High-Performance Communication: RDMA, UCX & HPC-X

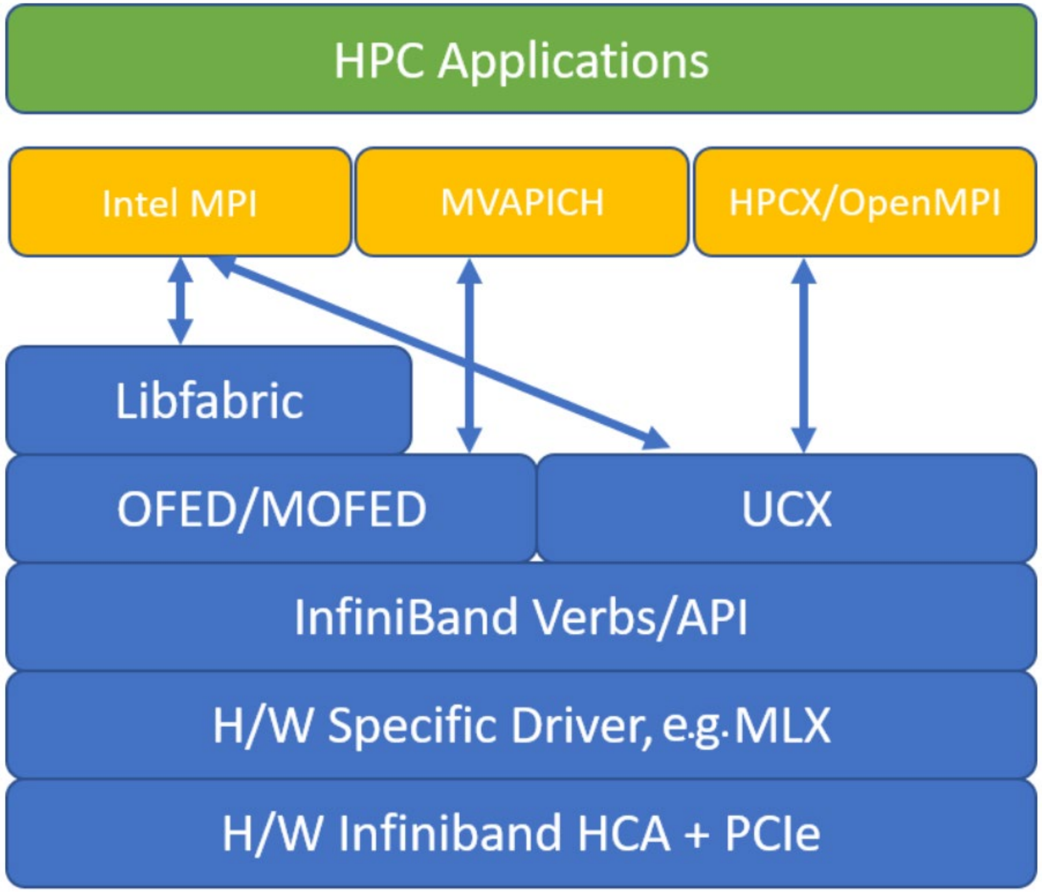
授課老師: 周志遠

NARLabs 財團法人國家實驗研究院
國家高速網路與計算中心
National Center for High-performance Computing

國立清華大學叢集電腦競賽團隊
Student Cluster Competition Team of NTHU

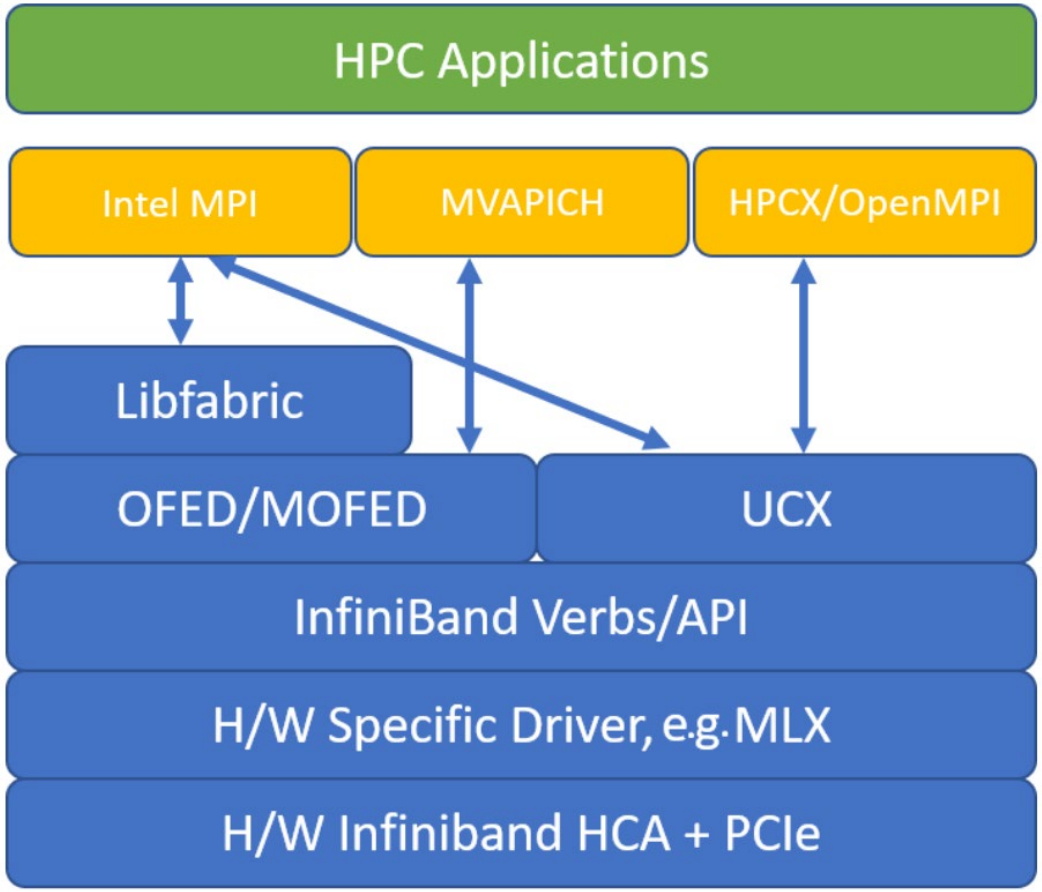
Outline

- RDMA Technology
- Verbs
- UCX
- HPC-X



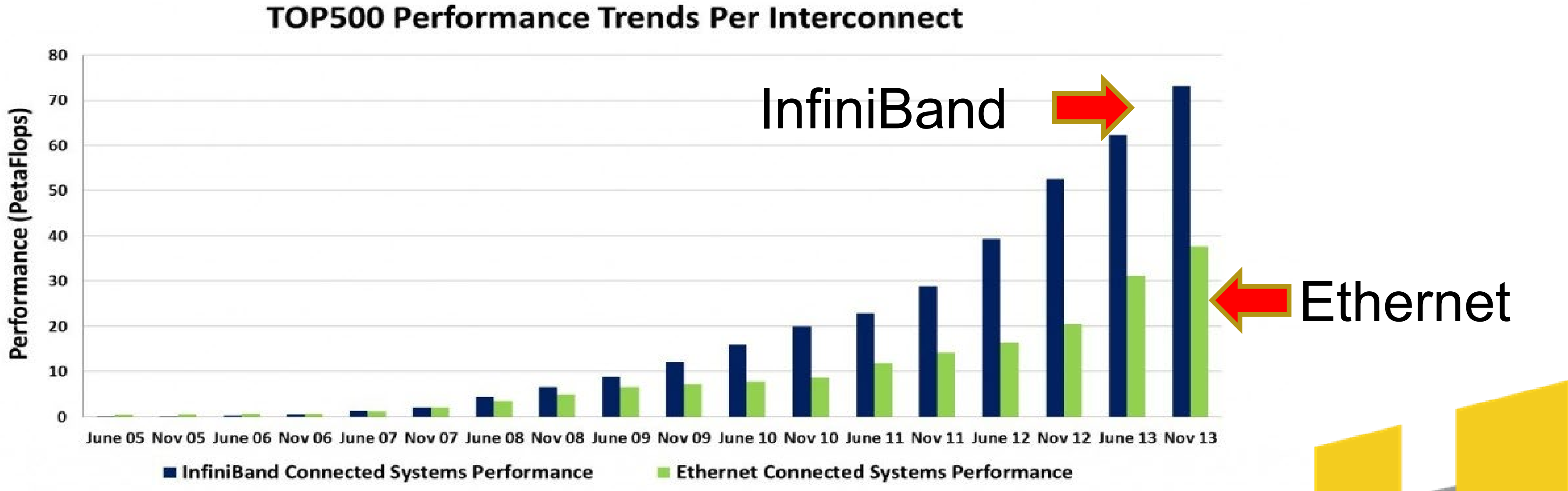
Outline

- RDMA Technology
- Verbs
- UCX
- HPC-X



Network Device: InfiniBand

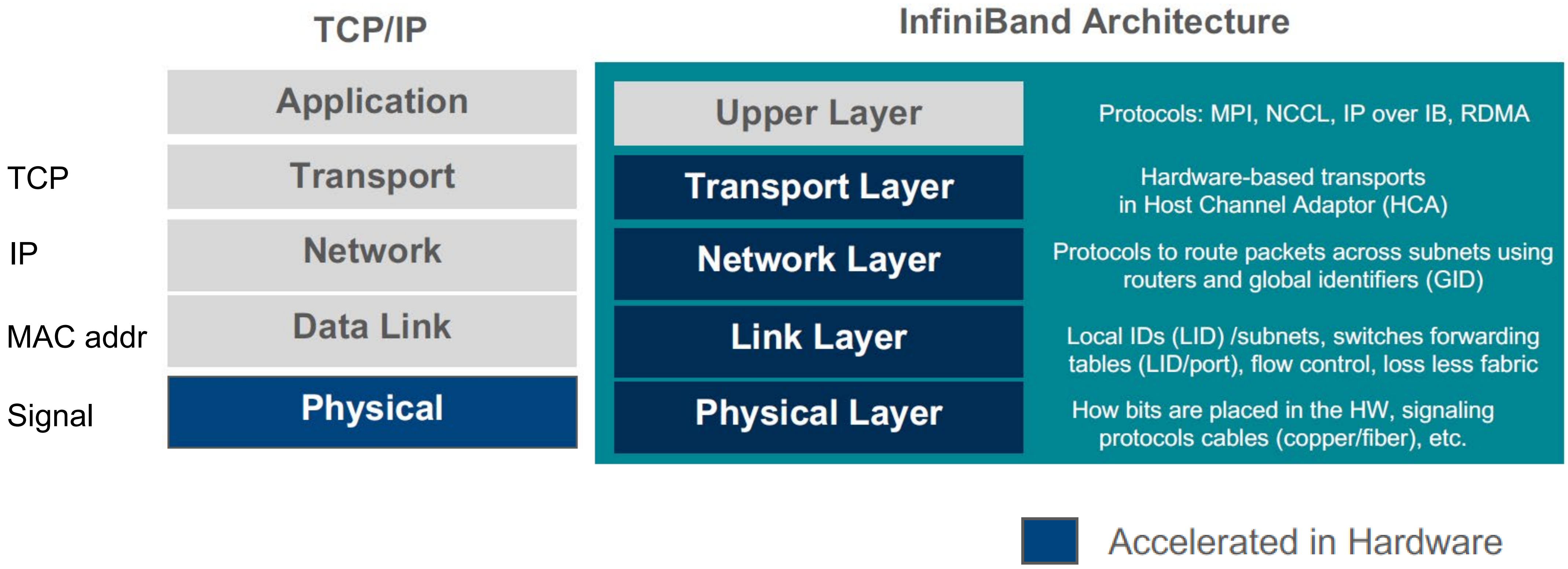
- A computer network communications link used in high-performance computing featuring very high throughput
- It is the most commonly used interconnect in supercomputers
- Manufactured by Mellanox



InfiniBand vs. Gigabit Ethernet

	InfiniBand	Ethernet
Protocol	Guaranteed credit based flow control	Best effort delivery
	End-to-End congestion management	TCP/IP protocol. Designed for L3/L4 switching
	Hardware based retransmission	Software based retransmission
RDMA	YES	NO (only now starting)
Latency	Low	High
Throughput	High	Low
Max cable length	4km	upto 70km
