Whamcloud

LUG 2022: Unaligned DIO & I/O Path Futures

Patrick Farrell

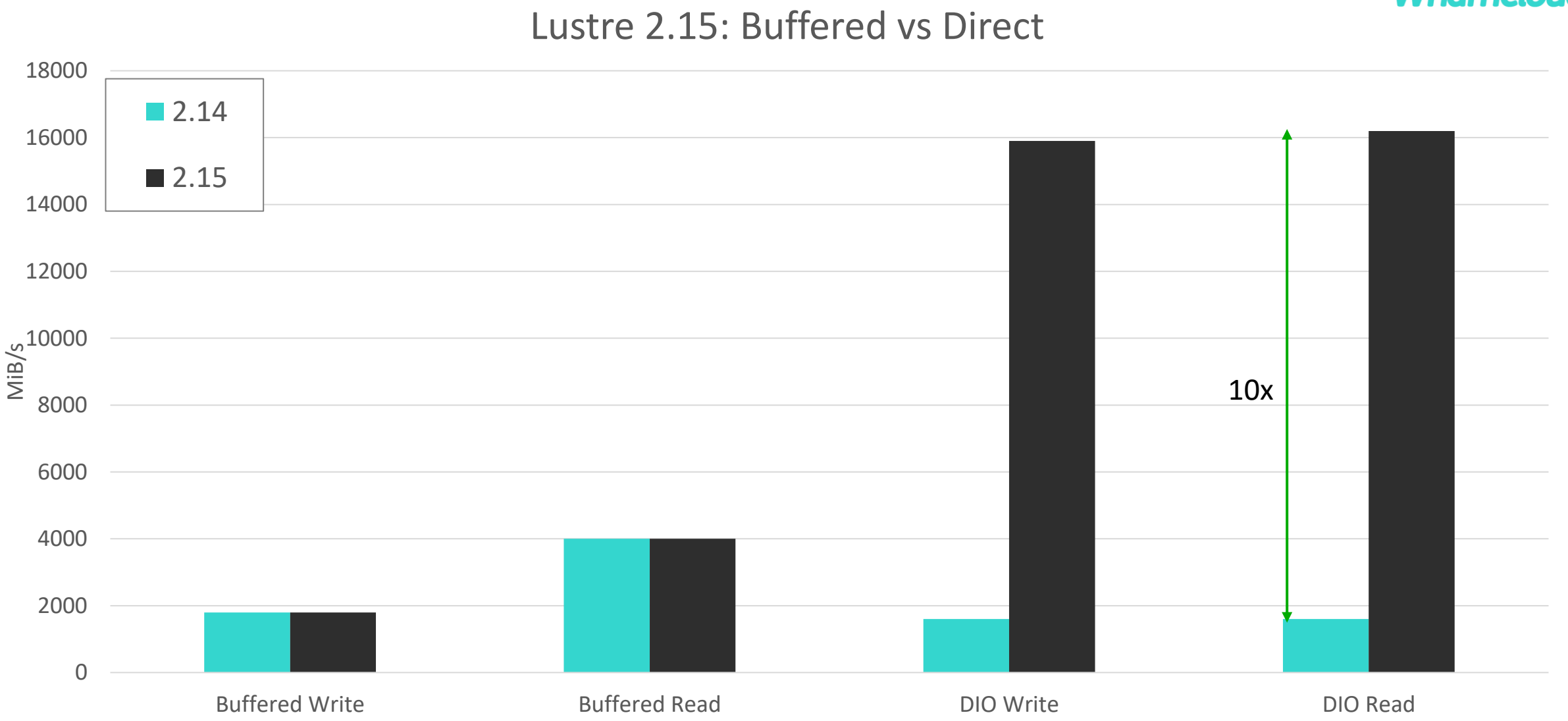
ddn

Lustre Data I/O Path

Whamcloud

- The data I/O path is “What Lustre does when you call read() or write()” (or access mmap’ed data)
- Data flows from userspace, through the client, over the network, and to storage (and back)
- We’re going to talk about the client part.
- POSIX gives two ways to do data I/O:
  Buffered I/O
  Direct I/O
  - Mmap is a type of buffered I/O

