intel Look Inside.™

Lustre* Client I/O Parallelization

Dmitry Eremin

LUG17

Agenda

- What issues are we solving?
- Current I/O performance
- I/O data flows and Flame Graphs
- Ideas of Improvements
- Results

What issues are we solving?

- Single-threaded applications cannot utilize performance benefits of multiple I/O operations in Lustre* if a single core is slow
- The Network bandwidth cannot be saturated because of
  - Intel® OPA fabric uses a host CPU for packets processing
  - High overhead of single thread execution
  - Slow memory transfer operations
- Blocking the userspace I/O operation
  - To do Read Ahead

Number of cores in client machines is continually increasing, yet single-threaded I/O is still common.

A solution other than adding more compute nodes is needed.

Hardware

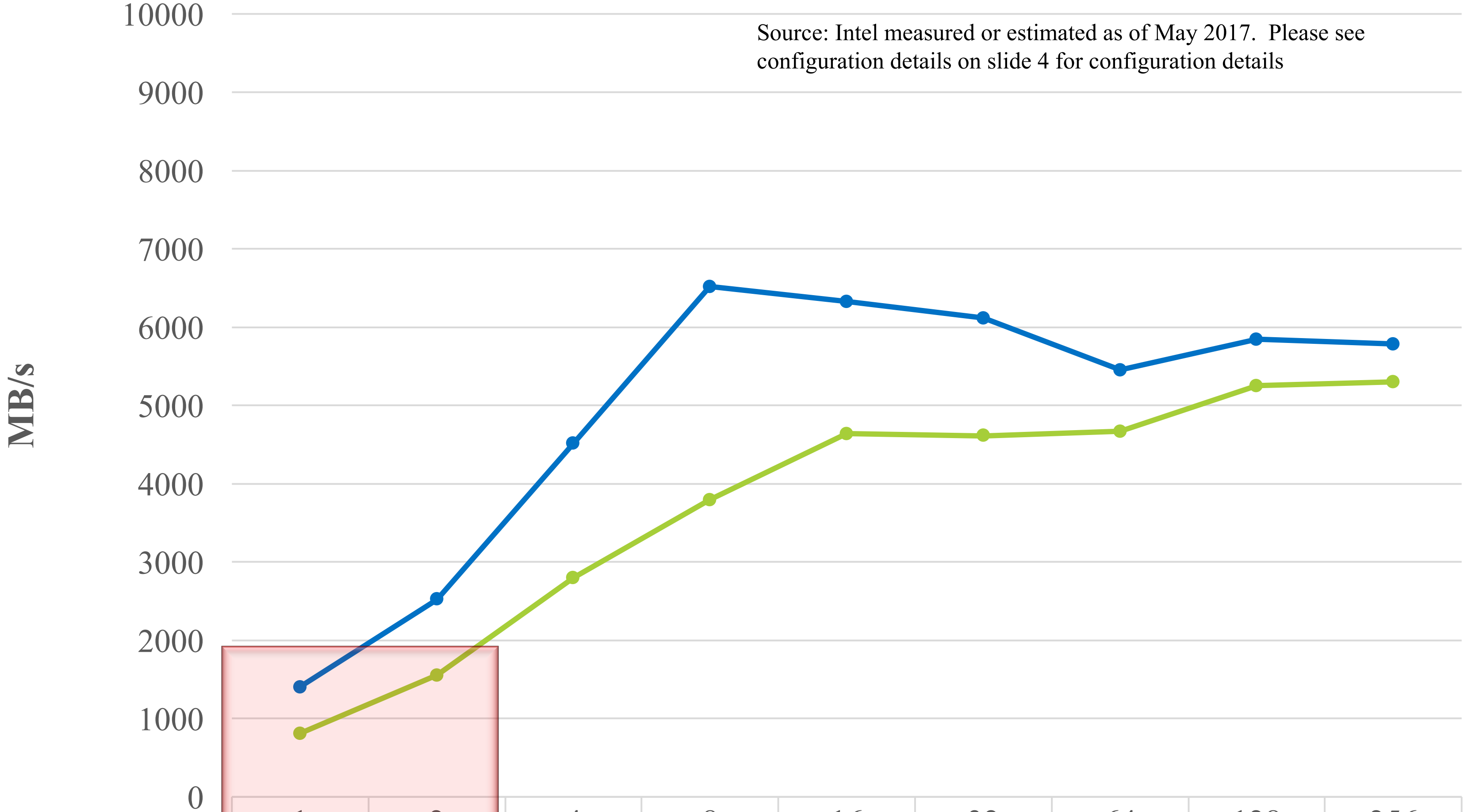
Intel® Xeon® CPU E5-2697 v4 @ 2.30GHz

- L2 cache: 256K, L3 cache: 46080K
- CPUs: 72, Cores: 36, Threads per core: 2
- Intel® OPA HFI Silicon 100 Series [discrete]

