Accelerating Parallel Write via Deeply Integrating Predictive Lossy Compression with HDF5

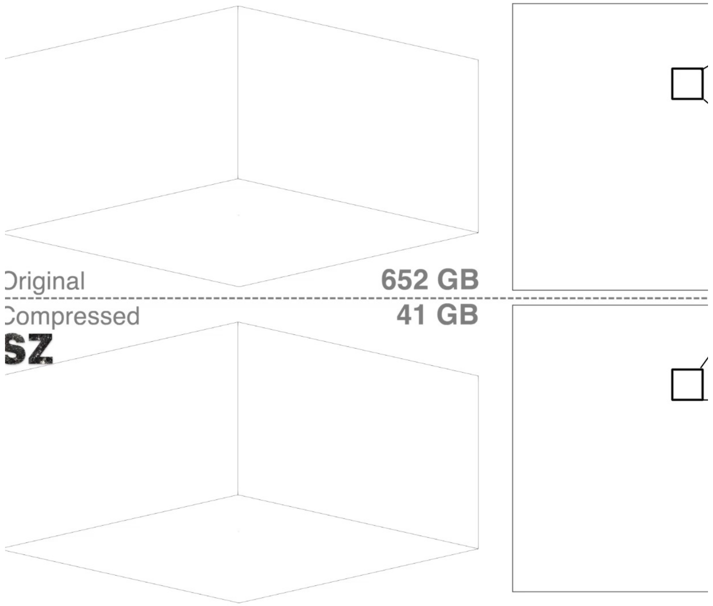
Sian Jin⋆, Dingwen Tao⋆, Houjun Tang‡, Sheng Di†, Suren Byna‡, Zarija Lukic‡, Franck Cappello†
⋆Luddy School of Computing, Informatics, and Engineering, Indiana University, Bloomington, IN, USA
†Mathematics and Computer Science Division, Argonne National Laboratory, Lemont, IL, USA
‡Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley, CA, USA

INDIANA UNIVERSITY BLOOMINGTON
Argonne NATIONAL LABORATORY
BERKELEY LAB Bringing Science Solutions to the World
SC22 Dallas, TX hpc accelerates.

Introduction

Why Compression

- Large-scale scientific applications generate extremely large amounts of data
- Limited storage capacity (even for large-scale parallel computers)
- The I/O bandwidth can create bottlenecks in the transmission



application	data scale	passive solution (?)	to reduce
HACC cosmology simulation	**20 PB** per one-trillion-particle simulation	**use up FS** 26 PB for Mira@ANL	**10x** in need
CESM climate simulation	20% vs **50%** of h/w budget for storage 2013 vs **2017**	**5h30m** to store NSF Blue Waters, I/O at 1 TBps	**10x** in need
APS-U High-Energy X-Ray Beams Experiments	hundreds of **PB** brain initiatives	**100-PB** buffer or, connection at 100 GBps	**100x** in need

↑ Nyx cosmological simulation: can generate up to 2.8 PB of data at 4096 scale

Introduction

INDIANA UNIVERSITY
BLOOMINGTON

Why Compression

- Large-scale scientific simulations generate extremely large amounts of data
- Limited storage capacity (even for large-scale parallel computers)
- The I/O bandwidth can create bottlenecks in the transmission
- Write is slow!

Lossy Compression

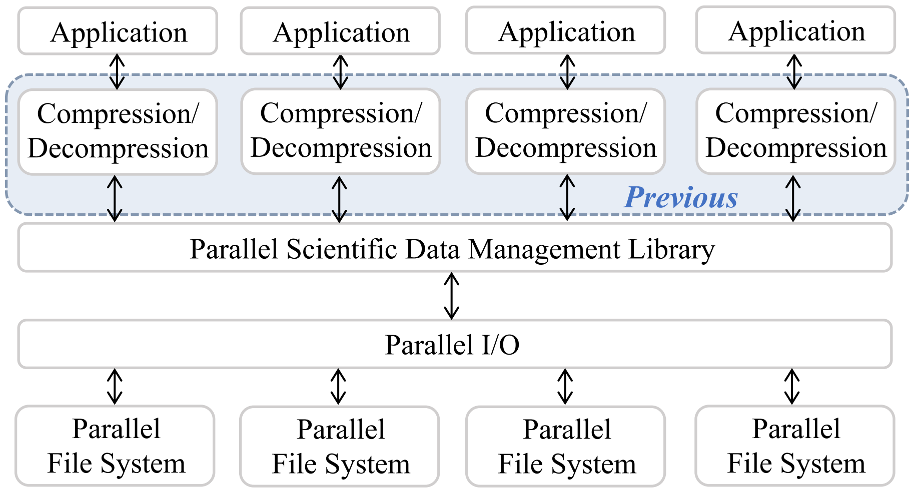
- High compression ratio
- Controllable compression error
- Improve overall performance!

application	data scale	passive solution (?)	to reduce
HACC cosmology simulation	**20 PB** per one-trillion-particle simulation	**use up FS** 26 PB for Mira@ANL	**10x** in need
CESM climate simulation	20% vs **50%** of h/w budget for storage 2013 vs **2017**	**5h30m** to store NSF Blue Waters, I/O at 1 TBps	**10x** in need
APS-U High-Energy X-Ray Beams Experiments	hundreds of **PB** brain initiatives	**100-PB** buffer or, connection at 100 GBps	**100x** in need

Introduction

Parallel I/O Libraries for HPC Applications

- Access and manage scientific data efficiently
- Move data between compute nodes and storage
- Compression Filter
  - Reduce storage footprint
  - Improve I/O performance

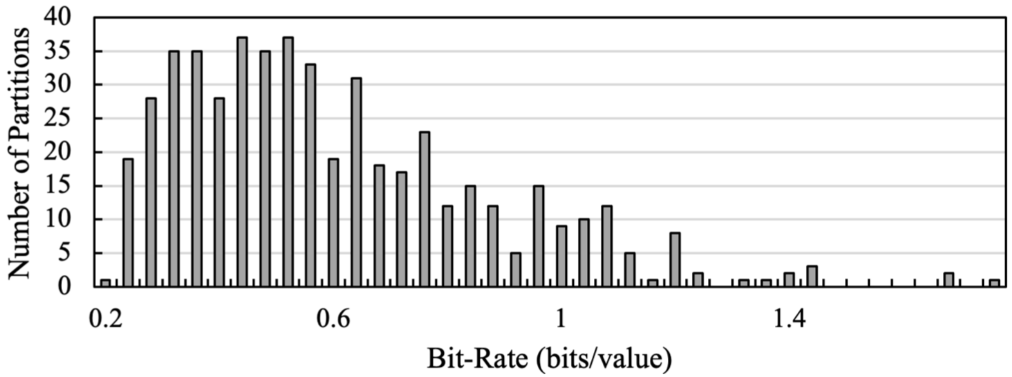


↑ Scientific data management with compression.

Introduction

Parallel I/O Libraries for HPC Applications

- Access and manage scientific data efficiently
- Move data between compute nodes and storage
- Compression Filter
  - Reduce storage footprint
  - Improve I/O performance



↑ Compression bit-rate distribution on a Nyx dataset with 512 partitions. Every partition uses the same compression configuration.