Price	36port switch: 25k USD QDR adapter: 500USD	36port switch: 1.5k USD Network card: 50 USD

TCP/IP vs InfiniBand Architecture



	TCP/IP	InfiniBand Architecture	
	Application	Upper Layer	Protocols: MPI, NCCL, IP over IB, RDMA
TCP	Transport	Transport Layer	Hardware-based transports in Host Channel Adaptor (HCA)
IP	Network	Network Layer	Protocols to route packets across subnets using routers and global identifiers (GID)
MAC addr	Data Link	Link Layer	Local IDs (LID) /subnets, switches forwarding tables (LID/port), flow control, loss less fabric
Signal	Physical	Physical Layer	How bits are placed in the HW, signaling protocols cables (copper/fiber), etc.

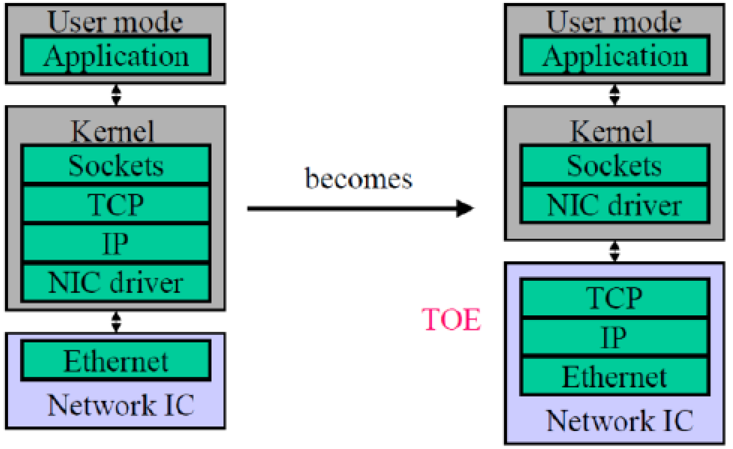
■ Accelerated in Hardware

Key Features

- Transport offload
- Bypassing the OS
- Communication Model
  - Two-sided communication model
    - Send and receive model
  - One-sided communication model
    - Remote memory access and atomics
- Rendezvous
  - Two sides exchange meta-data and use one-sided operations for bulk transfer

