Tim Dettmar

In-Network Computing & DPUs

State of the Art

Table of contents

Outline

What is “in-network” computing?

- Moving operations traditionally performed by general-purpose hardware (e.g. CPUs/GPUs) to network hardware

[Zil19]

Rationale

If the CPU can do everything, why?

- It can do everything, but **slowly**
- We paid for the CPU to crunch data, not process packets
- The 100 Gbps+ era **requires** acceleration!

What can we optimise?

- The NIC itself
- Network equipment

[Zil19]

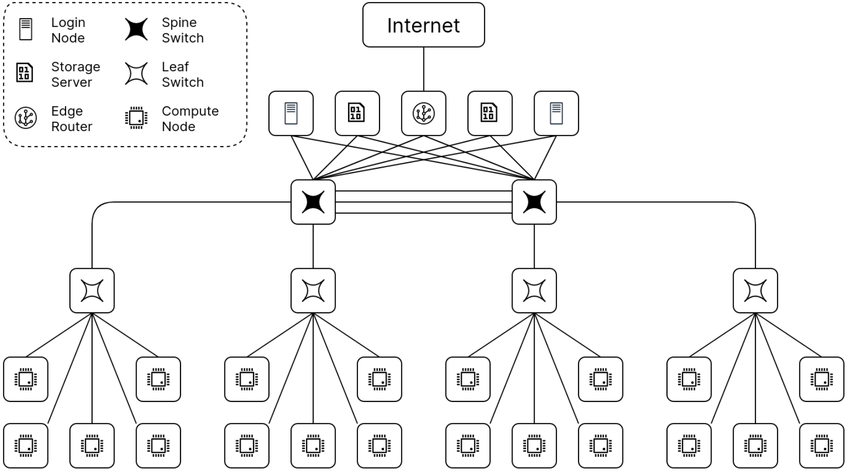
Data Analytics Workflow

Analysis of large datasets can be broken down into basic steps

- Load data from non-volatile storage
- Slice and distribute data across compute nodes
- Perform data processing
- Gather and transform the results

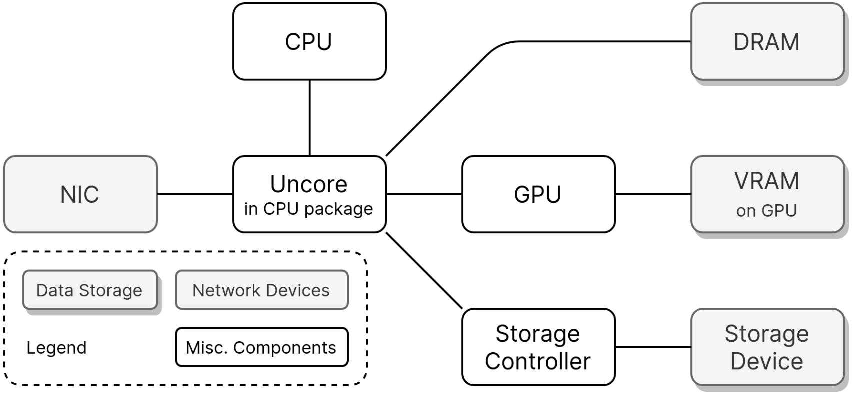
Can we do some of these steps **in-network**?

Simplified HPC network configuration



- In reality, there can be more network layers, nodes, etc.

Compute Node



- Uncore contains the “non-core" components (e.g. PCIe/DRAM controller)
- aka “System Agent", “Data Fabric", “Infinity Fabric", etc.

Loading data from non-volatile storage

What does “loading data” from a storage server involve, anyway?

- Finding the data you actually want to serve
- Creating and storing connection states
- Receiving and loading the data on the client side

Loading data from non-volatile storage

Finding the data you actually want to serve

- Translating a file path into block number and length
- Requesting the block + length from the disk = **data**

Creating TCP/UDP connections

- Keeping track of send/receive buffers
- Constructing packets with our **data**
- Temporarily storing in-flight data
- Checking firewall rules
- Performing checksum calculations
- Copying data between kernel and userspace
- Packet loss recovery

And again on the receiving side!

