DeLiBA: An Open-Source Hardware/Software Framework for the Development of Linux Block I/O Accelerators

Babar Khan, Carsten Heinz, Andreas Koch

Embedded Systems and Applications Group, TU Darmstadt, Germany

TECHNISCHE UNIVERSITÄT DARMSTADT

Embedded Systems & Applications

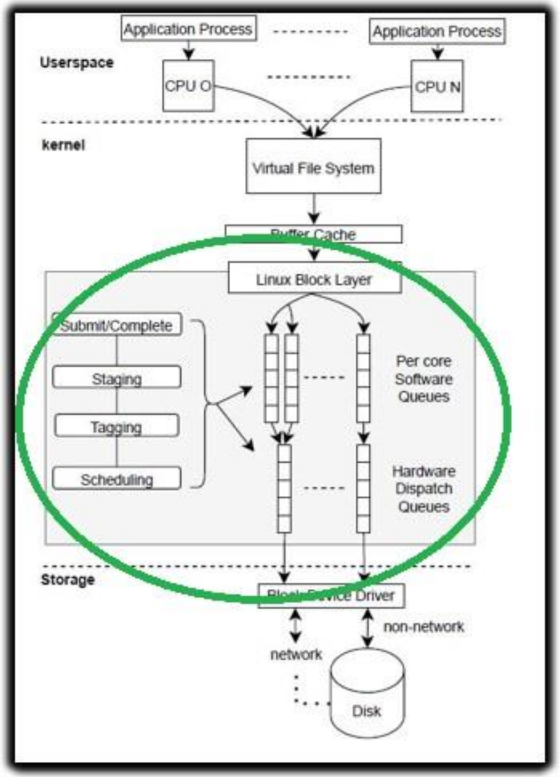
GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

32nd International Conference on Field Programmable Logic and Application (FPL) 2022
hosted by Queens University Belfast, United Kingdom
29th August - 2nd September 2022

What is Multi-Queue (MQ) Linux Block I/O Layer

- Part of **Linux** operating system.
- Responsible for handling block devices like **Hard Disks**, **SSDs**.
- Interface between **Applications** and **Storage**
- **Multi-Queue (MQ)** = For **Multi-Core Systems**

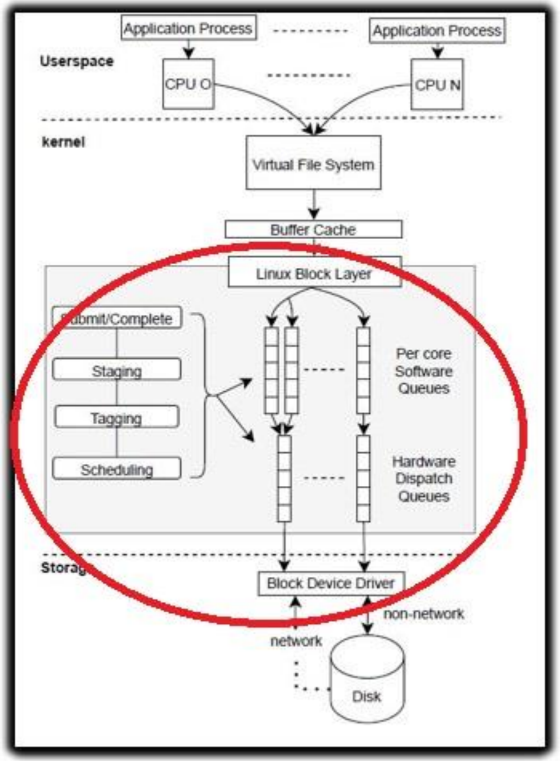


Bottleneck & Complexity in Block I/O Layer

- **18K** - **20K** instructions in single **4KB** request.

- Approx. **60%** and **90%** of total execution time in kernel on x86 and on ARM resp for **4KB** request.

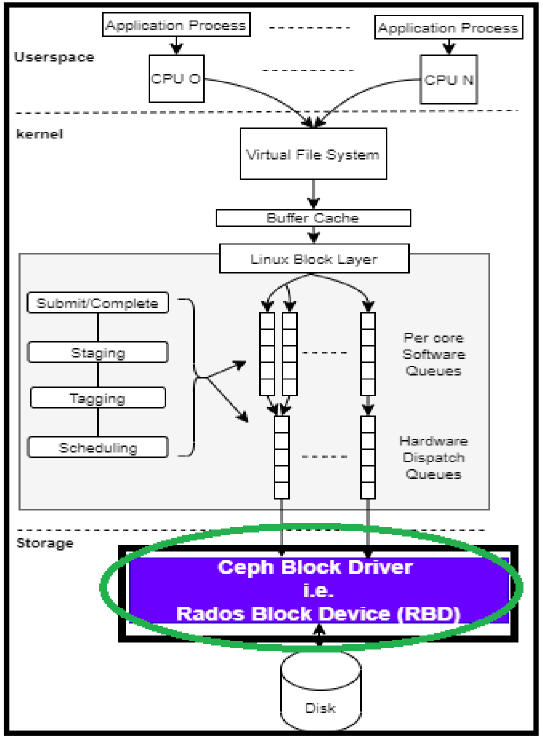
- Around **64K** lines of codes *excluding* drivers.



