Parallel Computing 78 (2018) 72–78

Contents lists available at ScienceDirect

Parallel Computing

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

Student cluster competition 2017, team NTHU: Reproducing vectorization of the tersoff multi-body potential on the Intel Skylake and Nvidia P100 architecture

ChanJung Chang[a], YungChing Lin[a], YuHsuan Cheng[a], YuCheng Wang[a], LiYu Yu[a], TienChi Yang[a], Jerry Chou[a,*]

[a] *Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan, ROC*

ARTICLE INFO

Article history:
Received 8 March 2018
Revised 22 May 2018
Accepted 25 June 2018
Available online 19 July 2018

ABSTRACT

Markus Höhnerbach et al. recently published a work to optimize the performance of Tersoff potential, which is a computing scheme used in the LAMMPS molecular dynamics (MD) code. The optimization solver was implemented with three different computation precisions, namely single, mixed, and double. As a special activity of the Student Cluster Competition at SC17 conference, we aimed to reproduce their experimental studies of the optimization solvers in terms of accuracy, performance and scalability. We conducted our experiments on a cluster with Intel Xeon Gold 6130 CPUs and Nvidia Tesla P100 GPUs, while the original work was on a cluster with Intel Xeon E5-2650 CPUs and K40 GPUs. Desite of the differences in computing systems, we demonstrate the claims from the original work can be successfully reproduced by showing the solvers with less precision implementation can still achieve high accuracy results, and exhibit good performance speedup and scalability.

1. Introduction

A state-of-the-art approach for molecular dynamics simulator was recently published by Markus Höhnerbach et al [1]. They developed an extension of the LAMMPS molecular dynamics simulator with a new, optimized and portable vectorization scheme implementation of the Tersoff multi-body potential. To deliver cross-platform performance, their approach abstracted target architecture and computing precision, and their experiments showed between 3 times and 5 times speedup over a pure MPI reference on an Intel Xeon Phi cluster. As a special activity of the Student Cluster Competition [8] at SC17 conference, we were requested to verify the reproducibility of their work on our own GPU cluster. Hence, we reported our reproducibility study results in this paper, and we referred their work as the *Tersoff paper* for the rest paper.

Several efforts have been made to speed up the simulation of pair potentials [9] and multi-body potentials [10–12]. But all these previous approaches are only suitable for rigid problems. Hence the Tersoff paper [1] proposed an approach for general scenarios, using the following optimization techniques.

- Scalar optimizations improved parameter lookup, eliminated redundant calculations.
- Vectorization parallelized execution.

* Corresponding author.
E-mail address: jchou@lsalab.cs.nthu.edu.tw (J. Chou).

https://doi.org/10.1016/j.parco.2018.06.008
0167-8191/© 2018 Elsevier B.V. All rights reserved.

Table 1
Hardware specifications of the GPU cluster node sponsored by QCT (Quanta Cloud Technology [2]).

Component	Specification	Vendor	Quantity
CPU	Xeon Gold 6130 2.1 GHz 16 Cores	Intel	2
Accelerator	Tesla P100	Nvidia	4
Memory	384 GB DDR4 2400 MHz	Samsung	–
Disk	800GB SSD	Intel	–
Operating system	CentOS 7.4	–	–

- Optimizations that aided vectorization, avoided calculations that did not contribute to the final result.

In the Tersoff paper [1], the authors implemented these techniques in an optimization solver with three different computation precisions, namely *single, mixed*, and *double*. Experiments were conducted to compare the optimization solver with different precisions in the following three aspects. First, they examined how much accuracy was sacrificed from choosing single or mixed precision solver instead of double precision solver. Second, they evaluated how much performance could gain from choosing a less precise solver instead of the standard double precision. Last, they demonstrated the strong scalability of their implementations under increasing number of cores.

According to the Tersoff paper [1], they made the following claims.

1. Accuracy study: relative deviation should stay less than 1 percent.
2. Performance study: optimized solver should have greater performance than the reference solver, and mixed version should offer a compromised version between speed and accuracy.
3. Scalability study: a 6 times performance speedup should be obtained when using 8 times computing cores.

Hence, the goal of this paper is to reproduce their experimental approach on our own cluster, and verify the sustainability of all claims above.

