LUG 2026

Whamcloud

Status of Lustre’s Hybrid I/O

LUG 2026

Marc-André Vef, Patrick Farrell, Yingjin Qian

ddn

Abstract

Whamcloud

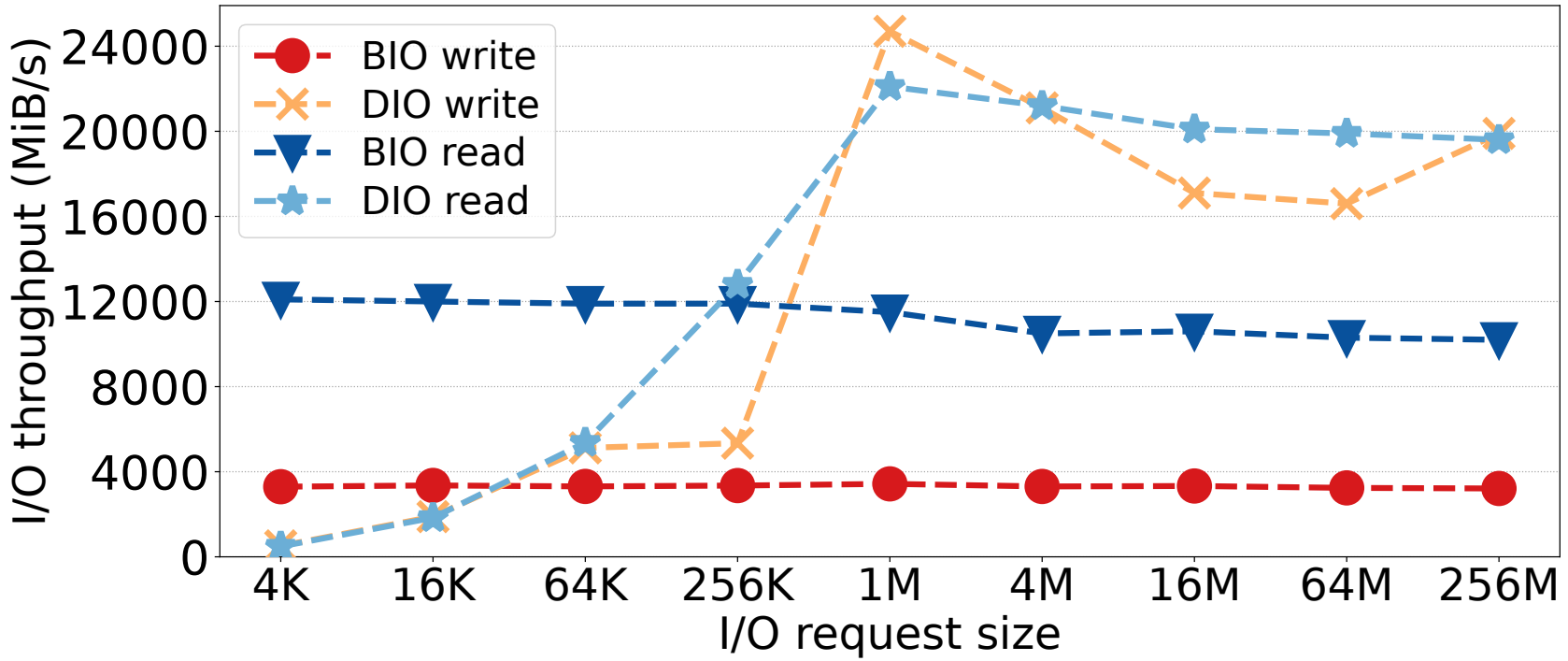
- Hybrid I/O is a new data path for the Lustre client
  - Combines buffered and direct I/O depending on the I/O size
  - Relies on unaligned DIO support removing memory alignment requirements
  - Transparent to the application with no impact to strong consistency guarantees
- Today we'll talk about …
  - Hybrid I/O v1 and what made it into Lustre 2.17
  - Research on additional features with real application experiments
  - Hybrid I/O v2 and what additional features are in-flight
  - Further performance improvements
- Notes on presented numbers
  - All numbers should be understood as general guidance: look at trends, not specific numbers

Introduction

- 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

Introduction

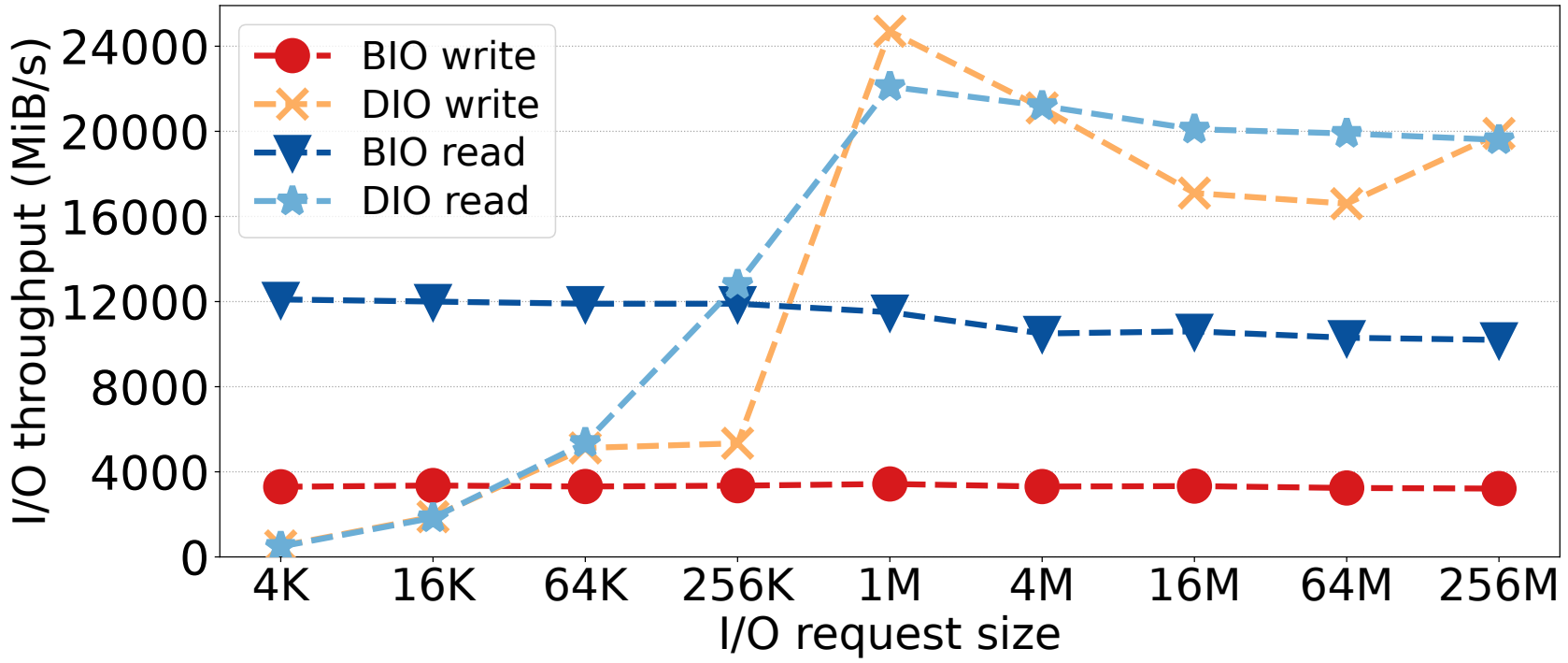
- Linux's default I/O mode is *buffered I/O* utilizing the page cache
- The alternative *direct I/O* bypasses the Linux page cache and can be more beneficial