Motivation: Ceph

What is **Ceph**:

- Ceph is a software-defined distributed storage protocol.
- Ceph **Multi-Queue (MQ) Block Device Driver** is part of **Linux** operating system.
- Our first use case: **Ceph I/O Accelerator**



Research Problem

Revisiting MQ Linux Block I/O layer gives **2** research problems:

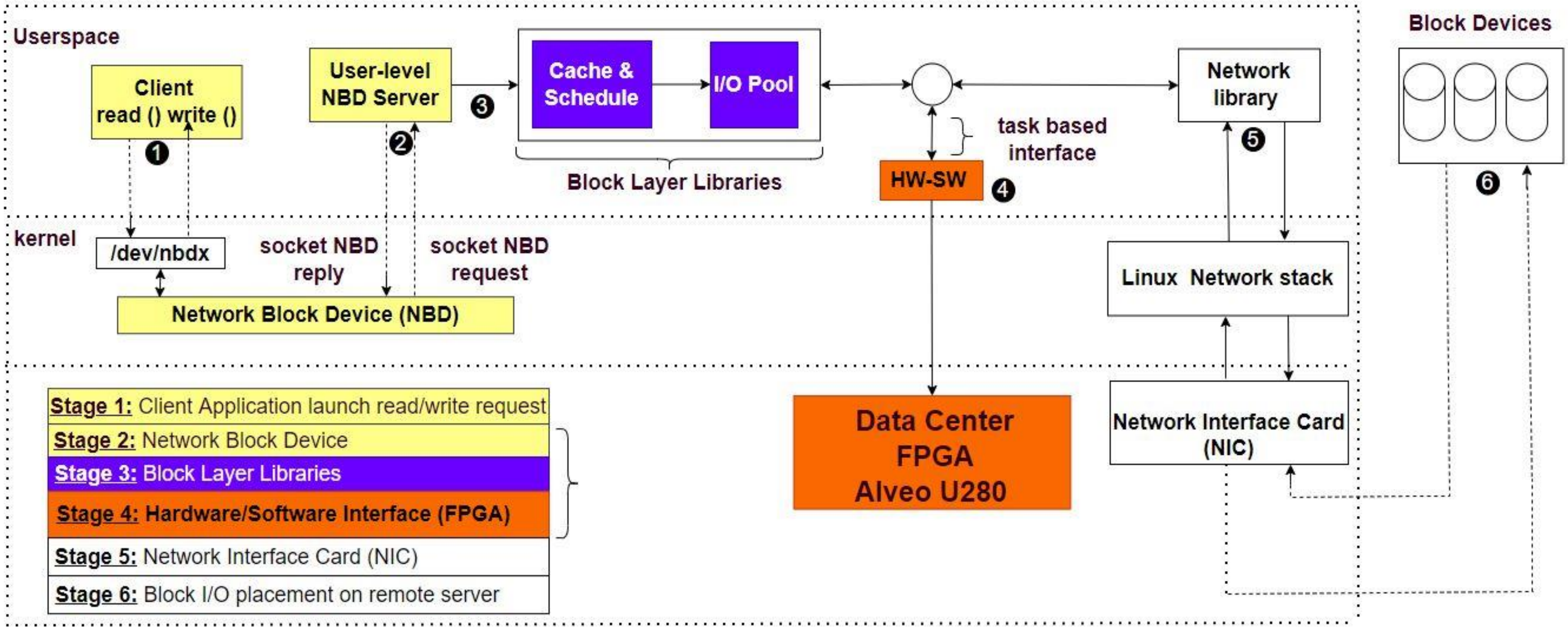
- **First**: MQ Block I/O Layer still has a ***performance bottleneck***.
- **Second**: MQ Block I/O Layer codebase is ***notoriously complex***.

DeLiBA Framework

DeLiBA addresses **both** research problems:

- Enables use of programming tools *at* ***userspace*** to tackle *complexity*
- Enables use of ***FPGA accelerators*** to tackle *performance bottleneck*

DeLiBA Architecture



Userspace

Client
read () write ()

❶

User-level NBD Server

❷

❸

Cache & Schedule

I/O Pool

Block Layer Libraries

task based interface

HW-SW

❹

Network library

❺

Block Devices

❻

kernel

/dev/nbdx

socket NBD reply

socket NBD request

Network Block Device (NBD)

Linux Network stack

Network Interface Card (NIC)