Overall, our study showed the Tersoff optimizations proposed by Markus Höhnerbach et al. could be successfully reproduced on our GPU cluster by getting similar experimental results in their original paper. But we also noticed two differences from our experiments. One is that the performance measured from our cluster was greater than the numbers reported by the Tersoff paper by comparing the performance of original LAMMPS reference solver. This could be expected because the hardware specification of our cluster had greater max turbo frequency than the one used in the original paper. The other difference is that the scalability becomes worse and less predictable because we were using a smaller dataset.

The rest of paper is organized as follows. Section 2 describes the machine specification of our experiments. Section 3 details building and running the code. The experimental results of accuracy, performance and scalability are discussed in Section 4, 5, and 6, respectively. Finally, Section 7 concludes this paper.

2. Machine description

Our experiments were conducted on a single GPU cluster node equipped with 2 Xeon Gold 6130 CPUs, 4 Nvidia Tesla P100 cards and 384 GB memory as detailed in Table 1. Hence the computing capacity of the machines in our study is much higher than the machines used in the original paper. As a result, the performance numbers reported from this work are higher than the original paper. But noted our reproducibility claims only depend on the trend of relative performance rather than the absolute performance values.

3. Compilation & run

Because the flags of compilation and execution could affect application performance, this section provides our step-by-step commands to compile and run the LAMMPS molecular dynamics (MD) simulator and Tersoff code on our cluster. Noted, these instructions might need to be adjusted slightly in a different environment:

Step1 Clone the application from github.
```
$ git clone https://github.com/HPAC/lammps-tersoff-vector.git
```
Step2 Download the driver and change configuration for Nvidia P100 in build.sh.
```
$ cd machines/rwth-sb_tesla
$ sed '1 iexport CUDA_ROOT=/usr/local/cuda-8.0′ build.sh -i
$ sed 's/Kepler35/Maxwell53/' build.sh -i
```
Step3 Run build.sh, which creates directory lammps-10Mar16.
```
$ ./build.sh
```
Step4 Change sm_35 to sm_60 in /lib/kokkos/config/nvcc_wrapper.
```
$ cd lammps-10Mar16; sed 's/sm_35/sm_60/' /ibkokkos/config/nvcc_wrapper -i
```
Step5 Copy the missing power 8 header file.
```
$ cp ./src/USER-INTEL/intel_intrinsics_power8.h ./src/intel_intrinsics_power8.h
```

Table 2
Input for experiments shown in this paper.

Experiment	Figure	Input file	Number of atoms
Accuracy study	1, 2	*in.tersoff-acc* [4]	32,000
Single-threaded performance	3	*in.tersoff* [5]	32,000
Single-node performance	4	*in.tersoff_bench* [6]	512,000
	5	*in.porter* [7]	216
GPU performance	6	*in.tersoff_bench* [6]	512,000
Strong scalability	7	*in.tersoff_bench* [6]	512,000
	8	*in.porter* [7]	216

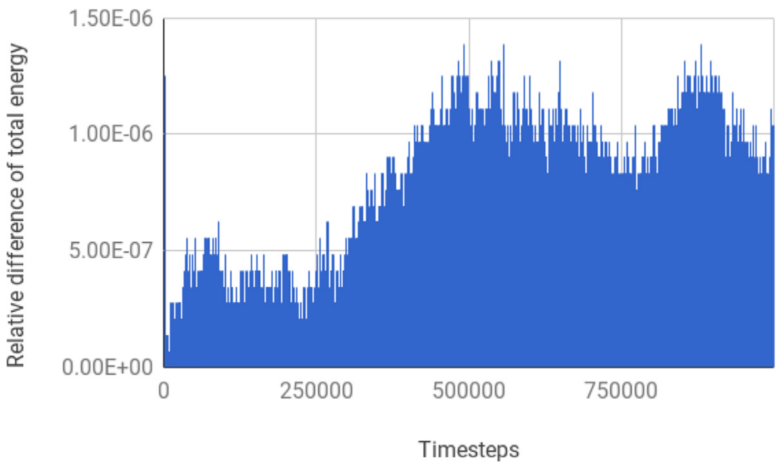

Fig. 1. Validation of the single precision solver. Relative difference of total energy between single and double solver.

