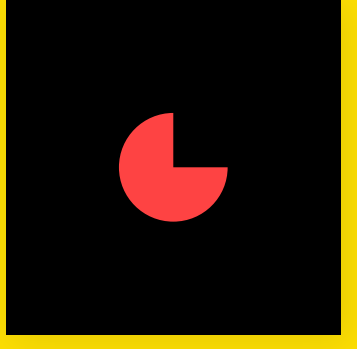


Ocelot

Optimizing Scientific Data Transfer on Globus with Error-bounded Lossy Compression

Authors: Yuanjian Liu, Sheng Di, Kyle Chard, Ian Foster, Franck Cappello

Presenter: Yuanjian Liu

Argonne NATIONAL LABORATORY

THE UNIVERSITY OF CHICAGO

globus labs

Table of Contents

PART 01

Motivation

Why do we need **Ocelot** in HPC systems? How can it improve current research pipelines?

PART 02

Background

What is **lossy compression** and what **challenges** do we face?

PART 03

Method

We focus on two parts: (1) **predict** compression performance; (2) **reduce** the overall transfer time.

PART 04

Evaluation

We show that our prediction method can accurately predict the compression time/ratio, PSNR, and Ocelot accelerates large dataset transfer.

PART 05

Conclusion

Ocelot can reduce the overall transfer time for large datasets by applying lossy compression.

PART 01

Motivation

Motivation

WHY DO WE NEED OCELOT?

PART 01

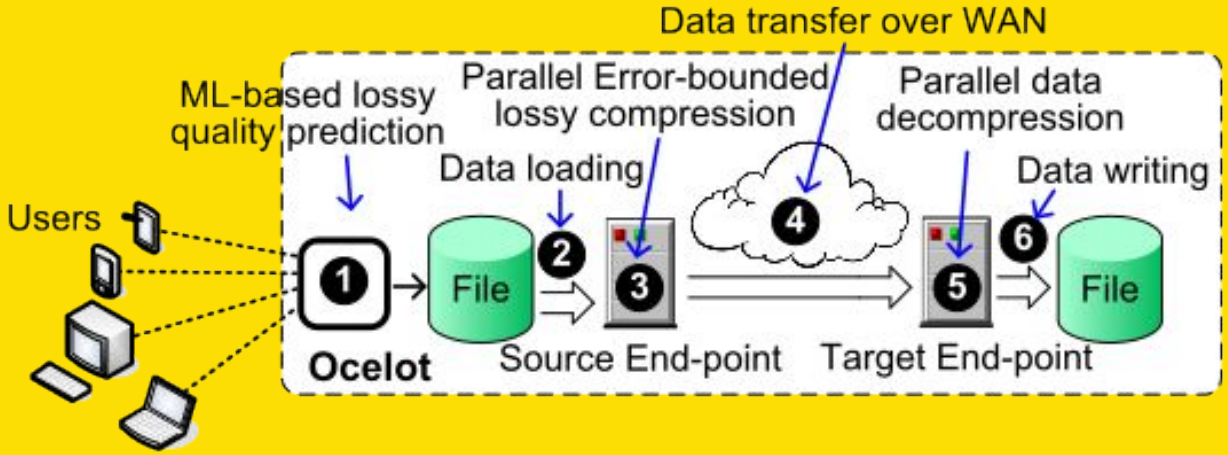
Lossy Compression PLUS Globus Transfer

Large amounts of data are produced by High Performance Computing (HPC) applications. They need to be transferred among multiple sites for analysis. Globus Transfer is widely used to address this need.

However, there was not an effective way to integrate lossy compression and Globus Transfer to maximize the transfer performance. Ocelot is proposed as a new framework for transferring data with lossy compression.

Motivation

WHY DO WE NEED OCELOT?



Improve Research Pipelines

When involving compression into transfer, there are always three main parts for a task: (1) compression; (2) transfer; (3) decompression.

Compression can reduce file sizes for a faster transfer, but it also requires computation resources. There is a trade-off between compression and transfer. We need a mechanism to ensure the integration of compression can improve performance instead of weaken it.

Moreover, a bad compression setting can result in unusable data in lossy compression. We need a way to figure out what setting is suitable.

PART 01

PART 02

Background

Lossy Compression

SZ, SZ2, SZ3, MGARD, ZFP, TTHRESH, etc.

There are mainly two types of lossy compression algorithms: (1) prediction-based methods; (2) transform-based methods. Our method applies to the first type of lossy compression.

Data Transfer

SCP, Globus Transfer

Globus Transfer applies GridFTP to provide more reliable, secure, and higher-performance data transfer. There was not any lossy compression involved in the transfer pipeline.

Compression Performance Prediction

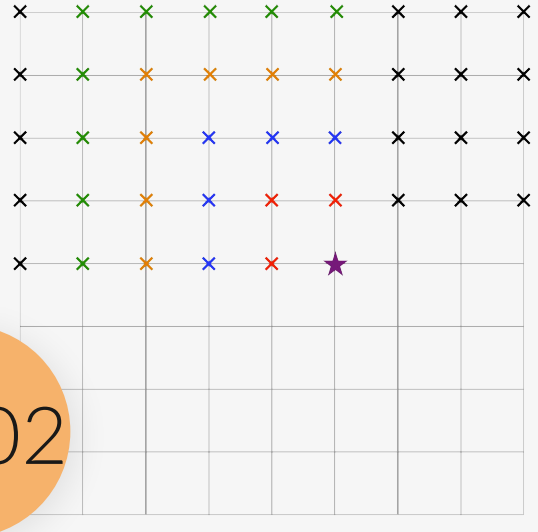
Sian 2022

The prediction is based on derived values of sampled quantization bins' distribution and a fixed formula. It is only correct on very limited datasets, and the fixed formula limits the generalization possibilities.

