Ocelot: Distributed Lossy Compression and Data Transfer System

Yuanjian Liu* Sheng Di*† Kyle Chard*† Ian Foster*†

*Department of Computer Science, University of Chicago

†Argonne National Laboratory

Overview of Lossy Compression

Lossy compression [1, 2, 3] is used for reducing data sizes by a significant amount, the compression ratio of which ranges from tens to hundreds. Modern scientific simulations can produce extraordinary volumes of data. For example, climate and weather simulations can produce terabytes of data in a matter of seconds, and the Hardware/Hybrid Accelerated Cosmology (HACC) simulation code can generate petabytes of data from a single run. Fortunately, in many cases, consuming applications (e.g., for analysis, visualization, and machine learning) can achieve acceptable performance on reduced-precision data. Most often, We just need to ensure each decompressed data value is within a pre-defined error-bound (e.g. 0.001).

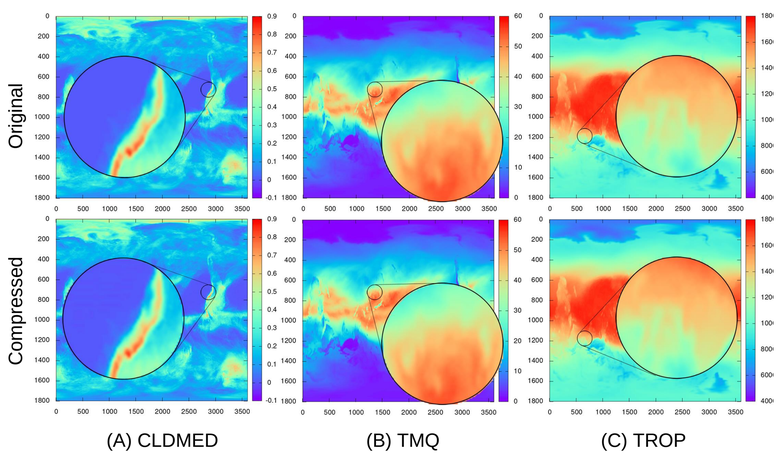

Figure 1. CESM data visualization comparison between original and compressed data: When setting an error bound of 0.001, the compression ratios are 12.51, 6.79, 7.42; the PSNRs are 59.64, 96.80, and 146.05 respectively. There is no obvious visual difference between the original and compressed data.

Figure 1 shows that when each data point has a disparity less than 0.001, the lossy compression ratio is higher than the lossless compression ratio (which is usually lower than 2), while there is no visually obvious data loss.

Challenges

Challenges for distributed lossy compression include (but are not limited to):

1. **Datasets can locate in multiple clusters**. The users may need to log in to each machine individually to access the data.
2. **Command line interface is not straightforward for setting error bounds**. Users usually use SSH to log in to the computing clusters and run (de)compression commands via text-based terminals. For more complicated error-bound settings, this can be too troublesome and indirect.
3. **Inconvenient for data transferring**. After compression, users often need to transfer the data to other machines for decompression and further analysis. There is no direct way to compose the compress-transfer-decompress workflow.
4. **Execution time estimation is hard for SLURM jobs**. Users need to submit batch jobs to compress large files on SLURM systems, but there is no good way to estimate the time for compression. If they set a longer amount of time, the job can be scheduled much slower than a shorter one.

In addition to these challenges, selecting the most suitable compressor can be hard. It is important to predict the performance to make a reasonable choice.

Function-as-a-service (FaaS) & Globus Compute

Function-as-a-Service (FaaS) is a cloud computing model that allows developers to write and execute code on the edge without managing infrastructure. FaaS uses serverless architecture to execute small pieces of code called functions.

Globus Compute works like other FaaS platforms: users first register a function with Globus Compute by specifying the function body (in Python), they may then execute that function by specifying the function ID and input arguments. Unlike traditional FaaS platforms, users also specify the endpoint ID on which they wish to execute the function. Figure 2 illustrates the procedure of running functions via Globus Compute.

