OPENFABRICS ALLIANCE

2020 OFA Virtual Workshop

SPDK BASED USER SPACE NVME OVER TCP TRANSPORT SOLUTION

Ziye Yang, Cloud Software Engineer

Intel

NOTICES AND DISCLAIMERS

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration.

No computer system can be absolutely secure.

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. For more complete information about performance and benchmark results, visit http://www.intel.com/benchmarks .

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/benchmarks .

Benchmark results were obtained prior to implementation of recent software patches and firmware updates intended to address exploits referred to as "Spectre" and "Meltdown." Implementation of these updates may make these results inapplicable to your device or system.

Intel® Advanced Vector Extensions (Intel® AVX)* provides higher throughput to certain processor operations. Due to varying processor power characteristics, utilizing AVX instructions may cause a) some parts to operate at less than the rated frequency and b) some parts with Intel® Turbo Boost Technology 2.0 to not achieve any or maximum turbo frequencies. Performance varies depending on hardware, software, and system configuration and you can learn more at http://www.intel.com/go/turbo.

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

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.

Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

OUTLINE

- SPDK NVMe-oF development history & status
- SPDK NVMe-oF TCP transport design detail
- Smartly use Kernel TCP/IP stack with SPDK sock library
- Experimental results (selected)
- Ongoing work and development plan
- Conclusion

SPDK NVME-OF DEVELOPMENT HISTORY & STATUS

SPDK NVME-OF TARGET TIMELINE

20.07+: Muser transport support

19.04 – 20.04 : Continuing Improvement

19.01: TCP transport released

17.11 – 18.11: Rdma TRANSPORT IMPROVEMENTS

17.03 – 17.07 : Functional hardening

July 2016: Released with RDMA transport support

SPDK NVME-OF HOST TIMELINE

20.07+: Muser transport support

19.04 - 20.04: Continuing Improvement

19.01: TCP Transport released

17.11 – 18.11: Rdma TRANSPORT IMPROVEMENTs

17.03 – 17.07 : Functional hardening (e.g., interoperability test w target)

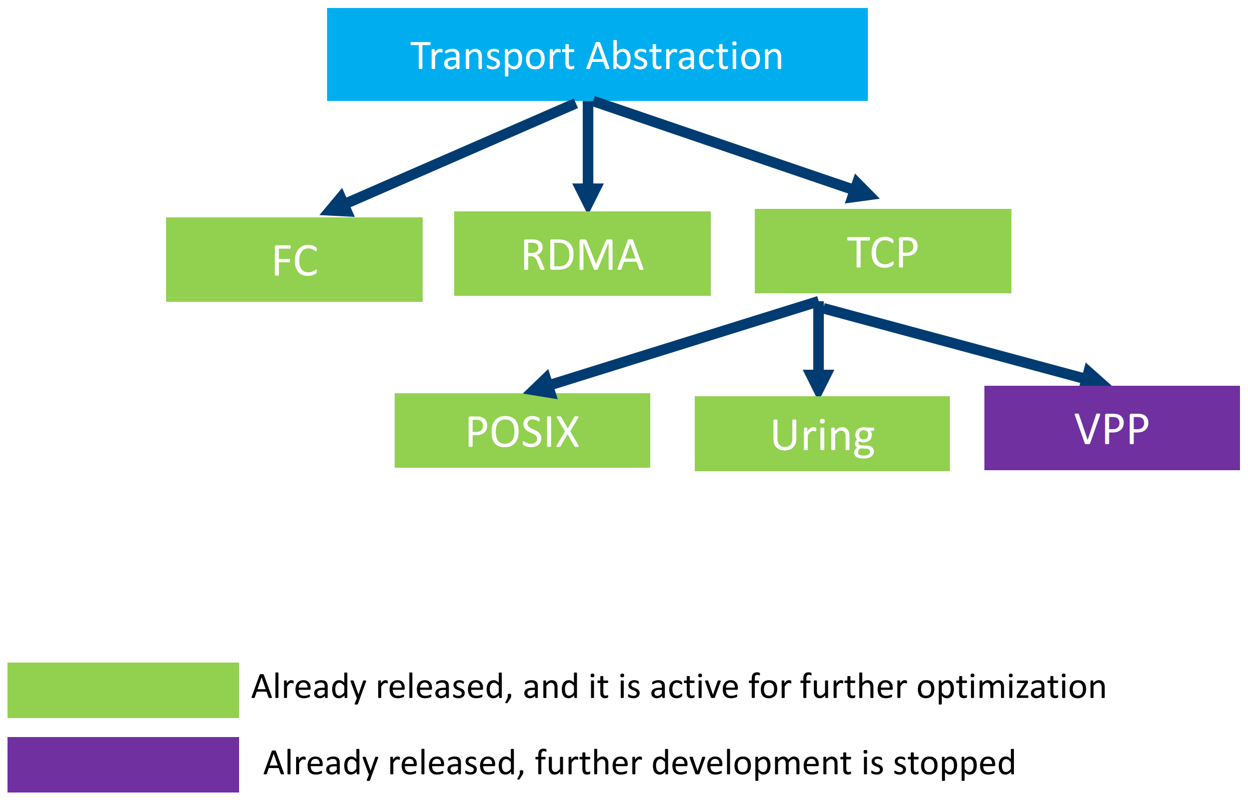
Dec 2016: released with rdma transport support

SPDK NVME-OF TARGET DESIGN HIGHLIGHTS

NVMe* over Fabrics Target Features	Performance Benefit
Utilizes user space NVM Express* (NVMe) Polled Mode Driver	Reduced overhead per NVMe I/O
Group polling on each SPDK thread (binding on CPU core) for multiple transports	Better scaling to many connections
Connections pinned to dedicated SPDK thread	No synchronization overhead
Asynchronous NVMe CMD handling in whole life cycle	No locks in NVMe CMD data handling path

