


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

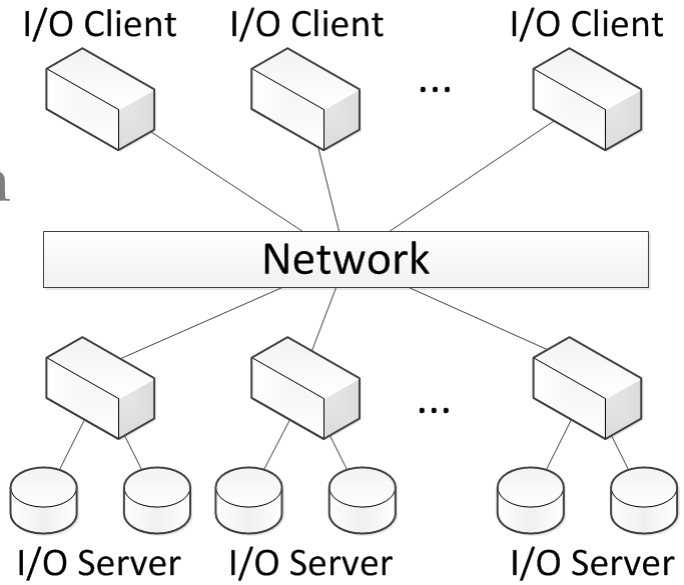
Outline

- Motivation
  - Examples
  - Basic idea
- Design
  - System Overview
  - Trace collecting
  - Pattern classification
  - I/O Trace analyzer
  - Remapping table
  - MPI-IO remapping layer
- Evaluation
  - Remapping overhead
  - Pattern variation
  - Benchmarks
- Conclusion & Future Work

Motivation

Parallel File Systems

A typical parallel file system



- Important Factors
  - **Number of requests**
  - **Contiguousness of accesses**

Network overhead
IOPS
Locality
…

Mismatch

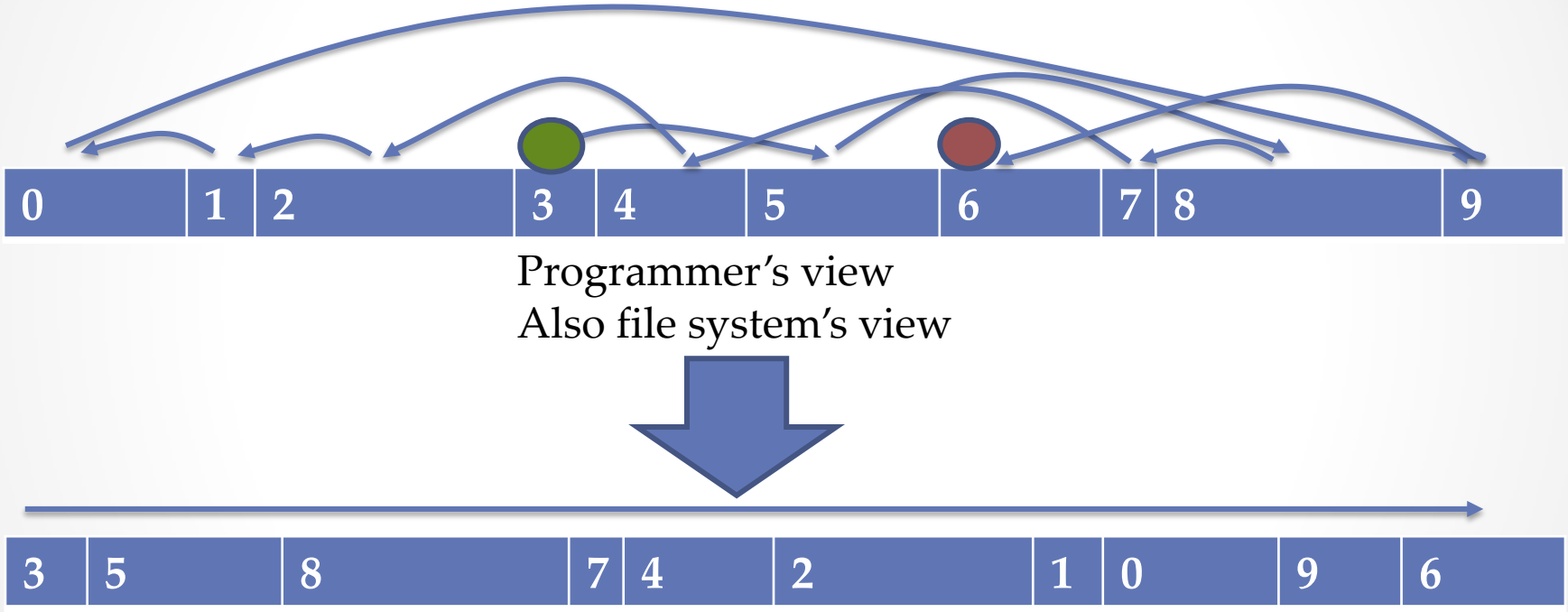
- Logical data
  - Developer's understanding, for programmability and runtime performance
  - -> Logical organization -> Access pattern
- Physical data
  - Where the data blocks are stored
  - -> Physical data organization