[Ros14]

Outline

Basic (Foundational) NICs

- Most likely in your home computer
- Basic offload capabilities, such as checksums
- Cheap, low power, works well enough for consumers

Figure: Realtek RTL8111B, a popular GbE chip

[NVI21a; Haa13]

Converged & Enterprise NICs

- Midrange option found in general-purpose servers
- Fixed offload options e.g. TCP stack, RDMA, FC
- Good price-performance ratio

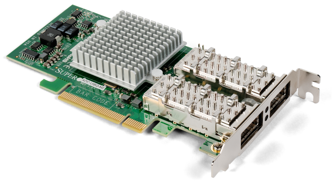


Figure: Mellanox ConnectX-2 with FC and RDMA offload

[NVI21a; Nos15]

SmartNICs

- Beyond marketing, what makes a network adapter smart?
- The ability to be **reprogrammed**
- Multiple ways to realise this

[NVI21a]

FPGA-based SmartNICs

- FPGA placed in the path of data traffic
- Allows for fully custom, user-defined offload
- ASIC-like speeds for e.g. real-time IPSec, packet inspection
- Difficult to program

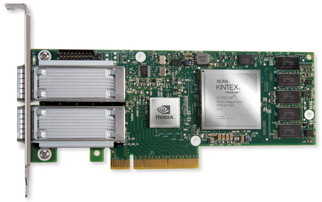


Figure: Mellanox Innova-2 Flex FPGA SmartNIC

[NVI21a; NVI22a]

SoC-based SmartNICs

- Commonly known as a DPU (Data Processing Unit)
- Custom/semi-custom SoC(s) on board (typ. ARM/MIPS)
- Runs a standard operating system (typ. Linux)
- Easily programmable

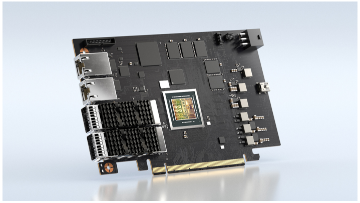


Figure: Mellanox BlueField-2 DPU

[NVI21a; NVI22b]

Exotic SmartNICs

- Custom NICs designed to address a specific niche
- Tightly integrated packages

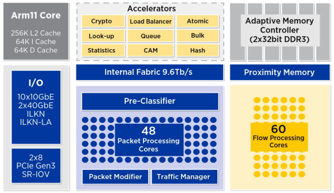


Figure: Netronome NFP-4000 Flow Processor ASIC

[Net20]

Outline

Most obvious solution: Offloading TCP

If TCP has so much overhead, why not use the NIC?

- A solution called the TCP Offload Engine (TOE)

[Wil18; Che22; Mog03]

Most obvious solution: Offloading TCP

If TCP has so much overhead, why not use the NIC?

- A solution called the TCP Offload Engine (TOE)
- Criticised for its...

[Wil18; Che22; Mog03]

Most obvious solution: Offloading TCP

If TCP has so much overhead, why not use the NIC?

- A solution called the TCP Offload Engine (TOE)
- Criticised for its...
  - Lacklustre performance

[Wil18; Che22; Mog03]

Most obvious solution: Offloading TCP

If TCP has so much overhead, why not use the NIC?

- A solution called the TCP Offload Engine (TOE)
- Criticised for its...
  - Lacklustre performance
  - Security flaws

[Wil18; Che22; Mog03]

Most obvious solution: Offloading TCP

If TCP has so much overhead, why not use the NIC?

- A solution called the TCP Offload Engine (TOE)
- Criticised for its...
  - Lacklustre performance
  - Security flaws
  - Limited support period

[Wil18; Che22; Mog03]

Most obvious solution: Offloading TCP

If TCP has so much overhead, why not use the NIC?

