Exploring Data Compression in Distributed File Systems

Dongfang Zhao

CS554 Term Project, IIT Chicago

December 2013

Outline

- Introduction
- Design and implementation
- Evaluation
- Conclusion

Introduction

- Many scientific applications are data-intensive, NOT bounded on computation
- Two typical solutions:
  - Parallelize data process (HDFS, PVFS, etc.)
  - Squeeze data size (LZO, gzip, etc.)
- Combine the two solutions?
  - Yes, HDFS offers the compression option
  - But, there are some issues

Introduction

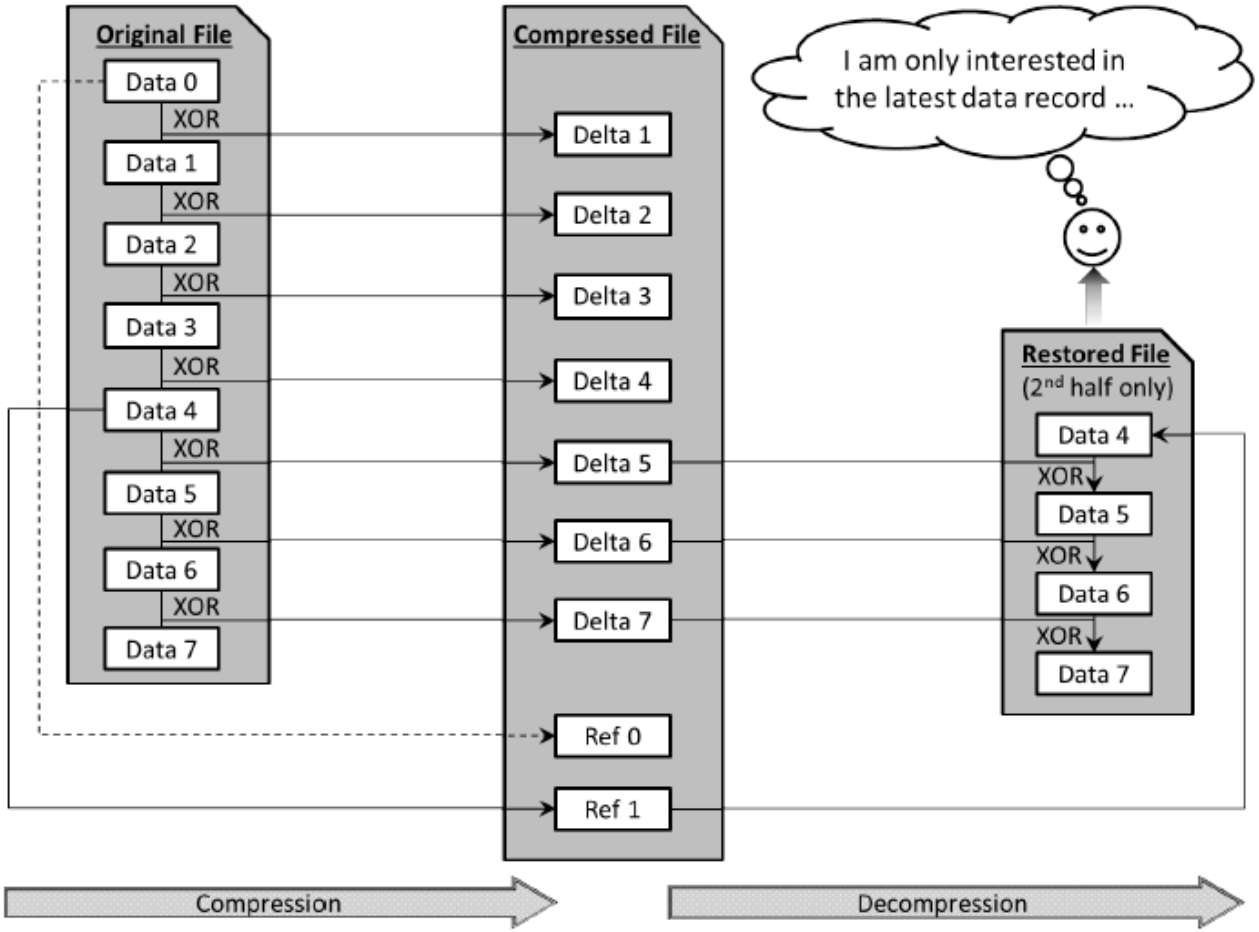
- So why not using HDFS with compression for scientific applications?
  - HDFS is implemented in Java
  - HDFS does not (natively) support POSIX
  - The compression is agnostic about the chunk size
    - Large chunk → bigger decompression overhead
    - Small chunk → less redundancy to compress
- We need a distributed file system implemented in C, supporting POSIX, and providing smart compressions

