Artifact Evaluation Report

Fernanda Foertter
Voltron Data
Oak Ridge, TN USA
0000-0002-8906-3176

Quentin Guilloteau
University of Basel
Basel, Switzerland
0009-0003-7645-5044

I. Overview of Reproduction of Artifacts

The following table provides an overview of each computational artifact's reproducibility status. Artifact IDs correspond to those in the AD/AE Appendices.

Artifact ID	Available	Functional	Replicated
A_1	●	○	○
Badge awarded	yes	no	no

II. Reproduction of Computational Artifacts

A. Timeline

The artifact evaluation was conducted from July 10, 2024, to July 24, 2024.

B. Computational Environment and Resources

The experiments in the artifacts were simulations and were conducted on a local cluster of 2x10 Core Intel Xeon E5-2640 v4 2.40GHz Processor with 64GB of memory.

C. Details on Artifact Reproduction

- **Access to artifact**: The artifact has been shared with `anonymous.4open.science`. One might be careful to make sure when using this service that links continue to be made available into the future. Repositories shared through this service are unnecessary as the Artifact Evaluation is a single-blind process and no such requirement was mentioned in the submission process. webpage[1].
- **Content of the artifact**: The Artifact Description (AD) does not match the actual content of the artifact. The authors mentioned for example in the AD that they "supply default job-files (for Slurm cluster manager)" but those files were not found in the repository. Similarly, the authors mentioned the use of Jupyter notebooks for the analysis and the plotting, but there are no notebooks in the artifact.
- **Downloading the traces**: The authors provide a `perl` script to download the traces. These traces are downloaded from different sources whose long-term availability will likely be a challenge, especially given the multiple URL and fragmentation of the artifacts (`mega.nz`, `dropbox.com`, `dpc3.compas.cs.stonybrook.edu`). Downloading the traces was long (mostly because the `dpc3.compas.cs.stonybrook.edu` server has a limited bandwidth) and might have been possible to do in parallel. Indeed, we found that only 4 traces from `mega.nz` links were still available (at the time of this review). The authors mentioned in the AD "The 70 representative simulations....". The traces lacked appropriate labeling or documentation and made it difficult to identify which were part of these 70 simulations. In the following of the review, we found 142 traces.
- **Compilation**: The compilation process was confusing overall, and below are a few examples.
  - **DRAMSim3**: In the AE, the authors write "simply calling 'make' is usually all it takes". The actual process required a call to `cmake` first and then `make`. This was what was indicated in the `README` of DRAMSim3, but not found in the artifact documentation.
  - **ChampSim3**: The ChampSim3 `README` mentions a compilation process with the `config.sh` script and the configuration file `champsim_config.json`. But those files are not in among the artifact. Instead, `config.py` script was run (after changing some paths in the script code) with the configuration files in the `configs` folder. The `config.py` script generates a `Makefile` that was called with `make`.
- **Simple execution of ChampSim3**: There was a loader issue when running ChampSim3:

```
./bin/12C_4X_CXL_50ns_1MBLLC_TH70: error
    while loading shared libraries:
    libdramsim3.so: cannot open shared
    object file: No such file or directory
```

  This was solved by using the `LD_PRELOAD` environment variable:

```
LD_PRELOAD=/path/to/DRAMsim3/libdramsim3.
    so ./bin/12C_4X_CXL_50ns_1MBLLC_TH70 --
    warmup_instructions 2000 --
    simulation_instructions 5000 ../Pythia/
    traces/602.gcc_s-734B.champsimtrace.xz
```

- **Experiments**: The authors provided a `runall.py` Python script. This script will run **all** the simulations at once on the current machine. Again, the `LD_PRELOAD` trick was needed in order to run `champsim_run.py`.

[1] https://sc24.supercomputing.org/program/papers/reproducibility-appendices-badges/

The reviewers did not have a server as powerful as the authors, so one reviewer wrote a `Snakefile`[2] (see Listing 1) to execute the simulations on a local cluster (and remove the `&` in the `champsim_run.py`). The authors mentioned that the simulations can last to at most 12 hours. The reviwers note that 27 simulations could not finish under 16 hours.

