Exploring the Feasibility of In-Line Compression on HPC Mini-Apps

Dakota Fulp
LANL Mentor: Dr. Laura Monroe
Clemson Mentor: Dr. Jon C. Calhoun
August 12th 2020

Los Alamos NATIONAL LABORATORY EST. 1943

NNSA National Nuclear Security Administration

Managed by Triad National Security, LLC for the U.S. Department of Energy's NNSA

LA-UR-20-26131

Overview

- Background
  - ASC BML Inexact Computing Project
  - Lossy Compression
  - In-Line Lossy Compression
  - ZFP's Fixed-Rate Mode
- Implementation
- Experimental Setup
- Effects of In-Line Compression
  - Accuracy
  - Storage
  - Throughput
- Conclusions
- Extending Our Work

ASC BML Inexact Computing Project

Present technology is not capable of doubling the number of transistors in integrated circuits every two years

Moore's Law is coming to an end

New forms of advancement needed

Inexact computing trades precision for:

- Gains in computing efficiency
- Significant energy savings

This project aims to:

- Help teams adapt to the end of Moore's law
- Improve computation efficiency in mission codes
- Integrate efforts in future codes and platforms

Lossy Compression

Form of approximate computing

Uses inexact approximations to reduce overall data size

Inaccuracies controlled via error bounding metrics

Example:

Before Lossy Compression:

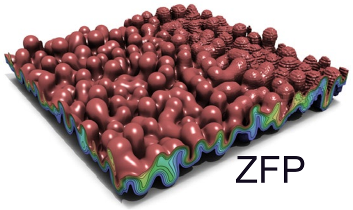
```
23.0348327
10.99
51.1
```

Absolute Error Bound: 0.1 →

After Lossy Compression:

```
23.0
10.9
51.1
```

Many different lossy compression algorithms:



In-Line Lossy Compression

Compress individual floating-point arrays within an application

Useful in reducing active memory footprint during runtime

Two types of in-line compression:

- **Full**: All data must be decompressed to access any data value
- **Partial**: Only a chunk of data must be decompressed to access a data value

Partial in-line compression:

- Less overhead to access data values
- Not all compression algorithms compatible

This work focuses on partial in-line compression

ZFP Fixed-Rate Lossy Compression

ZFP's fixed-rate mode enables partial in-line compression:

- Data divided into 4^d sized blocks, where d = dimensionality
- User defined rate determines compressed size of data blocks
- Each block can be decompressed independent of any other block

$$Compressed\_Block\_Size = \boldsymbol{Rate} * 4^d \qquad d = data\ dimensionality$$

1-D Double Array Example:

Rate = 48:

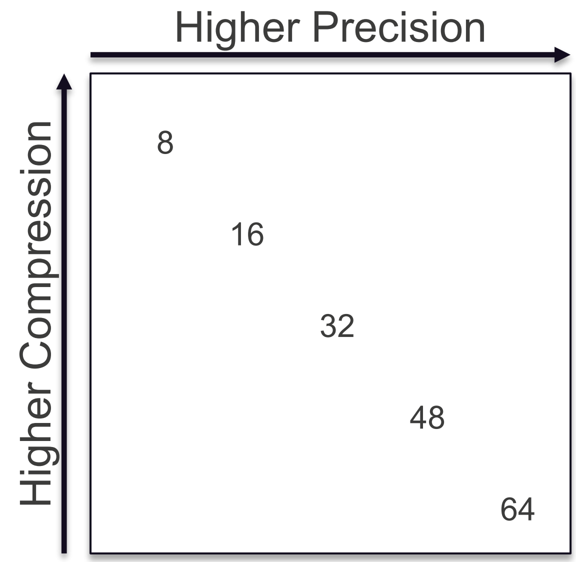
$$CompressedBlockSize = Rate * 4^d$$
$$Rate * 4^d = 48 * 4^1$$
$$48 * 4 = 192$$

192 Bits Per Compressed Block

Rate = 16:

$$CompressedBlockSize = Rate * 4^d$$
$$Rate * 4^d = 16 * 4^1$$
$$16 * 4 = 64$$

64 Bits Per Compressed Block



Experimental Setup

LANL HPC Mini-Apps

– PENNANT:
  - Unstructured mesh mini-app
  - Fewer large floating-point arrays

– Branson:
  - Monte Carlo transport mini-app
  - Many smaller floating-point arrays