SPDK NVME-OF TCP TRANSPORT

GENERAL DESIGN AND IMPLEMENTATION



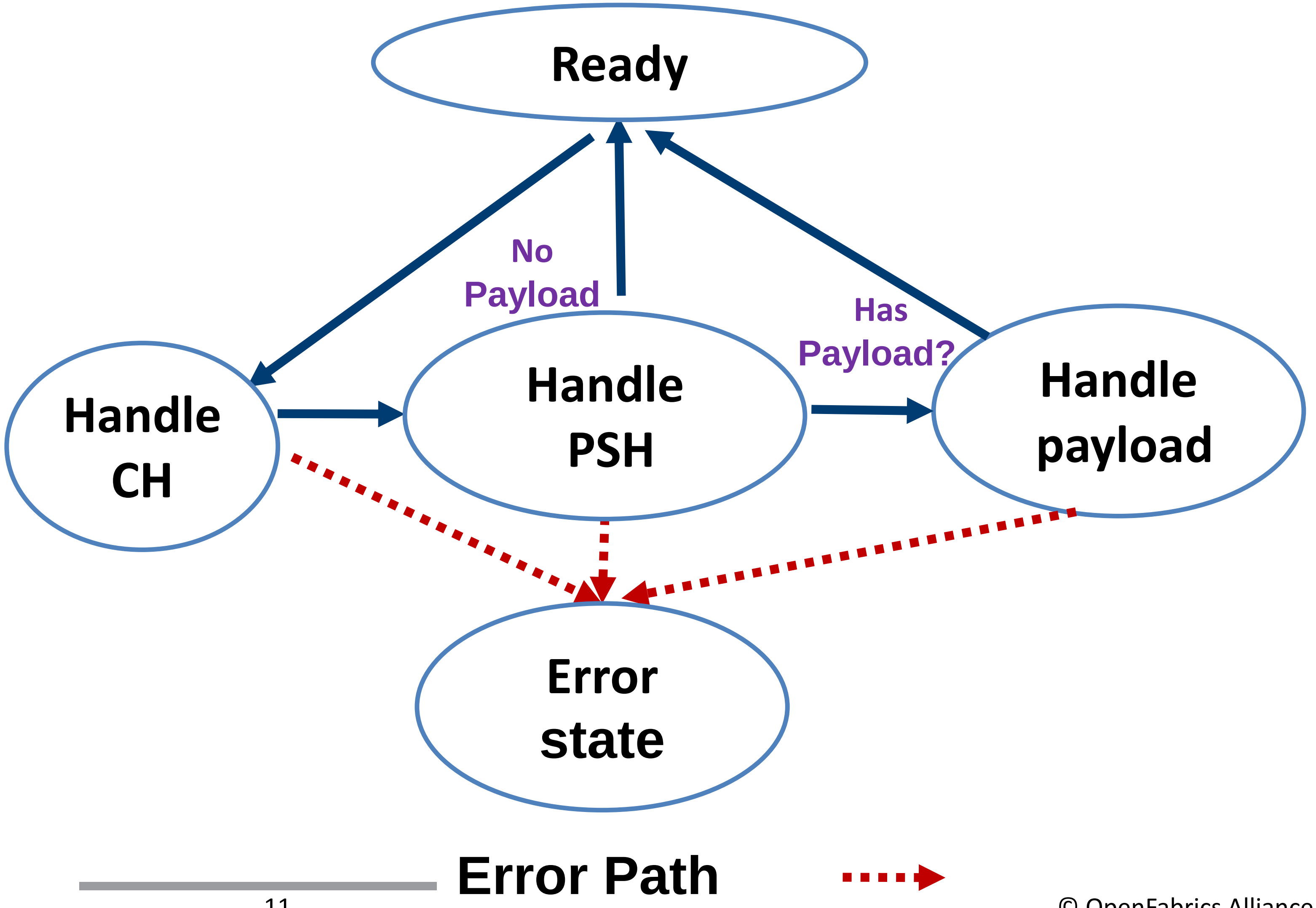
- **SPDK NVMe-oF TCP transport main code location.**
  - Host side code: lib/nvme/nvme_tcp.c
  - Target side code: lib/nvmf/tcp.c