Intel® Xeon Phi™ CPU 7250 @ 1.40GHz

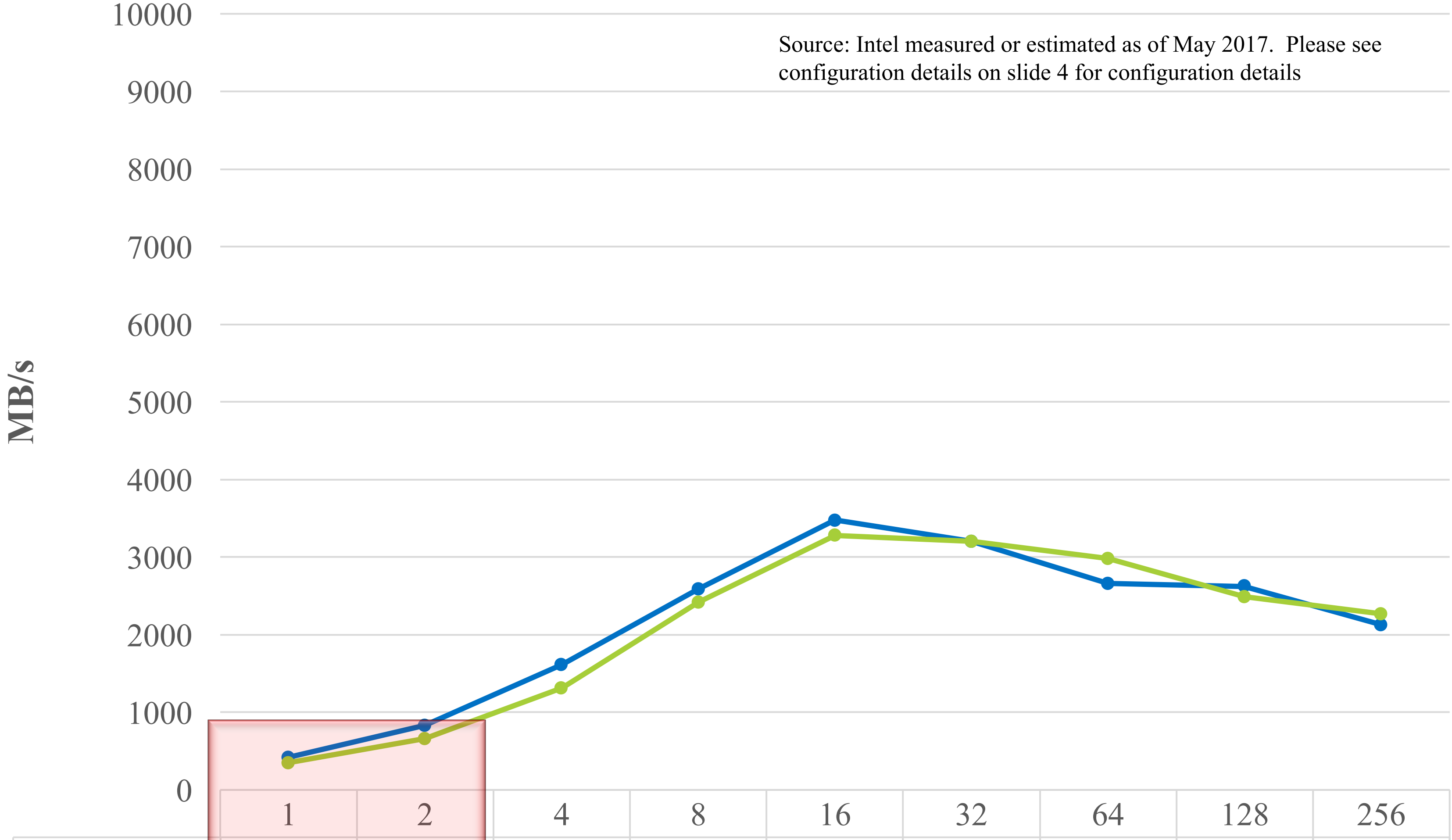
- L2 cache: 1024K
- CPUs: 272, Cores: 68, Threads per core: 4
- Intel® OPA HFI Silicon 100 Series [discrete]

Intel® Xeon® E5-2697 IOR results current version of Lustre*



	1	2	4	8	16	32	64	128	256
BDW write	1407.32	2524.64	4519.91	6518.61	6324.85	6112.59	5457.38	5845.68	5780.83
BDW read	811.99	1552.78	2800.55	3795.03	4643.87	4614.23	4668.15	5250.87	5300.38

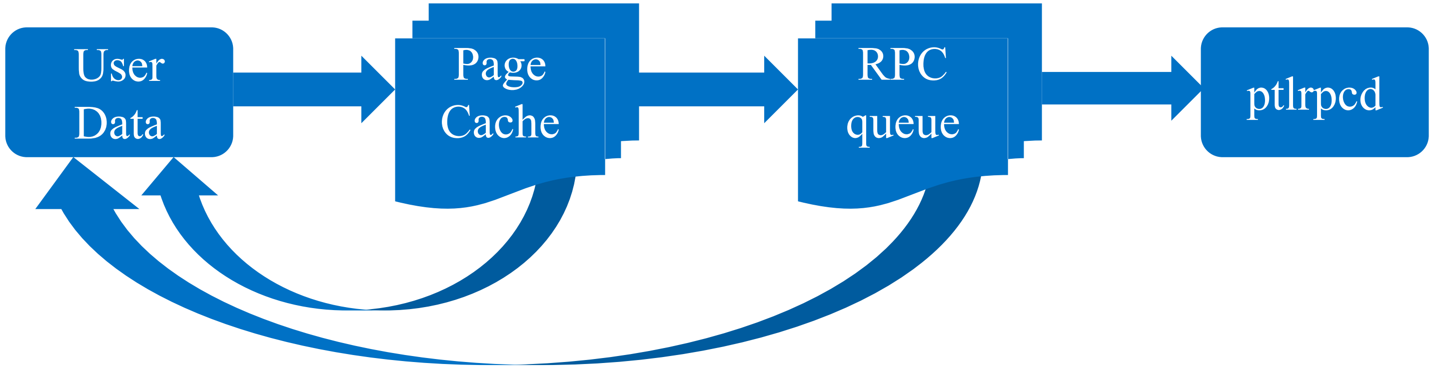
Intel® Xeon Phi™ 7250 IOR results current version of Lustre*



	1	2	4	8	16	32	64	128	256
KNL write	420.17	829.18	1619.05	2588.43	3478.02	3202.02	2663.39	2625.08	2131.31
KNL read	351.04	663.89	1313.74	2419.18	3278.37	3201.24	2984.83	2488.51	2271.53

