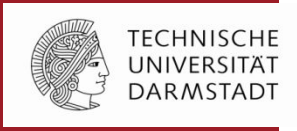
Exploiting SSD-Arrays with Asynchronous I/O

Dwarakanandan B M

(Tobias Ziegler)



Original Topic - Userspace Network Stacks in Fast Networks

TECHNISCHE UNIVERSITÄT DARMSTADT

Motivation:

- Economic viability of high speed networks with bandwidth reaching 100GbE
- Due to the overhead of Traditional TCP/IP stacks, explore alternatives such as DPDK, mTCP, eRPC

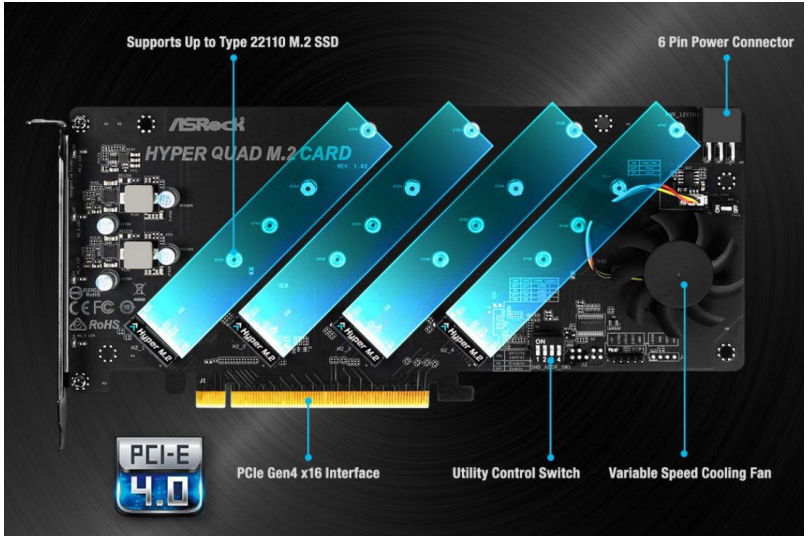
DPDK with Infiniband:

- Mellanox ConnectX-5 network adapters configured to use DPDK 20.11 MLX5 Poll mode drivers
- Current MLX5 PMD for Infiniband only support Ethernet as their link layer

Switch interconnecting these nodes only supported Infiniband. Issue has since been resolved by inter-connecting the adapters using an Ethernet switch.

Why SSD Arrays ?

- As data sizes grow, storing data completely in-memory becomes infeasible
- Flash storage has seen tremendous growth in capacity, throughput, latency, etc.
- High speed interfaces like NVMe M.2
- PCIe Gen 3.0 x16 lanes Max throughput of 15.754 GB/s*



ASRock HYPER QUAD M.2 CARD PCIe 4.0 x16

Four Samsung 980 PRO 1TB M.2

*Source: PCI Express specification: https://en.wikipedia.org/wiki/PCI_Express

Asynchronous IO to the Rescue !!

TECHNISCHE UNIVERSITÄT DARMSTADT

- High speed interconnects such as PCIe expose the large scale parallelism inherent to flash storage
- Traditional Operating System I/O stack becomes a performance bottleneck

Research question

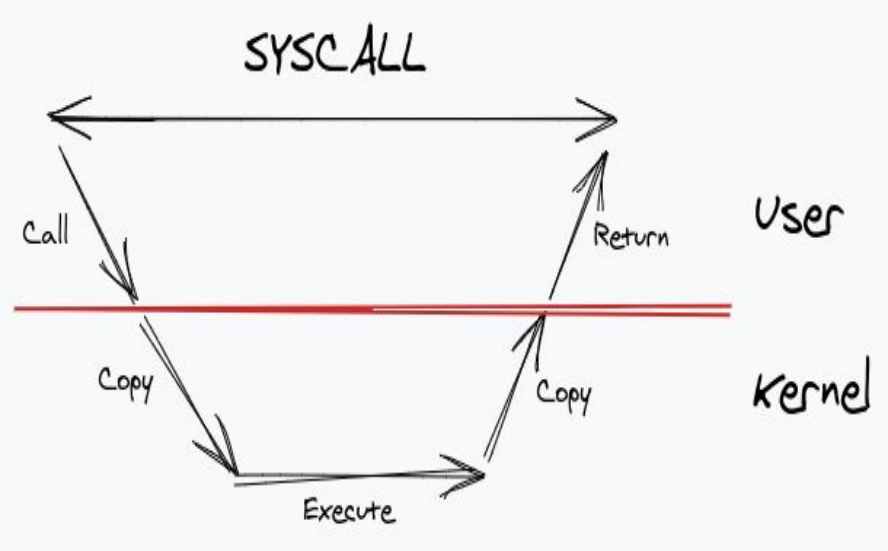
Evaluate the feasibility of Asynchronous I/O stacks to exploit SSD-Array parallelism by comparing