- A solution called the TCP Offload Engine (TOE)
- Criticised for its...
  - Lacklustre performance
  - Security flaws
  - Limited support period
- TCP was designed well before SmartNICs were even a consideration!
- Windows TOE support was introduced in 2003 and deprecated in 2012
- The upstream Linux kernel **never** supported it
- Only Chelsio actively manufactures TOE NICs in 2022

[Wil18; Che22; Mog03]

A New Framework

How about a framework for zero-copy that's easier to implement in hardware?

- A class of protocols under the term Remote Direct Memory Access (RDMA)
- Enables zero-copy networking without involving the kernel
  - No context-switching overhead
  - Direct, low-latency communication with the NIC
  - Direct data placement into **user buffers**
- Widely implemented in HPC
  - Ethernet RoCE (Mellanox, Intel, Marvell, Broadcom)
  - InfiniBand RDMA (Mellanox)
  - iWARP (Intel, Marvell, Chelsio)
  - Omni-Path (Intel)
  - Proprietary, e.g. Amazon EFA, IBM BlueGene/Q

[Ros14]

RDMA Programming

- To enable direct communication and simpler hardware, the protocol is extremely **low-level**
- Many implementations conform to the de-facto standard “Verbs” API or use a more flexible library, e.g. libfabric

[Mel15; OFI22; Ros14]

RDMA Programming

TCP

- Send/recv buffers
- Active connection FDs

[Mel15; Ros14]

RDMA Programming

TCP

- Send/recv buffers
- Active connection FDs

Ping-pong: < 100 lines

[Mel15; Ros14]

RDMA Programming

TCP

- Send/recv buffers
- Active connection FDs

Ping-pong: < 100 lines

RDMA

- Send/recv buffers
- Active connection queue pairs
- Send/recv completion queues
- Completion channels
- Memory pinning details
- Memory protection domains

[Mel15; Ros14]

RDMA Programming

TCP

- Send/recv buffers
- Active connection FDs

Ping-pong: < 100 lines

RDMA

- Send/recv buffers
- Active connection queue pairs
- Send/recv completion queues
- Completion channels
- Memory pinning details
- Memory protection domains

Ping-pong: < 1000 lines

Many libraries have done the work for you though (e.g. MPI)

[Mel15; Ros14]

Benchmarks

- Performed on GWDG cluster with Intel OPA @ 100 Gbit/s
- Benchmark: `fi_pingpong`
  - Libfabric 1.10 with psm2 and tcp;ofi_rxm transports
  - Power of 2 message sizes from 1 byte - 8 MiB

