Eco-Comp: Towards Responsible Computing in Materials Science

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Bridging the time and length scales and the use of large molecular dynamics (MD) simulations in material science is expected to surge in the next few years, partially due to the development of highly accurate machine learning inter-atomic potentials that enable the simulation of multi-million atomic systems. We also expect a high demand for material science simulations using multiple nodes within high-performance computing facilities (HPCs) due to their computational intensity. Through the analysis of catalysis simulation setups consisting of bulk metallic systems with adsorbed molecular species on the surface, we identified various factors that affect parallel computing efficiency. To foster sustainable and ethical computing practices, this study employs the Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS) to find the optimal allocation of computing resources based on the simulation input. We thus propose guidelines to promote responsible computing within HPC architecture: Eco-Comp is a user-friendly automated Python tool that allows material scientists to optimize the power consumption of their simulations using one command. This tutorial gives a broad overview of the Eco-Comp software and its potential use for the material science community through an interactive guide.

1 Introduction

High-Performance Computing, or HPC, has revolutionized the field of science, enabling researchers and scientists to perform simulations that were once impossible. Molecular dynamics (MD) simulation has become a key aspect of focus in material science, as HPC are utilized to simulate, for example, catalytic reactions, enzyme active site exploration, and mechanical properties on the atomic scale. The emergence of easy-to-use machine learning potentials (MLP), such as the Machine-Learned Spectral Neighbor Analysis Potential (ML-SNAP) is indicative of the growing accessibility and versatility of this tool in the scientific community. It allows scientists and engineers to understand key chemical reactions at the atomic level relevant to the development of green energy and environment solutions such as catalysts, batteries, and solar cell. Running MD simulations with millions of atoms and time-steps is extremely time-consuming and resource-demanding. Thus, implementing responsible high-performance computing requires users to pay more attention to allocating the optimal computing resources for parallel atomic simulations. The Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS) is renowned for its extensible and substantial documentation, and has seen a surge in adoption owing to the incorporation of new packages and functionalities tailored to ML-driven simulations. While this expansion offers users exciting opportunities to explore complex biological and physical systems, it also underscores the importance of benchmarking and resource optimization. This work focuses on using LAMMPS with the aim of finding the optimal computing resource usage. We start by analyzing and profiling bulk metallic systems to understand the relationship between the number of nodes used in the calculation and the LAMMPS parallelization efficiency. The Kokkos package and other kernel libraries are utilized to assess their impact on computational speed and efficiency.

We find that surface reactions involving a metallic slab and a vacuum with reacting molecules (in this case, carbon monoxide) suffer significant degradation of the parallel efficiency upon increasing the number of nodes. Parallel efficiency drops by 30% if ten nodes are used on atomic systems of 20-60 thousand atoms. Profiling analysis indicates wasteful looping within specific functions and an increase in delays of their execution time as a primary cause behind performance degradation. After a series of calculations related to parallel efficiency, we build a set of guidelines for responsible computing to be used in HPC environments. Performance analysis of these calculations have been used to develop an automated tool that can recognize, evaluate, and recommend choices on resource usage to a user based on the comparison, enabling them to save precious compute time and energy. This work aims to lower the carbon footprint of computational loads used in developing some technologies of our future and ensures that it will be built upon a foundation that prioritizes sustainability and the ethics of responsible computing.

2 Software

Eco-Comp is a software that finds the optimal allocation of computing resources through benchmarking statistics run in an automated manner. With great emphasis on ease of use, especially for new users who do not possess the technical prowess of the architecture of the software, the present-day best practice programming language of Python was chosen as our foundation. After a simple installation, the software takes inputs from the user regarding details of the supercomputer and the resources available. Using this information, a unique Python script is created for the user, which can be used to run their simulation data. It automatically makes submissions via a job scheduler hence benchmarking on the user's specific production setup. This process, that lasts few seconds, provides an accurate metric related to parallel efficiency. In addition, based on the data extracted from the job submissions and the optimal configuration of computing resources, a ready-to-use customized job submission file (Slurm scheduler bash file) is created for the user tailored for their simulation needs.

3 Methods

Eco-Comp can extract the required information from simulation data, and run bench-marking based on the system complexity. This tool should then extract data from job submissions, suggest the optimal configuration of use to the user, and, more importantly, provide them with a ready-to-use customized job scheduler submission file based on their needs.

