Reproducibility Report for ACM SIGMOD 2021 Paper: "LIMA: Fine-grained Lineage Tracing and Reuse in Machine Learning Systems"

MILOS NIKOLIC, University of Edinburgh, UK

The experiments are reproducible and support the key findings of the paper. All experiments provide performance numbers similar to those from the paper except for one experiment involving the Intel MKL library.

1 INTRODUCTION

This report summarizes the reproducibility evaluation for the paper entitled "LIMA: Fine-grained Lineage Tracing and Reuse in Machine Learning Systems" by Arnab Phani, Benjamin Rath, and Matthias Boehm, all from Graz University of Technology [1]. The paper appeared in the Proceedings of the 2021 ACM SIGMOD International Conference on Management of Data (SIGMOD'21).

2 SUBMISSION

The repository containing scripts and reproducibility instructions for this paper is available at:

https://github.com/damslab/reproducibility

under the sigmod2021-LIMA-p32 directory. The source code is available in the Apache SystemDS repository at https://github.com/apache/systemds. The code used in the memory usage experiment (Figure 6b) is available at https://github.com/phaniarnab/systemds/tree/reproducibility.

The reproducibility repository contains scripts for setting up the experimental environment, cloning and building the source code, downloading and preparing the datasets[1], running the experiments, regenerating all the plots, and recompiling the paper. The repository provides detailed instructions on how to run these scripts in either fully-automated or step-by-step mode.

3 HARDWARE AND SOFTWARE ENVIRONMENT

Table 1 shows the hardware and OS environments reported in the paper, recommended in the reproducibility instructions, and used in the reproducibility evaluation.

The difference in the hardware specification affects raw performance numbers but not the overall trends and key findings. The more memory available in the repro machine has no effect on the experiments involving LIMA and plain SystemDS since the JVM heap size was capped at 110GB.

Table 1. Hardware & Software environment

	Paper	Recommended	Repro Review
OS	Ubuntu 20.04	Ubuntu 20.04	Debian GNU/Linux 10
CPU	AMD EPYC 7302	Intel Xeon Platinum 8175M	Intel Xeon Silver 4214
Cores (physical)	16	16	48
GHz	3.0-3.3 GHz	2.5 GHz or above	2.2 GHz
RAM	128GB	128GB	188GB
Storage	SSD	SSD	HDD

[1]The datasets, APS and KDD 98, are available in the UCI Machine Learning repository at http://archive.ics.uci.edu/ml.

4 REPRODUCIBILITY EVALUATION

4.1 Process

I followed the step-by-step instructions provided in the reproducibility repository. Using the fully-automated script was impractical due to its lengthy execution of approximately 44.5 hours. The step-by-step method provided regular checkpoints as a means to check that the evaluation process was running without any problems. Due to a lack of admin rights on the used machine, I had to modify the system-setup procedure and install OpenJDK 1.8 and Intel MKL locally. Apart from that, the whole reproducibility process went smoothly and without any issues.

4.2 Results

The conclusions of the reproducibility evaluation are as follows.

- **Figure 6a: Runtime Overhead.**
  This experiment is *fully reproducible.*
- **Figure 6b: Space Overhead.**
  This experiment measures the space overhead of lineage tracing by forcing a JVM garbage collection after every 100 instructions. *This method occasionally returns inconsistent results due to the non-deterministic nature of garbage collection in JVM.*
  The first run of this experiment reported numbers identical to those from the paper except in one case – LTD and the batch size of 2 – where the reported memory usage was unusually high. The authors suggested repeating this experiment a few more times. Indeed, the second run produced results identical to those from the paper. The third run again produced one significantly higher data point, this time for the Base configuration and the batch size of 2.
- **Figures 7: Partial Reuse and Multi-level Reuse**
  This experiment is *fully reproducible.*
- **Figure 8: Cache Eviction Policies.**
  This experiment is *fully reproducible.*
- **Figure 9: Performance of End-to-end ML Pipelines.**
  This experiment is *reproducible.* The execution times are higher in plots (a)-(c) and slightly lower in plots (d)-(e) compared to the numbers from the paper. The obtained speedups in plot (f) are up to 20% lower in three cases, slightly higher in one case, and about the same in one case. But the overall trends and behavior of the competitors match those from the paper. This end-to-end experiment is also reproducible with the JVM heap size of 55GB (i.e., 50% of the initial heap). The trends and key findings still hold, and the only difference is in plot (c) where the two Base competitors crash when processing more than 700,000 rows. This end-to-end experiment with restricted memory is not in the original paper.
- **Figure 10 (a): ML Systems Comparison.**
  This experiment is *partially reproducible*, but the key finding about LIMA's ability to reuse computation holds. The execution times for the Base and LIMA systems are around 50% higher than those from the paper. The authors narrowed down the problem to the Intel MKL library, claiming its performance varies significantly across different processor architectures.
- **Figure 10 (b)-(d): ML Systems Comparison.**
  This experiment is *fully reproducible.*

REFERENCES

[1] Arnab Phani, Benjamin Rath, and Matthias Boehm. 2021. LIMA: Fine-grained Lineage Tracing and Reuse in Machine Learning Systems. In *Proceedings of the 2021 International Conference on Management of Data.* 1426–1439.