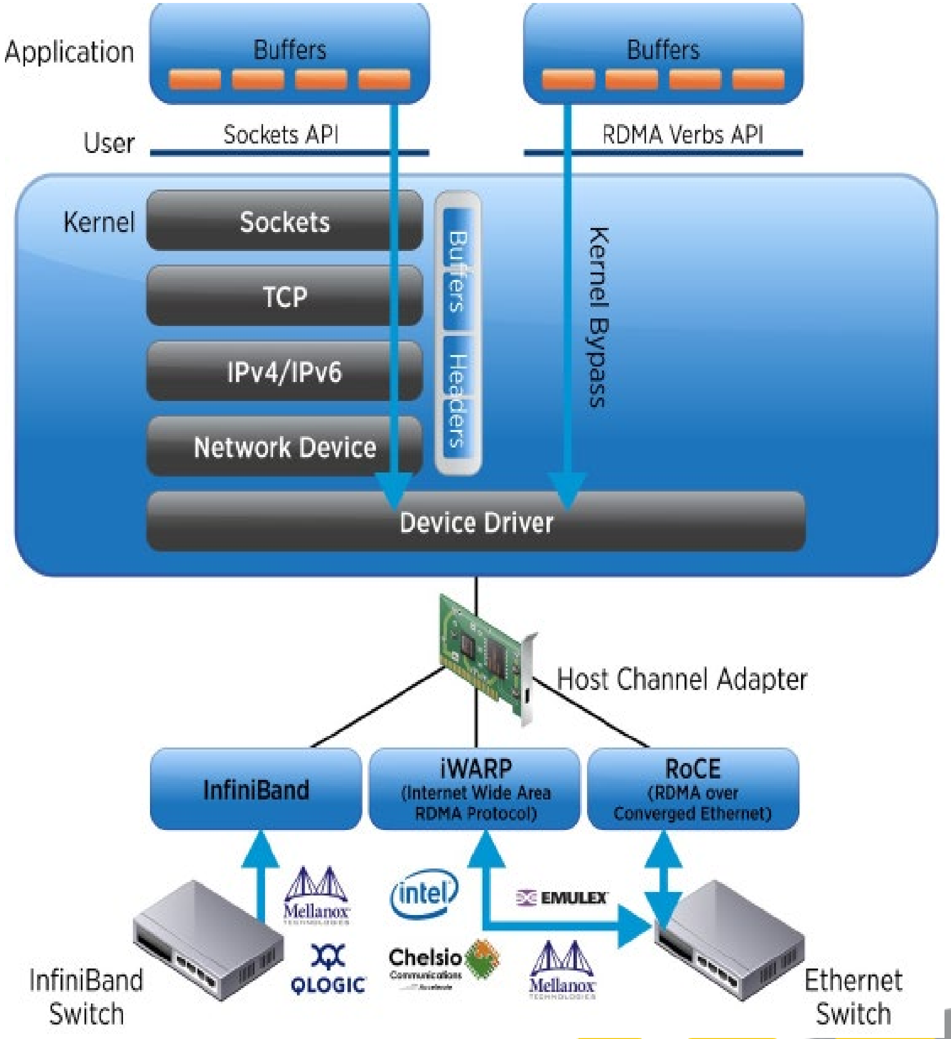
Acceleration by Offloading

- What is offloading? Asking somebody else to do the work!
  - TCP offloading: Moving IP and TCP processing to the Network Interface (NIC)
  - Checksum offloading: Moving the checksum calculation to the NIC (special circuits)
- Main justification for communication offloading
  - Reduction of host CPU cycles for protocol header processing, checksumming
  - Fewer CPU interrupts
  - Fewer bytes copied over the memory bus
  - Potential to offload expensive features such as encryption
- New performance metric: CPU Utilization
  - The rate (or % of time) the CPU is used for actual work
  - Time spent on communication is time wasted...



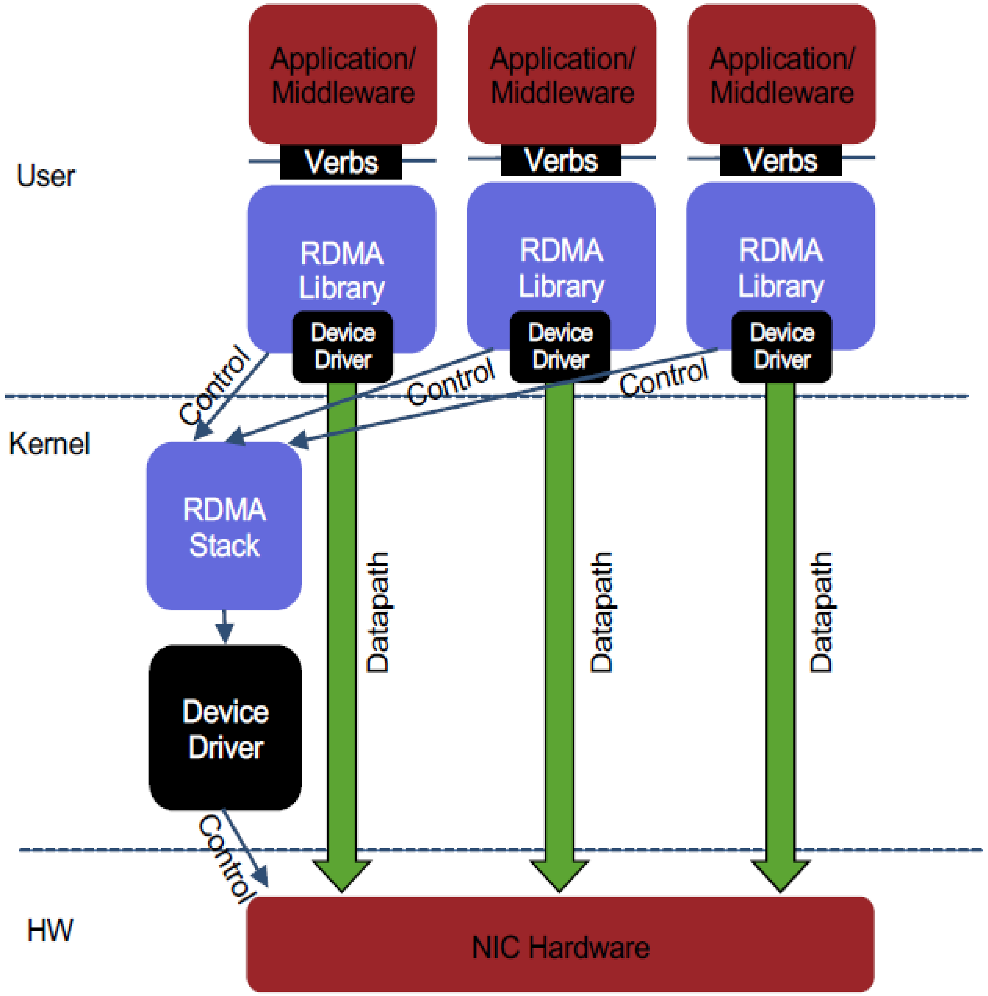
RDMA (Remote Direct Memory Access): Bypassing the OS

- Basic working principles:
  - RDMA traffic sent directly to NIC without interrupting CPU
  - A remote memory region registers with the NIC first
  - NIC records virtual to physical page mappings.
  - When NIC receives RDMA request, it performs a Direct Memory Access into memory and returns the data to client.
  - Kernel bypass on both sides of traffic