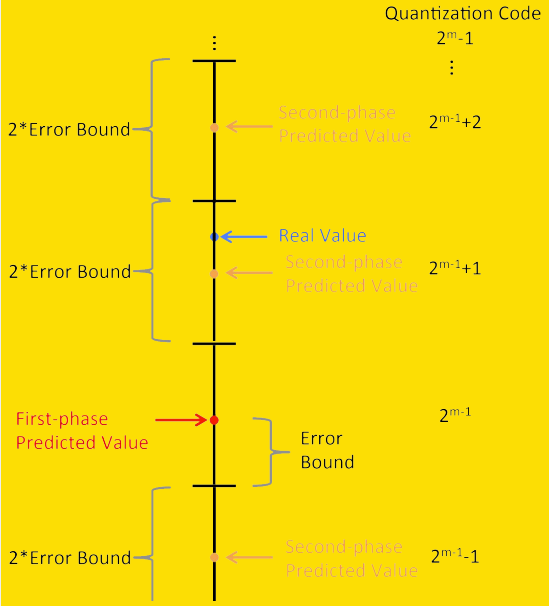
SZ3 Compression

Predicting
Compression
Performance

Prediction



Quantization



Huffman Encoding

The quantization code can be significantly compressed by huffman encoding if the prediction stage is accurate.

Lossless Compression

Lossless algorithms such like run-length, or more complicated ones like zstd, bzip can further compress the data.

PART 02

Challenges

Ocelot faces 3 main difficulties

1 Valid Compression

Lossy is a problem

The compression is lossy, and we need a proper setting to ensure the data is still usable after compression. We hope the program can automatically configure a valid setting based on compression quality prediction.

2 Overhead

Prediction has overheads

The most accurate way to predict the compression performance is to run the compression, but we need a faster way to obtain an approximate result.

3 Resources

Compute nodes may not be available immediately

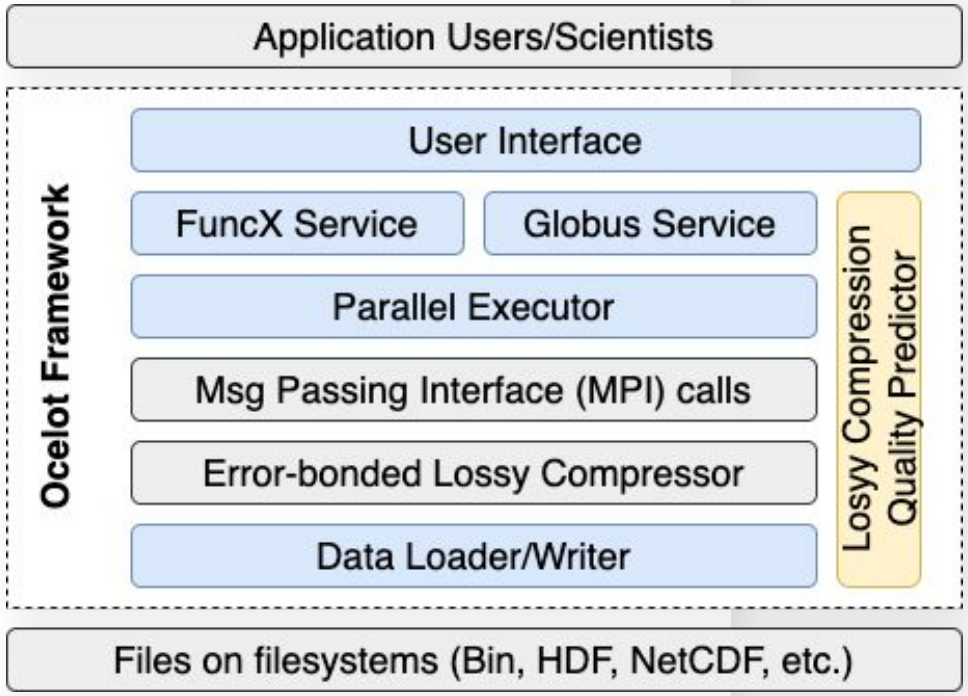
We need compute nodes to perform parallel compression, but there might not be enough idle nodes in a shared system.