We used LAMMPS to evaluate the impact of characteristics such as vacuum space in different atom systems, type & number of atoms present, the impact loaded packages such as Kokkos, ReaxFF, and the Machine Learning SNAP (ML-SNAP) potential. Parallel efficiency using strong-scaling speedup is calculated to understand performance degradation across various types of simulations, after which Google Performance Tools are used to analyze potential bottlenecks from the profiling data.

To calculate the strong scaling speedup on N nodes:

$$Strong\ Scaling\ Speedup\ on\ N\ nodes = \frac{Time\ on\ 1\ node}{Time\ on\ N\ nodes}$$

To calculate the parallel efficiency with N nodes as a percentage:

$$Parallel\ Efficiency\ with\ N\ nodes\ (\%) = \frac{(Strong\ Scaling\ Speedup\ on\ N\ nodes) \times 100}{Number\ of\ nodes}$$

This provided various metrics that were used to understand how these subtle changes affected runs and their parallel efficiency. Tests showed us that utilizing the Kokkos package improved efficiency and speed by almost two-fold. When a vacuum was present in the atomic system, there was an increase of 30% computational speed when using the ReaxFF potential, whereas there was a greater 45% increase when using the Machine Learning Snap Potential. Through Google Performance Tools, we established 66% as a threshold after profiling analysis; any efficiency below this value is unsustainable and wastes resources and power.

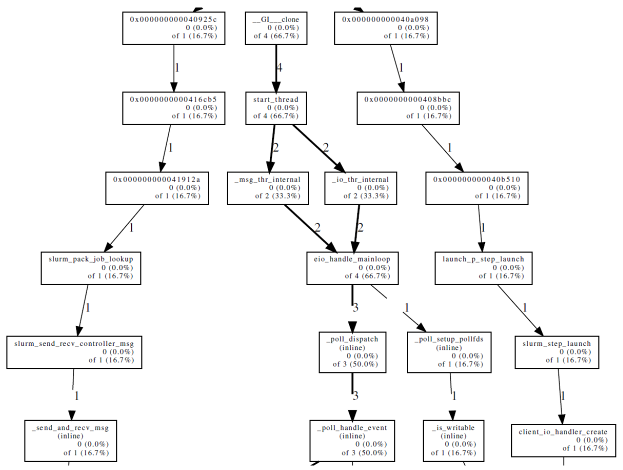

0x000000000040925c 0 (0.0%) of 1 (16.7%)
__GI___clone 0 (0.0%) of 4 (66.7%)
0x000000000040a098 0 (0.0%) of 1 (16.7%)
0x0000000000416cb5 0 (0.0%) of 1 (16.7%)
start_thread 0 (0.0%) of 4 (66.7%)
0x0000000000408bbc 0 (0.0%) of 1 (16.7%)
0x000000000041912a 0 (0.0%) of 1 (16.7%)
_msg_thr_internal 0 (0.0%) of 2 (33.3%)
_io_thr_internal 0 (0.0%) of 2 (33.3%)
0x000000000040b510 0 (0.0%) of 1 (16.7%)
slurm_pack_job_lookup 0 (0.0%) of 1 (16.7%)
eio_handle_mainloop 0 (0.0%) of 4 (66.7%)
launch_p_step_launch 0 (0.0%) of 1 (16.7%)
slurm_send_recv_controller_msg 0 (0.0%) of 1 (16.7%)
_poll_dispatch (inline) 0 (0.0%) of 3 (50.0%)
_poll_setup_pollfds (inline) 0 (0.0%) of 1 (16.7%)
slurm_step_launch 0 (0.0%) of 1 (16.7%)
_send_and_recv_msg (inline) 0 (0.0%) of 1 (16.7%)
_poll_handle_event (inline) 0 (0.0%) of 3 (50.0%)
_is_writable (inline) 0 (0.0%) of 1 (16.7%)
client_io_handler_create 0 (0.0%) of 1 (16.7%)

Figure 1: Bottleneck analysis on Google Performance Tools - visual call-graph output of a run at sub-par parallel efficiency