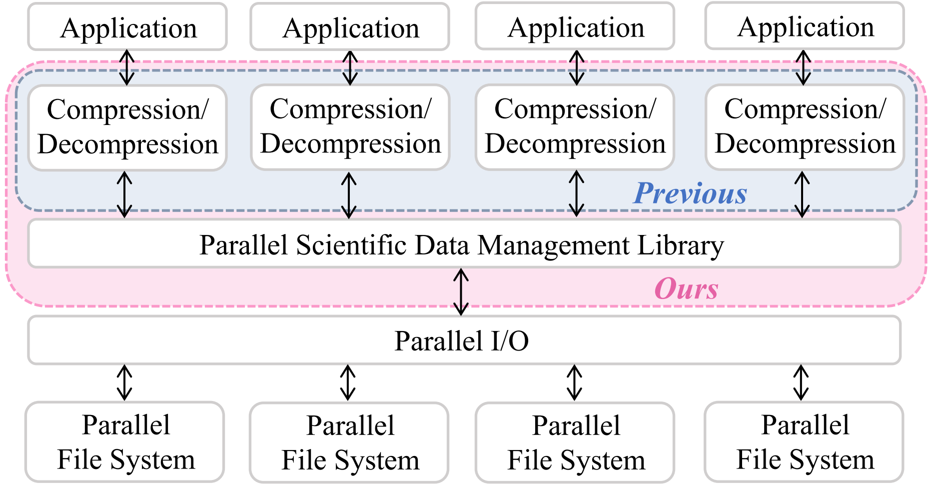
What Are The Limitation?

- High overhead: compression and I/O are in sequential
- Compression is not deeply integrated
  - Compression information is unknown to I/O libraries

Introduction

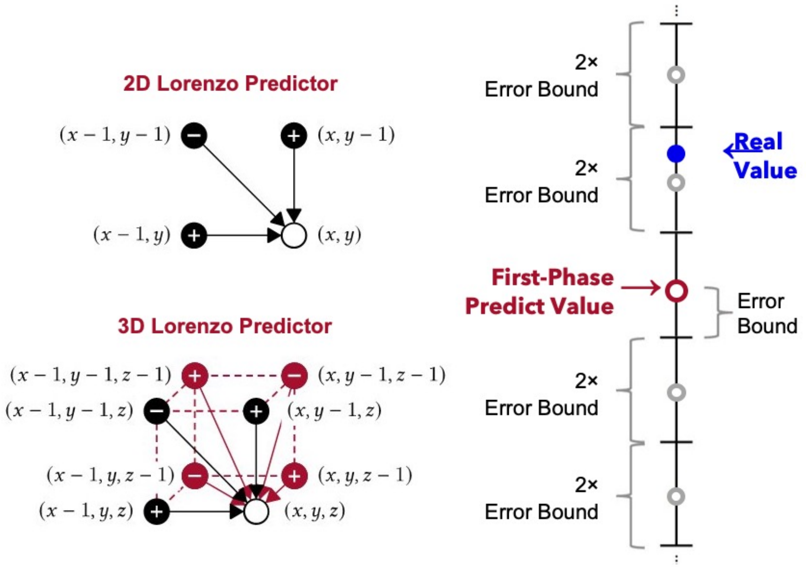
Our Solution & Contributions

- Extend the prediction model to estimate the offset and performance of parallel I/O
- Overlap I/O with compression
- Optimization for reorder compression tasks to achieve higher performance
- Improve the parallel-write performance by up to 4.5× and 2.9× compared to the HDF5 write without compression and with the SZ lossy compression filter, respectively, with only 1.5% storage overhead

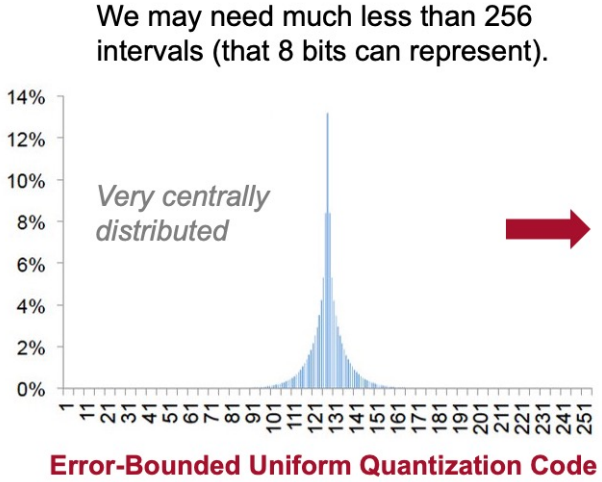


↑ Scientific data management with compression.