Performance

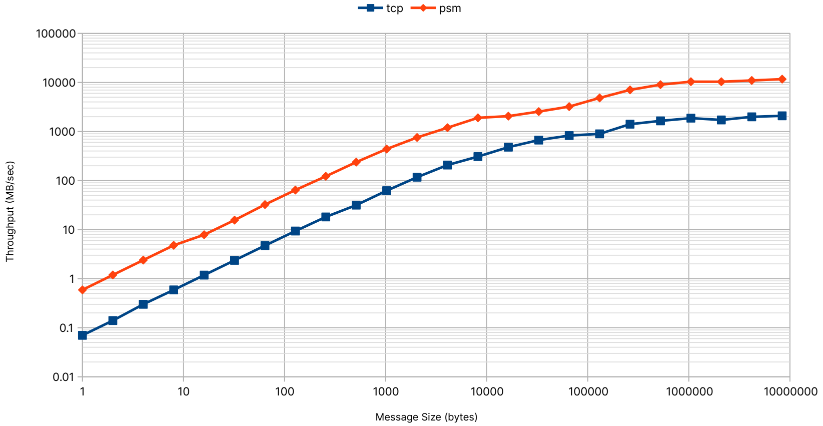


Figure: Throughput comparison

Performance

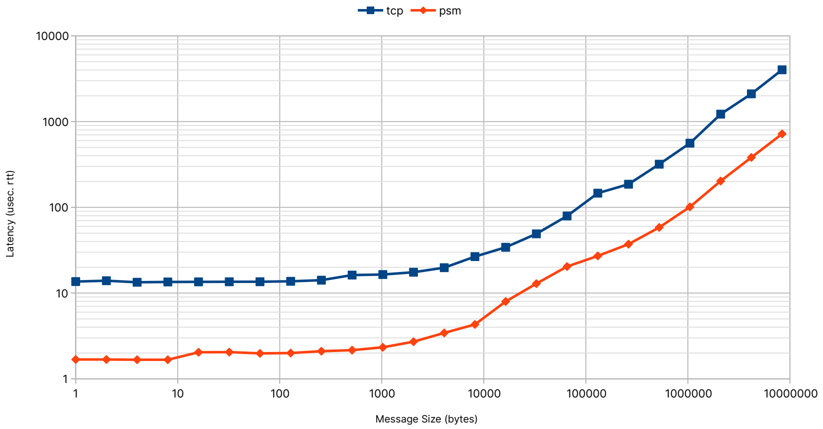


Figure: Latency comparison

Performance

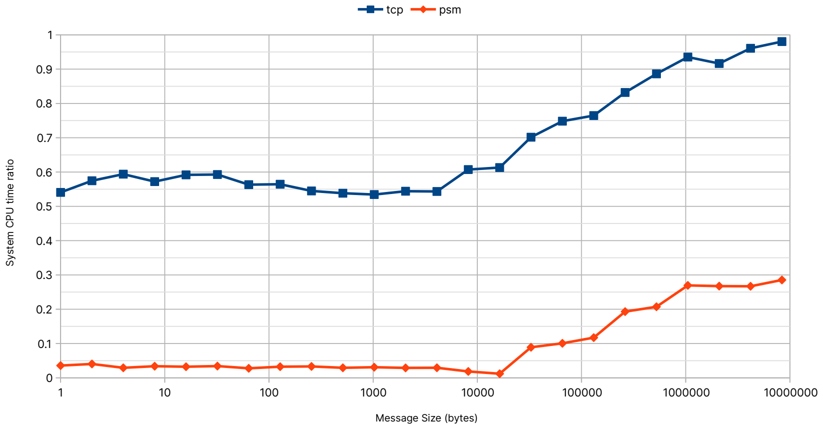


Figure: Kernel CPU time as a ratio of total CPU time

Summary

- Higher throughput
  - Get your data from A to B faster
  - Process data sooner, higher node utilization
- Lower latency
  - Increases performance of collective operations
  - Better performance in communication-heavy workloads
- Lower kernel time
  - Less time spent processing TCP means more time for useful work

Outline

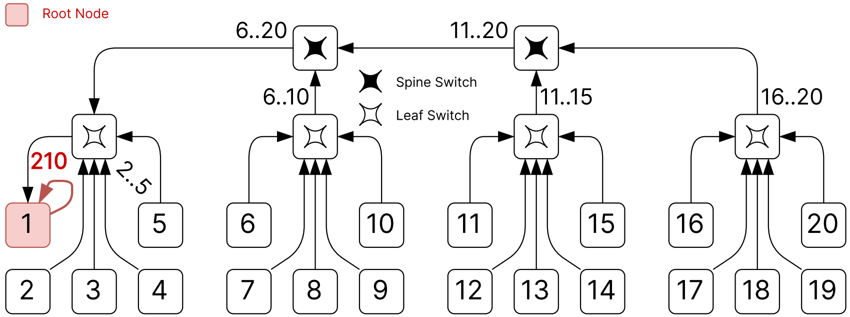
SHARP

The **S**calable **H**ierarchical **A**ggregation and **R**eduction **P**rotocol

- Introduced by Mellanox (NVIDIA) in the SwitchIB-2 series
- Performs on-switch aggregation to accelerate MPI collectives
  - Integer + FP ALU integrated into the switch ASIC
  - Supports sum, min, max, MinLoc, MaxLoc, OR, AND, XOR
- Can be accessed in OpenMPI with the `hcoll` package

[Gra+16]