Step6
- Remove the undefined enum "CU_COMPUTEMODE_EXECLUSIVE", and remove the if condition "_properties[i].computeMode = = CU_COMPUTEMODE_EXCLUSIVE".

Step7 Specify the path for CUDA library in ./lib/gpu/Makefile.lammps.standard
- set gpu_SYSPATH = /usr/local/cuda-8.0/lib64

Step8 Specify SM architecture for GPU, and pass –fPIC flag. in ./lib/gpu/Makefile.linux.
- set CUDA_ARCH = -arch = sm_60; add -Xcompiler -fPIC to CUDA_OPTS; add -fPIC to CUDPP_OPT

Step9 Build all binaries including CPU and GPU version

$ cd .. && ./build.sh

$ We also compiled different vectorized CPU versions, including AVX, AVX2, AVX512. For instance, we added the value of “-D__AVX512F__” to CCFLAGS in build.sh to build the AVX512 version.
- To run the CPU reference version, "mpirun -np CORE -machinefile MA.CHINEFILE PATH_TO_BIN/BINARY -in INPUT -v p vanilla".
- To run the optimized version, add the flags "-sf intel -pk intel 0 mode PRECISION" to the running command.
- To run GPU version, add the flags "-sf gpu" to the running command
- To run KOKKOS version, add the flags "-sf kk -k on t 0 g 1"to the running command.

For all the results reported in the following sections, we repeated each experiment 5 times, and plotted the mean, max, and min values in the figures by the error bar. However, for the experiments using *in.porter* as input, we executed only once since it already included several cases. Table 2 lists the information of the input files of all experiments in this paper. Note that *in.porter* is used in bond-order paper [7] and the rest of the input are used in Tersoff paper.

4. Accuracy study

The goal of accuracy study is to understand how much accuracy will be sacrificed for choosing a less precise solver compare to the reference double precision solver. We used 32 MPI processes to run the accuracy study and used *in.tersoff-acc* [4] as our input. We recorded the total potential energy from each simulation step, and observed the relative differences between different precision solvers. Fig. 1 shows the difference between single and double precision solver, and Fig. 2 shows

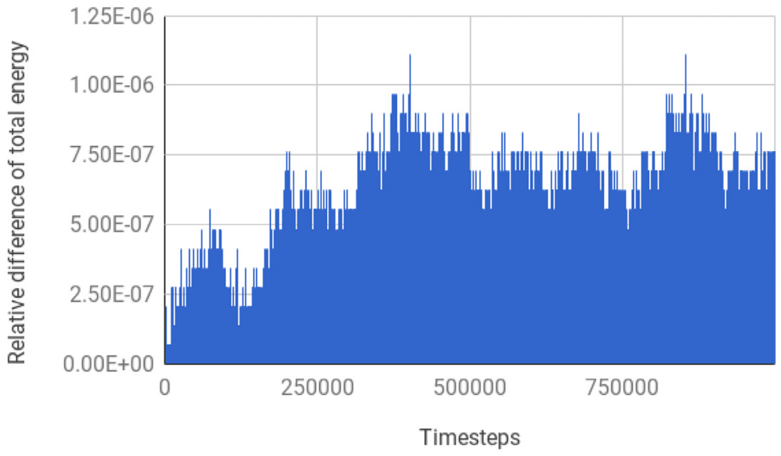

Fig. 2. Validation of the mixed precision solver. Relative difference of total energy between mixed and double solver.

Table 3
CPU used for benchmarking in Tersoff paper.

ISA	CPU	Cores
SSE	Intel Xeon X5675	2×6
AVX	Intel Xeon E5-2450	2×8
AVX2	Intel Xeon E5-2680v3	2×12

Table 4
Speedup comparison of single-threaded performance between the results from our cluster and the Tersoff paper.

	Paper double	Our exp. double	Paper single	Our exp. single
AVX	3.1	2.88	4	3.6
AVX2	3.1	3.4	4.8	4.8
AVX512	N/A	4.2	N/A	5.8

the difference between mixed and double precision solver. Both of the differences of total energy are less than 0.002%, which meets the claim of Tersoff paper.