PART 02

PART 03

Method

Architecture

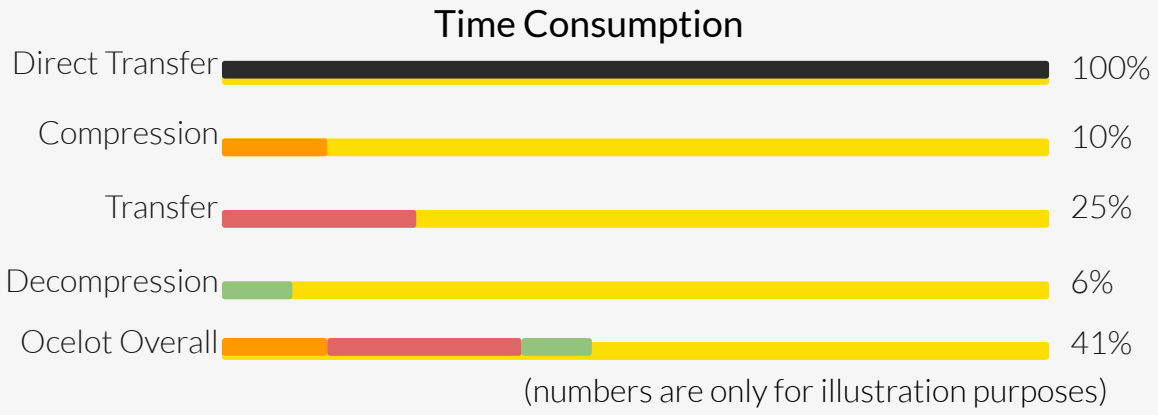


PART 03

Modules In Ocelot

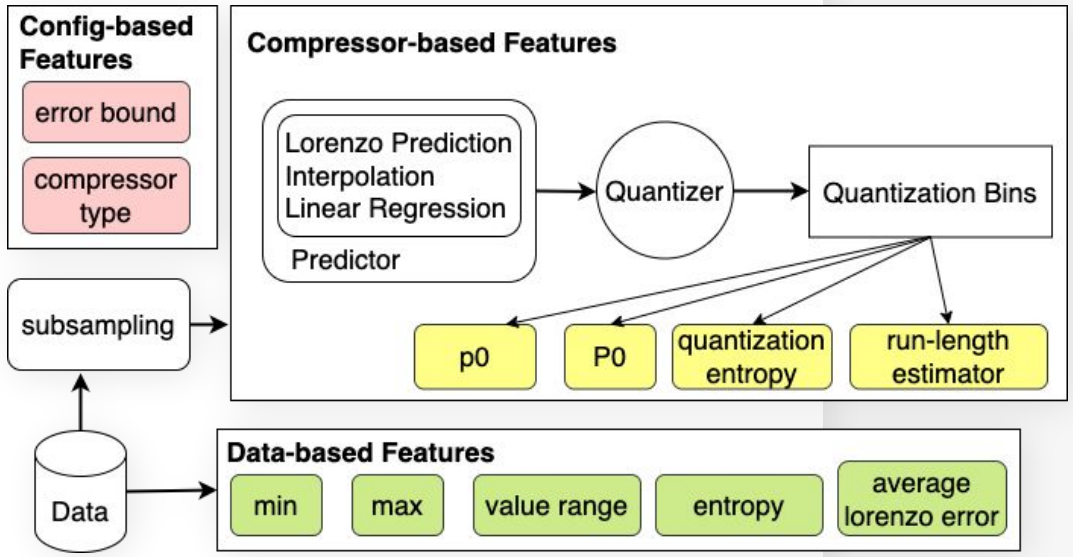
Ocelot is a complete framework for data loading, compression, transfer, decompression, and compression quality prediction.

Ocelot is good at compressing datasets with many files by utilizing parallel compression to reduce the total file sizes in a short amount of time.



(numbers are only for illustration purposes)

Quality Predictor



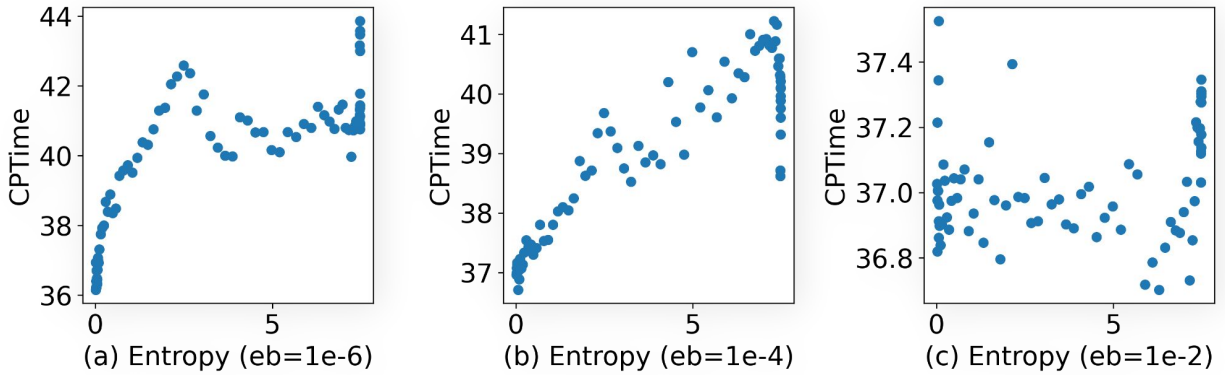
PART 03

Feature Extraction

We use features from 3 places: (1) config, (2) compressor, (3) data. Compared to previous work, we use more features to capture the characteristic of both data and compression algorithm.

Instead of proposing a formula to relate these features, we use a machine learning model to find the hidden formula, which is more generalizable to new datasets.