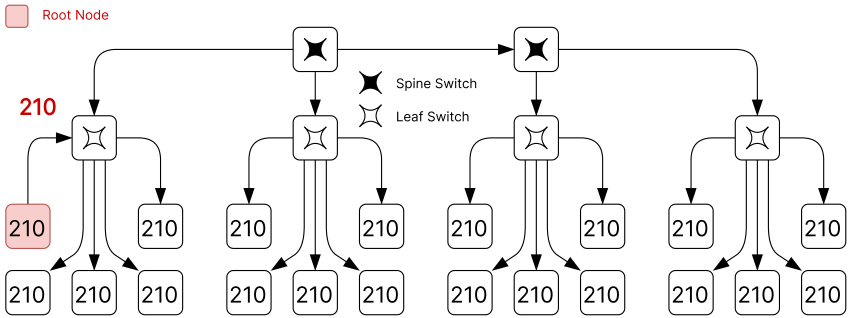
MPI Allreduce without SHARP



- Network demand of (n-1)x, where n = node count, x = data size
- Limited by root node-switch bandwidth, inter-switch bandwidth

Note: Many MPI implementations have more efficient methods
(e.g., recursive doubling, topology-aware)

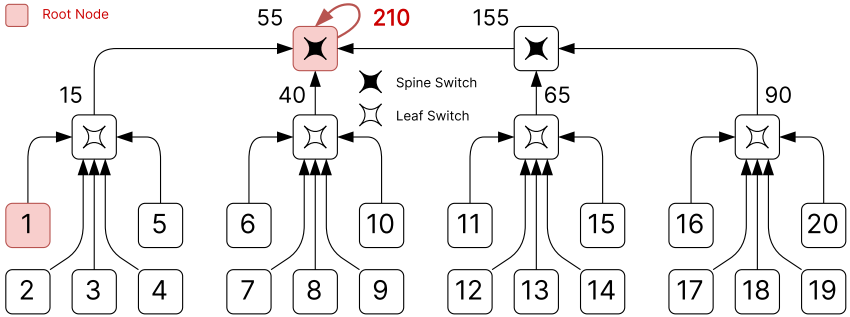
MPI Allreduce without SHARP



- Network demand of (n-1)x, where n = node count, x = data size
- Limited by root node-switch bandwidth, inter-switch bandwidth

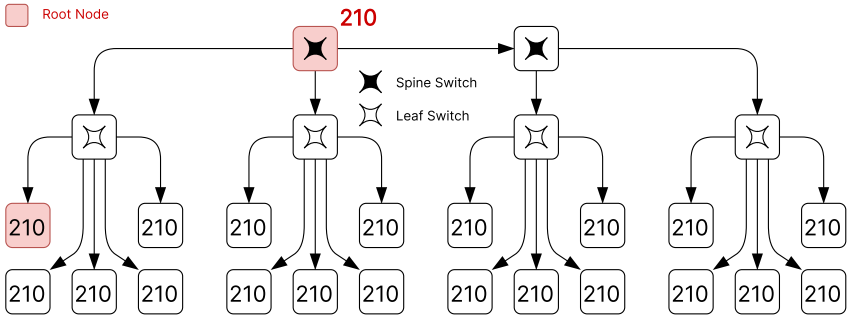
Note: Many MPI implementations have more efficient methods
(e.g., recursive doubling, topology-aware)

MPI Allreduce with SHARP



- Switches cooperate by aggregating data to be sent
- Constant network bandwidth utilization only dependent on data size
- Limited only by the slowest node in the network due to sync requirement

MPI Allreduce with SHARP



- Switches cooperate by aggregating data to be sent
- Constant network bandwidth utilization only dependent on data size
- Limited only by the slowest node in the network due to sync requirement