5. Performance study

The goal of performance study is to evaluate the performance improvement of optimized solvers under various resource settings. Three different resource settings were tested, single-threaded, single node, and GPU offloading. For single-threaded execution, *in.tersoff* [5] was used as input, while single node and GPU tests, *in.tersoff_bench* [6] *and* a smaller dataset *in.porter* [7] were used as input dataset. Table 3 lists the hardware used for benchmarking in the Tersoff paper, where ISA is the instruction set architecture.

5.1. Single-threaded execution

To evaluate the speedup of the optimized code, we performed experiment on single MPI process first. Fig. 3 shows the performance of single thread execution on our machine. In Fig. 3, Ref represents the original LAMMPS code, while Opt-D, Opt-S, Opt-M, represent double, single, mixed solver optimized by Tersoff Paper. By comparing speedup of 4 different vectorized binaries, we can see performance improvement when switching to more advanced vector instruction set architecture (ISA). Table 4 shows the speedups comparison between the results from our cluster and the result from Tersoff paper. The relative difference of double and single precision for both AVX and AVX2 are within 10%.

5.2. Single node execution

To further verify the effect of the optimized code, we performed the experiment using a whole node CPU cores, that is, 24 MPI process running mixed solver. Fig. 4 shows the results for the execution on entire node for input file *in.tersoff_bench* [6]. The observed relative speedup between Ref and Opt-M-AVX2, Opt-M-AVX512 were 3.92 and 4.07 respectively. As in the Tersoff paper, highest speedup among all tested CPU is 3.15. Since the communication takes time, the improvements are

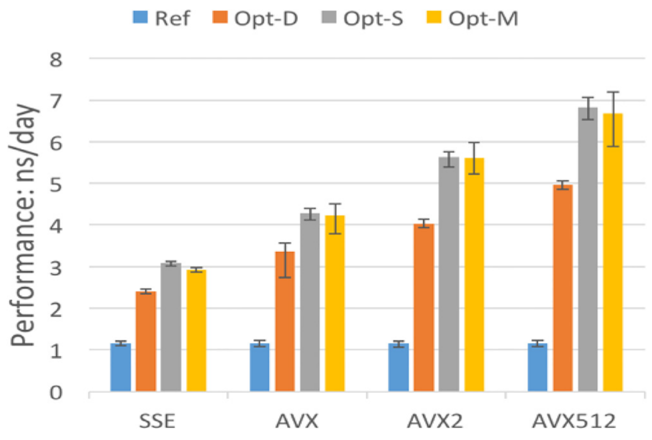

Fig. 3. Single-threaded performance of reference solver, and SSE, AVX, AVX2, AVX512 optimized solvers of double, single, mixed precision.

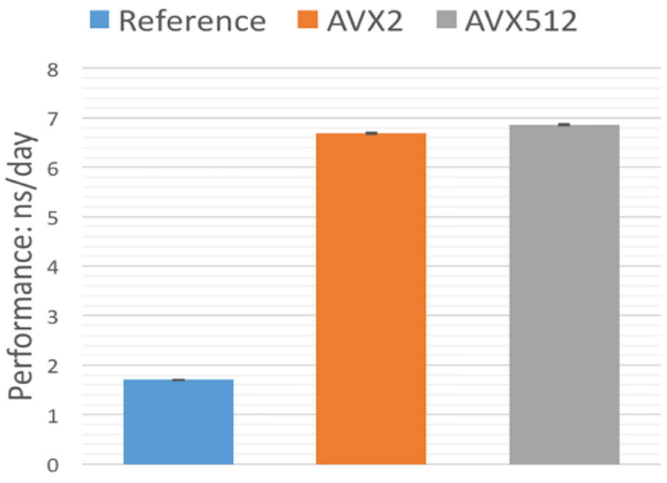

Fig. 4. Single node performance of reference solver, and AVX2, AVX512 optimized solvers of mixed precision. In.tersoff_bench as input.