PERFORMANCE DESIGN CONSIDERATION FOR TCP TRANSPORT IN TARGET SIDE

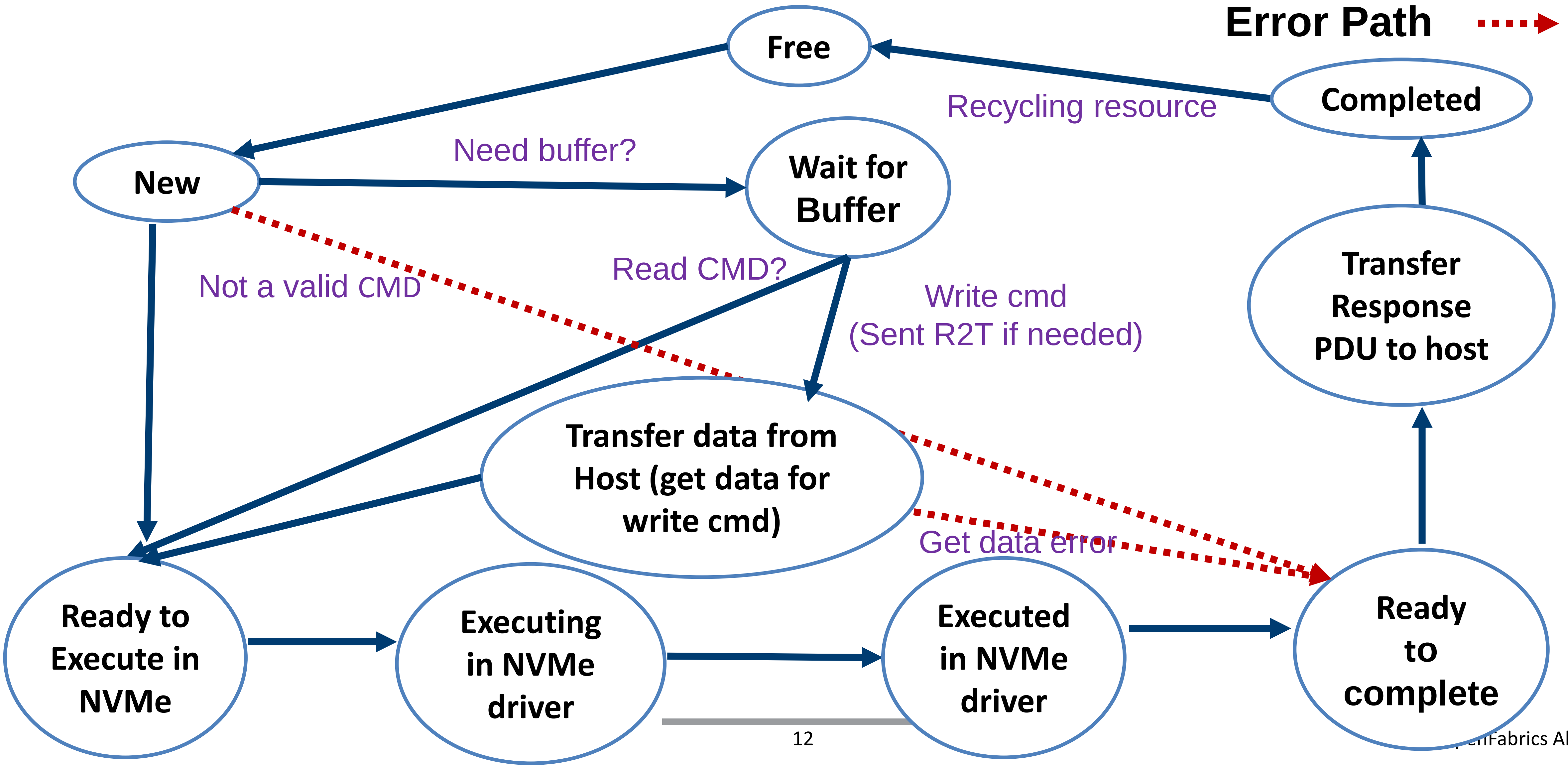
Ingredients	Methodology
Design framework	Follow the general SPDK NVMe-oF framework (e.g., polling group)
TCP connection optimization	Use the SPDK encapsulated Socket API (preparing for integrating other stack, e.g., VPP)
NVMe/TCP PDU handling	Use state machine to track
NVMe/TCP request life time cycle	Use state machine to track (Purpose: Easy to debug and good for further performance improvement)

NVME TCP PDU RECEIVING HANDLING FOR EACH CONNECTION

```
enum nvme_tcp_pdu_recv_state {
        /* Ready to wait PDU */
        NVME_TCP_PDU_RECV_STATE_AWAIT_PDU_READY,

        /* Active tqpair waiting for any PDU common header */
        NVME_TCP_PDU_RECV_STATE_AWAIT_PDU_CH,

        /* Active tqpair waiting for any PDU specific header */
        NVME_TCP_PDU_RECV_STATE_AWAIT_PDU_PSH,

        /* Active tqpair waiting for payload */
        NVME_TCP_PDU_RECV_STATE_AWAIT_PDU_PAYLOAD,

        /* Active tqpair does not wait for payload */
        NVME_TCP_PDU_RECV_STATE_ERROR,
};
```

