Using Lossy Compression for Interactive Analysis Over Network

Rei Aoyagi[1], Keichi Takahashi[1,2], Yoichi Shimomura[2], and Hiroyuki Takizawa[1,2]
1 Graduate School of Information Sciences, Tohoku University
2 Cyberscience Center, Tohoku University
rei.aoyagi.r8@dc.tohoku.ac.jp

Introduction

- To gain scientific insights from simulation results stored in High performance computing (HPC) centers, researchers conduct post-processing by using:
  1. Batch jobs managed by job schedulers such as Slurm.
  2. **Interactive data analysis tools such as Jupyter Notebook**
- In certain scenarios, transferring the simulation results directly from the center is essential.
  - e.g., the cases where special hardware and/or licensed software is available only on a particular system remote from the HPC center.
- To achieve interactive data processing, only a necessary part of the data could be streamed over the network. Under this assumption, there are two challenges for achieving interactivity:
  1. **Limited network bandwidth.**
  2. **Long network latency.**

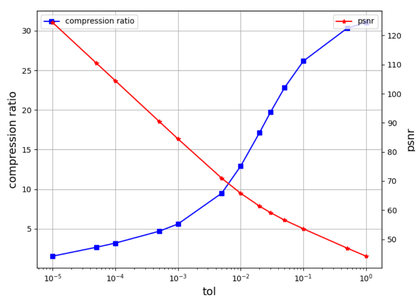
Error-bounded Lossy Compression

- For certain data analyses that allow degradation in quality, error-bounded lossy compression can be applied.
- Features of error-bounded lossy compression are:
  1. A higher compression ratio can be achieved for floating-point data by allowing a larger error.
  2. Acceptable error by lossy compression can be adjusted by users.

$$PSNR = 10\ log\frac{MAX_I^2}{MSE}$$

MAX_I : *Maximum value of the data*

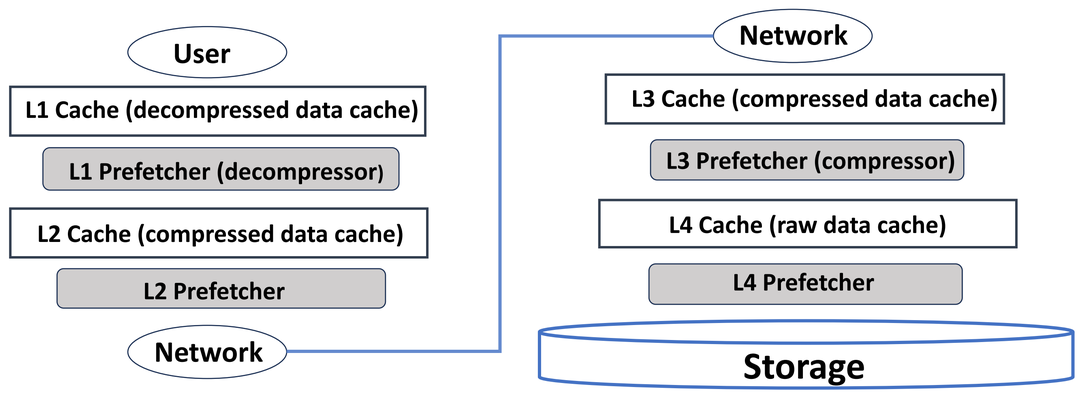
This work proposes a mechanism to properly use error-bounded lossy compression that can find a good trade-off point.



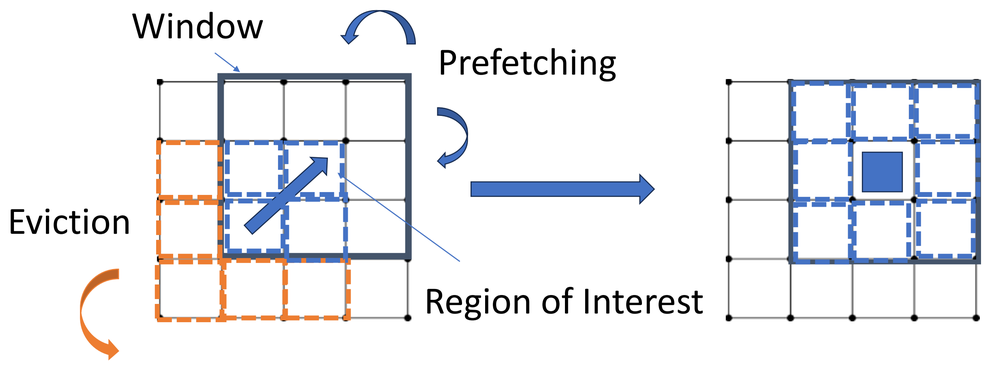
Proposed Middleware

We propose middleware with the following features to support interactive array analysis over network.

1. Using **error-bounded lossy compression** to increase the effective network bandwidth.
2. Using **multi-level caching and prefetching** to hide the network latency.
   - At each step of data transfer from server to client (reading, compressing, transferring over network and decompressing), we introduce:
     1. Caches to take advantage of data access locality.
     2. Prefetchers to improve the cache hit ratio.



