FAST '24

Combining Buffered I/O and Direct I/O in Distributed File Systems

Yingjin Qian[1], Marc-André Vef[2], Patrick Farrell[3], Andreas Dilger[3], Xi Li[1], Shuichi Ihara[1], Yinjin Fu[4], Wei Xue[5], André Brinkmann[2]

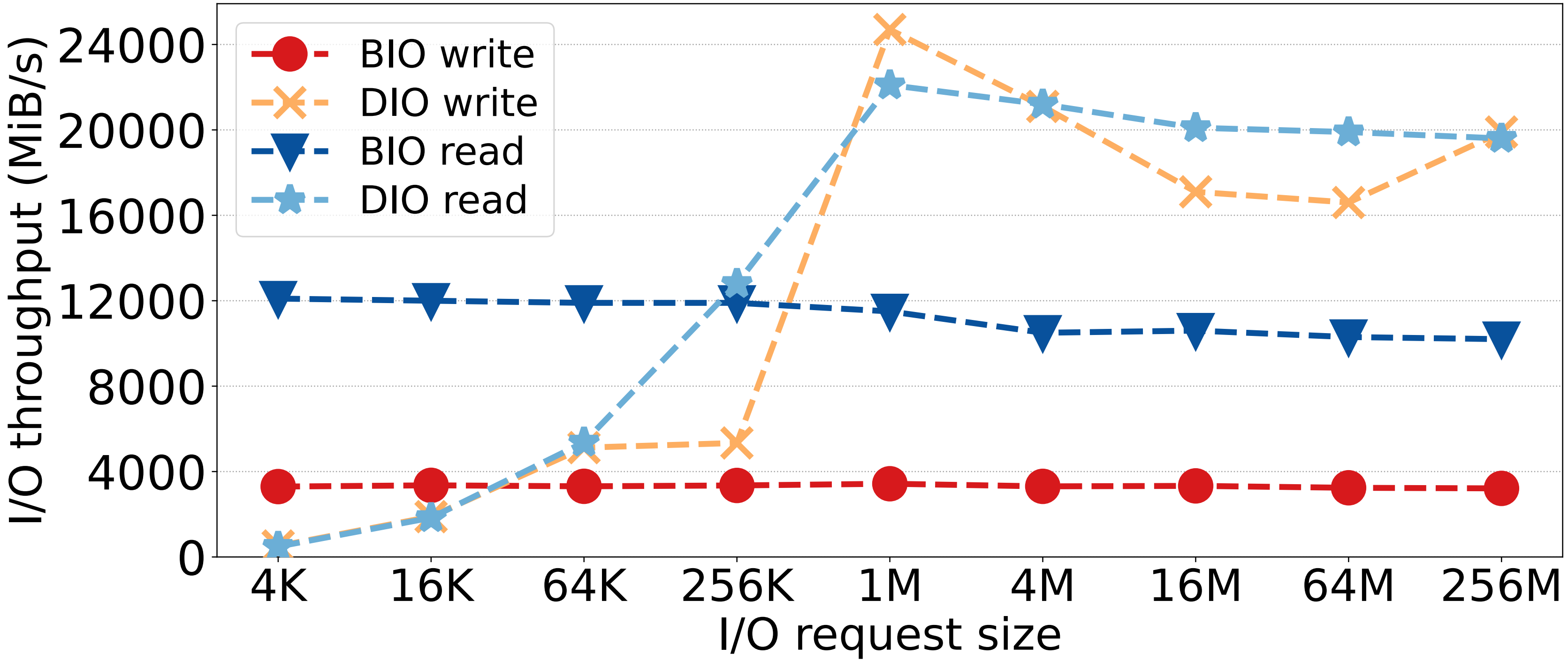
[1]Data Direct Networks (DDN)
[2]Johannes Gutenberg University Mainz
[3]Whamcloud Inc.
[4]Sun Yat-Sen University
[5]Tsinghua University & Qinghai University

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

02/27/24

Motivation

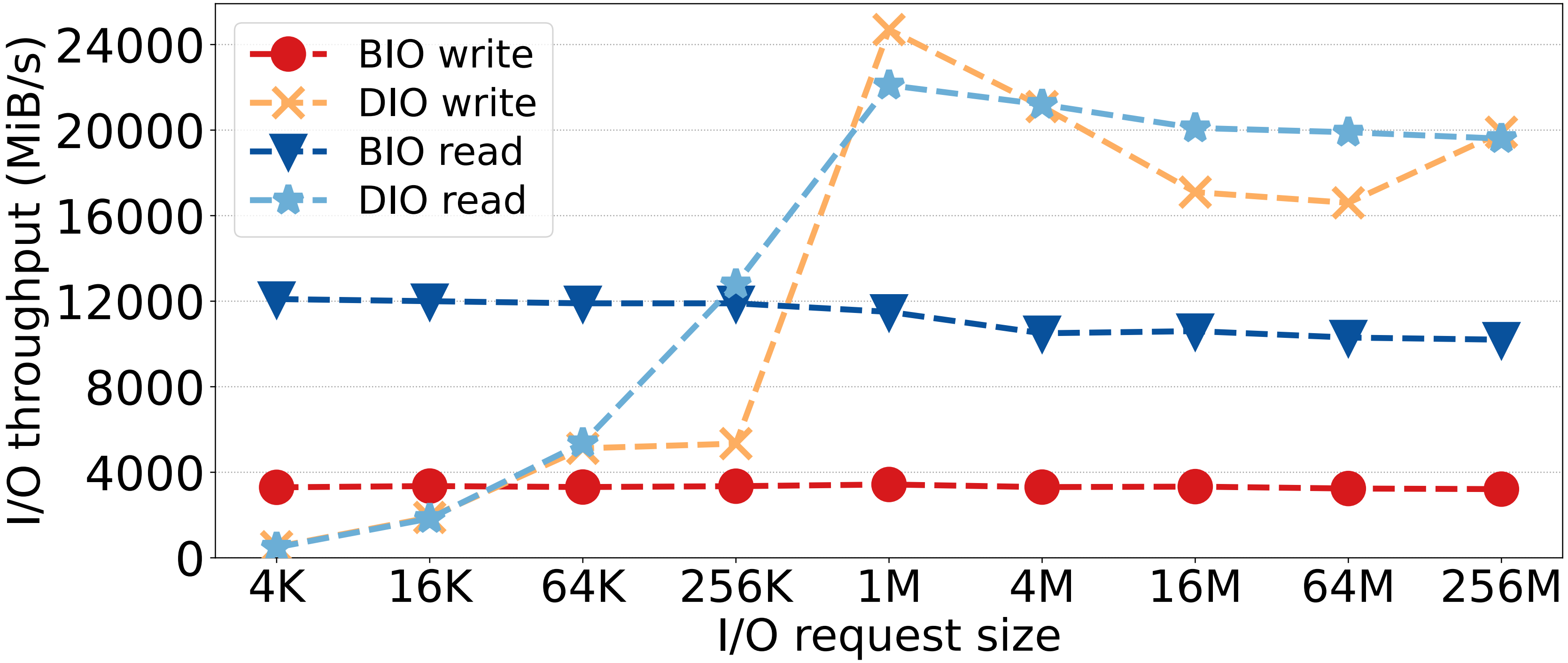
- Linux's default I/O mode is *buffered I/O* utilizing the page cache
- The alternative *direct I/O* bypasses the Linux page cache and can be more beneficial



Local `ldiskfs` performance for various I/O sizes

Motivation

- Various challenges hinder higher direct I/O adoption
  - Users tend to use the familiar I/O mode
  - Alignment constraints can be difficult to accommodate
  - It is often unclear which I/O mode performs better



