Pattern-Aware File Reorganization in MPI-IO

Jun He[1]**,** Huaiming Song[1], Xian-He Sun[1], Yanlong Yin[1], Rajeev Thakur[2]

1: Illinois Institute of Technology, Chicago, Illinois

2: Argonne National Laboratory, Argonne, Illinois

ILLINOIS INSTITUTE OF TECHNOLOGY

Argonne NATIONAL LABORATORY

Motivation

Two Important Factors in Parallel File Systems:

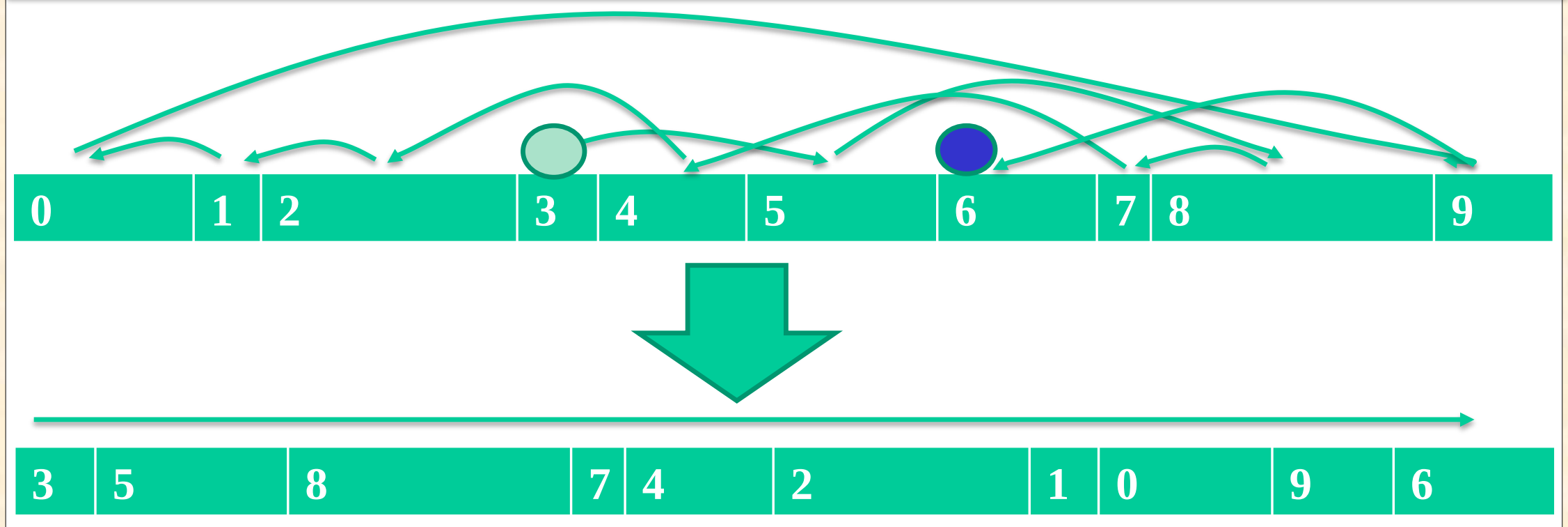
- Number of requests
- Contiguousness of accesses

One Mismatch:

- **Logical data**
  Developer's understanding, for programmability and runtime performance.
  -> Logical organization -> Access pattern
- **Physical data**
  The data blocks are stored on disk.
  -> Physical data organization

Good logical organization
!=
Good physical organization for better I/O performance