Data Center FPGA Alveo U280

Stage 1: Client Application launch read/write request

Stage 2: Network Block Device

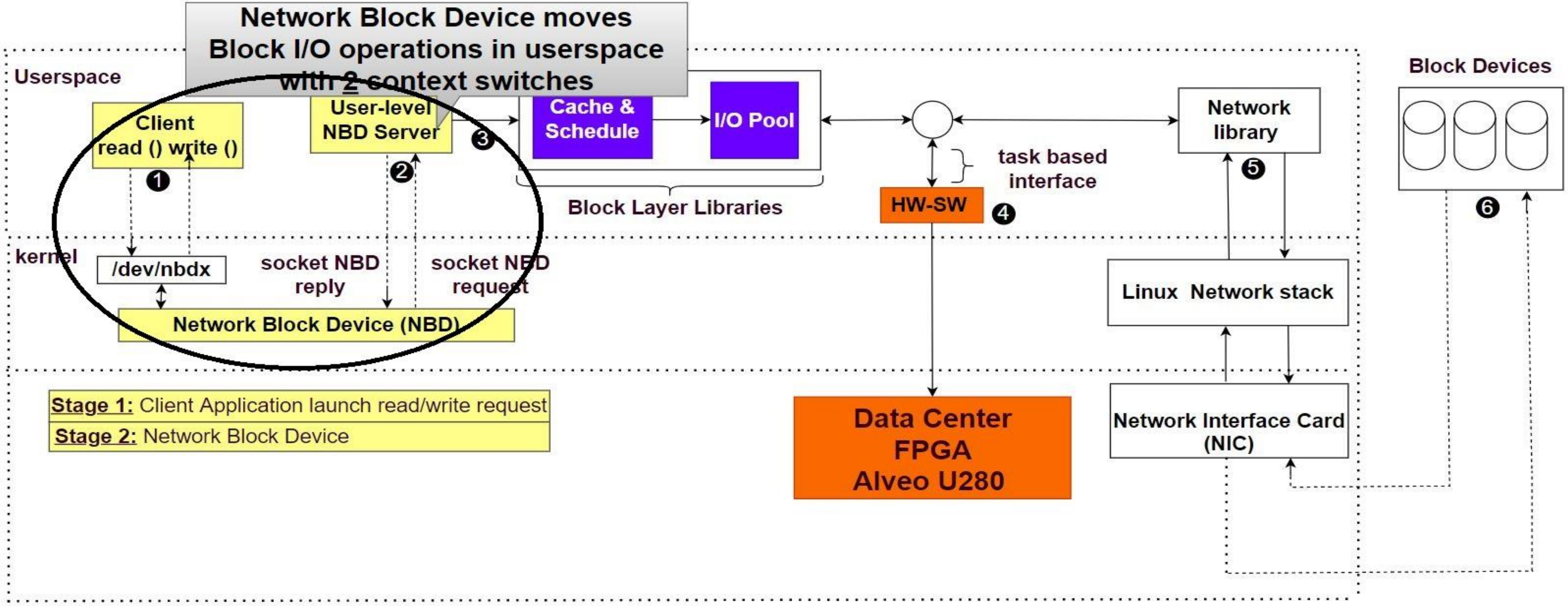
Stage 3: Block Layer Libraries

Stage 4: Hardware/Software Interface (FPGA)

Stage 5: Network Interface Card (NIC)

Stage 6: Block I/O placement on remote server

Userspace: Network Block Device (NBD)

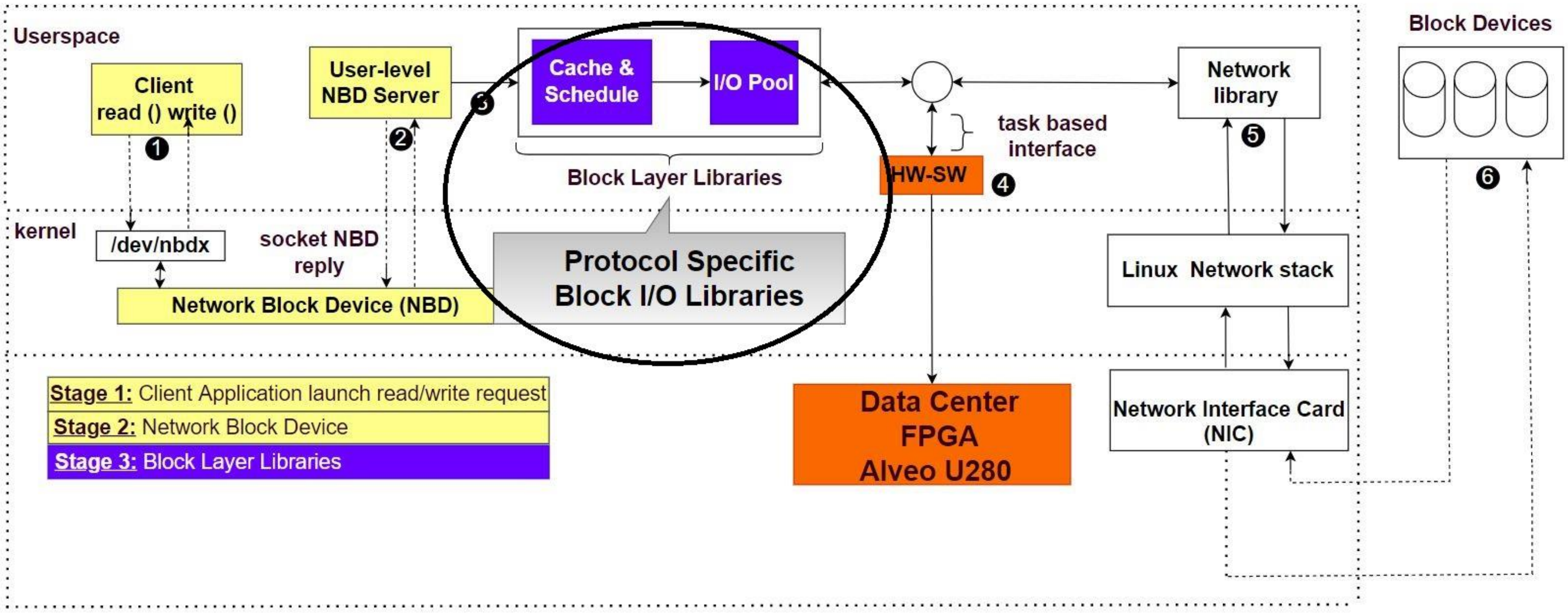


Network Block Device moves Block I/O operations in userspace with 2 context switches

