Scalable Parallel I/O Systems by Client-Side Optimizations

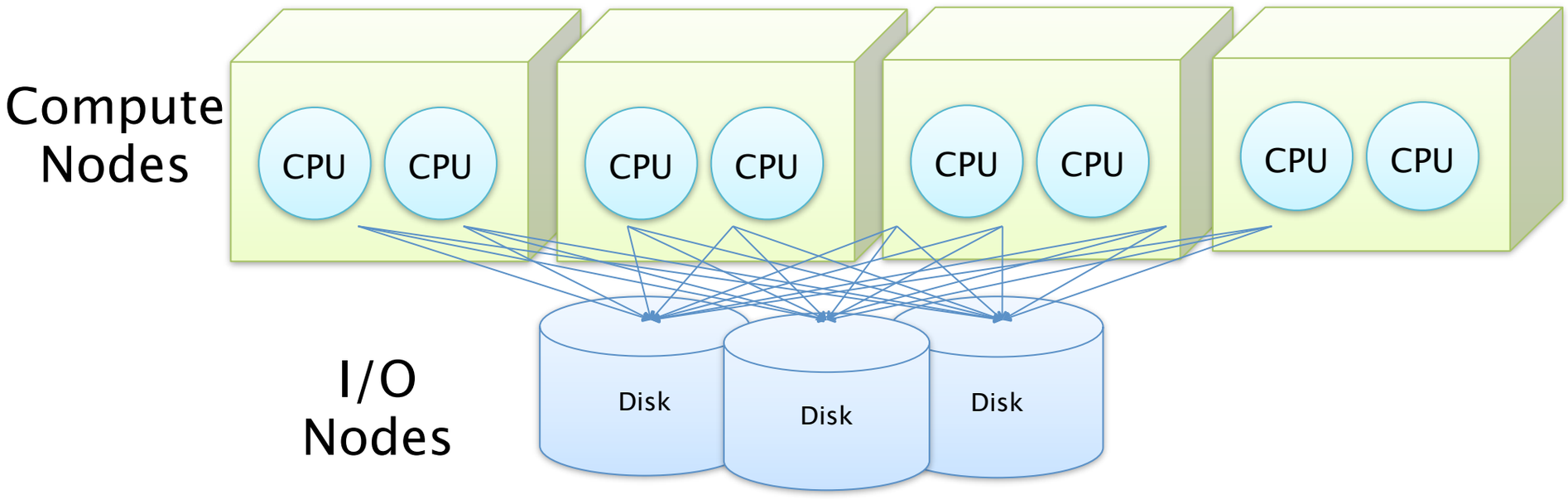
Kazuki Ohta
Yutaka Ishikawa

Background

- The increase of the number of cores
  - Multi-Core CPU is now commodity
  - The number of total cores in a cluster is also increasing
    - LANL RoadRunner (129600 core), ORNL Jaguar (150120 core), ANL BG/P (163840 core)
- The size of application data set is increasing
  - Faster CPUs, larger Disks
  - The ratio of CPU Speed/Disk Speed getting worse
  - I/O is now a major bottleneck in the whole applications

