Parallel Computing 77 (2018) 118–124

Contents lists available at ScienceDirect

Parallel Computing

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

# Student Cluster Competition 2017, team Nanyang Technological University: Reproducing vectorization of the Tersoff multi-body potential on the Intel Broadwell architecture

Siyuan Liu*, Meiru Hao, Bu-Sung Lee

*School of Computer Science and Engineering, Nanyang Technological University, Block N4, 50 Nanyang Avenue 639798, Singapore*

ARTICLE INFO

*Article history:*
Received 5 March 2018
Revised 23 May 2018
Accepted 25 June 2018
Available online 27 June 2018

*Keywords:*
Reproducibility
LAMMPS
Molecular dynamics
Tersoff potential
Vectorization

ABSTRACT

Reproducing previous work plays an important role in validating its effectiveness under different experiment settings. In this paper, we reproduce the results from the paper "The vectorization of the Tersoff multi-body potential: an exercise in performance portability" [1]. In particular, we reproduce the accuracy of the reduced precision solvers, the speed-up of the vectorized implementation, and the strong scaling test using Intel Xeon E5-2699 v4 Broadwell CPUs. We show that our reproduced results match the claims made in the original paper.



## 1. Introduction

Molecular dynamics (MD) is a computer simulation technique for simulating movement of particles. In order to model the interactions among particles, pairwise functions such as the Lennard–Jones and Coulomb potentials are used in many cases. However, in some cases multi-body potentials, which involve more than two particles, are necessary to accurately model the systems [2]. There are several popular open-source MD software, such as GROMACS [3], LAMMPS [4], and NAMD [5], that contain high-performance vectorized implementation of various potentials. However, they usually do not have optimization for multi-body potentials. Further more, the existing optimizations are implemented in various ways (e.g. assembly, intrinsics, and compiler directives), making it difficult to achieve performance portability on different architectures. The paper "The vectorization of the Tersoff multi-body potential: an exercise in performance portability" [1] (Tersoff paper) introduces a series of optimizations for the Tersoff multi-body potential [6] in the LAMMPS software. The optimizations include both scalar and vectorization optimizations. One important contribution of the Tersoff paper is that the performance improvement brought by the optimizations is portable to a variety of computer architectures, including ARM CPUs, Intel CPUs, Intel Xeon Phis, and NVIDIA GPUs. This portability is achieved by utilizing different vectorization schemes for different vector lengths and modularizing the vectorized implementation to ease the porting to multiple architectures. There are three claims from the Tersoff paper that we reproduce. First, the author claims that the single and mixed precision solvers only introduce small deviations to the simulation result compared to the double precision solver. In particular, the difference

* Corresponding author.
*E-mail addresses:* sliu019@e.ntu.edu.sg (S. Liu), haom0001@e.ntu.edu.sg (M. Hao), ebslee@ntu.edu.sg (B.-S. Lee).

https://doi.org/10.1016/j.parco.2018.06.006
0167-8191/

**Table 1**
Machine description.

| | |
|---|---|
| **Number of nodes** | **2** |
| **CPU** | Intel Xeon E5-2699 v4 x 2 (44 cores, no Hyper-Threading) |
| **GPU** | NVIDIA Tesla V100 x 8 |
| **Memory** | 256GB 2133MHz DDR4 RAM |
| **Interconnect** | Mellanox EDR InfiniBand |
| **Operating system** | CentOS 7.3 |
| **Vendor** | Supermicro |

**Table 2**
CPU models used in the Tersoff paper.

| Model | Architecture | Total cores |
|---|---|---|
| ARM Cortex-A15 | ARM | 8 |
| Intel Xeon X5675 | WM | 12 |
| Intel Xeon E5-2450 | SB | 16 |
| Intel Xeon E5-2680 v3 | HW | 24 |
| Intel Xeon E5-2697 v3 | HW | 28 |
| Intel Xeon E5-2697 v4 | BW | 36 |

should be small (below 1%) and should oscillate. Second, the performance improvement should be at least 2x on any architecture when compared to the reference implementation. Third, the performance improvement scales to clusters in strong scalability tests. We say the Tersoff paper is reproducible on the Intel Broadwell architecture if we could obtain experimental results that match the above-mentioned claims on our own cluster, which differs in both hardware and software from the original experiment environment.