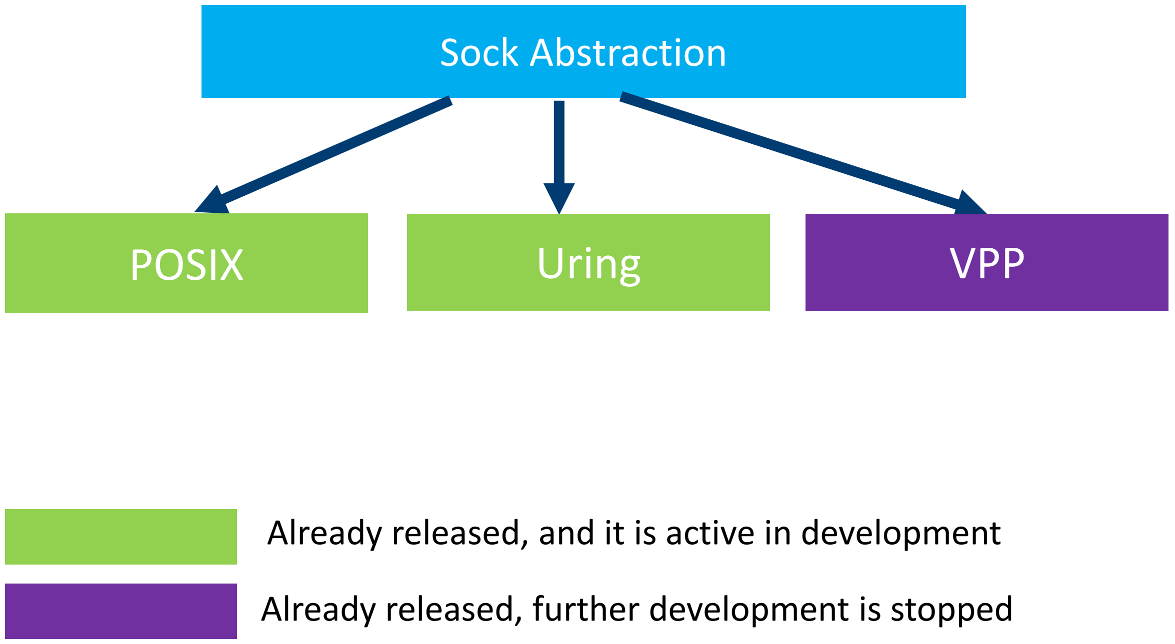


STATE MACHINE FOR NVME I/O FOLLOW IN SINGLE CONNECTION ON TARGET SIDE



SMARTLY USE KERNEL TCP/IP STACK WITH SPDK SOCK LIBRARY

THE SOCK IMPLEMENTATIONS IN SPDK

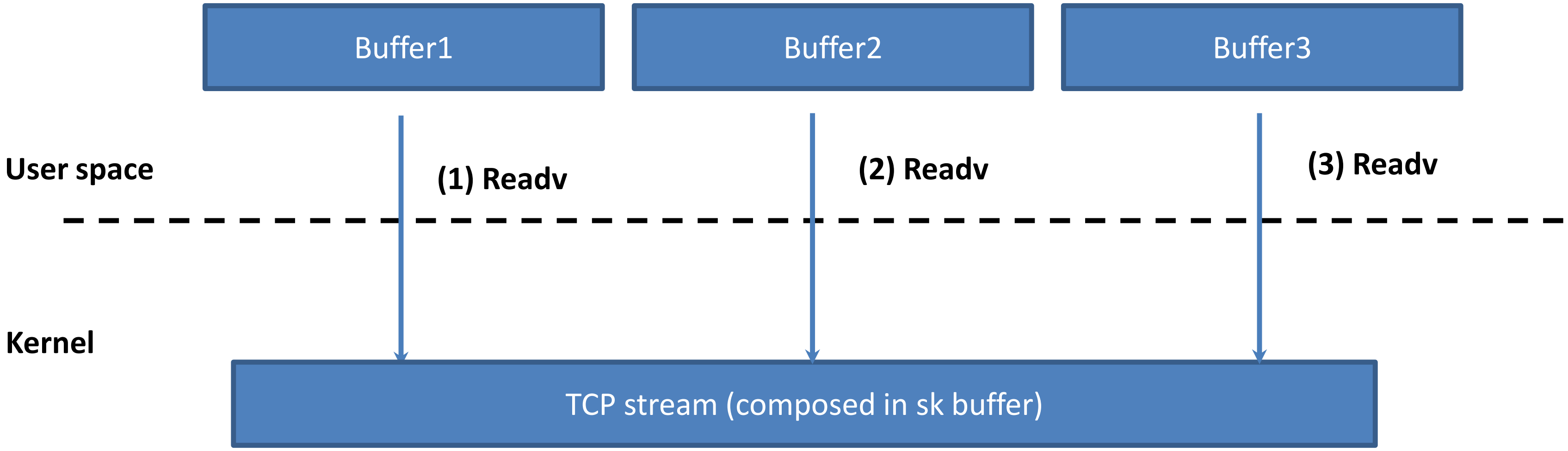


- Current Recommendation:
  - POSIX (Stable, no dependency on kernel)
  - Uring (Request Linux kernel > 5.4.3), currently it is experimental .
  - VPP : We may investigate it more if VPP supports library integration mode but not only the standalone process mode.

COMMON KNOWLEDGE TO SMARTLY USE KERNEL TCP/IP STACK FOR IMPROVING NVME-OF TCP

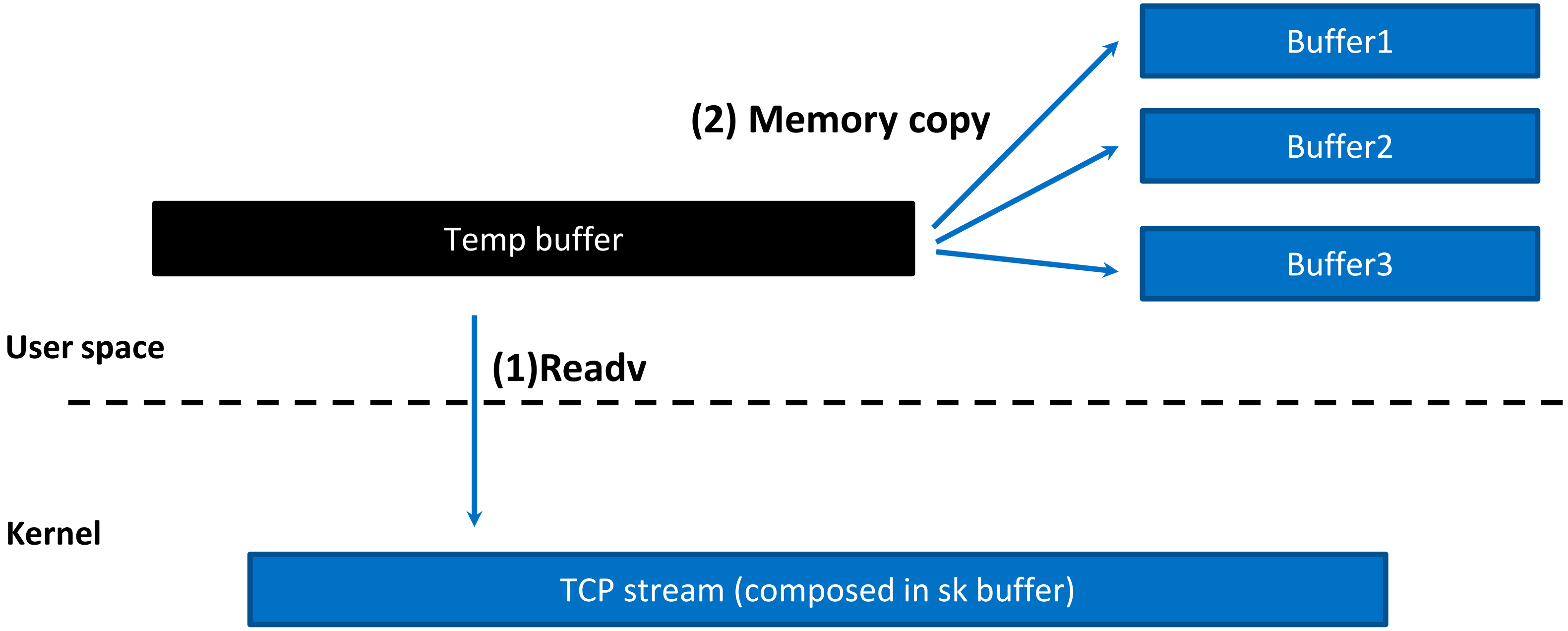
- **Nonblock mode:** O_NONBLOCK setting on FD
- **Group based strategy on Pollin, Read(v), write(v) for many TCP connections.**
  - Benefit: Reduce the system call overhead
  - For example, (1) Group Pollin reduce number of readv calls; (2)Group based writev operations via uring sock on 16 TCP connections can reduce 15 system calls in one round.
- **Dedicated CPU core handling:** Each connections on file descriptor should be handle by dedicated thread or CPU core.
- **Buffered Read on each socket:** Reduce system call overhead
- **Merged write on each socket:** To reduce system call and improve throughput

