Parallel Computing 78 (2018) 47–53

Contents lists available at ScienceDirect

Parallel Computing

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

Student cluster competition 2017, team Tsinghua University: Reproducing vectorization of the tersoff multi-body potential on the Intel Skylake and NVIDIA Volta architectures

Ka Cheong Jason Lau, Yuxuan Li, Lei Xie, Qian Xie, Beichen Li, Yu Chen, Guanyu Feng, Jiping Yu, Xinjian Yu, Miao Wang, Wentao Han, Jidong Zhai*

Department of Computer Science and Technology, Tsinghua University, Beijing, China

ARTICLE INFO

Article history:
Received 9 March 2018
Revised 2 July 2018
Accepted 17 July 2018
Available online 19 July 2018

Keywords:
Reproducibility
Vectorization
Molecular dynamics simulation
Tersoff potential
Student cluster competition

ABSTRACT

A paper of SC '16 entitled "*The Vectorization of the Tersoff Multi-Body Potential: An Exercise in Performance Portability*" Höhnerbach et al. (2016) [1] implemented reduced precision calculation and cross-platform vectorization for Tersoff potential, which the authors claimed as accurate, efficient, scalable and portable. In this report, we focus on recently released computing architectures, Intel Skylake and NVIDIA Volta, to present our results and compare them with the Tersoff paper. With new input provided by Porter et al. (1997) [2], we run the given testcases on our cluster and obtain results that not consistent with the performance improvements and scalability claimed in the original publication. Deeper analysis demonstrate that it is the communication bottleneck caused by special characteristics of the new input data that limit the reproducibility.

1. Introduction

LAMMPS is a parallel molecular dynamics simulator written in C++ [3], supporting not only MPI-based parallelization, but also OpenMP, USER-INTEL, KOKKOS [4] and GPUs [5] modes. With the improvement of computational power, people expect to make simulations more accurate on higher performance platforms. Therefore, though costly, multi-body potential, in contrast to pair potential, has higher popularity and better multi-body potential computation optimization. However, the complexity of multi-body potential kernel limits the range of available architectures with native vectorization porting [1].

Höhnerbach, Ismail and Bientinesi attempt to overcome these challenges by implementing a general kernel approach in their SC '16 paper, *The Vectorization of the Tersoff Multi-Body Potential: An Exercise in Performance Portability* [1], which we will refer to as "the Tersoff paper". In this paper, a mechanism named general building blocks is designed, which is consistent in different architectures.

By identifying general building blocks, abstracting corresponding vectorization back-end, applying methods in different modes including USER-INTEL [6] and KOKKOS package [4], and reducing precision on well-examined calculations, authors claim that performance could be gained and scaled well without accuracy reduction in the Tersoff paper. More specifically, three claims are made:

* Corresponding author.
E-mail addresses: my@ucla.edu (K.C.J. Lau), zhaijidong@tsinghua.edu.cn (J. Zhai).

https://doi.org/10.1016/j.parco.2018.07.002
0167-8191/

Table 1
Environment specification.

Nodes	Dell EMC PowerEdge R740 × 4
CPU per node	Intel Xeon Platinum 8176 × 2
GPU per node	NVIDIA Tesla V100 × 2
Memory per node	DDR4 2666 MT/s 16GB × 12
Storage	Intel SSD DC P3700 1.8 TB × 2
Network	Mellanox EDR InfiniBand
Operating System	CentOS Release 7.4.1708
Kernel	Linux kernel 3.10.0

Table 2
Compilers and environment.

Compilers	Intel C++ Compilers 2018 CUDA 9.0
MPI	Intel MPI Library 2018
Math Kernel Library	Intel MKL 2018

Accuracy Maintenance due to the limited interaction with one given atom, round-off error might not accumulate since only interactions contribute to the force.

Performance Gain speedup by applying the vectorization method can reach 2 × to 3 × on most CPUs, and between 3 × and 5 × on accelerators.

Strong Scaling the optimization scales to multi-node circumstances automatically without degrading due to the overhead, e.g. communication in the real world.

We attempt to reproduce the three claims above in this report.

2. Reproduction environment

Our cluster that is used to reproduce the result of the Tersoff paper is set up as listed in Table 1.

Intel Xeon Platinum 8176 is of 28 cores per socket and AVX-512 ISA including conflict detection extension. With it, we can validate or challenge the claim that higher performance can be achieved with longer vectors, 512 bits, and can also test if the conflict detection instructions can improve the serialized updating as the authors expected in the "future".

NVIDIA Tesla V100 GPUs provide powerful computation ability. The warp of CUDA works similarly as $32 \times 64 = 2048$ bits SIMD, which is wide enough to evaluate the effectiveness of vectorization.

