' ability to understand and interpret large-scale simulation results, hereby defeating the purpose of developing large-scale MD simulations [17].

The analysis of large-scale MD simulations is computationally expensive and requires sufficient working memory to accommodate the entire atomistic dataset. To eliminate this computation and memory constraint, a distributed-memory scalable parallel computational solution is the only feasible approach. Actually, many researchers agree that data analysis algorithms not only need to be parallel, but also must scale to the same size and efficiency as the simulations [18]. By distributing the atomistic data and analysis tasks over more computer processors, user's post-processing requests can get faster response, accesses to all data at full resolution are possible, and scientific discoveries are subsequently accelerated. By contrast, many common analysis frameworks (e.g., MapReduce [19], Paraview [20], and VISIT [21]) that have been developed for distributed-memory environments are not directly targeted for MD simulation studies.

In this paper, we focus on overcoming the bottlenecks encountered by microstructure analysis programs that deal with large-scale atomistic datasets using hundreds to thousands of processor cores. At first glance, it might appear that parallelizing traditional analysis code could solve the problem. However, the analysis needs for material scientists are always varied, it is time-consuming and inefficient to develop parallel analysis program in a case-by-case mode. On the other hand, parallel computing requires substantial programming efforts and material researchers may not be experts in computer science. These are dual challenges posed by large-scale MD data analysis. To address these challenges, we have developed a flexible processing framework, which consists of an interface that allows a user to write analysis programs sequentially, and the machinery that ensures these programs execute in parallel automatically. We demonstrate how sequential analysis algorithms and data-parallel can be integrated into a framework to solve mentioned challenges. Our main contribution is an alternative end-to-end solution to enable microstructure analysis of MD datasets at previously unexplored scales. To the best of our knowledge, no other existing MD post-simulation data analysis tools provide similar efficiency and scalability to support parallel analysis on very large supercomputers.

2. Overview of the scalable parallel analysis framework

Before we are going to discuss the framework and its design, the data-parallel environment is described firstly, and then we briefly introduce software packages on which we depend for this work.

2.1. Data parallelism

Most of the current supercomputer architectures consist of multiple independent compute nodes, each of which is configured with several multiple-core CPU processors sharing in-node memory (see Fig. 1). Different processors have access only to their local memory, and access to remote data is accomplished by passing messages (i.e., Message Passing Interface, MPI). Based on this architecture, methods of data parallelism can be classified into two broad categories: shared-memory model (or thread-level parallelization) and distributed-memory model (or process-level parallelization). In shared-memory model, little effort is required to break up a problem into parallel tasks because thread support libraries (e.g., OpenMP) shield thread-parallel details for users. However, shared-memory model is constrained to a single node and is usually memory-bound. Thus, thread-level parallel programs address many of limitations of scalability. Conversely, distributed-memory model utilizes system-wide compute nodes and CPU cores in order to satisfy scalability constraints. It distributes the data across many different nodes which operate the data in parallel, and thus could scale to very large systems.

2.2. Flexible processing framework

In general, scalable parallel analysis of large-scale MD datasets poses several hurdles among which the most pressing challenges are: the decomposition of the analysis problem among a large number of processors, the efficient data exchange among them, scalable analysis algorithms, and data transport between processors and a parallel storage system. Inspired by the work of Tu et al. [17] and Google's MapReduce framework [19], our main idea is to provide a data-processing architecture that supports parallel execution of microstructure analysis programs and a user-friendly, unified programming interface that allows users to add new functionalities easily. The analysis framework encapsulates modules that are reusable for different analysis applications, while

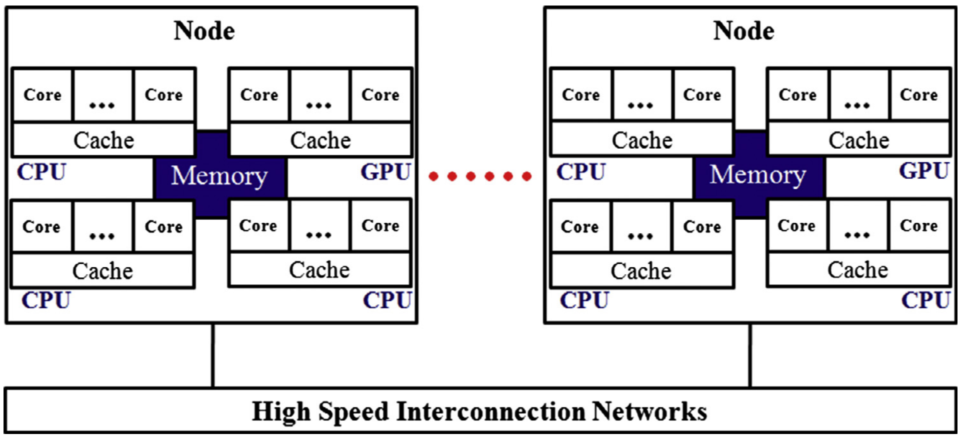

Fig. 1. The typical distributed-memory infrastructure of supercomputers.