Performance of Molecular Dynamics Acceleration Strategies on Composable Cyberinfrastructure

Richard Lawrence
rlawrence@tamu.edu
Texas A&M University
HPRC[1]
College Station, TX, USA

Dhruva K. Chakravorty
chakravorty@tamu.edu
Texas A&M University
HPRC[1]
College Station, TX, USA

Francis Dang
francis@tamu.edu
Texas A&M University
HPRC[1]
College Station, TX, USA

Lisa M. Perez
perez@tamu.edu
Texas A&M University
HPRC[1]
College Station, TX, USA

Wesley Brashear
wbrashear@tamu.edu
Texas A&M University
HPRC[1]
College Station, TX, USA

Zhenhua He
happidence1@tamu.edu
Texas A&M University
HPRC[1]
College Station, TX, USA

Honggao Liu
honggao@tamu.edu
Texas A&M University
HPRC[1]
College Station, TX, USA

James X. Mao
xjamesmao@tamu.edu
Texas A&M University
HPRC[1]
College Station, TX, USA

Chun-Yaung Lu
alu@tacc.utexas.edu
University of Texas
Texas Advanced Computing Center
Austin, TX, USA

ABSTRACT

Modern powerful accelerators and composable infrastructures put our simulation frameworks to the test. We will show that the acceleration of a simulation framework is absolutely critical for good performance and scaling. Building on our previous work using research software as a benchmark for computing clusters, the High Performance Research Computing Group (HPRC)[1] compares the Kokkos and GPU acceleration packages of LAMMPS (Large-scale Atomic/Molecular Massively Parallel Simulator) molecular dynamics software with NVIDIA H100 and Intel Data Center GPU Max 1100 accelerators on the composable ACES cyber-infrastructure at Texas A&M University. We observe different computational and communication patterns emerge from the different codes, which in turn result in different performance and scaling characteristics across these accelerators. We observe an opportunity for growth in the synergy between Intel's oneAPI toolchain and the Kokkos framework to enable effective scaling of molecular dynamics simulations on composable infrastructure.

CCS CONCEPTS

• **Hardware** → **Testing with distributed and parallel systems**; • **General and reference** → **Performance**; • **Computer systems organization** → **Reconfigurable computing**; • **Software and its engineering** → *Interoperability*.

[1]High Performance Research Computing (HPRC)

PEARC '24, July 21–25, 2024, Providence, RI, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0419-2/24/07
https://doi.org/10.1145/3626203.3670631

KEYWORDS

Molecular Dynamics, LAMMPS, NVIDIA, Intel, GPU, Kokkos, Benchmarking, Liqid, Composable Cyberinfrastructure

ACM Reference Format:
Richard Lawrence, Dhruva K. Chakravorty, Francis Dang, Lisa M. Perez, Wesley Brashear, Zhenhua He, Honggao Liu, James X. Mao, and Chun-Yaung Lu. 2024. Performance of Molecular Dynamics Acceleration Strategies on Composable Cyberinfrastructure. In *Practice and Experience in Advanced Research Computing (PEARC '24), July 21–25, 2024, Providence, RI, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3626203.3670631

1 INTRODUCTION

The native language of all NVIDIA GPUs is the proprietary CUDA language. Intel GPUs do not have a monolithic native language, but instead rely on community languages such as OpenCL and SYCL. Community languages can, in theory, program any GPU, but they have not yet been able to displace CUDA as the primary method of programming NVIDIA GPUs. For Intel GPUs, meanwhile, Intel OneAPI integrates a variant of SYCL to enable application developers to target multiple GPU platforms simultaneously, which may provide performance portability [3]. The community framework Kokkos is an ever higher-level interface that targets maximum performance portability by enabling developers to write code that can be later translated to any of several lower-level interfaces, including CUDA and SYCL [16]. The shift from the direct use of hardware-specific languages to higher-level interfaces is led by developers who wish to avoid maintaining multiple branches of code targeting different hardware interfaces.

Community codes from research applications offer a more holistic method for benchmarking hardware than direct hardware capability tests and often yield tangential benefits to the community; this strategy has repeatedly resulted in recognizable and informative results, providing valuable insights into the performance of research workflows on the FASTER and ACES composed GPU systems [8][2][11][4][9]. This approach was exemplified in our previous work, where we utilized established molecular dynamics research workflows as benchmarks for composed GPU systems [7].