Moreover, the compressor-based features need to be extracted after the quantization stage, which can be time consuming. We use a sampling method to capture the features of a sample of the whole dataset to reduce the prediction overhead.

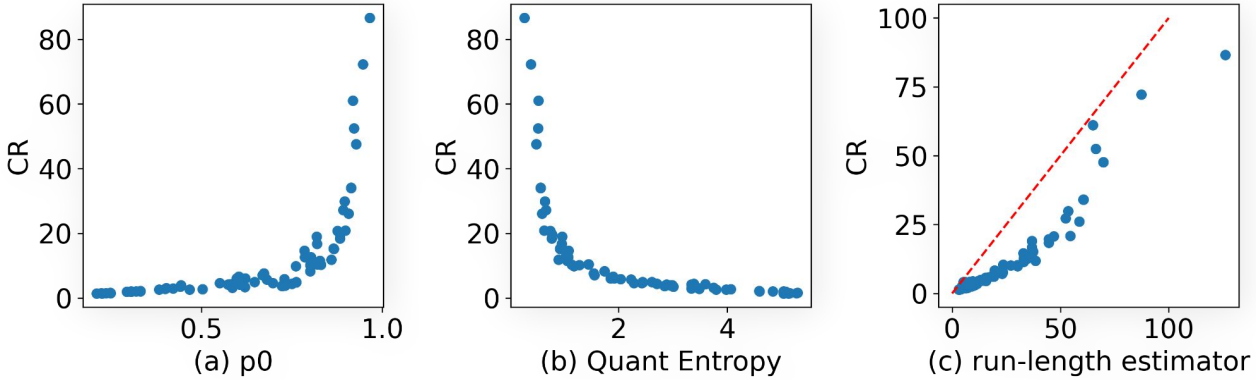


Entropy vs Compression Time

The error bound will affect the predictability of certain features. We use an ML model to handle the complex relationships between the features and the target.

Compressor Features vs Compression Ratio

The red dotted line is the predicted compression ratio using the previous method (Sian). Because the formula is fixed, it fails to cover many datasets that do not follow that simple formula.



PART 03

I/O Optimization

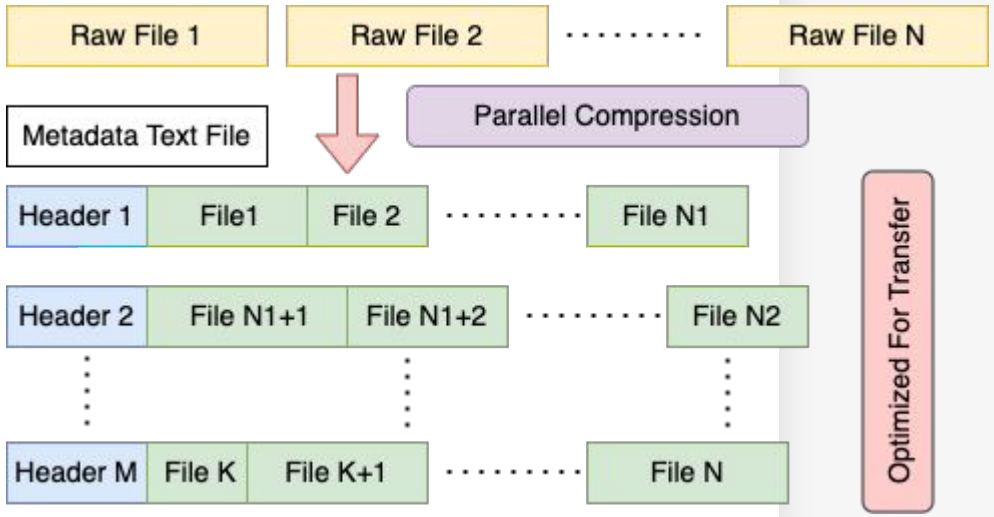


TABLE II
FILE TRANSFER PATTERNS BETWEEN TWO SUPERCOMPUTERS: NERSC CORI AND ARGONNE BEBOP

Total size	File size	# Files	Speed (MB/s)	Duration (s)
300GB	1M	300000	247.0	1235
300GB	10M	30000	921.1	325
300GB	100M	3000	1120.0	267
300GB	1000M	300	1060.0	281