Introduction

- FusionFS is a new distributed file system with POSIX support, for HPC (high-performance computing) applications
- This project aims to extend FusionFS with a compression layer that overcomes the dilemma on the chunk size

Design

- Multiple-Reference Compression (MRC)



Design

- How many references?

The overhead to write the extra R reference points for data compression is

$$T_c = \frac{R \cdot S \cdot W_o}{N \cdot B_w},$$

and we would save the following time in decompression:

$$T_d = \frac{(S - \frac{S}{R}) \cdot W_i}{B_r}.$$

Now, we want to maximize

$$F(R) = T_d - T_c.$$

By taking the derivative on R (suppose $\hat{R}$ is continuous) and solving the following equation

$$\frac{d}{d\hat{R}}(F(\hat{R})) = \frac{S \cdot W_i}{B_r \cdot \hat{R}^2} - \frac{S \cdot W_o}{B_w \cdot N} = 0,$$

we have

$$\hat{R} = \sqrt{N \cdot \frac{B_w}{B_r} \cdot \frac{W_i}{W_o}}.$$

Note that

$$\frac{d^2}{d\hat{R}^2}(F(\hat{R})) = -\frac{S \cdot W_i}{B_r \cdot \hat{R}^3} < 0,$$

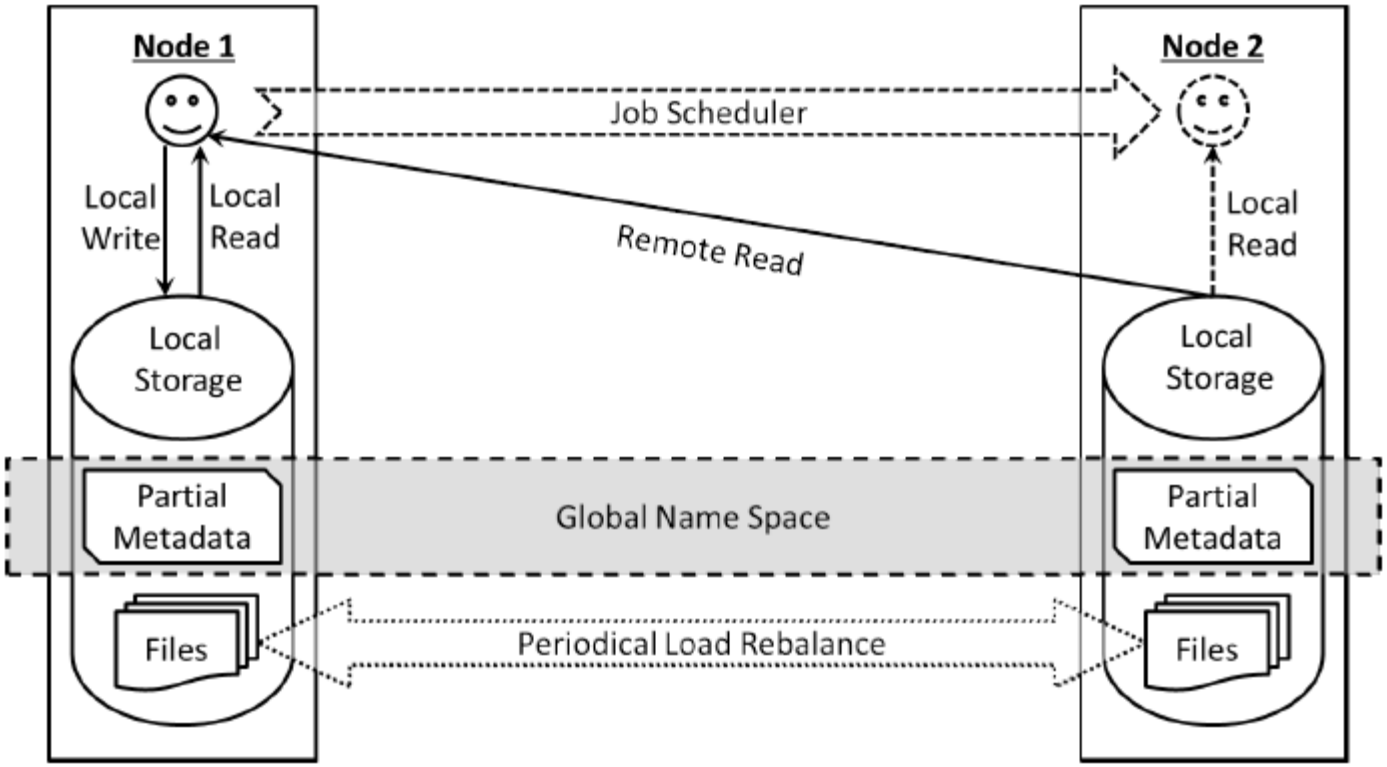
since all parameters are positive. And because R is an integer, the optimal R is:

$$\arg\max_R F(R) = \begin{cases} \lfloor \hat{R} \rfloor & \text{if } F(\lfloor \hat{R} \rfloor) > F(\lceil \hat{R} \rceil) \\ \lceil \hat{R} \rceil & \text{otherwise} \end{cases}$$

Variable	Description
B_r	Read Bandwidth
B_w	Write Bandwidth
W_i	Weight of input
W_o	Weight of output
S	File Size
N	Number of Data Entries
R	Number of Reference Points

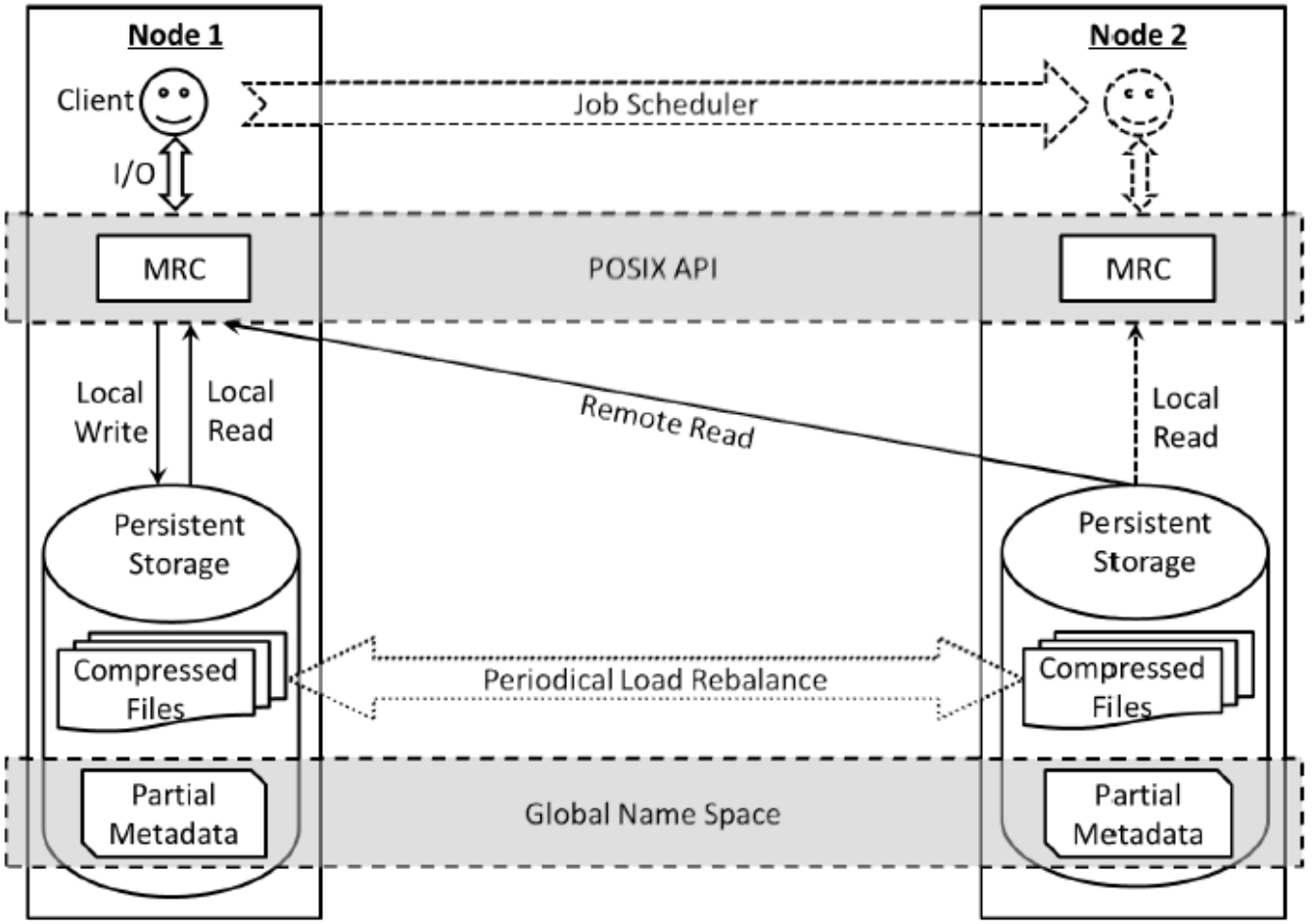
Design

- FusionFS architecture



Design

- Integrate MRC into FusionFS

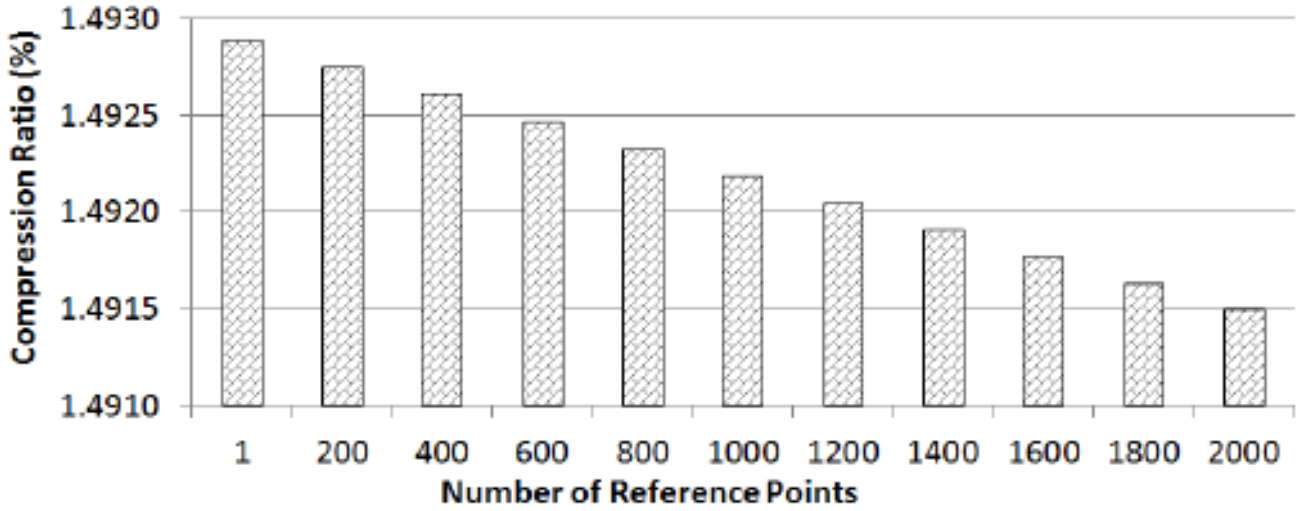


Implementation

- We illustrate the effectiveness of MRC with an exemplary compression method: XORing every pair of data points
- Compression is implemented in fusion_write()
- Decompression is implemented in fusion_read()