Stage 1: Client Application launch read/write request

Stage 2: Network Block Device

Ceph Protocol Specific Libraries

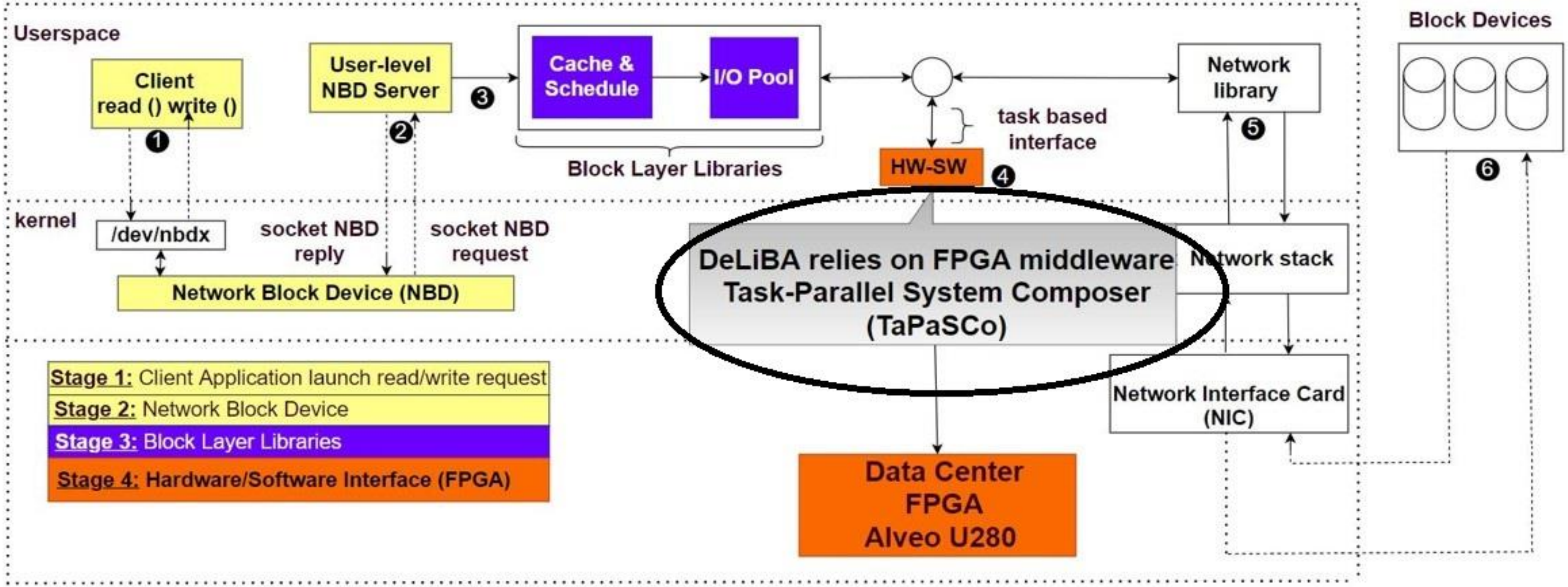


Stage 1: Client Application launch read/write request

Stage 2: Network Block Device

Stage 3: Block Layer Libraries

Hardware/Software Interface for FPGA



DeLiBA relies on FPGA middleware Task-Parallel System Composer (TaPaSCo)