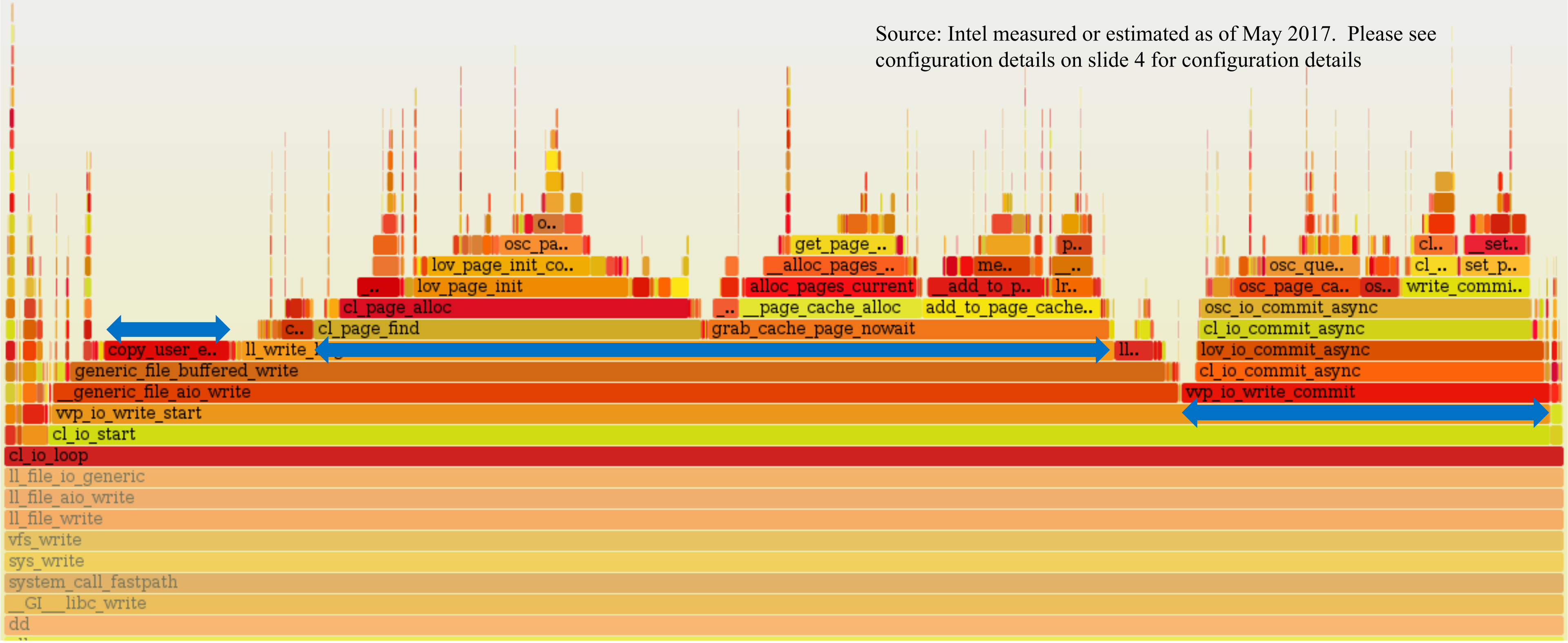
Write operation data flow

- For each PAGE_SIZE of user data:
  - Allocate new page in page cache (`cl_page_alloc()`, `grab_cache_page()`)
  - Copy data from user space to page cache (`copy_user_enhanced_fast_string()`)
- Submit pages to RPC queue (`vvp_io_write_commit()`)
- Wake up ptlrpcd thread and wait for completion



Flame Graph of Write operation

- Page cache allocation is ~½ of all write time
- Submit pages to RPC queue
- Copy user data

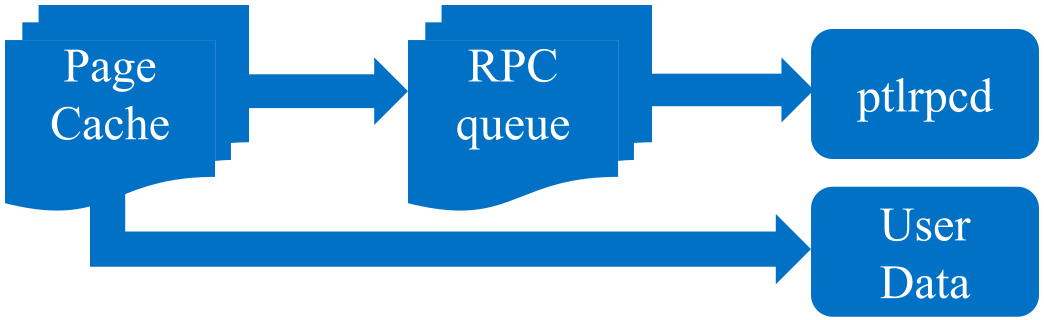


Read operation data flow

- Read Ahead (`ll_readahead()`) for each page:
  - Allocate new page in page cache (`cl_page_alloc()`, `grab_cache_page()`)
  - Submit page to RPC queue for read (`cl_io_submit_rw()`)
- Wake up ptlrpcd thread and wait for completion

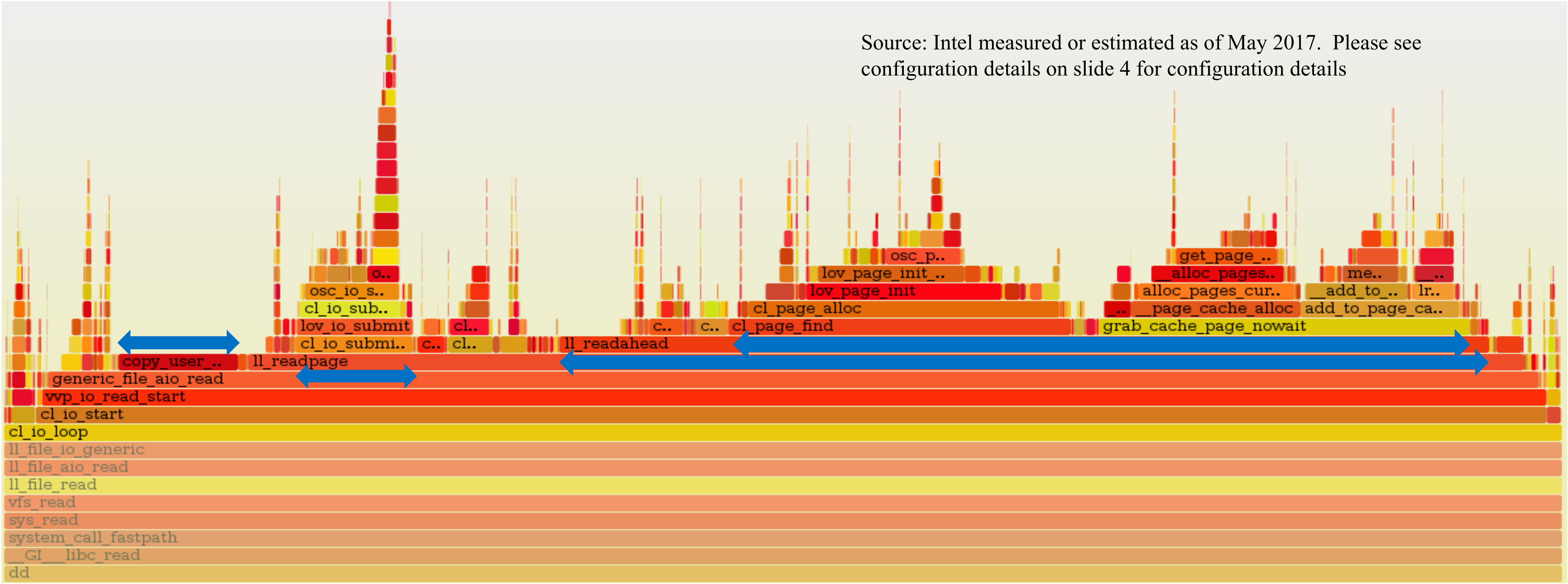
…

- Copy data from page cache to user space (`copy_user_enhanced_fast_string()`)



Flame Graph of Read operation

- Read Ahead is ~½ of all read time
  - Page cache allocation is ~¾ of Read Ahead
- Submit pages to RPC queue for read
- Copy user data



Source: Intel measured or estimated as of May 2017. Please see configuration details on slide 4 for configuration details

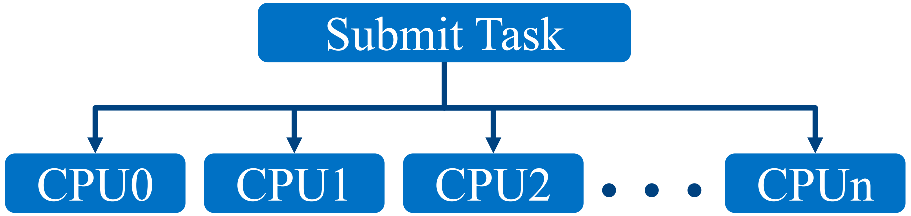
Ideas of Improvements

- Split time consuming operations into several threads
  - Page cache allocation in parallel
  - Submit pages into several RPC queues in parallel
  - Copy user data in parallel**
- Move Read Ahead into asynchronous threads
  - Resume main read thread as soon as possible
  - Avoid cost of unused reads (false read ahead)
- Utilize more ptlrpcd threads
  - Process network packets in parallel**
- Multiple network connection
  - Use multiple IB/OPA endpoints between nodes

