Appendix: Artifact Description/Artifact Evaluation

Artifact Description (AD)

1 Overview of Contributions and Artifacts

1.1 Paper's Main Contributions

Below are the main contributions of the paper.

C_1 We discover the problems of conflicting Huffman encoder and LZ-like compressor in the original SZ compressor.

C_2 We integrate lpaq, a well-known modeling-based arithmetic encoder into the SZ compressor to see if modeling-based arithmetic coding helps in terms of solving the problem.

C_3 We propose MAC and integrate it into the SZ compressor as described in the paper.

1.2 Computational Artifacts

The artifacts include:

A_1 The standalone test scripts of our approach against the original SZ compressor and other approaches. This artifact also includes experiments related to our motivation of the paper. https://github.com/OceanCT/sc25-macsz-ae

A_2 The hpc test scripts and logs of our hpc test. https://doi.org/10.5281/zenodo.15285482

Artifact ID	Contributions Supported	Related Paper Elements
A_1	C_1, C_2, C_3	Tables 1-2, 4-5 Figures 3-4, 9,10,13
A_2	C_2, C_3	Tables 11,12

2 Artifact Identification

2.1 Computational Artifact A_1

Relation To Contributions

This artifact includes three parts: 1. the source code of LPAQ-SZ, MAC-SZ and other approaches we use in the evaluation step. 2. scripts that help support our motivation 3. scripts that run evaluations in a non-hpc server.

Expected Results

1. Quantization factors should be the majority part of the data after SZ process. 2. Quantization factors should be centralized in a relative small range. 3. LZ4 and Huffman should individually achieve decent compression ratio. Yet combining them in a sequent way should not achieve a significantly higher compression ratio for quantization factors than exploiting Huffman coding alone. 4. MAC-SZ should achieve a superior compression ratio compared to original SZ compressor and a much faster compression and decompression speed than LPAQ-SZ.

Expected Reproduction Time (in Minutes)

The expected computational time of this artifact on Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz is 1800 min.

Artifact Setup (incl. Inputs)

Hardware. A server with a Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz.

Software. The server runs Ubuntu 18.04.1.

Datasets / Inputs. Datasets are downloaded from https://sdrbench.github.io. Our tests use CESM-ATM, EXAALT, Hurricane ISABEL and HACC.

Installation and Deployment. Git clone from https://github.com/OceanCT/sc25-macsz-ae and follow the instructions shown in the readme.

Artifact Execution

- Download all the datasets:
```
cd ./scripts && ./download.sh
```
- Compile all the approaches:
```
cd ./scripts && ./compile.sh
```
- Run the standalone non-hpc tests:
```
cd ./scripts/pw_rel_test && nohup szt_monitor.sh >
↪  monitor_log 2>&1 &
```
- Check if the non-hpc tests are all finished:
```
cd ./scripts/pw_rel_test && python3 progress.py
```
If all the test_flags are "False", tests are all finished.
- Run the sensitivity analysis on prefix configurations:
```
cd ./scripts/prefix_test && python3 test.py
```
This might take about 3-5 hours.

Artifact Analysis (incl. Outputs)

- Sort the results, the output file is "./scripts/visualize/datap/pw_rel.csv", these results support Table 1.
```
cd ./scripts/visualize && python3 preprocess.py
```
- To get Figure 3, run the following command. Results are shown in "./scripts/motivation/qf_ratio.pdf".
```
cd ./scripts/motivation && python3 qfsize.py
```
- To get Figure 4, run the following command. Results are shown in "./scripts/motivation/*_compression_ratios.pdf".
```
cd ./scripts/motivation && python3 lz_huff.py
```
- To get Table 2, run the following command. Results are shown in stdout.
```
cd ./scripts/motivation/qfcal && python3
↪  calculate.py
```
- To get Table 4, run the following command. Results are shown in stdout.
```
cd ./scripts/visualize && python3 qfcr.py
```
- To get Table5, run the following command. Results are shown in stdout.
```
cd ./scripts/visualize && python3 cr.py
```
- To draw Figure 9, run the following command. Results are shown in "./scripts/visualize/pics/CESM-ATM_0.01_qfsize.pdf".
```
cd ./scripts/visualize && python3 qfr.py
```
- To draw Figure 10, run the following command. Results are shown in "./scripts/visualize/pics/ctime.pdf" and "./scripts/visualize/pics/dtime.pdf".

```
cd ./scripts/visualize && python3 ct.py && python3
↪  dt.py
```

- To draw Figure 13, run the following command. Results are shown in "./scripts/prefix_test/prefix.pdf".

```
cd ./scripts/prefix_test && python3 draw.py
```

2.2 Computational Artifact A_2

Relation To Contributions

This artifact includes the test scripts and log in the Tianhe-2 hpc-test. They support the HPC evaluation Figures we give in the paper (Figure 11 and Figure 12).

Expected Results

The overall performance of MAC-SZ gets better when the core number increases. In the 8192-core scenario, MAC-SZ performs better both in terms of compression ratio and in terms of throughput.

Expected Reproduction Time (in Minutes)

The expected computational time of this artifact on Tianhe-2 is 1440 min.

Artifact Setup (incl. Inputs)

Hardware and Software. Tianhe-2 system.

Datasets / Inputs. We use test files from CESM-ATM dataset. They are already included in the zip file.

Installation and Deployment. Download the zip from zenodo and extract it into the Tianhe-2 system.

Artifact Execution

To run an approach with configuration of n cores, go to the corresponding directory, run "sbatch x_*.slurm". For example, to test dumping and loading performance of the original SZ compressor in 8192 cores, go to the "orisz" directory, and run "sbatch orisz_8192.slurm" and "sbatch oriszd_8192.slurm". Logs will be shown in "orisz_8192.log" and "oriszd_8192.log".

Artifact Analysis (incl. Outputs)

Analysis scripts are included in the "https://github.com/OceanCT/sc25-macsz-ae". To draw Figure 11 and Figure 12, run the following command. Results are shown in "scripts/visualize/pics/hpc/".

```
cd ./scripts/visualize && python3 hpc.py
```