Local `ldiskfs` performance for various I/O sizes

Goals and contributions

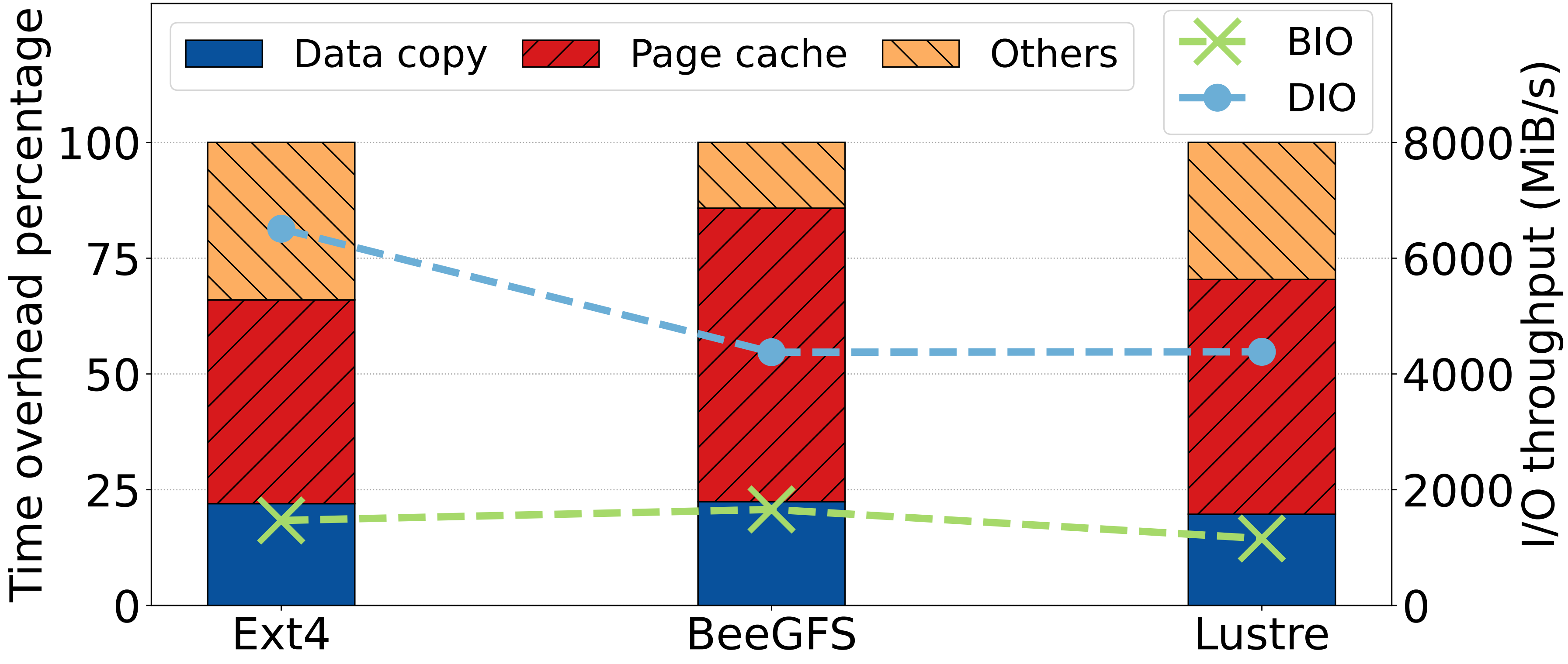
We propose **combining the benefits of buffered I/O and direct I/O**

Key points and contributions:

- Transparent I/O mode decision within the file system
- Decision are based on I/O size, lock contention, and cache usage
- Additional optimizations, e.g., adaptive server-side write-back, and others
- Implemented in the Lustre parallel file system

Background

- Only about 20% of the overall time is spent copying data
- More than 40% of the overall time is spent on page cache management