## 2. Environment setup

In this Section, we describe the hardware and software setup used for this reproducibility work. All work was performed during the SC17 Student Cluster Competition. In the competition, all teams were asked to reproduce the results obtained in the Tersoff paper using their competition clusters. The only additional constraint during the competition is a power cap of 3000 Watts. However, the power cap did not affect our cluster's performance during the reproducibility experiments in any ways.

### *2.1. Machine configuration*

The specification of the machines used for this reproducibility work is presented in Table 1. The CPU models used in the Tersoff paper that represent different CPU architectures is presented in Table 2. Other information, such as the memory and the interconnect used, is not provided in the Tersoff paper.

### *2.2. Software environment*

We use version 2018 of the Intel Parallel Studio XE Cluster Edition which contains both the Intel C++ compiler and Intel MPI. The tersoff paper also used Intel compiler and Intel MPI.

### *2.3. Source code*

The source code used in this reproducibility work is obtained from the Tersoff repository.[1] The version of LAMMPS is 10Mar16. An additional patch is added on top of that to fix a bug in the NVE integrator of the USER–INTEL package,[2] which makes both single and double precision code better at energy conservation.

## 3. Experiments

Three types of experiments are carried out for this reproducibility work. Specifically, the accuracy experiments, the performance experiments, and the scalability experiments are conducted. This section describes our results and analysis on these experiments.

[1] https://github.com/HPAC/lammps-tersoff-vector.
[2] https://github.com/HPAC/lammps-tersoff-vector/commit/092e33b.