ZFP Rates

– 64.0, 48.0, 32.0, 16.0, 8.0

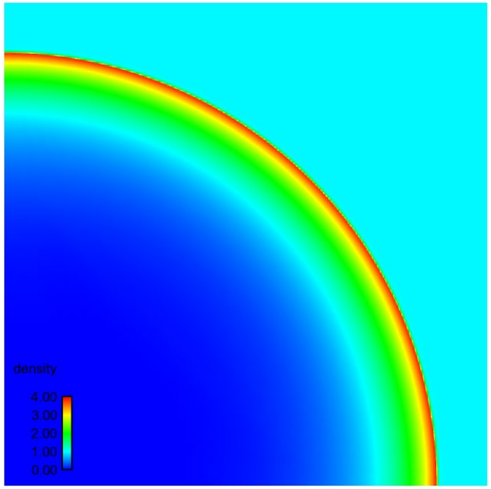
Input Files

– Sedov: Sedov-Taylor expansion

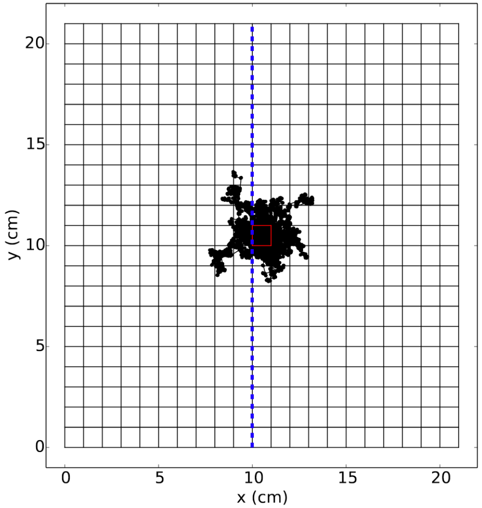
– Cube: Cube decomposition transport mesh

System

– Potatohead Cluster: 8 Intel Xeon CPU’s 126GB RAM



Sedov-Taylor Expansion Example



Particle Transport Mesh Example

Implementation

Replace all standard floating-point arrays with ZFP arrays

```
## Original
double* x = std::malloc(25 * sizeof(double));
```

```
## ZFP Array
zfp::array1<double> x(25, rate);
```

Extra steps needed when replacing object arrays

```
## Original
class Mesh {
        double* x;
}
x = std::malloc(25 * sizeof(double));
```

```
## ZFP Array
class Mesh {
        zfp::array1<double> x;
}
x.set_rate(rate);
x.resize(25);
```

Improved vector datatype support needed

Implementation

Convert back to standard floating-point arrays before MPI calls

```
## Create temporary array
double* tmp_x = std::malloc(25 * sizeof(double));
for (int i = 0; i < 25; i++){
    tmp_x[i] = x[i]
}
```

```
## Make MPI calls
MPI_Send(&tmp_x, 25, MPI_DOUBLE, 1, 0, MPI_COMM_WORLD);
```

```
## Copy temporary array back to original
for (int i = 0; i < 25; i++){
    x[i] = tmp_x[i]
}
```

Causes unnecessary overhead, reducing productivity

Currently developing MPI and OpenMP support

Effects of In-Line Compression

Accuracy:

- Results gathered at set cycle counts
- Peak Signal-to-Noise Ratio (PSNR) used to quantify data quality