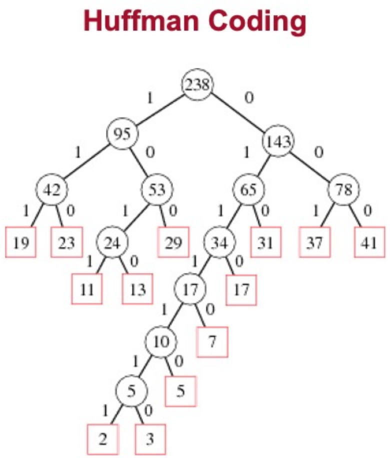
Background



prediction



quantization

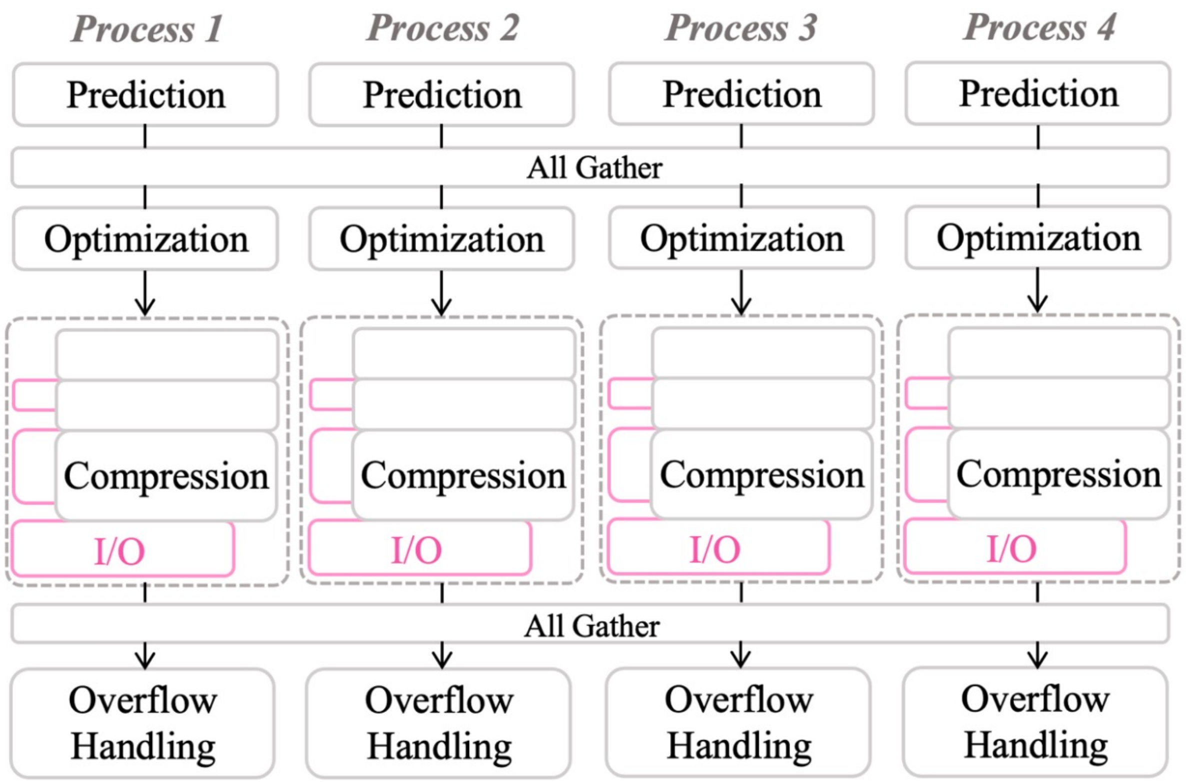


coding

Error Bounded Lossy Compression

- Compression ratio: ratio between original and compressed data size
- Bit-Rate: bits per value to encode the data
- Data distortion: reconstructed data quality compared to the original
- Error Bound: ensures differences between original and reconstructed data do not exceed the error bound

Design Methodology

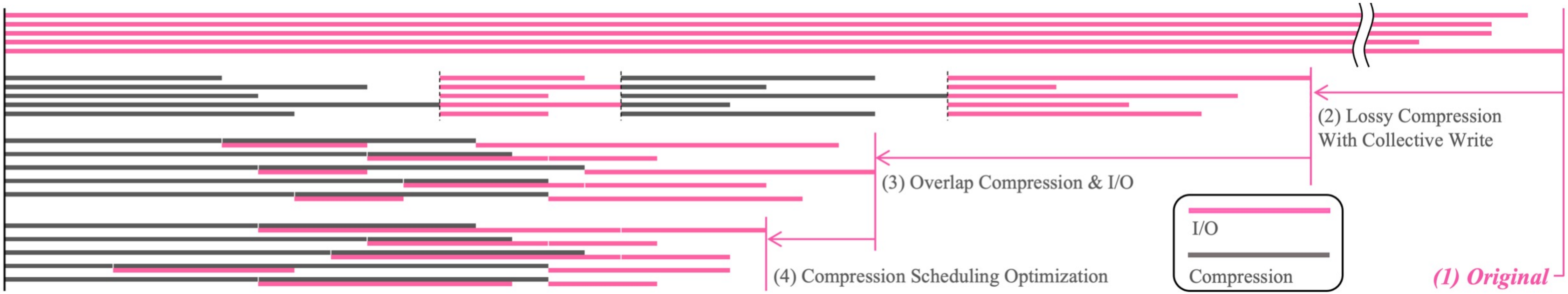


↑ Overview of our proposed solution.

Overall Design

- Predict ratio and throughput
- Distribute the estimated compression ratio of each partition to all processes
- Computes the offset for parallel write
- Optimize the order of compressing different data fields in each process
- Overlap compressions and writes
- Distribute overflow information
- Handle overflowed data

Design Methodology



↑ Timeline of data aggregation with 5 processes and 2 data fields.

How We Improves Over Previous Solutions

- Previous solutions:
  - (1) Original: non-compression solution
  - (2) Lossy compression solution using HDF5 filter (H5-SZ)
- Our Solutions:
  - (3) Overlap compression & I/O
  - (4) Overlap compression & I/O + compression scheduling optimization

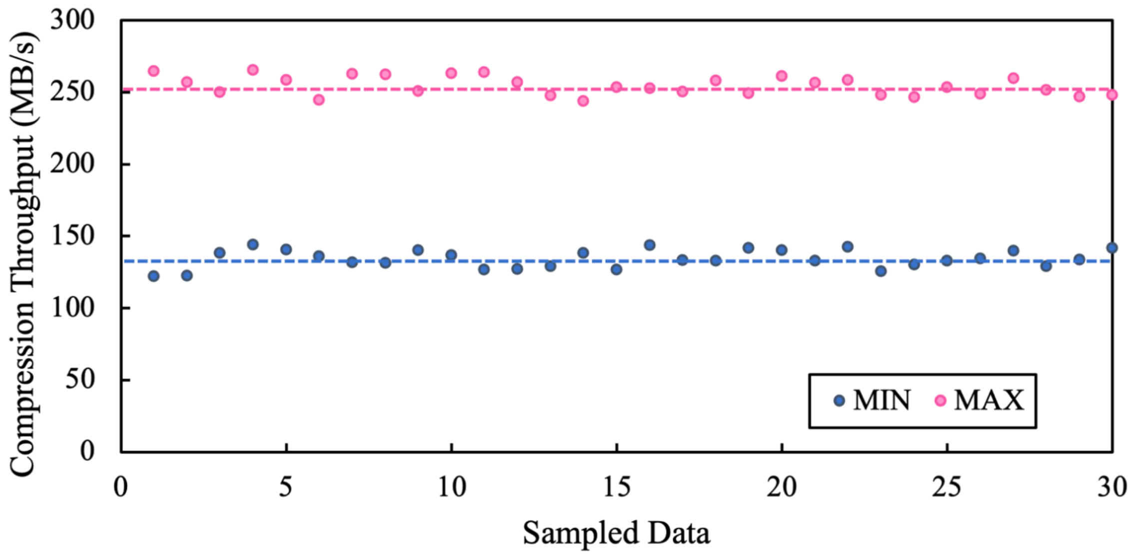
Design Methodology

INDIANA UNIVERSITY
BLOOMINGTON

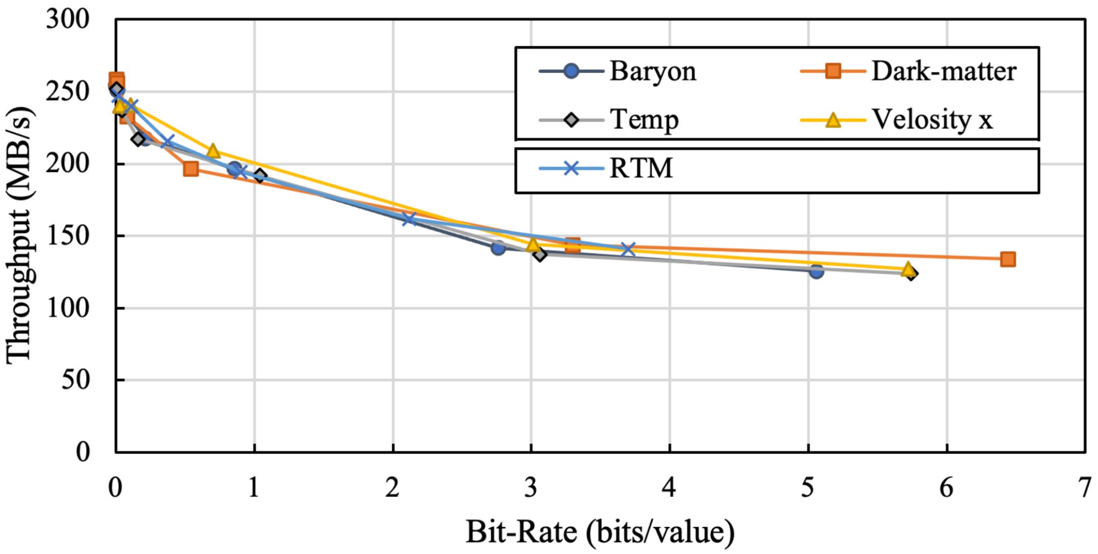
Compressor Throughput Estimation

- Min & max compression throughput are similarly bounded across different data samples
- Bitrate-throughput curve for each data sample is highly consistent

$$T_{comp} = D/S$$
$$= (B_{ori} \times n)/(((C_{max} - C_{min}) \times 3^{-a})B^a + C_{min})$$