** *Very important for less performant cores.*

Introduce new Parallel Tasks API

- Farm tasks out to be done in parallel on multiple CPUs
- Use system thread pool (don't create new thread pool)
- Provide ordered or disordered task completion
- Submit for execution in Round Robin fashion



https://jira.hpdd.intel.com/browse/LU-8964

Multiple IB/OPA Endpoints

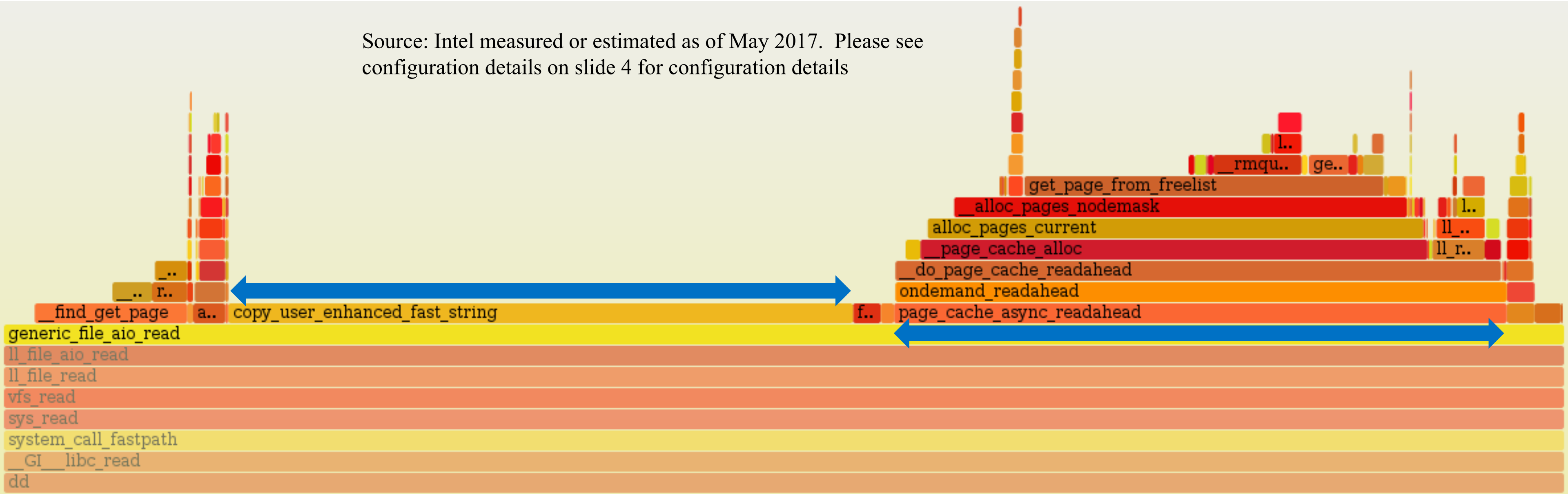
- Lustre* use a single network channel between nodes
  - Intel® OPA fabric use a host CPU for packets processing, so on machine with slow cores it cannot utilize full network bandwidth

- Optimization of IB/OPA LND driver
  - create multiple endpoints and balance the traffic over them

- https://jira.hpdd.intel.com/browse/LU-8943

Flame Graph of Read with async RA

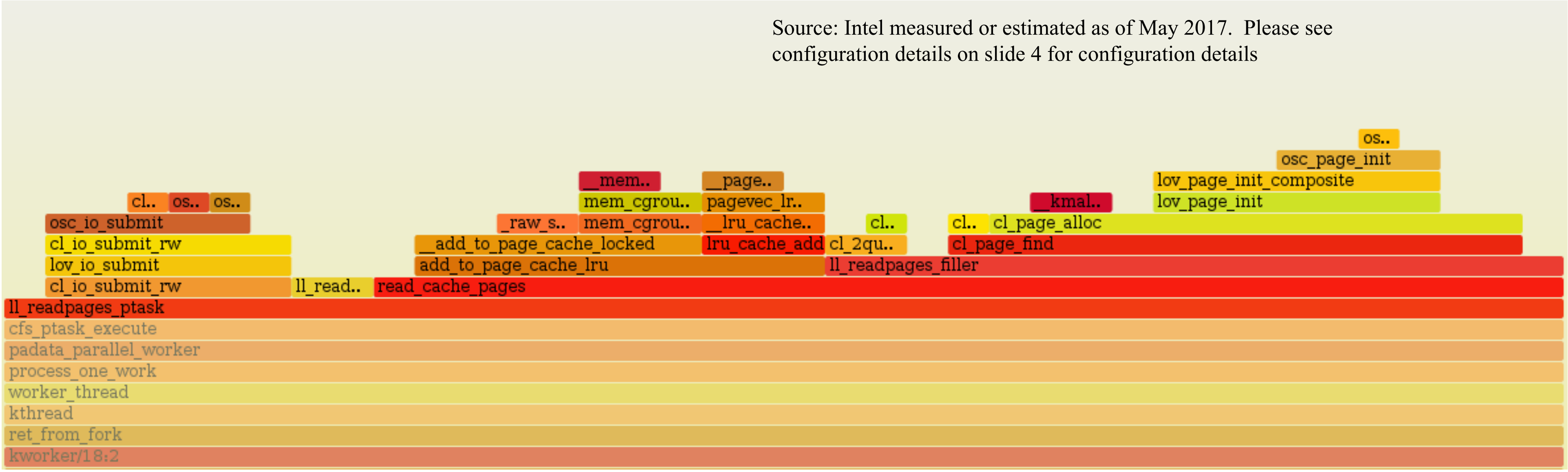
- Copy user data is ~½ of all read time
- Initial Read Ahead