$$PSNR = 20 * \log_{10}((max_{true} - min_{true})/RMSE)$$

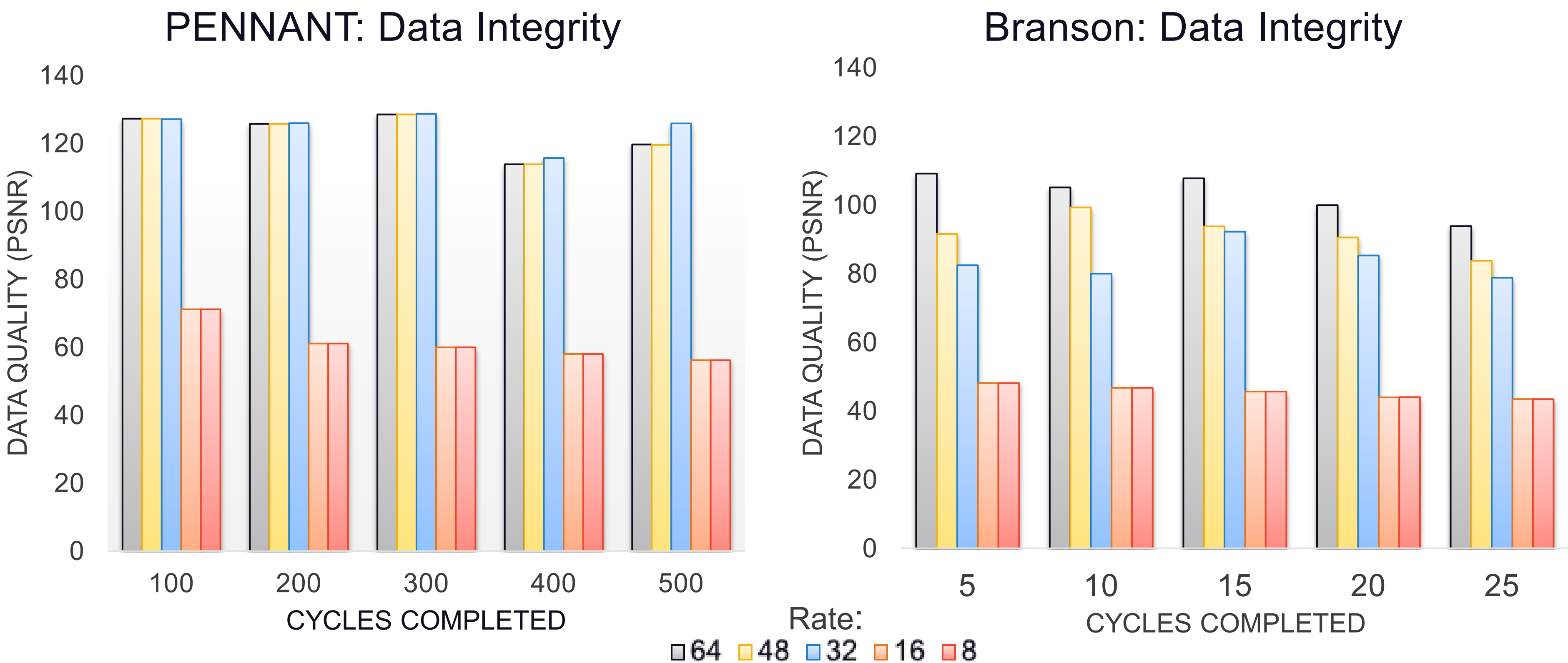
Storage:

- Compare size of original floating-point arrays with new ZFP arrays
- Determine the compression ratio

Throughput:

- Compare original runtime with:
  - ZFP-based application running with MPI
  - ZFP-based application running without MPI