↑ Minimum and maximum compression throughput of a given data partition based on 30 samples from Baryon density



↑ Single-core compression throughput with different bit-rates on a Nyx and a RTM datasets

Design Methodology

INDIANA UNIVERSITY
BLOOMINGTON

Compressor Throughput Estimation

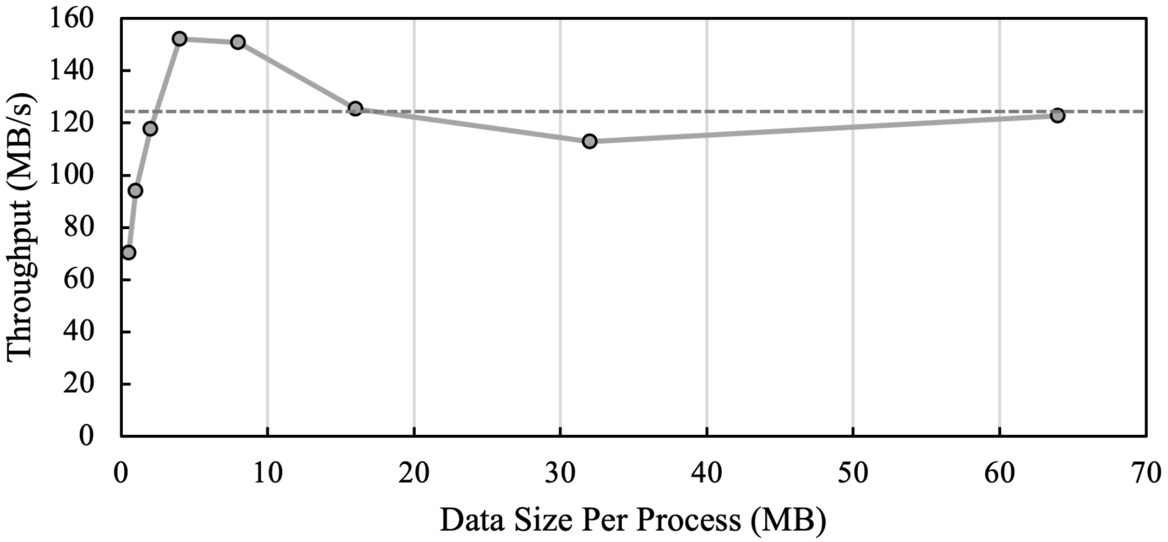
- Min & max compression throughput are similarly bounded across different data samples
- Bitrate-throughput curve for each data sample is highly consistent

$$T_{comp} = D/S$$
$$= (B_{ori} \times n)/(((C_{max} - C_{min}) \times 3^{-a})B^{a} + C_{min})$$

Write Time Estimation

- Not to provide a highly accurate write-time estimation for each data partition, but to provide a capability to estimate the relative write time across different data sizes
- Stabilizes after the data size reaches a certain point

$$T_{write} = (B \times n)/C_{thr}$$



↑ Independent write I/O throughput per process with different data sizes per process

Design Methodology

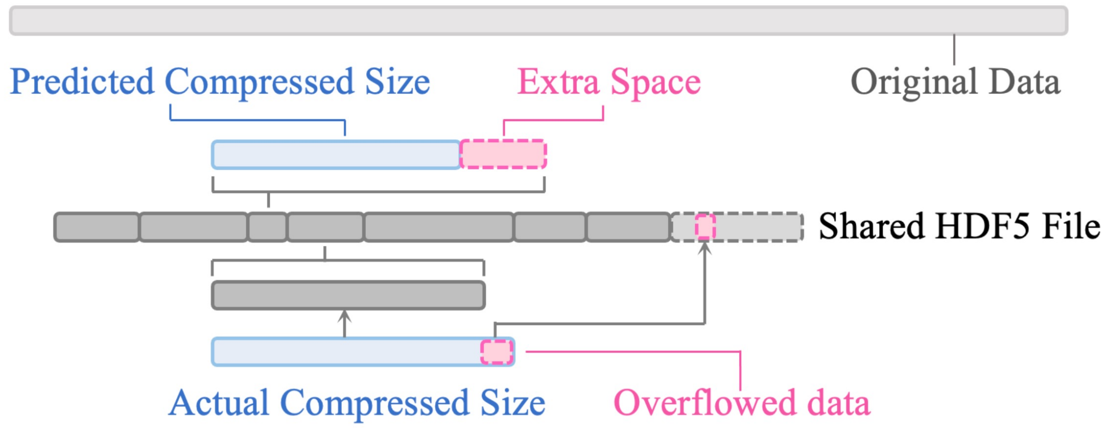
Overlapping Compression and Write

- Estimate the offset based on predicted compression ratio
- Reserve an extra space for unpredicted compressed data overflow
- Extra space ratio can be adjusted to balance between performance and compressed size overhead

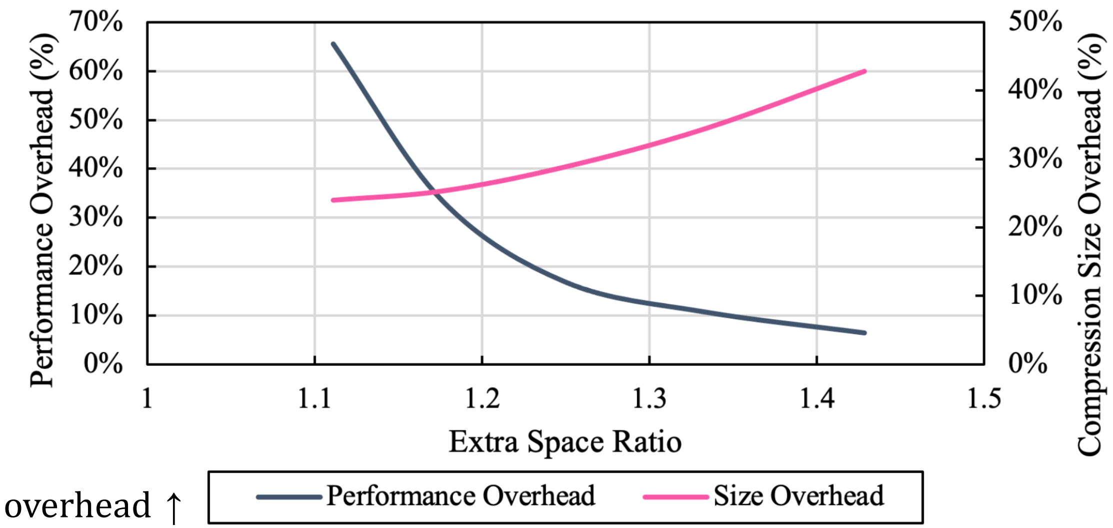
Extra Space Ratio

- Default at 1.25 for most partitions
- Adjusted for partitions with high estimated compression ratio

$$r_{space} = \min(2, 1 + (R_{space} - 1) \times 4), \quad where \quad r_{comp} > 32.$$



↑ Overflow data handling with preserved extra space.