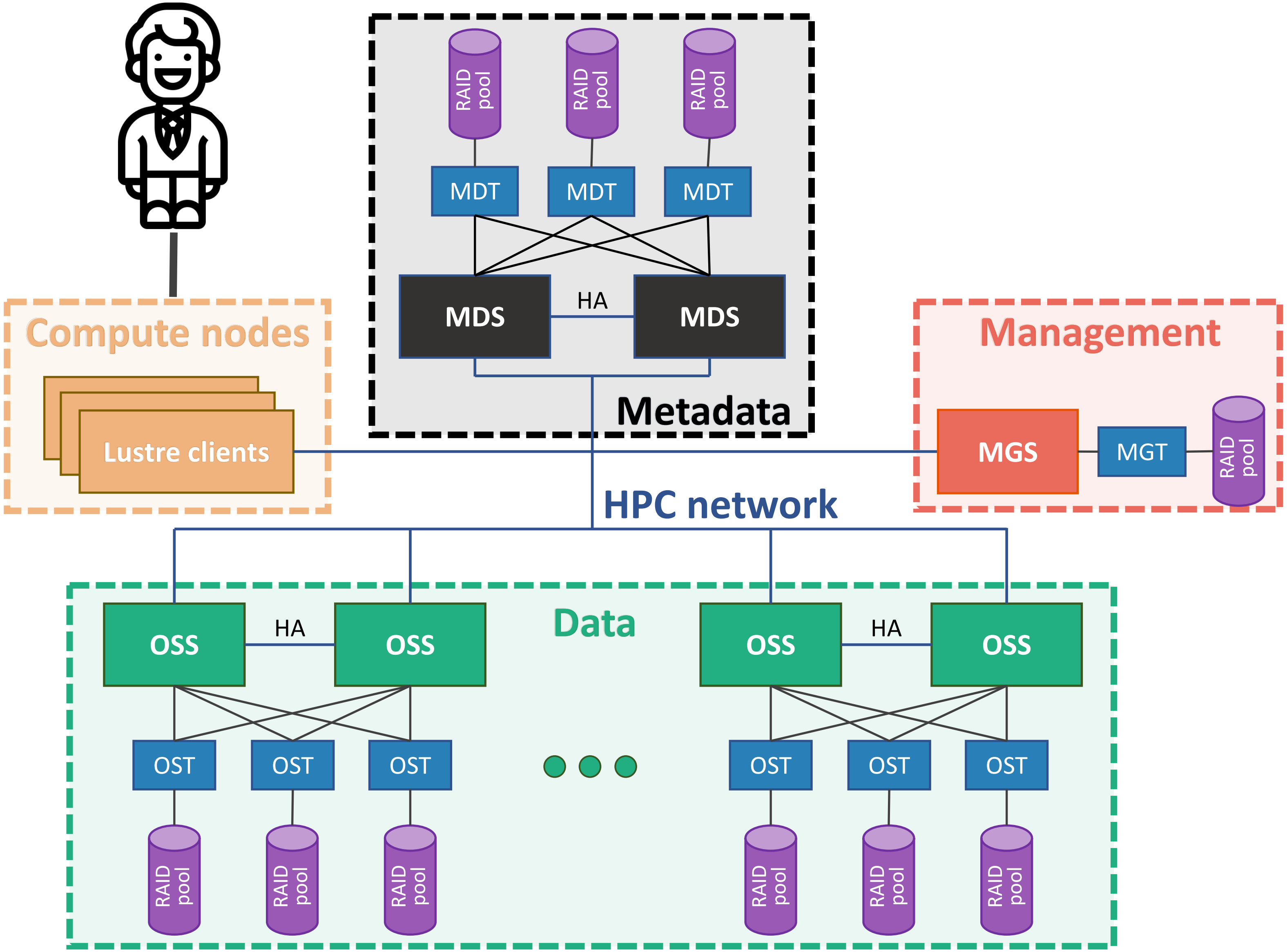


I/O time breakdown for buffered I/O writes (16 MiB I/O size) via `perf`

Background

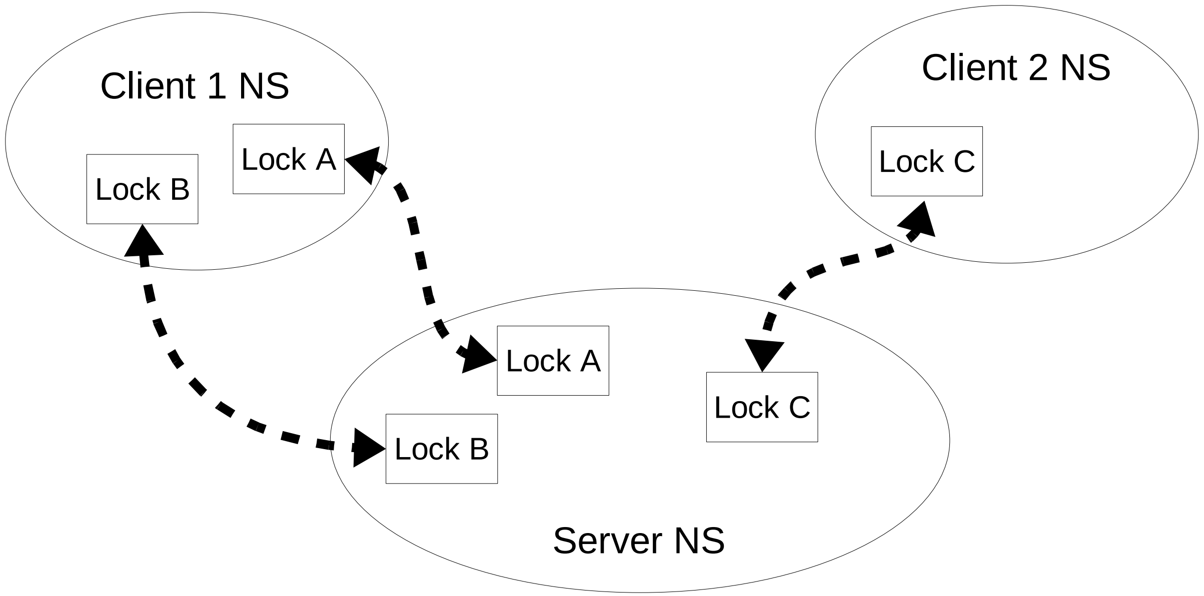
Terminology

- MDS: Metadata Server
  - Processes metadata requests
- MDT: Metadata Target
  - Stores metadata content
  - Manages file access
- OSS: Object Storage Server
  - Processes I/O requests
- OST: Object Storage Target
  - Stores data content
  - File sizes, blocks count, mtime



Background

- Lustre DLM is used for file synchronization and metadata access
- A lock corresponds to a certain resource ID in a namespace (NS)
  - Each Lustre target (OST, MDT, MGT) has a DLM namespace
  - Each Lustre target has full authority about its namespace
- Clients have a copy of a server lock for locally-accessed resources (shadow namespaces)



Shadow namespaces on clients

Wang et al. "Understanding Lustre filesystem internals.", 2009

Weighing I/O modes in Lustre

I/O case	Buffered I/O	Direct I/O
Small I/O size	✓	✗
High latency storage	✓	✗
Unaligned I/O	✓	✗
Large sequential I/O	✗	✓
Many running processes/nodes	✗	✓
System under memory pressure	✗	✓

Feature	Abbreviation
Original Lustre	`vanilla`
Unaligned direct I/O support	-
Client-side I/O mode decision	`autoIO`
Server-side write-back	`svrWB`
Cross-file batching for buffered writes	`XBatch`
Delayed allocation	`delalloc`

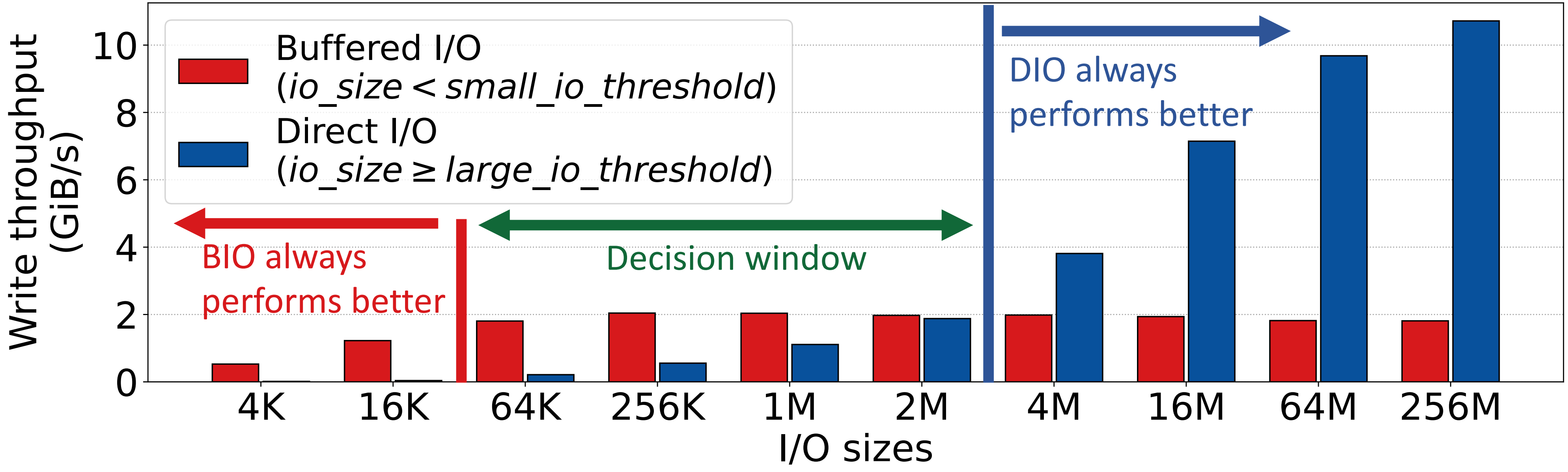
Areas of improvement:

- Use best of both worlds in a given situation
- Remove direct I/O alignment contraints
- Improve many small file performance and reduce file fragmentation

AutoIO

- Automatic alignment of unaligned direct I/O
- Primary I/O mode decision based on I/O size
- Two I/O thresholds allow a *decision window* for
  - Lock contention
  - Memory pressure and low cache reusage
  - Default decision window: [32 KiB, 2 MiB)

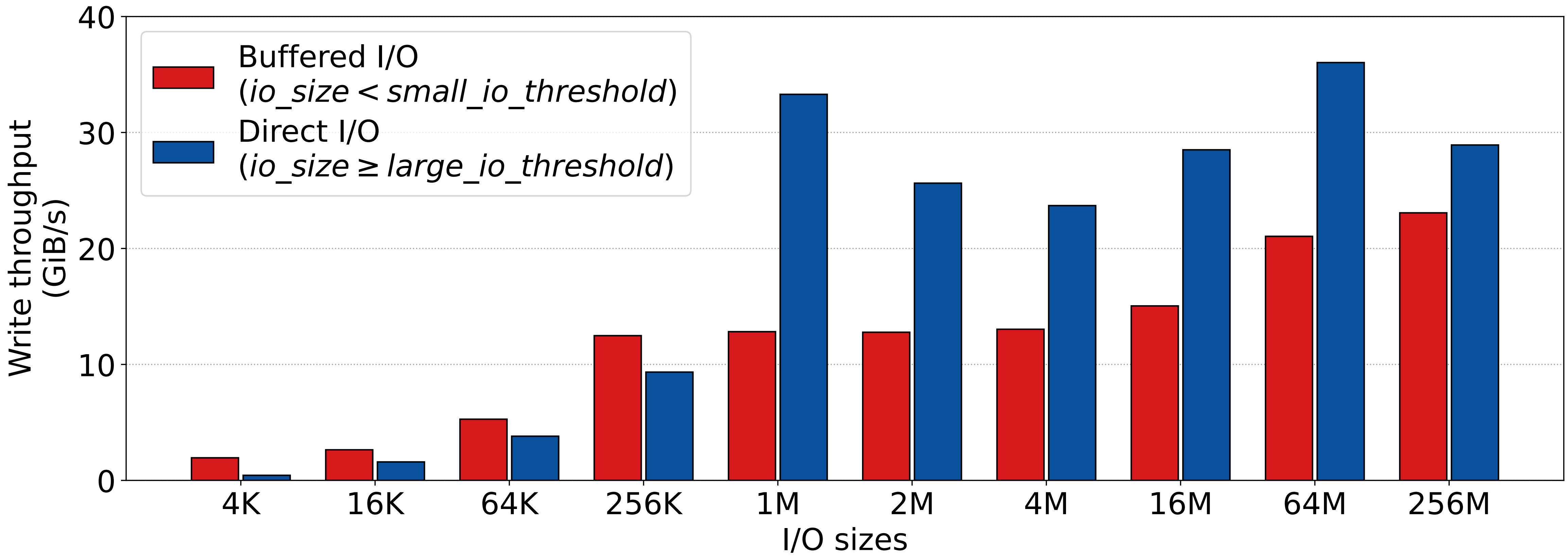
Feature	Abbreviation
Unaligned direct I/O support	–
Client-side I/O mode decision	autoIO