TCP READ OPERATION: ORIGINAL USAGE



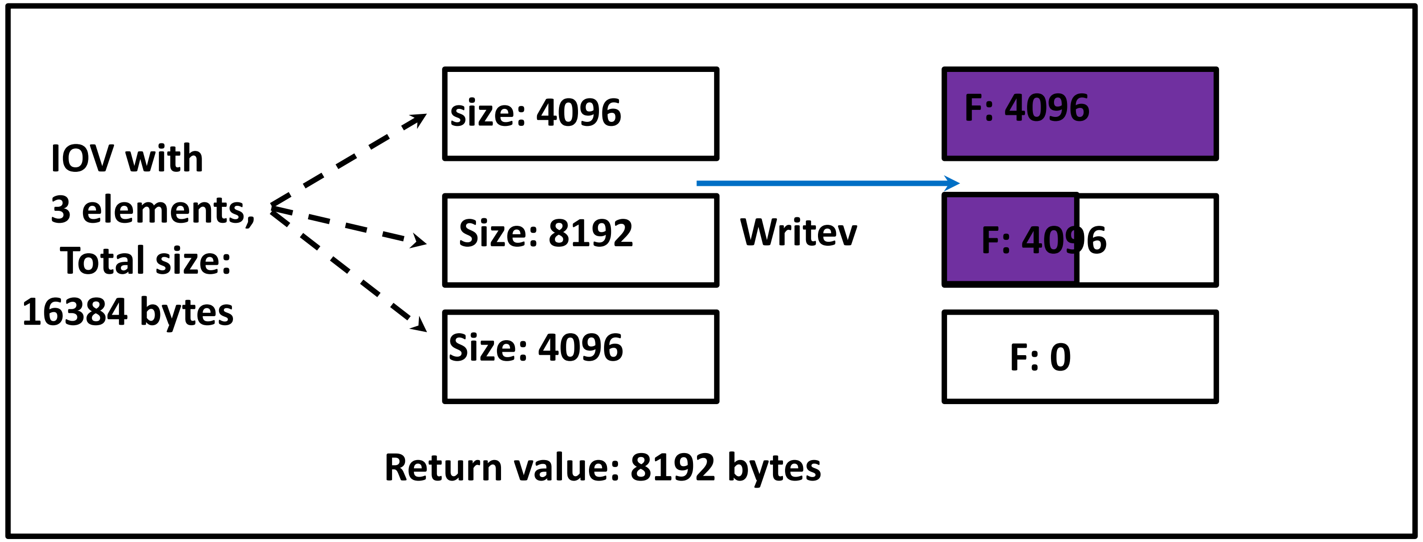
- The address of Buffer1, Buffer2, and Buffer3 may be not known in the beginning, the application cannot issue submit one readv system call to construct one IOV vector array, thus 3 system calls (readv) are needed.