Good logical organization

!=

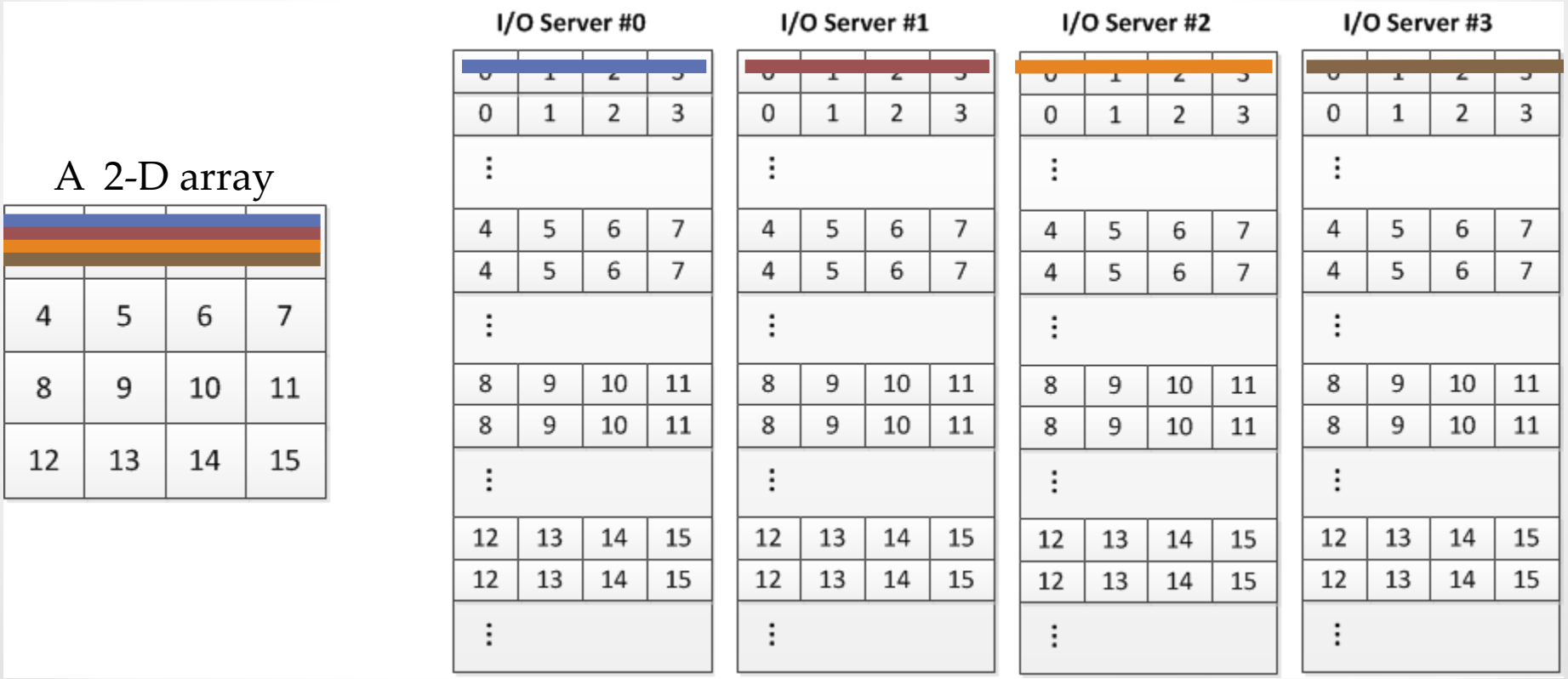
Good physical organization for better I/O performance

A Tiny Example for Irregular Data



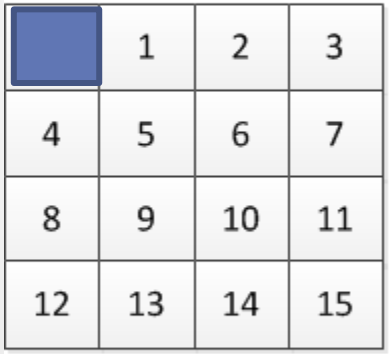
Potential benefit:
Better spatial locality
Easier for some optimization to take effect
Less disk head movements
…

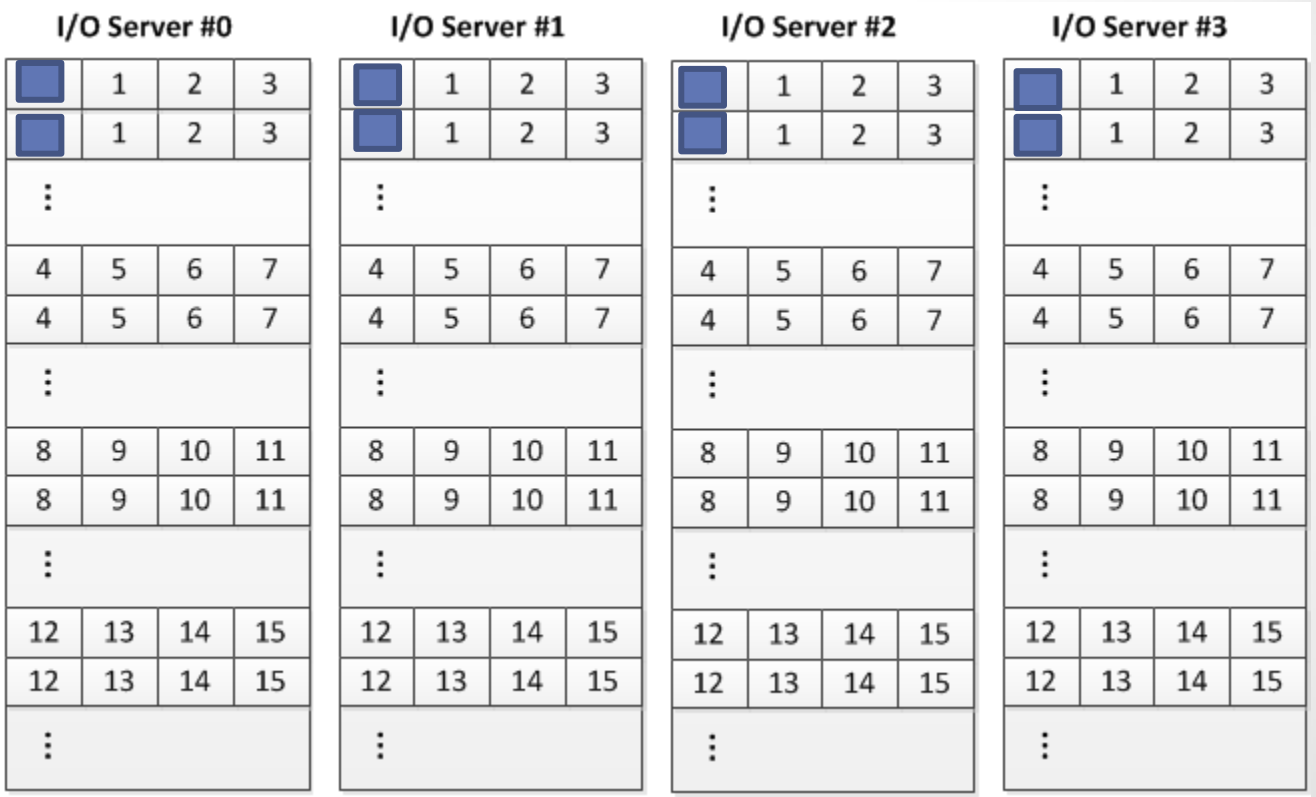
An Example for Regular 2-d Array Default Organization



