Flash Memory Summit

Improved Flash Performance Using the New Linux Kernel I/O Interface

Vishal Verma: Performance Engineer, Intel

Acknowledgements: John Kariuki, Jens Axboe

Agenda

- Existing Linux IO interfaces & their challenges
- IO_uring- the new efficient IO interface
- Introduction to Liburing library
- Performance of IO_uring on Non-volatile media
- Summary

Existing Linux Kernel IO Interfaces

- Synchronous I/O interfaces:
  - Thread starts an I/O operation and immediately enters a wait state until the I/O request has completed
  - read(2), write(2), pread(2), pwrite(2), preadv(2), pwritev(2), preadv2(2), pwritev2(2)

- Asynchronous I/O interfaces:
  - Thread sends an I/O request to the kernel and continues processing another job until the kernel signals to the thread that the I/O request has completed
  - aio_read, aio_write, async io (aio)

Existing Linux User-space IO Interfaces

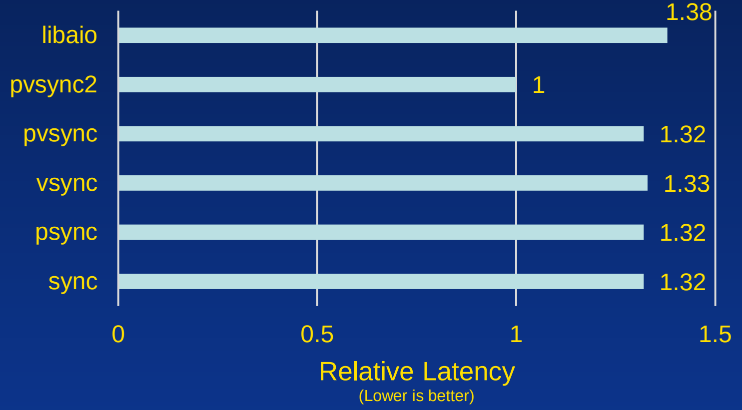
- SPDK: Provides a set of tools and libraries for writing high performance, scalable, user-mode storage applications
- Asynchronous, polled-mode, lockless design
- https://github.com/spdk/spdk.git

This talk will cover Linux Kernel IO Interfaces

The Software Overhead Problem

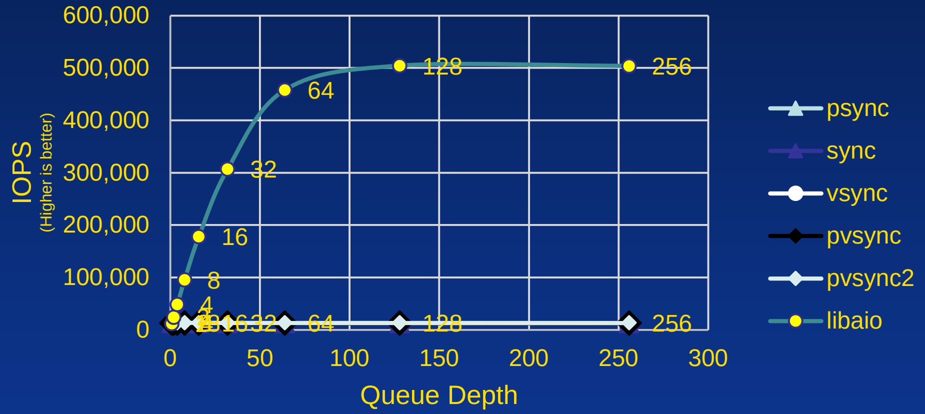
Intel® Optane™ SSD

4K Random Read Avg. Latency (us), Queue Depth=1



Intel® SSD DC P4610

4K Random Read IOPS, numjobs=1



Over 30% SW overhead with most of I/O interfaces vs. pvsync2 when running single I/O to an Intel® Optane™ SSD

Single thread IOPS Scale with increasing iodepth using libaio but other I/O interfaces doesn't scale with iodepth> 1

For test configuration details please see slide # 16

[1]Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance. Performance results are based on testing or projections as of July 17, 2019 and may not reflect all publicly available security updates. See configuration disclosure for details. No product or component can be absolutely secure.

For more complete information about performance and benchmark results, visit www.intel.com/benchmarks.