With hexa channel provided by Xeon 8176, 126GB/s memory bandwidth can be achieved per socket. Hyper-threading is enabled to overlap memory access and computation. With the Tesla V100, applications can run with an exceptional memory bandwidth – 900GB/s. Therefore, the impact of memory bottleneck is slightly ruled out so as to focus on computational optimization.

Equipped with this recent high-performance hardware, we are able to demonstrate that the original claim "the optimization will be 'future proof' " is tenable and validate the portability of Tersoff paper's results with SIMD architectures that it focused on.

3. Compilation and run

To better reproduce the results described, we choose to clone from the Tersoff Git repository,[1] attached as an artifact in the Tersoff paper's appendix, since the code in official LAMMPS repository is kept updating and might be mixed with other optimization.

The compilers and running environment is listed as below (Table 2):

By using *icc 2018* and *CUDA 9.0*, we can verify the portability on the compiler side. But we have to do minor modifications on *intel_simd.h* to the warp SIMD data types by construction based on LAMMPS official version 11Aug17, and to change some deprecated options in *Makefile.intel_cpu* after confirming these would not change the underlying behavior. Also, the corresponding GPU arch settings are changed to be "sm_70" to support the Tesla V100.

Using the provided workflow in the Tersoff paper, we can use the scripts *build.sh* and *bench-*.sh* to compile and run the reference CPU version, different CPU versions of the optimized binary with configuration macros, optimized and original KOKKOS versions and reference GPU versions using corresponding packages and parameters.

[1] github.com/HPAC/lammps-tersoff-vector.

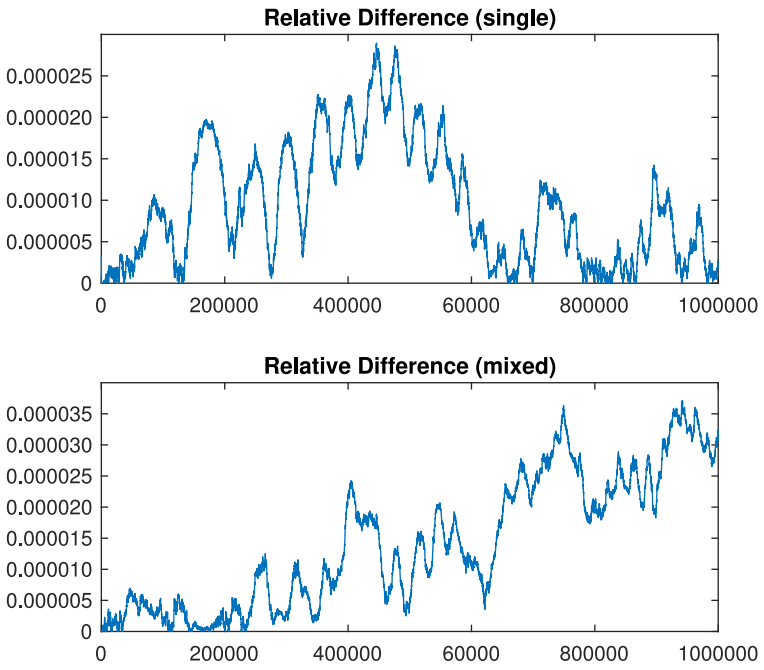

Fig. 1. Relative Δ total energy between the *single*/*mixed* and *double* precision solvers for a system of 32 000 atoms for 10^6 timesteps on *16 processes*.

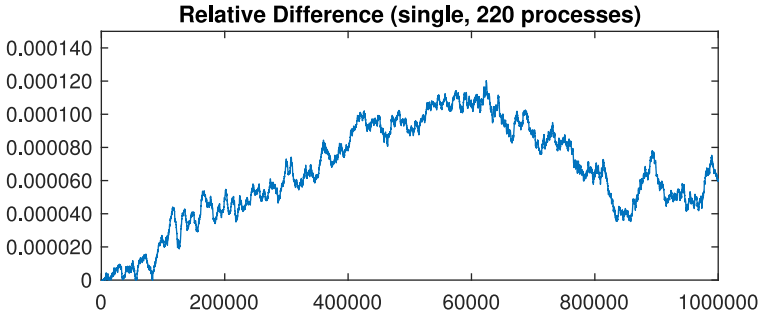

Fig. 2. Relative Δ total energy between the *single* and *double* precision solvers for a system of 32 000 atoms for 10^6 timesteps on *220 processes*.