- **syncio** - Traditional Synchronous I/O
- **libaio** - Async I/O using the Linux AIO Interface
- **io_uring** - Async I/O using io_uring interface

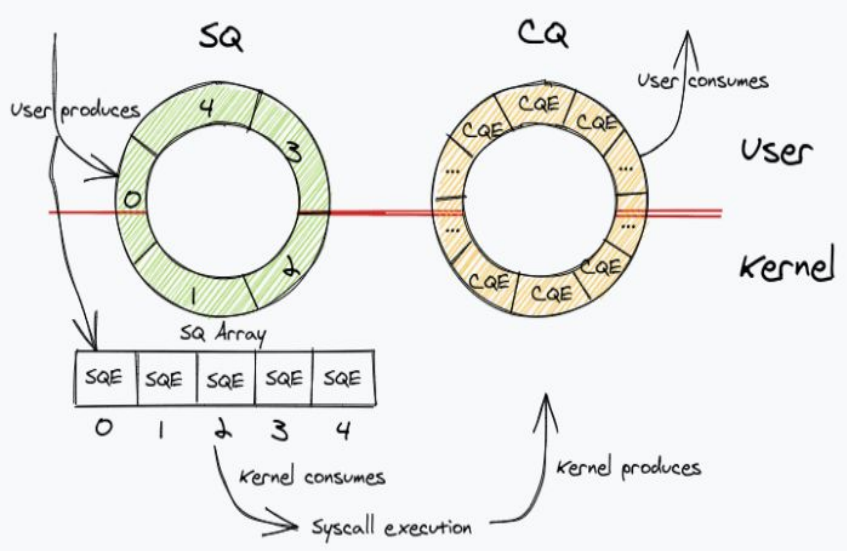
I / O ?