Within the LAMMPS molecular dynamics simulation software [15], the "GPU package" created in 2010 [1] follows an effort-efficient development philosophy, and thus is only partially GPU-accelerated. The most impactful simulation steps, such as pairwise force calculations, can be offloaded to the GPU, while others can only be performed on a CPU. Steps that are not accelerated include the execution of LAMMPS' *fix* commands, FFT computation in the PPPM calculation, and the atom modify step. As a result, the GPU package must move the entire simulation between the GPU and the host on every timestep, which places demands on communication bandwidth. In addition, the GPU backends–CUDA and OpenCL–are directly integrated into the code so redundant development is needed to keep them both supported. After 2016, the community largely began concentrating development of LAMMPS on a Kokkos build option. The Kokkos package today is more completely GPU-accelerated; for some simulations, all of the steps can be performed on the GPU [10]. The older GPU package of LAMMPS that uses CUDA and OpenCL directly still exists, but has not benefited from the same level of development and lacks many of the features that the Kokkos package enjoys [14]. Thus, the Kokkos variant of LAMMPS is the community standard for performance on supported GPUs. Previous benchmarks have shown that the all-on-the-GPU acceleration strategy, currently available only from the Kokkos package, outperforms the partial acceleration strategy for simulation workloads above a certain size [13] [10]. Since simulation workloads have only gotten larger over time, this advantage is expected to remain.

This paper focuses on the acceleration strategies employed by the Kokkos and GPU packages of LAMMPS. Despite the names, we will not be evaluating the GPU interfaces these packages utilize.

2 MATERIALS AND METHODS

The performance benchmarks were performed on the National Science Foundation ACES cluster. ACES is a composable testbed supported by High Performance Research Computing at Texas A&M University. ACES offers a variety of accelerators, including NVIDIA H100 GPUs and Intel Intel Data Center GPU Max 1100 GPUs, also known as Ponte Vecchio (PVC). These GPUs are connected to server nodes via a PCIe 4 Liqid composable fabric, which can provide 8 or more GPUs to a single host. While a traditional cluster communication fabric centralizes communication to serve shared resources, such as a network filesystem, a composable fabric is distributed throughout the cluster to provide more direct routes between nearby devices. This configuration provides unique opportunities for efficient parallelization of accelerated code.

Several versions and variants of LAMMPS were tested, built from source code released between 2023 and 2024; for brevity, we will refer to them by abbreviation. LAMMPS with Kokkos with Cuda for NVIDIA GPUs (KCN) was retrieved from the NVIDIA container registry, image tag patch_15Jun2023 [12]. LAMMPS with the GPU package with Cuda for NVIDIA GPUs (GCN) was built from source, tag 2Aug2023_update2, using the Cmake build strategy [6]. LAMMPS with the GPU package with OpenCL for Intel GPUs (GOI) was built from source, from the develop branch on Feb 21, 2024, [5] using the Makefile strategy and the Intel OneAPI toolchain, with OpenMP support [6].

Multiple LAMMPS test problems were utilized to load the GPUs. Our previous work selected three established LAMMPS benchmark problems to explore the scaling of bonding and non-bonding force calculations: the Lennard-Jones (LJ), Embedded Atom Model (EAM), and Rhodopsin systems. Each system presents unique computational characteristics and challenges, providing a comprehensive benchmark for the GPUs [7]. Since only non-bonded Lennard-Jones interactions are involved, the Lennard-Jones (LJ) system is the simplest and least computationally demanding. In contrast, the Rhodopsin system is the most complex and computationally demanding, as it involves all types of interactions: bonded, non-bonded, and electrostatic.

Comparison of performance between the GPU package and the Kokkos package of LAMMPS is not straightforward. Because LAMMPS with Kokkos does not use the CPU for calculations, the optimal number of CPU processes per GPU in the Kokkos framework is exactly one - with a strong performance penalty for additional processes. The number of CPU cores is not expected to be a limiting factor for performance of the Kokkos package. However, because a GPU package build of LAMMPS does perform calculations using CPU, we must choose some representative number of CPUs in order to compare the benchmarks. One could find the optimal ratio of CPUs to GPUs to efficiently utilize resources while achieving good performance across a range of problem scales; on ACES nodes we find that this corresponds to a linear scaling of approximately 28 CPUs per GPU, with some variance related to the node's composition and the molecular system. See supplemental documentation for details [6]. Therefore, for small numbers of GPUs, we scale the number of CPUs linearly according to this ratio. However, when scaling up to higher numbers of GPUs on a single composed Liqid fabric node, this strategy fails because we are limited by available CPUs. Instead, the number of CPUs stops scaling at an optimal maximum of 84 for LJ and EAM and 48 for Rhodopsin, again based on our findings on ACES nodes.