I/O modes under lock contention

- Lock contention workload: Strided I/O over 10 nodes

➢ Under extreme lock contention, direct I/O becomes beneficial earlier

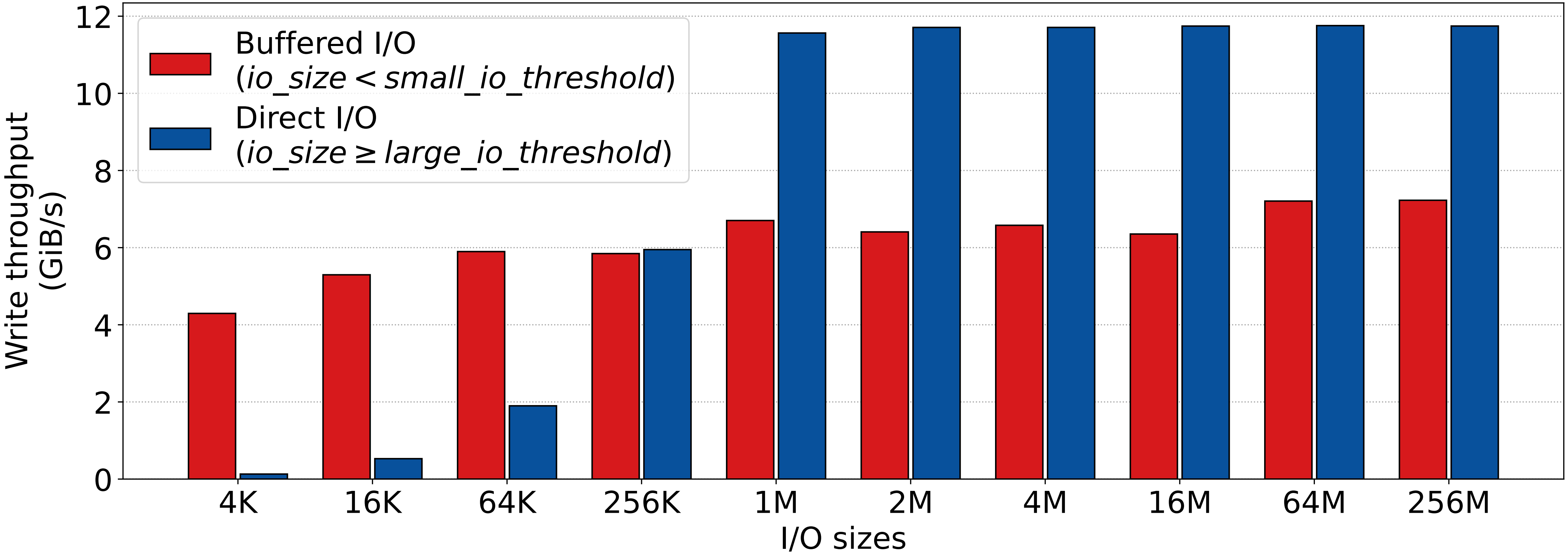


I/O throughput for various I/O sizes and I/O modes under lock contention

I/O modes under cache overusage

- Over caching workload: Cached pages are not reused

➢ Under memory restrictions, direct I/O becomes beneficial earlier



I/O throughput for various I/O sizes and I/O modes under cache over usage

Further optimizations

- Server-side write-back
  - Vanilla Lustre uses write-through
  - ➢ Lustre servers can switch to write-back at a threshold
- Cross-file batching for buffered writes
  - Vanilla batches dirty pages into large bulk RPCs (1 MiB)
  - ➢ This can improve network and disk efficiency
  - But, it can prolong flush operations of many small files
  - ➢ Batch dirty pages of multiple files into one large bulk RPC
- Delayed allocation
  - Improves `svrWB` further to reduce file fragmentation
  - File fragmentation can be caused during strided writes to a single file from many clients
  - ➢ Delayed allocation collects and merges small and non-contiguous I/O requests

Feature	Abbreviation
Server-side write-back	`svrWB`
Cross-file batching for buffered writes	`XBatch`
Delayed allocation	`delalloc`

Consistency and usage

Potential for consistency conflicts

- Data regions from buffered and direct I/O can overlap
- Unaligned direct I/O may need a file region cached on another node
- DLM protects against such conflicts

➢ No impact on Lustre's strong consistency guarantees

Usage and configuration options

- All features and (most) individual thresholds are controlled via `lctl`
- Enable `autoIO`: `lctl set_param llite.*.bio_as_dio=1`
- Enable `svrWB`: `lctl set_param osd-ldiskfs.*.writeback_max_io_kb=64`
- Refer to our Artifacts[1] for further usage options