IO_uring: The new IO interface

- Designed with low latency devices in mind
- Efficient in terms of per I/O overhead
- High I/O performance & scalable:
  - Zero-copy: Submission Queue (SQ) and Completion Queue (CQ) place in shared memory
  - No locking: Uses single-producer-single-consumer ring buffers
- Easy to use
- Supports both block and file I/O

Introduction to Liburing library

- Provides a simplified API and easier way to establish IO_uring instance
- Initialization / De-initialization:
  - ***io_uring_queue_init():*** Sets up io_uring instance and creates a communication channel between application and kernel
  - ***io_uring_queue_exit():*** Removes the existing io_uring instance
- Submission:
  - ***io_uring_get_sqe():*** Gets a submission queue entry (SQE)
  - ***io_uring_prep_readv():*** Prepare a SQE with readv operation
    io_uring_prep_writev(): Prepare a SQE with writev operation
  - ***io_uring_submit():*** Tell the kernel that submission queue is ready for consumption

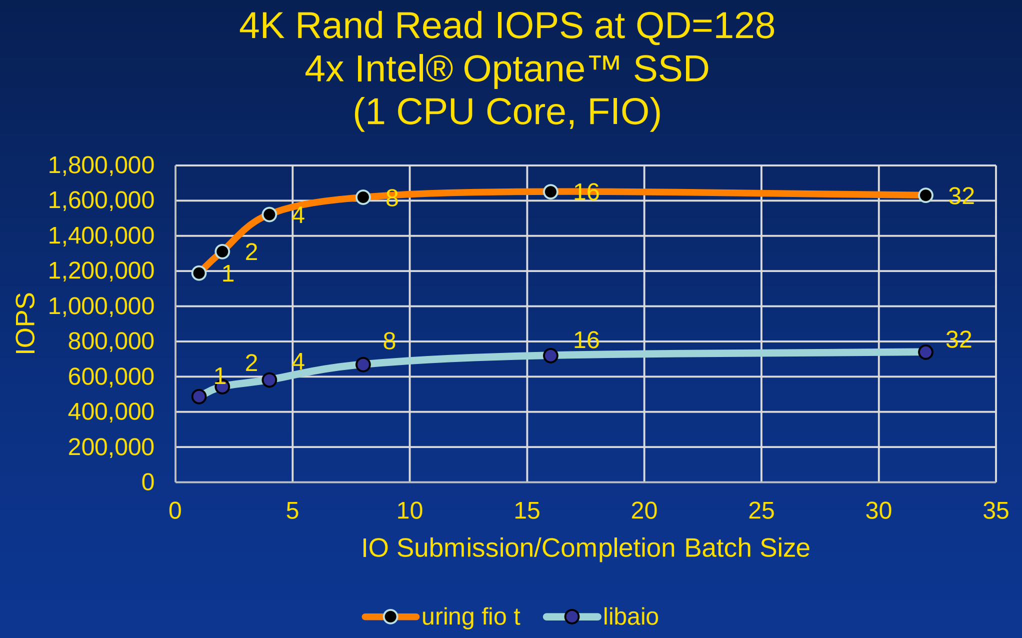
Introduction to Liburing library

- Completion:
  - ***io_uring_wait_cqe():*** Wait for completion queue entry (CQE) to complete
  - ***io_uring_peek_cqe():*** Take a peek at the completion, but do not wait for the event to complete
  - ***io_uring_cqe_seen():*** Called once completion event is finished. Increments the CQ ring head, which enables the kernel to fill in a new event at that same slot.
- More advanced features not yet available through liburing
- For further information about liburing
  - *http://git.kernel.dk/cgit/liburing*

I/O Interfaces comparisons

SW Overhead	Synchronous I/O	Libaio	IO_uring
System Calls	At least 1 per I/O	At least 2 per I/O	At least 1 per batch, zero when using SQ submission thread. Batching reduces per I/O overhead
Memory Copy	Yes	Yes – SQE/CQE	Zero-copy. Shared SQ & CQ
Context Switches	Yes	Yes	Minimal context switching
Interrupts	Interrupt driven	Interrupt driven	Supports both Interrupts and polling I/O
Blocking I/O	Synchronous	Asynchronous	Asynchronous
Buffered I/O	Yes	No	Yes

IO_uring: Single Core Max IOPS



- 4x Intel® Optane™ SSDs used to avoid I/O bottleneck
- IO Submission and completion batch sizes were increased from 1 to 32
- IOPS increases with increased submission and completion batch size from 1 to 8
- Max single core IOPS at 1.6M per core using IO_uring
- Libaio maxes out at ~600K per core