https://jira.hpdd.intel.com/browse/LU-8964

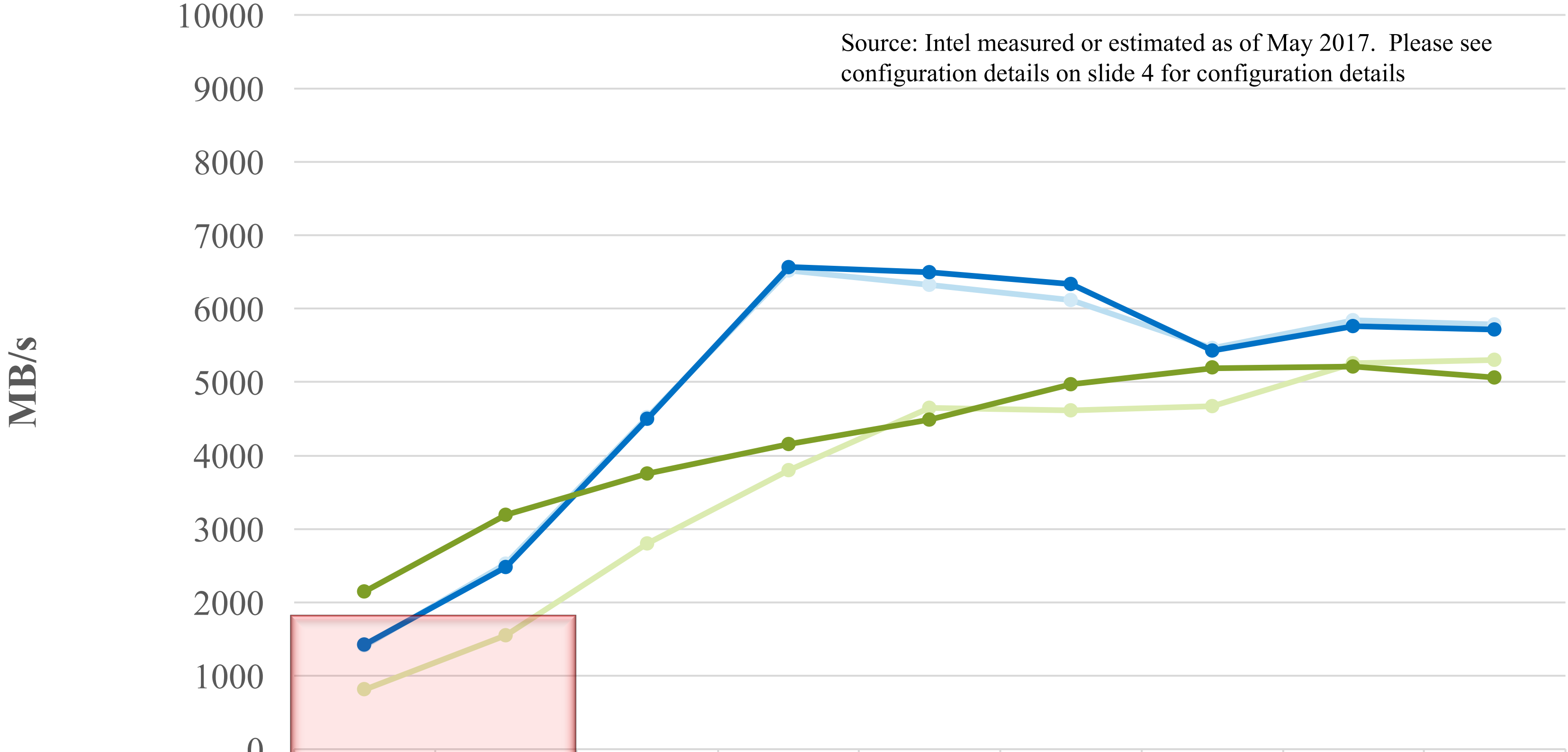
Flame Graph of Read Ahead thread

- Load the data in parallel (don’t serialize an user read)
- Can be stopped asynchronously at any time



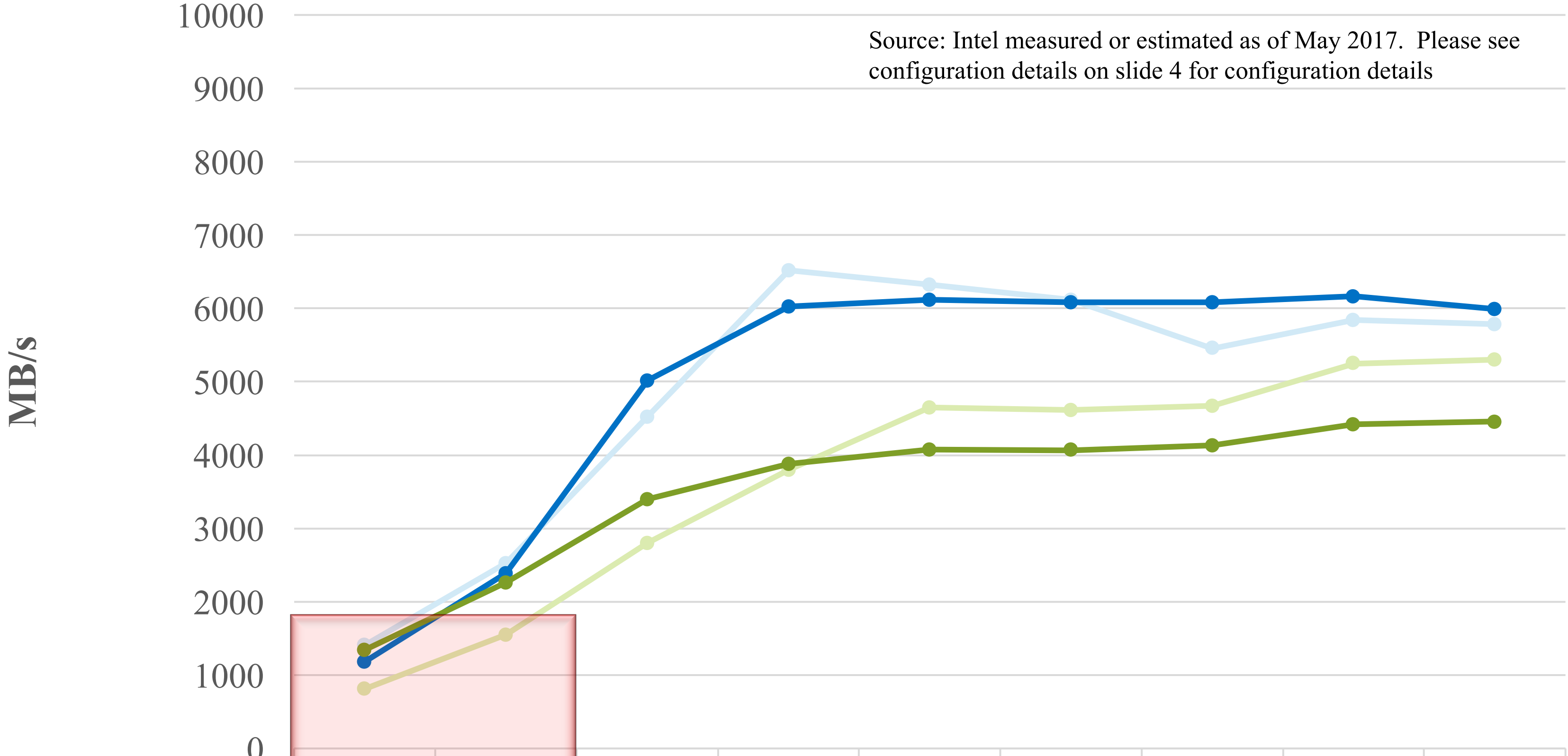
https://jira.hpdd.intel.com/browse/LU-8964