4. Reproduction work

Four pieces of work are done to fully reproduce the Tersoff paper's result: *Accuracy Study* validates the reduced precision implementations; *Single-Threaded Execution* gives the purest representation of the speedup obtained by the optimizations; *Single Node Execution* shows the results on an entire node; while *Strong Scalability* scales to a cluster with multiple nodes, which is a real-world scenario.

We evaluate the accuracy using the same *in.tersoff-acc*, and obtain the results of single-threaded execution with the same *in.tersoff* used in the Tersoff paper. For single and multiple nodes executions, besides the original *in.tersoff_bench* [1] file, a new input from the paper: *Empirical bond-order potential description of thermodynamic properties of crystalline silicon* [2] by Porter et al. is also used.

4.1. Accuracy study

In the Tersoff paper, they create versions that compute Tersoff potential in single and mixed precision. The energy measurement in a long-running simulation is used to validate these reduced precision implementations. Here we repeat the procedure to test relative difference, including single to double and mixed to double, in order to validate that the accuracy can be maintained under our hardware and software setup.

As Fig. 1 illustrates, in a running of 16 MPI processes, exactly the same numbers of processes the Tersoff paper used, the Δ total energy of both single and mixed reside within 0.004% of the double reference, slightly exceed the original results

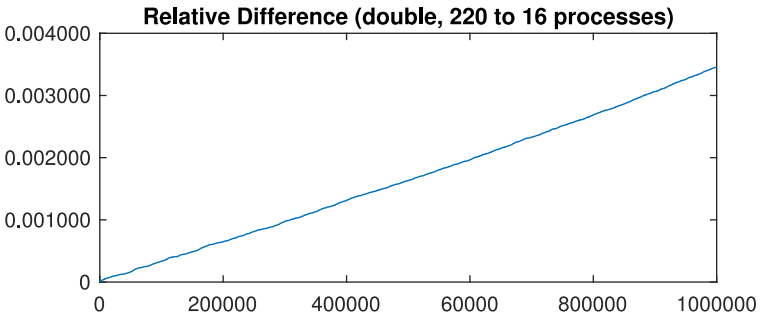

Fig. 3. Relative Δ total energy between *220 processes* and *16 processes* double precision solvers for a system of 32 000 atoms for 10^6 timesteps.

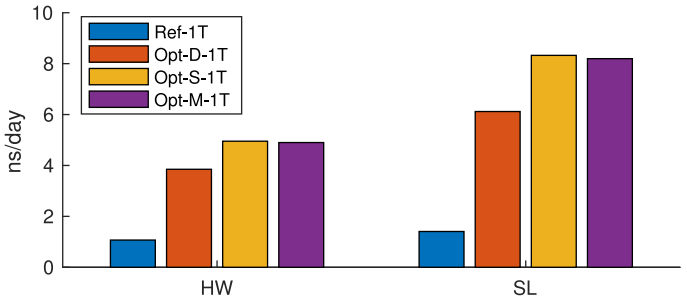

Fig. 4. Performance across different CPU architectures; single-thread run on *in.tersoff*. 32 000 atoms.

of 0.002% as shown in Fig. 3 of the Tersoff paper, but still in a reasonable range. This can be justified because different architectures and environments can result in different random states.

Several extra runs reveal that with MPI process count increasing, the relative difference gets worse. With 32 processes, the difference doubles. Fig. 2 shows that the deviation of single exceeds 0.01% with 220 processes. But by comparing to the relative difference accumulated between the results of 220 processes and 16 processes using the same double precision solver, we believe that the error caused by reducing precision can be neglected.

Therefore, a change from double to single or mixed would neither cause a significant nor accumulate error. We successfully reproduce the accuracy experiment.

4.2. Execution performance

In the Tersoff paper, Fig. 4 shows single-threaded performance for all different execution modes: Ref,[2] Opt-D, Opt-S and Opt-M[3]; 5 shows realistic single node performances for Ref and Opt-M; 6 presents single GPU results for two different GPUs: K20x and K40. Architectures including ARM, Kepler, Westmere, Sandy Bridge, Haswell, and Broadwell are tested, which give substance to portability.

To provide more data points as evidence, we choose Skylake (SL) with AVX-512 that support to run this experiment. The cloud component provided by CycleCloud gives us a chance to have tests on another architecture, Haswell (HW), during the competition, as a replication of existing results in the Tersoff paper.

4.2.1. Single-threaded execution

Since the single-threaded execution can eliminate the effect of parallelism from multiple threads or nodes, it shows purely the essential speedup using vectorization optimization.

Note that the scale of the used test data is tiny, therefore sensitive to interference, we try to disable all unused daemon on the system, and set a negative[4] nice value to the running application. With 100 data points, we can display more accurate performance metrics.