BUFFERED READ SUPPORTED IN SPDK



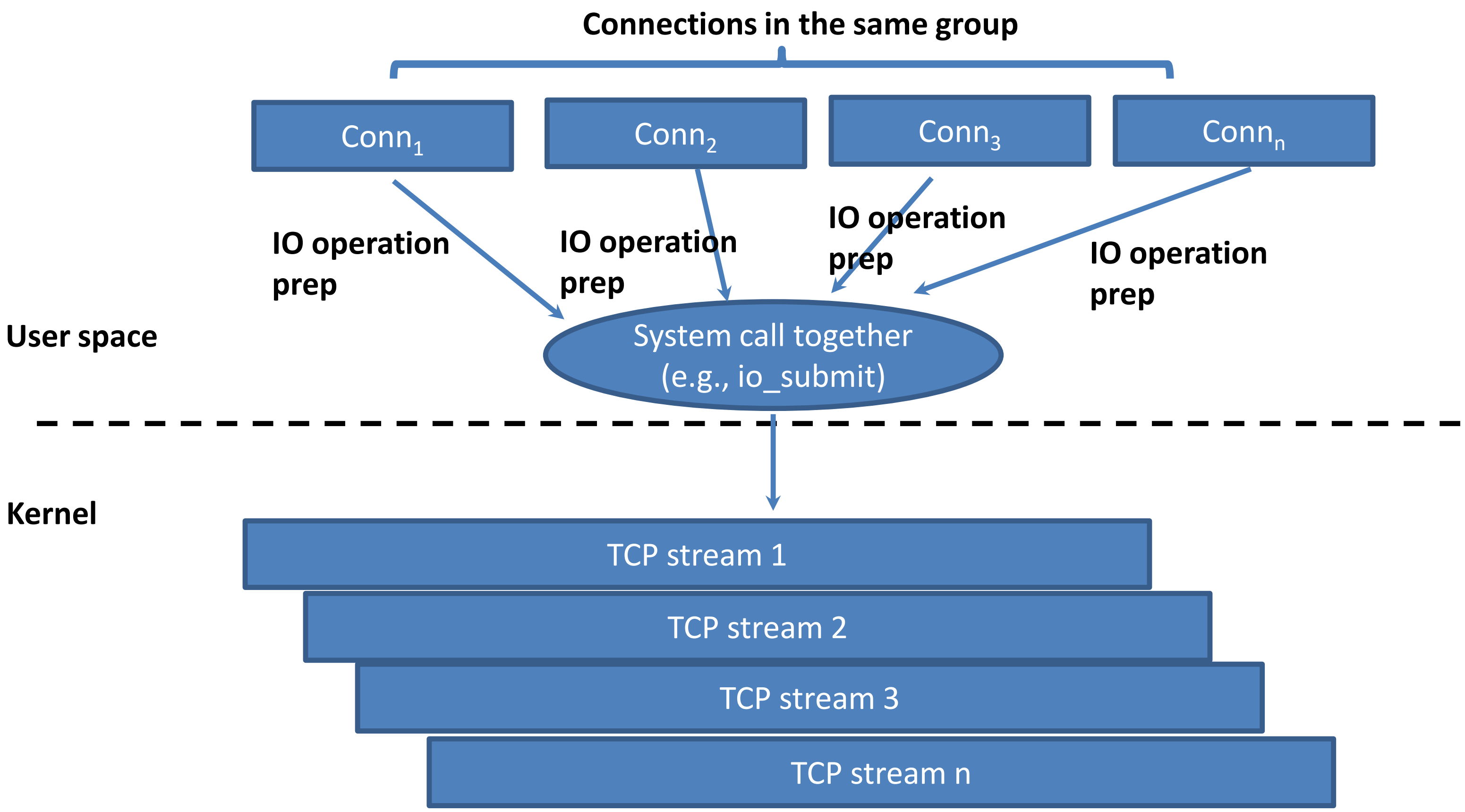
- In this cased, Buffer1, 2, 3 can be determined by application's own logic. And this solution tries to reduce the system call overhead, but introduces the memory copy overhead, so use IOAT or driver to drive CBDMA to mitigate the copy overhead by CPU.
- SPDK has **util** library(located in lib/util in spdk source folder) which supports the read buffer with pipe usage manner.

MERGED WRITE SUPPORT IN SPDK



- SPDK posix/uring libraries can merge the write I/O from app into big vectors in order reduce system calls.
- But with Merged write, we still need to handle partial write if we use NONBLOCK I/O.

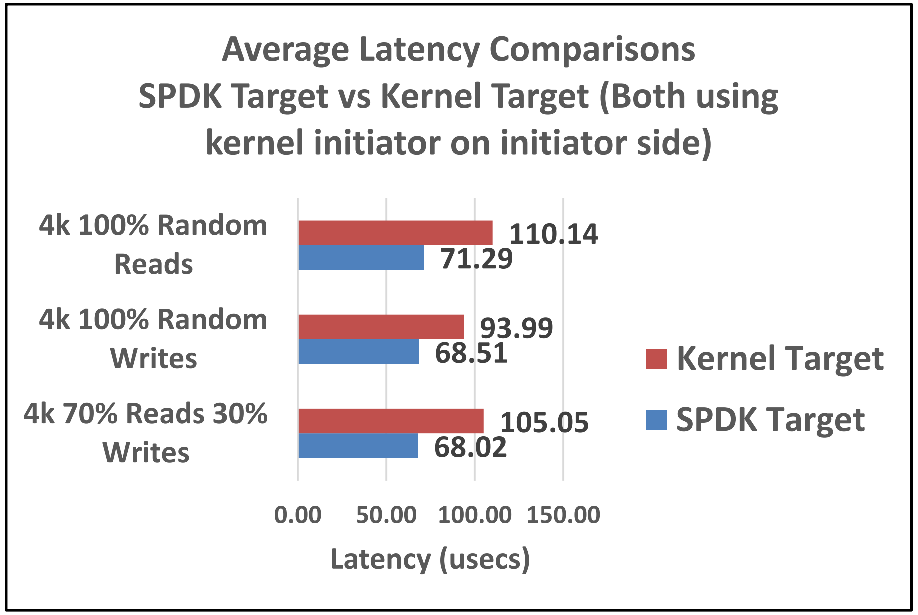
GROUP BASED ASYNCHRONOUS I/O OPERATION WITH URING