Intel® Xeon® E5-2697 IOR results
Parallel I/O off, async Read Ahead



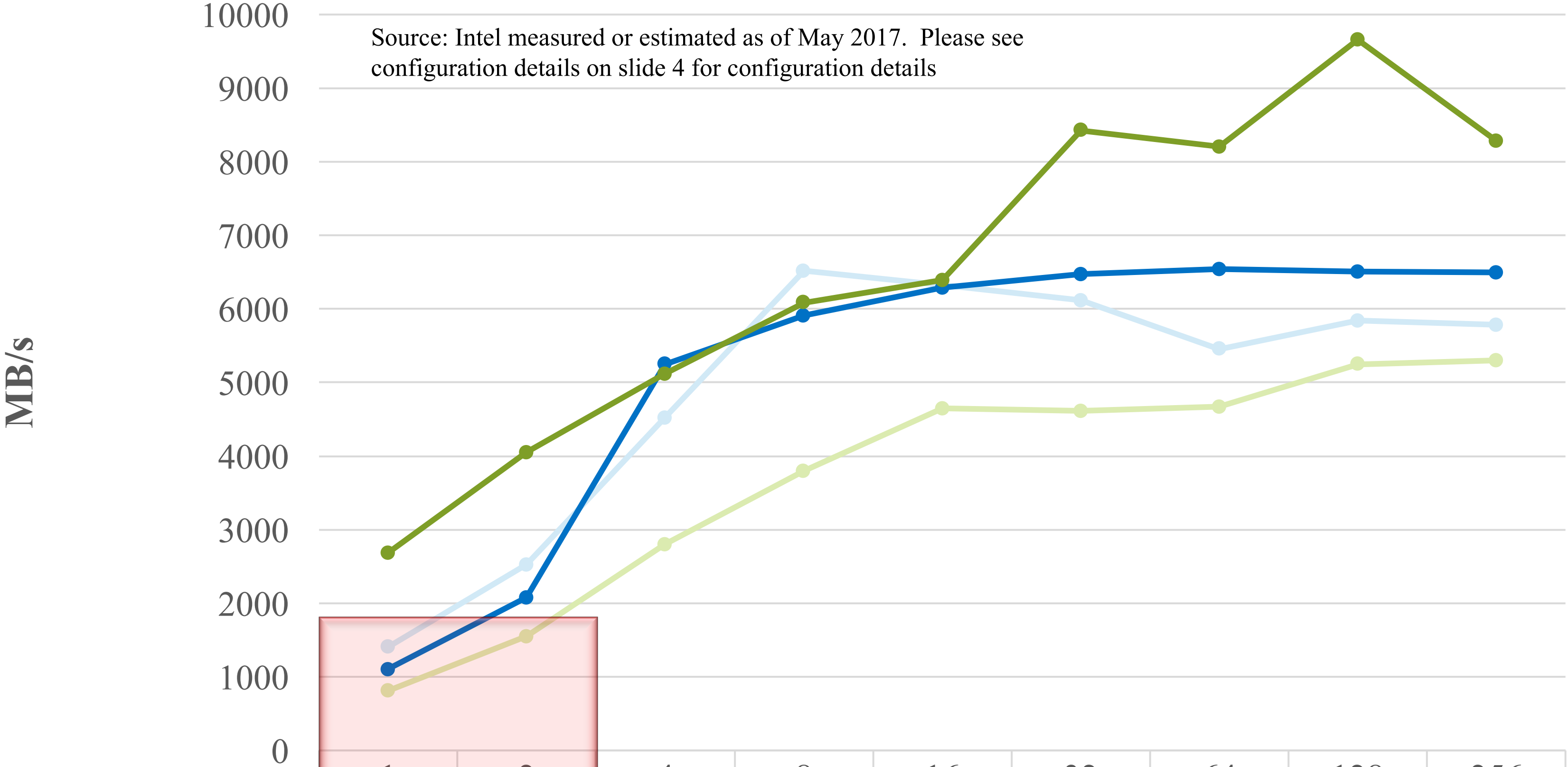
	1	2	4	8	16	32	64	128	256
BDW write	1407.32	2524.64	4519.91	6518.61	6324.85	6112.59	5457.38	5845.68	5780.83
BDW read	811.99	1552.78	2800.55	3795.03	4643.87	4614.23	4668.15	5250.87	5300.38
BDW-p0 write	1426.41	2476.15	4502.94	6567.18	6498.8	6331.32	5423.69	5760.38	5715.16
BDW-p0 read	2147.67	3194.38	3754.85	4155.02	4487.33	4965.86	5193.18	5210.98	5066.07

Intel® Xeon® E5-2697 IOR results
Parallel I/O on, async Read Ahead



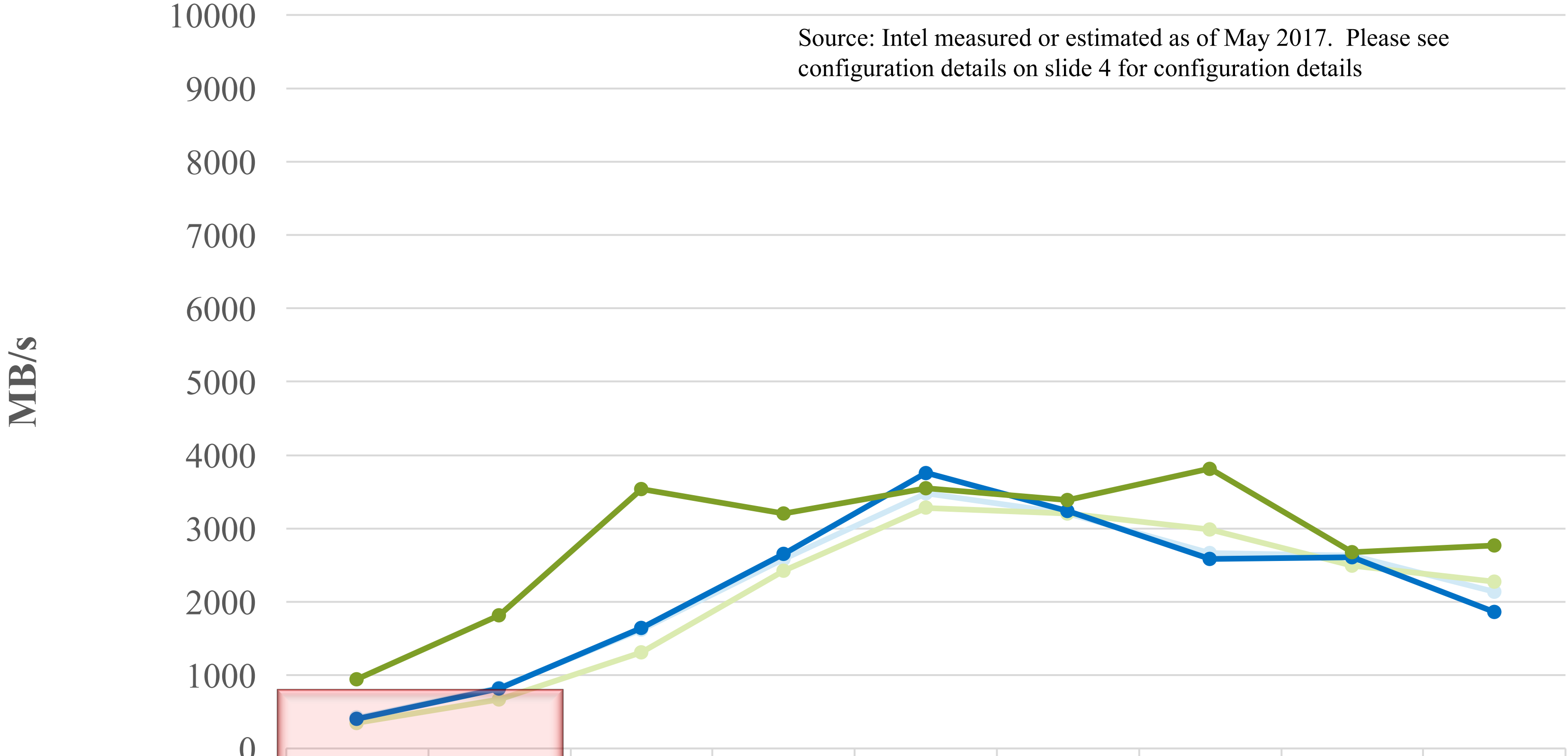
	1	2	4	8	16	32	64	128	256
BDW write	1407.32	2524.64	4519.91	6518.61	6324.85	6112.59	5457.38	5845.68	5780.83
BDW read	811.99	1552.78	2800.55	3795.03	4643.87	4614.23	4668.15	5250.87	5300.38
BDW-p1 write	1183.07	2386.63	5013.92	6026.65	6118.27	6087.45	6077.98	6165.4	5993.75
BDW-p1 read	1343.58	2262.91	3397.42	3878.06	4078.49	4069.03	4133.6	4423.03	4457.81