Parallel File Systems

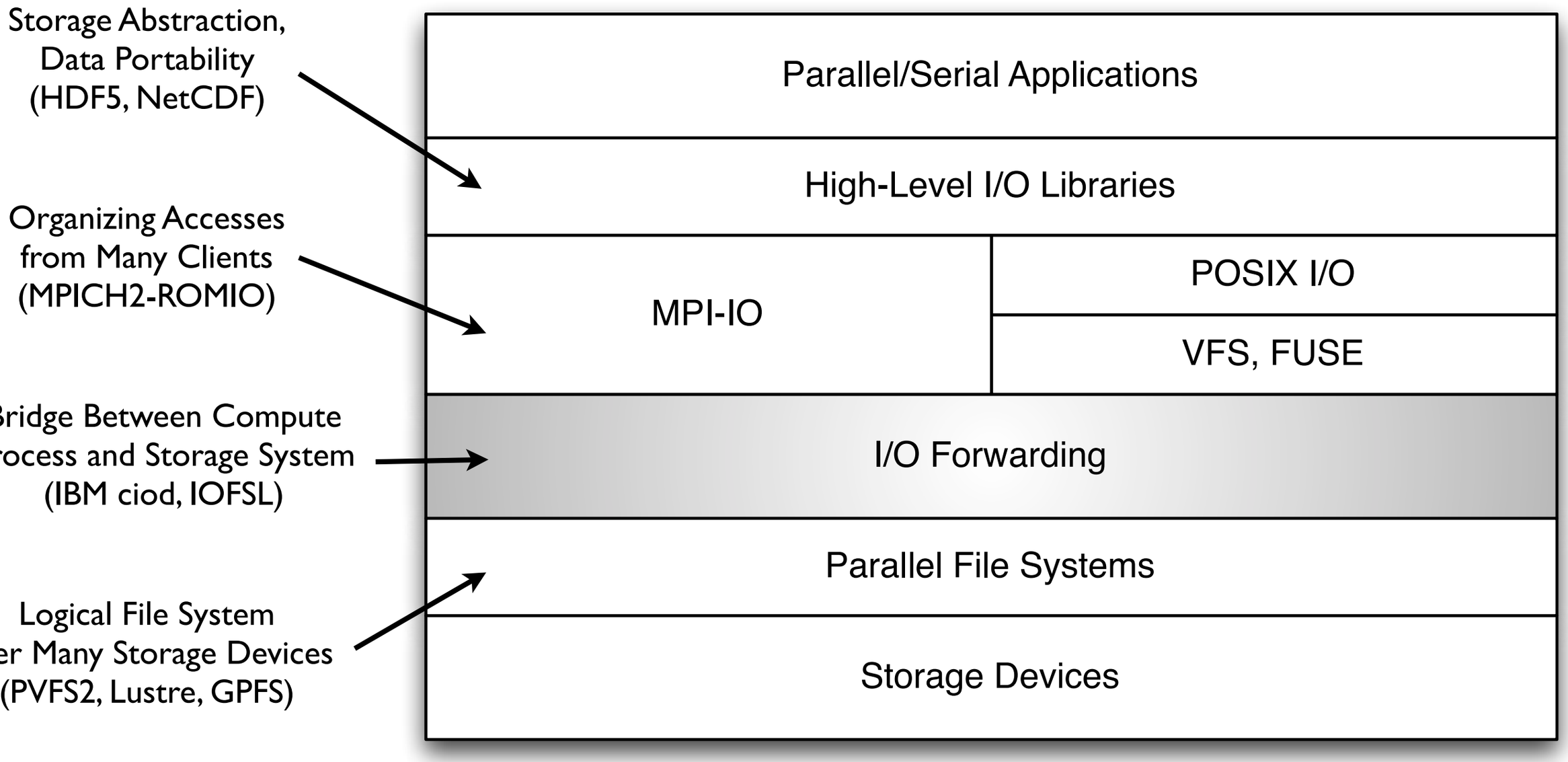
- Provide one logical file system over multiple disks (e.g, PVFS [Carns 2000], GPFS [Schmuck 2002], Lustre [Schwan 2003], etc)
- Files are splitted into fixed-size stripes, and distributed over the disks.