Local `ldiskfs` performance for various I/O sizes

Introduction

- 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

Buffered I/O vs direct I/O

Whamcloud

- Buffered I/O means “uses the page cache”
- Pros: Flexible
  - No alignment requirements from user space, allows read ahead, write aggregation
- Cons: Not scalable
  - Significant overhead for cache management, locking hinders multi-process scalability

- Direct I/O means “direct from user memory, does not use the page cache”
- Pros: Scalable
  - High single stream performance for large I/O sizes, scalable with processes (no locking)
- Cons: Inflexible
  - Synchronous, exposes disk latency, and requires memory alignment

Hybrid I/O v1

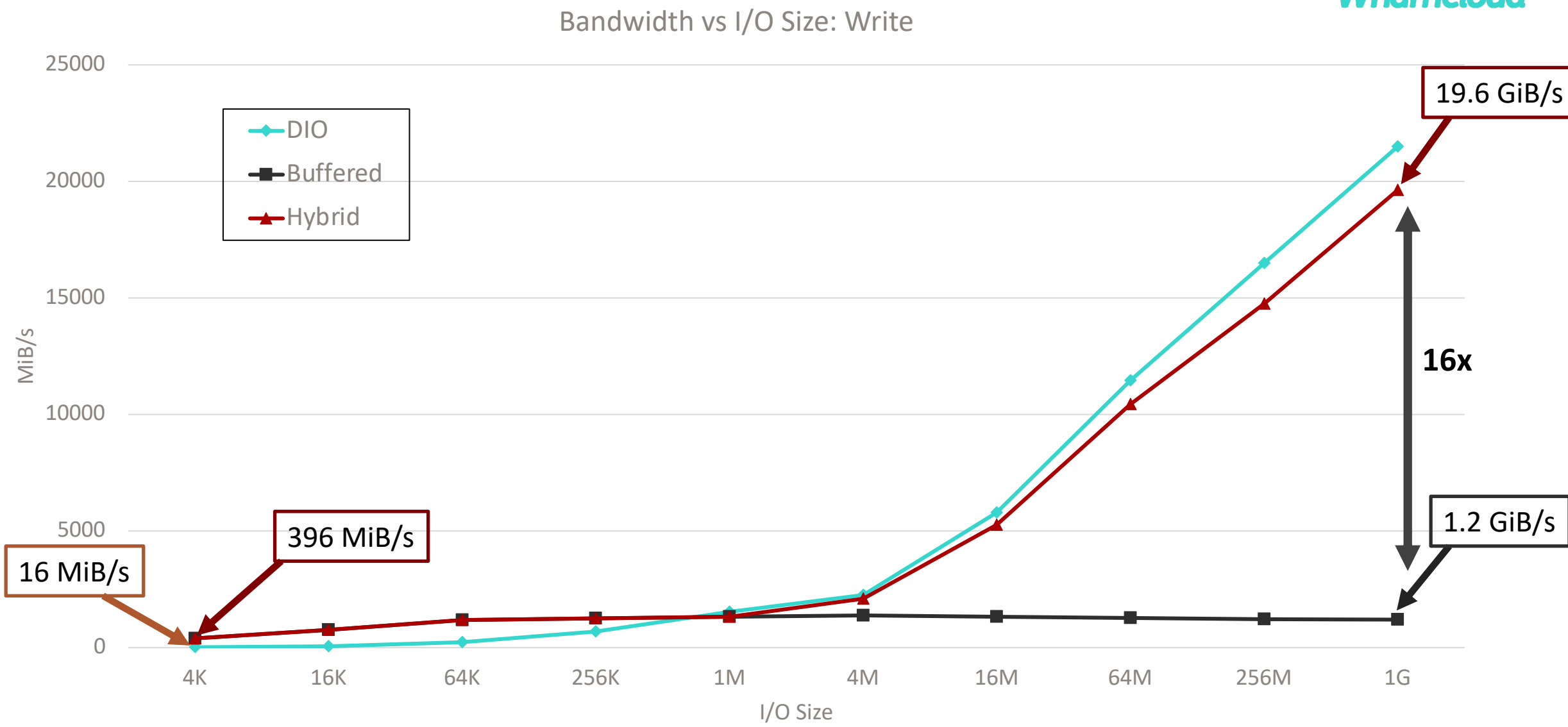
Whamcloud

Combine the benefits of buffered I/O and direct I/O

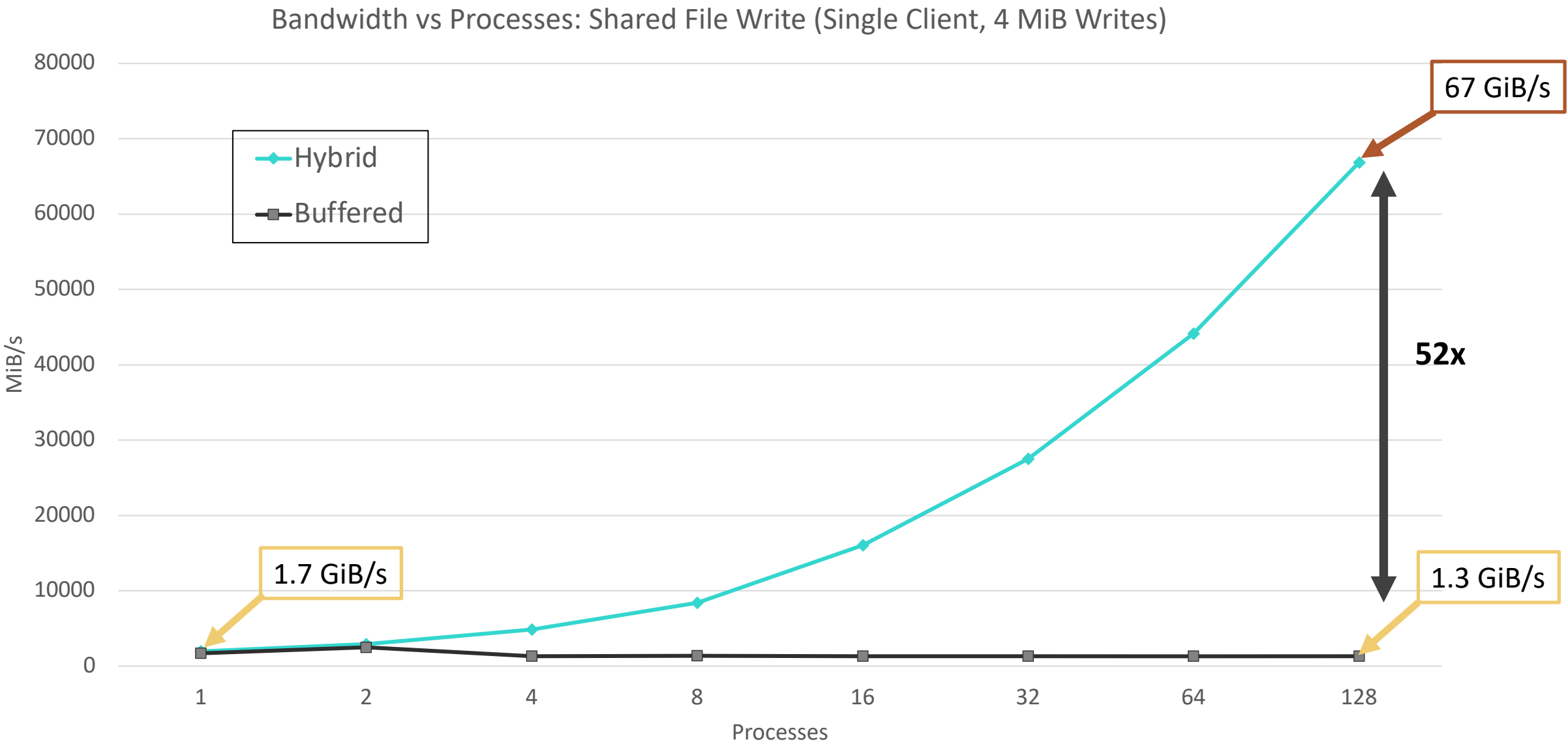
Key points:

- Opt-in hybrid I/O introduced with Lustre 2.16
- Automatically switches between buffered and unaligned direct I/O based on size
- Fully transparent to the application
- Read and write threshold to switch to direct I/O
  - Default values: 2 MiB for writes and 8 MiB for reads
  - Thresholds are customizable
- Since 2.17, hybrid I/O is enabled by default

Hybrid I/O: Write Performance



Single Client Shared File: Writes



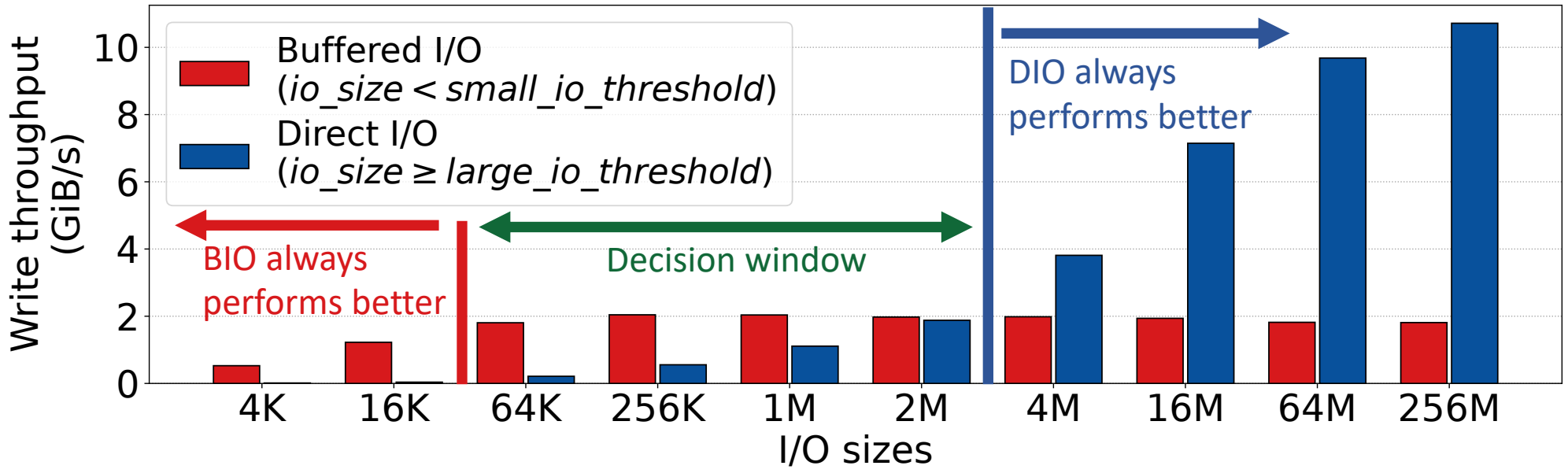
Research: AutoIO

- AutoIO builds on hybrid I/O and presents a proof-of-concept research prototype
- AutoIO was based on Lustre 2.15.58 exploring the following additions to hybrid I/O:
  1. Two I/O thresholds to allow a decision window for lock contention and low cache reuse
  2. Server-side write-back at a threshold and delayed (block) allocation
  3. Cross-file batching for buffered writes
- In this talk, we'll focus on the first two