Digression: Direct I/O Improvements in 2.15



Lustre Data I/O Path: Direct I/O

Whamcloud

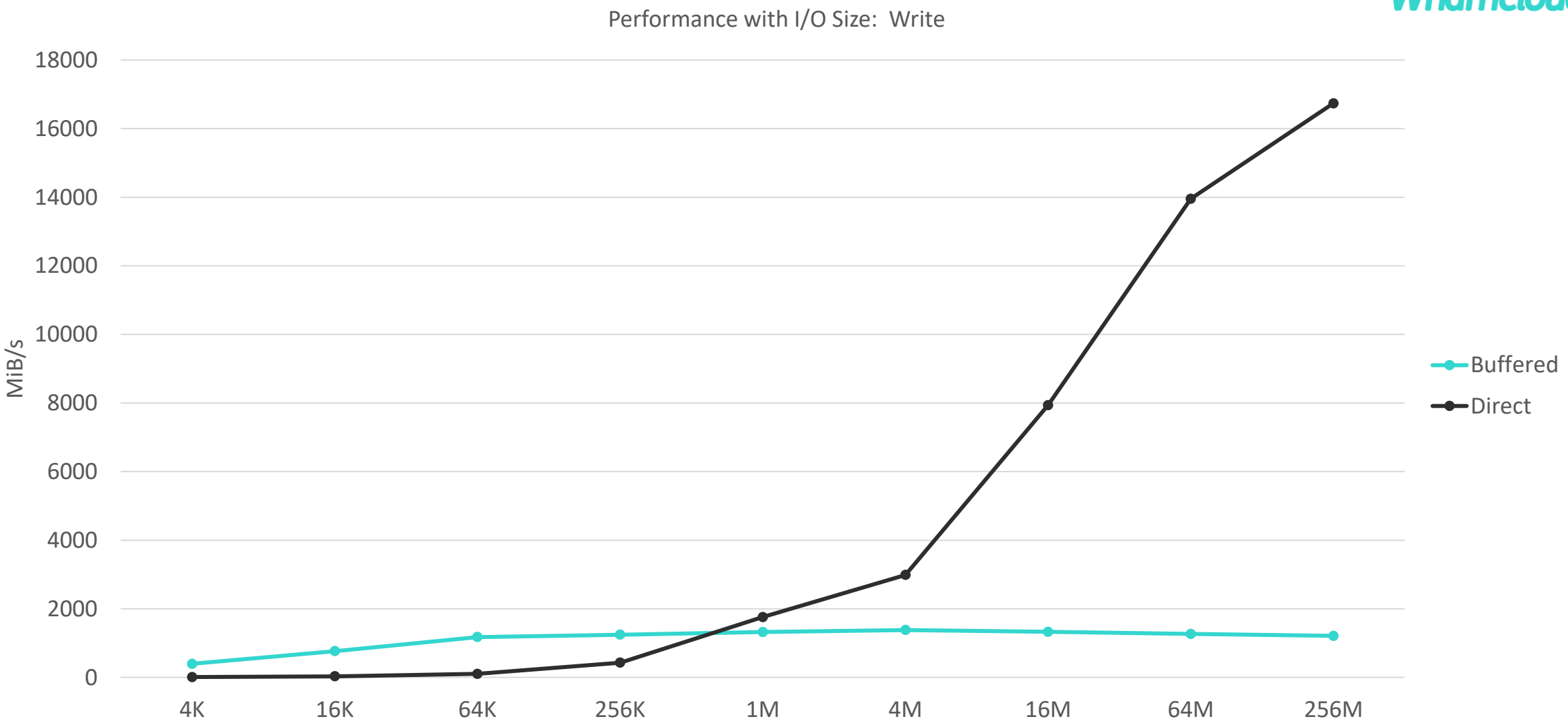
- Buffered means ‘Uses the Linux page cache’
  - Good: Allows read ahead and write aggregation, converting small application I/O to large file system operations
  - Good: Async writes and readahead are perfect for hiding latency of slow devices (HDD)
  - Good: Allows any I/O – no alignment requirements
  - Bad: Low single stream performance (max a few GiB/s) due to cost of the page cache
  - Bad: Bottlenecks for multiple processes to 1 file, due to page cache locking
- Direct I/O means ‘Direct from user memory, does not use the page cache’
  - Good: Very high single stream performance with large I/O – 18+ GiB/s
  - Good: Minimal locking, means no bottlenecks
  - Good: Very linear scaling as processes are added (to 1 file or many files)
  - Bad: Synchronous. I/O must go directly to disk → Exposes latency of slow devices. (Bad for small I/O.)
  - Bad: Alignment requirements. Both size of I/O and location in memory must be a multiple of page size. (Means most applications cannot use it.)