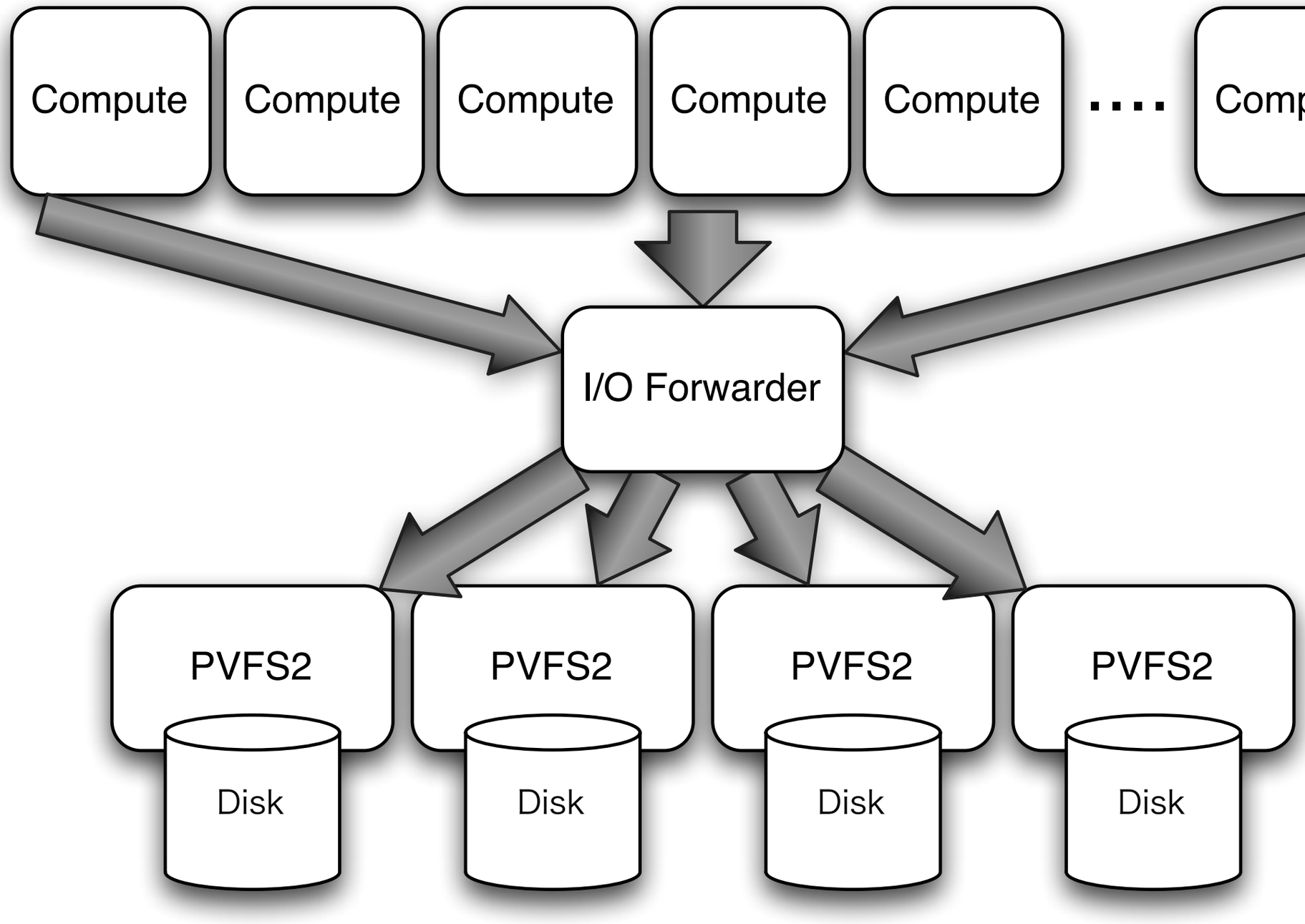
Problem: Scalability

- The Parallel File Systems cannot handle the I/O traffic from millions of processes
  - The burst access in I/O phase
  - The metadata performance problem
  - The locking problem
  - The increase of the disk seeks
    - Non-contiguous requests from the processes
    - Contiguous requests, but interleaved with each other
  - The congestion of the burst network traffic
  - etc.

Parallel I/O Software Stack



I/O Forwarding



The problem in I/O Forwarding

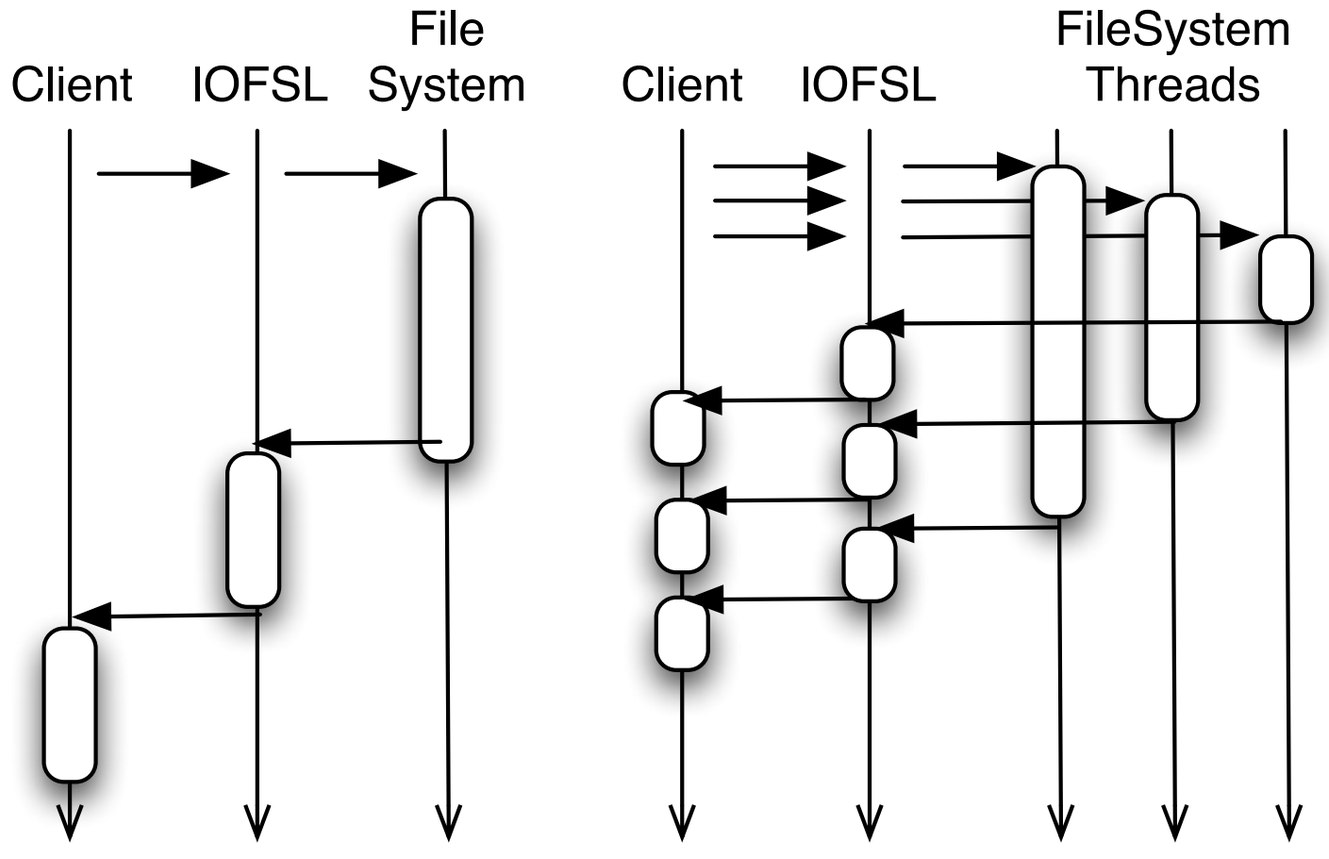
- Large Requests
  - Forward Latency
  - Variety of backend nodes performance
  - Memory limit of the forwarder
- Small Requests
  - Number of seeks at the file systems
  - Request processing overheads at the file systems