An Example

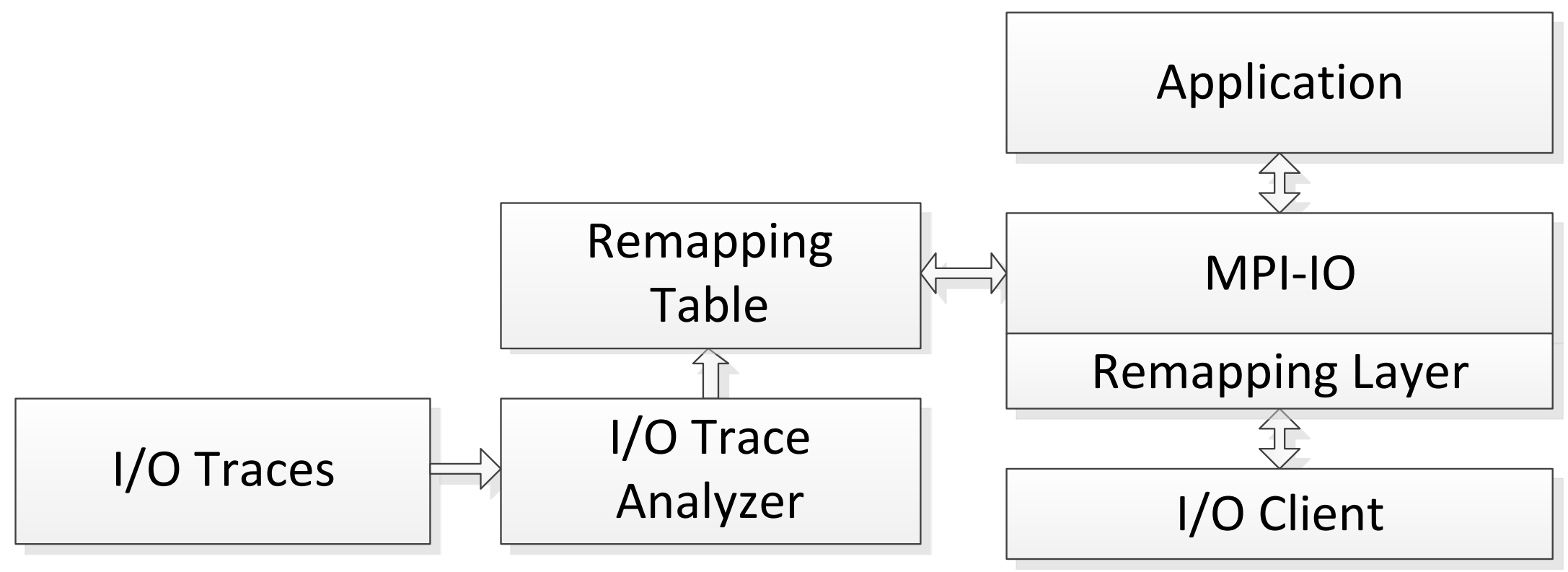


The Idea

- Be aware of repeating non-contiguous access patterns.
- Try to reorganize the data so that data is contiguous.

Design

System Overview



Application

Remapping Table

MPI-IO

Remapping Layer

I/O Traces

I/O Trace Analyzer

I/O Client

Trace Collecting

- Wrap the original function call
- Get: process ID, MPI rank, file descriptor, type of operation, offset, length, data type, time stamp, and file view

Pattern Classification

Spatial Pattern

- Contiguous
- Non-contiguous
  - Fixed strided
  - 2d-strided
  - Negative strided
  - Random strided
  - kd-strided
- Combination of contiguous and non-contiguous patterns

Request Size

- Fixed
- Variable
- Small
- Medium
- Large

Repetition

- Single occurrence
- Repeating

Temporal Intervals

- Fixed
- Random

I/O Operation

- Read only
- Write only
- Read/write

I/O Trace Analyzer

I/O Signature

{I/O operation, initial position, dimension, ([{offset Pattern}, {request size pattern}, {pattern of number of repetitions}, {temporal pattern}], [...]), # of repetitions}

Pattern matching
•Sort Traces by time
•Separate by process
•Find out patterns

I/O-signature-based Remapping Table

Old	New
File, {MPI_READ, offset0, 1, ([(hole size, 1), LEN, 1]), 4}	Offset0'

MPI-IO Remapping Layer

Convert old offsets to new ones

Read ***m*** bytes data from offset ***f***.

