Appendix: Artifact Description

Artifact Description (AD)

I. Overview of Contributions and Artifacts

A. Paper's Main Contributions

- C_1 Novel mapping and dataflow EAM MD algorithm for atomistic simulations on the Cerebras Wafer-Scale Engine (WSE).
- C_2 Reference SOTA performance measurements of EAM MD on GPUs and CPUs.
- C_3 Observed performance improved by orders of magnitude.
- C_4 Energy consumption improved by an order of magnitude or more.

B. Computational Artifacts

- A_1 CPU reference run descriptions (in this document).
- A_2 GPU reference run descriptions (in this document).
- A_3 WSE run description. The WSE code in the Tungsten dataflow programming language is undergoing a legal review for release.
- A_4 Power metrics

Artifact ID	Contributions Supported	Related Paper Elements
A_1, A_2	C_2	Table III Figure 2
A_3	C_1, C_3	Tables II-III Figure 2
A_3	C_1	Figure 8
A_4	C_4	Figures 2b and 2c

II. Artifact Identification

A. Computational Artifact A_1

Relation To Contributions

This artifact is the timing data of the CPU data points in Figure 2. The best performance points for the CPU systems are also displayed in Table III.

Expected Results

These computational experiments demonstrate the performance of the LAMMPS simulation code on CPUs. The resulting timings are used to compare to corresponding timing data for the WSE CS-2 code. These CPU results are significantly slower than the corresponding simulations on the WSE CS-2 chip, and puts its speed in context to leading conventional hardware and algorithms.

Expected Reproduction Time (in Minutes)

The expected computational time for a single benchmark experiment using dual-socket nodes with Intel Xeon E5-2695v4 CPUs, is up to 7 minutes using 8 nodes. The runtime is inversely proportional to the node-count up to around 50 nodes.

Artifact Setup (incl. Inputs)

Hardware: The experiments were conducted using the Quartz cluster at LLNL. Each node has 2 Xeon E5-2695v4 18C 2.1GHz CPUs, 128GBytes of DRAM. The nodes are interconnected with Intel Omni-Path network. Quartz is further documented here:

https://www.top500.org/system/178971/.

To reproduce all data points in Figure 2, 400 nodes are required.

Software: These experiments used the LAMMPS software, available for public download from lammps.org. The LAMMPS version was "Large-scale Atomic/Molecular Massively Parallel Simulator - 8 Feb 2023", and was retrieved from git with the specific commit hash tag of

`730e5d2e64106f3e5357fd739b44c7eec19c7d2a`

of the `develop` branch. The following LAMMPS packages were enabled:

MANYBODY, MEAM, MOLECULE, RIGID.

We used the MPI communication library "MPI v3.1: MVAPICH2 Version: 2.3.7 MVAPICH2 Release date: Wed March 02 22:00:00 EST 2022". The operating system version was Linux "Red Hat Enterprise Linux 8.9 (Ootpa)" 4.18.0-513.18.1.2toss.t4.x86_64 x86_64, and the SLURM scheduler was used for job submitting and launching jobs.

Datasets / Inputs: Scripts for generating initial conditions, and for running the timing benchmarks, as well as all necessary input files to LAMMPS, are in the git repository

`https://github.com/CerebrasResearch/Cerebras-Trilabs`,

in the `MD/CS2-AD-AE/cpu_benchmarks` sub-directory.

Installation and Deployment: To compile LAMMPS we used the standard MPI makefile `Makefile.mpi` included with the source code, which had the compiler and linker flags set to `-g -O3 -std=c++11`.

Compiler: Intel Classic C++ 20.21.6 / Intel(R) C++ g++ 10.3.1 mode with OpenMP not enabled C++ standard: C++11

Artifact Execution

An example command to perform the Cu benchmark on 8 nodes is:

```
nnodes=8
srun -N$nnodes -n$(($nnodes*36)) \
  lmp_mpi_quartz -in in.bench \
  -var element Cu -var pbc 0 \
  -var nx 172 -var ny 192 -var nz 6 \
  -log bench.log
```

To generate all initial conditions and to run all benchmarks, follow the procedure below:

1) Clone the git repository: `https://github.com/CerebrasResearch/Cerebras-Trilabs`.
2) Go into the sub-directory `MD/CS2-AD-AE/cpu_benchmarks`.

3) Edit the variable `lmp` in the beginning of `make_max_cfgs.sh` and `scale_bench.sh` to point to the LAMMPS executable to use.
4) Edit the `srun` commands in `make_max_cfgs.sh` and `scale_bench.sh` to be the correct job running/job submission commands for your machine.
5) Adjust the `nnodes` loop in `scale_bench.sh` to run over the desired node counts.
6) Run or source the `make_max_cfgs.sh` script to produce initial conditions. Using 8 nodes (as listed on the `srun` line in the script), this process will run three jobs which each should complete in less than two minutes. The resulting configuration files can be found in {`Cu,Ta,W`}`/cfgs`.
7) Run or source the `scale_bench.sh` script to perform the benchmark simulations. It runs one job per element per node count. On 8 nodes such a job takes up to 7 minutes. The runtime is inversely proportional to the node-count up to around 50 nodes.
8) The runtime for the simulations can be extracted by the command:
```
grep Loop nbench.log.*
```

B. Computational Artifact A_2

Relation To Contributions

This artifact is the timing data of the GPU data points in Figure 2. The best performance points for the GPU systems are also displayed in Table III.

Expected Results

These computational experiments demonstrate the performance of the LAMMPS simulation code on GPUs. The resulting timings are used to compare to corresponding timing data for the WSE CS-2 code. These GPU results are significantly slower than the corresponding simulations on the WSE CS-2 chip, and puts its speed in context to leading conventional hardware and algorithms.