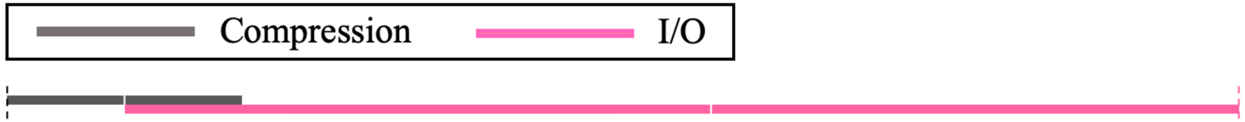


Trade-off between performance overhead and compression size overhead ↑

Design Methodology

Compression Order Optimization

- Improve overlapping efficiency
  - I/O of each partition happens after compression
  - Avoid unnecessary wait time for I/O
- Suitable: compression time and I/O time are similar
- Limited improvement:
  - I/O is significantly longer
  - Compression is significantly longer



(a) Compression time is significantly shorter

(b) Compression time is significantly longer

↑ An example of extremely unbalanced compression time and write time, limiting the benefit from our reordering.

Algorithm 1 Compression Order Optimization

Notation: data fields in current process: ℓ; compression queue: Q; compression queue after insert and additional data: Q°; possible insert locations in a queue: β; time to compress: t_c; time to write: t_w; predicted compression time: $P_c(\ell)$; predicted write time: $P_w(\ell)$

Global: $P_c(\ell), P_w(\ell)$

```
procedure TIME(q)
    t_c, t_w ← 0
    for ℓ ← data fields in q do
        t_c ← t_c + P_c(ℓ)
        t_w ← P_w(ℓ) + max(t_c, t_w)
    end for
    return t_w
end procedure

procedure SCHEDULINGOPTIMIZATOR
    for ℓ ← data fields in current process do
        for β ← all possible insert location do
            Q° ← insert ℓ to β
            if TIME(Q°) < TIME(Q) or first β then
                Q ← Q°
            end if
        end for
    end for
    return Q
end procedure
```

Evaluation

INDIANA UNIVERSITY
BLOOMINGTON

Evaluation Setup

- Implemented our approach with HDF5 and SZ3
- Two HPC systems
  - The Summit at ORNL, up to 4096 ranks
    - IBM POWER9 processors, Mellanox EDR InfiniBand interconnect
  - The Bebop at ANL, up to 512 ranks
    - two 18-core Intel Xeon E5-2695v4 CPUs, Omni-Path Fabric Interconnect

I/O-Intensive HPC Applications

- Nyx cosmology simulation
  - hydrodynamics code designed to model astrophysical reacting flows
- VPIC (vector particle-in-cell) plasma physics simulation

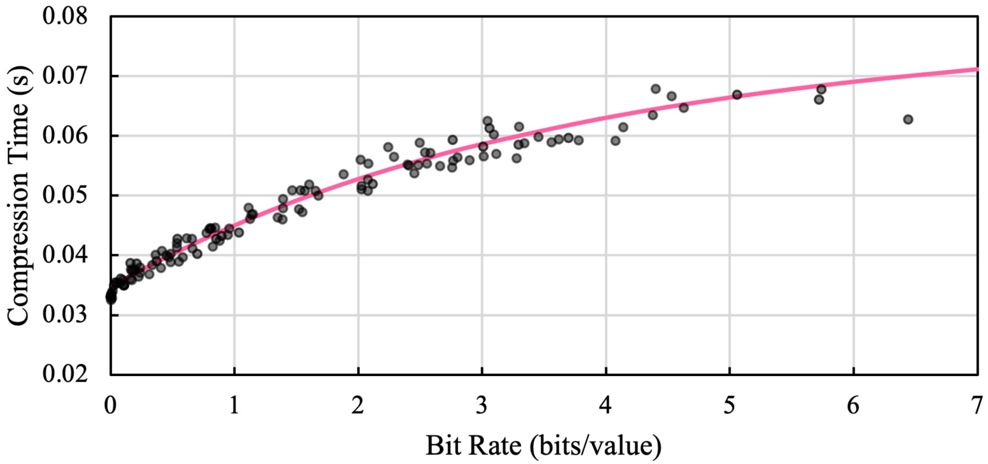
Name	Description	Scale	Size
nyx [18]	Cosmology simulation	4096×4096×4096	2.47 TB
		2048×2048×2048	206.15 GB
		1024×1024×1024	25.76 GB
		512×512×512	3.22 GB
VPIC [52]	Particle simulation	161,297,451,573	4.62 TB

↑ Details of Tested Datasets.