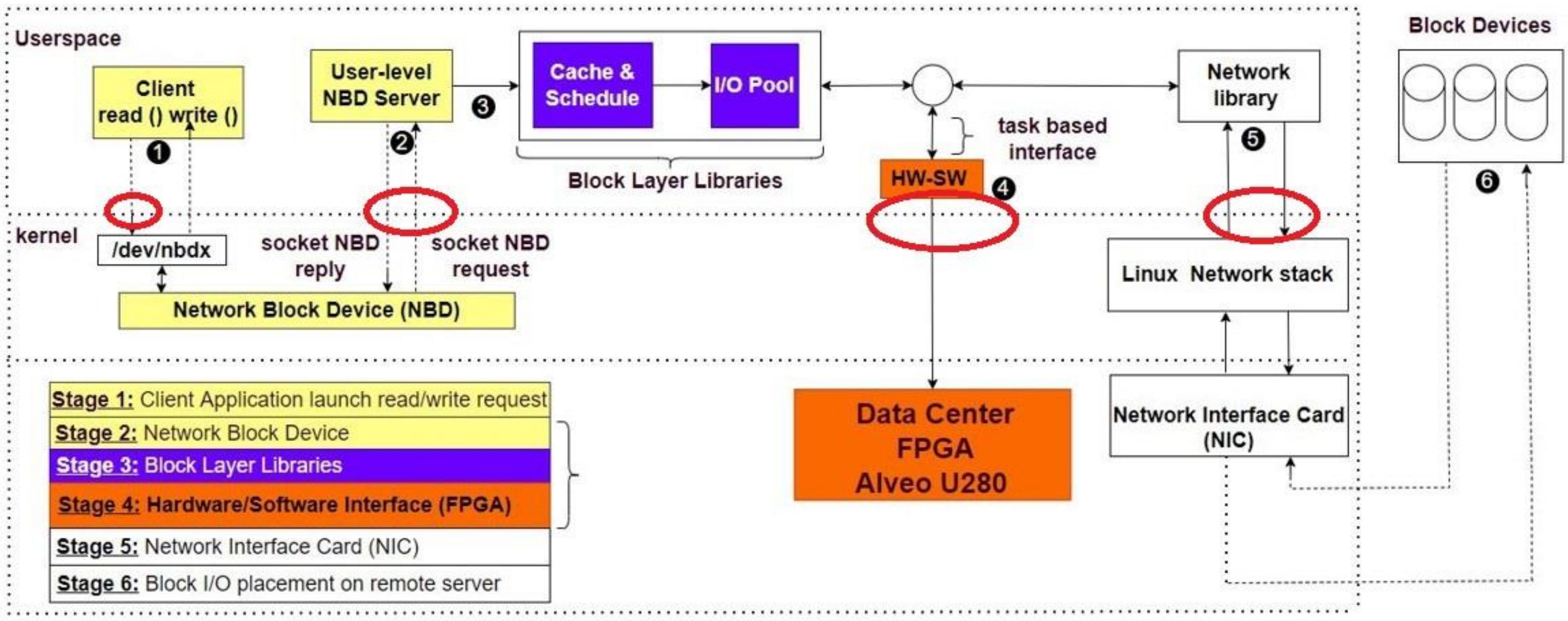
Stage 1: Client Application launch read/write request

Stage 2: Network Block Device

Stage 3: Block Layer Libraries

Stage 4: Hardware/Software Interface (FPGA)

Overall Caveats i.e. Context Switches



Stage 1: Client Application launch read/write request

Stage 2: Network Block Device

Stage 3: Block Layer Libraries

Stage 4: Hardware/Software Interface (FPGA)

Stage 5: Network Interface Card (NIC)

Stage 6: Block I/O placement on remote server

I/O Accelerator

Ceph I/O Accelerator is based on C++ Vitis High-Level Synthesis (HLS) for target clock frequency **300 MHz** (pre-synthesis):

- **Interface Synthesis:**
  AXI memory mapped automated by TaPaSCo.

- **Algorithmic Synthesis:**
  HLS based transformations i.e. Loop and Memory optimizations.

Hardware Results and Speedups

kernel	Software Execution Time	Hardware kernel Execution	Total Execution with Hardware
Straw Bucket (pure HLS code)	85 µs	0.675 µs	70 µs
Straw Bucket (using Vitis ln x function IP)	85 µs	0.885 µs	70 µs
List Bucket	65 µs	0.280 µs	72 µs
Uniform Bucket	20 µs	2.240 µs	25 µs
Tree Bucket	45 µs	0.810 µs	45 µs

- **Per kernel speedup:** 120x
- **Overall Speedup:** 1.2x (huge potential for further acceleration)