For the GOI build of LAMMPS, Intel provides a tool for MPI orchestration named "Compute Aggregation Layer" which we use to reduce the communication overhead of the many MPI processes. While the OpenMP framework is available to fill a similar role, it has not been found to be beneficial for GPU-focused simulations.

In our previous work, we showed that GPUs perform better for atomic simulation when the problem size is large enough to occupy a significant fraction of memory [7]. Based on those results, we select our LJ and EAM problem size to be 32 million atoms. For the Rhodopsin case, the problem size is chosen to be 4 million atoms. Strong scaling was measured, wherein the system size remains constant while the assigned computing resources are increased.

3 RESULTS

3.1 Processor scaling (GPU package build)

We performed an experiment to show how the GOI build scales with the number of processes on the host. As shown in Figure 1, the GPU package requires a significant number of cores to obtain good

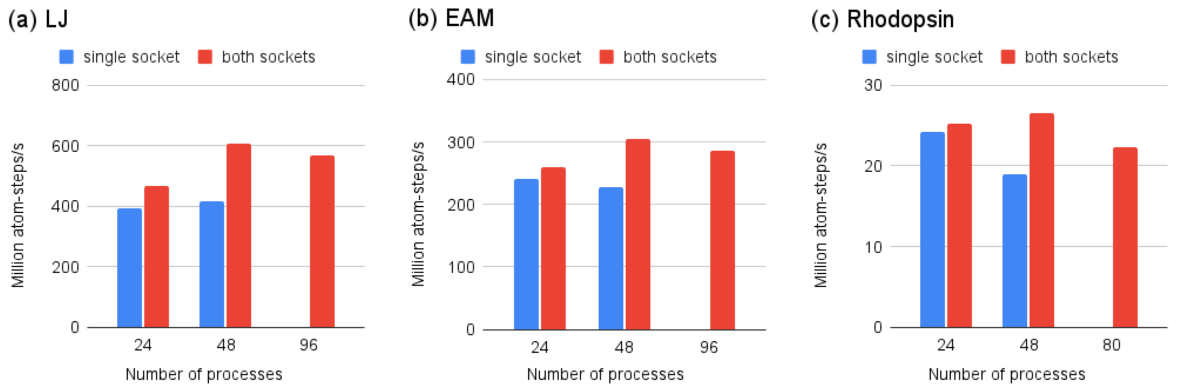

Figure 1: Performance (in million atom-steps/s) of the LAMMPS GPU package as function of the number of processes for the benchmarks of (a) LJ, (b) EAM, and (c) Rhodopsin in two scenarios: pinning processes to a single socket (blue) and to two sockets (red). The experiment was performed on a traditional PVC GPU node with two sockets per node and 48 cores per socket.

performance – around half of the cores on this node for a single GPU. We further observe that the optimal number of processes is less than the total number of cores on the node. This experiment shows that the LAMMPS GPU package has a demand for both cores and other host resources. The host resource requirements limit the ability of the simulation to scale up on multiple GPUs on the Liqid fabric, where the GPU count can grow while the CPU count remains fixed. Additionally, the processes should be spread across a node as much as possible; confining the processes to a single socket (48 cores) or filling an entire node with processes results in a performance penalty. This limits the ability of the GPU package to scale up on a single node. These observations foreshadow that attempting to scale up GPUs and processes at the same time on a single Liqid fabric node is a lost cause.

3.2 GPU scaling (three builds)

We performed experiments with both the GPU and Kokkos packages to see how they each scale with the number of GPUs on a Liqid fabric. For the GPU package, the number of processes increases with the number of GPUs up to a maximum in accordance with our scaling strategy, which occurs at three GPUs. As shown in Figure 2, we can see that the GPU package struggles to extract any additional performance from four or more GPUs on a single node. By contrast, the Kokkos package has no difficulty scaling up beyond four GPUs because it has very low host resource requirements. The Kokkos package scales much better than the GPU package on Liqid composable fabric nodes. This is because it isn't limited by host resources such as cores and is less communication-intense. The significant boost in performance ascribed to the all-on-the-GPU strategy of the Kokkos package is consistent with previous benchmarks performed with NVIDIA GPUs [10]. As seen in Figure 3, communication accounts for a very large fraction of time expent during the LJ and EAM tests of the GOI build. The performance difference for Rhodopsin is less pronounced, because Rhodopsin requires more operations per atom; communication between the host and the GPU is a relatively smaller fraction of the total simulation wall time.