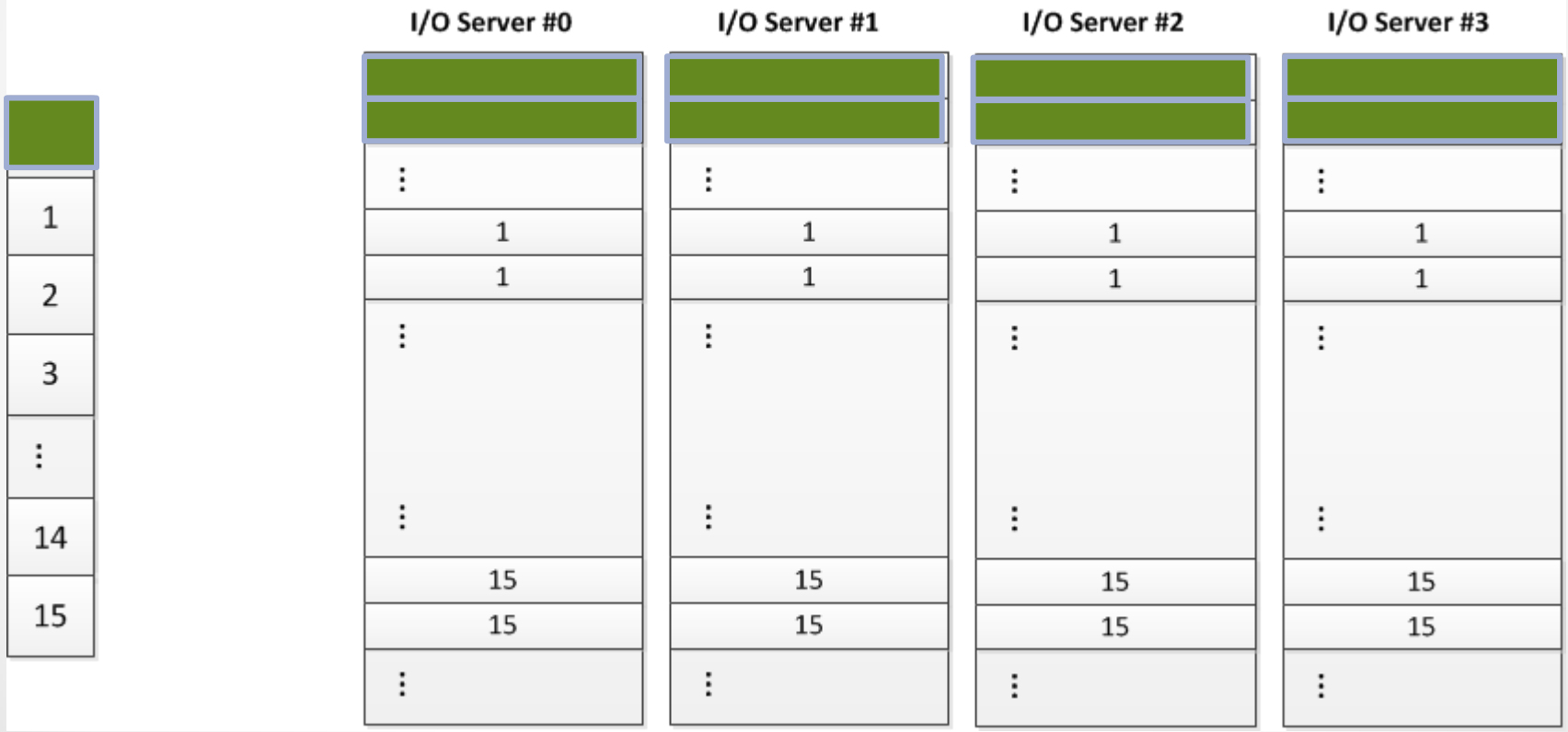
Read a Subarray

A 2-D array

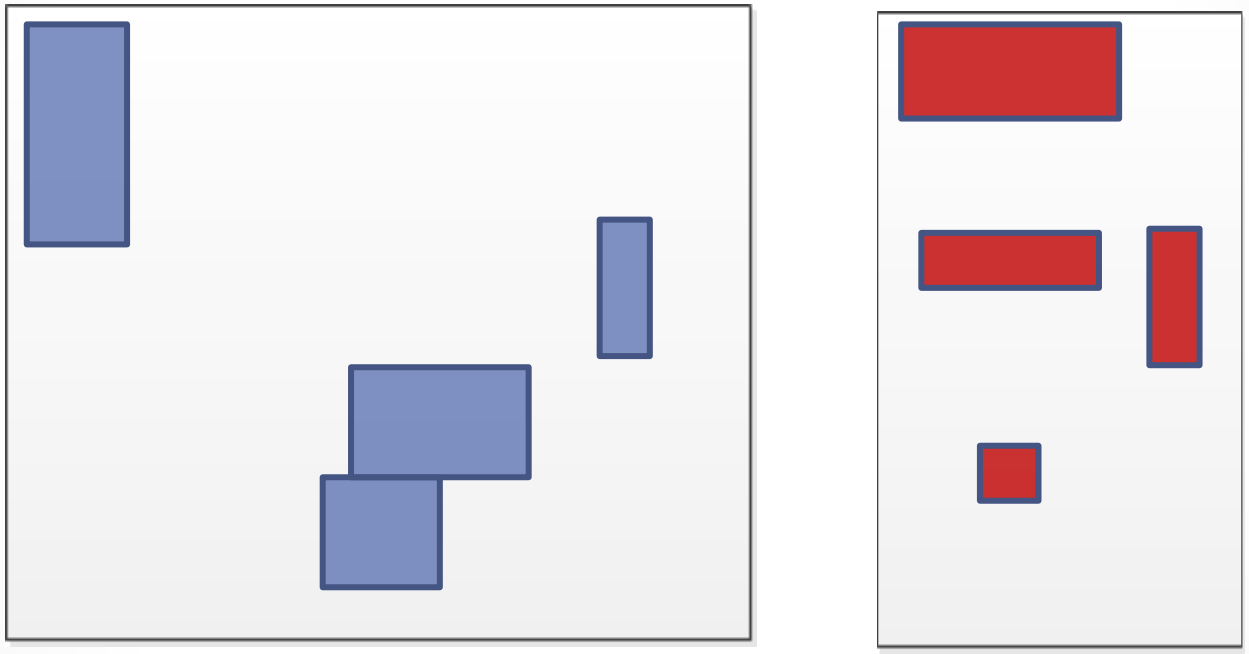




After Re-organizing



A Messier One



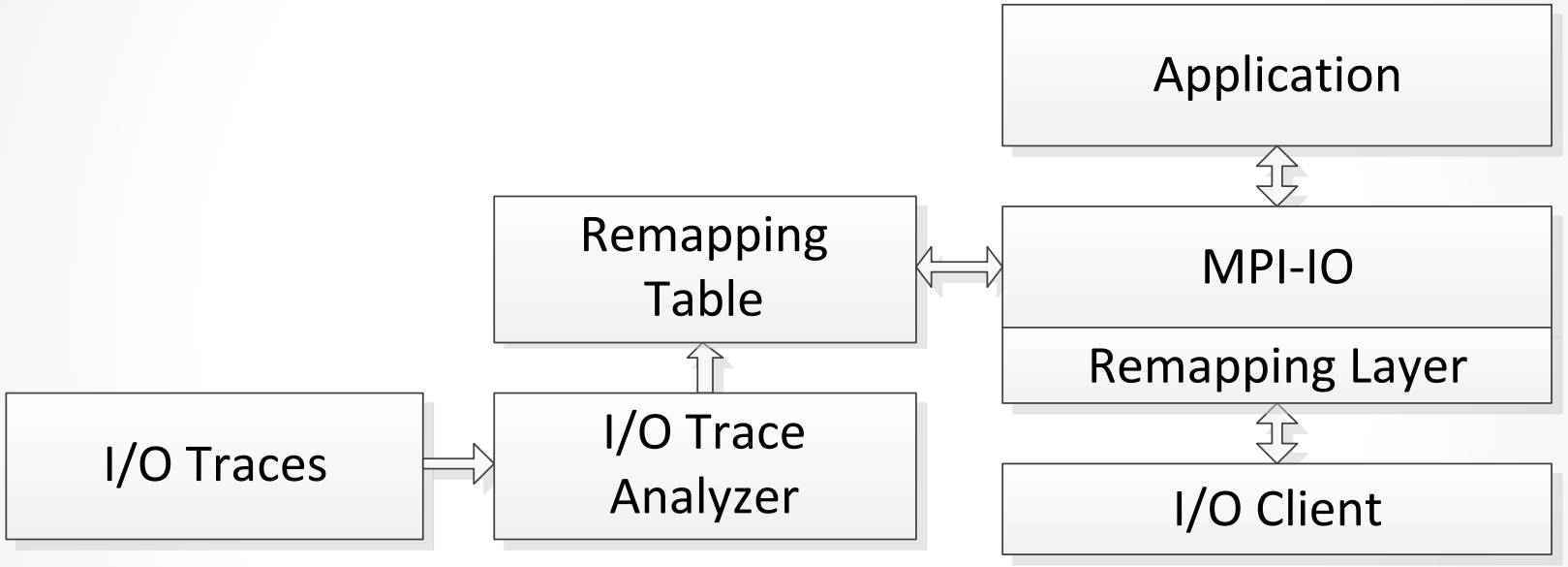
- Irregular data
- Very complex data model
- Computation which involves multiple data fields

Pattern-Aware Reorganization

- Be aware of repeating non-contiguous access patterns
  - n-d strided and irregular
- Try to reorganize the data so that data is contiguous.
  - Less network overhead
  - Less IO operations