Expected Reproduction Time (in Minutes)

The expected computational time on Frontier for a single benchmark data point is less than 2 minutes in all cases. All runs combined should take less than 30 minutes.

Artifact Setup (incl. Inputs)

Hardware: The experiments were conducted using the OLCF Frontier exascale supercomputer. Each Frontier node consists of 8 AMD Instinct MI250X GPU compute dies (GCDs) attached to an AMD Optimized 3rd Generation EPYC 64C 2GHz CPU. A total of 9,408 nodes are connected together by HPE's Slingshot-11 network. Frontier is further documented here: https://www.top500.org/system/180047/.

To reproduce all GPU data points in Figure 2, 32 nodes are required.

Software: The LAMMPS version was "Large-scale Atomic/Molecular Massively Parallel Simulator - 7 Feb 2023", and was retrieved from git with the specific commit hash tag of `edcbd2e7618518e6bb1a5d843081474d7176872d` of the `develop` branch.

Datasets / Inputs: Scripts for running the timing benchmarks, as well as all necessary input files to LAMMPS, are in the git repository

`https://github.com/CerebrasResearch/Cerebras-Trilabs`,

in the `MD/CS2-AD-AE/gpu_benchmarks` sub-directory.

Installation and Deployment: The following modules were loaded on top of the default modules: `cray-mpich/8.1.27`, `rocm/5.7.1`.

LAMMPS was compiled using `Makefile.frontier_kokkos` included with the source code. The following LAMMPS packages were enabled:

KOKKOS, MANYBODY.

The following environment variables were set:

- `MPICH_GPU_SUPPORT_ENABLED=1`
- `MPICH_OFI_NIC_POLICY=NUMA`
- `LD_LIBRARY_PATH=${CRAY_LD_LIBRARY_PATH}:${LD_LIBRARY_PATH}`

The SLURM scheduler was used for job submitting and launching jobs.

Artifact Execution

An example command to perform the Cu benchmark on 1 GCD is:

```
srun -u -N1 -n1 -c1 \
  --cpu-bind=map_cpu:50 \
  --gpus=1 ./lmp_frontier_kokkos \
  -in in.bench -k on g 1 -sf kk \
  -pk kokkos neigh full newton off \
  -var element Cu \
  -var nx 172 -var ny 192 \
  -var nz 6 -var pbc 0 \
  -log bench.log
```

An example command to perform the Cu benchmark on 8 nodes is:

```
srun -u -N 8 --ntasks-per-node=8 \
  --cpus-per-task=6 --gpus-per-task=1 \
  --gpu-bind=closest \
  ./lmp_frontier_kokkos \
  -in in.bench -k on g 1 -sf kk \
  -pk kokkos neigh full newton off \
  -var element Cu \
  -var nx 172 -var ny 192 \
  -var nz 6 -var pbc 0 \
  -log bench.log
```

To run all benchmarks, follow the procedure below:

1) Clone the git repository
   `https://github.com/CerebrasResearch/Cerebras-Trilabs`
2) Go into the sub-directory `MD/CS2-AD-AE/gpu_benchmarks`

3) Update the `#SBATCH --account=xxx` command in `run_multi.sh` to use your account on Frontier
4) Update the variable `EXE` in the `run_multi.sh` script to point to your LAMMPS executable
5) Submit the `run_multi.sh` to the Frontier queue using the SLURM `sbatch` command. This process will run several LAMMPS jobs, each of which should complete in less than two minutes.
6) The runtime for the simulations can be extracted by the command:
```
grep "Loop time" bench.log.*
```

C. Computational Artifact A_3

Relation To Contributions

This artifact is the timing data of the WSE data points in Figure 2. The performance points for the system are also displayed in Table III.

Expected Results

These computational experiments demonstrate the performance of the EAM MD implementation on WSE. The performance exceeds LAMMPS on CPU and GPU systems by a few orders of magnitude.

Expected Reproduction Time (in Minutes)

Each run takes less than 5 minutes.

Artifact Setup (incl. Inputs)

Hardware: WSE experiments were conducted on a CS-2 system equipped with 769×1044 processing elements. System clock rate is configurable and was set to 850MHz.

Software: The EAM algorithm was written in the Tungsten programming language and compiled with an internal compiler (April 2024 release).

Cerebras plans to make the compiler accessible to researchers through partnerships with PSC, EPCC, and participation in the NAIRR Pilot program.

Datasets / Inputs: The datasets from the LAMMPS runs are used for the WSE runs.

D. Computational Artifact A_4

Relation To Contributions

This artifact is the power data used to assess energy consumption for the CPU and GPU runs.

Expected Results

For the dual-socket CPU runs on the Quartz cluster, we assumed a power consumption of 300 W per node (36 cores). The is an estimate based on the maximum power use of the CPUs at 2×120 W per socket plus an additional 25% to allow for memory (DRAM), network equipment and auxiliary power usage. For the GPU runs on Frontier, we used the published power consumption[1] of 22.8 MW, and divided by 9472 compute nodes, to get a power consumption of about 2400 W per node.

[1]https://top500.org/lists/top500/list/2024/06, accessed online June 28, 2024

WSE systems include a power meter which constantly monitors operational power draw. In our experiments the readouts were stable and never exceeded the 23kW level which is the nominal power limit for CS-2 systems.

We believe these power estimates are sufficiently accurate to clearly illustrate the large difference in energy consumption between conventional hardware and the Cerebras CS-2 for the molecular dynamics simulations presented in this paper.