Y. Qian, M.-A. Vef, P. Farrell, A. Dilger, X. Li, S. Ihara, Y. Fu, W. Xue, A. Brinkmann.
Combining Buffered I/O and Direct I/O in Distributed File Systems.
In Proceedings of the 22nd USENIX Conference on File and Storage Technologies – FAST '24

AutoIO

- Primary I/O mode decision based on I/O size
- Two I/O thresholds allow a *decision window* for
  - Lock contention & low cache reuse
  - Default decision window: [32 KiB, 2 MiB)
- Decision window defaults to buffered I/O

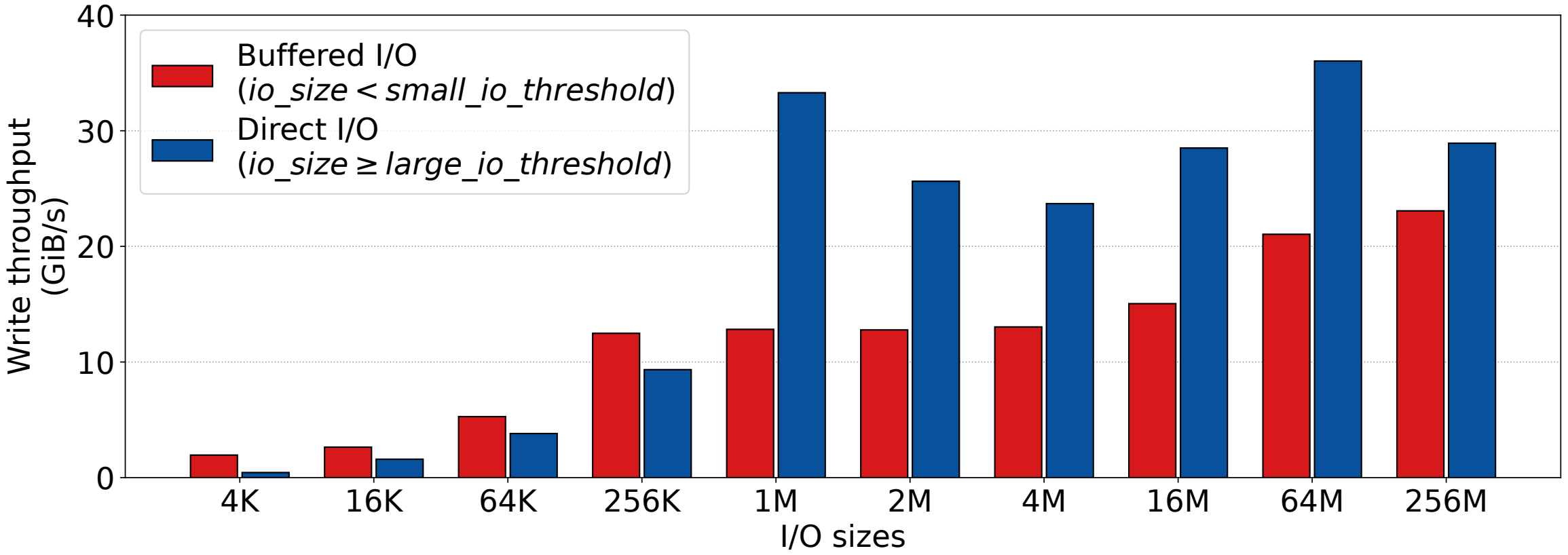


AutoIO under lock contention

Whamcloud

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

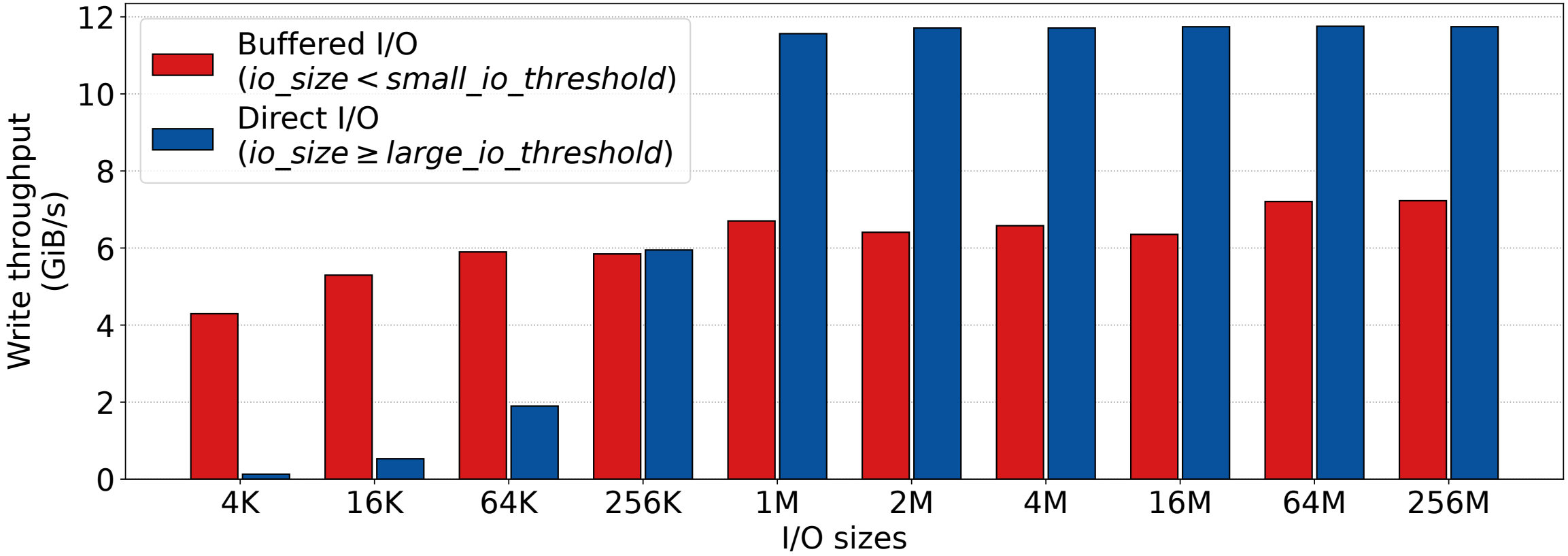
➢ Under extreme lock contention, direct I/O becomes beneficial earlier