  - Better locality
  - Beneficial for other optimizations, e.g. data sieving…
- Motivating Scenarios
  - Application start-up
  - Data analysis, visualization
  - …
- Where it does not apply
  - Patterns do not repeat from run to run.

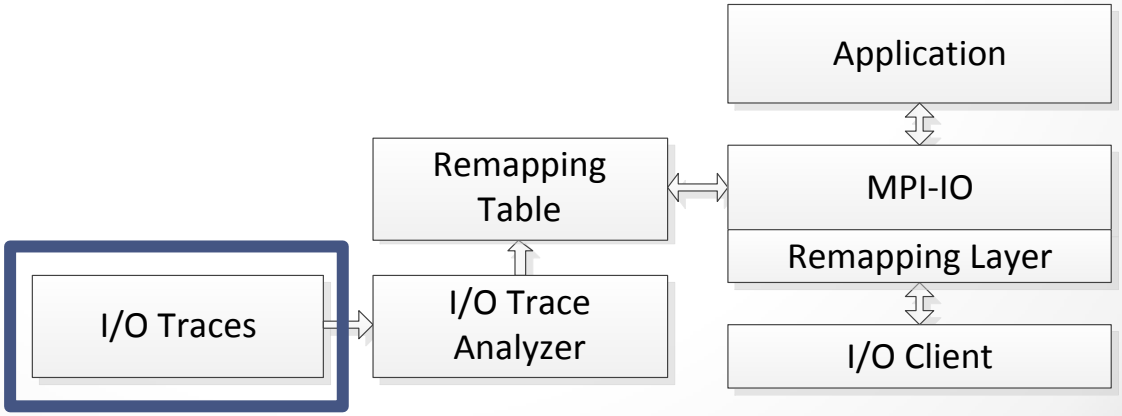
Design

System Overview



Trace Collecting

- Wrap the original function call
  - Add recording function
  - Call original function inside
- Process ID, MPI rank, file path, 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

