Reproducibility Report for ACM SIGMOD 2023 Paper: "AWARE:Workload-aware, Redundancy-exploiting Linear Algebra"

AUNN RAZA, EPFL, Switzerland
WENQING LIN, Tencent, China

This report tries to reproduce the Workload-aware Compression framework. Two independent reviewers reproduce the experiments in the paper, and we can reproduce the experimental results and the major findings. All the reproduced experiments succeeded and produced numbers supporting the author's claim. We reproduced the numbers on a single machine only, which supports the author's claims and, in the reviewer's opinion, supports the paper's central idea, and the concept is extendable to multiple machines in a distributed setup.

1 INTRODUCTION

This report tries to reproduce the result of the Workload-aware Compression framework [1]. The workload-aware approach exploits the redundant operations in linear algebra algorithms, in addition to data compression methods, for optimizing end to end execution times for machine learning pipelines.

Two reviewers independently reproduced this paper. The submitted experiments support the paper's central results and claims, and the reproduced results depict the same pattern and conclusion as those submitted in the original paper. We reproduced the experimental results on a single machine only. However, the results support the paper's claim and, in the reviewer's opinion, are sufficient for extrapolating to multiple machines in a distributed setup.

2 SUBMISSION

The submission includes the source of *workload-aware compression* through GitHub with a detailed readme, explaining the deployment environment and setup instructions, as well as scripts for generating data, running all or individual experiments, and generating the graphical plots. Following is the list of summary of submission contents with the URL for source code, setup instructions, and the data generator:

- GitHub repository with code and scripts at: Github link [2]
- Detailed readme file at URL: /readme/README.pdf (link)
- Data generation scripts at URL: /experiments/setup_local_data.sh (link)

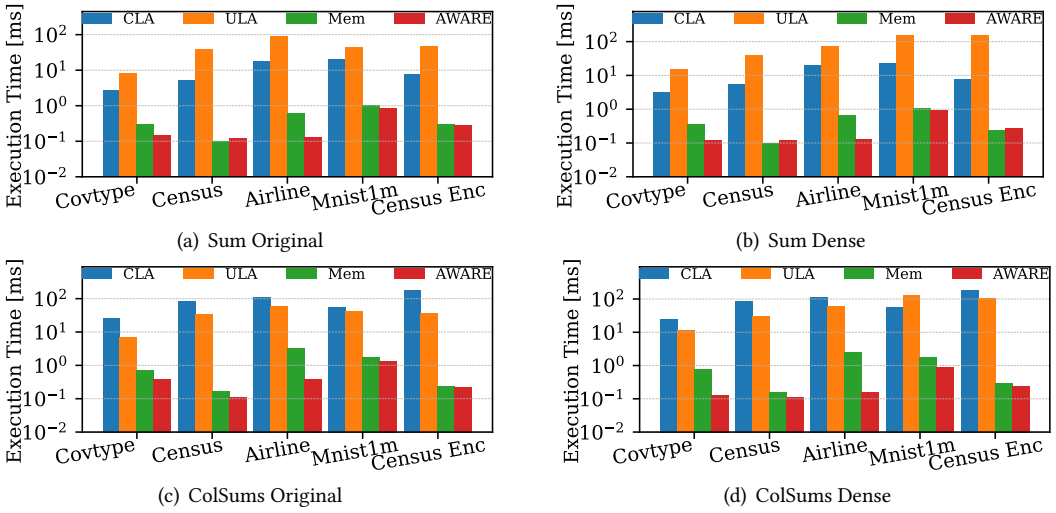
3 REPRODUCIBILITY EVALUATION

3.1 Hardware and Software Environment

All experiments in this section were reproduced on a single machine with 4x18-core Intel Xeon E7-8890v3 processor (32-KB L1I + 32-KB L1D cache, 256-KB L2 cache, and 45-MB LLC) clocked at 2.50 GHz, with Hyper Threads, summing to a total of 144 hardware threads, 1.8TB of SSD and 512-GB of DRAM. We used ubuntu 18.04, OpenJDK Java 11.0.22, Apache Hadoop 3.3.6, and Apache Spark 3.2.4.