4 DISCUSSION AND CONCLUSIONS

We highlight the need to account for data transfer capability in addition to raw compute power for GPUs. In the Liqid fabric, communication is fast but bandwidth is limited because communication hardware is shared among multiple devices. Thus, scalability is highly dependent on the data strategy of the parallelization framework. In this case, the Kokkos package of LAMMPS delivers the best performance because it minimizes the need for data movement, while the GPU package struggles to scale up. We predict that this will continue to be the winning strategy for the foreseeable future as composable fabrics and other novel HPC architectures become commonplace.

The results here show that the scaling of the LAMMPS GPU package has a strong need for CPU cores and other host resources, and struggles to scale up beyond 3 GPUs on a Liqid fabric node. We suspect that competition between processes for host resources, such as memory or communication bandwidth, limits the ability of processes to scale up. On the other hand, the Kokkos package has no difficulty scaling beyond 4 GPUs on a single node.

NVIDIA supports LAMMPS development on the Kokkos package. Meanwhile, Intel supports a direct OpenCL implementation for LAMMPS within the GPU package. The LAMMPS Kokkos package with the OneAPI SYCL backend for Intel GPUs remains a work-in-progress (see Appendix B). We anticipate that benchmarking results for Intel Data Center GPU Max 1100 would show improvement over what has been reported in this paper if Intel invested in the development of the Kokkos framework to enable the use of LAMMPS with OneAPI's SYCL backend.

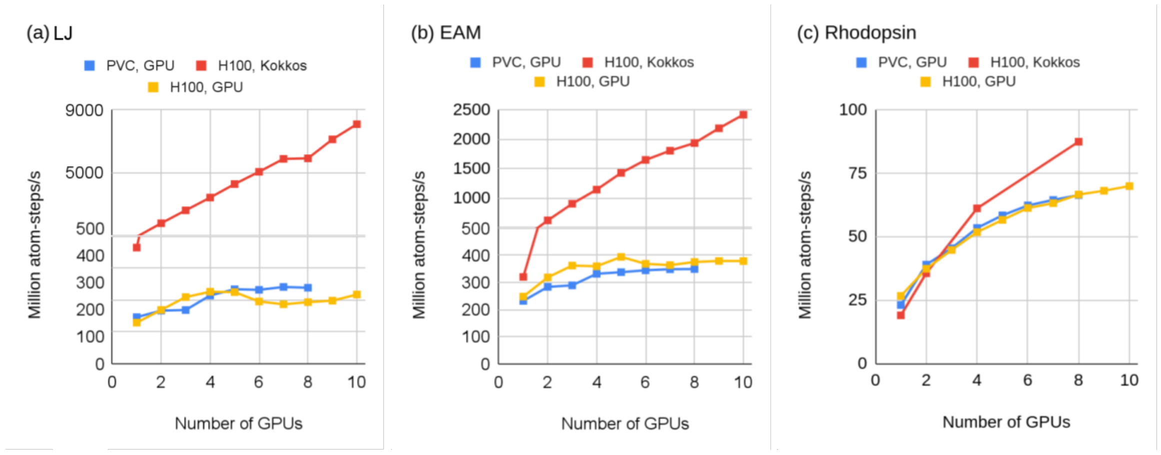

Figure 2: Performance of the three LAMMPS builds under strong scaling for the benchmarks of (a) LJ, (b) EAM, and (c) Rhodopsin, using the previously defined MPI strategy and accelerated by Intel PVC (blue) and NVIDIA H100 GPUs (red, yellow) on the Liqid composable fabric nodes. Note that the vertical axis is piecewise in subplots (a) and (b). The similarity between GCN and GOI builds indicates that the acceleration strategy is the limiting factor rather than hardware capability.