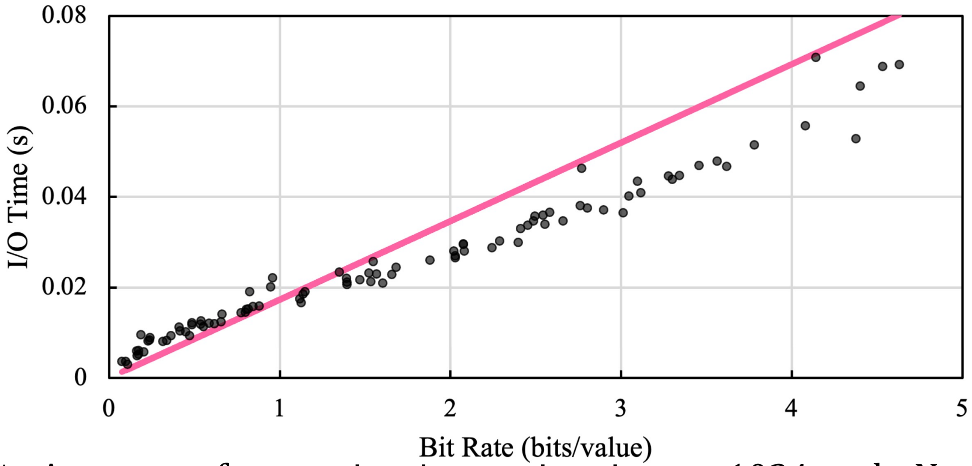
Evaluation

Accuracy of Compression and I/O Throughput Estimation

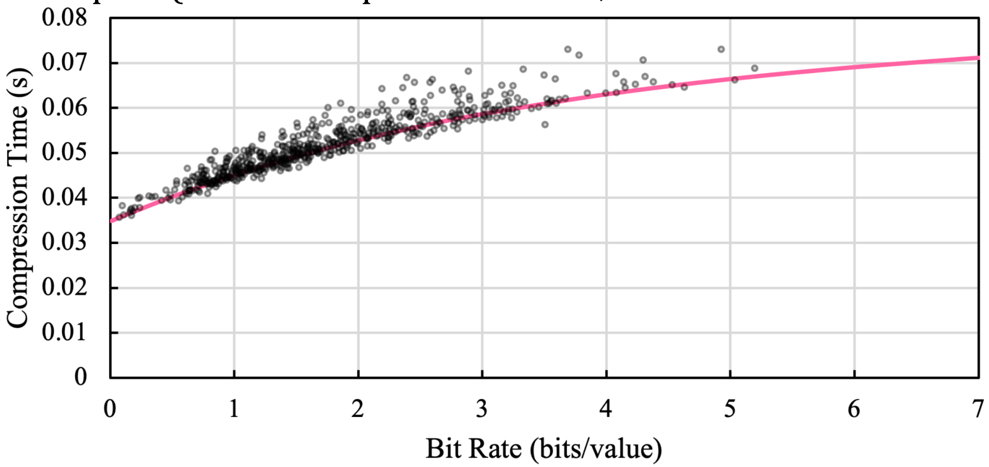
- High accuracy on compression time estimation
  - Different partitions
  - Different data scale
- High accuracy on write time estimation
  - Have some distortion but NOT affect our optimization



↑ Accuracy of our compression-time estimation on 512 scale Nyx data samples (red line is predicted time; black dots are actual time)

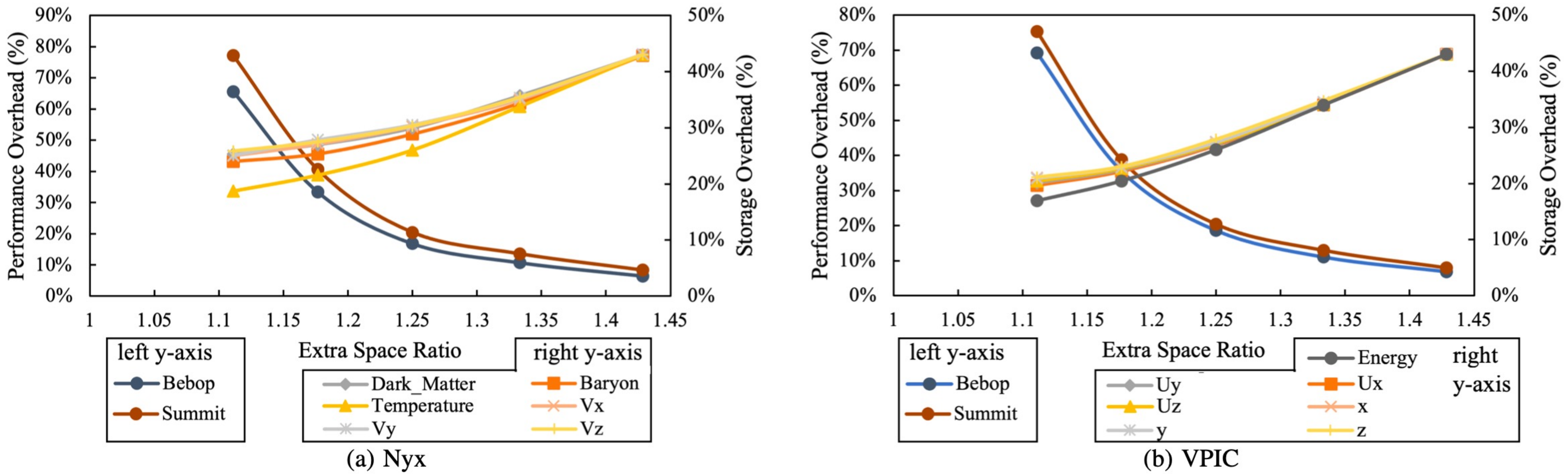


↑ Accuracy of our write time estimation on 1024 scale Nyx data samples. Red line is predicted time; black dots are actual time.



↑ Accuracy of our compression-time estimation on 1024 scale Nyx data samples. Red line is predicted time; black dots are actual time.

Evaluation



↑ Trade-off between performance overhead and storage overhead based on different extra space ratios on Nyx dataset (6 data fields) and VPIC dataset (7 data fields) on both Bebop and Summit with 512 processes.

Evaluation on Extra Space Ratio

- Trade-off curve between performance and storage are highly similar
- Lower the extra space ratio can result in extremely high performance overhead
- We choose the extra space ratio of 1.25 as default
- Can also custom the extra space ratio