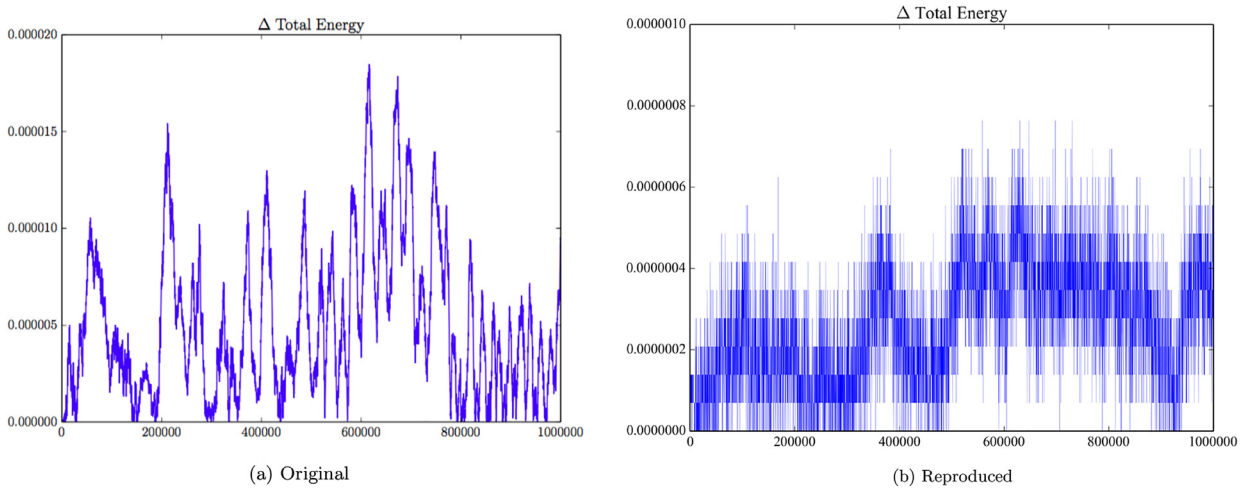


**Fig. 1.** Relative difference between the single and double precision solvers.

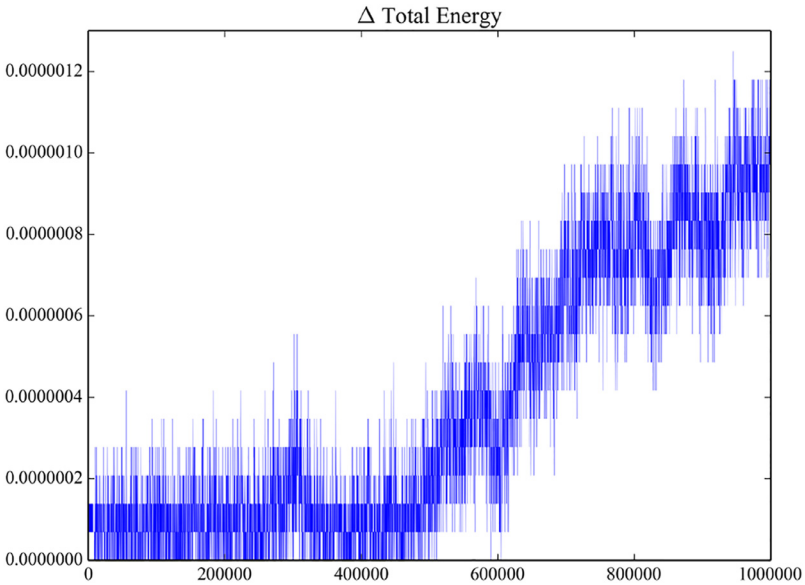


**Fig. 2.** Relative difference between the mixed and double precision solvers.

### *3.1. Accuracy experiments*

In the Tersoff paper, single precision and mixed precision solvers are implemented alongside the double precision solver. To validate the accuracy of reduced precision solvers, a long simulation in a NVE ensemble, whose total energy ($E$) does not change, is performed. The total energy produced by single and mixed precision solvers are compared against that of the reference double precision solver, and the relative difference is computed as $\frac{|ref-trial|}{|ref|}$. The accuracy experiment result from the Tersoff paper, which was performed on Intel Xeon Phi, is presented in Fig. 1a. The input file used in the accuracy experiment runs for 1,000,000 timesteps. In our attempt to reproduce the result, we compare both single and mixed precision solvers to the double precision solver. The total energy difference between single and double precision solvers is presented in Fig. 1b and the difference between mixed and double precision solvers is presented in Fig. 1b. We also present in Fig. 3 the difference between single and double precision solvers without the bugfix mentioned in Section 2.3. Figs. 1b and 2 show that the difference in total energy between reduced precision solvers and double precision solver is below 1 percent, which corresponds to the claim made in the Tersoff paper. In fact, the difference is around 0.0001% in both cases. By comparing Figs. 1b and 3, we can see that the bugfix in the NVE integrator significantly improved the accuracy of the solvers. The difference in Fig. 1b is three orders of magnitude smaller than the difference in Fig. 3. However, only the dif-

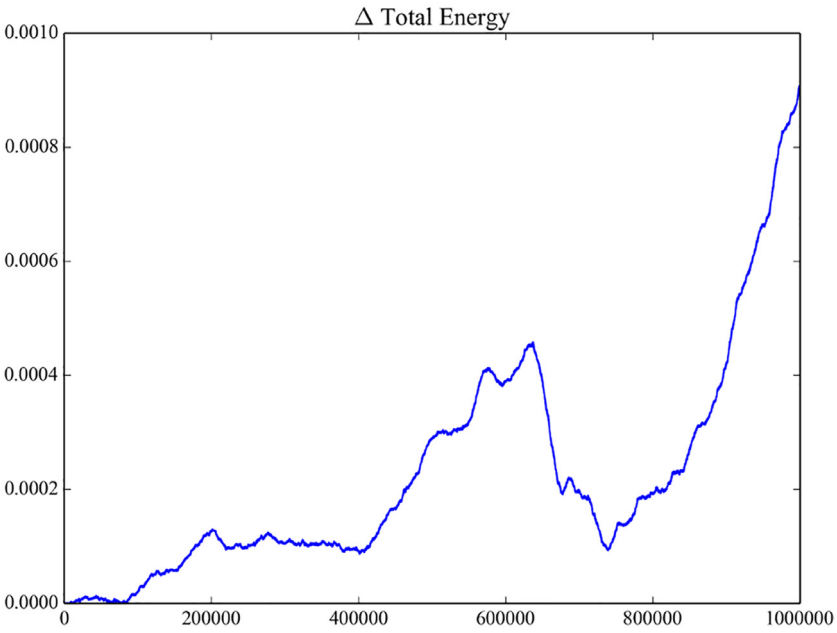


**Fig. 3.** Relative difference between the single and double precision solvers without bugfix.

ference in Fig. 1b oscillates while the difference in Fig. 2 goes up towards the end of the simulation. We are not able to identify the root cause of such behavior, and the exact reason requires further investigation.

### 3.2. Performance experiments

To demonstrate the effectiveness of the vectorized implementation of Tersoff potential, performance experiments are conducted for both single-threaded and single node runs. Single-threaded runs represent the best speed-up brought by the vectorization optimizations because there is no communication overhead involved. Single node runs provide a more realistic speed-up given the fact that most of the real world runs are in parallel. For single-threaded runs, the input file is `in.tersoff`. For single node runs, both `in.tersoff_bench` and `in.porter` are used as the input files. The `in.tersoff` and `in.tersoff_bench` input files were used in the Tersoff paper, while the `in.porter` input file was used in [7].