PART 03

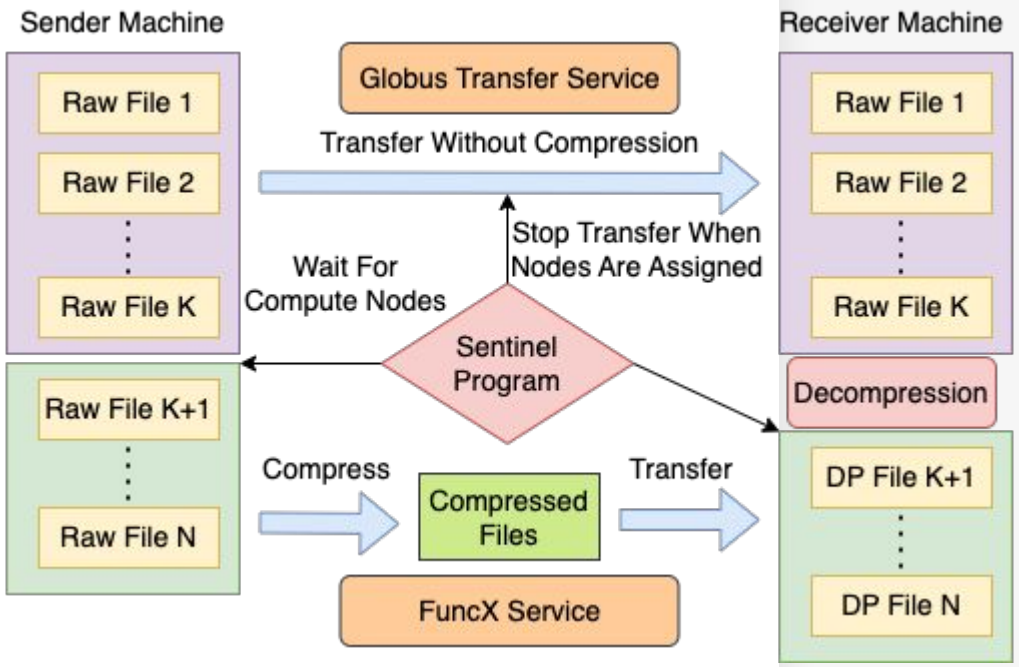
File Reorganization

Globus Transfer uses multiple threads for data transfer. Transferring one extremely large file or too many very small files will both be slow, as can be verified through Table II. When compressing thousands of files in a dataset, we need to reorganize the files to reach a better transfer performance.

Our approach is to connect small files together into a large file. The number of files and each single file size can be configured to match the optimal performance for different sites.

Each raw file is independent to each other, and therefore we can utilize parallel compression to handle this. The overhead for file reorganization is negligible.

Resources Waiting



Don’t Wait!

In shared systems, the compute nodes are not always available immediately when a transfer task initiates. If we wait for the compression task to complete, it may stuck at waiting for compute nodes. Therefore, we should directly start transferring a portion of the files without compression if there are no compute nodes available.

Ocelot has a sentinel program monitoring the status of the compute node waiting and the transfer service. It will start the transfer service if the compute node is not immediately available. Once the compute node is assigned, the sentinel program will kill the transfer service and allow the rest files to be compressed before transferring.