Buffered vs Direct: Summary

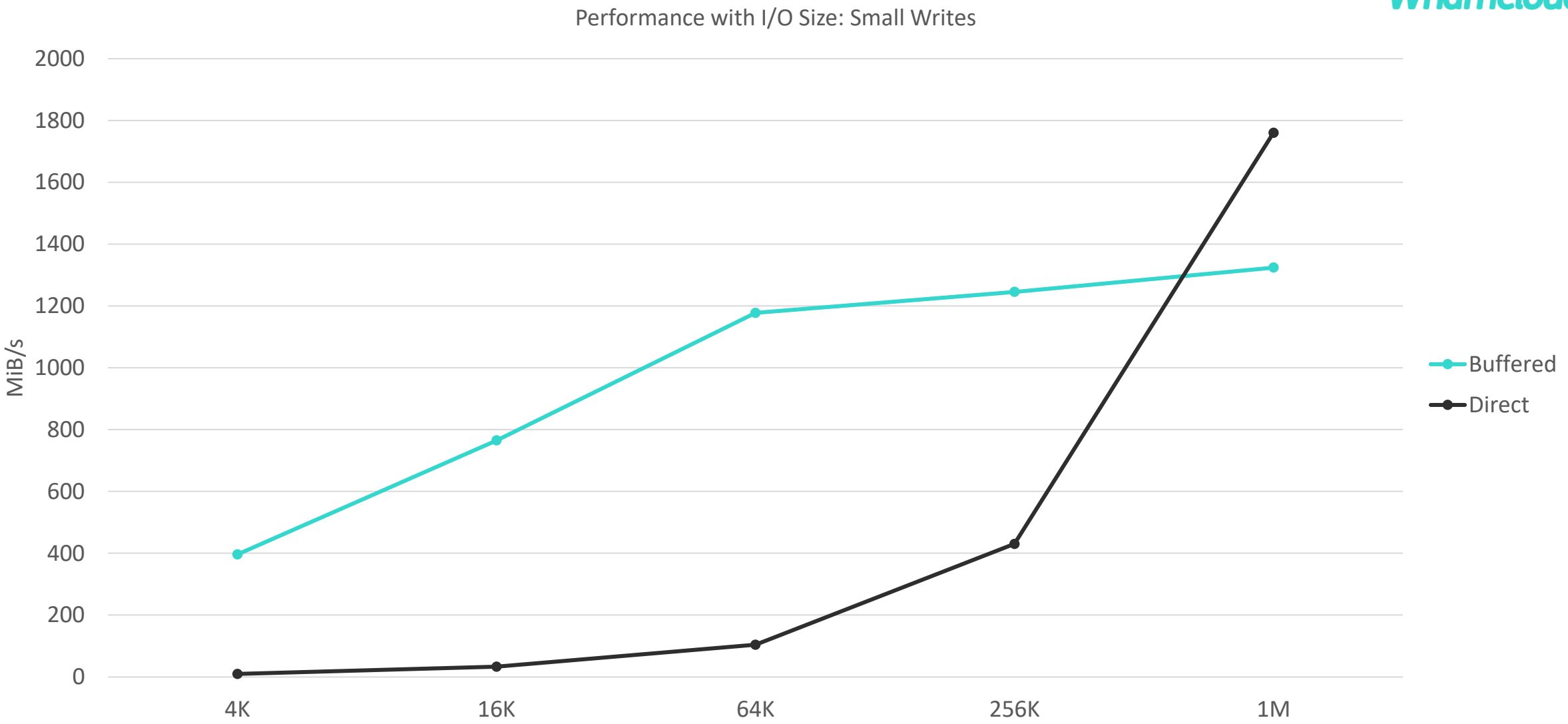
Whamcloud

	Buffered I/O	Direct I/O
Small I/O Performance	✓	X
Large I/O Performance	X	✓
Many Processes	X	✓
High latency Storage	✓	X
Unaligned I/O	✓	X

Buffered vs Direct: Performance with I/O Size



Buffered vs Direct: Small I/O Performance



Buffered + Direct: Can we have it all?