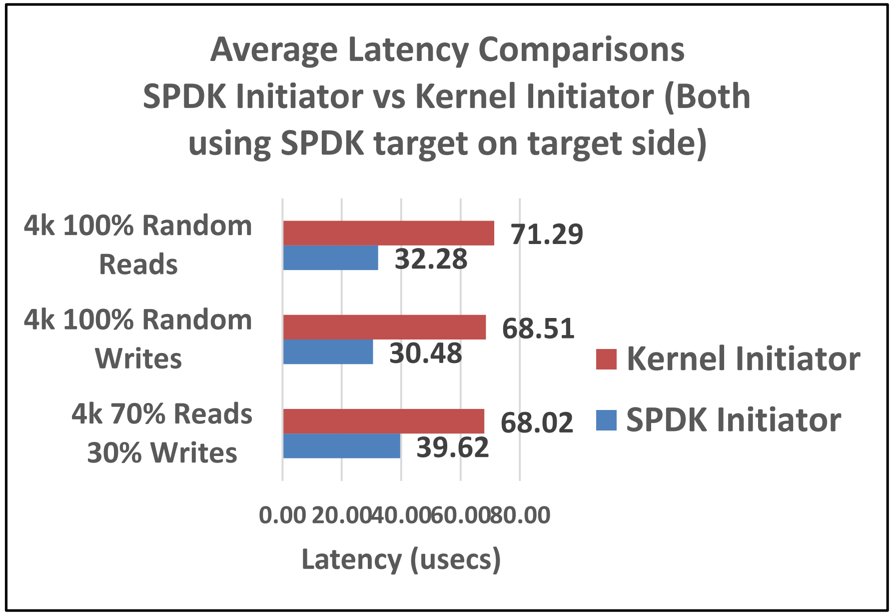


- Supported in SPDK uring sock library (Located in module/sock/uring in spdk folder)

EXPERIMENTAL RESULTS (SELECTED)

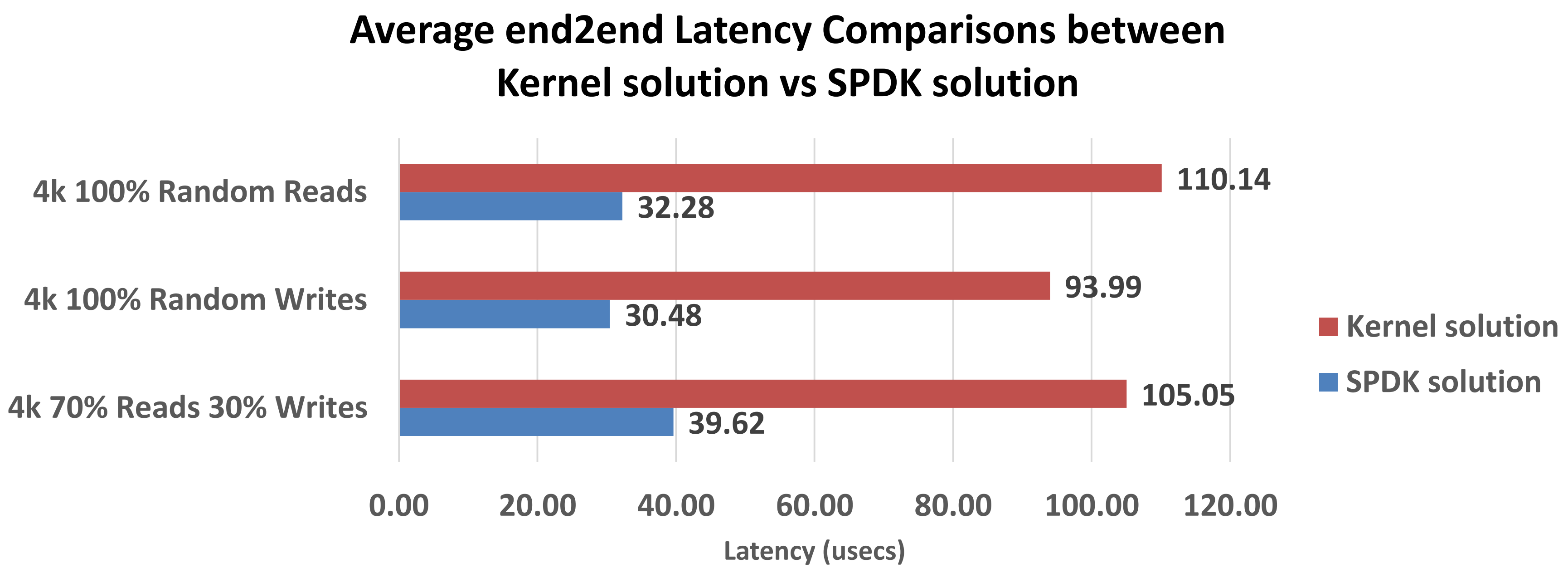
LATENCY COMPARISON BETWEEN SPDK AND KERNEL (NULL BDEV IS USED)





Experimental Configuration is located from Page5 to Page 7 in: https://ci.spdk.io/download/performance-reports/SPDK_tcp_perf_report_2001.pdf

LATENCY COMPARISON BETWEEN SPDK AND KERNEL (NULL BDEV IS USED)



Experimental configuration is located from Page5 to Page 7 in https://ci.spdk.io/download/performance-reports/SPDK_tcp_perf_report_2001.pdf