[1] https://zenodo.org/doi/10.5281/zenodo.10425915

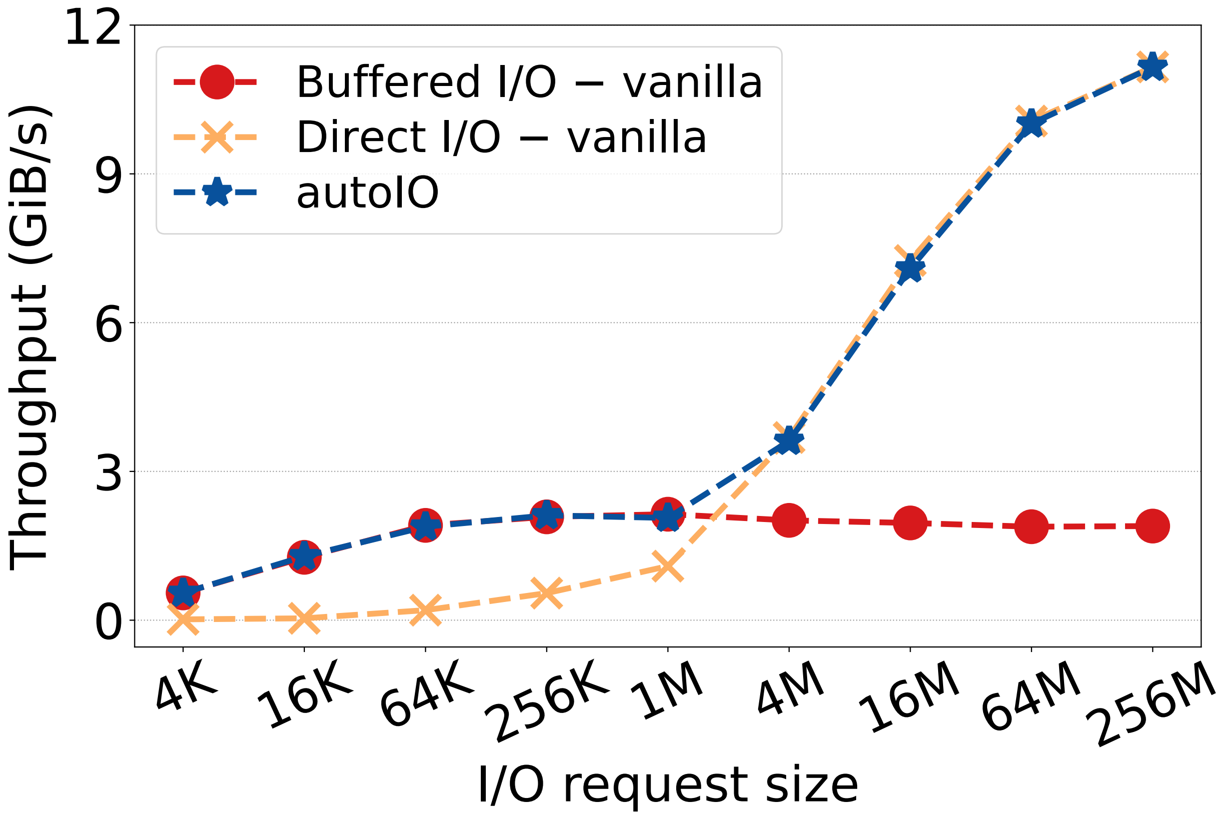
Evaluation

- Lustre 2.15.58 cluster with CentOS 8.7
  - 4 Meta Data Target (MDT)
  - 8 Object Storage Targets (OSTs)
  - Servers using DDN AI400X Appliance (20x Samsung 3.84 TiB NVMe, 4×IB-HDR100)
  - 32 client nodes using Intel Gold 5218 processor, 96 GiB DDR4 RAM, IB-HDR 100, 1 Gbps Ethernet
- BeeGFS 7.4.0
  - Offers two client-side file cache modes
  1. `buffered` (default): Write-back and read-ahead using static buffers
  2. `native`: Relies on the Linux page cache - switches to direct I/O on a set I/O threshold (512 KiB)
- OrangeFS 2.10.0
  - Offers two server-side I/O modes
  1. `alt-aio` (default): Buffered asynchronous I/O
  2. `directio`: Direct I/O mode

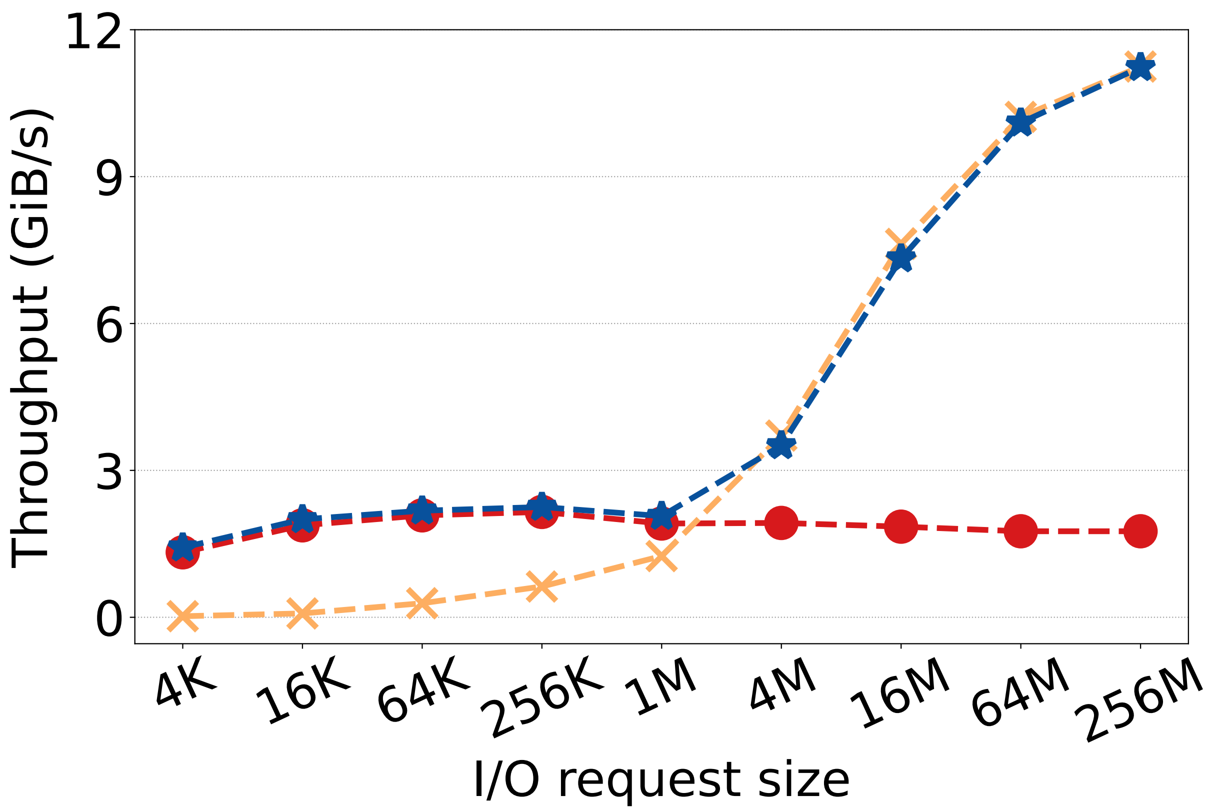
Kindly refer to our paper for further experiments

Single process I/O streaming

- Represents the main use case of autoIO: sequential I/O for a single process



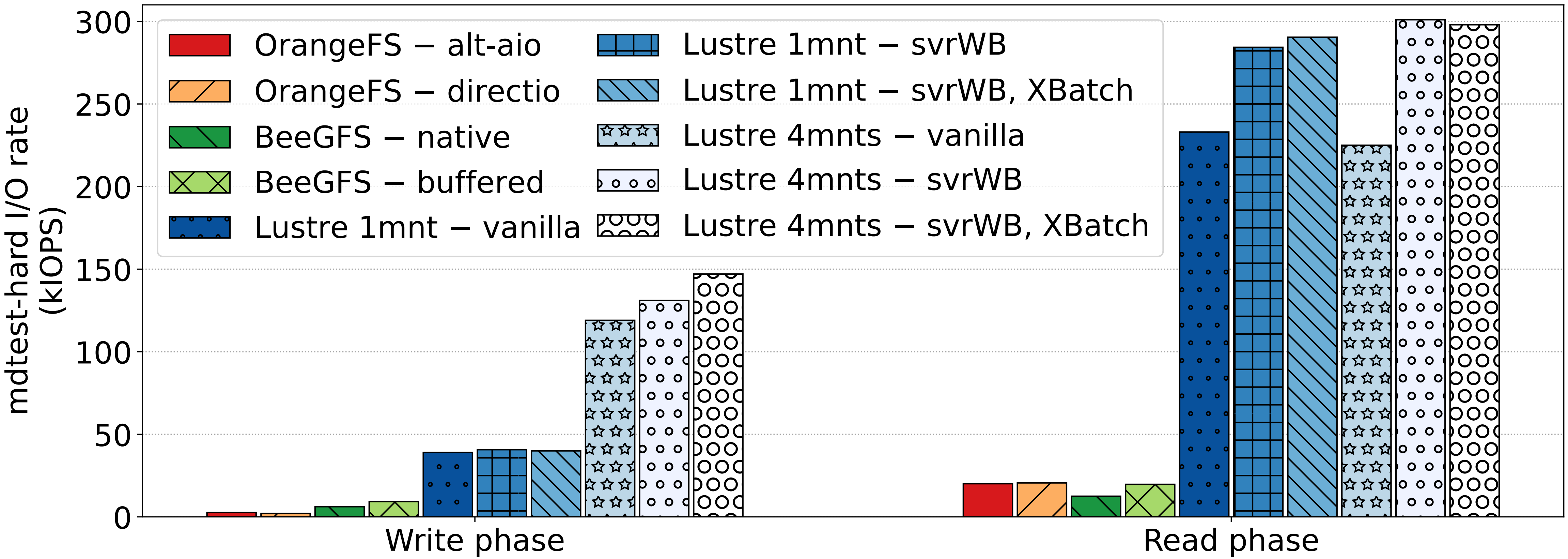
Write



Read

IO500 (10-node challenge)