Whether this access falls in a 1-d strided pattern ?

starting offset ***off***, read size ***rsz***, hole size ***hsz***, number of accesses of this pattern ***n***

$$(f\text{-}off)/(rsz+hsz) < n \qquad (1)$$

$$(f\text{-}off)\%(rsz+hsz) = 0 \qquad (2)$$

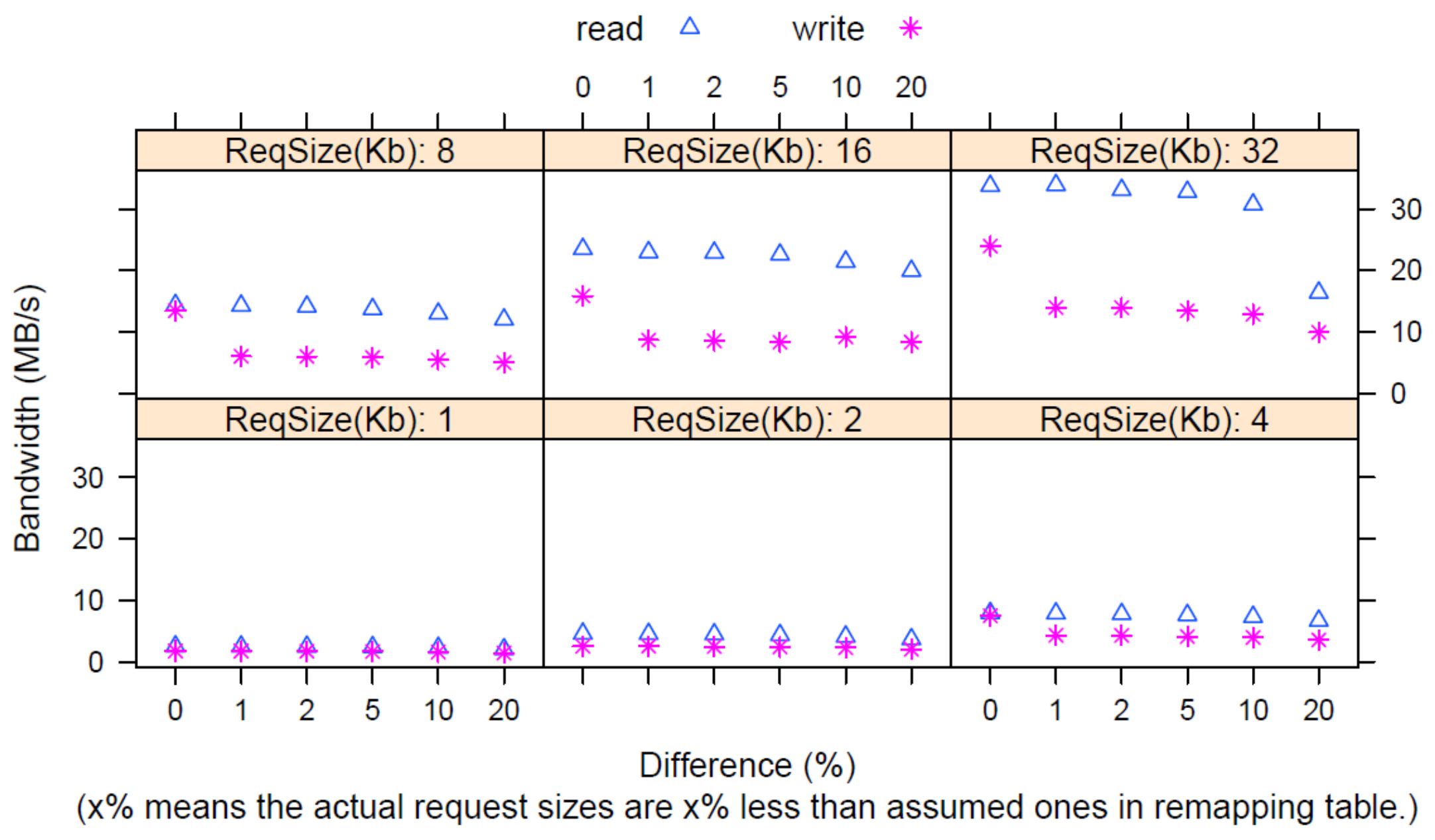
