Parallel Computing 78 (2018) 28–32

Contents lists available at ScienceDirect

Parallel Computing

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

Student Cluster Competition 2017, Team Peking University: Reproducing vectorization of the Tersoff multi-body potential on the Intel Broadwell architecture

Zhenxin Fu, Lei Yang, Wenbin Hou, Zhuohan Li, Yifan Wu, Yihua Cheng, Xiaolin Wang, Yun Liang*

Peking University, 5 Yiheyuan Road, Haidian, Beijing 100871 China

ARTICLE INFO

Article history:
Received 10 March 2018
Revised 24 May 2018
Accepted 25 June 2018
Available online 27 July 2018

Keywords:
Multi-body potential
Reproducibility

ABSTRACT

Höhnerbach et al. [1] proposed a general optimization method for the multi-body potential problem by exploiting vectorization on various of architectures using LAMMPS. As a task of the Student Cluster Competition in The International Conference for High Performance Computing, Networking, Storage and Analysis (SC17), we tried to reproduce the optimizations proposed in the paper. The molecular dynamics simulation of crystalline silicon is performed with LAMMPS on both our own single-node Intel Xeon machine and the cloud resource. The accuracy of the computation results is examined. The performance of the optimized program is compared with the original version to study the effectiveness and the scalability of the proposed optimization.

1. Introduction

The Tersoff multi-body potential is a model used to compute intermolecular potential and to perform molecular dynamics (MD) simulations. The original paper [1] applied vectorization to the computation of multi-body potential, using an approach which is effective on a wide range of computing platforms, achieving a considerable acceleration that is portable among different hardwares. In particular, four optimization methods were proposed in the original paper:

- Pre-calculating derivatives, in order to avoid redundant multiplications.
- Choosing different vectorizations depending on the input data set.
- Avoiding masking or divergence during computation.
- Filtering the neighbor list by setting a maximum radius.

In the original paper, a set of experiments were conducted to demonstrate the effectiveness and the scalability of the proposed optimization. During the competition, we performed three of them, as listed below:

Accuracy study: we examined the difference between the single to double solver and the mixed to double solver to reproduce the correctness of the optimized program. The results are described in Section 4.

* Corresponding author.
E-mail addresses: fuzhenxin@pku.edu.cn (Z. Fu), yangl1996@pku.edu.cn (L. Yang), houwenbin@pku.edu.cn (W. Hou), zhuohan123@pku.edu.cn (Z. Li), evanwu@pku.edu.cn (Y. Wu), chengyihua@pku.edu.cn (Y. Cheng), wxl@pku.edu.cn (X. Wang), ericlyun@pku.edu.cn (Y. Liang).

https://doi.org/10.1016/j.parco.2018.06.010
0167-8191/

Table 1
Configuration details of our machine (SuperMicro SYS-4028GR-TR2) and the cloud server (CycleCloud).

Name	SuperMicro SYS-4028GR-TR2 (single node, no network components)		CycleCloud (Azure H-series)
Part	Model	Vendor	Model
CPU	2 × Intel Xeon E5-2690 v4 (14 Cores, 2.6GHz)	Intel	Intel Xeon E5-2667 v3 (3.20GHz)
GPU	10 × NVIDIA Tesla V100	NVIDIA	N/A
Memory	512GB DDR4	N/A	110GB
OS	Ubuntu 16.04	N/A	CentOS Linux 7.1

Performance comparison: we compared the performance of the optimized program and the original version to reproduce the effectiveness of the proposed optimization methods. In particular, we processed the input file *in.tersoff* to reproduce Fig. 4 in the original paper, and processed file *in.tersoff_bench* and *in.porter* for Fig. 5 in the original paper. The results are shown in Section 5.

Scalability study: we compared the performance of the optimized program when running on different number of CPU cores to reproduce the strong scalability of the optimizations claimed in the original paper. In particular, we processed input file *in.tersoff_bench* and *in.porter* on the CycleCloud cloud instance and reproduced Fig. 9 in the original paper. The results are shown in Section 6.

2. Machine description

As stated above, we used our server to perform the accuracy study and the performance comparison, and used the CycleCloud cloud instance to perform the scalability study. The hardware and software configurations of both machines are shown in Table 1 (our server and CycleCloud cloud instance). It should be noticed that all three experiments were performed in single-node settings.

3. Compilation/run description

3.1. Compilation description

The following steps were taken to compile the program.

- Install Intel compilers and MPI library. Version 2017 was used.
- Install other dependencies, including *libjpeg*.
- Execute scripts *build.sh* located in *machines/machine_type*. These scripts automatically compile the source code into corresponding architecture.