Whamcloud

- Strengths and weakness of buffered I/O and direct I/O pair up perfectly
- Can we dynamically select the one to use?
- Use buffered I/O for small I/O and direct I/O for large I/O
- We could even maybe do it inside the file system – no need for app changes
- But there's a sticking point:
  Alignment requirements. Can't do arbitrary I/O as direct I/O, because I/O can be any size (and memory isn't usually aligned).

- Let's step back and consider.

I/O Alignment: Why is it required?

Whamcloud

- No read-modify-writes for blocks (block size is always <= PAGE_SIZE)
  - Conflicting read-modify-write ops to the same block must be resolved
  - Traditionally done in the page cache
  - Direct I/O avoids this by not allowing unaligned I/O
  - But Lustre can handle read-modify-write transparently on the server(!)
- Data for RDMA must be page aligned
  - Yes, some devices support unaligned RDMA, but it's much slower.
  - Buffered I/O data is aligned by the page cache before sending out
- Key point:
  Direct I/O is aligned in userspace, buffered I/O is aligned by the page cache, but both are aligned

Buffered I/O: Costs of Alignment(?)

Whamcloud

- Buffered uses page cache to get alignment, but page cache is expensive, because:
  - Memory allocation
  - Memcopy()
  - Cache management: Setup & tracking
- Let's break down the cost of these…
- **"Data copying is bad"**
  - Article of faith.
- But cache setup is **much** worse.
- For large buffered I/O, Lustre spends:
  - 15% of time on data copy
  - **~65%(!)** on cache setup
  - 10% allocation (10% other)
- Allocating every page, locking it and inserting into the cache…. Costs pile up.

Aligning I/O: No cache required

- We don’t need a cache to get alignment – just a buffer:
- A cache can be used repeatedly & accessed from multiple threads
  - Requires concurrency management and locking
- Buffer is local to the I/O which created it
  - Only one thread can ever access it – no locking required
- No need for cache setup or locking(!)