$$m = rsz \qquad (3)$$

Evaluations

Remapping Overhead

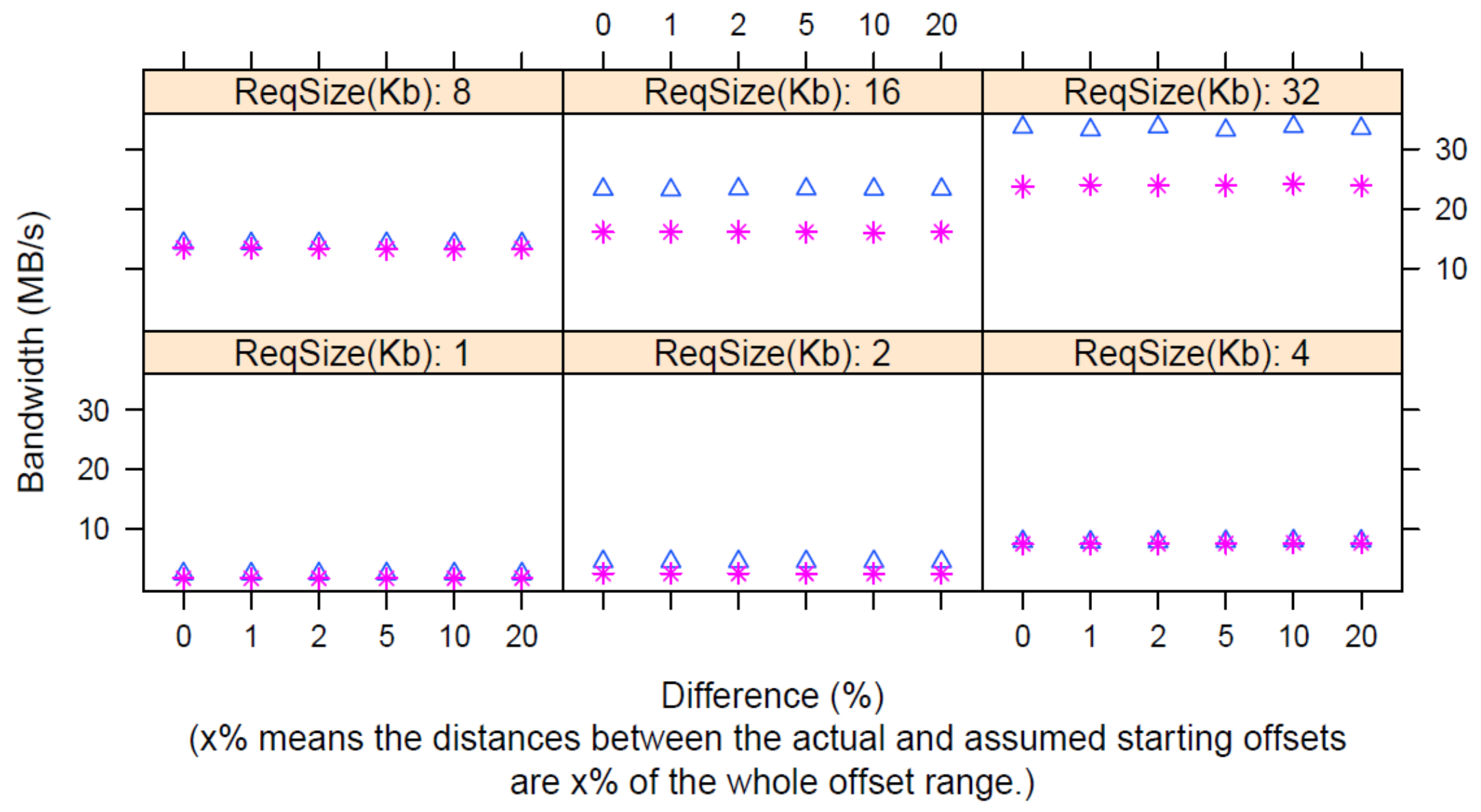
Table Type	Size (bytes)	Building time (sec)	Time of 1,000,000 lookups (sec)
1-to-1	64,000,000	0.780287	0.489902
I/O Signature	**28**	**0.000000269**	**0.024771**