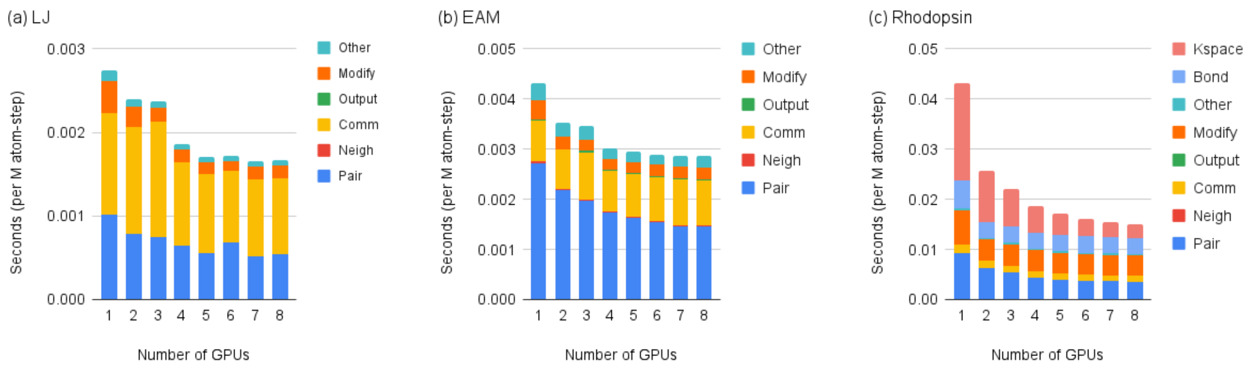

Figure 3: Time spent on by category for the LAMMPS GOI build during strong scaling benchmarks of (a) LJ, (b) EAM, and (c) Rhodopsin, using the previously defined MPI strategy and accelerated by Intel PVCs on the Liqid composable fabric nodes. Comm represents time spent waiting for transfer of data between the host and the GPU. Only the Pair and Kspace calculations are GPU-accelerated in this build.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation (NSF) award number 2112356 ACES - Accelerating Computing for Emerging Sciences; and NSF CC* Cyberteam award number 1925764 SWEETER - SouthWest Expertise in Expanding, Training, Education and Research.

REFERENCES

[1] W. Michael Brown, Peng Wang, Steven J. Plimpton, and Arnold N. Tharrington. 2011. Implementing molecular dynamics on hybrid high performance computers – short range forces. *Computer Physics Communications* 182, 4 (2011), 898–911. https://doi.org/10.1016/j.cpc.2010.12.021

[2] Zhenhua He, Aditi Saluja, Richard Lawrence, Dhruva K. Chakravorty, Francis Dang, Lisa M. Perez, and Honggao Liu. 2023. Performance of Distributed Deep Learning Workloads on a Composable Cyberinfrastructure. In *Practice and Experience in Advanced Research Computing* (Portland, OR, USA) *(PEARC '23)*. Association for Computing Machinery, New York, NY, USA, 12 pages. https://doi.org/10.1145/3569951.3603632

[3] Intel. 2024. *Data Parallel C++: the oneAPI Implementation of SYCL**. Retrieved April 2024 from https://www.intel.com/content/www/us/en/developer/tools/oneapi/data-parallel-c-plus-plus.html

[4] Druva Gunda Kumar, Zhenhua He, Lujun Zhai, Dhruva K. Chakravorty, Francis Dang, Lisa M. Perez, and Honggao Liu. 2024. Accelerator Performance for AI/ML Workloads on Composable Infrastructure. In *Practice and Experience in Advanced Research Computing (PEARC '24)*. Association for Computing Machinery, New York, NY, USA, 12 pages.

[5] LAMMPS. 2024. *LAMMPS Source Code*. https://github.com/lammps/lammps "develop (Wed Feb 21)".

[6] Richard Lawrence. 2024. *Supplemental Documents for PEARC24 Proceedings paper 139: Performance of Molecular Dynamics Acceleration Strategies on Composable Cyberinfrastructure.* https://github.com/rarensu/pearc24-LAMMPS-supplement

[7] Richard E Lawrence, Dhruva K Chakravorty, Zhenhua He, Francis Dang, Lisa M Perez, Wesley A Brashear, and Honggao Liu. 2023. Developing Synthetic Applications Benchmarks on Composable Cyberinfrastructure: A Study of Scaling Molecular Dynamics Applications on GPUs. In *Practice and Experience in Advanced Research Computing* (Portland, OR, USA) *(PEARC '23)*. Association for Computing Machinery, New York, NY, USA, 216–220. https://doi.org/10.1145/3569951.3597556