Out-Of-Order
Pipeline Transfer

Request
Scheduler

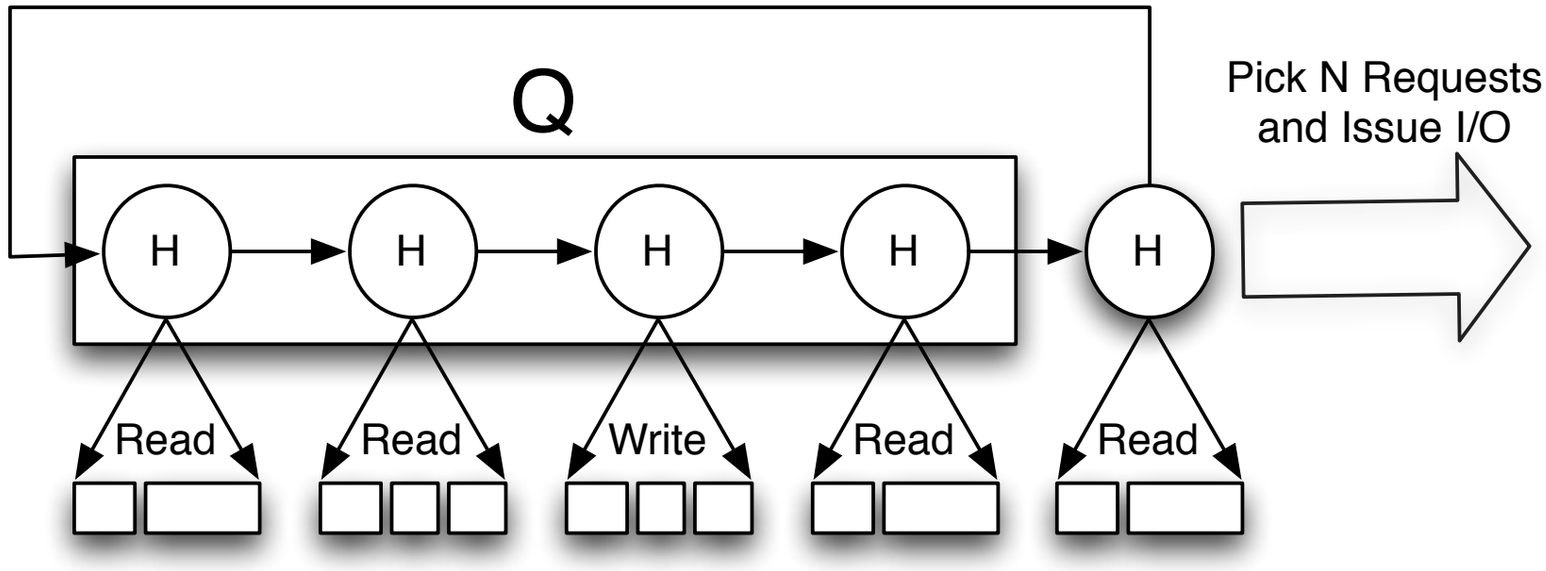
Ouf-Of-Order Pipeline Transfer

- Splitting large I/O requests into small chunks
- I/O Completions are informed in out-of-order way
  - Reduce forward latency
  - I/O sizes are not limited by the memory size of the forwarder
  - Reduce I/O completion time varieties