PART 03

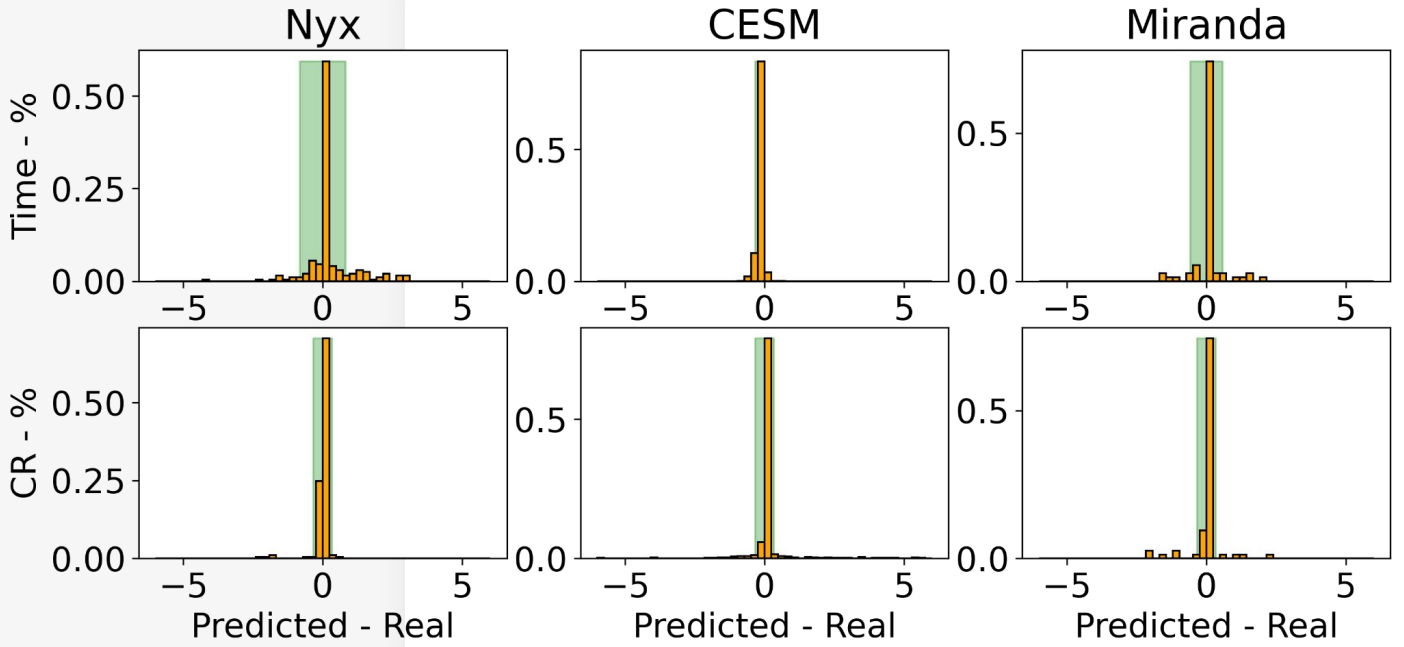
PART 04

Evaluation

PART 04

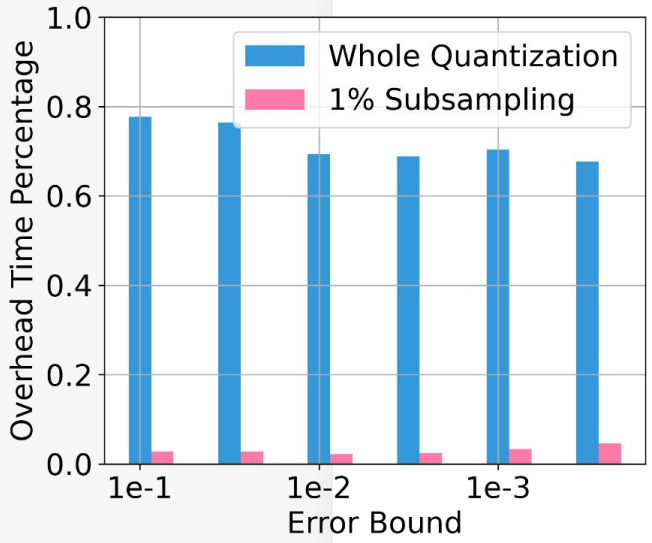
Performance Prediction

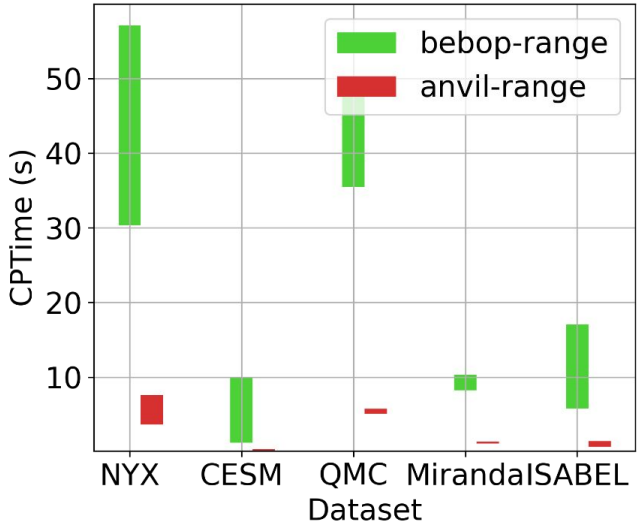
Compression Time | Ratio; PSNR



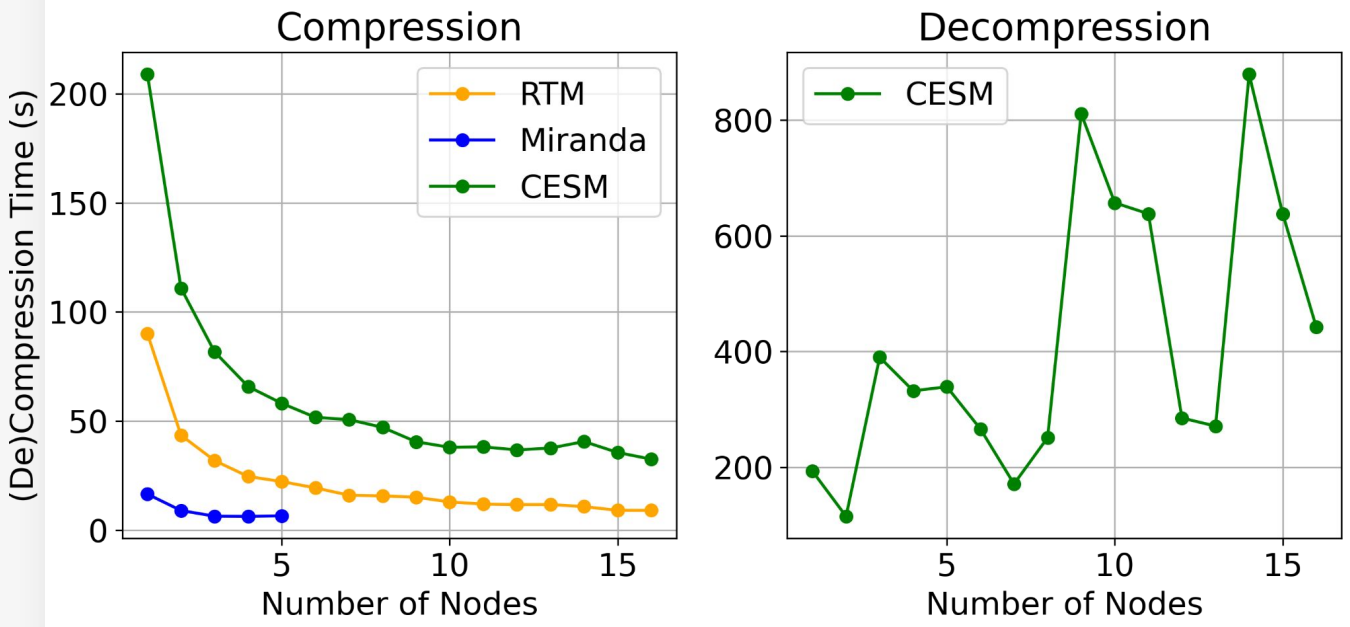
The compression time/ratio prediction error is quite small and is in an acceptable range. The predicted PSNR has a higher error rate in some datasets but is still close enough in most cases.