Evaluation on Hardware

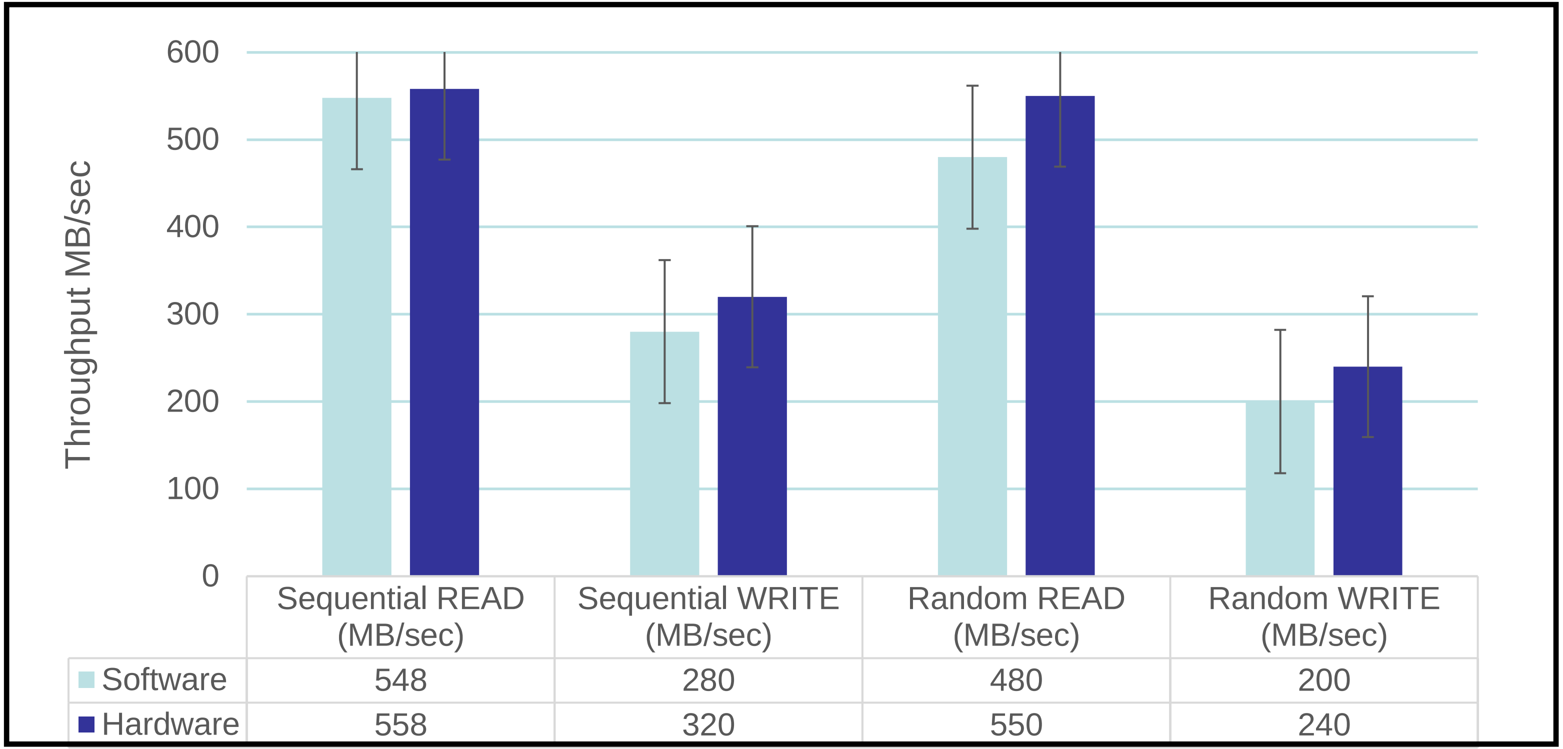
Following Hardware setup:

- AMD EPYC Rome 7302P
  16-core CPU with **128GB** of memory, attached by **10 Gb/s** Ethernet to the Ceph server.

- Xilinx Alveo U280 FPGA card attached to the client host by **PCIe Gen3 x8** and uses a system clock of **200 MHz**

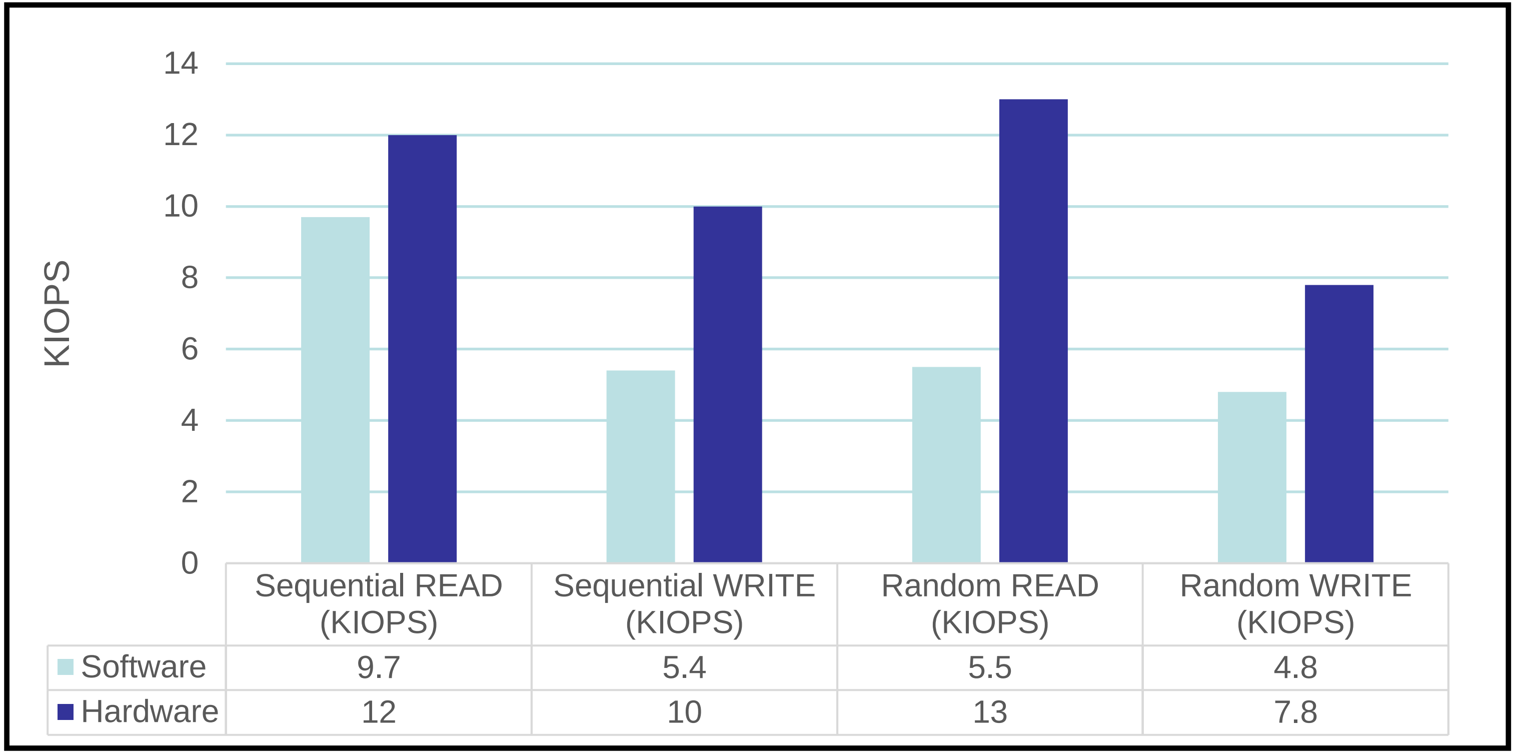
Xilinx Alveo U280 FPGA Card at ESA Group