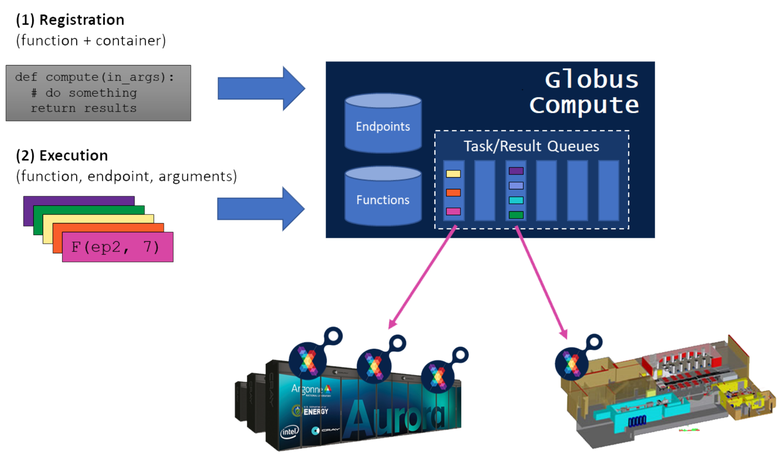

Figure 2. **Globus Compute** efficiently manages connected resources by elastically deploying workers and containers across nodes and can execute millions of functions across thousands of nodes.

Ocelot Architecture

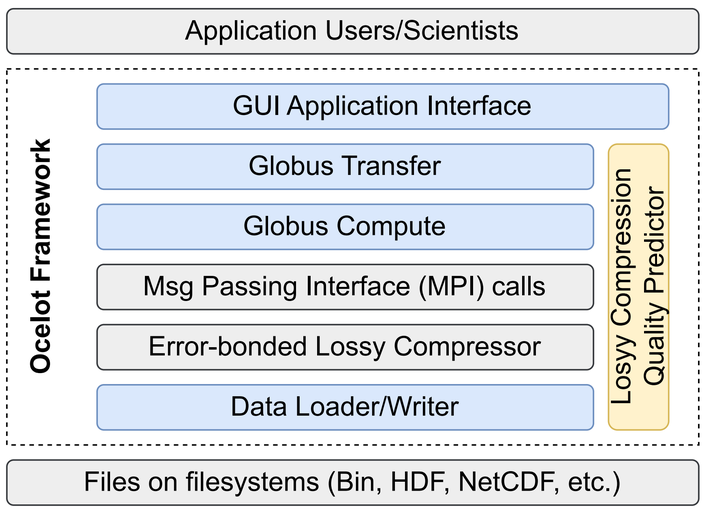

Figure 3. Ocelot system architecture

Ocelot [4] is our distributed compression and transfer framework. It is actively under development and supports controlling (de)compression on multiple clusters or personal computers with one interface. Its *Globus Compute* and *Globus Transfer* foundation enable *Ocelot* with its core strengths which include (but are not limited to):

- **Managing (de)compression & transfer tasks in one place**. *Ocelot* offers a single Qt-based GUI interface for compression, transfer, and decompression.
- **Easy configuration for multiple error bounds**. Ocelot makes range and region selection intuitive by offering a data preview window where users can select multiple regions or ranges to set different error bounds.
- **Compression performance prediction capability**. Ocelot utilizes benchmarking information and previous (de)compression/transfer experiments to predict compression and transfer performance.

Multi-Region & Multi-Range Compression

We proposed a multi-region and multi-range compression method earlier[3] for SZ3[1], but it was unintuitive to set the error bounds. We now improve it by adding a graphical selection window as shown in Figure 4 to set error bounds easily.