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

I/O modes under low cache reuse

- 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 low cache reuse

Further optimizations

- Server-side write-back (srvWB)
  - Vanilla Lustre: write-through — writes are synchronous (latency-bound for small I/O)
  - svrWB: writes below threshold stay in server page cache
  - Enables batching and RAID-friendly coalescing on the server
  - Tunable: `osd-ldiskfs.*.writeback_max_io_kb` (default 64 KiB)
- Delayed allocation
  - Extends svrWB by deferring block allocation until flush
  - Once ≥ 1 MiB stripe-aligned, flush as one RAID-aligned BIO by a worker thread
  - E.g., dcp 4 TiB copy with 3× fewer extents on disk

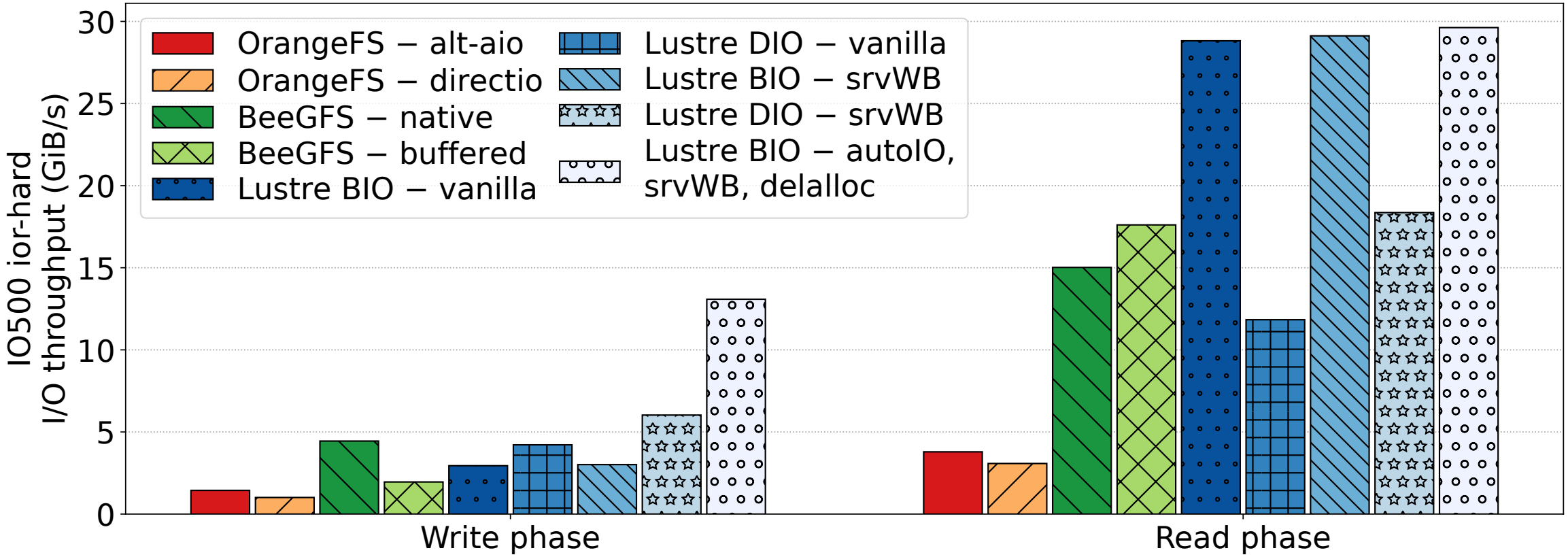
Experimental setup

- Lustre 2.15.58 cluster with CentOS 8.7
  - 4 MDTs, 8 OSTs
  - Servers using DDN AI400X Appliance (20x Samsung 3.84 TiB NVMe, 4x IB-HDR100)
  - 32 client nodes using Intel Gold 5218 processors, 96 GiB DDR4 RAM, IB-HDR 100
- 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

Artifacts of setup and all experiments: https://zenodo.org/doi/10.5281/zenodo.10425915