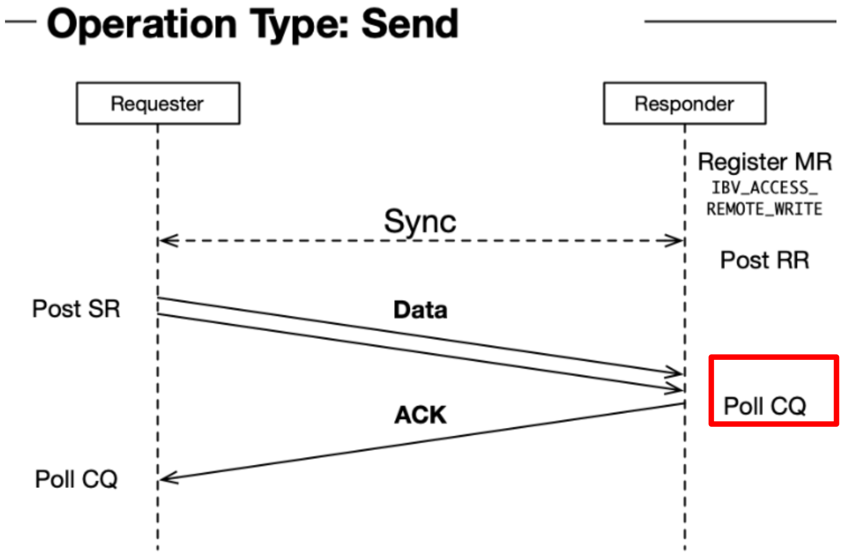
Enabling Kernel Bypass

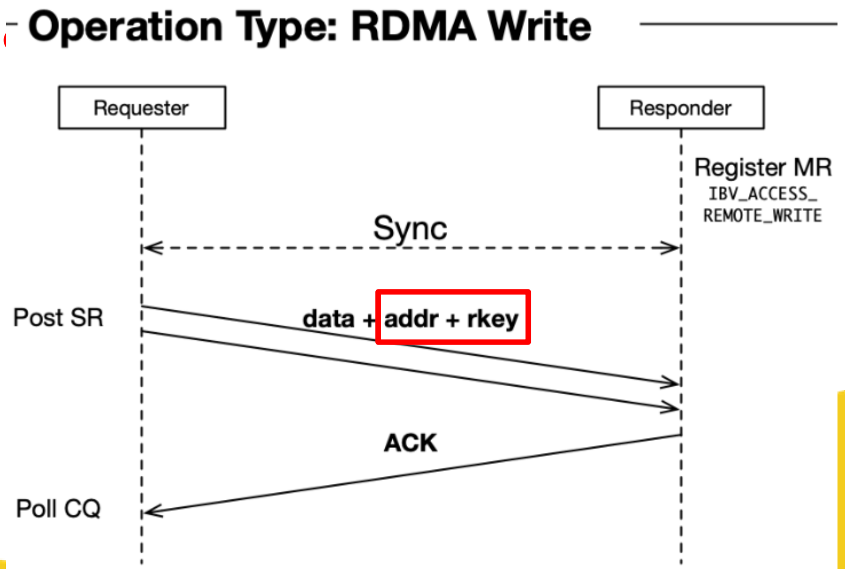
- Separation of Control and Data paths
- Control path
  - Resource setup
  - Memory management
  - Connection establishment
- Data path (only after control path)
  - Post Send, Post Receive
  - Poll for Completion, Request event
  - Connection establishment



RDMA Communication Semantics

- Send / Receive
  - Send / Receive with TAG matching
  - May enhanced by zero-copy
  - Two-sided communication
    - CPUs still involves on both sides
- RDMA Read and Write
  - One-sided communication
    - Only the CPU of reader/writer involves
  - Require the memory address and key on the rem
- Atomic Operations on Remote Memory
  - SWAP, CSWAP, ADD, XOR
- Group Communication directives
  - Reduce, Allreduce, Scatter, Gather, AlltoAll

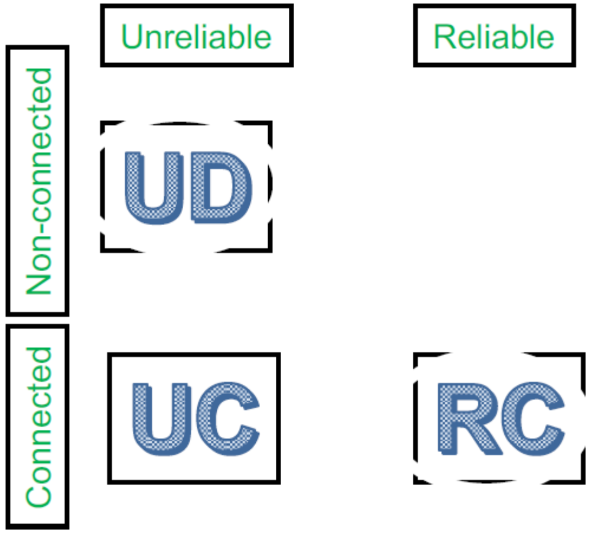




Transport Services

- Reliable Connection (RC):
  - Reliable transport, connection oriented
- Unreliable Datagram (UD):
  - Unreliable transport, not-connected
- Unreliable connection (UC):
  - Unreliable transport, connection oriented

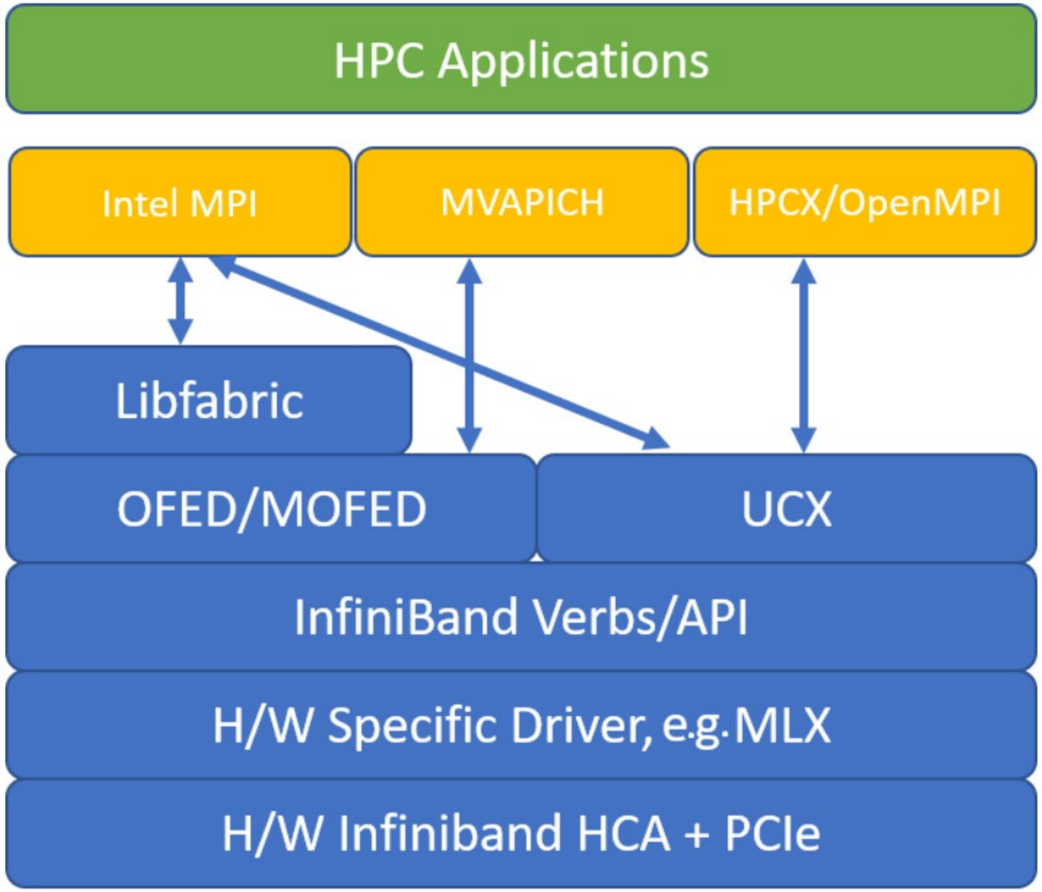
- Reliable
  - exactly once, in-order delivery
- Connected
  - a strong paring of end-nodes
  - Connection establishment is required



	UD	UC	RC
Send / Receive	√	√	√
RDMA Write	X	√	√
RDMA Read / Atomic	X	X	√
Max Send Size	MTU	2GB	2GB
Reliability	X	X	√
Scalability (per-process for N processes)	1	N	N