For single-threaded experiments, the running time of four different implementations are recorded. They are the reference version without optimization and the optimized version with double, mixed, and single precisions. The results of single-threaded experiments from the original paper are presented in Fig. 4a and the results of our experiments are presented in Fig. 4b. From the comparison between Fig. 4a and b, we can see clearly that our speed-up of the three optimized implementations match the results in the Tersoff paper. The speed-up of Opt-D, Opt-S and Opt-M over Ref are 3.46x, 4.92x, and 4.87x respectively.

For single-node experiments, the running time is only recorded for two implementations. They are the reference version without optimization and the optimized version with mixed precision. Only the mixed precision version is used among the optimized versions because the mixed precision version is most likely to be used in real world runs. This follows the experiments run in the Tersoff paper. The results of the single node experiments from the original paper are presented in Fig. 5a and the results of our experiments are presented in Fig. 5b for `in.tersoff_bench` and Fig. 6 for `in.porter`. The speed-up of Opt-M over Ref is 4.55x for `in.tersoff_bench`, which matches the result from the Tersoff paper. The speed-up is less than the speed up of single-threaded runs which is expected as the single node runs incur communication cost between processes. The speed-up for `in.porter` is only 1.34x. This is due to the fact that `in.porter` only has 216 atoms, which is too small for the optimizations to become effective.

### 3.3. Strong scaling experiments

To show that the performance improvement of the vectorized implementation scales to clusters, a strong scaling test is performed. In the Tersoff paper, the test was performed with the reference version on Intel Ivy Bridge CPUs, the optimized double precision version on Intel Ivy Bridge CPUs and Intel Ivy Bridge CPUs with Intel Xeon Phis. In our work, we reproduce it with reference and optimized double precision version on Intel Broadwell CPUs. The input files for the strong scaling test are `in.tersoff_bench` and `in.porter`. As our cluster has only two nodes with 88 CPU cores, we conducted the strong