IO500 (10-node challenge) – ior-hard

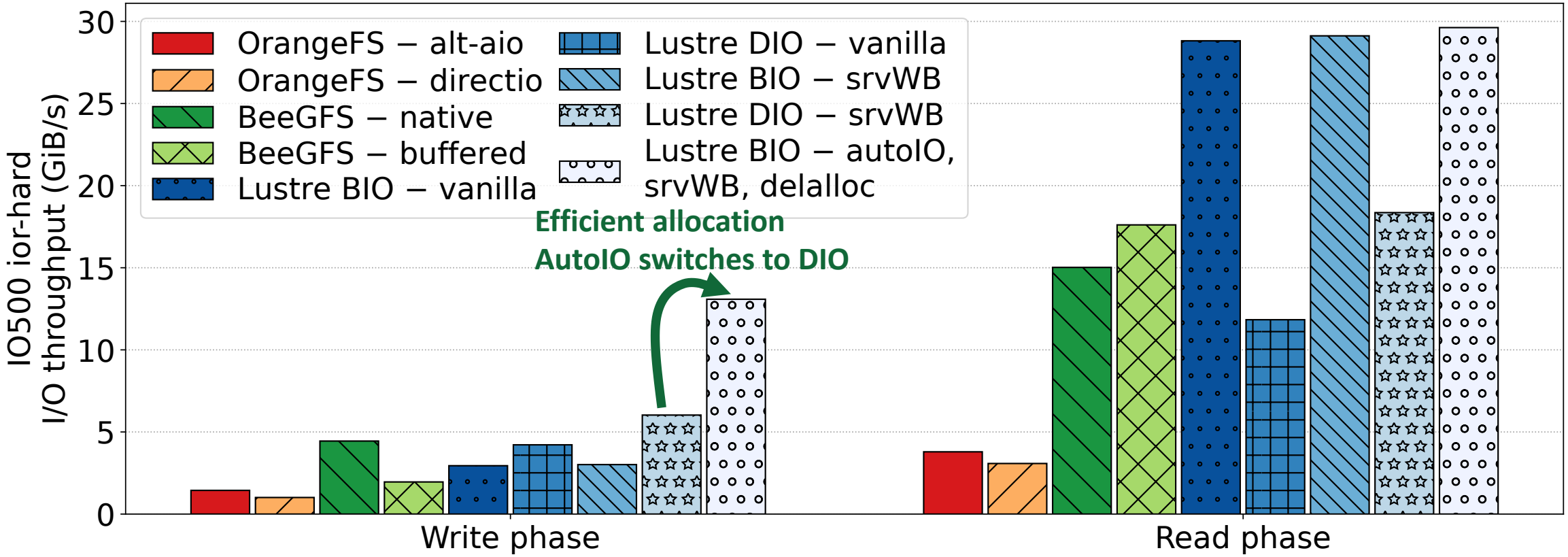
- ior-hard generates unaligned, strided I/O (47,008 bytes in size) to a single shared file
- BeeGFS and OrangeFS don’t support unaligned DIO => fallback to BIO



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

IO500 (10-node challenge) – ior-hard

- 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

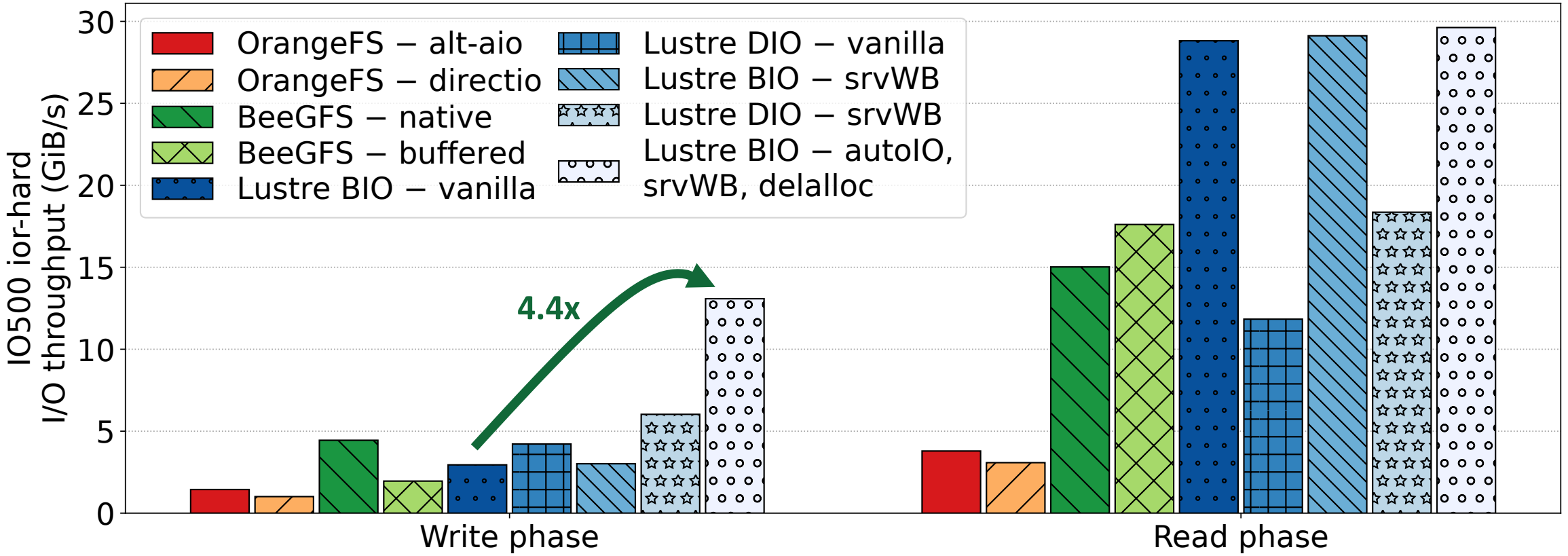


Workload for 10 clients (16 processes each) across file systems and configurations