Outline

- RDMA Technology
- Verbs
- UCX
- HPC-X



RDMA Standard - Verbs

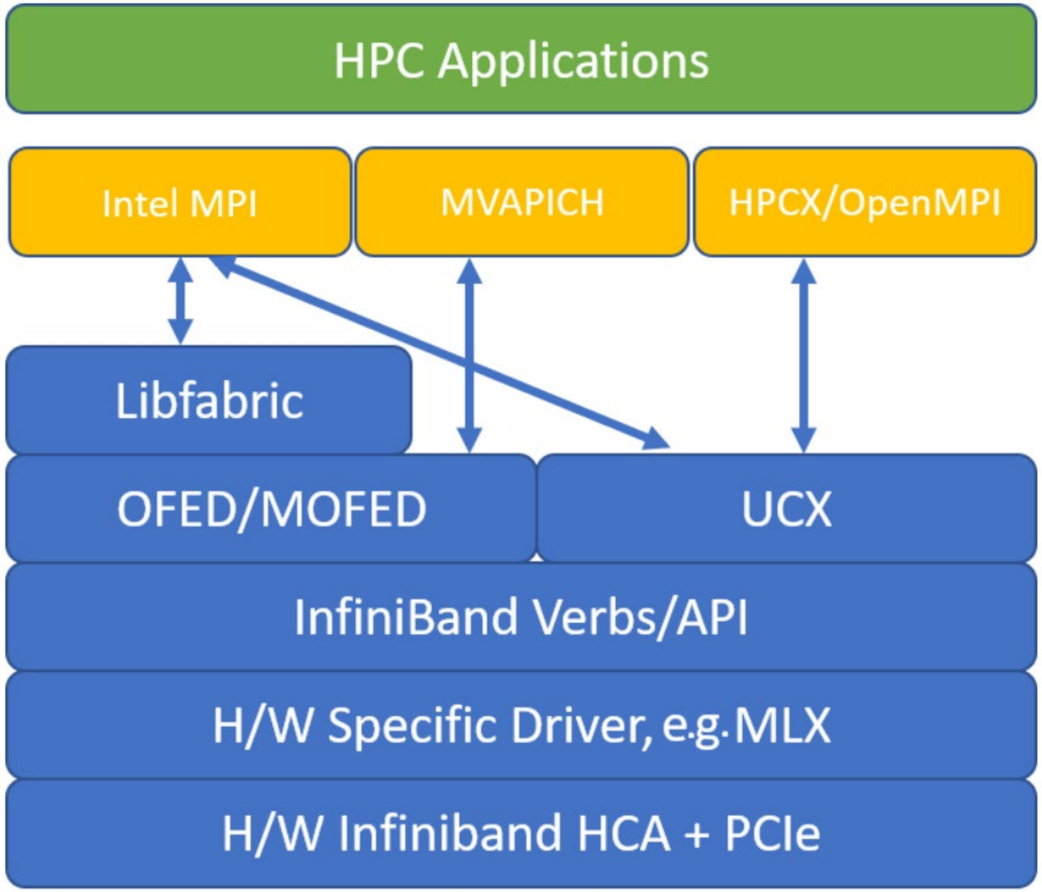
- Verbs is an abstract description of the functionality that is provided for applications for using RDMA.
  - Verbs is not an API
  - There are several implementations for it
- Verbs is a low-level description for RDMA programming
  - Verbs are close to the “bare-metal” and provide best performance
    - Latency, BW, Message rate
  - Verbs can be used as building blocks for many applications
    - Sockets, Storage, Parallel computing
- Any other level of abstraction over verbs may harm the performance

libibverbs

- libibverbs, developed and maintained by Roland Dreier since 2006, are de-facto the verbs API standard in *nix
  - Developed as an Open source, community project
  - The kernel part of the verbs is integrated in the Linux kernel since 2005
  - There are low-level libraries from several HW vendors
- Same API for all RDMA-enabled transport protocols
  - Infiniband Networks
    - Used extensively in HPC machines (Supercomputers)
    - Expensive, requires specialized hardware (physical network and NIC)
  - RoCE: RDMA done over Ethernet instead of Infiniband (RDMA over Converged Ethernet)
    - Still requires specialized hardware
    - Cheaper because only needs specialized NICs
    - RoCE seems to perform worse at scale (Ethernet is lossy)
  - iWARP
    - RDMA over TCP
    - Once again, cheaper; only needs specialized NICs

Outline

- RDMA Technology
- Verbs
- UCX
- HPC-X

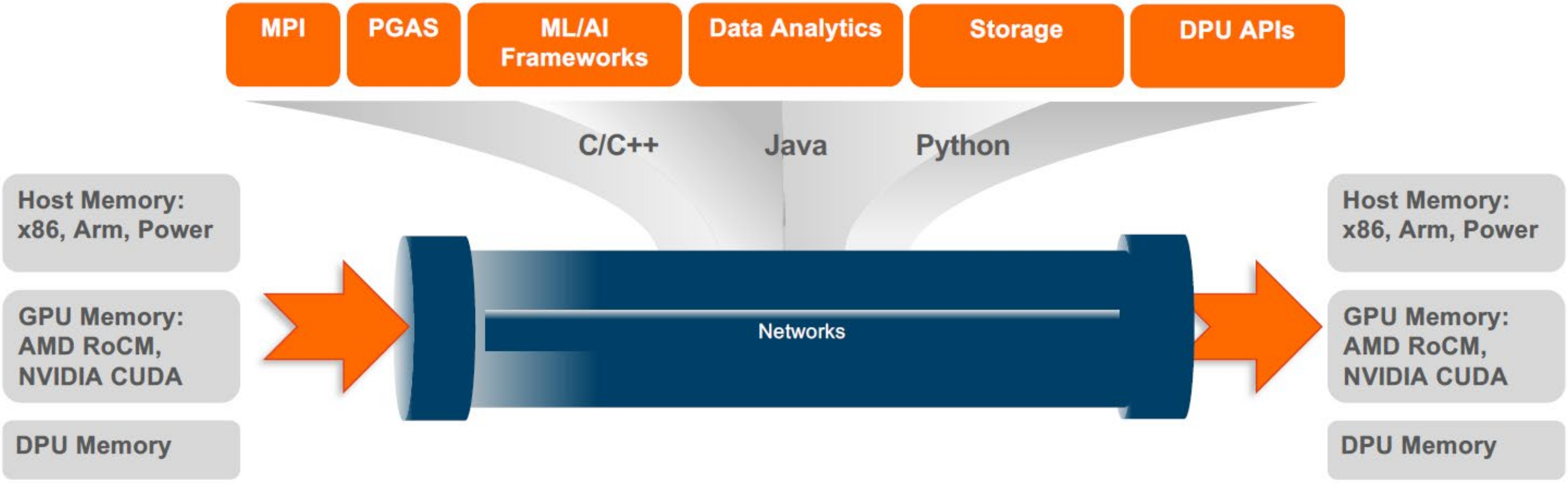


Oscar Hernandez
on behalf of Gilad Shainer/UCF

https://github.com/gt-crnch-rg/ucx-tutorial-hot-interconnects

HOTI 2022: UCX TUTORIAL

What is UCX?


MPI
PGAS
ML/AI Frameworks
Data Analytics
Storage
DPU APIs
C/C++
Java
Python
Host Memory: x86, Arm, Power
GPU Memory: AMD RoCM, NVIDIA CUDA
DPU Memory
Networks
Host Memory: x86, Arm, Power
GPU Memory: AMD RoCM, NVIDIA CUDA
DPU Memory

High-Performance Universal Data Mover

Unified Communication Framework (UCF) Consortium

MISSION: Collaboration between industry, laboratories, and academia to create production grade communication frameworks and open standards for data centric, ML/AI, and high-performance applications

Projects & Working Groups

- **UCX – Unified Communication X – www.openucx.org**
- SparkUCX – www.sparkucx.org
- OpenSNAPI – Smart NIC Project
- UCC – Collective Library
- UCD – Advanced Datatype Engine
- HPCA Benchmark – Benchmarking Effort

Board members

- **Jeff Kuehn**, UCF Chairman (AMD)
- **Gilad Shainer**, UCF President (NVIDIA)
- **Pavel Shamis**, UCF Treasurer (Arm)
- **Yanfei Guo**, Board Member (Argonne National Laboratory)
- **Perry Schmidt**, Board Member (IBM)
- **Dhabaleswar K. (DK) Panda**, Board Member (Ohio State University)
- **Steve Poole**, Board Member (Open Source Software Solutions)

UCF Unified Communication Framework

NVIDIA. Los Alamos NATIONAL LABORATORY EST. 1943 AMD THE OHIO STATE UNIVERSITY Argonne NATIONAL LABORATORY IBM arm

Join https://www.ucfconsortium.org or info@ucfconsortium.org

Unified Communication X (UCX)