Evaluation Hardware – Throughput (128KB)



	Sequential READ (MB/sec)	Sequential WRITE (MB/sec)	Random READ (MB/sec)	Random WRITE (MB/sec)
■ Software	548	280	480	200
■ Hardware	558	320	550	240

Evaluation Hardware – IOPS (4KB)



	Sequential READ (KIOPS)	Sequential WRITE (KIOPS)	Random READ (KIOPS)	Random WRITE (KIOPS)
■ Software	9.7	5.4	5.5	4.8
■ Hardware	12	10	13	7.8

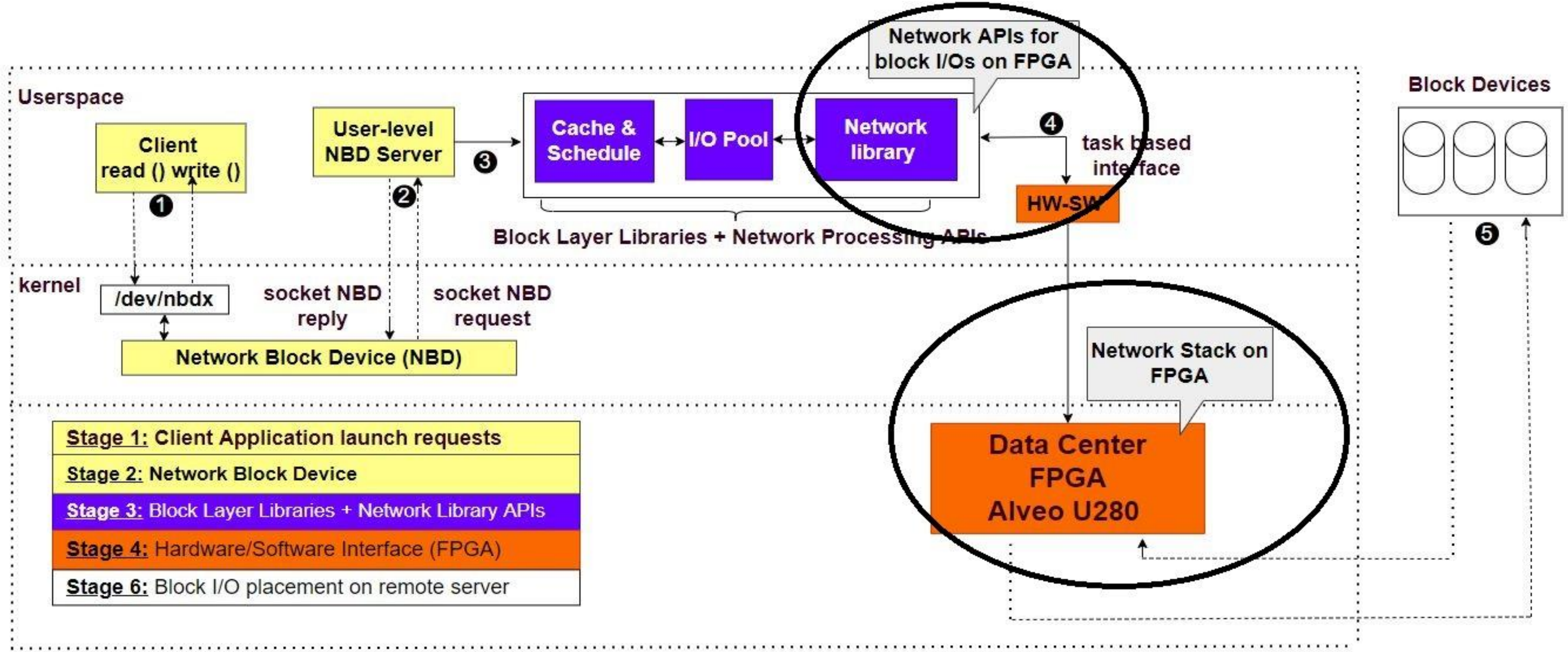
Conclusion

- **Performance Gain (Speedups) Throughput**:
    - **1.2x** & **1.9x** for Rand writes (128KB) & Seq Writes (4KB) resp.