IO500 (10-node challenge) – ior-hard

- 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

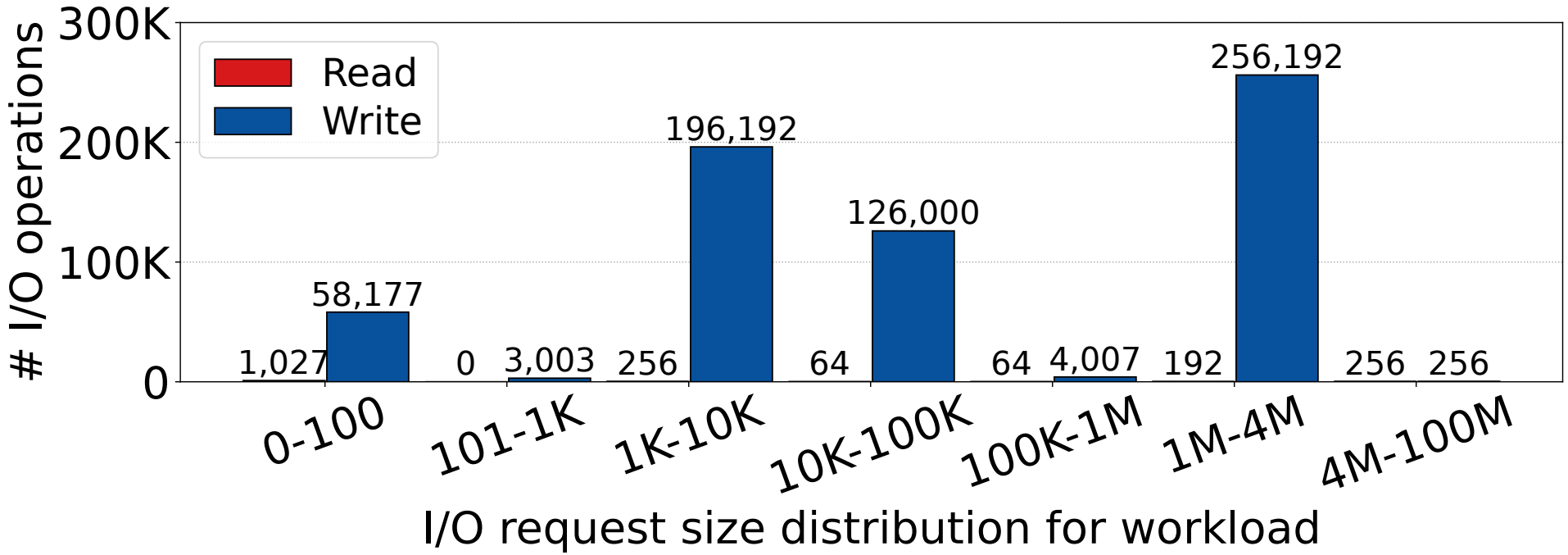


Workload for 10 clients (16 processes each) across file systems and configurations

Nek5000 (CFD workload) – I/O pattern

Whamcloud

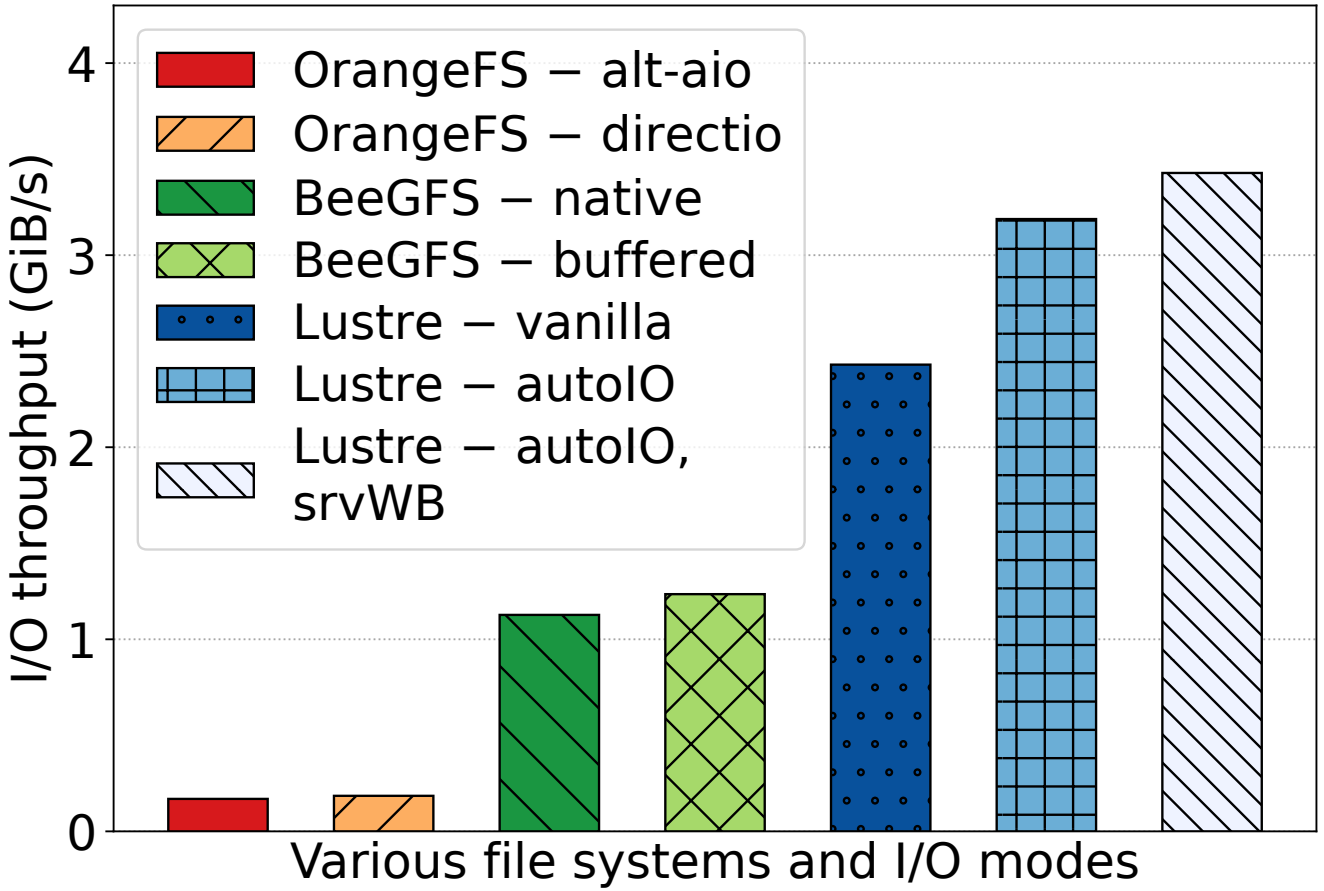
- Running the turbulent flow workload with the Nek5000 bulk-synchronous application
- 512 processes (over 32 nodes) collectively write to one 600 MiB file per step boundary
- 10 minute workload and a wide I/O size distribution => 600 GiB of data