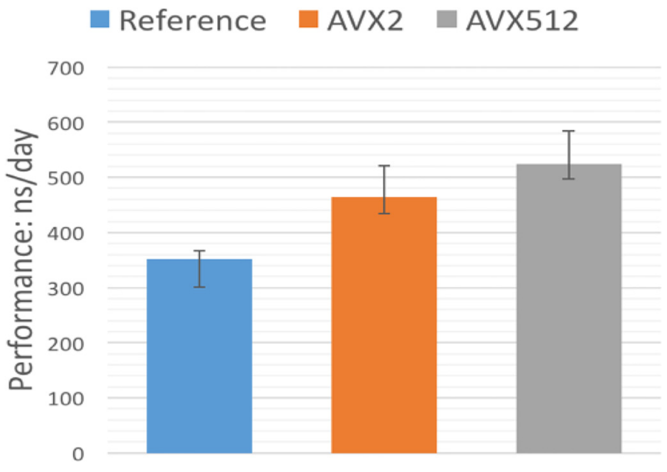

Fig. 5. Single node performance of reference solver, and AVX2, AVX512 optimized solvers of mixed precision. In.porter as input.

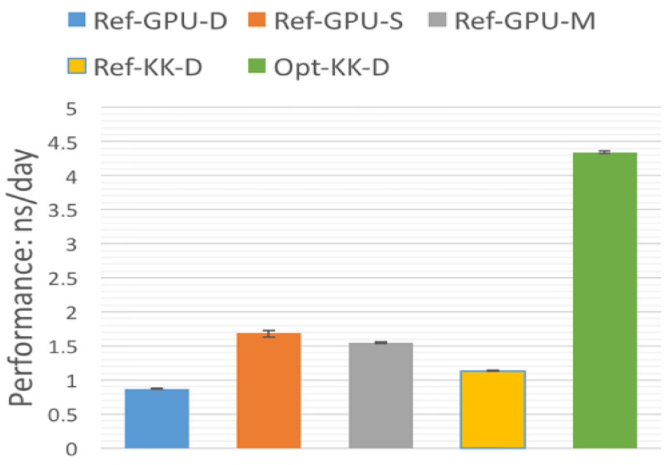

Fig. 6. GPU performance of reference solver of double, single, mixed precision, KOKKOS reference, KOKKOS optimized double solver.

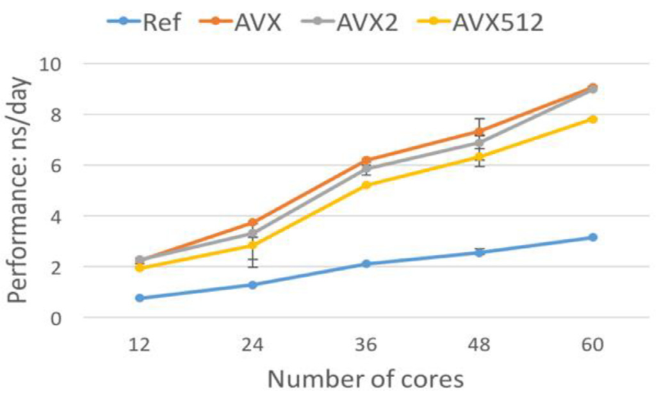

Fig. 7. Strong scaling using *in.tersoff_bench* as input for the reference LAMMPS (Ref) and Tersoff implementation using AVX, AVX2 and AVX512 optimized binaries.

lower than the results from single-thread execution shown in Fig. 3, which is 4.8 times speedup. Fig. 5 shows the results of *in.porter* [7] dataset. Because *in.porter* [7] was a smaller dataset, we observed less performance differences.

5.3. GPU offloading

We ran our test on Nvidia Tesla P100 at lower clock rate 999 MHz, due to our power strategy. Fig. 6 reports performance measurements when running *in.tersoff_bench* [6] on the GPU. In Fig. 6, Ref-GPU-D, Ref-GPU-S, Ref-GPU-M represent original LAMMPS GPU double, single, mixed solver respectively, while Ref-KK-D and Opt-KK-D represent the original and optimized KOKKOS double solver. Comparing Ref-KK-D and Opt-KK-D, our reproduction shows 3.8 times improvement, which is similar to 3 times as the Tersoff paper [1] stated. However, we failed to run the *in.porter* dataset on GPU due to an error "Cannot yet use minimize with Kokkos" reported in LAMMPS document [3].