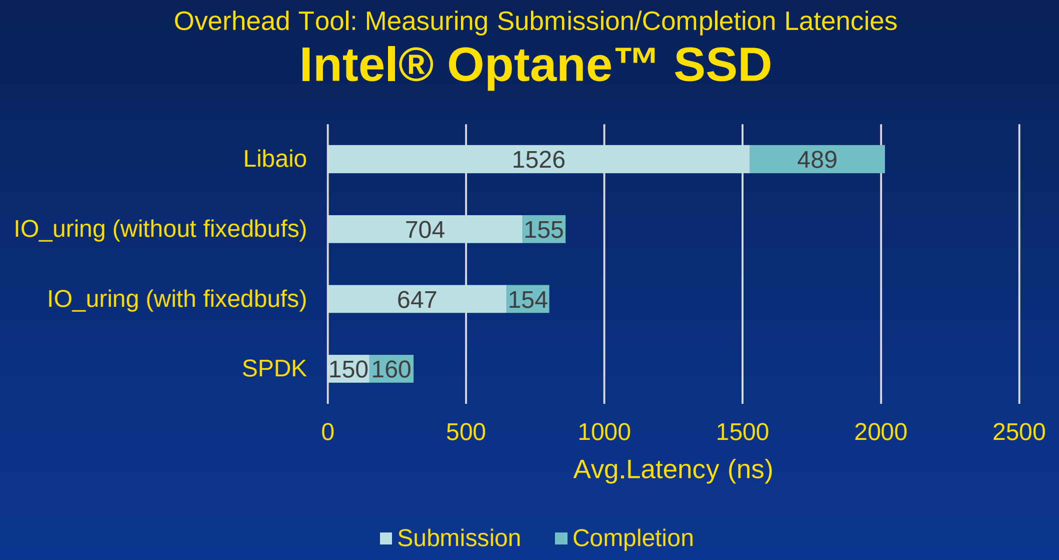
For test configuration details please see slide # 16

[1]Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance. Performance results are based on testing or projections as of July 17, 2019 and may not reflect all publicly available security updates. See configuration disclosure for details. No product or component can be absolutely secure.

For more complete information about performance and benchmark results, visit www.intel.com/benchmarks.

Flash Memory Summit

Measuring per I/O Latencies: Libaio vs. IO_uring



- Using overhead test app within SPDK. Measures software overhead of I/O submission and completion
- Runs a random read, queue depth = 1 I/O to a single device
- **Submission Latency**: Captures TSC before and after the I/O submission
- **Completion Latency**: Captures TSC before and after the I/O completion check

IO_uring (without fixedbufs) submission overhead reduces by 50% and completion overhead by 70% compared to libaio

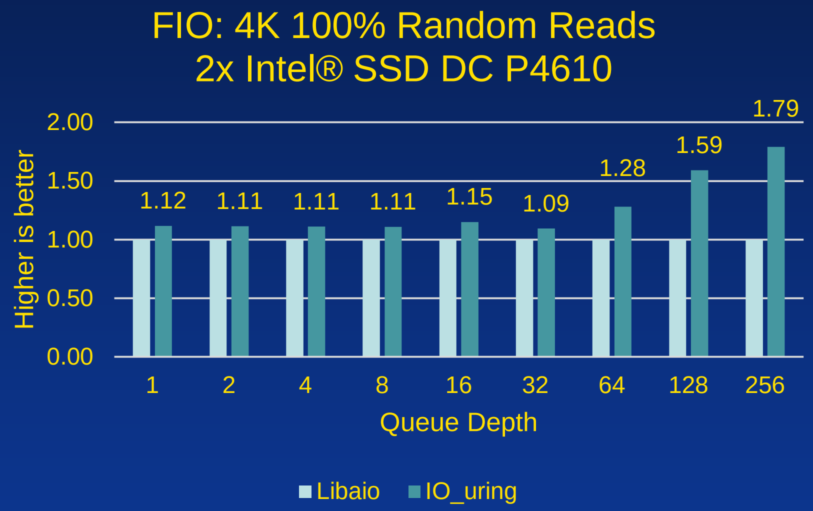
Fixedbufs skips the entire mapping of pages, which improves submission latency

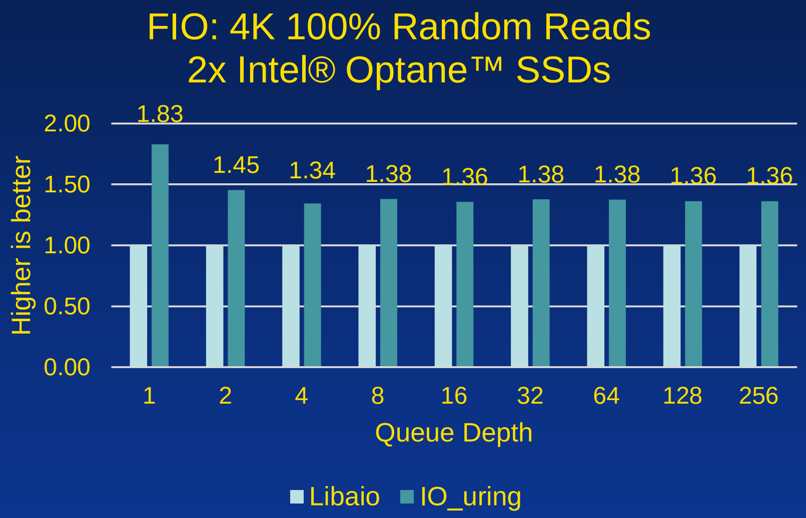
For test configuration details please see slide # 16

[1]Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance. Performance results are based on testing or projections as of July 17, 2019 and may not reflect all publicly available security updates. See configuration disclosure for details. No product or component can be absolutely secure.