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

Nek5000 (CFD workload) – I/O throughput

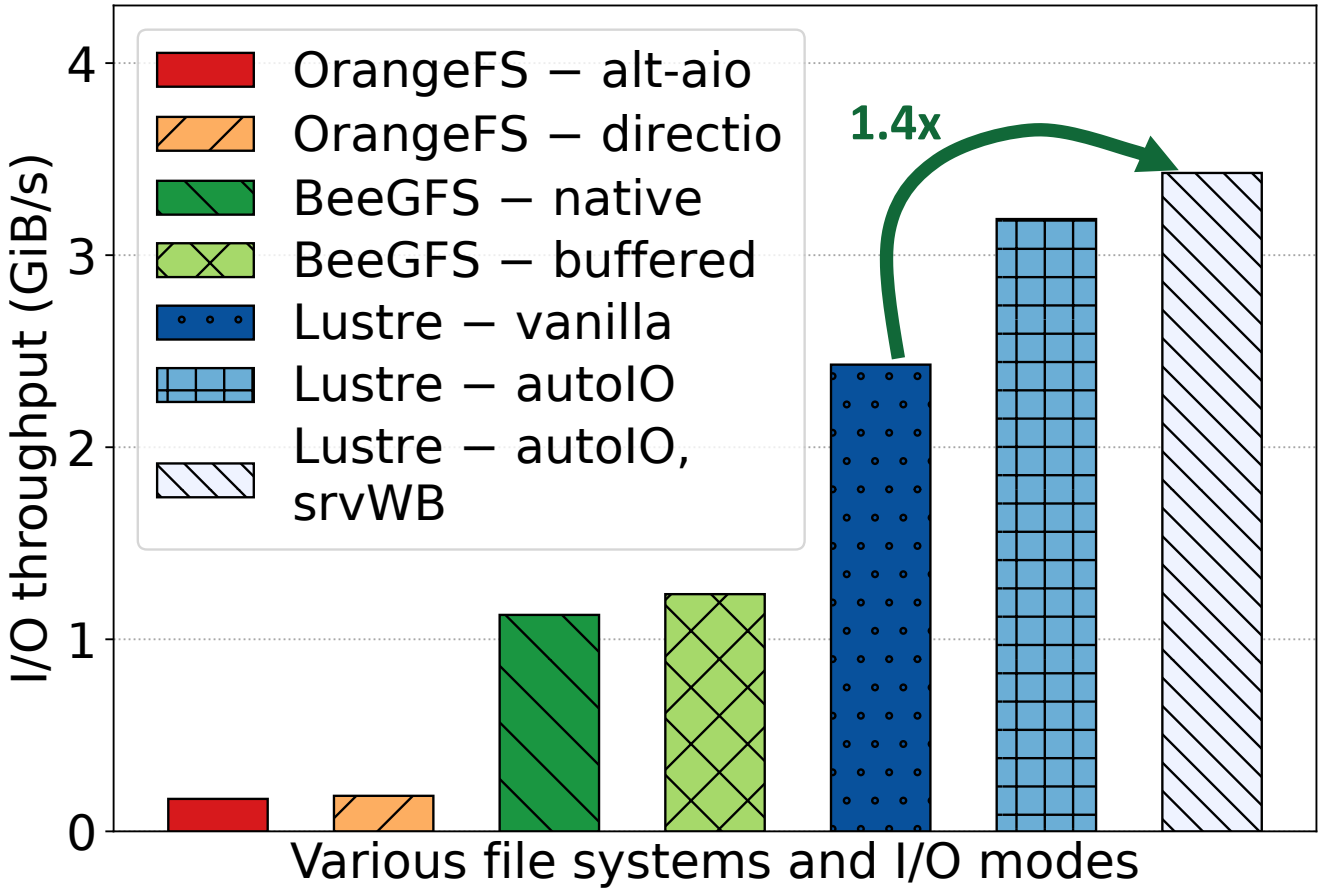
- Nek5000 turbPipe workload for 32 nodes (16 processes each)



Nek5000's turbPipe I/O throughput

Nek5000 (CFD workload) – I/O throughput

- 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	372,281
Direct I/O - large threshold	128,000
Direct I/O - lock contention	132,000
Buffered I/O - default	65

Hybrid I/O v2

- Switch to direct I/O under more conditions where it is provably improves performance
- Lock contention detection – LU-12550 (#35287) + LU-16964 (#51679)
  - Server counts conflicting locks over a 4 s sliding window to flag contended files
  - Flag latched for a few seconds (reduced from 30 s)
  - Shared-file workloads switch to DIO in a decision window
- HDD-aware threshold – LU-13802 (#52776, #52777)
  - Per-file detection of HDDs
  - Raise cutoff
  - Buffered read-ahead amortizes seek cost

Hybrid I/O v2 cont’

- Server-side write-back (srvWB) (LU-12916)
  - Small writes hit server page cache — no per-RPC disk wait
  - Tunable (TBD): `osd-ldiskfs.*.writeback_max_io_kb` (tentative default 64 KiB)
  - Beneficial, e.g., for IO500 ior-hard write workloads
- What about the DIO path rewrite (LU-13814)?
  - Converts DIO from cl_page state-machine to simple page arrays
  - Hybrid + LU-13814 → Around 45 GiB/s for a single thread, up from 20 GiB/s
  - Seems superseded by large folios
- Large folios (LU-19895) - Shaun Tancheff (HPE)
  - Per-folio bookkeeping instead of per-page, e.g., ~512× fewer per-page ops for a 2 MiB I/O
  - Narrows the gap between Hybrid and pure DIO — bounce-buffer memcpy is the remaining ceiling

Recap and closing

- Hybrid I/O combines the benefits of buffered I/O and direct I/O
  - Transparent to the application
  - Buffered performance for small reads & writes
  - Direct I/O-like scaling for large reads & writes
  - Direct I/O is unchanged – if `O_DIRECT` is used, it remains direct I/O
- Hybrid I/O is enabled by default since Lustre 2.17
- AutoIO research highlighted further beneficial extensions
- Some can be considered for Hybrid I/O v2, e.g.,
  - Lock contention detection
  - A decision window to adjust the direct I/O threshold, e.g., under lock contention
  - Server-side write-back to improve small I/O on the OSTs
  - HDD-aware threshold

Whamcloud

Thank you!

Marc-André Vef - mvef@whamcloud.com