The prediction overhead is small because we use 1% sampling. Using the whole quantization can slightly improve the prediction accuracy but the overhead would be too huge, taking more than 60% of the whole compression time.





PART 04



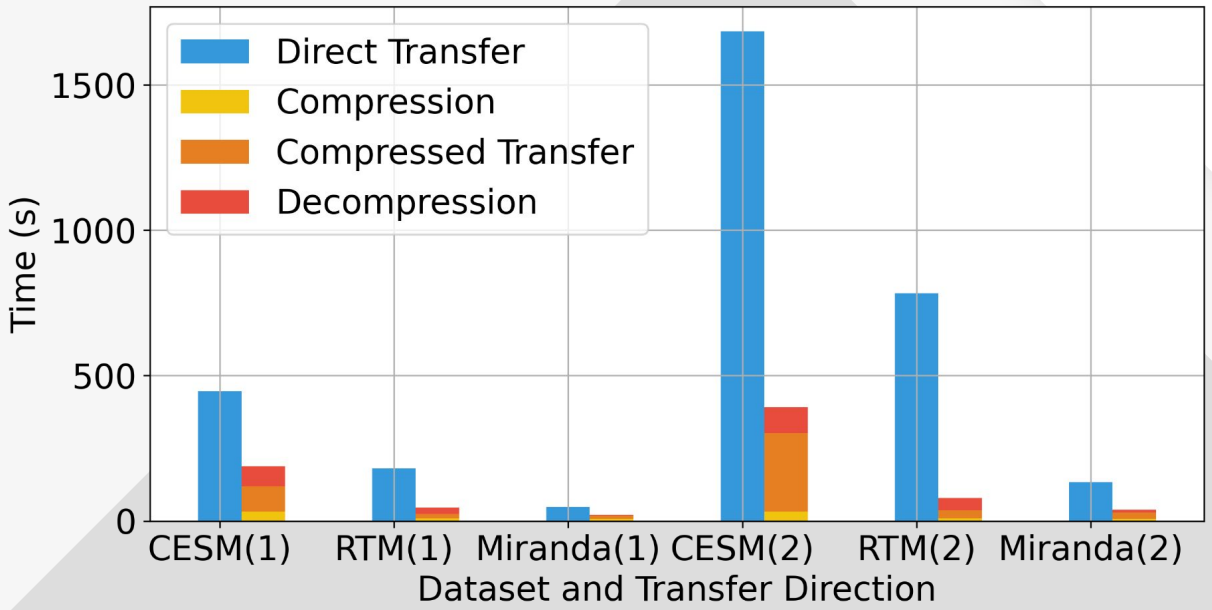
Parallel Compression

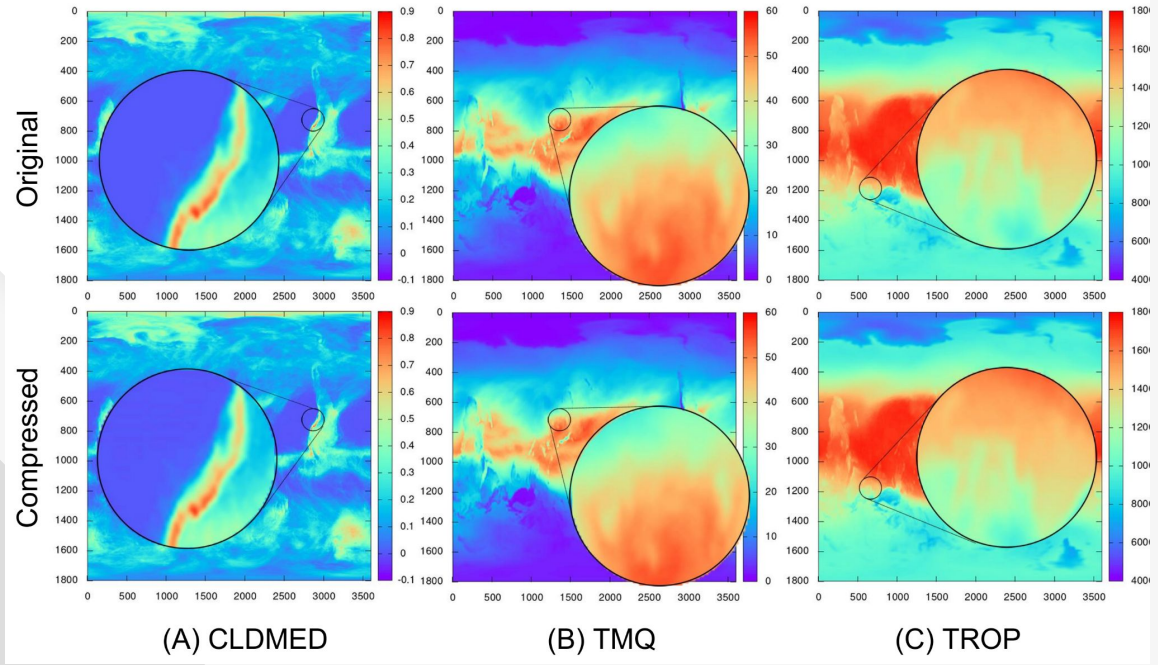
Compression performance can be hugely improved when given more compute nodes because data files are independent to each other, and the compressed files are small to write to disk. Decompression time will not necessarily decrease when given more compute nodes because writing so many large files simultaneously can cause I/O contention.