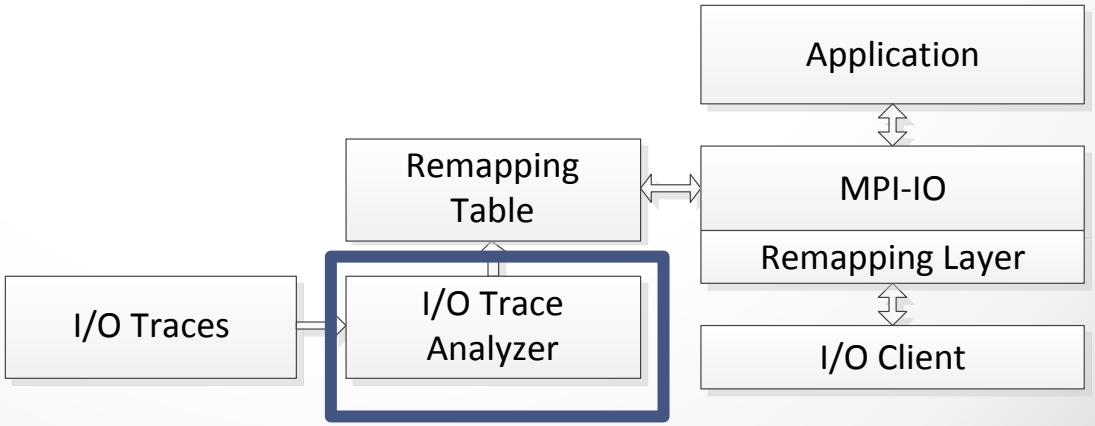
I/O Operation

- Read only
- Write only
- Read/write

I/O Trace Analyzer

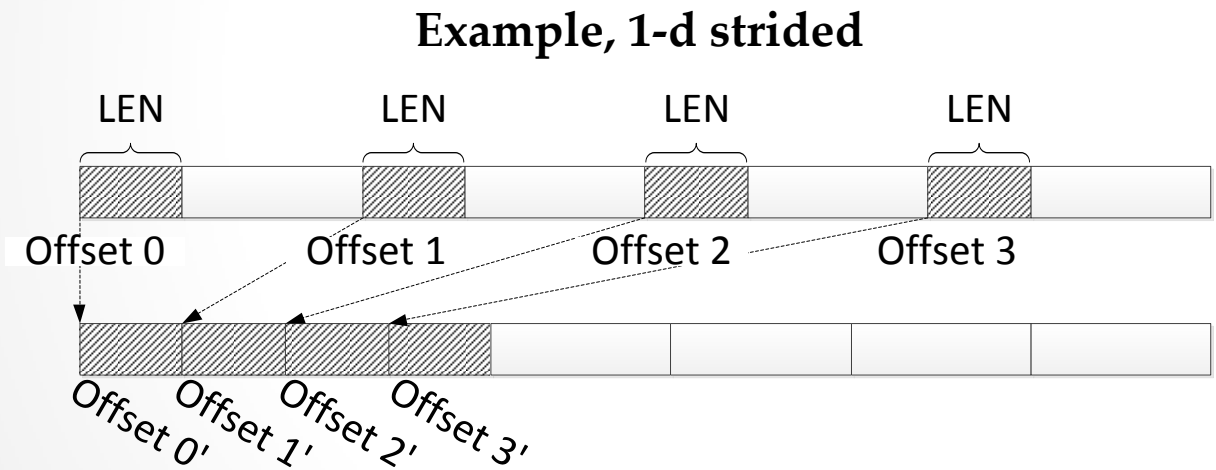
- Pattern matching
  - Sort Traces by time
  - Separate by process
  - Find out patterns
- I/O Signature

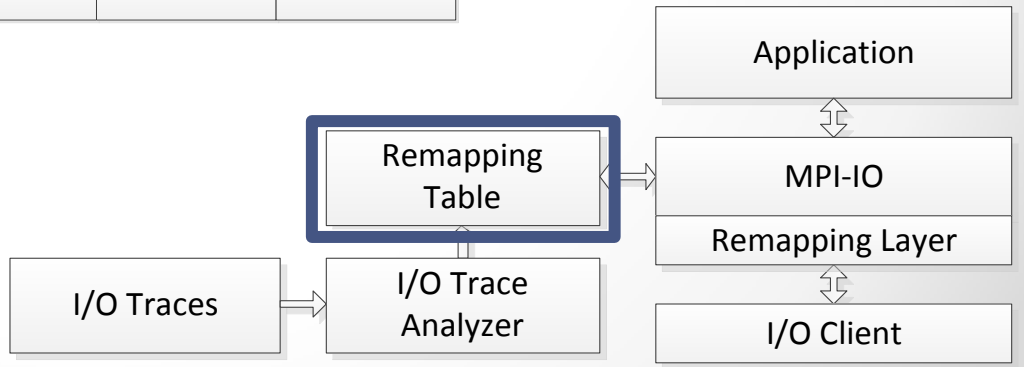
{I/O operation, initial position, dimension, ([{offset Pattern}, {request size pattern}, {pattern of number of repetitions}, {temporal pattern}], [...]), # of repetitions}



I/O-signature-based Remapping Table

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



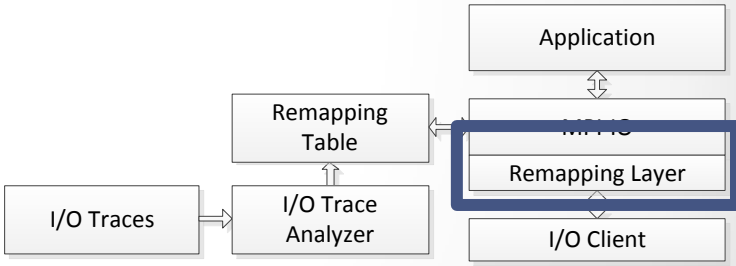


MPI-IO Remapping Layer

- *Convert old offsets to new ones*

Example:

- *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-off)/(rsz+hsz) <n* (1)
- *(f-off)%(rsz+hsz) = 0* (2)
- *m = rsz* (3)



$$newoff = off+rsz*(f-off)/(rsz+hsz)$$


Evaluation

• • •

Experiment Environment

- Dual 2.3GHz Opteron quad-core processors
- 8G memory
- 250GB 7200RPM SATA hard drive
- 100GB PCI-E OCZ Revodrive X2 SSD (read: up to 740 MB/s, write: up to 690 MB/s).
- Ethernet/Infiniband

- Ubuntu 9.04 (Linux kernel 2.6.28-11-server)
- **PVFS2** 2.8.1: stripe size 64 KB
- MPICH2 1.3.1