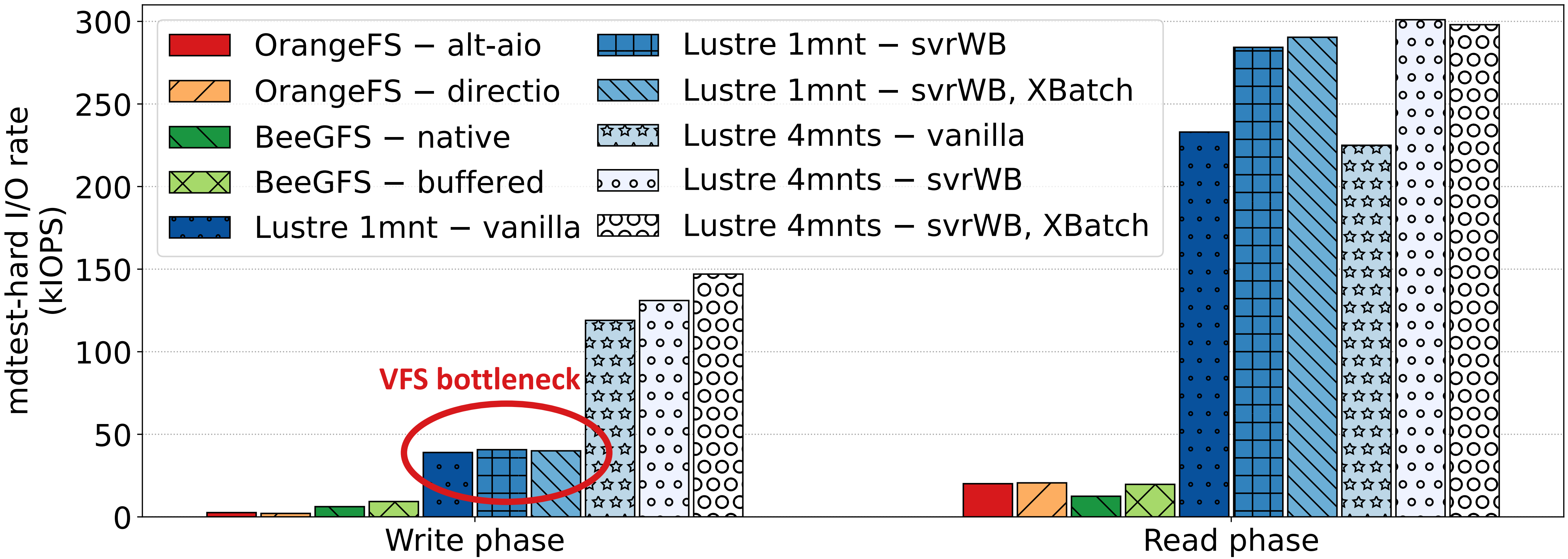
- mdtest-hard generates many small files (3091 bytes in size) in a single directory
- No impact of client-side autoIO: All I/O is buffered



Workload for 10 clients (16 proc each) across file systems and configurations

IO500 (10-node challenge)

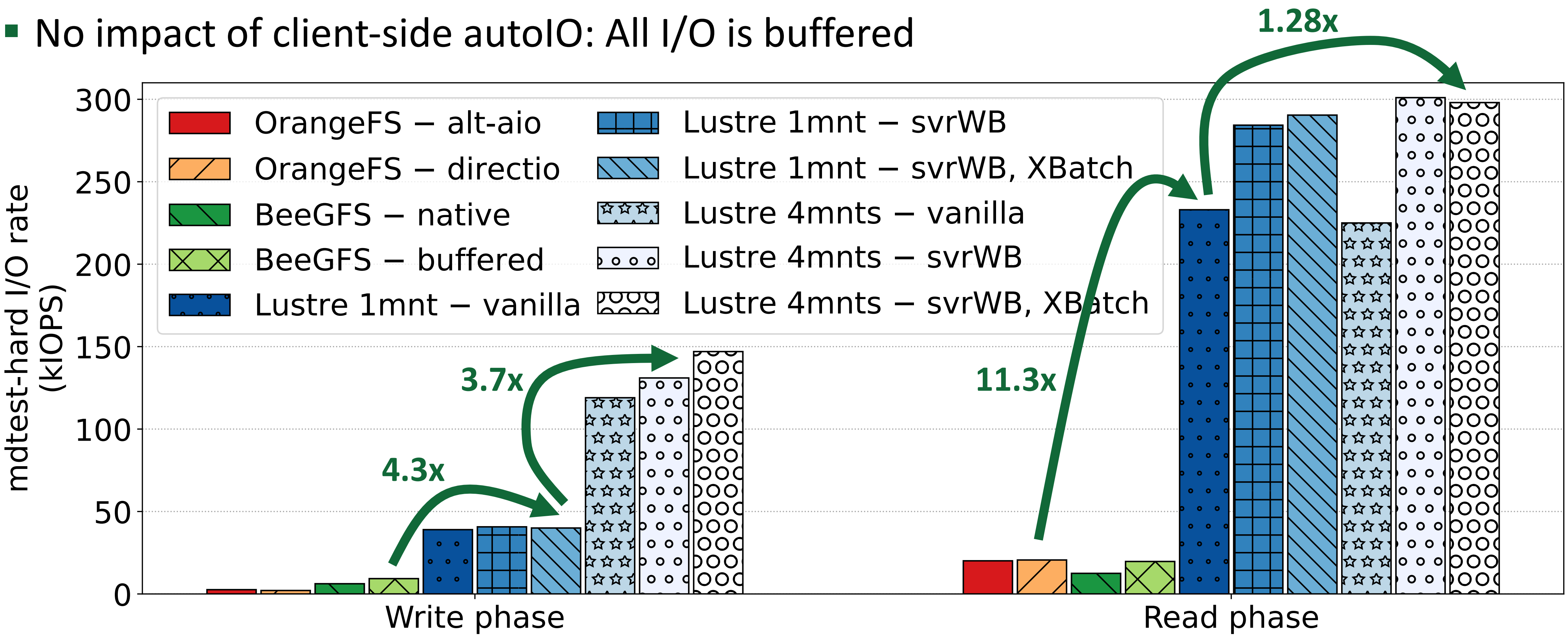
- mdtest-hard generates many small files (3091 bytes in size) in a single directory
- No impact of client-side autoIO: All I/O is buffered



Workload for 10 clients (16 proc each) across file systems and configurations

IO500 (10-node challenge)

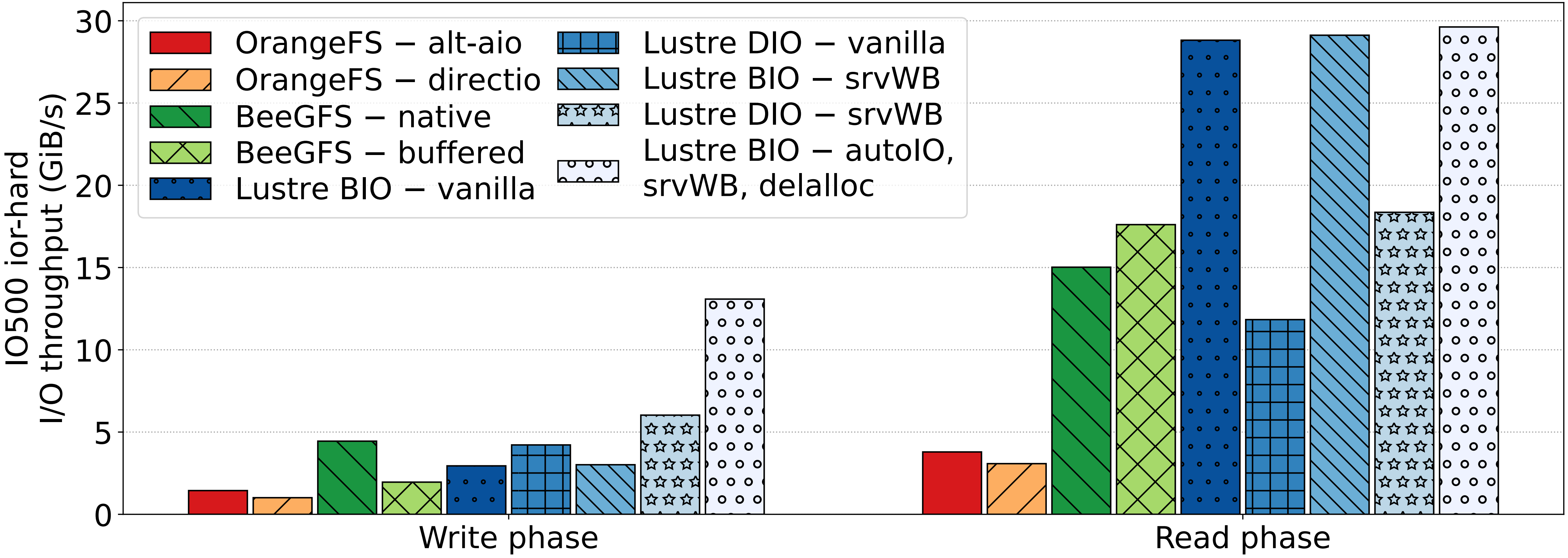
- mdtest-hard generates many small files (3091 bytes in size) in a single directory
- No impact of client-side autoIO: All I/O is buffered