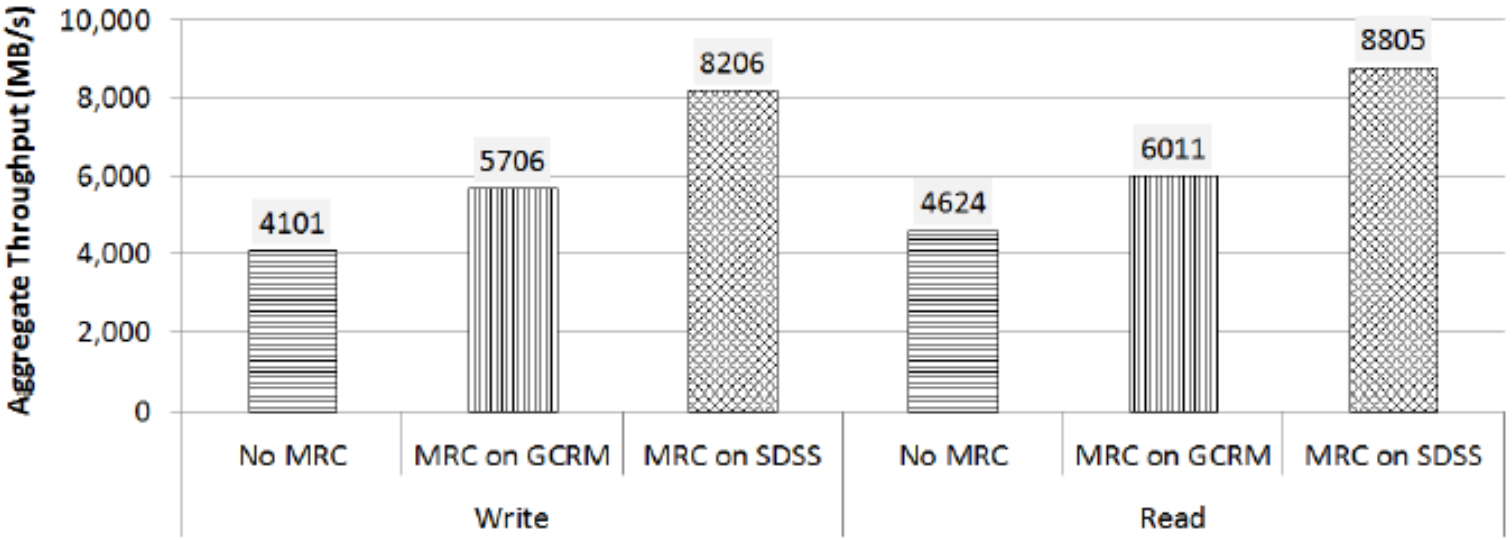
Evaluation

- Compression ratio



Evaluation

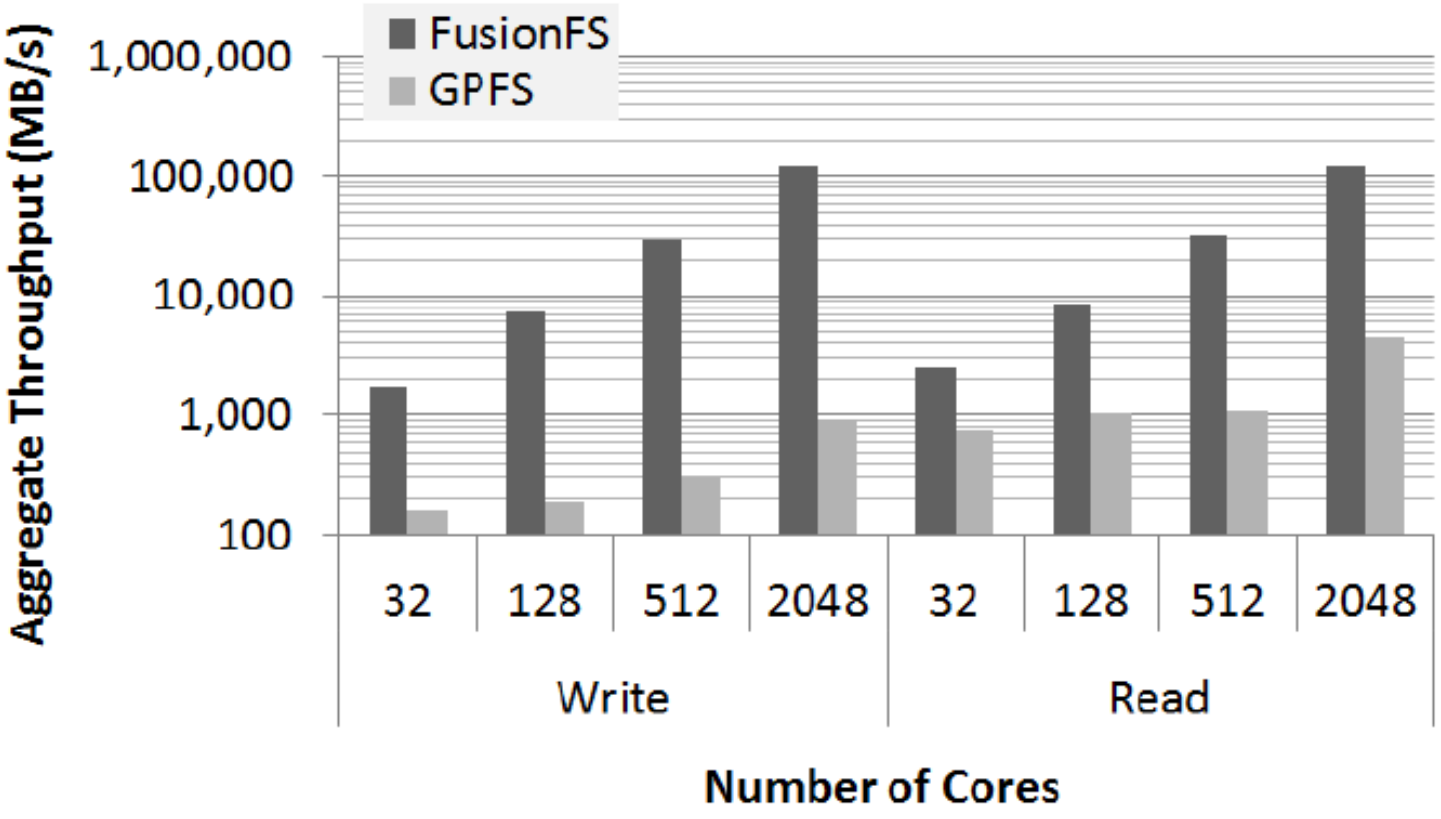
- Throughput on a 64-node Linux cluster



The global cloud resolving model. http://kiwi.atmos.colostate.edu/gcrm/.
SDSS Query. http://cas.sdss.org/astrodr6/en/help/docs/realquery.asp.