PART 04

Overall Improvement

We tested data transfer with Ocelot on three sites: Purdue Anvil, Argonne Bebop, and NERSC Cori. They have different network and compute node conditions. The overall transfer performance with lossy compression is hugely better than direct transfer, and the data quality is very good.



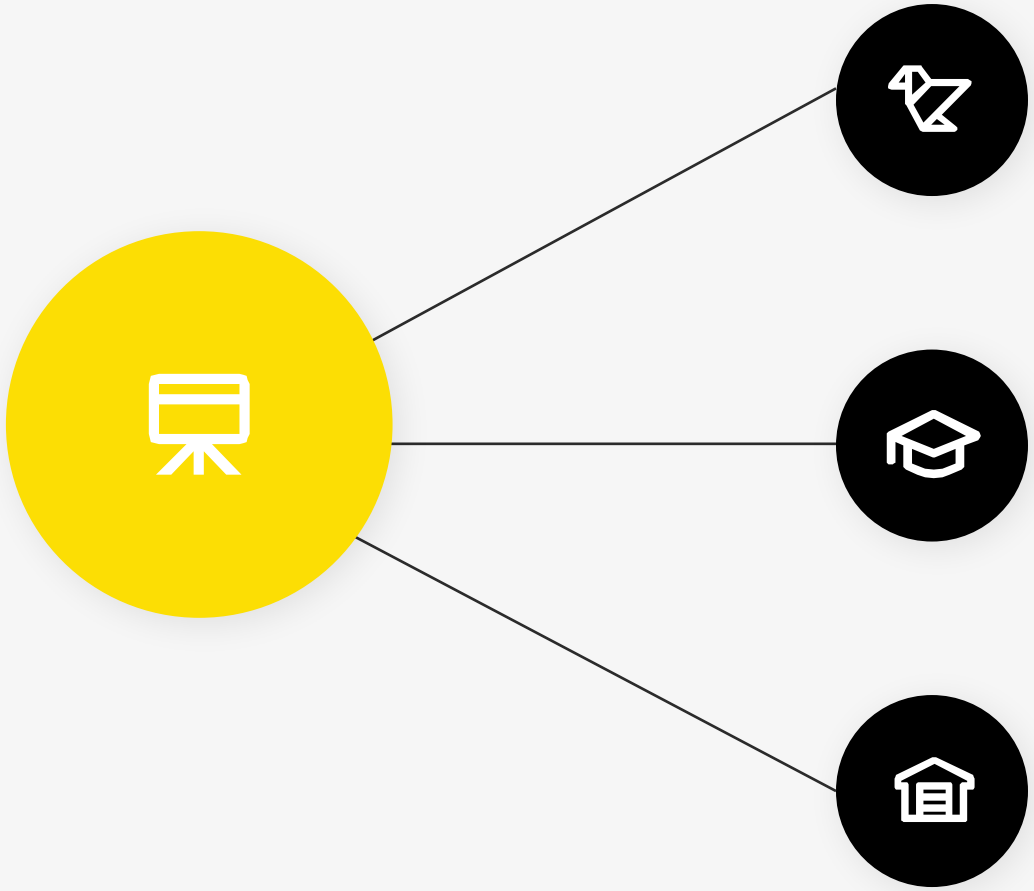


PART 05

Conclusion & Future Works

Main Contributions

What can Ocelot do?



Performance Prediction

Ocelot can accurately predict the compression time/ratio and data quality (PSNR) with very small cost (by 1 % sampling). This ability helps automatically set an appropriate compression setting.

Data Transfer Improvement

Scientific data transfer can be greatly improved by applying lossy compression. File grouping and no-wait procedure helps improve network transfer efficiency.

Parallel Compression Discovery

The compression stage and decompression stage require different levels of parallelism to reach maximum performance.

Future Work

What to do next?

Future Researches

Prediction for more types of compressors

Ocelot currently is only capable of predicting compression performance of SZ3 and its variants. There are other transform-based compressors like ZFP, T-THRESH that can be further studied to predict their performance.

Other Features

We manually extracted a few features to predict the performance. There can be more features to explore, and methods without feature extraction to do the work.

PART 05

Acknowledgement

The material 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 under Grant OAC-2003709 and OAC-2104023. We acknowledge the computing resources provided on Bebop (operated by Laboratory Computing Resource Center at Argonne).

Argonne NATIONAL LABORATORY

THE UNIVERSITY OF CHICAGO

globus labs

THANKS!

Optimizing Scientific Data Transfer on Globus with Error-bounded Lossy Compression

Authors: Yuanjian Liu, Sheng Di, Kyle Chard, Ian Foster, Franck Cappello

Presenter: Yuanjian Liu

Argonne NATIONAL LABORATORY

THE UNIVERSITY OF CHICAGO

globus labs