For more convenient installation, we applied Network File System (NSF) to share files among cloud nodes. We also wrote a script to install some necessary tools like vim, git and g++ when initializing a new cloud instance.

3.2. Run description

Firstly, we modified the running script to fit the tasks. The modification was for processing new data and running specific modes. The modes indicate different binary with a different configuration, for example, the configuration contains the number of threads. Then, the script accomplished all the tasks one by one. And, the results were stored in a specific folder with well-recognized name. Finally, plotting code was executed to get figures for visualization and analysis.

3.3. Data sets

As required by the competition, different data sets were used to perform different experiments, as shown below.

in.tersoff-acc, with 32,000 atoms for 10^6 timesteps, was used for the accuracy study.

in.tersoff, with 32,000 atoms for 100 timesteps, was used to evaluate the effectiveness of the optimization in single-thread execution.

in.tersoff_bench, with 512,000 atoms for 2420 timesteps, was used to evaluate the effectiveness of the optimization in single-node execution, and to study the scalability. It was run for five times.

in.porter, with 216 atoms for 10,000 timesteps, was used to evaluate the effectiveness of the optimization in single-node execution, and to study the scalability. This data set is from another paper [2].

3.4. Software version

The following softwares are used in the reproducibility task:

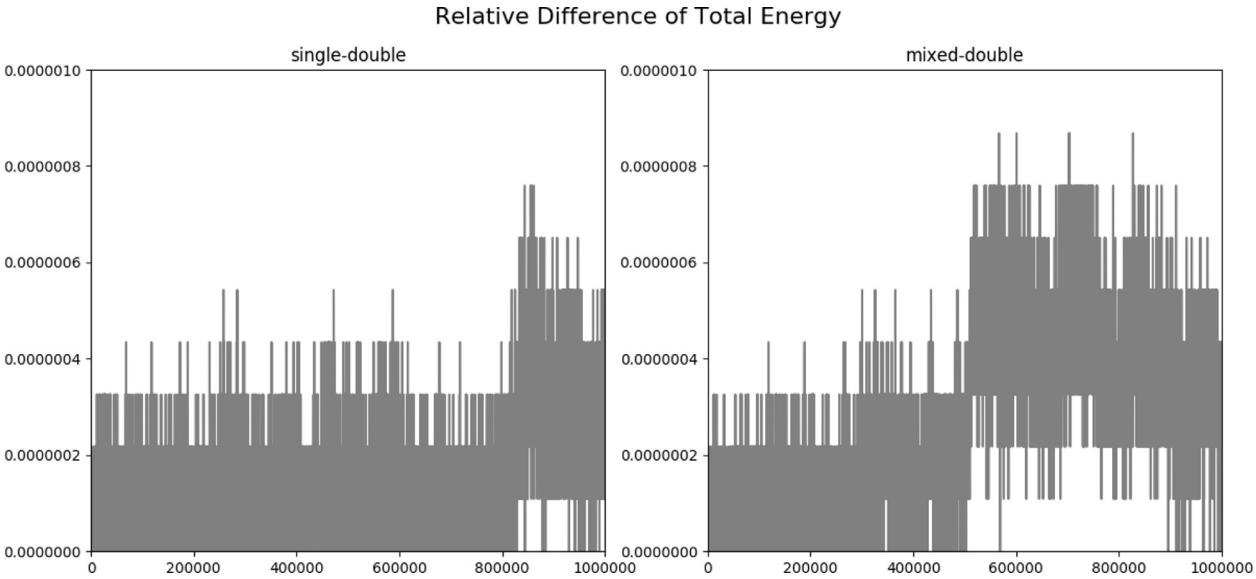

Fig. 1. Relative difference between the single-double precision solvers (left) and mixed-double precision solvers (right) for a system of 32,000 atoms for 10^6 timesteps.

ICC[1] (Intel C++ Compilers 2017) is a group of C and C++ compilers from Intel. It contains MPI compiler for parallel computing which we used in this report.

Lammps[2] (lammps-10Mar16, from Tersoff repository) is a classical molecular dynamics code, we used it to model atoms.

Matplotlib[3] (Version 2.1.2) is a python plotting library which makes plotting easy and our data visualization depends on Matplotlib.

4. Accuracy study

In the original paper, the relative difference between the single and double precision solvers for a system of 32 000 atoms for 10^6 timesteps is claimed to be small enough. The deviation is within 0.002% of the default double precision mode.

We tried to reproduce the relative difference in both the single and the mixed precision mode with 32,000 silicon atoms in a long simulation of 10^6 timesteps. See Fig. 1 for our results with input file *in.tersoff-acc*. The relative difference lies within 0.00010%, which supports the claim in the paper that optimization doesn't introduce big loss in accuracy.