Unaligned Direct I/O

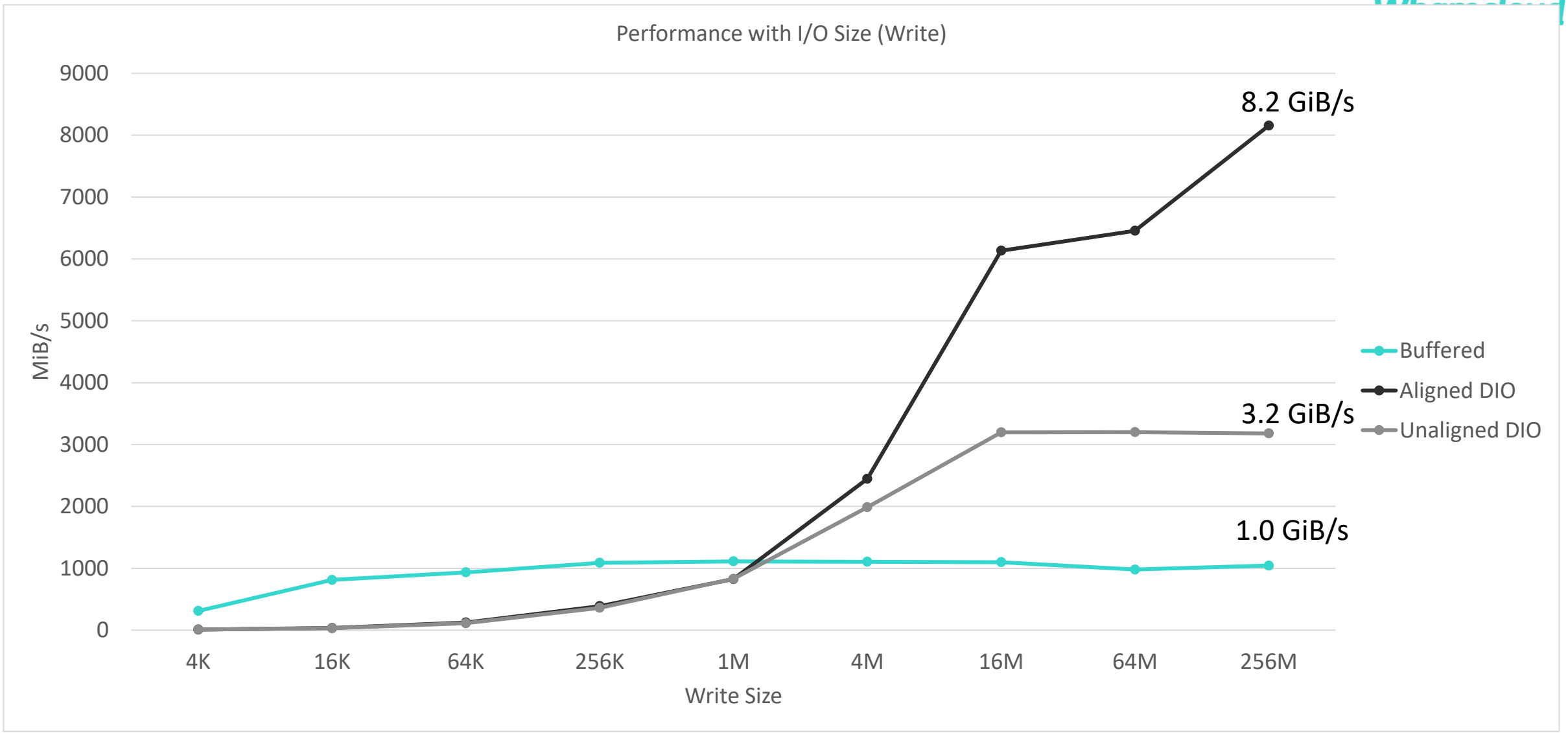
- Inside the kernel…
  - Allocate an aligned buffer
  - Copy data to/from the buffer
  - Do direct I/O from the buffer
    - (Swap last two steps for read vs write)
- Saves the 65% of time spent on cache setup
- Implies a buffered I/O speedup from 1400 MiB/s to ~4 GiB/s (for single threaded workloads)
- Let’s talk numbers…

Caveat on Numbers

Whamcloud

- Hardware is different – This hardware can only do ~10 GiB/s single threaded DIO, not 18 GiB/s (max on other hardware)
- Consider unaligned DIO in relative terms to DIO
- This is a prototype and missing various optimizations…