Request Scheduler

- Scheduling and Merging the small requests at the forwarder
  - Reduce number of seeks
  - Reduce number of requests, the file systems actually sees
- Scheduling overhead must be minimum
  - Ranges are managed by Interval Tree
  - Handle-Based Round-Robin algorithm for the fairness between files



Implementation: IOFSL

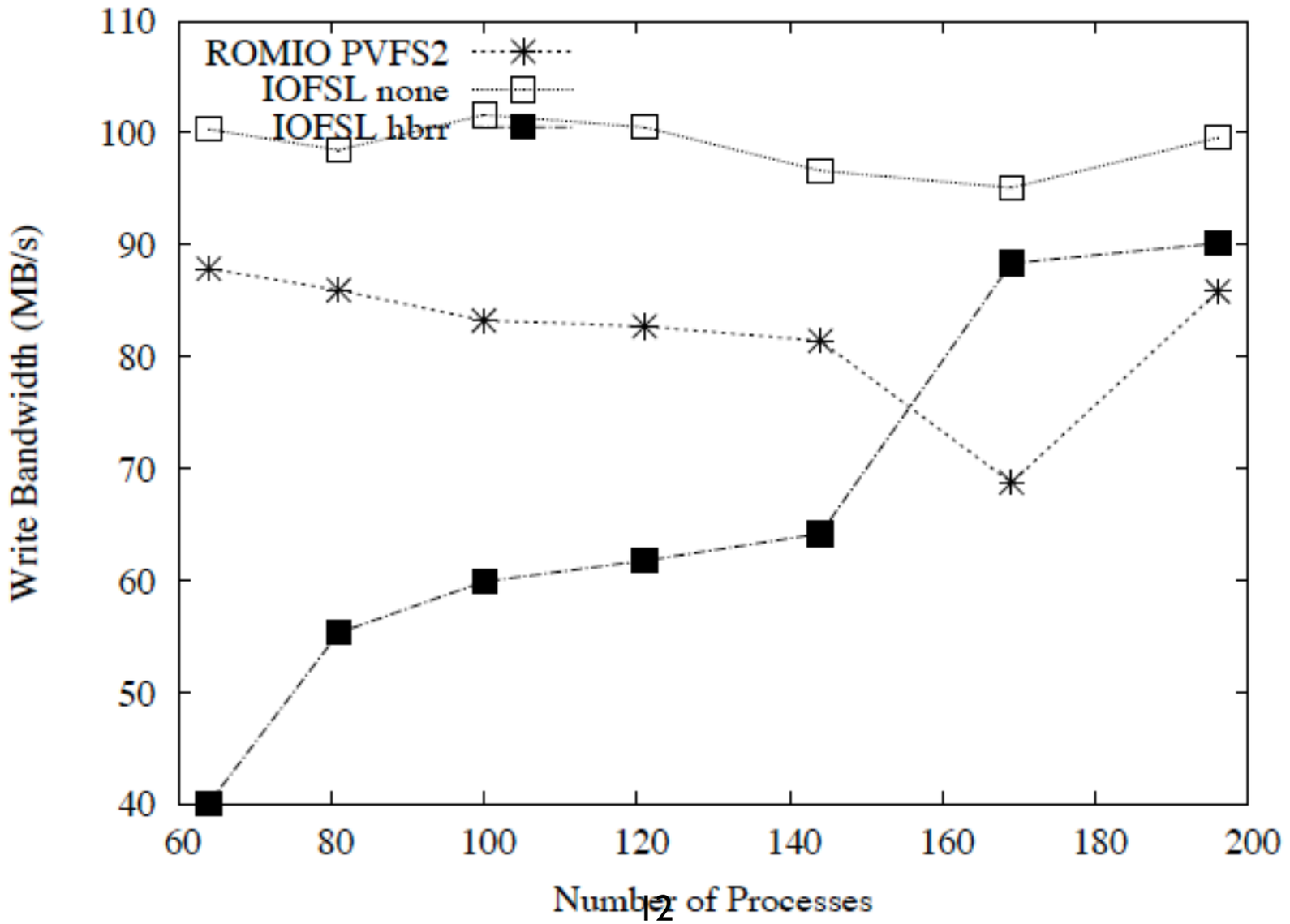
- IOFSL (http://www.iofsl.org/)
  - Open Source I/O Forwarding Implementation
    - developed by Argonne National Laboratory
  - Network Independent (Ether, IB, Myri, BG/P Tree, Portals)
  - File System Independent
  - MPI-IO/FUSE Client (Now in ROMIO from MPICH2 1.1.1)
  - Work on most HPC environments
    - Commodity PC Cluster to BlueGene/P
- I've implemented out-of-order pipelining transfer and the request scheduler in IOFSL
  - The environment on which IOFSL works, will gain the benefit