MVAPICH2-SHARP MPI Library Benchmarks

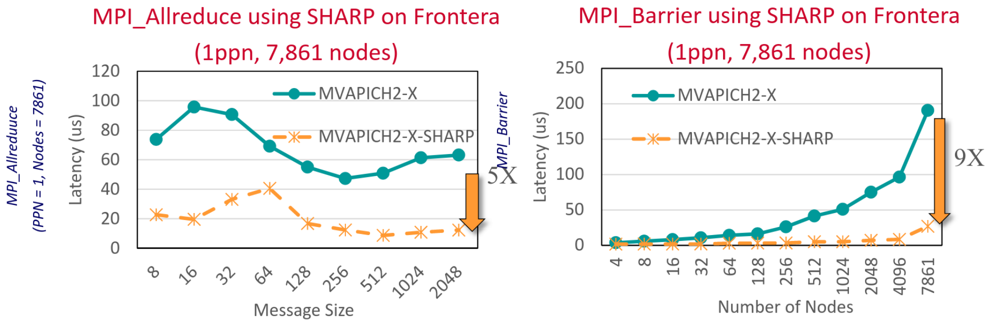


Figure: MVAPICH2 SHARP Benchmarks

[Sub21]

Outline

History

- DPUs have existed in some form or another for the past decade
  - 2006: Cavium Octeon Series (MIPS, C API)
  - 2010: Cavium LiquidIO Series (MIPS, Customizable SW Image)
- Interest in SmartNICs have increased over time as big players joined in
  - NVIDIA and Broadcom, among others
- Many SmartNICs now run full, **standard**, Linux images

[Cav06; Cav13; Bro19; NVI19]

Typical DPU architecture

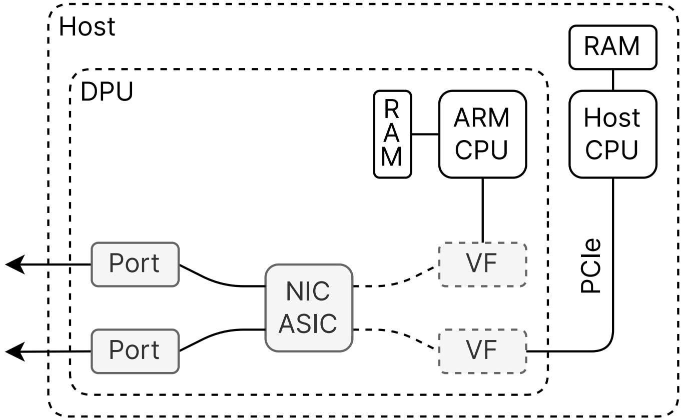


Figure: BlueField-2 Architecture

[NVI22c]

Use Cases

- DPUs can complement other existing in-network computing technologies
  - Burning CPU cycles for unnecessary tasks just increases your power bill
  - The presented SmartNICs can all be used with RDMA
  - The SHARP spec is transport provider-agnostic
- However, they offer an unprecedented level of flexibility
  - Anything you can program for a Linux machine
    - Manufacturers usually supply premade offloads (vSwitch, NVMeoF, etc.)
    - SDKs to simplify programming (DOCA etc.) but usually not cross-platform or OSS

[Bro19; NVI21b; Gra+16]

Data Transfer Offload

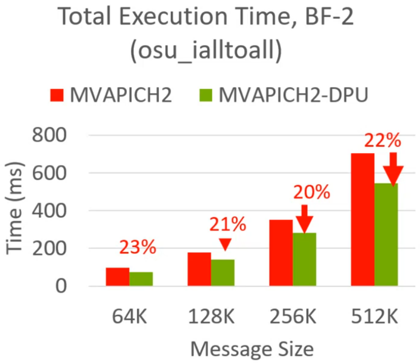
- RDMA can already transfer blocks of data without CPU involvement
- DPUs can perform **pre-processing** of the data as well
  - Normalization, discretization etc.
  - Removes the burden from the CPU allowing more **overlap**
    - Parallelized computation and data transfer/preprocessing stages
    - True parallelism with 100% host CPU util., not CPU time sharing