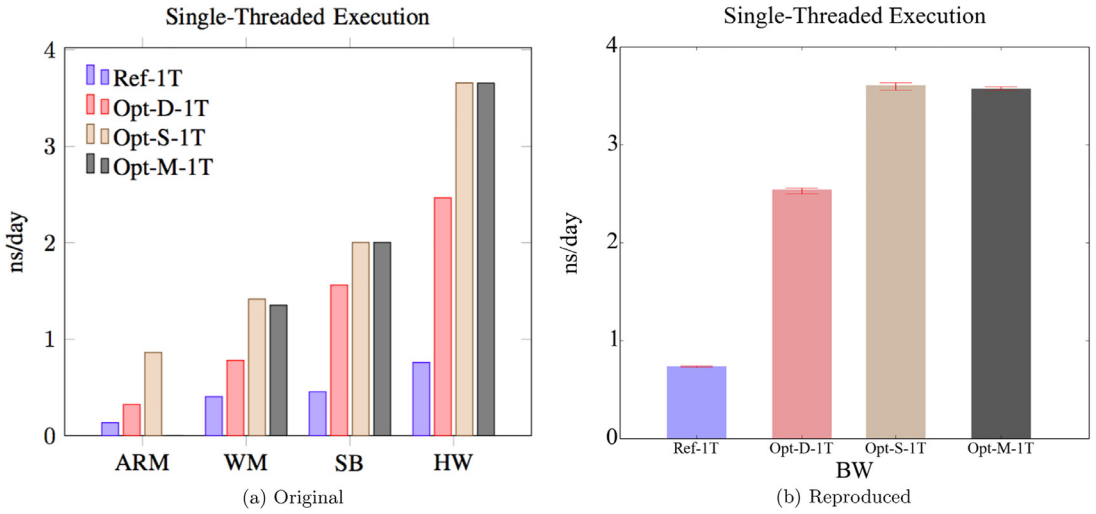


**Fig. 4.** Single-threaded performance on in.tersoff.

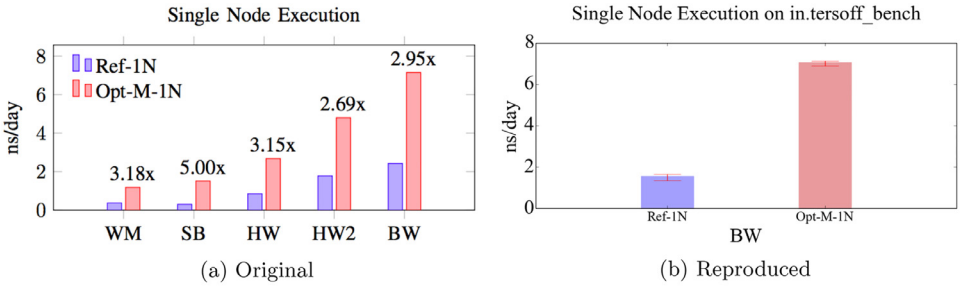


**Fig. 5.** Single node performance on in.tersoff_bench.

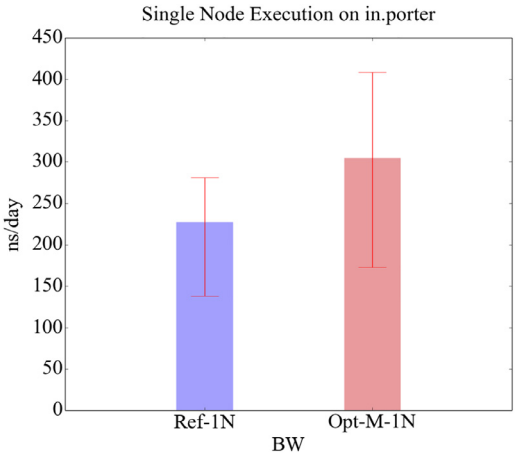


**Fig. 6.** Single node performance comparison on in.porter.

scaling test with 11 cores, 22 cores, 44 cores, and 88 cores. The strong scaling results from the Tersoff paper are presented in Fig. 7a. The results from our work are presented in Fig. 7b for `in.tersoff_bench` and in Fig. 8 for `in.porter`.

From Fig. 7a and b, we can see that the scalability exhibited in the Tersoff paper's test is reproduced on our machines. Our results show better scalability because only the run with 88 cores incurs inter-node communication and runs with other core count are within one node.

As for the results in Fig. 8, it shows very bad scalability for the `in.porter` input file. This is again caused by the fact that there are only 216 atoms in this simulation compared to the 32,000 atoms in the `in.tersoff_bench` input file. Both

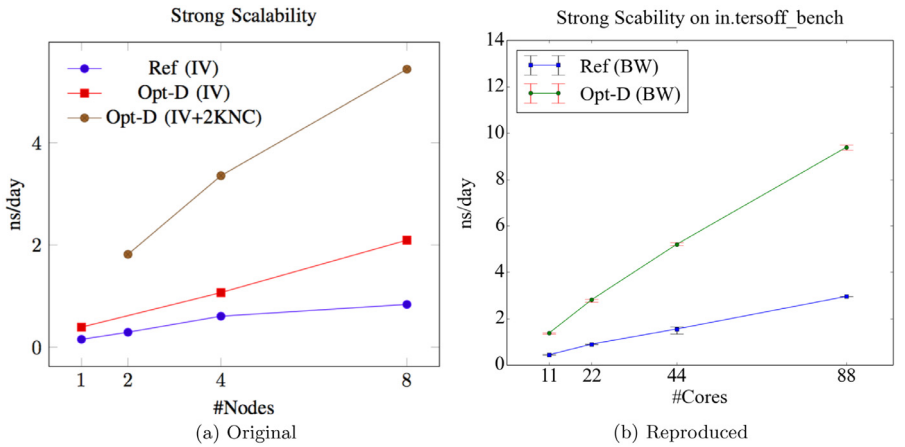


**Fig. 7.** Strong scaling experiment on in.tersoff_bench.

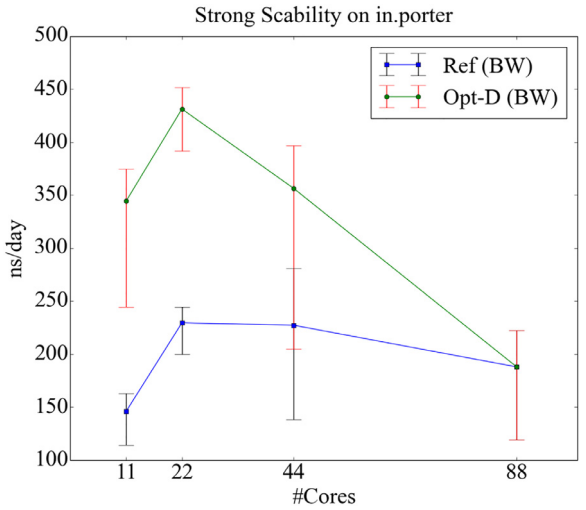


**Fig. 8.** Strong scaling experiment on in.porter.

the reference version and optimized version demonstrate bad scalability. Therefore, we can state that this is not caused by the optimizations.