For more complete information about performance and benchmark results, visit www.intel.com/benchmarks.

Relative IOPS Performance: Single Core: IO_Uring vs. Libaio





- Up to 10-15% improvement with IO_uring on Intel® SSD DC P4610 at lower queue depths

- IO_uring performs up to 1.8x better at lower queue depths on Intel® Optane™ SSDs

For test configuration details please see slide # 16

[1]Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance. Performance results are based on testing or projections as of July 17, 2019 and may not reflect all publicly available security updates. See configuration disclosure for details. No product or component can be absolutely secure.

For more complete information about performance and benchmark results, visit www.intel.com/benchmarks.

Summary

- IO_uring is the latest high performance I/O interface in the Linux Kernel (available since 5.1 release)
- Helps improve performance for low-latency media
- Eliminates limitations of current Linux kernel async I/O interfaces
- Up to 1.8x better in IOPS per core and 70% better in latency than libaio for a single thread

Flash Memory Summit

Legal Disclaimers

- Intel technologies may require enabled hardware, specific software, or services activation. Performance varies depending on system configuration. Check with your system manufacturer or retailer.
- Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors.

  Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit http://www.intel.com/performance.
- Cost reduction scenarios described are intended as examples of how a given Intel- based product, in the specified circumstances and configurations, may affect future costs and provide cost savings. Circumstances will vary. Intel does not guarantee any costs or cost reduction.
- All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.
- No computer system can be absolutely secure.
- Intel, the Intel logo, Xeon, Intel vPro, Intel Xeon Phi, Look Inside., are trademarks of Intel Corporation in the U.S. and/or other countries.
- *Other names and brands may be claimed as the property of others.
-

Backup

Performance Configuration

Flash Memory Summit

Performance configuration for slide 5 data:

Relative Latency: SuperMicro SYS-2029U-TN24R4T, Intel(R) Xeon(R) Platinum 8270 CPU @ 2.70GHz, 384GB DDR4, Ubuntu 18.04 LTS, Linux Kernel 5.2.0**, 1x** Intel® Optane™ 375GB SSD, fio-3.14-6-g97134, 4K 100% Random Reads, Iodepth=1, ramp time = 30s, direct=1 , runtime=300s, Data collected at Intel Storage Lab 07/17/2019

Throughput: SuperMicro SYS-2029U-TN24R4T, Intel(R) Xeon(R) Platinum 8270 CPU @ 2.70GHz, 384GB DDR4, Ubuntu 18.04 LTS, Linux Kernel 5.2.0**, 1x** Intel® SSD DC P4610 1.6TB, fio-3.14-6-g97134, 4K 100% Random Reads, Iodepth=1 to 256 varied (exponential 2), ramp time= 30s, direct=1, runtime=300s, Data collected at Intel Storage Lab 07/17/2019

Performance configuration for slide 10 data: SuperMicro SYS-2029U-TN24R4T, Intel(R) Xeon(R) Platinum 8270 CPU @ 2.70GHz, 384GB DDR4, Ubuntu 18.04 LTS, Linux Kernel 5.2.0**, 4x** Intel® Optane™ 375GB SSD, fio-3.14-6-g97134, t/fio app used with varied batching sizes, Data collected at Intel Storage Lab 07/17/2019

Performance configuration for slide 11 data: SuperMicro SYS-2029U-TN24R4T, Intel(R) Xeon(R) Platinum 8270 CPU @ 2.70GHz, 384GB DDR4, Ubuntu 18.04 LTS, Linux Kernel 5.2.0**, 1x** Intel® Optane™ 375GB SSD, SPDK overhead tool used, runtime = 300s, Data collected at Intel Storage Lab 07/17/2019

Performance configuration for slide 12 data: SuperMicro SYS-2029U-TN24R4T, Intel(R) Xeon(R) Platinum 8270 CPU @ 2.70GHz, 384GB DDR4, Ubuntu 18.04 LTS, Linux Kernel 5.2.0**, 2x** Intel® Optane® 375GB SSD, 2x Intel® SSD DC P4610 fio-3.14-6-g97134, runtime = 300s, Data collected at Intel Storage Lab 07/17/2019

Kernel Block layer Tuning Script

```
DEVS="nvme0n1 "

for dev in $DEVS; do
	echo "Prep /dev/$dev"
	SYSFS=/sys/block/$dev/queue

	echo 0 > $SYSFS/iostats
	echo 0 > $SYSFS/rq_affinity
	echo 2 > $SYSFS/nomerges
	echo 0 > $SYSFS/io_poll_delay
done
```