MPI Collective Offload

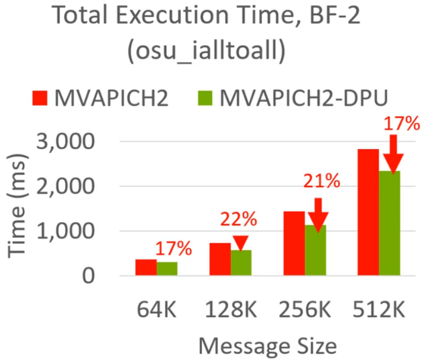
- Offloading of MPI collectives (e.g. All-All, AllGather, Bcast)
- DPUs can share results while host CPU works on new data
- Unfortunately solutions not open source or generally available
  - MVAPICH2-DPU - closed source version of MVAPICH with DPU support
  - dpu_mpi is a publicly available PoC we wrote for Ialltoall offload

[X-S22; Det+21]

MVAPICH2-DPU Benchmarks



32 Nodes, 16 PPN



32 Nodes, 32 PPN

Figure: MVAPICH2-DPU Benchmarks

[X-S22]

The perfect network

- ... A cluster filled with DPUs ...
- ... all with RDMA support ...
- ... and with SHARP capable switches connecting them all ...

The perfect network

- ... A cluster filled with DPUs ...
- ... all with RDMA support ...
- ... and with SHARP capable switches connecting them all ...

Then you see your budget, and reality hits you

Reality

- Evaluate what makes sense in **your** use case
  - Do I *really* have that many collectives in my code?
    - SHARP might make sense
  - Am I *really* transferring that much data?
    - RDMA NICs (even DPUs) might make sense
  - Should I get a faster NIC or just upgrade my CPUs?
    - The fastest NIC in the world won't help if your CPU can't keep up
    - ... and vice versa
    - DPUs may be cheaper than replacing the entire node

Further Reading

- Interested in programming for RDMA?
  - RDMA Aware Networks Programming User Manual
  - The Geek in the Corner - RDMA Programming
  - I’m also working on...
    - A clear, concise, easy to follow series of RDMA tutorials
    - Telescope Project (RDMA video/remote desktop streaming)
    - Let me know if you’re interested, or just submit a pull request!
- Interested in DPUs?
  - You’ll need Linux basics and a familiar grasp of a programming language
    - C if you want the best performance
    - You can also use other languages too with MPI libraries (e.g. Python)
- Interested in SHARP?
  - It’s as simple as enabling `hcoll` in OpenMPI, or the equivalent in your MPI

You’ll need to find the funding yourself, though...

[Mel15; Bed22; Tel22]

References I

[Bed22] Tarick Bedeir. *the geek in the corner - RDMA Programming Tutorial*. `https://github.com/tarickb/the-geek-in-the-corner`. 2022.

[Bro19] Broadcom. *Stingray™ PS250 Product Brief*. `https://docs.broadcom.com/doc/PS250-PB`. 2019.

[Cav06] Cavium Inc. *OCTEON™ XL NIC Accelerator Board Family - Product Brief*. `https://web.archive.org/web/20061021072446/http://www.cavium.com/pdfFiles/OCTEON_NIC_productBrief.pdf`. Internet Archive. 2006.

[Cav13] Cavium Inc. *Cavium LiquidIO Server Adapter Family - Product Brief*. `https://web.archive.org/web/20150415024533/http://www.cavium.com/pdfFiles/LiquidIO_Server_Adapters_PB_Rev1.2.pdf?x=1`. Internet Archive. 2013.

[Che22] Chelsio Inc. *Terminator 7 ASIC*. `https://www.chelsio.com/terminator-7-asic/`. 2022.

[Det+21] Tim Dettmar et al. *DPU_MPI Collective Offload engine for BlueField NICs*. `https://github.com/beanfacts/DPU_MPI`. 2021.

References II

[Gra+16] Richard L. Graham et al. “Scalable Hierarchical Aggregation Protocol (SHArP): A Hardware Architecture for Efficient Data Reduction”. In: *2016 First International Workshop on Communication Optimizations in HPC (COMHPC)*. 2016, pp. 1–10. DOI: 10.1109/COMHPC.2016.006.