Intel® Xeon® E5-2697 IOR results
Parallel I/O on, async Read Ahead, MQP



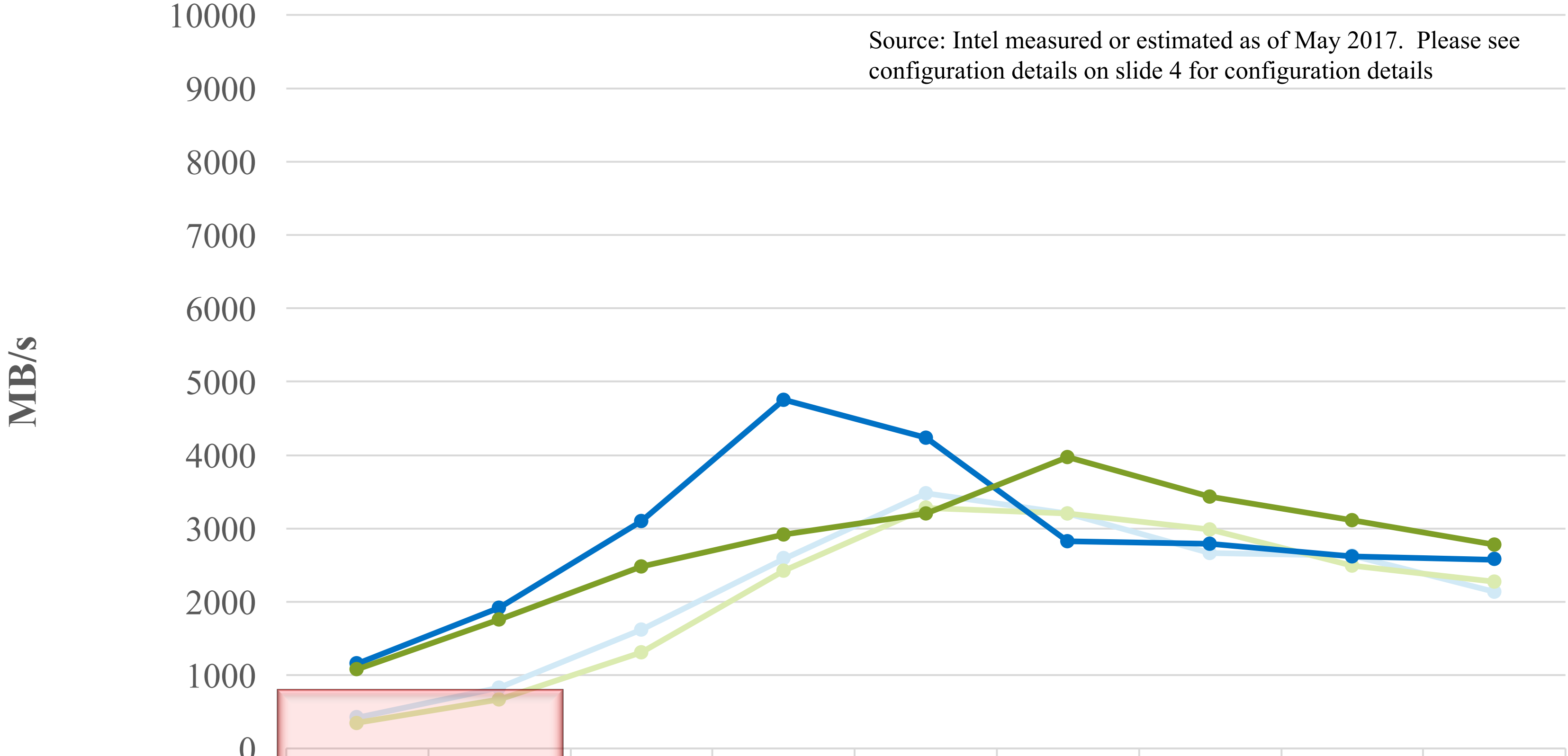
	1	2	4	8	16	32	64	128	256
BDW write	1407.32	2524.64	4519.91	6518.61	6324.85	6112.59	5457.38	5845.68	5780.83
BDW read	811.99	1552.78	2800.55	3795.03	4643.87	4614.23	4668.15	5250.87	5300.38
BDW-mp1 write	1108.65	2079.09	5250.85	5907.89	6291.11	6471.52	6541.4	6501.82	6500.9
BDW-mp1 read	2687.82	4055.54	5119.49	6088.47	6394.05	8428.6	8202	9657.81	8286.17