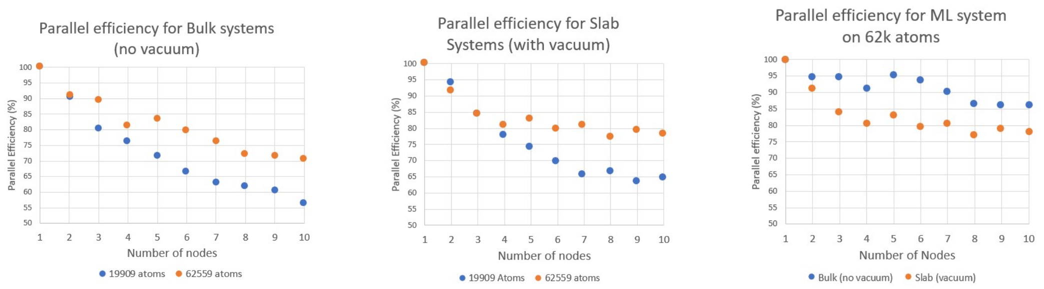
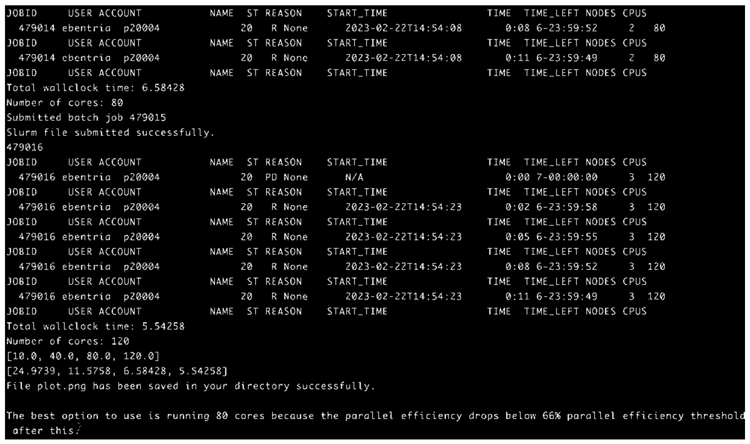


Figure 2: Impact of parallel efficiency on simulation runs using ReaxFF and ML-SNAP

4 Workflow of Software



```
JOBID      USER ACCOUNT            NAME  ST REASON    START_TIME                TIME  TIME_LEFT NODES CPUS
  479014 ebentria  p20004             20   R None       2023-02-22T14:54:08       0:08 6-23:59:52     2   80
JOBID      USER ACCOUNT            NAME  ST REASON    START_TIME                TIME  TIME_LEFT NODES CPUS
  479014 ebentria  p20004             20   R None       2023-02-22T14:54:08       0:11 6-23:59:49     2   80
JOBID      USER ACCOUNT            NAME  ST REASON    START_TIME                TIME  TIME_LEFT NODES CPUS
Total wallclock time: 6.58428
Number of cores: 80
Submitted batch job 479015
Slurm file submitted successfully.
479016
JOBID      USER ACCOUNT            NAME  ST REASON    START_TIME                TIME  TIME_LEFT NODES CPUS
  479016 ebentria  p20004             20  PD None       N/A                       0:00 7-00:00:00     3  120
JOBID      USER ACCOUNT            NAME  ST REASON    START_TIME                TIME  TIME_LEFT NODES CPUS
  479016 ebentria  p20004             20   R None       2023-02-22T14:54:23       0:02 6-23:59:58     3  120
JOBID      USER ACCOUNT            NAME  ST REASON    START_TIME                TIME  TIME_LEFT NODES CPUS
  479016 ebentria  p20004             20   R None       2023-02-22T14:54:23       0:05 6-23:59:55     3  120
JOBID      USER ACCOUNT            NAME  ST REASON    START_TIME                TIME  TIME_LEFT NODES CPUS
  479016 ebentria  p20004             20   R None       2023-02-22T14:54:23       0:08 6-23:59:52     3  120
JOBID      USER ACCOUNT            NAME  ST REASON    START_TIME                TIME  TIME_LEFT NODES CPUS
  479016 ebentria  p20004             20   R None       2023-02-22T14:54:23       0:11 6-23:59:49     3  120
JOBID      USER ACCOUNT            NAME  ST REASON    START_TIME                TIME  TIME_LEFT NODES CPUS
Total wallclock time: 5.54258
Number of cores: 120
[10.0, 40.0, 80.0, 120.0]
[24.9739, 11.5758, 6.58428, 5.54258]
File plot.png has been saved in your directory successfully.

The best option to use is running 80 cores because the parallel efficiency drops below 66% parallel efficiency threshold
 after this.
```

Figure 3: Implementation of Eco-Comp on HPCs

Typically, the functionality of Eco-Comp is split into three simple and intuitive steps: (1) installation of the Eco-Comp software, (2) running the Eco-Comp software and the analysis of the results. The following paragraphs will delve into each of the three steps in detail.