[8] Sambit Mishra, Freddie Witherden, Dhruva K. Chakravorty, Lisa M. Perez, and Francis Dang. 2023. Scaling Study of Flow Simulations on Composable Cyberinfrastructure. In *Practice and Experience in Advanced Research Computing* (Portland, OR, USA) *(PEARC '23)*. Association for Computing Machinery, New York, NY, USA, 6 pages. https://doi.org/10.1145/3569951.3597565

[9] Sambit Mishra, Freddie Witherden, Dhruva K. Chakravorty, Lisa M. Perez, and Francis Dang. 2024. Memory Bandwidth Performance across Accelerators. In *Practice and Experience in Advanced Research Computing (PEARC '24)*. Association for Computing Machinery, New York, NY, USA, 6 pages.

[10] Stan Gerald Moore. 2019. LAMMPS KOKKOS Package: The quest for performance portable MD. (8 2019). https://www.osti.gov/biblio/1641575

[11] Abhinand Nasari, Hieu Le, Richard Lawrence, Zhenhua He, Xin Yang, Mario Krell, Alex Tsyplikhin, Mahidhar Tatineni, Tim Cockerill, Lisa Perez, Dhruva Chakravorty, and Honggao Liu. 2022. Benchmarking the Performance of Accelerators on National Cyberinfrastructure Resources for Artificial Intelligence / Machine Learning Workloads. In *Practice and Experience in Advanced Research Computing* (Boston, MA, USA) *(PEARC '22)*. Association for Computing Machinery, New York, NY, USA, Article 19, 9 pages. https://doi.org/10.1145/3491418.3530772

[12] NVIDIA. 2023. *NVIDIA Container Registry: LAMMPS.* https://catalog.ngc.nvidia.com/orgs/hpc/containers/lammps "tag:patch_15Jun2023".

[13] Steve Plimpton, Aidan Thompson, Stan Moore, Axel Kohlmeyer, and Richard Berger. 2016. *LAMMPS Benchmarks.* Retrieved April 2023 from https://www.lammps.org/bench.html

[14] Steve Plimpton, Aidan Thompson, Stan Moore, Axel Kohlmeyer, and Richard Berger. 2024. *LAMMPS Accelerator Packages Comparison.* Retrieved April 2024 from https://docs.lammps.org/Speed_compare.html

[15] Aidan P. Thompson, H. Metin Aktulga, Richard Berger, Dan S. Bolintineanu, W. Michael Brown, Paul S. Crozier, Pieter J. in 't Veld, Axel Kohlmeyer, Stan G. Moore, Trung Dac Nguyen, Ray Shan, Mark J. Stevens, Julien Tranchida, Christian Trott, and Steven J. Plimpton. 2022. LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales. *Computer Physics Communications* 271 (2022), 108171. https://doi.org/10.1016/j.cpc.2021.108171

[16] Christian R. Trott, Damien Lebrun-Grandié, Daniel Arndt, Jan Ciesko, Vinh Dang, Nathan Ellingwood, Rahulkumar Gayatri, Evan Harvey, Daisy S. Hollman, Dan Ibanez, Nevin Liber, Jonathan Madsen, Jeff Miles, David Poliakoff, Amy Powell, Sivasankaran Rajamanickam, Mikael Simberg, Dan Sunderland, Bruno Turcksin, and Jeremiah Wilke. 2022. Kokkos 3: Programming Model Extensions for the Exascale Era. *IEEE Transactions on Parallel and Distributed Systems* 33, 4 (2022), 805–817. https://doi.org/10.1109/TPDS.2021.3097283

A ADDITIONAL BENCHMARK EXPERIMENTS

We also performed experiments to show how the GPU and Kokkos packages scale across GPUs on multiple nodes. See supplemental documentation for details [6]. We found that the performance of Kokkos across multiple nodes is slightly worse than a single node, which is consistent with our previous findings [7]. We found that the performance of the GPU package across multiple nodes is actually better than the scaling on a single node, which is logical because the additional nodes provide additional cores to perform computational work.

B KOKKOS FOR INTEL PVC GPUS

Building LAMMPS with Kokkos for Intel PVC GPUs was achieved using the Makefile strategy and SYCL from the Intel OneAPI toolchain. See supplemental documentation for details [6]. This build lacks features needed for benchmarking, so no tests were performed.