Request Size Variation



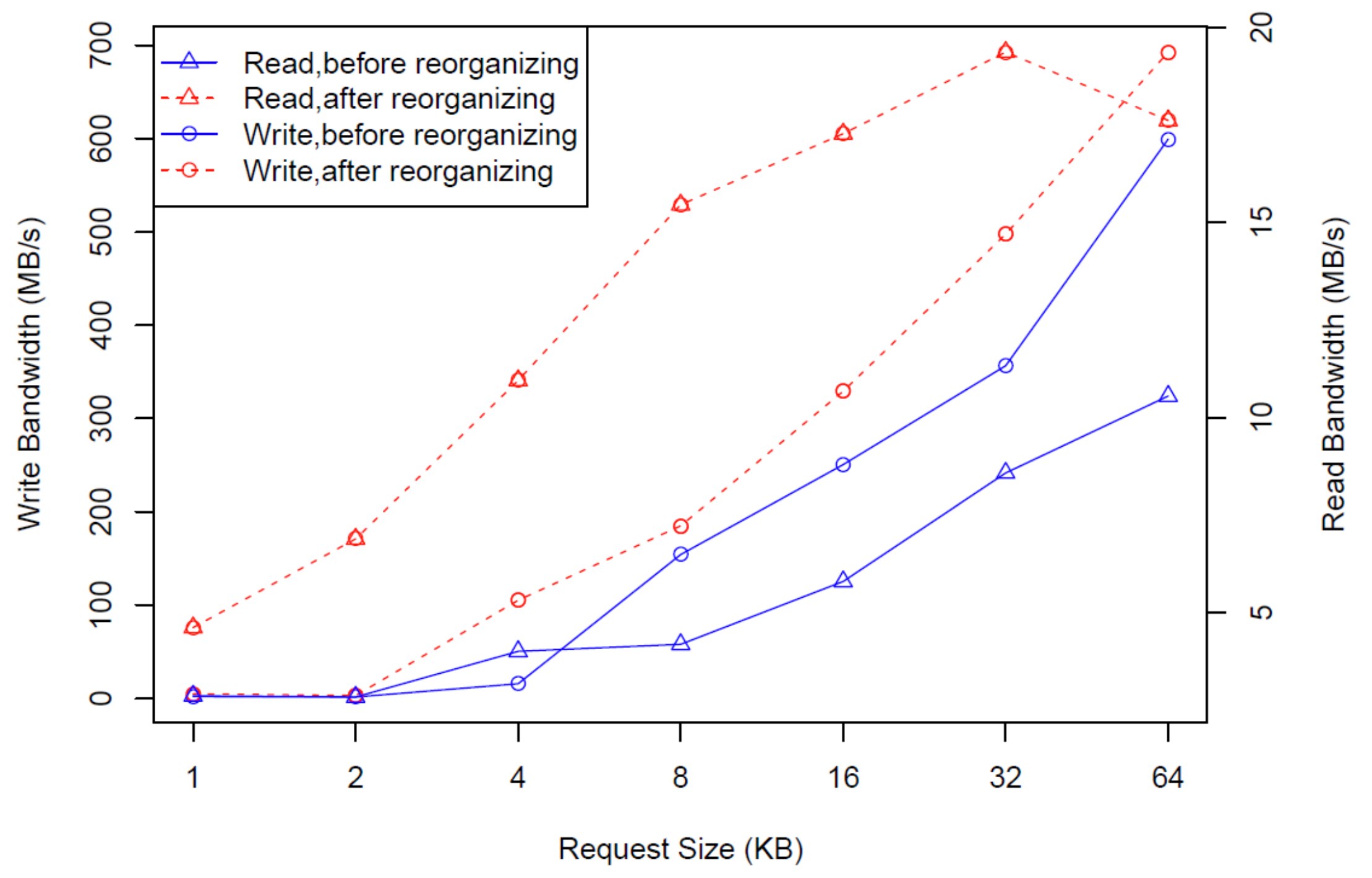
(x% means the actual request sizes are x% less than assumed ones in remapping table.)