6. Scalability study

Finally, we studied the strong scaling of the optimized solver by plotting the performance from using 12 MPI processes up to 60 MPI processes on our cluster. Figs. 7 and 8 show the strong scaling for reference LAMMPS and optimized binaries using *in.tersoff_bench* [6] and *in.porter* [7] dataset. The vectorized versions have greater speedup comparing to the reference version. All binaries show similar growth of 3.6 times performance improvement when the experiment scales to 5 times the number of cores. This result is similar to the numbers reported by the original Tersoff paper. However, for the results of shown in Fig. 8, the performance become even worse given more CPU. This is likely due to the size of *in.porter* [7] dataset is not large enough, which only contains only 216 atoms. As a result, the performance of the experiment taking *in.porter* [7] dataset as input can be easily affected by other environment factors, such as communication overhead.

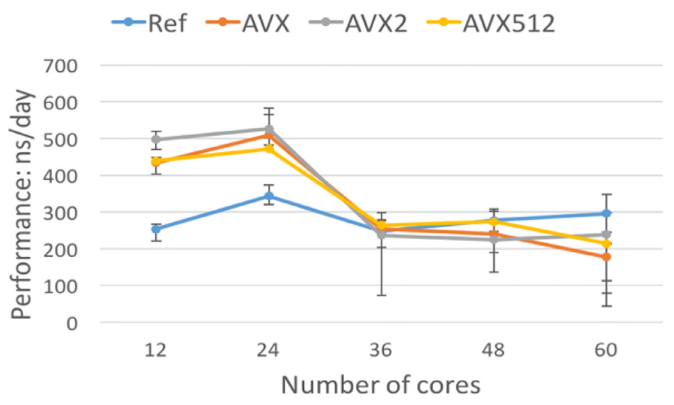

Fig. 8. Strong scaling using *in.porter* as input for the reference LAMMPS (Ref) and Tersoff implementation using AVX, AVX2 and AVX512 optimized binaries.

7. Conclusion

We replicated the experiment results in the Tersoff paper [1] on our GPU cluster. For accuracy study, we observed that the accuracy differences between double solver and less precise solvers, single and mixed solver, are less than 0.002% which satisfies the claim from the original paper. For performance study, we showed that all optimized solvers had greater performance than the reference solver as claimed by the original paper. Finally, for scalability study, we got similar results for strong scalability study when using the same input data size as the original paper. But the performance did not grow as we provided more computation unit when a less atom input was used. Overall, these results convinced us that work from Markus Höhnerbach et al. could be successfully reproduced on our cluster, even though the hardware architecture is different from the original paper.

References

[1] M. Höhnerbach, A.E. Ismail, P. Bientinesi, The vectorization of the tersoff multi-body potential: an exercise in performance portability, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '16), Piscataway, NJ, USA, IEEE Press, 2016 Article 7, 13 pages.
[2] Quanta Cloud Technology. Available at: https://www.qct.io/
[3] S. Corporation. 12. Errors — LAMMPS documentationDocumentation. Retrieved from: http://lammps.sandia.gov/doc/Section_errors.html
[4] in.tersoff-acc. Retrieved from: https://github.com/HPAC/lammps-tersoff-vector/blob/master/benchmarks/lammps/in.tersoff-acc
[5] in.tersoff. Retrieved from: https://github.com/HPAC/lammps-tersoff-vector/blob/master/benchmarks/lammps/in.tersoff
[6] in.tersoff_bench. Retrieved from: https://github.com/HPAC/lammps-tersoff-vector/blob/master/benchmarks/lammps/in.tersoff_bench
[7] L.J. Porter, et al., Empirical bond-order potential description of thermodynamic properties of crystalline silicon, J. Appl. Phys. 81 (1) (1997) 96–106.
[8] Student Cluster Competition, 2017. Available at http://www.studentclustercompetition.us/2017/index.html.
[9] S. Páll, B. Hess, A flexible algorithm for calculating pair interactions on SIMD architectures, Comput. Phys. Commun. 184 (2013) 2641–2650.
[10] W.M. Brown, J.-M.Y. Carrillo, N. Gavhane, F.M. Thakkar, S.J. Plimpton, Optimizing legacy molecular dynamics software with directive-based offload, Comput. Phys. Commun. (2015).
[11] W.M. Brown, M. Yamada, Implementing molecular dynamics on hybrid high performance computers—three-body potentials, Comput. Phys. Commun. (2013).
[12] C. Hou, J. Xu, P. Wang, W. Huang, X. Wang, Efficient GPU-accelerated molecular dynamics simulation of solid covalent crystals, Comput. Phys. Commun. (2013).