Remapping Overhead

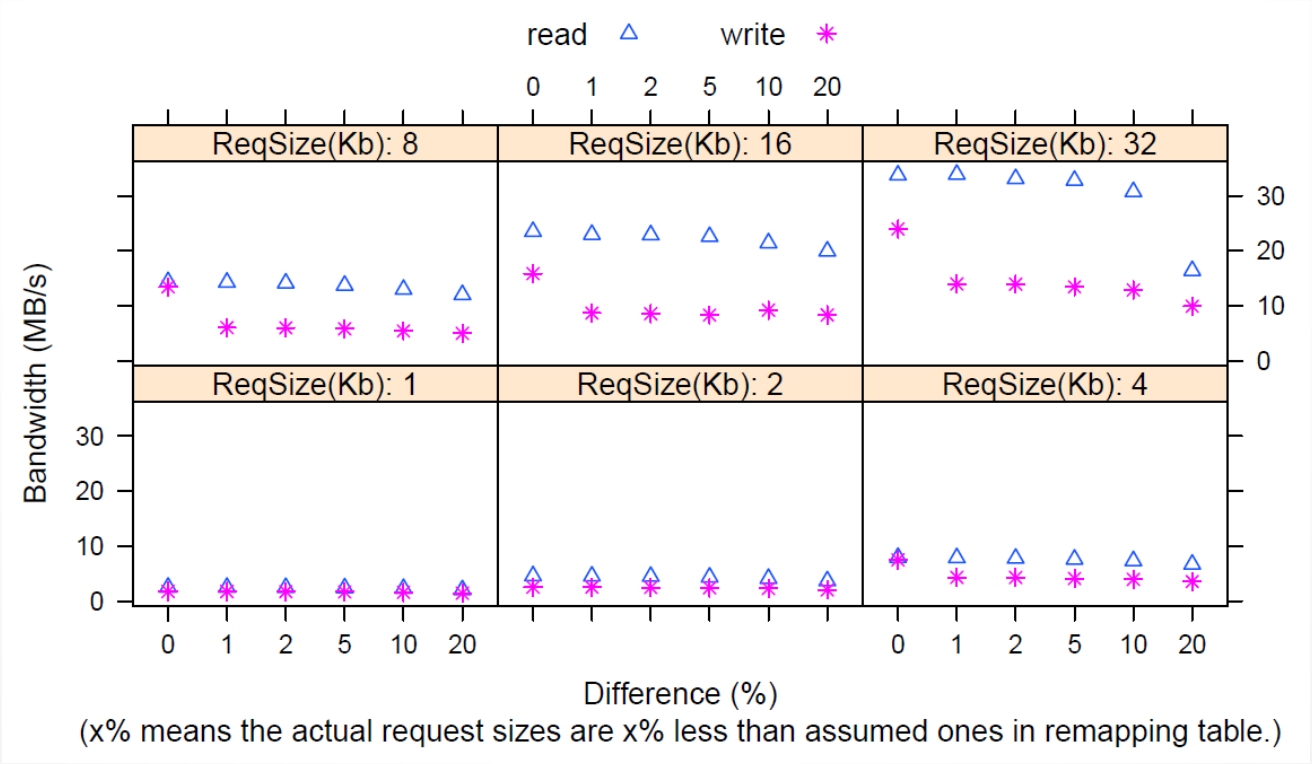
1-D Strided Remapping Table Performance (1,000,000 accesses)

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**

Who use 1-to-1:
PLFS uses 1-to-1 mapping table in index file.
Most OS file systems also use similar table to store free blocks in disk.

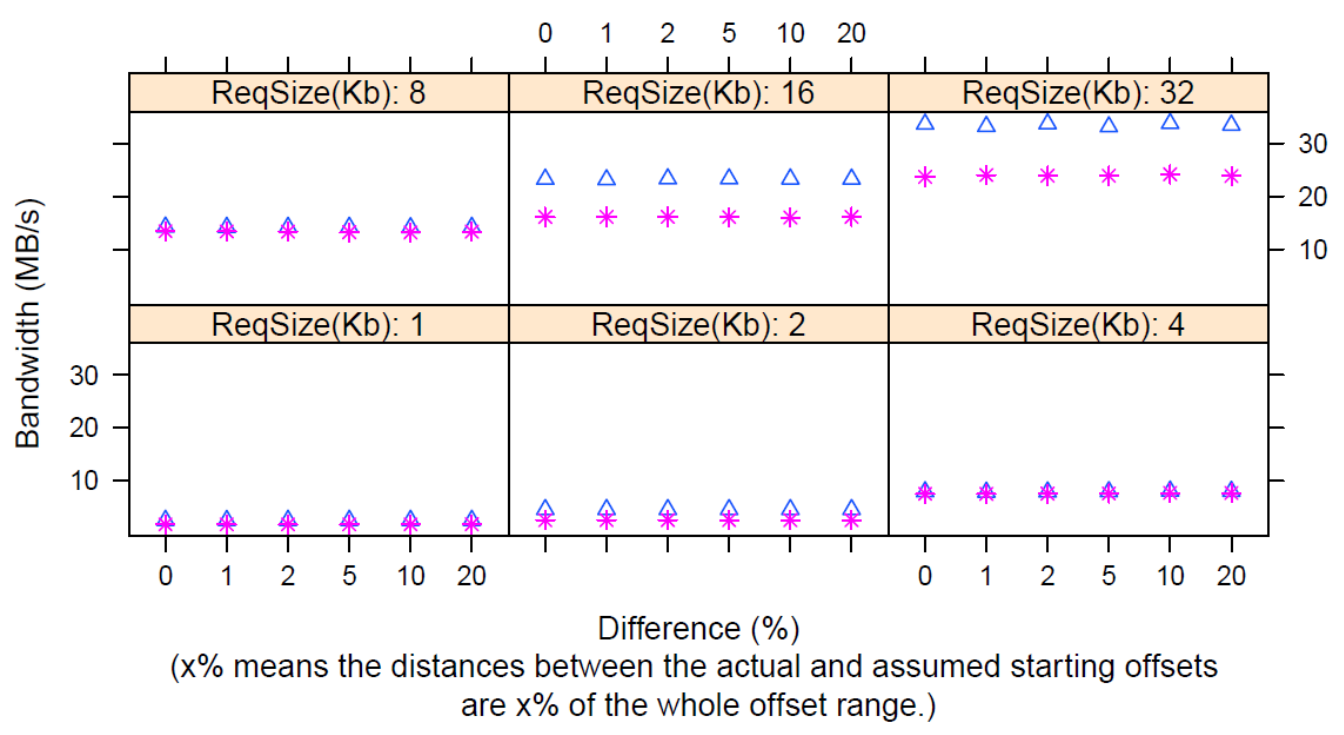
Request Size Variation

- X: different of request size. For example, 5% means the actual request size is 5% less than the one assumed.