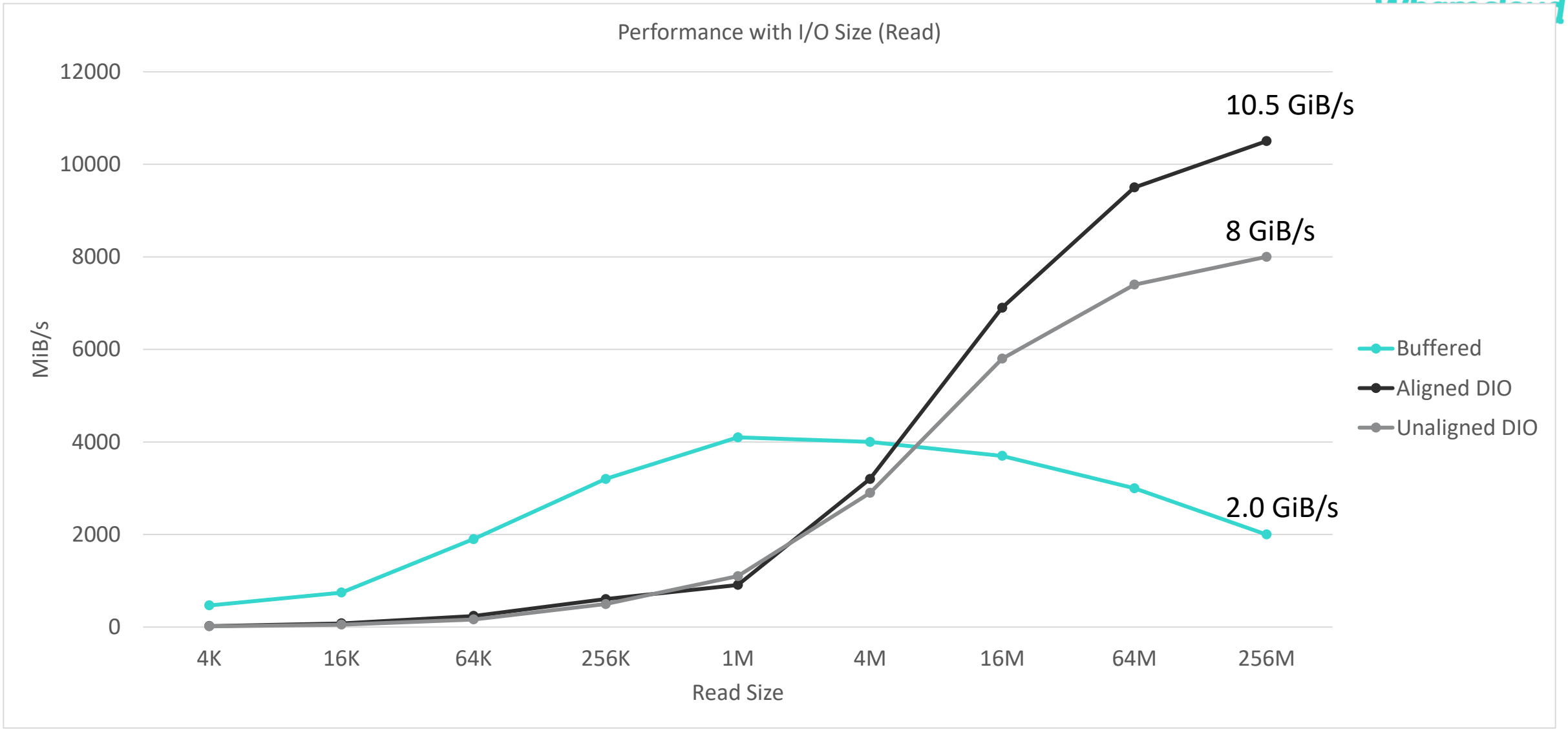
Unaligned DIO: Write Performance



Unaligned Direct I/O: Performance

- 3.2 GiB/s single threaded write is nice, but just 40% of aligned DIO (8 GiB/s here)
- Well, data copy and memory allocation are pretty time consuming.
- But, yes, we can do better.
- Memcopy() for buffered I/O is single threaded, because it's not any faster to parallelize – locking and coordination of cache bottlenecks
- But DIO is different…