Intel® Xeon Phi™ 7250 IOR results
Parallel I/O off, async Read Ahead



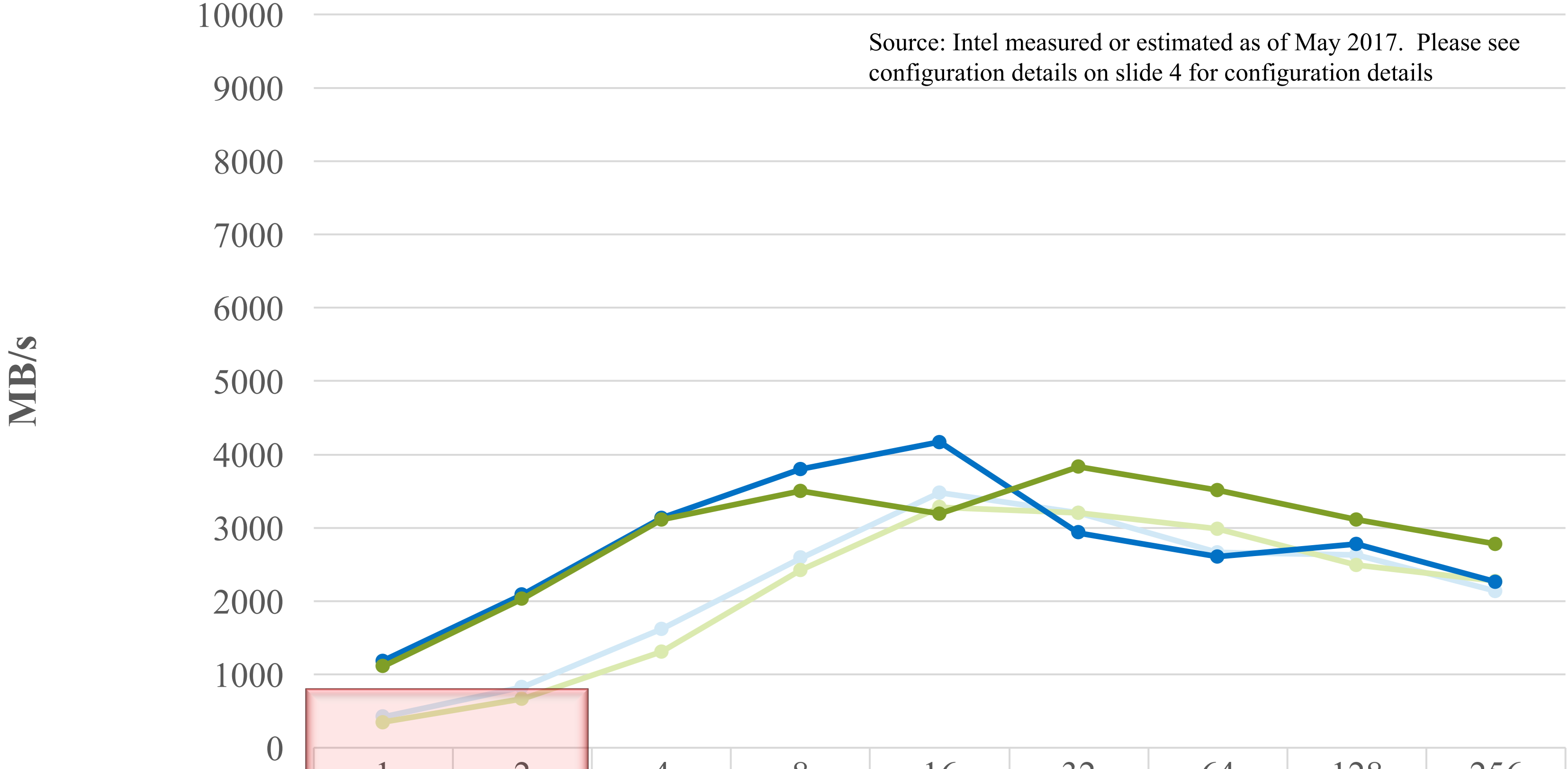
	1	2	4	8	16	32	64	128	256
KNL write	420.17	829.18	1619.05	2588.43	3478.02	3202.02	2663.39	2625.08	2131.31
KNL read	351.04	663.89	1313.74	2419.18	3278.37	3201.24	2984.83	2488.51	2271.53
KNL-p0 write	405.06	814.89	1638.32	2656.96	3758.21	3241.44	2586.71	2609.21	1859.98
KNL-p0 read	947.38	1816.44	3536.47	3200.26	3549	3391.45	3815.71	2679.95	2761.81

Intel® Xeon Phi™ 7250 IOR results
Parallel I/O on, async Read Ahead



	1	2	4	8	16	32	64	128	256
KNL write	420.17	829.18	1619.05	2588.43	3478.02	3202.02	2663.39	2625.08	2131.31
KNL read	351.04	663.89	1313.74	2419.18	3278.37	3201.24	2984.83	2488.51	2271.53
KNL-p1 write	1156.08	1918.27	3097.96	4753.87	4236.18	2826.15	2791.5	2619.21	2577.41
KNL-p1 read	1080.52	1758.95	2478.52	2919.73	3204.82	3971.72	3436.16	3111.95	2774.6

Intel® Xeon Phi™ 7250 IOR results
Parallel I/O on, async Read Ahead, MQP



	1	2	4	8	16	32	64	128	256
KNL write	420.17	829.18	1619.05	2588.43	3478.02	3202.02	2663.39	2625.08	2131.31
KNL read	351.04	663.89	1313.74	2419.18	3278.37	3201.24	2984.83	2488.51	2271.53
KNL-mp1 write	1188.43	2085.11	3128.45	3794.76	4170.03	2932.48	2605.06	2776.42	2264.78
KNL-mp1 read	1111.12	2031.99	3114.42	3496.88	3195.69	3831.24	3511.97	3110.61	2782.79

Hybrid Coordinate Ocean Model (HYCOM) application

Source: Intel measured or estimated as of May 2017. Please see configuration details on slide 4 for configuration details

Testing a real world application - see https://hycom.org

Current version of Lustre*

zaio**	calls =	6	time =	0.39432	time/call =	0.06572032
zaiord	calls =	99	time =	2.24527	time/call =	0.02267950
zaiowr	calls =	726	time =	**81.19858**	time/call =	0.11184378
total	calls =	1	time =	5203.09187	time/call =	5203.09187198

Parallel I/O on, async Read Ahead, MQP

zaio**	calls =	6	time =	0.31822	time/call =	0.05303649
zaiord	calls =	99	time =	3.48087	time/call =	0.03516028
zaiowr	calls =	726	time =	**65.78485**	time/call =	0.09061274
total	calls =	1	time =	5149.86002	time/call =	5149.86001992

Summary

- Using parallel I/O and multiple IB/OPA endpoints Lustre* now can utilize performance benefits of
  - Multi-cores systems even for single-threaded applications
  - Multiple I/O operations in Lustre even if a single core is less performant
  - Intel® OPA fabric which uses a host CPU for packets processing
- Using asynchronous Read Ahead Lustre now
  - Doesn't block the userspace I/O operation to do Read Ahead
- For some workloads up to 2x speed-up in reads/writes has been observed
- Write work expected to be in 2.10.0; read work in 2.10.x maintenance release

Notices and Disclaimers

Questions?