## 4. Conclusion

In this paper, we first established the context of both the Tersoff paper and our work. We then described the experiment setup used in our work and in the Tersoff paper. Lastly, we presented our results on accuracy, performance, and strong scaling experiments and compared those with the results from the Tersoff paper. Our comparison shows that the vectorization optimizations proposed in the Tersoff paper provide significant speed-up (more than 2x) over the reference implementation and scale well to a cluster of machines. The reduced precision solvers only introduce small deviation from the double precision solver, making them usable in production runs. However, the increase of total energy difference towards the end of the simulation when using mixed precision solver needs further investigation in future work. We note that the bugfix in the NVE integrator, which is already incorporated into the Tersoff paper's software repository, should be taken into consideration when reproducing the Tersoff paper or related work in the future since it brings significant improvement to computation accuracy as demonstrated in our work. In conclusion, we are able to reproduce the claims made in the Tersoff paper on our cluster with the Intel Broadwell CPU architecture.

## Acknowledgements

The authors would like to thank National Supercomputing Centre (Singapore), JOS, NVIDIA Corporation, and Mellanox Technologies for providing the hardware resources used in this reproducibility work.

## Appendix A. Compiling and running the code

In the source code provided in the Tersoff paper, an example compilation script for Intel CPU build is included. This build script installs the USER–INTEL and USER–OMP packages, and makes sure the KOKKOS and GPU packages are not installed. We have slightly modified the script to adapt to our software installations.

The Makefile used in the script, `Makefile.intel_cpu`, is modified for compatibility with version 2018 of the Intel Parallel Studio XE Cluster Edition. Specifically, the flags `-openmp`, `-no-offload`, and `-override-limits` are changed to `-qopenmp`, `-qno-offload`, and `-qoverride-limits` respectively as the old flags are deprecated.[3]

With the above modifications in place, the code is compiled on our machine with the command `$./build.sh`.

In the source code, scripts to run the experiments are also provided. The `accuracy.sh` script is used for the accuracy experiments, and the `bench-cpu2.sh` script is used for the single-threaded and single node performance experiments. For the strong scaling experiments, we prepared our own scripts. In all these scripts, the line to actually run LAMMPS has the following format,

```
$ mpirun -np <npn> -ppn <ppn> \
  -host <list-of-hosts> \
 lmp_intel_cpu_default_vector \
 -in <input> -v p vanilla \
 -sf intel -pk intel 0 mode <mode> \
```

When running the reference implementation, the last line in the above command is omitted so that the optimizations are disabled.[4]

## References

[1] M. Höhnerbach, A.E. Ismail, The vectorization of the tersoff multi-body potential: an exercise in performance portability, in: High Performance Computing, Networking, Storage and Analysis, SC16: International Conference for, IEEE, 2016, pp. 69–81.
[2] S.J. Plimpton, A.P. Thompson, Computational aspects of many-body potentials, MRS Bull. 37 (5) (2012) 513–521.
[3] H. Berendsen, D. van der Spoel, R. van Drunen, Gromacs: a message-passing parallel molecular dynamics implementation, Comput. Phys. Commun. 91 (1) (1995) 43–56.
[4] S. Plimpton, Fast parallel algorithms for short-range molecular dynamics, J. Comput. Phys. 117 (1) (1995) 1–19.
[5] J. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. Skeel, L. Kalé, K. Schulten, Scalable molecular dynamics with namd, J. Comput. Chem. 26 (16) (2005) 1781–1802.
[6] J. Tersoff, New empirical approach for the structure and energy of covalent systems, Phys. Rev. B 37 (1988) 6991–7000.
[7] L.J. Porter, S. Yip, M. Yamaguchi, H. Kaburaki, M. Tang, Empirical bond-order potential description of thermodynamic properties of crystalline silicon, J. Appl. Phys. 81 (1) (1997) 96–106.

[3] https://software.intel.com/en-us/cpp-compiler-18.0-developer-guide-and-reference-deprecated-and-removed-compiler-options.
[4] http://lammps.sandia.gov/doc/accelerate_intel.html.