IOPS/CORE COMPARISON BETWEEN SPDK AND KERNEL ON TARGET SIDE

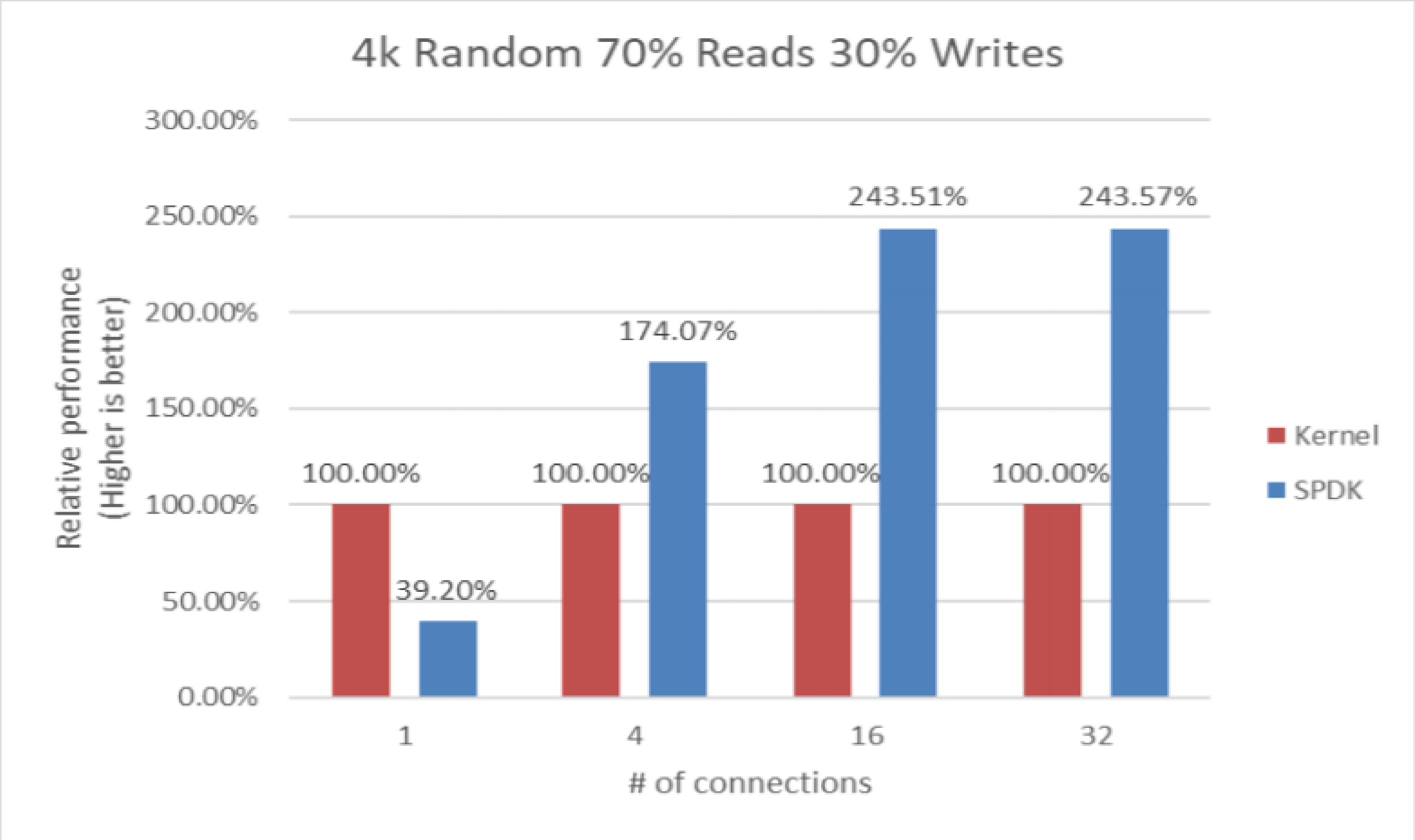


Figure 16: Relative Performance Comparison of Linux Kernel vs. SPDK NVMe-oF Target for 4KB Random 70% Reads 30% Writes

Diagram is located in Page 43 in https://ci.spdk.io/download/performance-reports/SPDK_tcp_perf_report_2001.pdf

ONGOING WORK AND DEVELOPMENT PLAN

NEW DEVELOPMENT WITH INTEL® ETHERNET 800 SERIES

- **Leverage Application Device Queues (ADQ) technology with the Intel® Ethernet 800 Series Network Adapter. Benefit: High IOPS with improved tail latency.**
  - ADQ is an application specific queuing and steering technology that dedicates and isolates application specific hardware NIC queues.
  - These queues are then connected optimally to application specific threads of execution.
- **Technique requirement:**
  - Kernel & driver: Busy polling; Socket option for NAPI_ID (SO_INCOMING_NAPI_ID); symmetric polling; ….
  - Application: Handle the socket with same NAPI_ID by dedicated thread/CPU.
- **Hardware:**
  - Application level filtering & traffic shaping; Flow based queue steering and load balance.

FURTHER DEVELOPMENT PLAN OF SPDK NVME-OF TCP TRANSPORT

- **Continue enhancing the functionality**
  - Including the compatible test with Linux kernel solution.
- **Performance tuning in software**
  - Work on kernel TCP/IP stack
    - Smartly using kernel TCP/IP stack through io uring will be our direction, i.e., continue improving the performance via uring based sock implementation in SPDK.
    - Code in SPDK repo: module/sock/uring
  - Work on user space TCP/IP stack
    - May continue investigate space stack: Seastar + DPDK. It depends on our optimization result with kernel TCP/IP stack.
- **Performance enhancement via hardware features**
  - **Networking hardware**: Continue using features from NICs for performance improvement, e.g., 100Gb Intel® Ethernet 800 Series Network Adapter with ADQ.
  - **Other hardware**: e.g., Figuring out TCP/IP offloading methods on FPGA and SmartNICs.

CONCLUSION

CONCLUSION

- **SPDK NVMe-oF solution is well adopted by the industry. In this presentation, followings are introduced, i.e.,**
  - The development status of SPDK NVMe-oF solution
  - SPDK TCP transport development status and optimization direction, e.g., How to use kernel TCP/IP stack to optimize the NVMe-oF TCP.
  - Some performance sharing with SPDK 20.01 release.
- **Further development**
  - Continue following the NVMe-oF spec and adding more features.
  - Continue performance enhancements and integration with other solutions.
- **Call for activity in community**
  - Welcome to bug submission, idea discussion and patch submission for NVMe-oF

2020 OFA Virtual Workshop

THANK YOU

Ziye Yang, Cloud Software Engineer

Intel