Starting Offset Variation



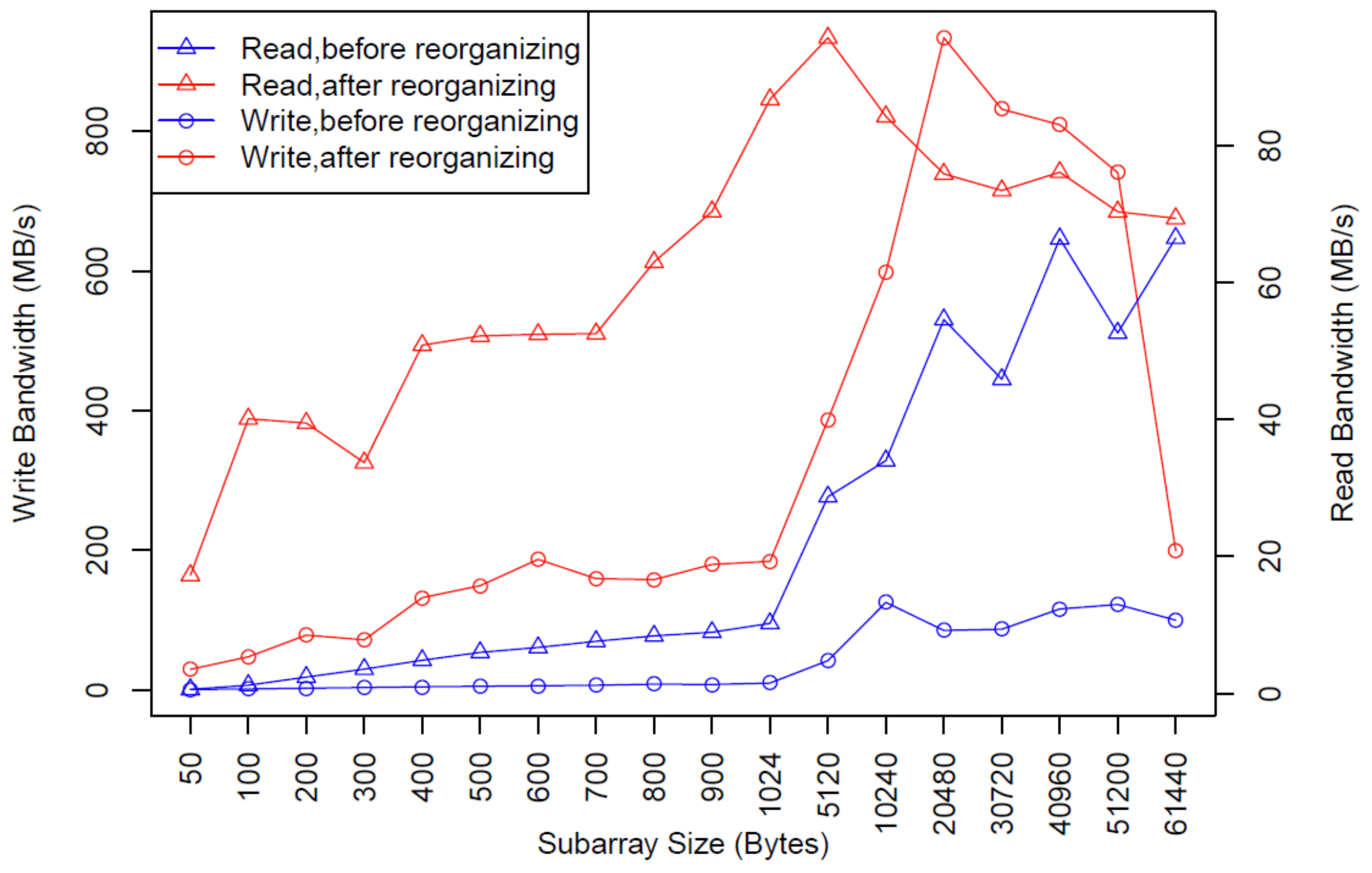
(x% means the distances between the actual and assumed starting offsets are x% of the whole offset range.)