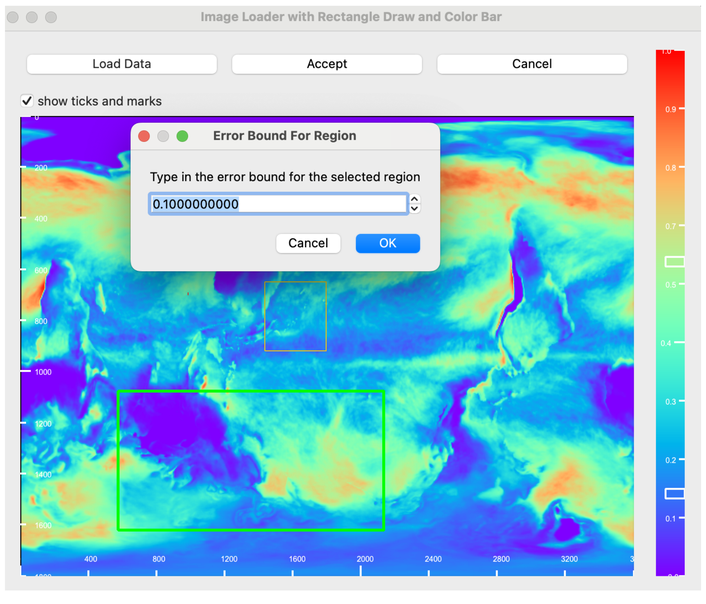

Figure 4. Region/Range-based compression with multiple error bounds: the graphical interface allows users to preview the visualization of the data and draw rectangles on the preview to set different error bounds for the regions. It also maps the value range to a color bar and users can set different error bounds for different ranges.

Experimental Setup

To demonstrate the compression performance prediction capability, we set different error bounds for each dataset and compare the predicted and measured metrics. We mainly compare the compression ratio (CR), peak signal-to-noise ratio (PSNR), and (de)compression time (CP/DPTime).

We conducted our experiments on LCRC Bebop, NERSC Cori, and Purdue Anvil computing clusters. We compare the overall time of compression, transfer, and decompression workflow against direct transfer to see if there is performance gain to involve compression. The datasets we used for experiments are listed in Table 1.

Table 1. Datasets

Application	Dimension	Science
Miranda	256x384x384	large turbulence simulations
CESM	1800x3600	climate, cloud
ISABEL	100x500x500	hurricane simulation
QMCPACK	33120x69x69	electronic structures
RTM	449x449x235	electronic
NYX	512x512x512	cosmology

Evaluation Results

Key Takeaways. To estimate compression time, benchmarking information is enough. The compression ratio can be predicted accurately after classifying the data into the correct application. Range and region-based methods improve the data quality in the interesting area and achieve a higher compression ratio by applying larger error bounds on other areas. The overall transfer time is improved when involving parallel compression.

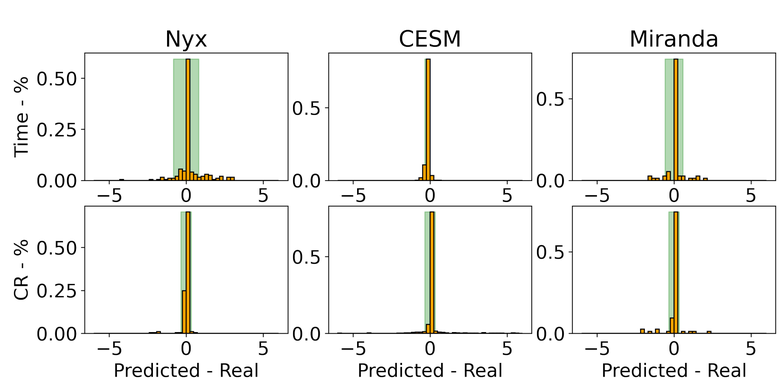

Figure 5. Nyx/CESM/Miranda application compression time and ratio prediction error distribution (measured on Bebop KNL partition): the X-axis is the difference between the predicted value and the real value, the Y-axis is the percentage for each small range of difference values.