5. Performance differences

In the original paper, different execution modes were tested with a single thread on multiple platforms. On all architectures tested, the optimized version all demonstrated better performance than the reference version, despite the fact that the vector lengths are different on those platforms. The effectiveness does change on different architectures with different vector lengths. Figs. 4 and Fig. 5 in the original paper demonstrates the portability of the performance across CPUs, with both single-threaded and single-node execution.

To reproduce the claim, we performed the simulation on our server with Intel Xeon E5-2690 v4.

- With input file *in.tersoff*, we reproduced Fig. 4 in the original paper for Ref-1T (the unoptimized reference version shipped with LAMMPS itself), Opt-D-1T (the optimized version with double precision), Opt-S-1T (the optimized version with single precision) and Opt-M-1T (the optimized version with the mixed precision of both single and double). Results are shown in Fig. 2. The results match the effectiveness of optimization claimed in the paper. The performance of Opt-S-1T is almost the same with Opt-M-1T, and the overall tendency is Ref-1T $<$ Opt-D-1T $<$ Opt-S-1T $\approx$ Opt-M-1T. The improvement of Opt-S-1T or Opt-M-1T contributes to lower accuracy functions comparing with Opt-D-1T.
- We used the input file *in.tersoff_bench* and *in.porter* to reproduce Fi. 5 in the original paper. For both files, we run the program in Ref-1N and Opt-M-1N mode. Results are shown in Fig. 3. For tersoff_bench, the Opt-M-1T is 3.8 times faster than Ref-1N, which are in the same range with original paper (2.69 times to 5 times).

[1] https://software.intel.com/en-us/c-compilers.
[2] http://lammps.sandia.gov/.
[3] https://matplotlib.org/index.html.

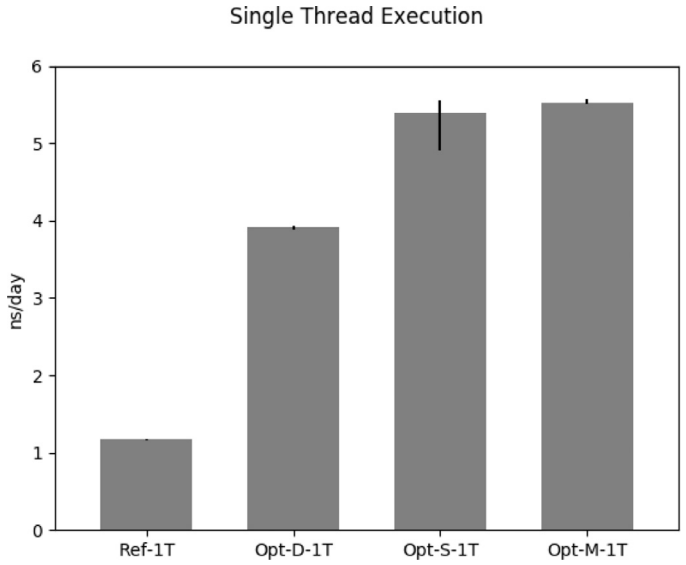

Fig. 2. Evaluation of performance improvement in single-thread execution on our server. The black lines on top center of the bar in the graph represent the maximum and minimum ns/day value for each case.

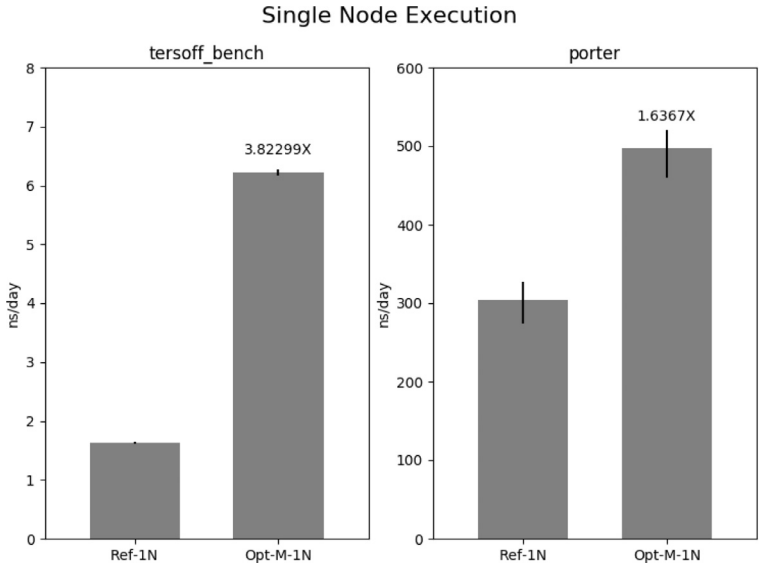

Fig. 3. Evaluation of performance improvement in single-node execution on our server, with input file *in.tersoff_bench* and *in.porter*. The black lines on top center of the bar in the graph represent the maximum and minimum ns/day value for each case.