[Haa13] Hans Haase. *PCIe Ethernet Controller RTL8111*. https://commons.wikimedia.org/wiki/File:PCIe_Ethernet_Controller_RTL8111_IMG_1030.JPG. Wikimedia Commons. 2013.

[Mel15] Mellanox Technologies Inc. *RDMA Aware Networks Programming User Manual*. https://web.archive.org/web/20220317144807/https://network.nvidia.com/related-docs/prod_software/RDMA_Aware_Programming_user_manual.pdf. Internet Archive. 2015.

[Mog03] Jeffrey C. Mogul. “TCP Offload is a Dumb Idea Whose Time Has Come”. In: *Proceedings of the 9th Conference on Hot Topics in Operating Systems - Volume 9*. HOTOS’03. Lihue, Hawaii: USENIX Association, 2003, p. 5.

[Net20] Netronome. *NFP-4000 Flow Processor ASIC*. https://www.netronome.com/media/documents/PB_NFP-4000-7-20.pdf. 2020.

References III

[Nos15] Dmitry Nosachev. *Supermicro AOC-UIBQ-M2 dual port InfiniBand HCA*. https://commons.wikimedia.org/wiki/File:Supermicro_AOC-UIBQ-M2_dual_port_InfiniBand_HCA.jpg. Wikimedia Commons. 2015.

[NVI19] NVIDIA Corporation. *NVIDIA BlueField DPU Platform Operating System v3.9.3.1 Documentation*. https://docs.nvidia.com/networking/display/BlueFieldDPUOSLatest. 2019.

[NVI21a] NVIDIA Corporation. *Choosing the best DPU based SmartNICs*. https://developer.nvidia.com/blog/choosing-the-best-dpu-based-smartnic/. 2021.

[NVI21b] NVIDIA Corporation. *NVIDIA BlueField-2 DPU*. https://resources.nvidia.com/en-us-accelerated-networking-resource-library/bluefield-2-dpu-datasheet?lx=LbHvpR&topic=networking-cloud. 2021.

[NVI22a] NVIDIA Corporation. *Innova-2 Flex*. https://www.nvidia.com/en-us/networking/ethernet/innova-2-flex/. 2022.

[NVI22b] NVIDIA Corporation. *NVIDIA BlueField Data Processing Units*. https://www.nvidia.com/en-us/networking/products/data-processing-unit/. 2022.

References IV

[NVI22c] NVIDIA Corporation. *NVIDIA BlueField Functional Diagram*. https://docs.nvidia.com/networking/display/BlueFieldDPUOSv370/Functional+Diagram. 2022.

[OFI22] OFI Working Group. *fi_provider(3) - Libfabric User Manual*. https://ofiwg.github.io/libfabric/v1.16.1/man/fi_provider.3.html. 2022.

[Ros14] Rami Rosen. *Linux Kernel Networking Implementation and Theory*. Apress, 2014.

[Sub21] Hari Subramoni. *The MVAPICH2 Project Latest Status and Future Plans*. https://www.mpich.org/static/docs/slides/2021-sc-bof/MVAPICH2.pdf. 2021.

[Tel22] Telescope Project Authors. *Telescope Project*. https://github.com/telescope-proj. 2022.

[Wil18] Brandon Wilson. *Why Are We Deprecating Network Performance Features (KB4014193)?* https://techcommunity.microsoft.com/t5/core-infrastructure-and-security/why-are-we-deprecating-network-performance-features-kb4014193/ba-p/259053. 2018.

[X-S22] X-Scale Solutions. *MVAPICH2-DPU*. https://x-scalesolutions.com/mvapich2-dpu/. 2022.

[Zil19] Noa Zilberman. *In-Network Computing*. https://www.sigarch.org/in-network-computing-draft/. 2019.

Contact Details

- Tim Dettmar
- Applied Computer Science MSc Student @ Uni Göttingen
- Email: hpc@timd.io