- **Analysis**: The authors provided a Python script (`collect_stat.py`) to gather and aggregate the results from the experiments and generate a CSV. This Python script tried to read a file named `DDR5_baselineepoch.json` that was not included among the artifacts. The only mention of this file in the artifact was in the `.gitignore`. The reviewers commented out the line in the `collect_stat.py` that asks to read this file. The `generate_pickle.py` script generates Pickle files from the `collected_stats.csv` generated by the previous script. The plots are generated with the `plot_all.py` script. There was a `division by zero` error when computing the `ipc_gains`. This script generated a *single* figure in two formats (pdf and png). This figure (see Figure 1) is similar in style to Figure 5 of the paper, but it was not possible to compare properly to Figure 5 of the paper.
- **Software Environment**: *Not a single version* of the software packages and libraries to use was provided by the authors, and some dependencies are missing and not documented as needed, (e.g., `numpy`).
- **Documentation**: The documentation of the artifact is not thorough and refers to files that are not present in the artifact. Fortunately, the Python scripts provided are simple enough – even if sometimes messy (a lot of commented lines of code) – to read and understand.
- **Data and Figures**: The plotting script in the artifact is for a single figure of the paper. However, in the AD, the authors mentioned that figures 2, 5-9 can be reproduced. We were unable to produce these. Moreover, the data used by the authors in the paper is not available. The data and the plotting scripts for *all* the figures was a glaring omission. **Update:** A request for these data and plotting scripts was made to the authors and these were added to the `SCRIPTS/PLOTTING_SCRIPTS` folder, allowing reviewers to regenerate all the figures from their paper (`SCRIPTS/PLOTTING_SCRIPTS/allstats.csv`).
- **Simplified Experiments**: The authors decided that a "simplified plot (and the running/collecting process) for the volunteers" (from AD) was necessary but unfortunately, this decision does not help the Reproducibility Committee to assess of the reproducibility of the artifact. The choice to reproduce partially or not the results should have been left to the Reproducibility Committee in the future.

Disclaimer: This Artifact Evaluation Report was crafted by volunteers with the goal of enhancing reproducibility in our research domain. The time period allocated for the reproducibility analysis was constrained by paper notification deadlines and camera-ready submission dates. Furthermore, the compute hours in the shared infrastructure (e.g., Chameleon Cloud) available to the authors of this report were limited and restricted the scope and quantity of experiments in the review phase. Consequently, the inability to reproduce certain artifacts within this evaluation should not be interpreted as definitive evidence of their irreproducibility. Limitations in the time allocated to this review and the compute resources available to the reviewers may have prevented a positive outcome. Furthermore, reviewers assess the reproducibility of the artifacts provided by the authors; however, they are not accountable for verifying that the artifacts support the main claims of the paper.

[2]https://snakemake.readthedocs.io/en/stable/

APPENDIX

The following is the `Snakefile` that helped run the simulations instead of using the `runall.py` script provided by the authors (see Paragraph "**Experiments**" above). Snakemake 7.32.3 and Python 3.10.8 were used. But it should be noted that this sort of effort is above and beyond what should be required of peers in order to reproduce results in a paper.

Listing 1. `Snakefile` used instead of `runall.py`

```
import os
path_to_traces=f"{os.getcwd()}/Pythia/traces/"
trace_files = [os.path.splitext(f)[0] for f in
     os.listdir(path_to_traces) if f.endswith(
    '.xz')]
configs = {
  "Baseline": "12C_base_DDR_NOPAM",
  "CXL": "12C_4X_CXL_50ns_1MBLLC_TH70"
}
RESULT_FOLDER="results"
SBATCH = "sbatch␣./sbatch.bash"

rule all:
  input:
    expand(f"{RESULT_FOLDER}/{{config}}_{{
        trace}}", config=configs.keys(), trace
        =trace_files),

rule run_champsim:
  input:
    champsim="SCRIPTS/champsim_run.py",
  output:
    directory(f"{RESULT_FOLDER}/{{type}}_{{
        trace_name}}")
  wildcard_constraints:
    type="|".join(configs.keys())
  params:
    cfg = lambda w: configs[w.type]
  shell:
    f"""
    {SBATCH} python3 {{input.champsim}} --cfg
        {{params.cfg}} --tr {path_to_traces
        }/{{wildcards.trace_name}}.xz --
        out_dir {{output}}
    """
```

The following script (`sbatch.bash`) is the Slurm script used in the `Snakefile`:

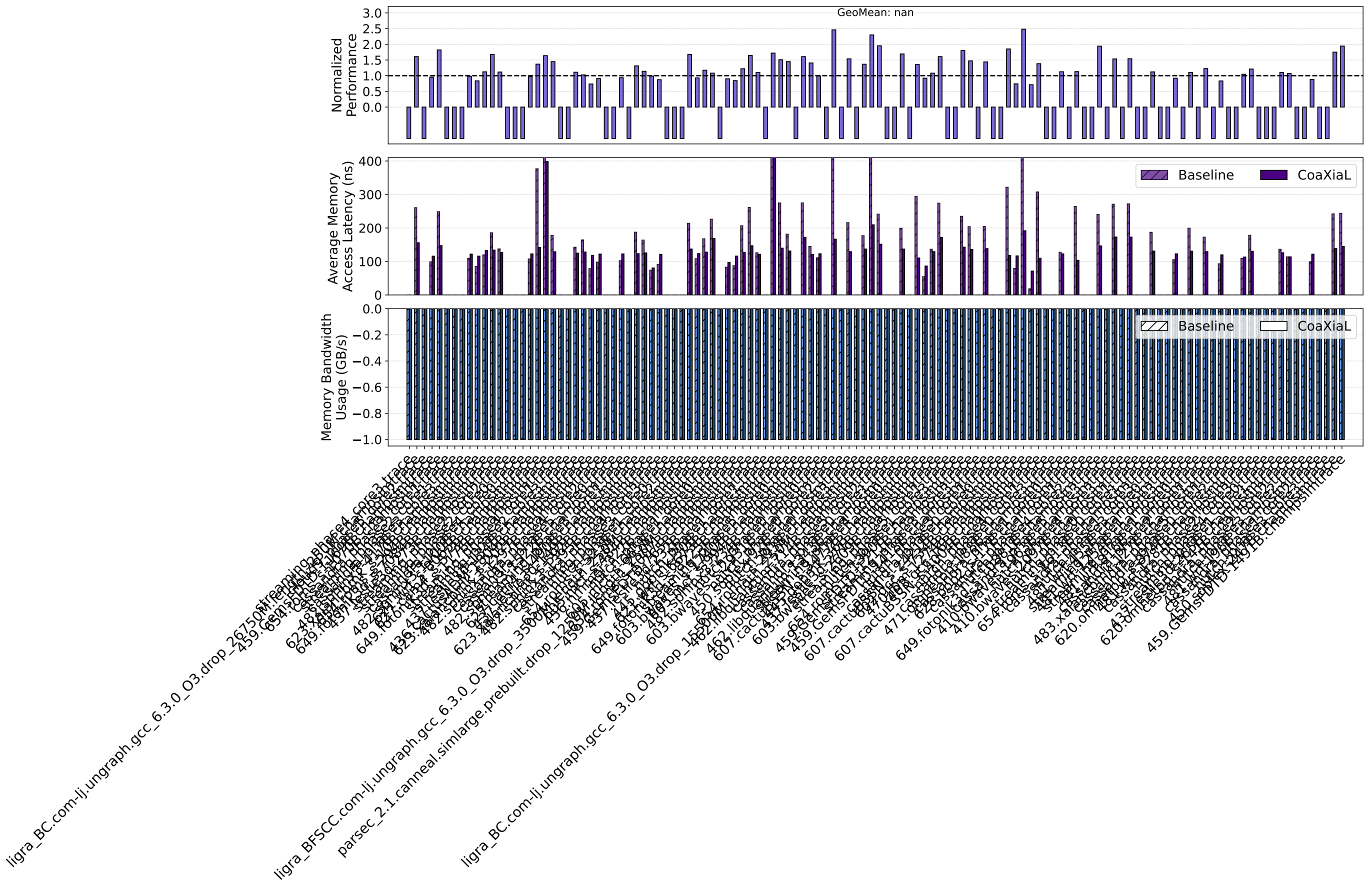

Fig. 1. Resulting plot of the reproduction attempt.

```
#!/bin/bash
#SBATCH --job-name=sc_ae
#SBATCH --mem-per-cpu=8GB
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#              d-hh:mm:ss
#SBATCH --time=0-16:00:00
#SBATCH --wait

set -ex
exec $@
exit 0
```