As shown in Fig. 4, both Skylake and Haswell perform expected speedup comparing Ref with Opt-*, e.g. a factor of 4.8 between Opt-S and Ref on Haswell as described in the Tersoff paper, and 5.9 on Skylake.

[2] As stated by Höhnerbach in an email reply, LAMMPS run without USER-INTEL package. OMP package is necessary on hyperthreading environment.
[3] Run with USER-INTEL package, in the default_vector flavor, set mode parameter with double (D), single (S), or mixed (M).
[4] High priority.

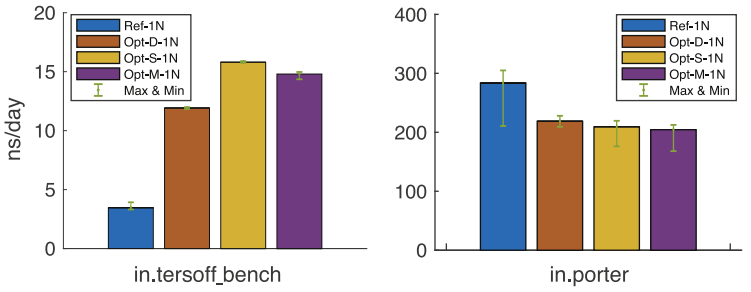

Fig. 5. Performance on Skylake (SL) Intel Xeon Platinum 8176 ×2; single-node multi-process run on *in.tersoff_bench* and *in.porter*.

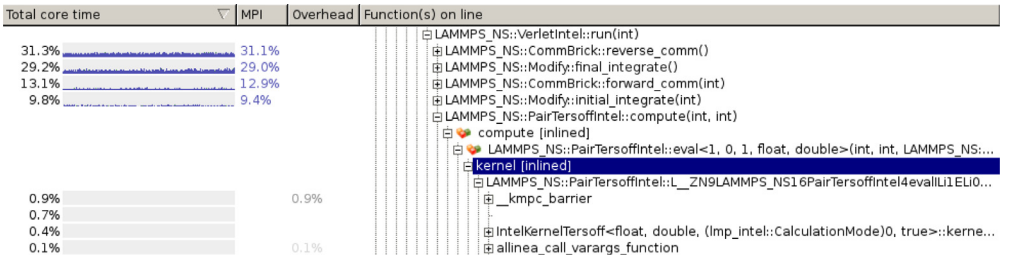

Fig. 6. Hotspots measured with Allinea MAP.

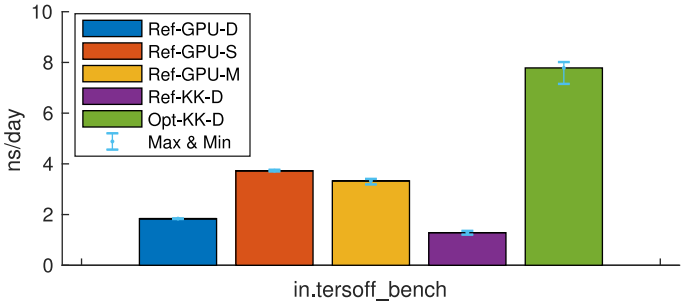

Fig. 7. Performance on NVIDIA Tesla V100; single-card run on *in.tersoff_bench*.

4.2.2. Single node execution

Single node execution mixes the vectorization and multi-thread speedup, introducing communication overhead and bandwidth problems. Due to the space limitations, we only present the results of Skylake in Fig. 5 with 4.4× speedup on *in.tersoff_bench*, and 0.74× on *in.porter* here. Besides, more experiments showing similar results can also be obtained on Haswell, the CPU architecture on the cloud.

Part of the performance reduction of Porter's input [2] could be attributed to the nature of its calculation. Porter's work is to calculate the thermodynamic properties of crystalline silicon, while silicon is in a state of solid, therefore the atoms are located densely. For a process, after computing the Newton force, f_i and f_j need to be updated in halo calculation, which might be handled by another process, so communication is required. Since the atoms in the model are dense, a process may have a lot of halo atoms sent to an adjacent MPI process, contributing to the high cost of communication.

With the Allinea MAP profiling tools provided by SCC, we can see in Fig. 6 that most of the time is consumed in MPI communication, while the Tersoff kernel used less than 1% of the total execution time, which proves that the communication cost caused by the property of the input accounts for the slowdown.

In conclusion, we can completely reproduce the expected performance increase on the exact same input provided in the Tersoff paper. But with the new input *in.porter* from Porter's paper [2], performance dramatically drops, due to the characteristic of the given task that requires lots of communication.