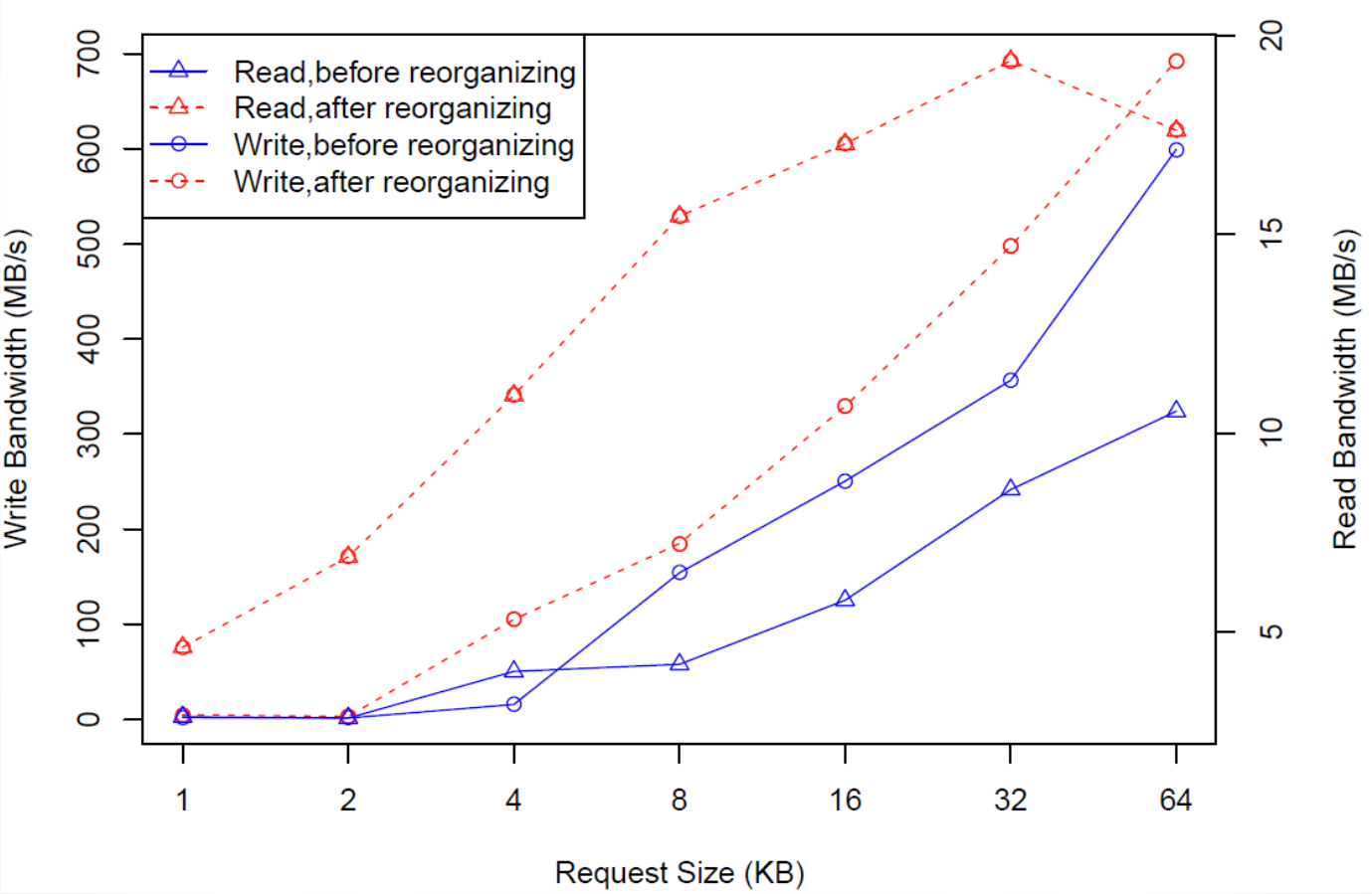
Variation of Starting Offset

- X: difference of starting offsets. 5% means that the starting offset moved to the 5%th of the whole access.



R/W Performance – on IOR

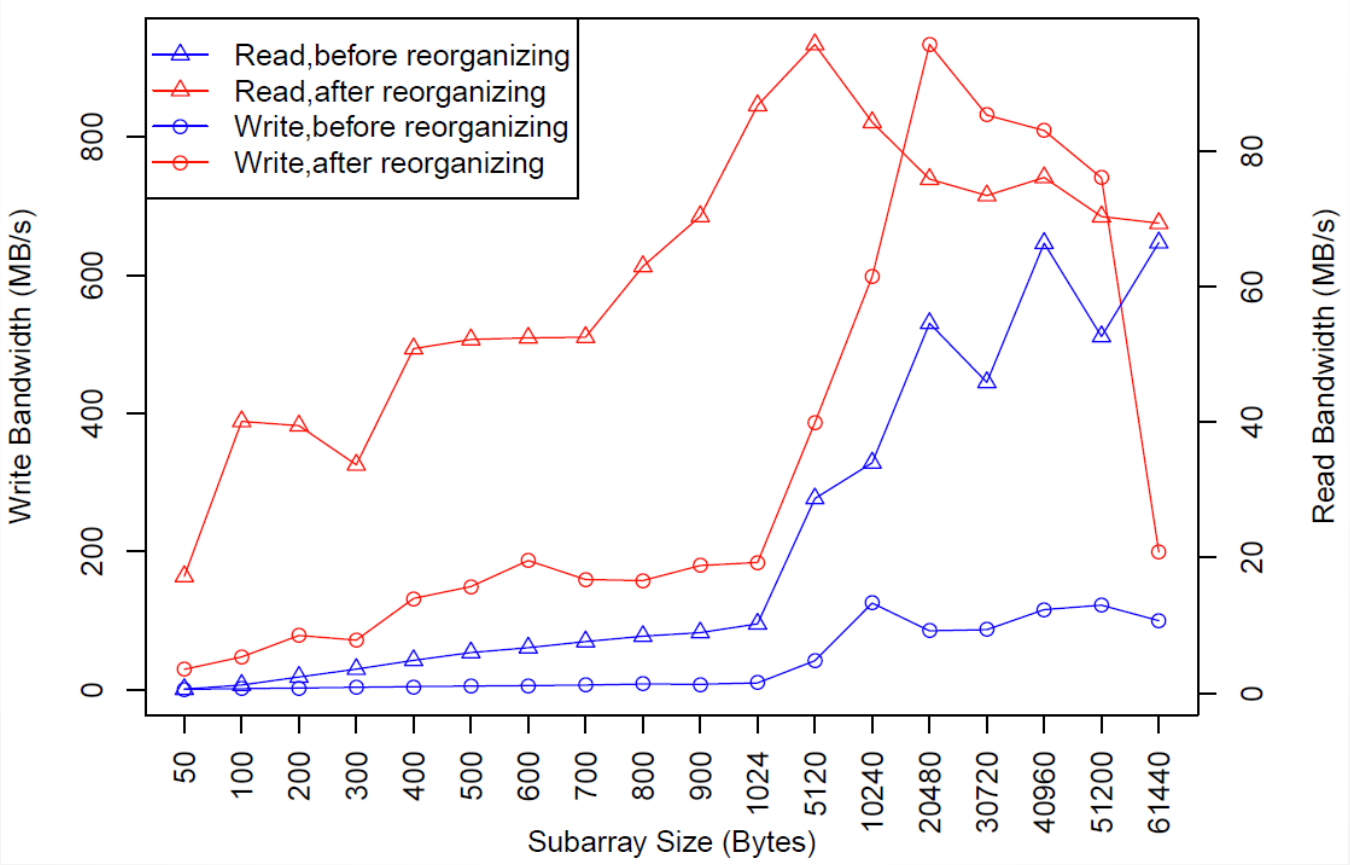
- 4 I/O clients, 4 I/O servers. 64 processes with HDD and Infiniband