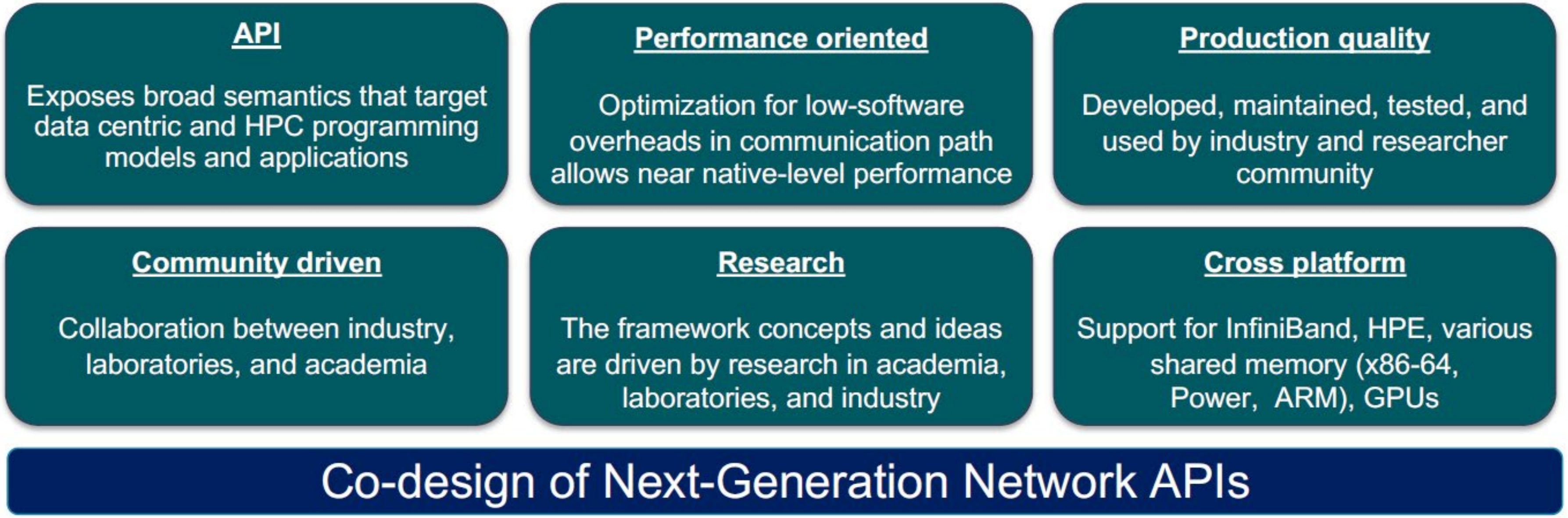
https://www.hpcwire.com/2018/09/17/ucf-ucx-and-a-car-ride-on-the-road-to-exascale/

UCX Useful Links

- Code
  - https://github.com/openucx/
- Website
  - www.openucx.com
- Mailing list
  - https://elist.ornl.gov/mailman/listinfo/ucx-group
- Contributor agreement
  - https://www.openucx.org/license/
- User documentation
  - https://openucx.readthedocs.io/

UCX Framework Mission

- Collaboration between industry, laboratories, and academia
- Create open-source production grade communication framework for HPC applications
- Enable the highest performance through co-design of software-hardware interfaces
- Unify industry - national laboratories - academia efforts



API

Exposes broad semantics that target data centric and HPC programming models and applications

Performance oriented

Optimization for low-software overheads in communication path allows near native-level performance

Production quality

Developed, maintained, tested, and used by industry and researcher community

Community driven

Collaboration between industry, laboratories, and academia

Research

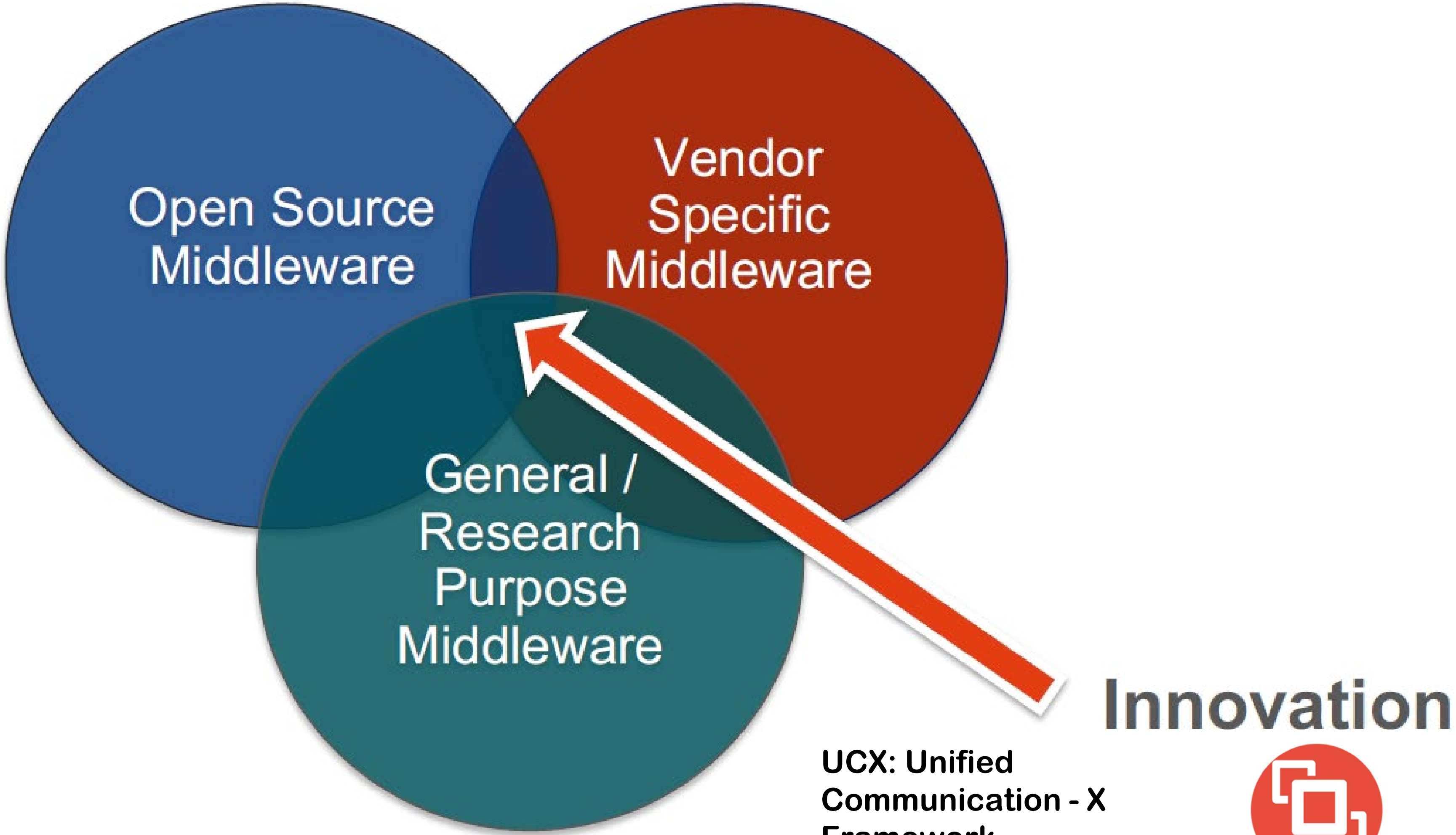
The framework concepts and ideas are driven by research in academia, laboratories, and industry

Cross platform

Support for InfiniBand, HPE, various shared memory (x86-64, Power, ARM), GPUs

Co-design of Next-Generation Network APIs

Network Programming Interfaces


Open Source Middleware
Vendor Specific Middleware
General / Research Purpose Middleware
Innovation