- **Performance Gain (Speedups) IOPS:**
    - **2.36x** for 4KB Rand reads (4KB)

- Through **DeLiBA** initial goal of easy ***programmability*** achieved

Future Work – DeLiBA SmartNIC



Stage 1: Client Application launch requests

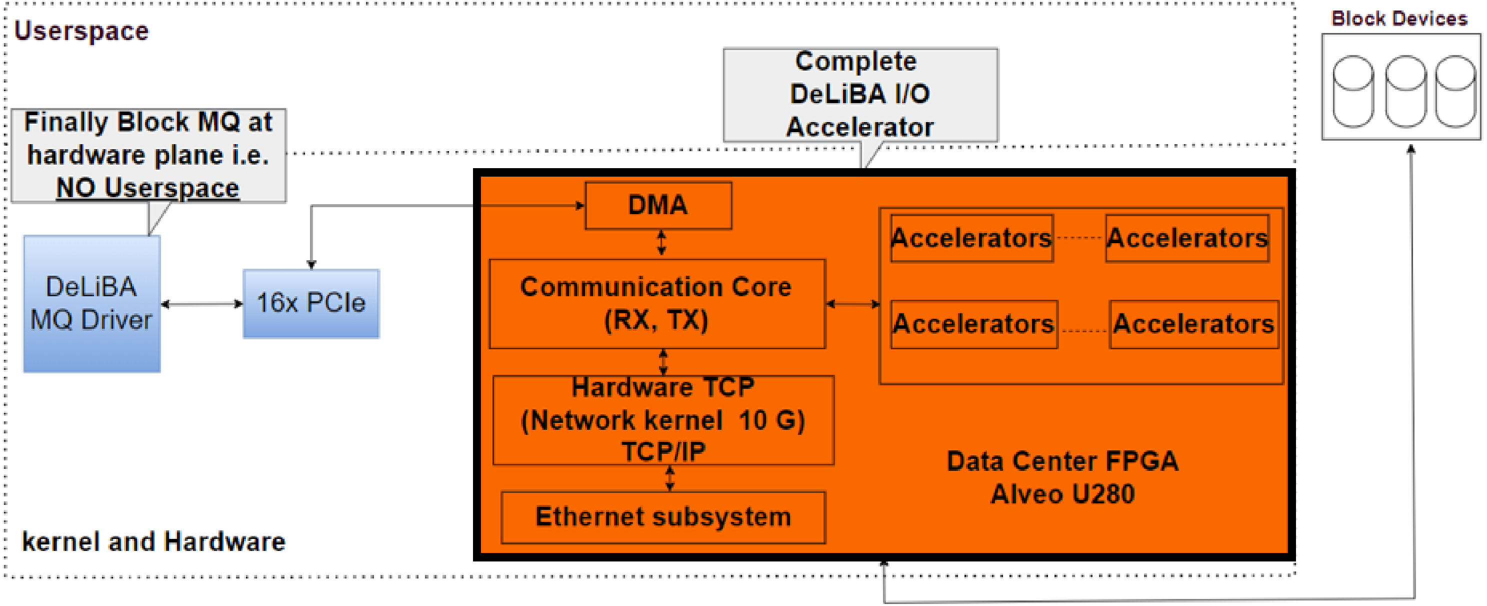
Stage 2: Network Block Device

Stage 3: Block Layer Libraries + Network Library APIs

Stage 4: Hardware/Software Interface (FPGA)

Stage 6: Block I/O placement on remote server

Future Work – DeLiBA MQ Driver


Userspace
Block Devices
Complete DeLiBA I/O Accelerator
Finally Block MQ at hardware plane i.e. NO Userspace
DeLiBA MQ Driver
16x PCIe
DMA
Communication Core (RX, TX)
Accelerators
Accelerators
Accelerators
Accelerators
Hardware TCP (Network kernel 10 G) TCP/IP
Ethernet subsystem
Data Center FPGA Alveo U280
kernel and Hardware

DeLiBA is open-source

- DeLiBA is available at our ESA github:

  https://github.com/esa-tu-darmstadt/deliba

QR code for our DeLiBA repo

References

[1] A. M. Caulfield, A. De, J. Coburn, T. I. Mollow, R. K. Gupta, and S. Swanson, “Moneta: A high-performance storage array architecture for next-generation, non-volatile memories,” in 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture, 2010

[2] H.-J. Kim, Y.-S. Lee, and J.-S. Kim, “NVMeDirect: A user-space I/O framework for application-specific optimization on NVMe SSDs,” in 8th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 16). Denver, CO: USENIX Association, Jun. 2016. [Online]. Available: https://www.usenix.org/conference/hotstorage16/workshop-program/presentation/kim

[3] A. Stratikopoulos, C. Kotselidis, J. Goodacre, and M. Luj ́an, “Fastpath: Towards wire-speed nvme ssds,” in 2018 28th International Conference on Field Programmable Logic and Applications (FPL), 2018,

THANKS FOR YOUR ATTENTION!

……. looking forward to interesting discussions in the **FPGA Design** panel of FPL 2022