Evaluation: Environemt

- T2K Open Super Computer Todai
  - 32 nodes Research Cluster
  - 16 cores
    - 2.3 GHz Quad-Core Opteron * 4
  - 32GB Memory
  - 10Gbps Myrinet Network
  - SATA Disk (Read: 49.52 MB/sec, Write 39.76 MB/sec)
- One IOFSL, Four PVFS
- Software
  - MPICH2 1.1.1p1
  - PVFS2 CVS (almost 2.8.2)

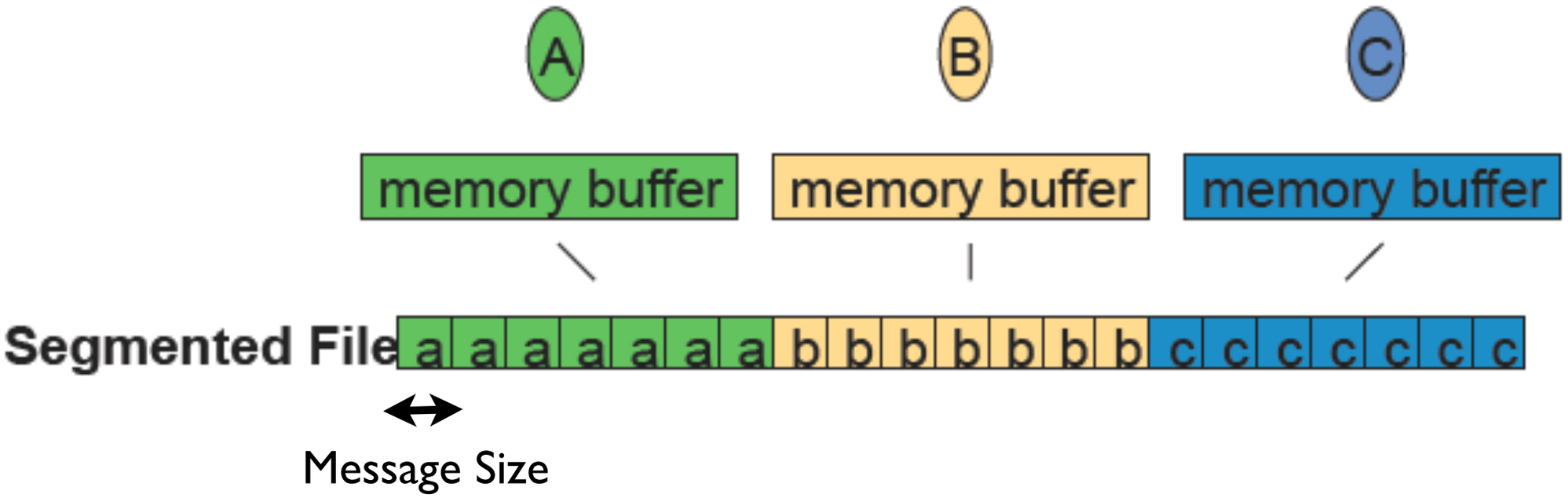
Evaluation: BTIO Benchmark

- None: 14.2%～38.3% increase compared to PVFS2
- HBRR: Slower than None, but increasing the BW as the number of process increases.