UCX: Unified Communication - X Framework

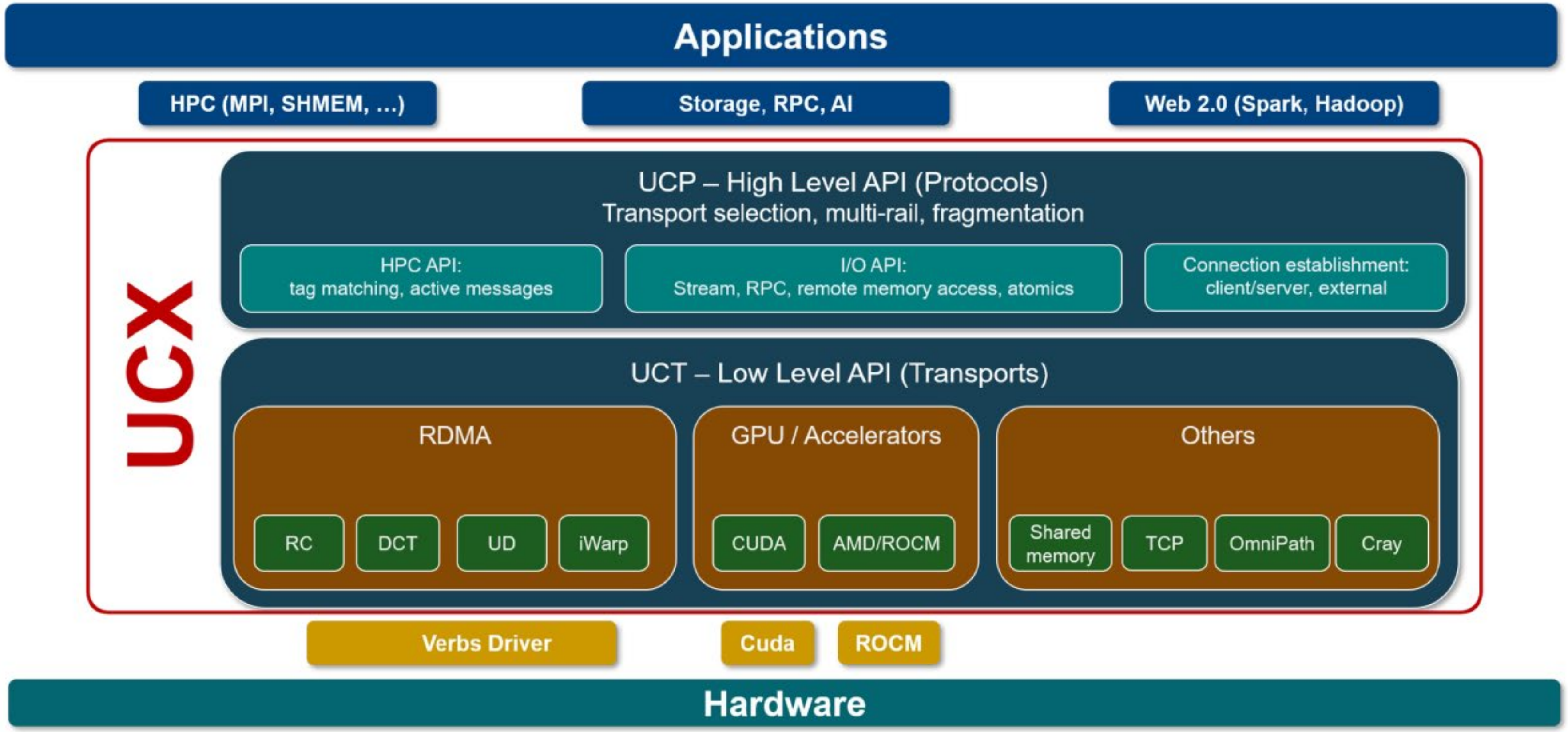
Network Programming Interfaces

	Pros	Cons
Vendor-Specific APIs	▪ Production Quality ▪ Optimized for Performance ▪ Support and maintenance	▪ Often “vendor” locked ▪ Optimized for a particular technology ▪ Co-design lags behind
Open-Source APIs	▪ Community (a.k.a. user) driven ▪ Easy to modify and extend ▪ Good for research	▪ Typically, not as optimized as commercial/vendor software ▪Maintenance is challenge
Research API	▪ Innovative and forward looking • A lot of good ideas for “free”	▪ Support, support, support ▪ Typically, narrow focus

What’s innovative about UCX?

- Simple, consistent, performance portable unified API
- Choosing between low-level and high-level API allows easy integration with a wide range of applications and middleware.
- Protocols and transports are selected by capabilities and performance estimations, rather than hard-coded definitions.
- Support thread contexts and dedicated resources, as well as fine-grained and coarse-grained locking.
- Accelerators are represented as a transport, driven by a generic “glue” layer, which will work with all communication networks.

UCX High-level Overview



Applications

HPC (MPI, SHMEM, …)

Storage, RPC, AI

Web 2.0 (Spark, Hadoop)

UCX

UCP – High Level API (Protocols)
Transport selection, multi-rail, fragmentation

HPC API:
tag matching, active messages

I/O API:
Stream, RPC, remote memory access, atomics

Connection establishment:
client/server, external

UCT – Low Level API (Transports)

RDMA: RC, DCT, UD, iWarp

GPU / Accelerators: CUDA, AMD/ROCM

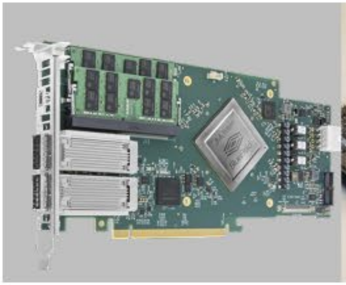
Others: Shared memory, TCP, OmniPath, Cray

Verbs Driver

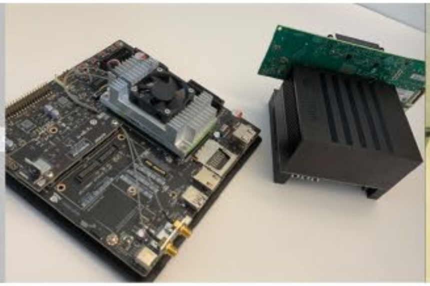
Cuda

ROCM

Hardware



BlueField DPU



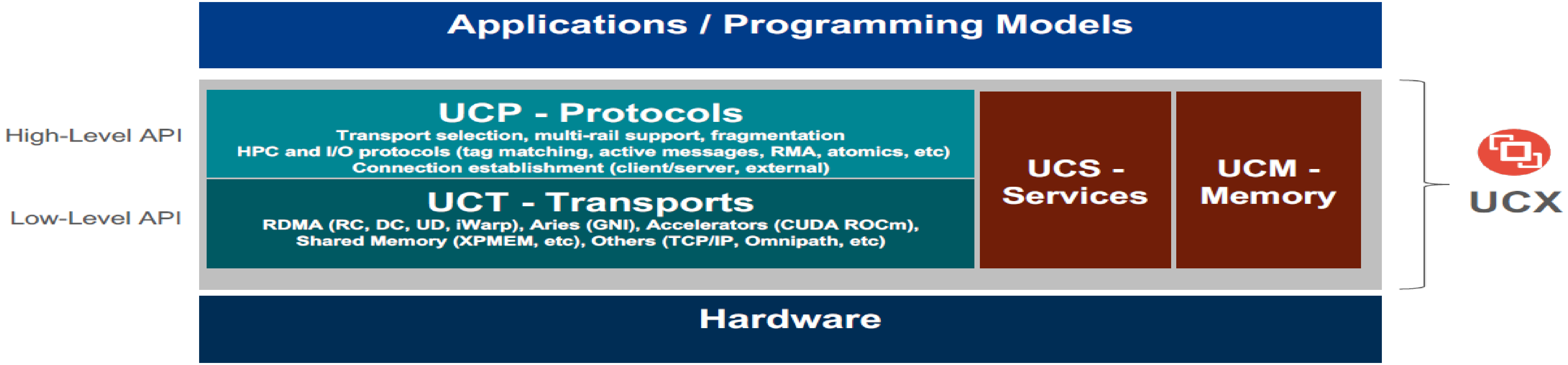
NVIDIA Jetson

Arm ThunderX2

Odroid C2

N1 SDP

UCX Framework



Applications / Programming Models

High-Level API

UCP - Protocols
Transport selection, multi-rail support, fragmentation
HPC and I/O protocols (tag matching, active messages, RMA, atomics, etc)
Connection establishment (client/server, external)

Low-Level API

UCT - Transports
RDMA (RC, DC, UD, iWarp), Aries (GNI), Accelerators (CUDA ROCm),
Shared Memory (XPMEM, etc), Others (TCP/IP, Omnipath, etc)

UCS - Services

UCM - Memory

UCX

Hardware

UC-P for Protocols

High-level API uses UCT framework to construct protocols commonly found in applications

Functionality:
Multi-rail, device selection, pending queue, rendezvous, tag-matching, software-atomics, etc.