Workload for 10 clients (16 proc each) across file systems and configurations

IO500 (10-node challenge)

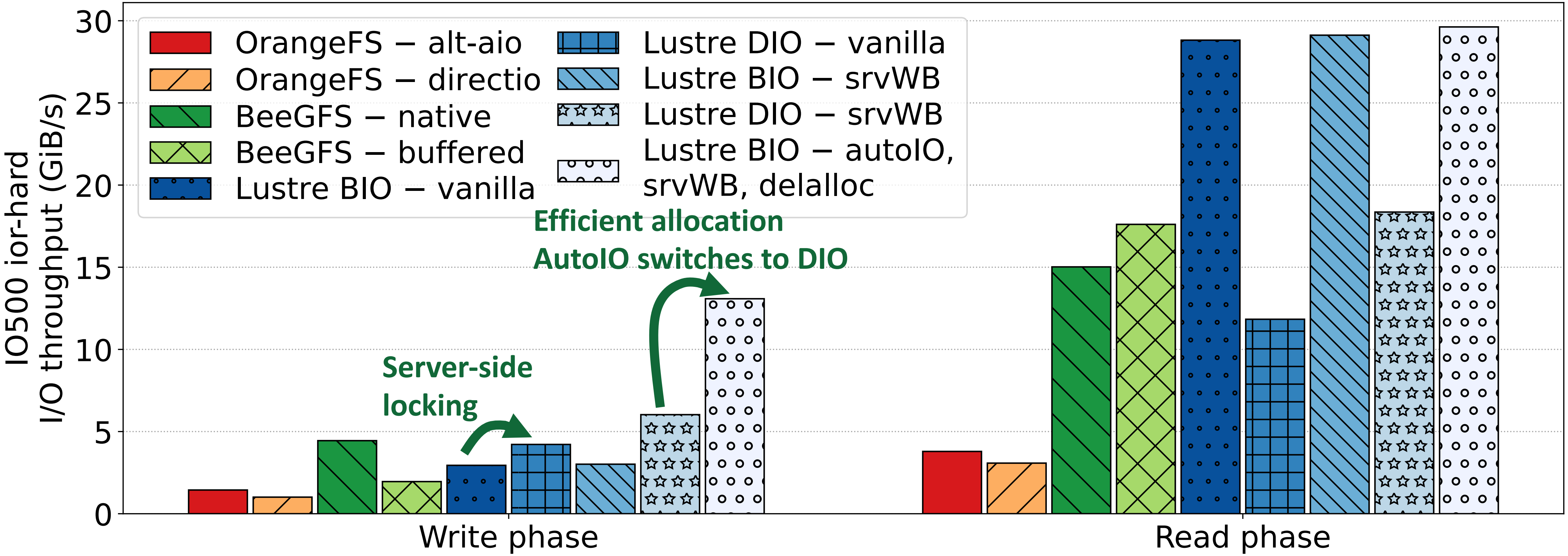
- ior-hard generates unaligned, strided I/O (47,008 bytes in size) to a single shared file
- BeeGFS and OrangeFS don't support unaligned DIO => fallback to BIO



Workload for 10 clients (16 processes each) across file systems and configurations

IO500 (10-node challenge)

- ior-hard generates unaligned, strided I/O (47,008 bytes in size) to a single shared file
- BeeGFS and OrangeFS don't support unaligned DIO => fallback to BIO

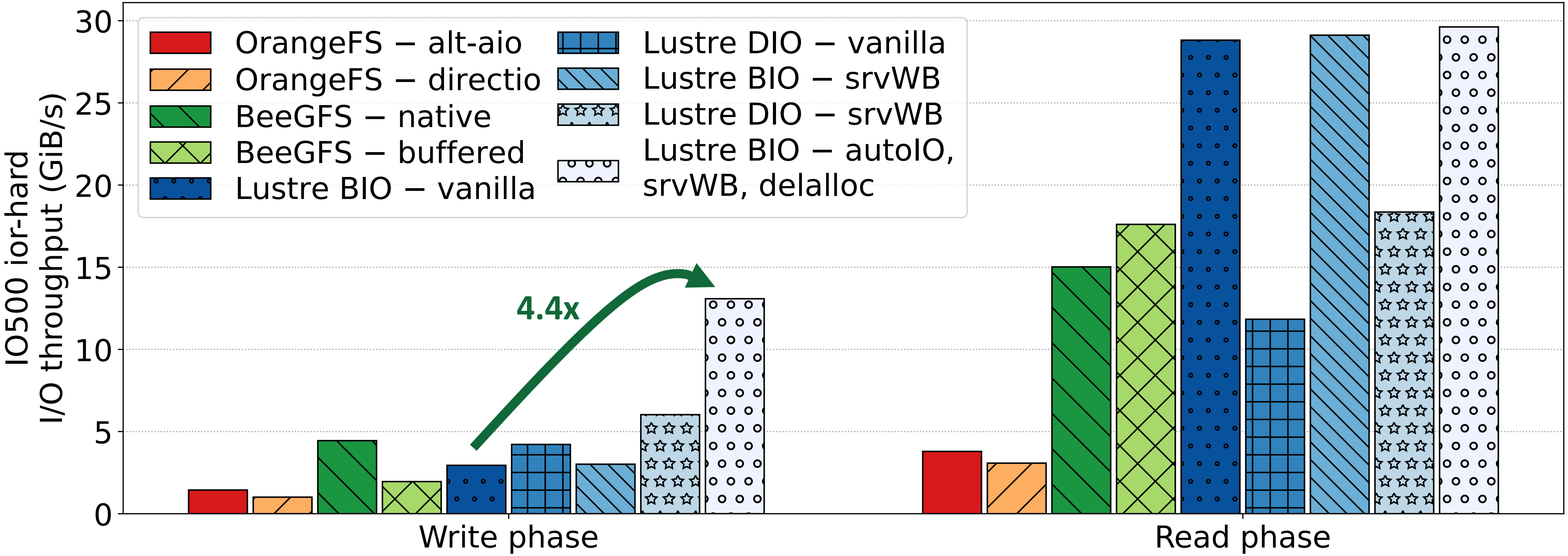


Workload for 10 clients (16 processes each) across file systems and configurations

IO500 (10-node challenge)

- ior-hard generates unaligned, strided I/O (47,008 bytes in size) to a single shared file
- BeeGFS and OrangeFS don't support unaligned DIO => fallback to BIO