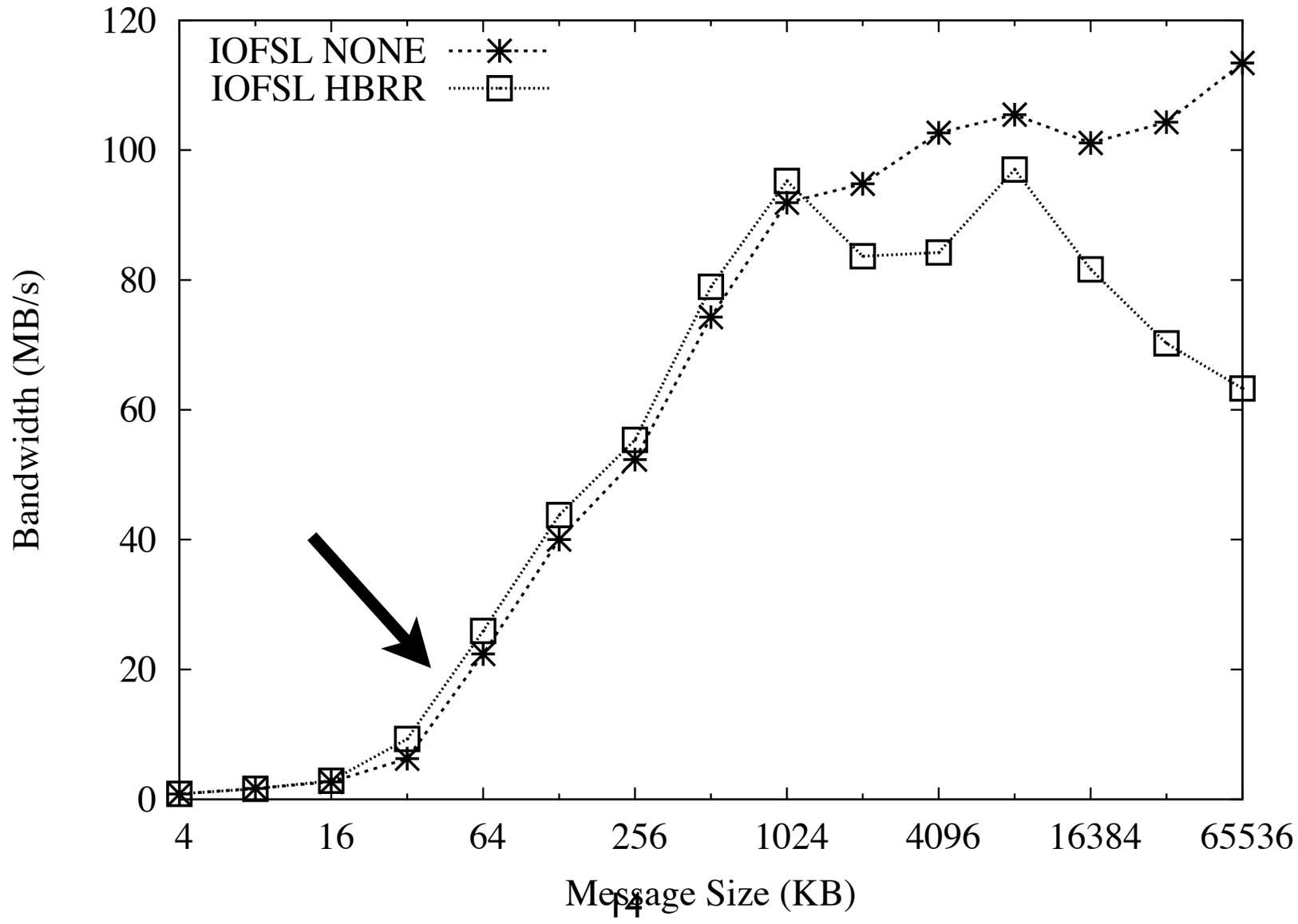
Evaluation: IOR Benchmark

- Each process issues the same amount of I/O
- Gradually increasing the message size, and see the bandwidth change
  - Four PVFS2, One IOFSL, 8 Compute Nodes (128 processes)