3.2 Process

We reproduced the experimental numbers for a single machine and then analyzed and matched them with the original paper's reported numbers and claims. The numbers are reproduced by following the author's detailed readme to set up the environment, install the required dependencies, prepare data,



(a) Sum Original (b) Sum Dense

(c) ColSums Original (d) ColSums Dense

Fig. 1. Operations Performance Aggregate [Corresponding to original paper's figure 6]

and execute the benchmark script. We had to update environment paths in *loadSysMLSettings.sh*, and the machine's name in *parameters.sh* to match the corresponding experimental setup. Further, as reported in the original paper, we skipped some of the experiments that took too long to execute.

3.3 Results

This report reproduces the major findings of the paper. specifically, from the original paper, reproducing table 8 (Table 1), figure 6 (figure 1), figure 7a (figure 2), figure 8b (figure 3), figure 9b (figure 4), figure 10 (figure 5), figure 11b (figure 6), figure 12 (figure 7). Our local results corresponds to the original papers claim.

Table 1. *AWARE* Workload TOPS (Data: US Census Enc) [Corresponding to original paper's table 8]

	ULA		***AWARE***			
Op · 100	TOPS	Time	Est. TOPS	TOPS	Comp	Time
SUM	3.38e+10	3,59 sec	1.29e+05	1.13e+05	8,51 sec	0,08 sec
SUM Dense	1.90e+11	12,24 sec	1.31e+05	1.14e+05	12,66 sec	0,12 sec
Plus	1.90e+11	1 618,17 sec	1.31e+05	1.14e+05	12,91 sec	0,11 sec
RMM-256	2.81e+13	975,68 sec	2.13e+07	1.94e+07	13,46 sec	0,30 sec
LMM-256	4.28e+12	245,49 sec	7.05e+11	7.51e+11	16,19 sec	39,96 sec
TSMM	6.32e+12	168,60 sec	1.00e+12	1.07e+12	15,24 sec	13,30 sec
ScaleShift	7.47e+11	14 229,99 sec	4.08e+05	3.40e+05	12,32 sec	0,27 sec
Euclidean-256	4.80e+13	2 821,92 sec	8.61e+11	9.04e+11	17,23 sec	212,40 sec

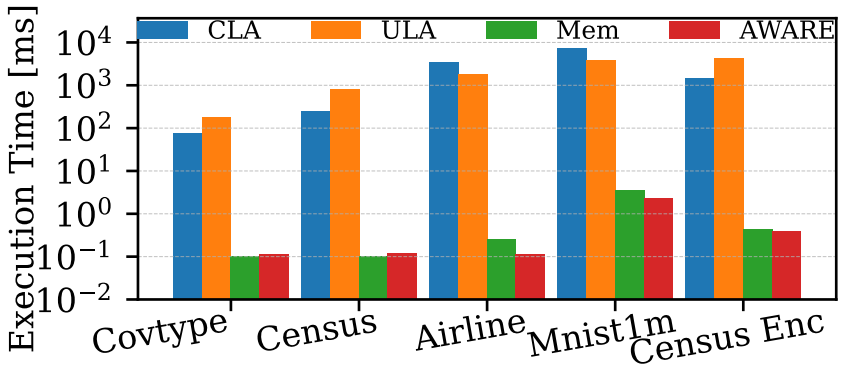


Fig. 2. Operations Performance Scalar (Plus Scalar, Original) [Corresponding to original paper's figure 7a]

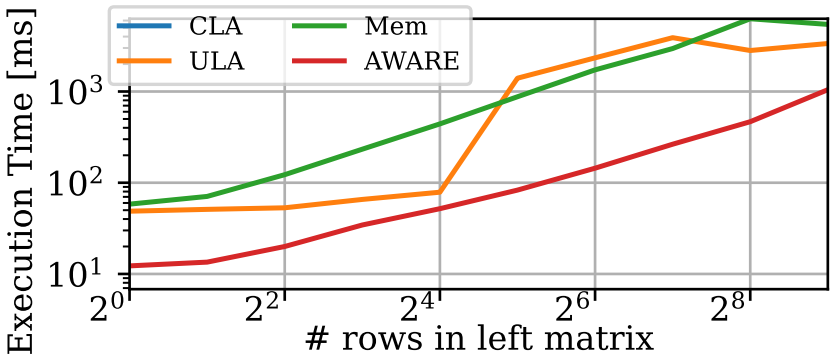


Fig. 3. LMM Census Enc Scaling [Corresponding to original paper's figure 8b]