Unaligned DIO: Read Performance



Unaligned Direct I/O: Performance

Whamcloud

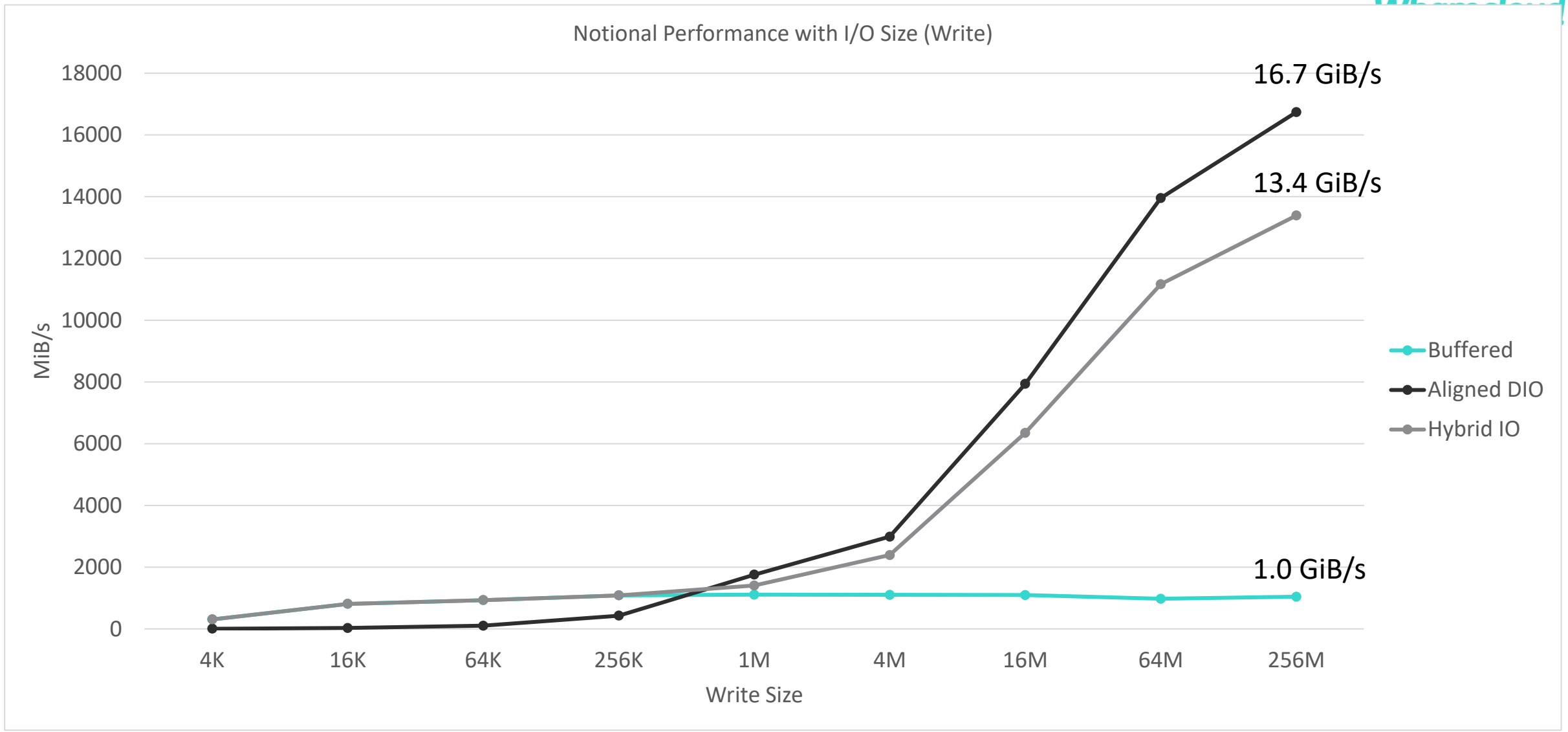
- Unaligned DIO read is at 8 GiB/s of 10.5 GiB/s for DIO (76%)
- Copy for unaligned DIO read is parallelized
  - Farms out data copy for each DIO to many daemon threads
- Data copy for write **will** be parallelized, but is trickier – not done yet in the prototype
- Read does not have allocation parallelized, just copy
- Read & write will have both allocation and copy parallelized, so expect >76% of DIO performance
- Will scale with DIO performance – 18 GiB/s DIO implies ~13 GiB/s unaligned DIO
  - Sublinear scaling, % relative to DIO will drop as DIO speed rises

Unaligned Direct I/O & Hybrid I/O: The Plan

Whamcloud

- Implement unaligned direct I/O
- Test and optimize
- Once performance is good and bugs worked out:
- Implement hybrid I/O path
  - Userspace does simple read() or write() calls
  - Lustre decides internally to do page cache I/O, or to do unaligned direct I/O (or aligned direct I/O if possible)
  - Gets the best of both worlds – readahead and write aggregation at small sizes, high efficiency at large sizes

Hybrid I/O: Dreaming big



Unaligned Direct I/O and direct I/O: Future work

Whamcloud

- Unaligned direct I/O: Lustre 2.16
  - Will allow direct I/O which is not a multiple of page size
  - Still strictly **opt-in**, does nothing if you're not using O_DIRECT
- Hybrid I/O: 2.16+
  - No firm plan – depends on other commitments
  - Aiming for gradual phase in - use in more situations as we can be sure it improves performance there
- Further DIO efficiency improvements
  - Referenced in previous work – DIO path is 18 GiB/s today, can be pushed to 25-35 GiB/s
  - Will boost hybrid I/O path

Thank you

- Thank you for listening.
- See LU-13805 for further details
- See my LAD ‘21 talk for more on DIO improvements
- Questions to pfarrell@whamcloud.com
- Thanks to Nathan Rutman for an interesting question in 2020