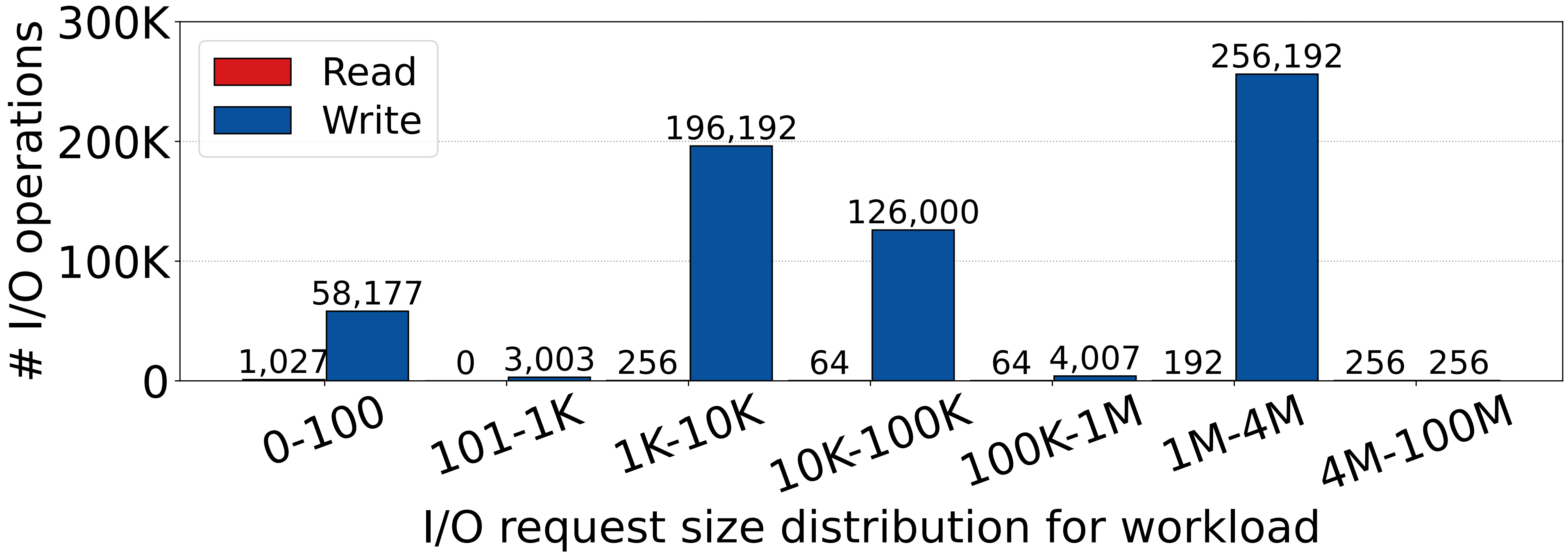
No lock contention: AutoIO stays in BIO



Workload for 10 clients (16 processes each) across file systems and configurations

Nek5000

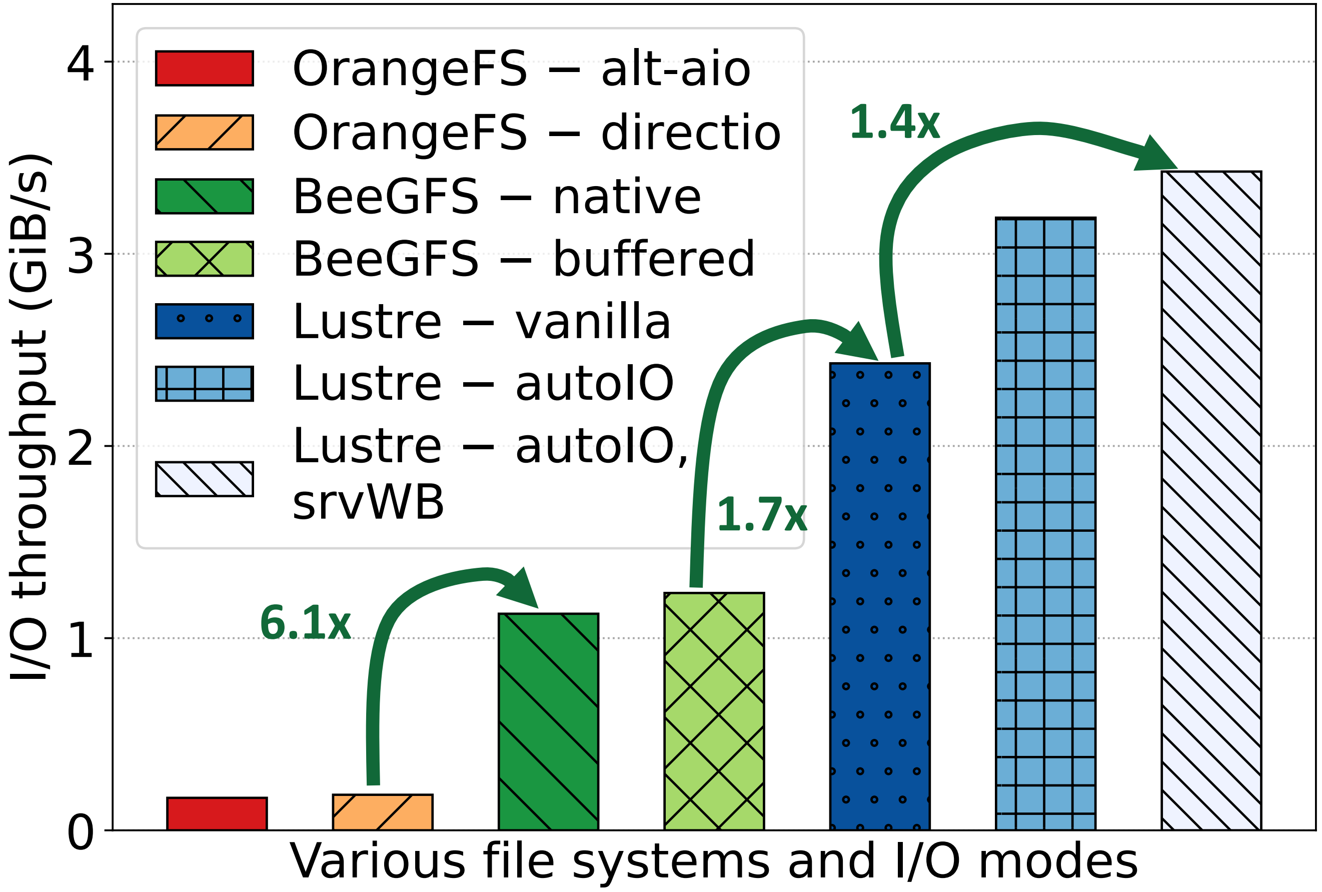
- Running the turbulent flow workload with the Nek5000 bulk-synchronous application
- 512 processes (over 32 nodes), each writing one 600 MiB file per step boundary
- 10 minute workload and a wide I/O size distribution => 600 GiB of data



Nek5000's turbPipe workload I/O access size distribution via Darshan

Nek5000

- Nek5000 turbPipe workload for 32 nodes (16 processes each)



Nek5000's turbPipe I/O throughput

I/O statistics for autoIO	**Count**
Buffered I/O - small threshold	327,281
Direct I/O - large threshold	128,000
Direct I/O - lock contention	132,000
Buffered I/O - default	65

Conclusion & future work

- We have presented a transparent approach to combine buffered I/O and direct I/O
- We integrated our approach into Lustre keeping its strong consistency guarantees
- Key technologies: autoIO, server-side write-back, cross-file batching, and delayed alloc.
- Productization is in progress
  - Unaligned direct I/O support merged in Lustre 2.16 (strictly opt-in; must use `O_DIRECT`)[1]
  - Hybrid I/O for Lustre 2.16 and 2.16+[1]
  - For issue tracking and the current status, refer to the JIRA links listed in our Artifacts' Readme[2]

- Future work
  - Extensive performance analysis of I/O sizes, thresholds, configurations, and application workloads
  - Automatic autoIO thresholds adjustments during runtime
  - Server-side algorithm which considers the server state

[1]LAD23: Buffered I/O, DIO & Unaligned DIO @ Lustre Admin & Dev Workshop 2023
[2]https://zenodo.org/doi/10.5281/zenodo.10425915

JOHANNES GUTENBERG UNIVERSITÄT MAINZ

Whamcloud

Thank You!

EuroHPC Joint Undertaking

ddn

Federal Ministry of Education and Research

Yingjin Qian	qian@ddn.com
Marc-André Vef	vef@uni-mainz.de
Patrick Parrell	pfarrell@whamcloud.com
Andreas Dilger	adilger@whamcloud.com
Xi Li	lixi@ddn.com
Shuichi Ihara	sihara@ddn.com
Yinjin Fu	fuyj27@mail.sysu.edu.cn
Wei Xue	xuewei@tsinghua.edu.cn
André Brinkmann	brinkman@uni-mainz.de

Acknowledgements:
We sincerely thank our shepherd Jinkyu Jeong and our anonymous reviewers.

ADMIRE
malleable data solutions for HPC

中山大學
SUN YAT-SEN UNIVERSITY

flaticon.com