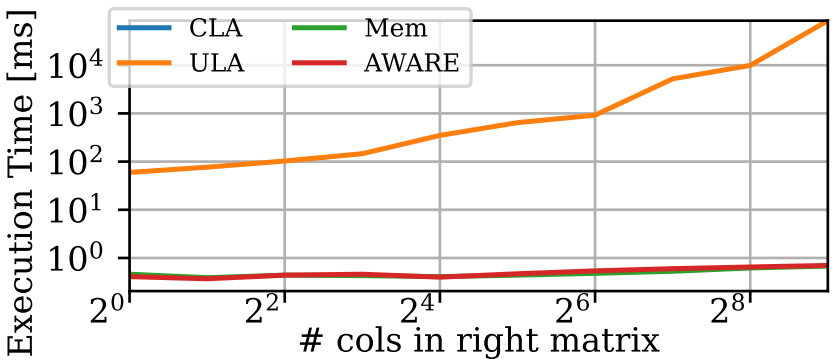
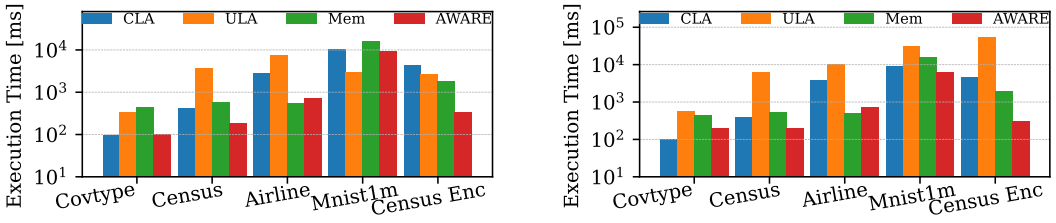


Fig. 4. RMM Census Enc Scaling [Corresponding to original paper's figure 9b]



(a) TSMM Sparse (b) TSMM Dense

Fig. 5. Operations Performance Transpose Self Matrix Multiplication [Corresponding to original paper's figure 10]

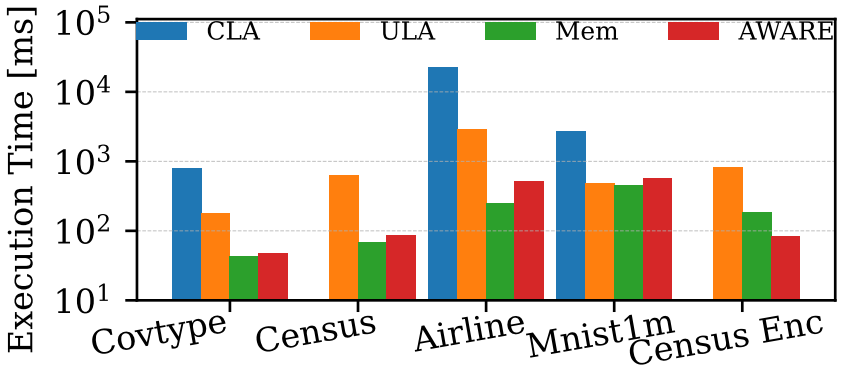


Fig. 6. Euclidean MinDist 16 Points Dense [Corresponding to original paper's figure 11b]

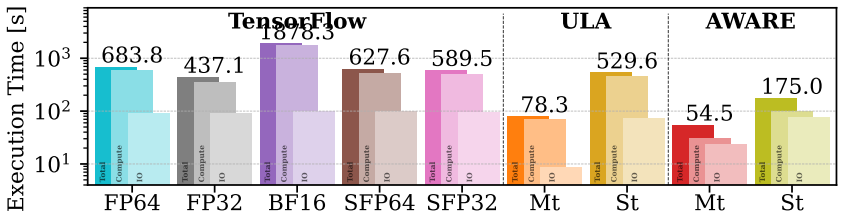


Fig. 7. TensorFlow Comparison [Corresponding to original paper's figure 12]

4 SUMMARY

The major figures and experimental conclusions in the paper have been reproduced, supporting the patterns reported in the paper. The authors are to be commended for the effort that went into preparation and submission of detailed run and plotting scripts.

REFERENCES

[1] Sebastian Baunsgaard and Matthias Boehm. 2023. AWARE: Workload-aware, Redundancy-exploiting Linear Algebra. *Proc. ACM Manag. Data* 1, 1 (2023), 2:1–2:28. https://doi.org/10.1145/3588682

[2] DAMSLab. 2023. Reproducibility Repository for SIGMOD 2023 AWARE p5. https://github.com/damslab/reproducibility/tree/master/sigmod2023-AWARE-p5. Accessed: April 1, 2024.