- PDSW'11

Performance on MPI-TILE-IO

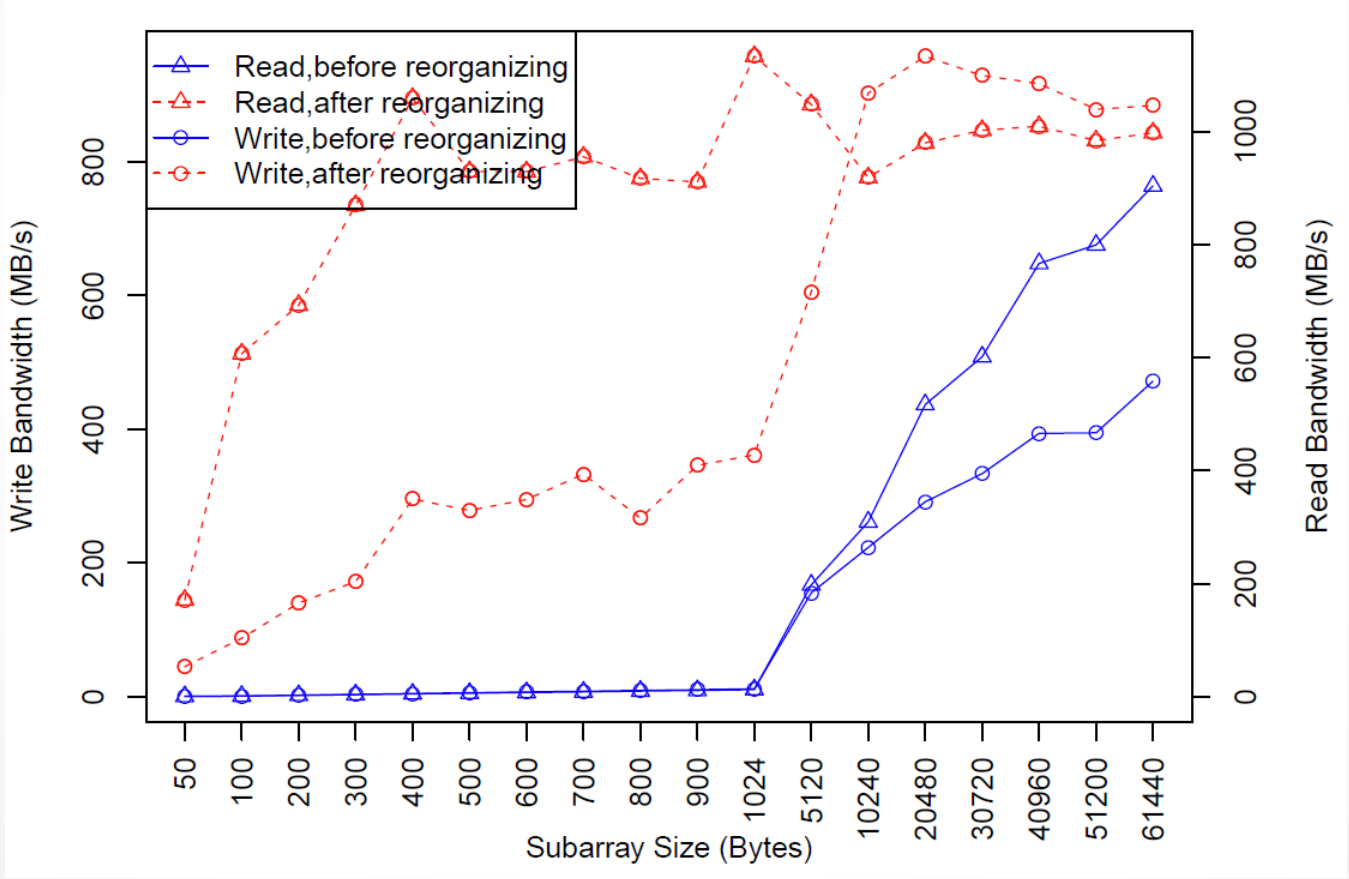
- 4 I/O clients, 4 I/O servers. 64 processes with HDD and Infiniband. Elements in a tile: 1024x1024.



- PDSW'11

Performance on MPI-TILE-IO with SSD

- 4 I/O clients, 4 I/O servers. 64 processes with SSD and Infiniband. Elements in a tile: 1024x1024.



- PDSW'11

Conclusion & Future Work

Conclusion

- Different file organizations lead to very different performance.
- Bridging logical data and physical data

  Access pattern
  -> better organization
  -> better performance

Future Work

- Multiple replicas with different organizations.
- More complicated access patterns, patterns with hints
- File reorganization for emerging storage medias, such as SSD

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-0937877
- Office of Advanced Scientific Computing Research, Office of Science, U.S. DOE, under Contract DEAC02-06CH11357.

Thanks!