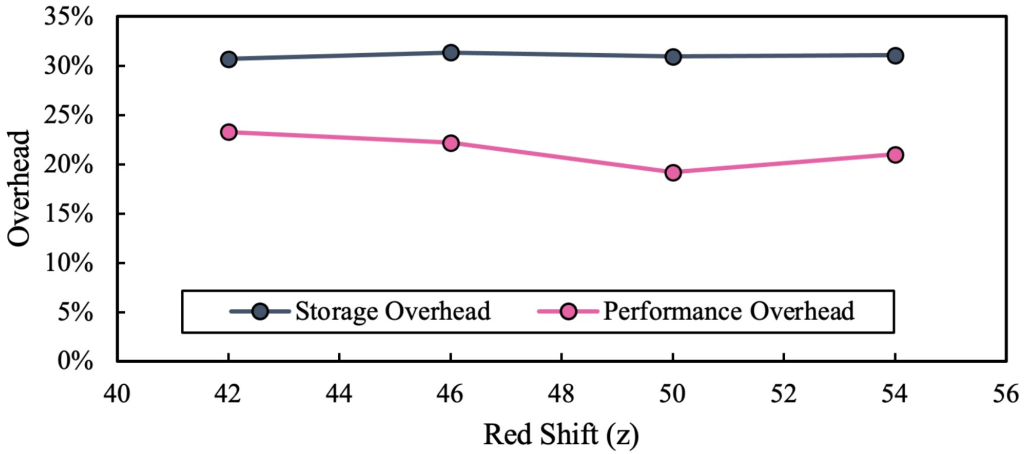
Evaluation

Comparison

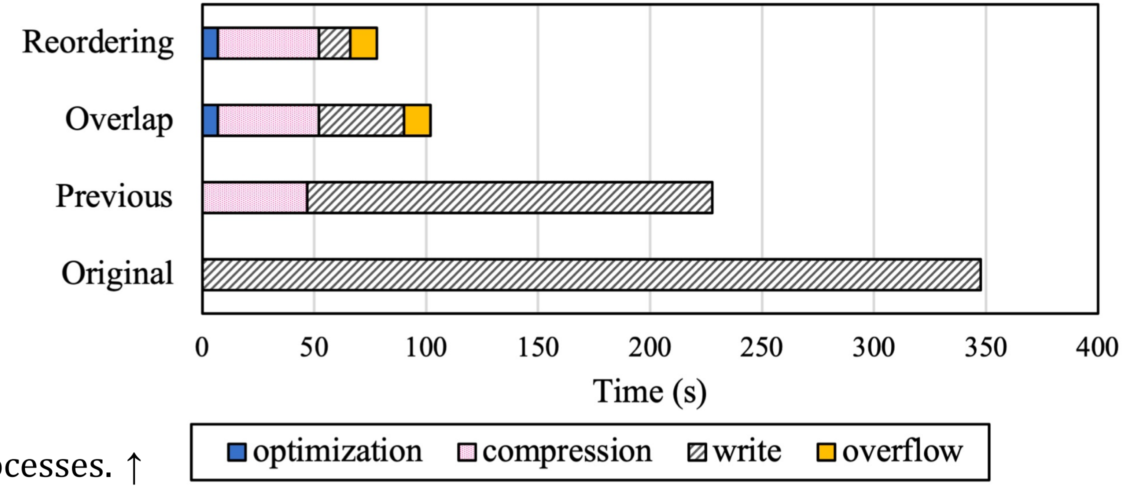
- Original: non-compression solution
- Previous: compression filter solution
- Overlap: our solution
- Reordering: overlap + reorder technique

Performance Improvement

- Stable performance over timesteps
- Original → Previous: 1.87×
- Previous → Overlap: 1.79×
- Overlap → Reordering: 1.30×
- Overall: 2.91× improvement from previous with a 1.5% storage overhead compared to original size.

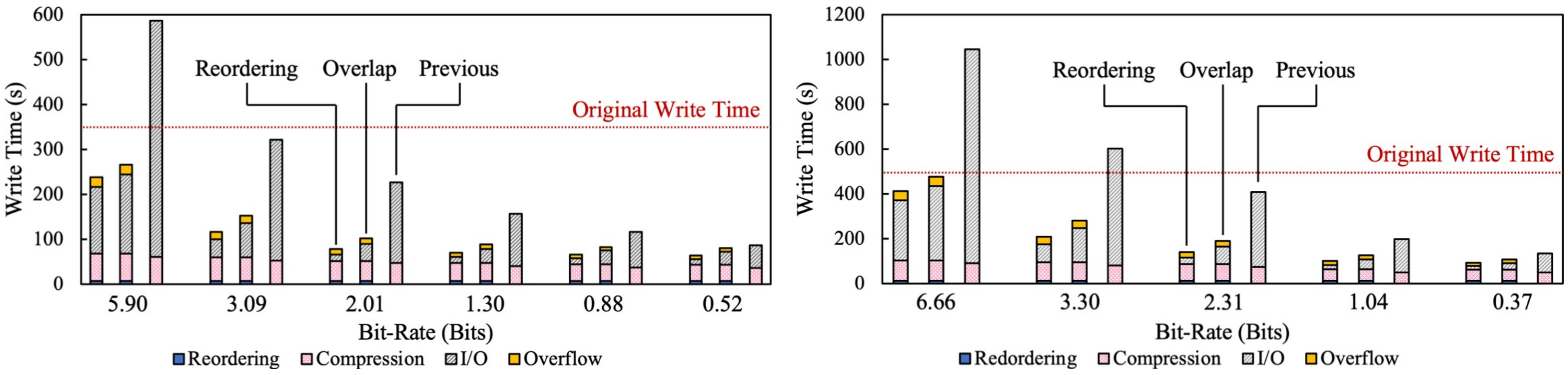


↑ Evaluation on the consistency of the storage and performance overheads using the same extra space ratio of 1.25 with 512 processes on Summit.



Performance comparison among our solution (overlapping and reordering), original non-compression solution, and previous compression-write solution on 4096^3 Nyx dataset with 512 processes. ↑

Evaluation

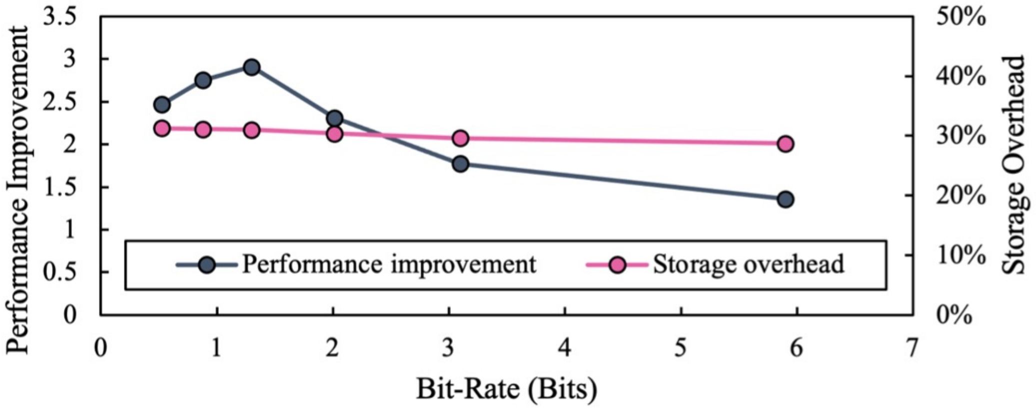


↑ Performance improvement with different overall data reduction ratios. Dashed red line is the baseline of HDF5 without compression. Left: 4096^3 Nyx dataset, right: VPIC dataset. Evaluated with 512 processes on Summit.

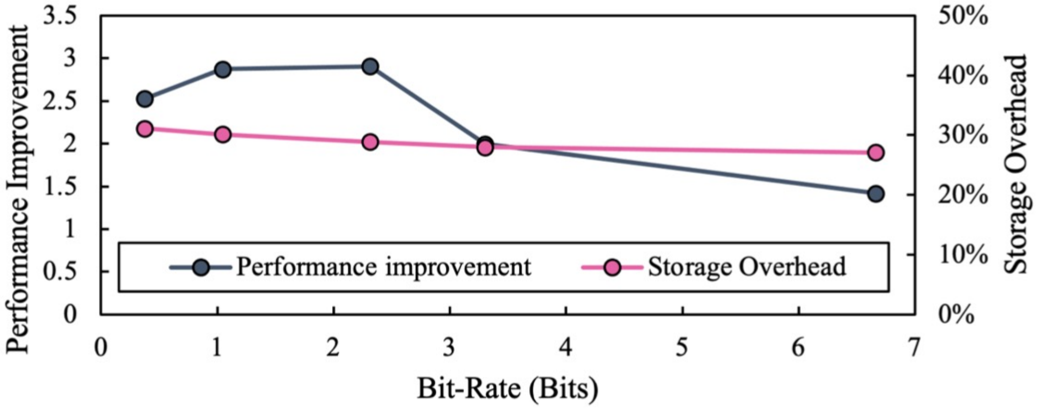
Performance with Different Overall Ratios

- Limited improvement from reordering optimization under extremely high/low bit-rate
  - High bit-rate: I/O time significantly larger than compression time
  - Low bit-rate: compression time significantly larger than I/O time
- Storage overhead is stable
- Performance improvement over previous is more significant at bit-rate of ~2 bits
  - Low/high bit-rate: compression-time/write-time dominate, less overlap efficiency