1. Installation All the user is required to fork from GitHub then install the program and execute it. The user will be prompted for various information required for future calculation purposes, such as the type of supercomputer in use, nodes and cores available, maximum wall-time, and the directory of the LAMMPS executable. Based on this information, a run.py script is automatically generated, which can be added to the bash script of the user.

2. Execution The user can then run the Python script within the directory within which their simulation data exists. This will prompt the user for the input file and data file names, then it calculates the system's complexity and categorizes it based on the aforementioned parameters. Depending on the complexity, the script automatically submits 4 runs to the job scheduler system using different number of cores, and collects CPU time once simulations are completed. The simulation is run for 100 MD steps such that the entire process finishes in a matter of seconds. Once simulated, the program reads the data from the output file, and calculates and prints the optimal configuration for the user to use. A plot is generated by Eco-Comp, visualizing the benchmarking times, and this data is also saved in a .JSON file for the user's future reference.

5 Conclusions

In conclusion, we have successfully built Eco-Comp, a user-friendly Python tool that optimizes the computing power of simulations for material scientists. By using just one command, one could considerably reduce the HPC carbon footprint within the material science community. This approach in the automatic bench-marking of software and their respective simulations is not limited to LAMMPS, and further efforts can be focused on other popular packages within the materials science community, such as the Vienna Ab initio Simulation Package (VASP). In the future, an expansion to the such as Graphical Processing Units (GPU) or hybrid CPU/GPU systems would be of interest.

6 Acknowledgment

This work is supported by the Qatar National Research Fund (QNRF) through the National Priorities Research Program (NPRP) under project number NPRP12S-0209-190063. The advanced computing facilities of Hamad Bin Khalifa University ("Hazeem Supercomputer") and "Raad2" in Texas A&M University at Qatar are used for all calculations.

References

[1] Armstrong, Brian & Kim, Seon & Eigenmann, Kim. (2000). *Quantifying Differences between OpenMP and MPI Using a Large-Scale Application Suite.* "Bondchk Failed with Reax/c." Materials Science Community Discourse, 7 Sept. 2016.

[2] Eichstädt, J. & Green, M. & Turner, M. & Peiró, J., & Moxey, D. (2018, April 5). *Accelerating high-order mesh optimisation with an architecture-independent programming model.* Computer Physics Communications.

[3] Google. *"Google/Pprof."* GitHub, https://github.com/google/pprof.

[4] He, Helen. *"Performance Engineering of Reactive Molecular Dynamics Simulations."* MIT Libraries, Massachusetts Institute of Technology, https://dspace.mit.edu/bitstream/handle/1721.1/139047/He-hmhe-meng-eecs-2021-thesis.pdf?sequence=1

[5] K. Cha, *"Performance Evaluation of LAMMPS on Multi-core Systems,"* 2013 IEEE 10th International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing, 2013, pp. 812-819, doi: 10.1109/HPCC.and.EUC.2013.117.

[6] LAMMPS Documentation. *LAMMPS Documentation* (23 Jun 2022 version) - LAMMPS documentation. (n.d.). https://docs.lammps.org/Manual.html

[7] LAMMPS molecular dynamics simulator. *LAMMPS Molecular Dynamics Simulator.* (n.d.).

[8] *"Main GPerfTools Repository."* GitHub, https://github.com/gperftools/gperftools.

[9] *"Parallel Performance."* Parallel Performance - Parallel Computing for Beginners, https://www.learnpdc.org/PDCBeginners/introduction/3.performance.html . [10] *"Performance Scaling."* Archer - Writing Scalable Parallel Applications with MPI.

[11] Yan, Beichuan & Regueiro, Richard. (2018). *Comparison between pure MPI and hybrid MPI-OpenMP parallelism for Discrete Element Method (DEM) of ellipsoidal and poly-ellipsoidal particles.* Computational Particle Mechanics. 6. 10.1007/s40571-018-0213-8.