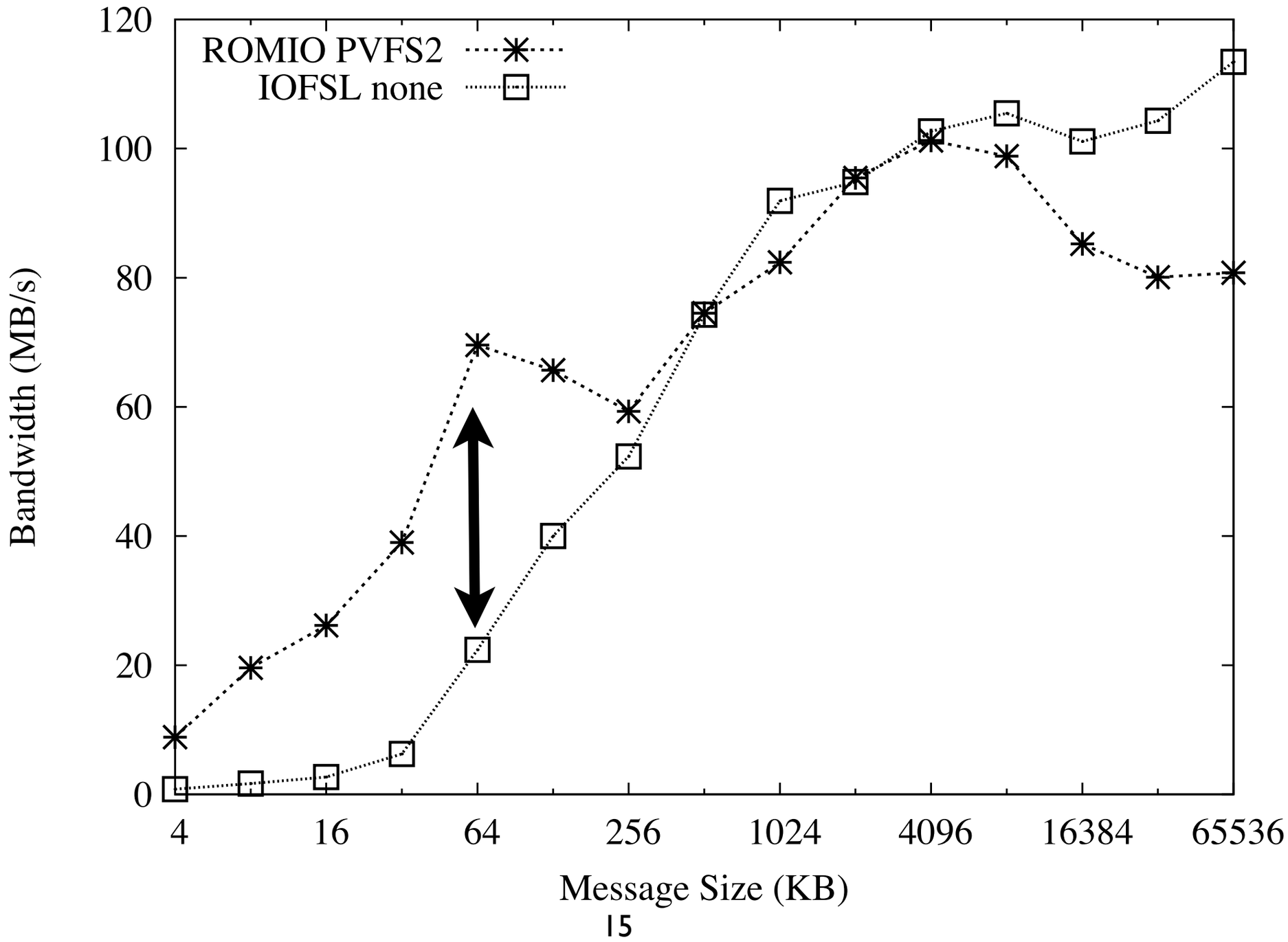
Evaluation: IOR Benchmark (Write)

- HBRR: 48.6% increase compared to None in 32kb
- When larger than 2M, None performs better
  - Disabling the scheduler for the large requests in default



Evaluatoion: IOR Benchmark (Write)

- Direct PVFS2 Access vs IOFSL Access



Related Work

- Computational Plant Project @ Sandia National Laboratory
  - First introduced I/O Forwarding Layer
- IBM GlueGene
  - Compute Nodes run on Lightweight Kernel, and I/O requests are forwarded to I/O nodes
  - ZOID: I/O Forwarding Project for BG/L [Kamil 2008]
    - Only on BlueGene/L
- Lustre Network Request Scheduler [Qian 2009]
  - Request scheduler at the parallel file system nodes
  - Only simulation results

Conclusion

- Propose the optimization techniques for I/O forwarding layer
  - Out-Of-Order Pipeline Transfer / Request Scheduler
- Evaluate on T2K Todai Research Cluster
  - About 30% bandwidth increase in large I/O using IOR Benchmark
  - Scheduler increases about 40% bandwidth, compared to no-scheduler case, in small I/O using IOR Benchmark
  - First shows that I/O forwarding is beneficial in clusters, not a BlueGene-scale
- Bugfix, enhancements of MPICH2, PVFS2 (Now in CVS)

Future Work

- Evaluation on BlueGene/P
- Implementing Caching Layer at I/O Forwarding
  - Provide cooperative file caching service among forwarders
- Hints from MPI-IO
  - Disable the scheduler in the case of collective I/O, etc.

Thanks