- Considering that the access pattern of interactive analysis on a multidimensional array exhibits a spatial locality,
  1. **Prefetcher at each level keeps fetching the blocks around the last accessed block until the cache becomes full.**
  2. **Each cache evicts blocks located farther than a certain distance from the last accessed block.**



Evaluation

Evaluation settings

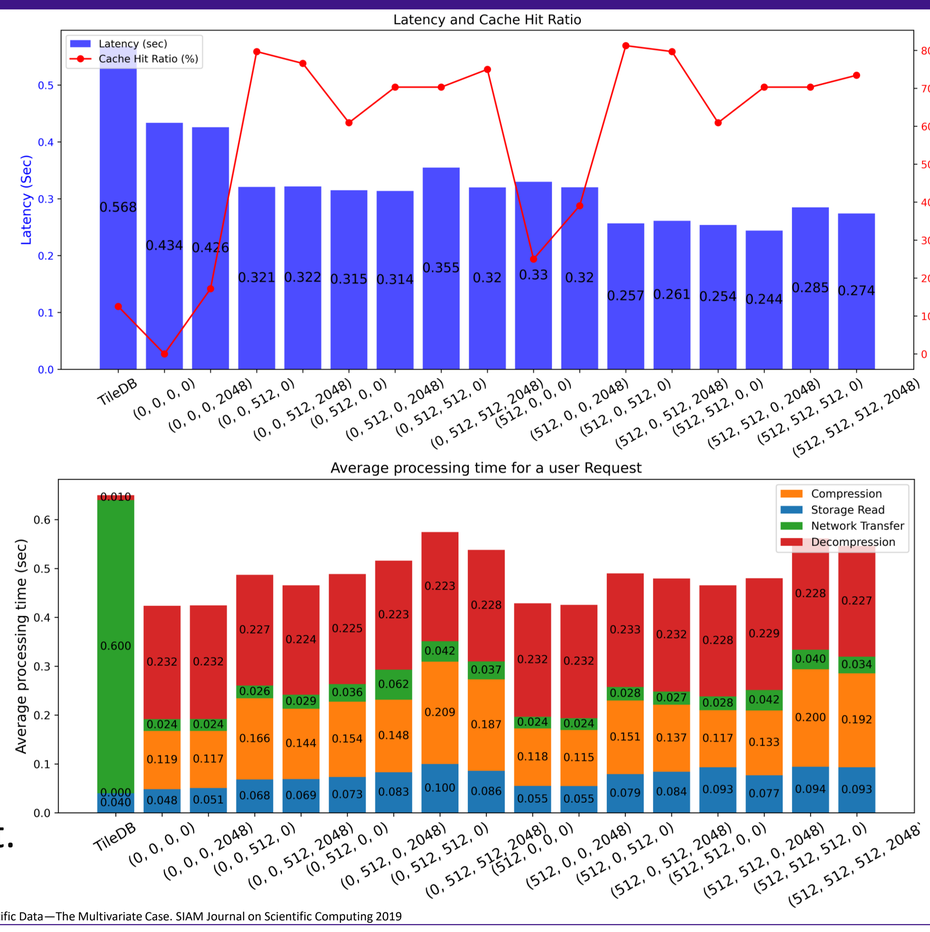
- L1, L2, and L3 cache sizes were set to either 0 MiB or 512 MiB. L4 cache size was set to either 0 MiB or 2048 MiB. The cache size of 0 represents that the cache is turned off and unused.
- Turbulence Flow simulation data [1] is used for the data to analyze.
- Each user requests retrieves a 64MiB block of 3-dimensional array. 64 Requests in total are sent at the interval of 1 second to simulate user's interactive data analysis.
- Error tolerance is set to 10% of the original data value.
- The proposed middleware is compared with TileDB [2], a state-of-the-art array databases with lossless compression and Least-Recently-Used cache replacement policy (no prefetching).

Evaluation metrics

- The average latency from the time the user request a block to obtaining the block.

Implementation details

- TileDB is used for array management in storage, MGARD [3] for error-bounded lossy compression and Python's standard library for key-value store of cache management.
- The server and client communicate over HTTP.



[1] Johns hopkins turbulence databases. https://turbulence.pha.jhu.edu/
[2] S. Papadopoulos, K. Datta, S. Madden, and T. Mattson. 2016. The TileDB Array Data Storage Manager. 2016
[3] M. Ainsworth, O. Tugluk, B. Whitney, and S. Klasky. 2019. Multilevel Techniques for Compression and Reduction of Scientific Data—The Multivariate Case. SIAM Journal on Scientific Computing 2019

Conclusion

- Compared to TileDB, our proposal reduced the average latency by up to **57%** because
  1. Introducing **lossy compression reduced the average network transfer time** from **0.6 sec to 0.042 sec.**
  2. Introducing **multi-level caching and prefetching improved the cache hit ratio** from **12.5% to 70.3%.**
- In the future work, machine learning will be applied to selecting blocks to be prefetched to improve the cache hit ratio.

HPC Asia 2024

This work was partially supported by JSPS KAKENHI Grant Number 20K19808.

URL https://www.hpc.is.tohoku.ac.jp/home-en
Email rei.aoyagi.r8@dc.tohoku.ac.jp