IOR Performance



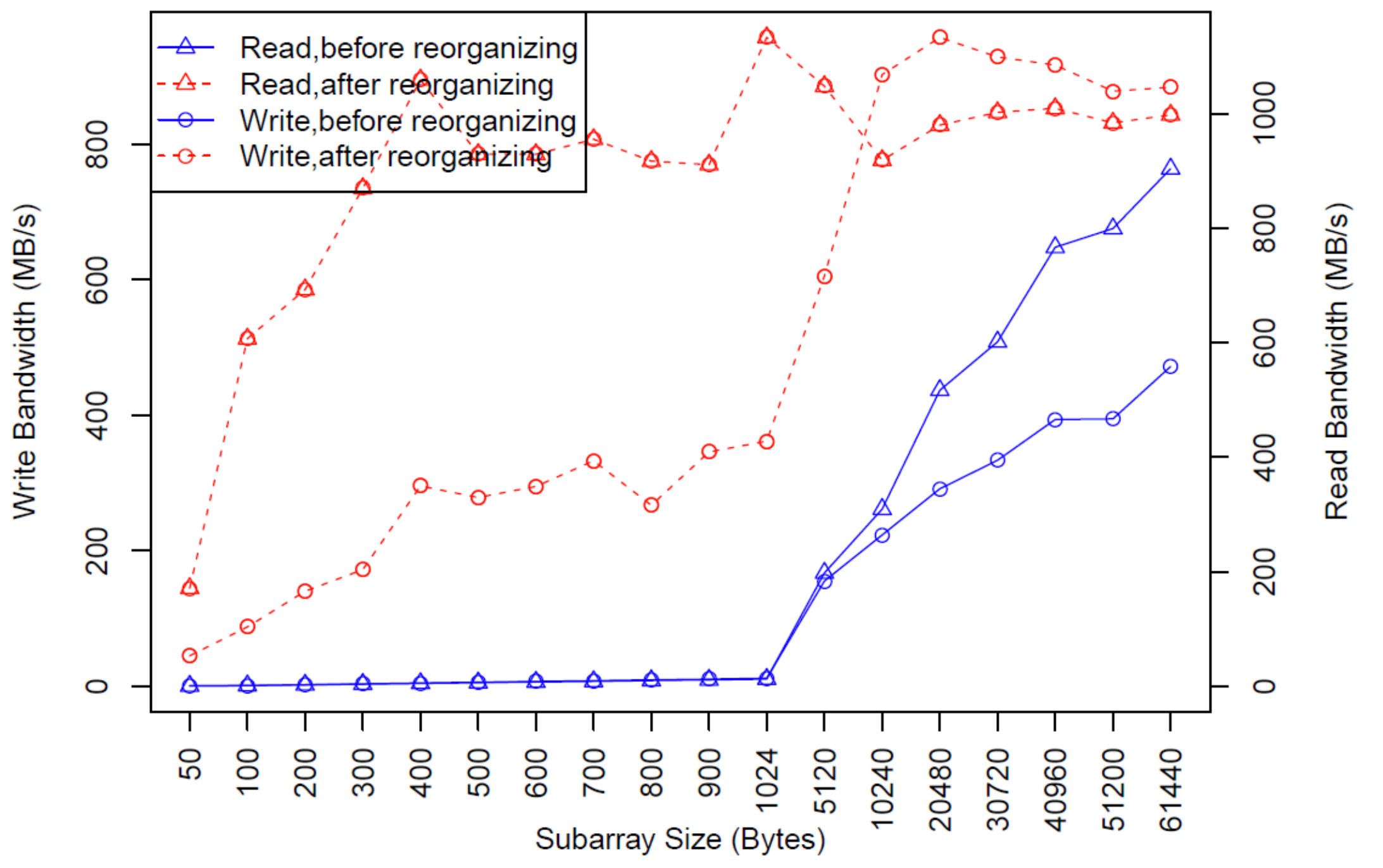
64 processes with HDD and Infiniband

MPI-TILE-IO



64 processes with HDD and Infiniband

MPI-TILE-IO with SSD



64 processes with SSD and Infiniband

Conclusions

Different file organizations lead to very different performance.

Bridging logical data and physical data

Access pattern
-> better organization
-> better performance

Acknowledgement

Hui Jin and Spenser Gilliland (Illinois Institute of Technology), Ce Yu (Tianjin University, China), Samuel Lang (Argonne National Laboratory)

NSF grant CCF-0621435, CCF-0937787, Office of Advanced Scientific Computing Research, Office of Science, U.S. DOE, under Contract DEAC02-06CH11357.