File *in.porter* only contains 216 atoms, while *in.tersoff_bench* has 512,000 atoms. The smaller scale of simulation for *in.porter* leads to its higher ns/day number. This explains their difference in performance on y-axis (ns/day), and in speed magnification.

6. Strong scaling study

The paper shows scaling results for 2 million atoms running on up to eight nodes. The effect of improvements in single-thread version is supposed to be combined with the performance gain achieved from parallel execution. However in practice, much of those performance gain is lost to various overheads.

We reproduced this claim on cloud instance with input file *in.porter* and *in.tersoff_bench*. The results are shown in Fig. 4, which is almost the same as the original paper, except that the loss of performance for *in.porter* from 8 cores to 16 cores is larger than expected (slight performance improvement from 8 cores to 16 cores). It may be from the relatively smaller

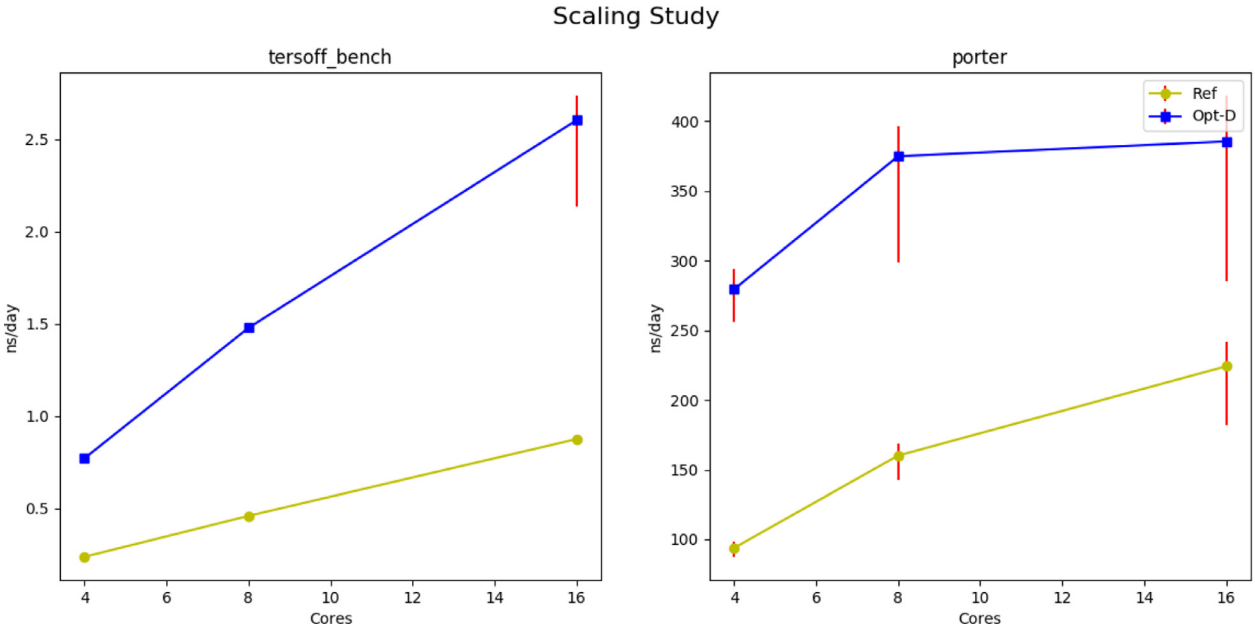

Fig. 4. Optimization results on CycleCloud cloud instances of tersoff_bench (left, 512,000 atoms) and porter (right, 216 atoms). The red lines on top center of the bar in the graph represent the maximum and minimum ns/day value for each case. (For interpretation of the references to color in the text, the reader is referred to the web version of this article.)

size of the input file (216 atoms of porter comparing to 512,000 atoms of tersoff_bench), which leads to needing more and smaller communication between cores.

7. Conclusion

As shown above, although differences exist between our results and the original paper, our results correlate to the conclusion of the paper, of which the reason has been described in above sections. The results of input file *in.porter* differ slightly from the original paper, which may be caused by its relatively small scale of only 216 atoms. As the paper demonstrated, the optimizations work well on a wide variety of architectures, and shows good performance gain on our platform as an example. By replicating the accuracy study, comparing performance differences and reproducing strong scaling study, we conclude that we have reproduced the results from the paper.

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] 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.