TECHNISCHE UNIVERSITÄT DARMSTADT

Traditional I/O

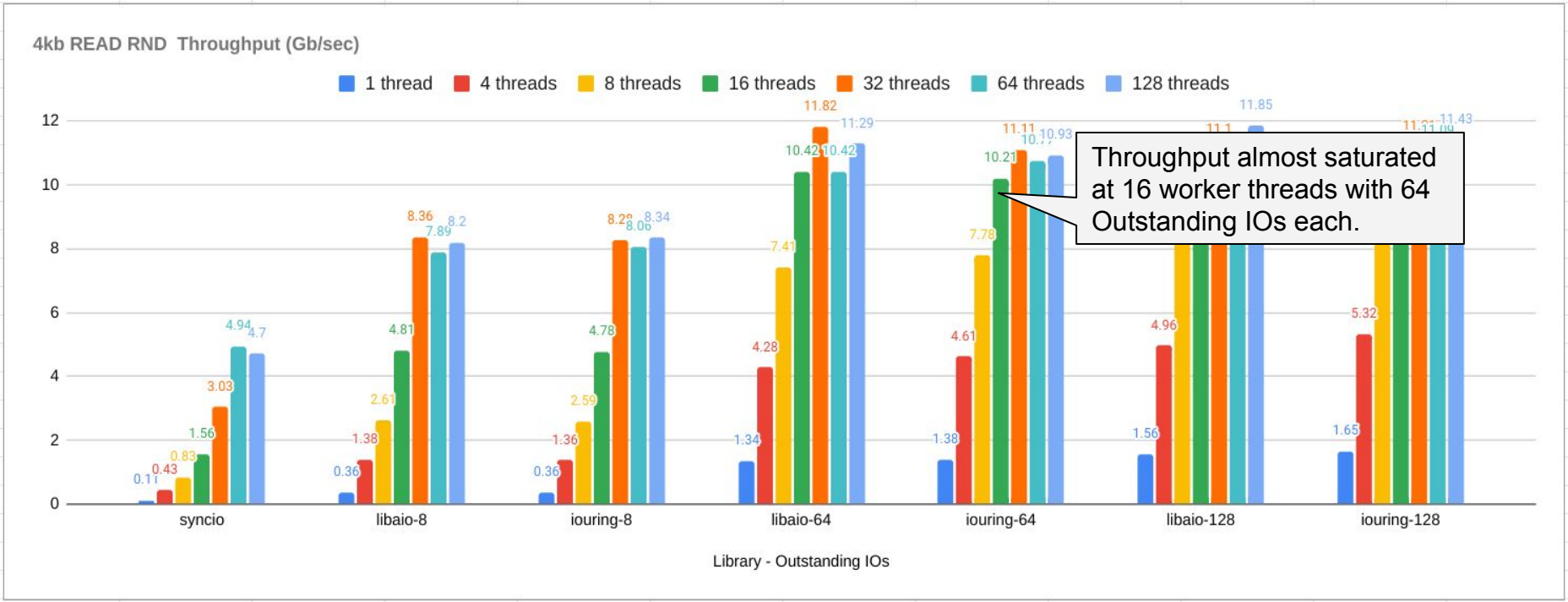


IO_URING I/O



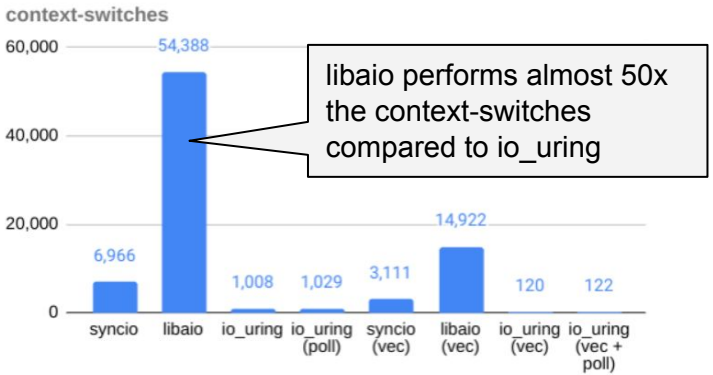
Source: Agniva De Sarker. (2020). Getting hands-on with io_uring. https://mattermost.com/blog/iouring-and-go/

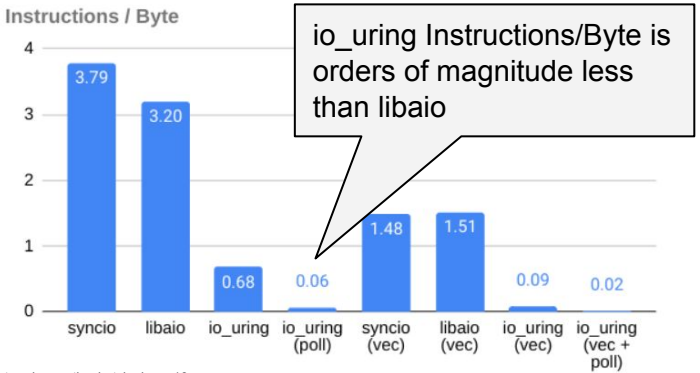
Throughput comparison



So why io_uring ?

- Supports buffered I/O whereas libaio falls back to synchronous mode without O_DIRECT.
- Lower CPU overhead due to less context switches . Big deal due to Spectre/Meltdown mitigations (Kernel page table isolation)
- Can optionally work in a polled manner as opposed to using syscalls. Overall lesser overhead.





Perf stats for 4kb RANDOM READ. Async Outstanding IO = 128. Vectored operation batch size = 10