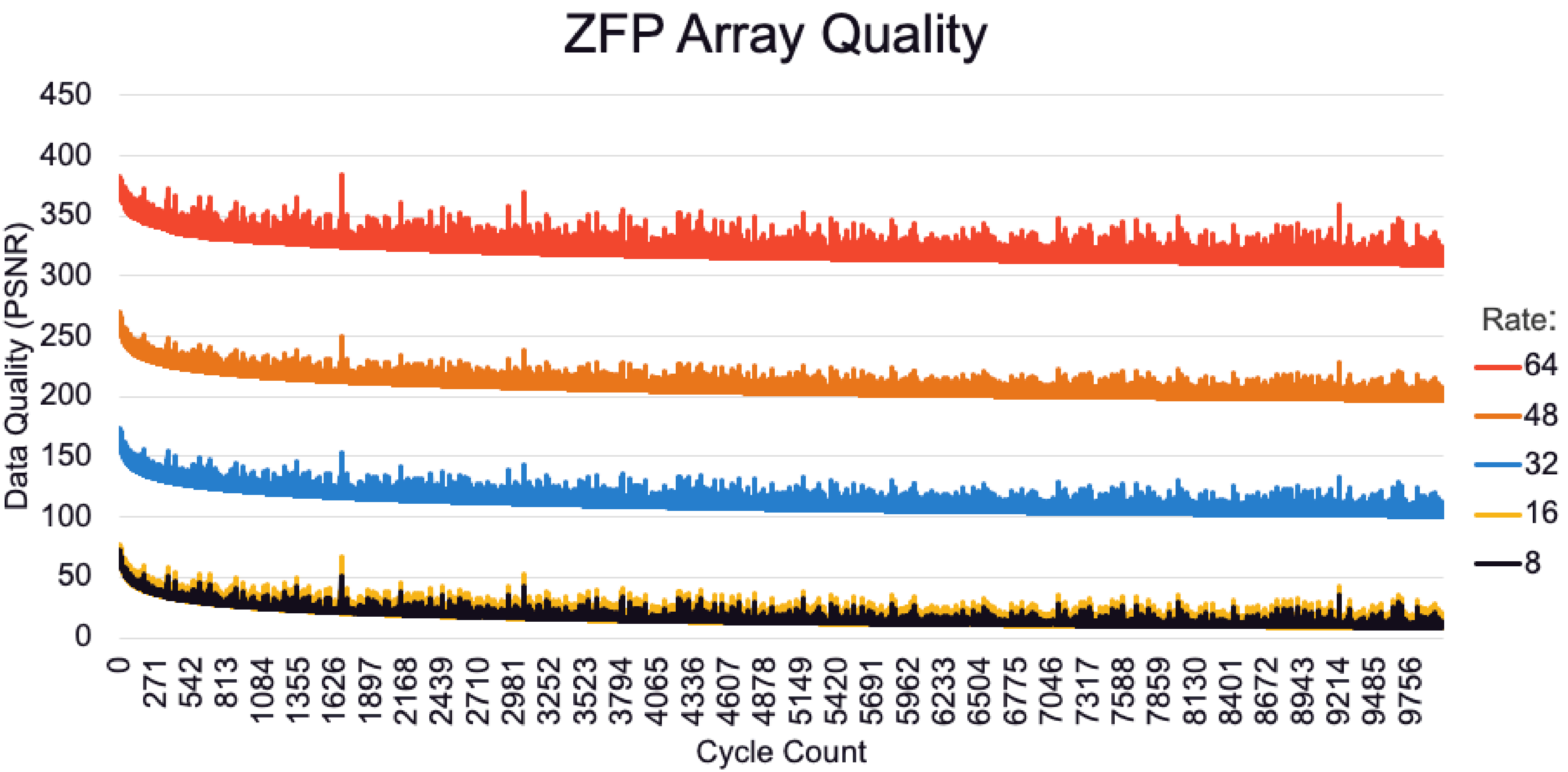
Effects on Accuracy



As more application cycles are completed, the data quality becomes less stable and begins to degrade