Evaluation

- FusionFS (MRC enabled) vs. GPFS on Intrepid



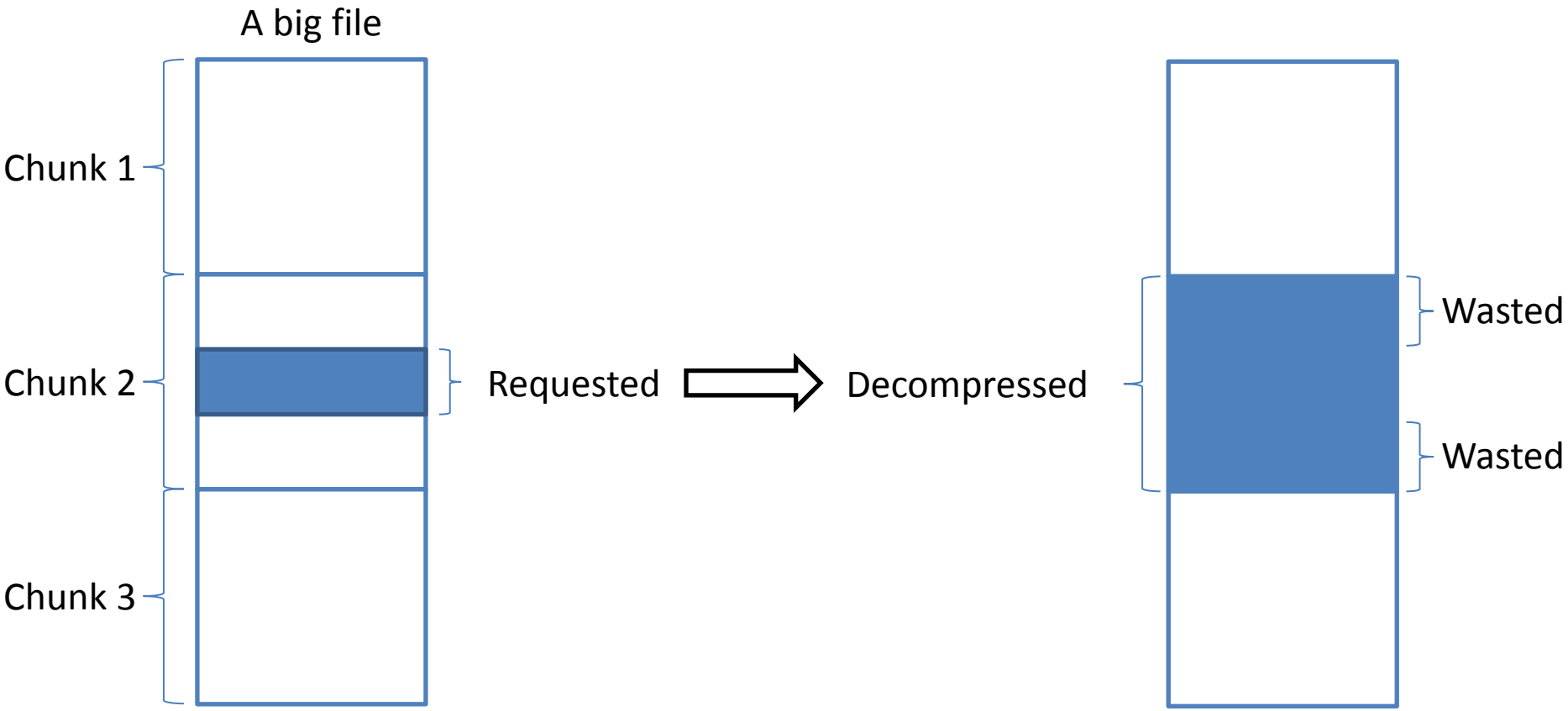
Conclusion

- MRC approach could help reduce the computational overhead in data compressions of distributed file systems by adding negligible space
- Our experiments show a 2X speedup on end-to-end I/O throughput at 2048-core scale
- We plan to improve the static equidistant partitioning into a dynamic strategy

Thanks

- For more information, send me an email at: dzhao8@hawk.iit.edu
- Questions?

Motivation - big chunks



Motivation – small chunks