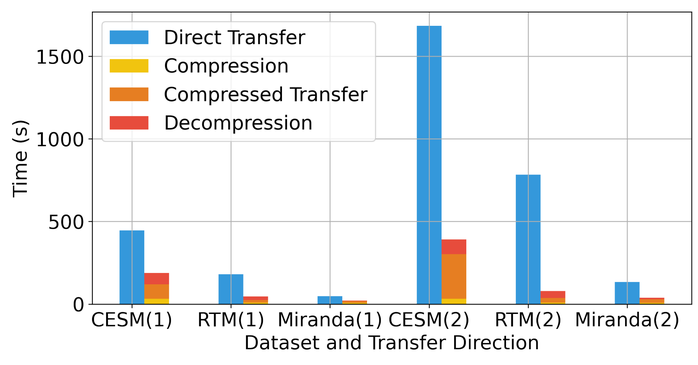

Figure 6. Transfer time comparison between direct transfer and transfer with compression. (1) means transferring from Purdue Anvil to NERSC Cori, (2) means transferring from Purdue Anvil to Argonne Bebop.

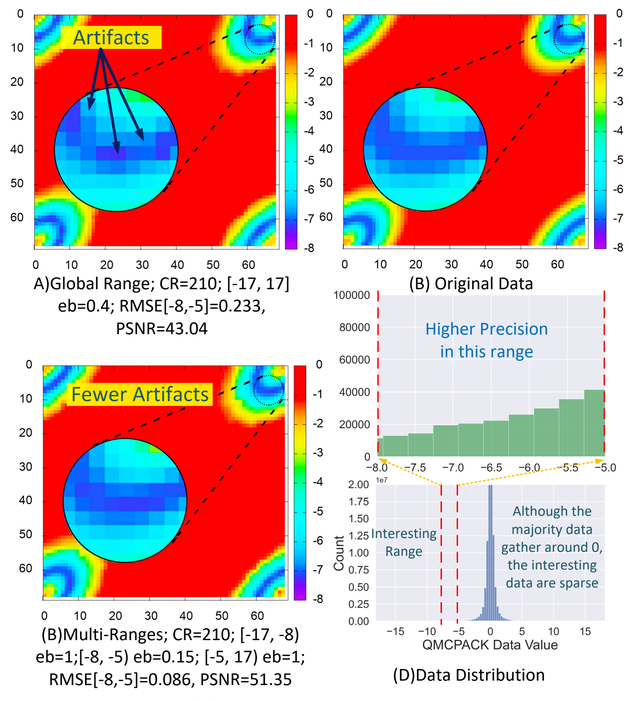

Figure 7. QMCPACK data: (A) The basic method is setting one error bound for the global range. (C) The interesting range [-8,-5) has a tighter error bound of 0.15 while leaving other ranges a higher error bound of 1.

Acknowledgement

This research was supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research (ASCR), under contract DE-AC02-06CH11357, and supported by the National Science Foundation (NSF) under Grant OAC-2003709, OAC-2104023, and OAC-2311875. We acknowledge the computing resources provided on Bebop (operated by Laboratory Computing Resource Center at Argonne) and Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) funded by NSF Grant #2138259, #2138286, #2138307, #2137603 and #2138296.

References

[1] Xin Liang, Kai Zhao, Sheng Di, Sihuan Li, Robert Underwood, Ali M. Gok, Jiannan Tian, Junjing Deng, Jon C. Calhoun, Dingwen Tao, Zizhong Chen, and Franck Cappello. Sz3: A modular framework for composing prediction-based error-bounded lossy compressors. *IEEE Transactions on Big Data*, pages 1–14, 2022. doi:10.1109/TBDATA.2022.3201176.

[2] Peter Lindstrom. Fixed-rate compressed floating-point arrays. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2674–2683, 2014.

[3] Yuanjian Liu, Sheng Di, Kai Zhao, Sian Jin, Cheng Wang, Kyle Chard, Dingwen Tao, Ian Foster, and Franck Cappello. Optimizing multi-range based error-bounded lossy compression for scientific datasets. In *2021 IEEE 28th International Conference on High Performance Computing, Data, and Analytics (HiPC)*, pages 394–399, 2021. doi:10.1109/HiPC53243.2021.00036.

[4] Yuanjian Liu, Sheng Di, Kyle Chard, Ian Foster, and Franck Cappello. Optimizing scientific data transfer on globus with error-bounded lossy compression. In *2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS)*, pages 703–713, 2023. doi:10.1109/ICDCS57875.2023.00064.