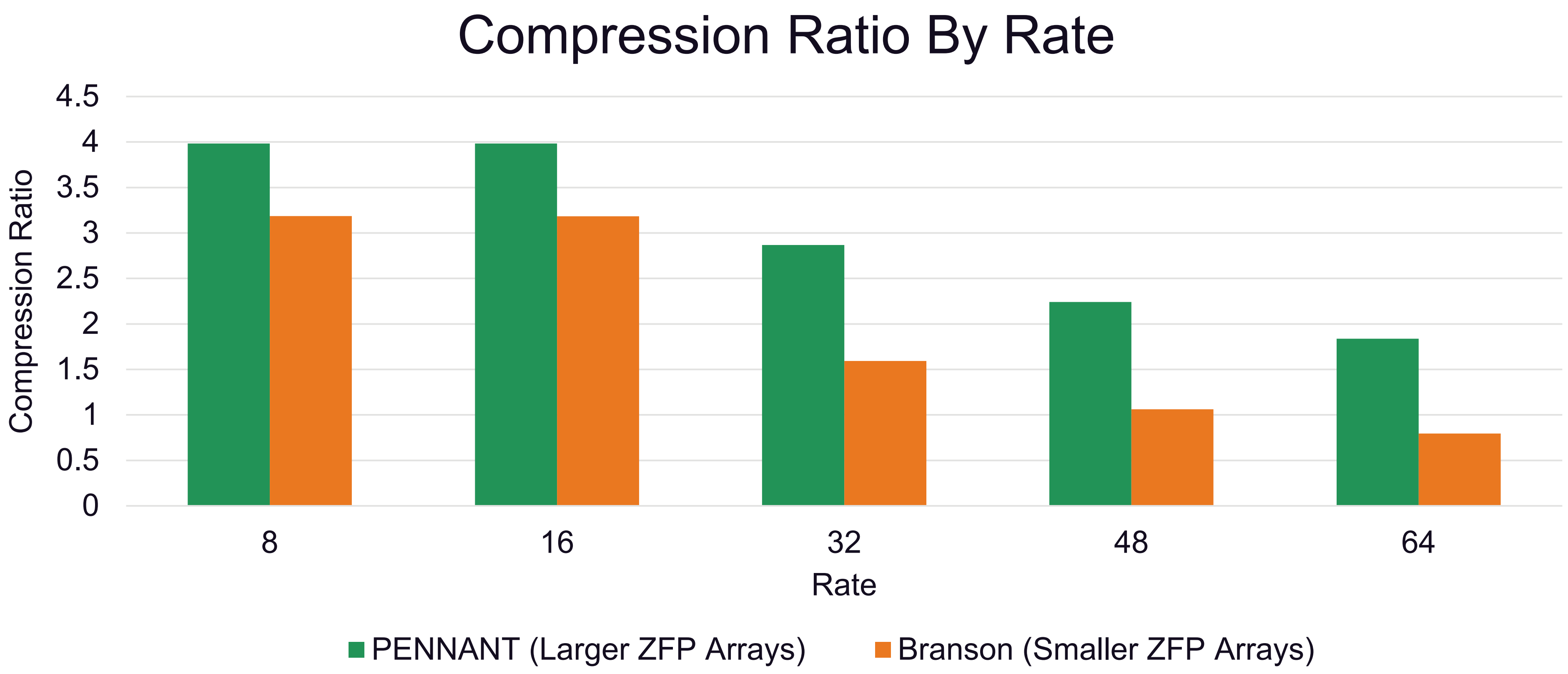
Due to propagation of inaccuracies

Effects on Accuracy



ZFP array quality logarithmically degrades over time

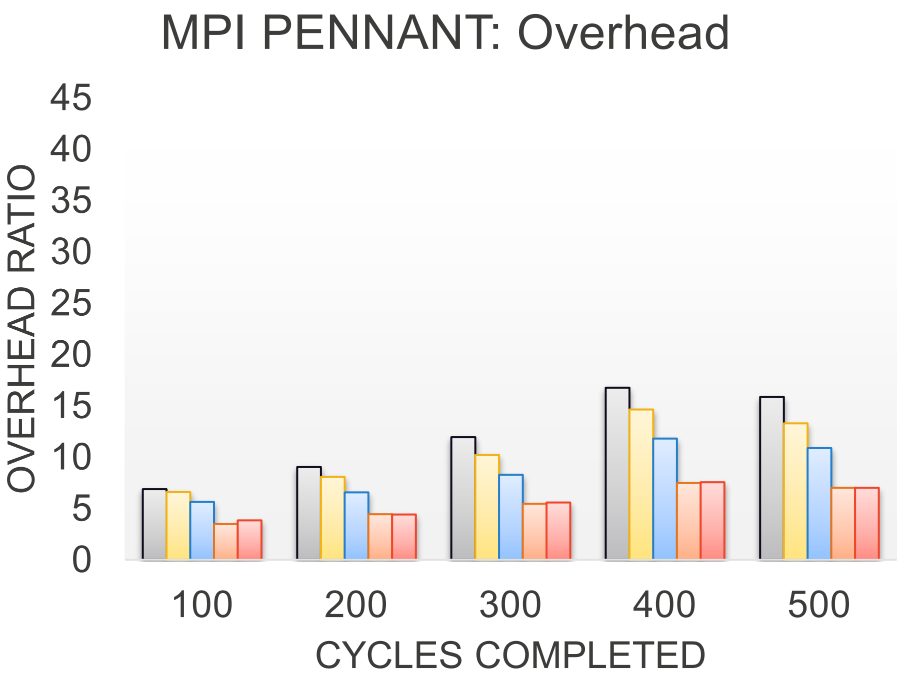
Effects on Storage

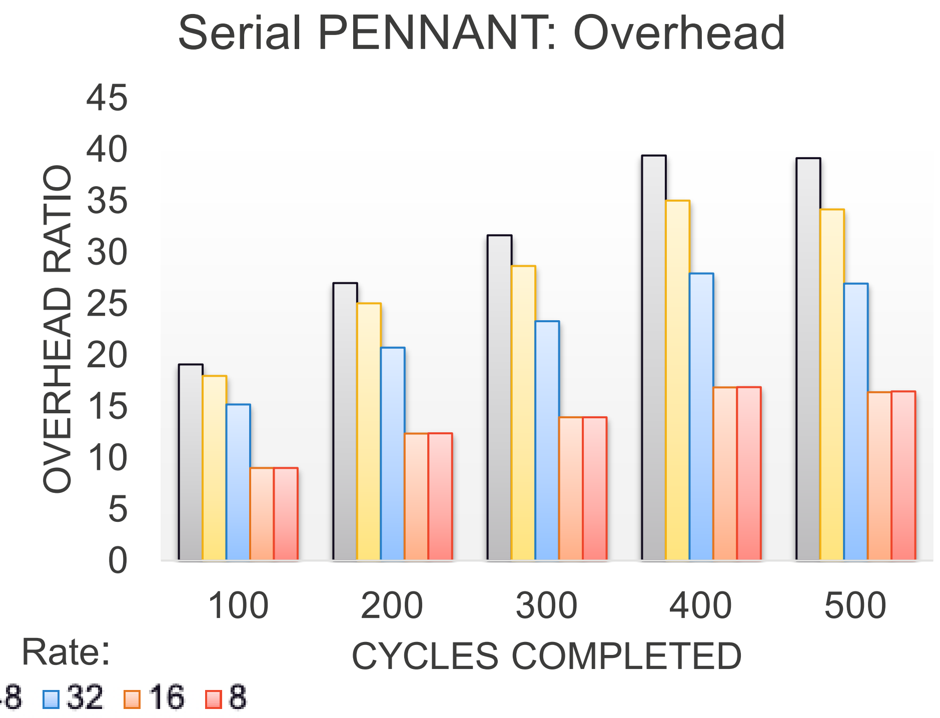


Larger ZFP arrays and lower ZFP rates result in higher levels of compression

Compression ratios range from 0.796x to 3.983x

Effects on Time

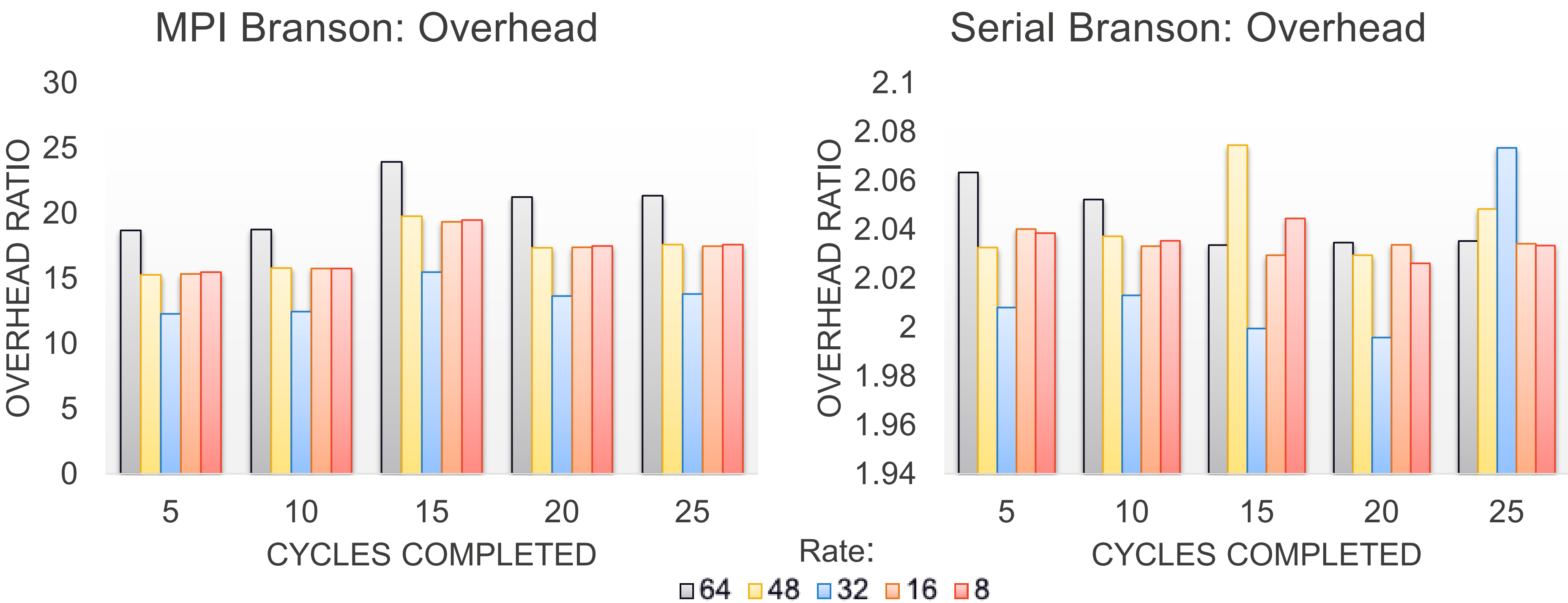




Overhead grows as cycles completed grows and is more prominent with higher rates