Evaluation



(a) Nyx with different compression ratio



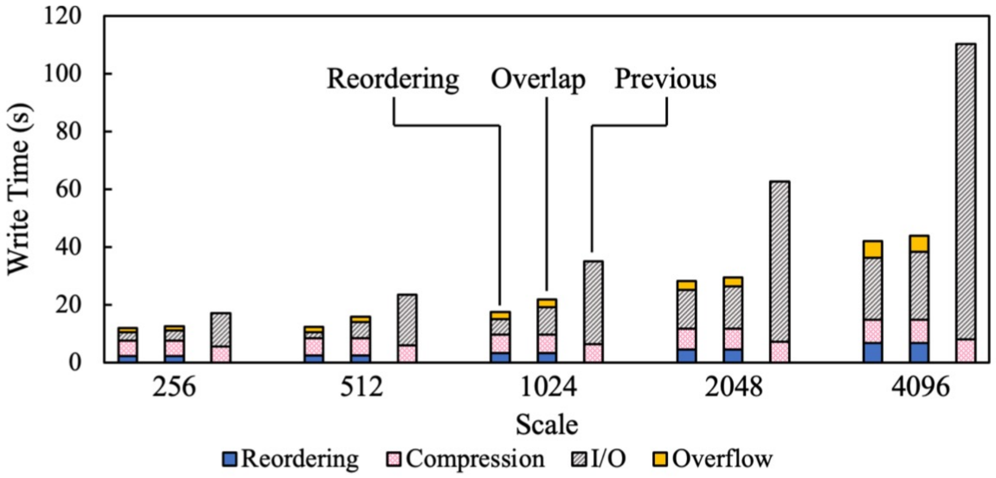
(b) VPIC with different compression ratio

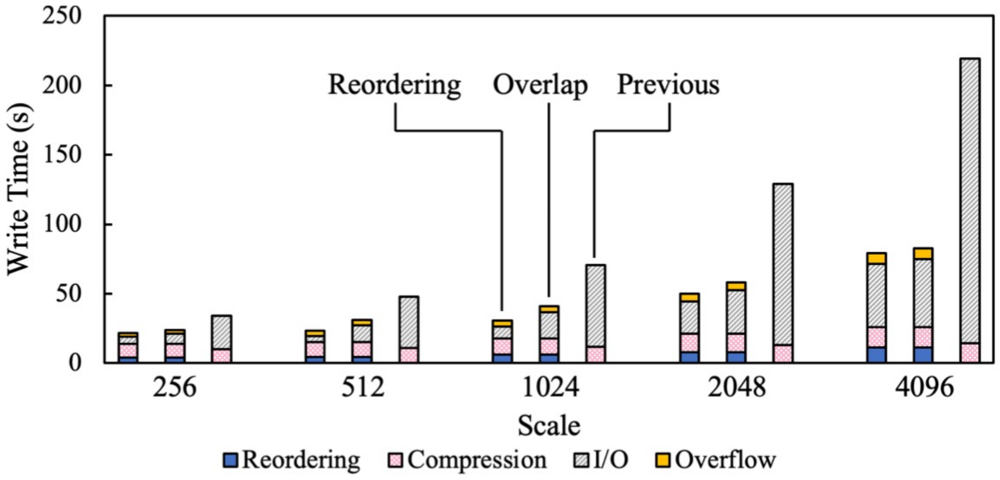
↑ Performance improvement (overall) and storage overhead of our solution compared to the previous solution on both 4096^3 scale Nyx and VPIC datasets. Evaluated with 512 processes on Summit.

Performance with Different Overall Ratios

- Limited improvement from reordering optimization under extremely high/low bit-rate
  - High bit-rate: I/O time significantly larger than compression time
  - Low bit-rate: compression time significantly larger than I/O time
- Storage overhead is stable
- Performance improvement over previous is more significant at bit-rate of ~2 bits
  - Low/high bit-rate: compression-time/write-time dominate, less overlap efficiency

Evaluation



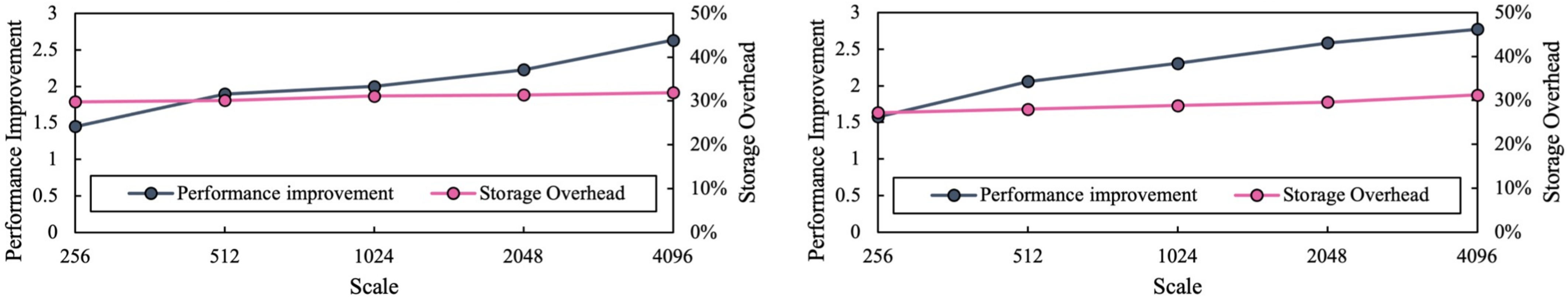


↑ Performance improvement with different scale on both Nyx (left) and VPIC (right) datasets. Dashed red line is the baseline of HDF5 without compression. Average bit-rate is 2. 256^3 or 39, 379, 260 data values per field for each rank.

Performance with Different Scales

- Improvement from reordering optimization is stable
- Storage overhead is stable
- Performance improvement over previous is more significant towards larger scale
  - Asynchronous write typically provides better scalability compared to the collective write used by the previous compression-write solution

Evaluation



(c) Nyx with different scale

(d) VPIC with different scale

↑ Performance improvement (overall) and storage overhead of our solution compared to the previous solution on both Nyx and VPIC datasets. (c) and (d) are evaluated with a target bit-rate of 2.

Performance with Different Scales

- Improvement from reordering optimization is stable
- Storage overhead is stable
- Performance improvement over previous is more significant towards larger scale
  - Asynchronous write typically provides better scalability compared to the collective write used by the previous compression filter solution

Conclusion

Conclusion

- We extend the prediction model for compression ratio to predict the throughputs of compression and parallel write for prediction-based lossy compression
- We propose a new compression-write scheme with HDF5 that can efficiently overlap compression with write based on our prediction models
- We optimize the execution order of compression tasks in each process to achieve higher parallel-write performance
- Our solution improves the parallel-write performance by up to 4.5× and 2.9× compared to the HDF5 write without compression and with the SZ lossy compression filter, respectively, with only 1.5% storage overhead

Future Works

- Extend our solution to other parallel I/O libraries such as ADIOS
- Support more lossy compressors such as ZFP
- Evaluate our solution on more real-world HPC datasets

Thank you!

Any questions are welcome!

Contact Dingwen Tao: ditao@iu.edu
Sian Jin: sianjin@iu.edu