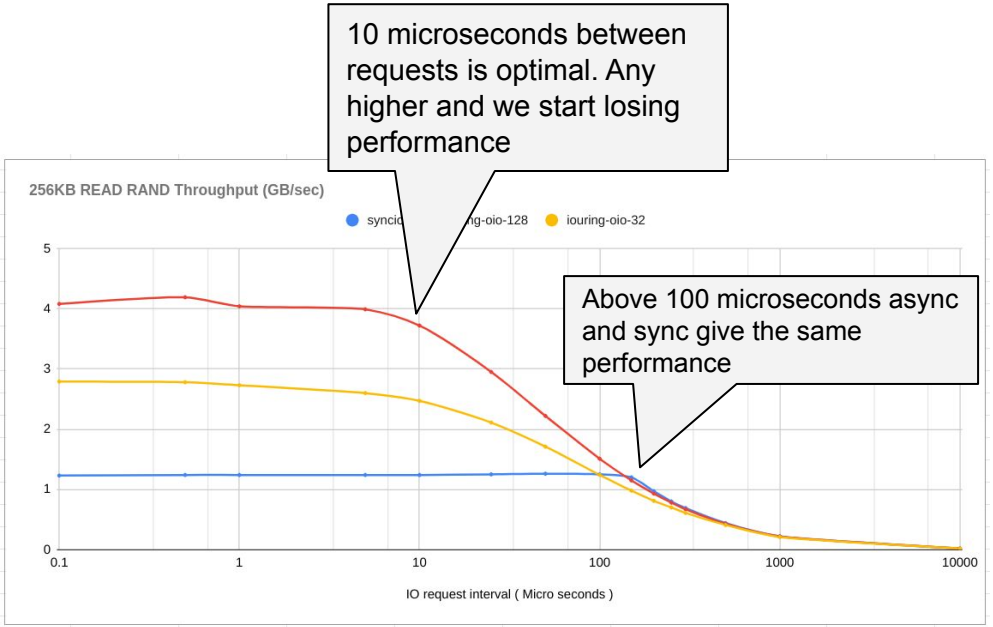
Optimal Message rate

TECHNISCHE UNIVERSITÄT DARMSTADT

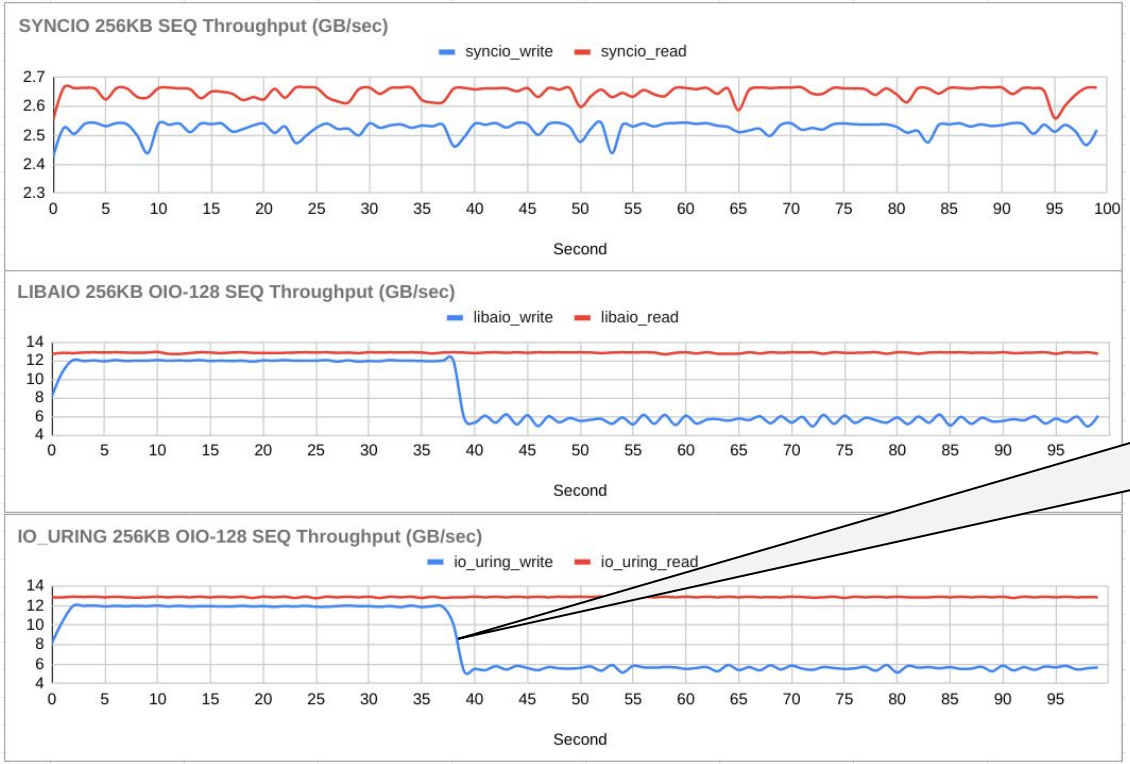
```
Thread-1 ( _io_request_producer ):=
while ()
{
    produce_one_IO_request()
    sleep(IO_REQUEST_INTERVAL)
}
```

```
Thread-2 ( _io_request_handler ):=
queued_operations:= 0
while ()
{
    consume_one_IO_request()
    queue_IO_operation_associated_with_request()
    queued_operations++

    if (queued_operations == OUTSTANDING_IO_LIMIT)
    {
        submit_all_and_wait_for_completion()
        queued_operations:= 0
    }
}
```



Effects of SLC Cache



After about 40 seconds of continuous operation we notice a sharp decline in write performance.

In Summary: Asynchronous I/O, especially implementations such as IO_URING are quintessential in exploiting the inherent parallelism of SSD-Arrays

Questions?

Benchmarking tool: https://github.com/dwarakanandan/dmlab