MPI reduces ZFP overhead even with conversion overhead

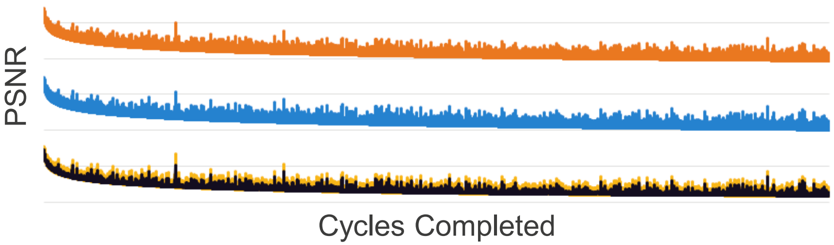
Effects on Time



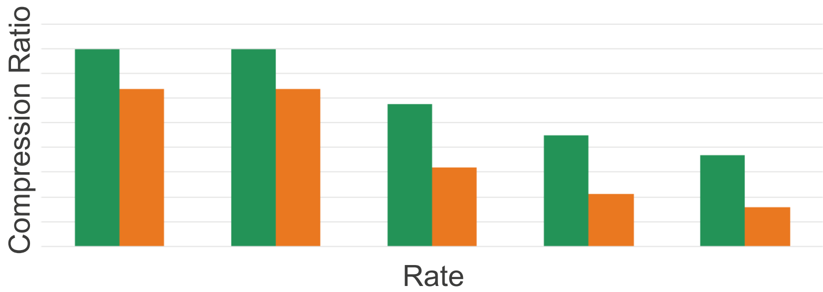
MPI overhead higher than PENNANT due to strong use of MPI

Serial overhead much lower than PENNANT due to many smaller ZFP arrays being used

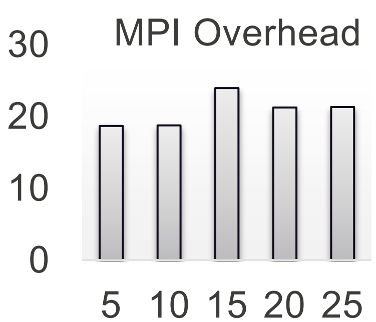
Conclusions

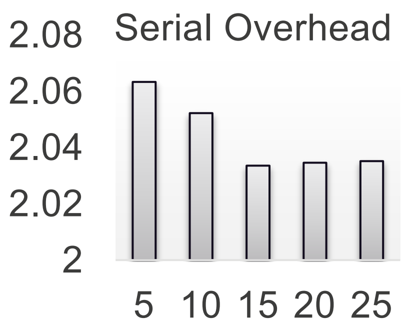


ZFP array quality degrades logarithmically over time



Larger ZFP arrays and lower ZFP rates result in higher compression ratios





Time overhead depends on data layout and MPI usage with smaller ZFP arrays demonstrating less additional overhead

ZFP arrays require improvements and optimizations in order to be viable on HPC applications

Extending Our Work

- Continue development of standard MPI datatype
- Resolve OpenMP race condition
- Improve ZFP API through improved vector support
- Profile the memory costs of using ZFP compressed arrays

Acknowledgements

Laura Monroe, LANL Mentor, lmonroe@lanl.gov

Jon Calhoun, Collaborator Mentor, jonccal@clemson.edu

ASC BML Project Team

USRC

NMC

Julie Wiens

Questions?