4.2.3. GPU execution

Additionally, we perform experiments on NVIDIA Tesla V100 to test the portability of the Tersoff paper claimed, as Fig. 7 shown. 6.1× speedup is achieved, greater than the 5× in the paper, which might be attributed to the improved bandwidth.

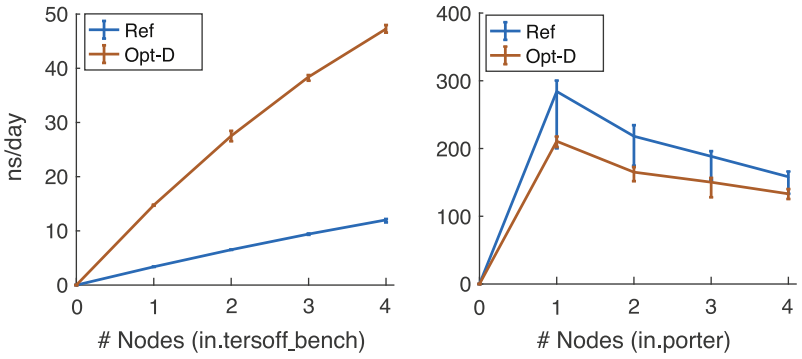

Fig. 8. Strong scalability with *in.tersoff_bench* and *in.porter* as input.

4.3. Strong scalability

As we all know, we care more about the performance at scale rather than single thread or single node execution because they rarely appear in realistic simulations. In this section, we measure the optimization effect on our cluster to validate its scalability.

With *in.tersoff_bench* as the input file, the performance on our cluster has a nearly linear improvement as expected while nodes are increased, see Fig. 8. But we cannot reproduce the result when we use *in.porter* as input. This is due to the same communication issue described in Section 4.2.2.

5. Utilizing AVX512-CD

AVX512-CD (Conflict Detection) is a new feature of AVX512 instruction set, which enables us to compute histogram in parallel. It is mentioned that the new feature can be utilized to optimize the Newton force f accumulation by constructing non-conflict set using conflict detection instruction *_mm512_conflict* and broadcasting instruction

_mm512_broadcast repeatedly and then updating array f by using the non-conflict set mask. But this idea is not implemented in the Tersoff paper.

Followed by Tersoff paper's idea, we have developed three versions with single, double and mixed precision respectively. Unfortunately, no matter which version we use, we got a lower performance (overall 7% reduction) by applying AVX512-CD to both f_i and f_j update because i index is easy to conflict. Besides, there is still overall 1% performance reduction even only AVX512-CD on f_j update are used. Through more analysis, we find that conflict detection introduces more overhead than benefits because only one addition can be vectorized.

The result shows that AVX512-CD is not suitable for the Tersoff potential computation. To make full use of the new feature, more compute-intensive patterns are necessary.

6. Conclusion

Through our effort to reproduce the Tersoff paper's result, we demonstrate that by using the input provided in the Tersoff paper, we can smoothly reproduce the accuracy analysis, the performance improvements and the strong scalability not only on the already tested platforms like Haswell architecture by using the cloud components provided by SCC, but also on our cluster equipped with the most advanced architectures, like Intel Skylake and NVIDIA Volta which was not available when the Tersoff paper released. This proves the portability of the methods proposed in the Tersoff paper.

However, we find that the performance improvements and scalability cannot be reproduced with the tasks provided in Porter's paper. After close scrutiny, it is found that the special characteristic of the task where atoms are dense causes intense communication. Therefore the communication becomes a bottleneck, causing the impossibility to have expected overall speedup with the method specified in the Tersoff paper.

Acknowledgment

We thank all reviewers for their insightful comments. This work is partially supported by the National Key R&D Program of China (Grant No. 2016YFB0200100), National Natural Science Foundation of China (Grant No. 61722208, 61472201).

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, in: SC '16, IEEE Press, Piscataway, NJ, USA, 2016, pp. 7:1–7:13.

[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.

[3] S. Plimpton, Fast parallel algorithms for short-range molecular dynamics, J. Comput. Phys. 117 (1) (1995) 1–19.
[4] H.C. Edwards, C.R. Trott, D. Sunderland, Kokkos: enabling manycore performance portability through polymorphic memory access patterns, J. Parallel Distrib. Comput. 74 (12) (2014) 3202–3216.
[5] W.M. Brown, P. Wang, S.J. Plimpton, A.N. Tharrington, Implementing molecular dynamics on hybrid high performance computers–short range forces, Comput. Phys. Commun. 182 (4) (2011) 898–911.
[6] 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. 195 (2015) 95–101.