UC-T for Transport

Low-level API that expose basic network operations supported by underlying hardware. Reliable, out-of-order delivery.

Functionality:
Setup and instantiation of communication operations.

UC-S for Services

This framework provides basic infrastructure for component-based programming, data structure support, and useful system utilities

Functionality:
Platform abstractions, data structures support, debug facilities.

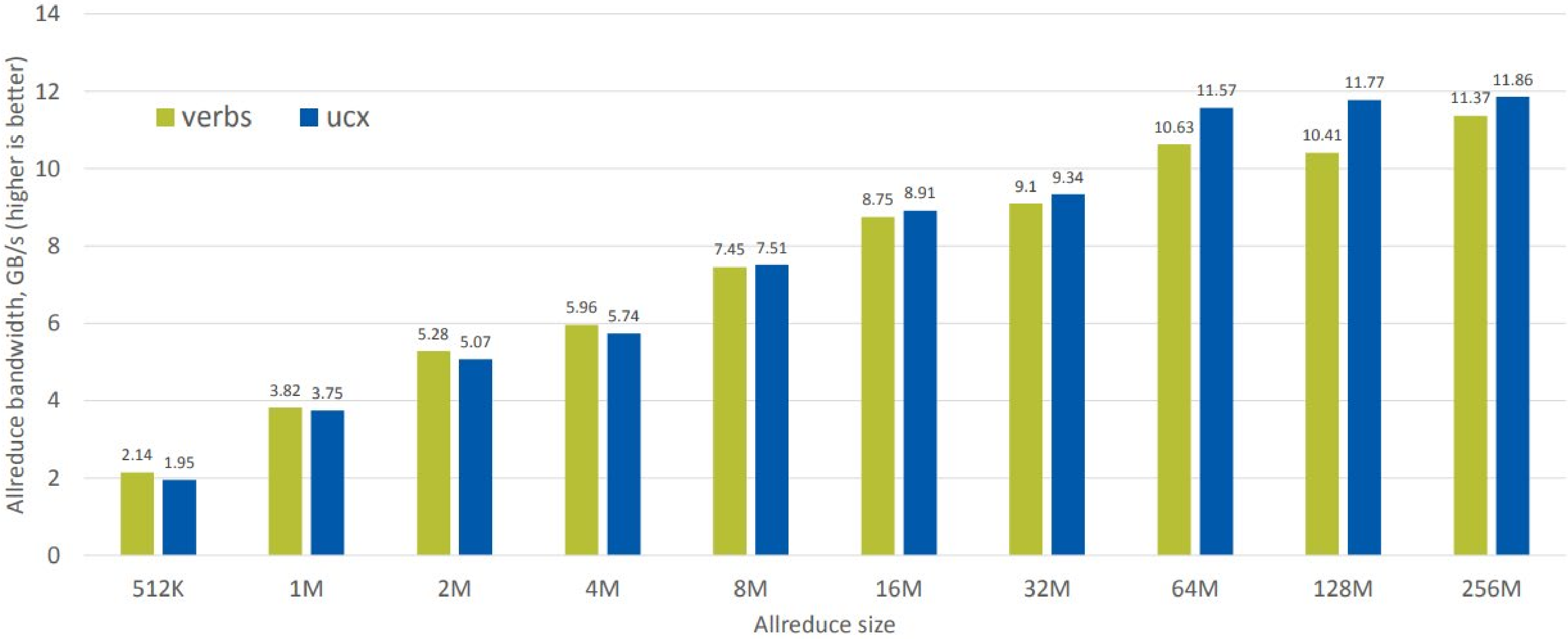
UC-M for Memory

This framework provides infrastructure for getting notifications about memory allocate and release events

Functionality:
Platform for memory allocations/dealloc. notifications across devices

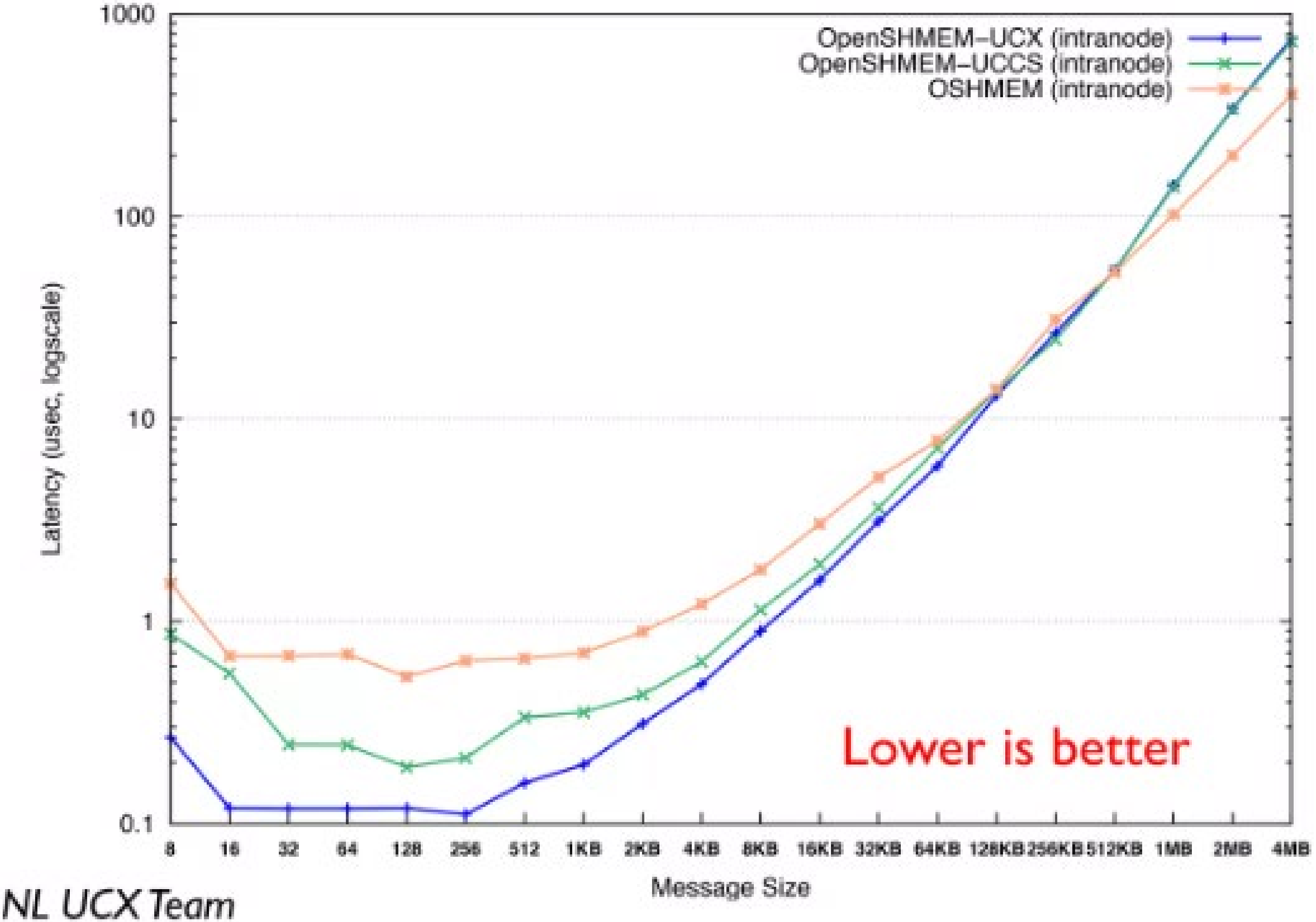
NCCL Internal Verbs vs NCCL UCX Plugin

- UCX plugin outperforms NCCL Verbs implementation up to 13% on large messages



NAR Labs 財團法人國家實驗研究院
國家高速網路與計算中心
National Center for High-performance Computing

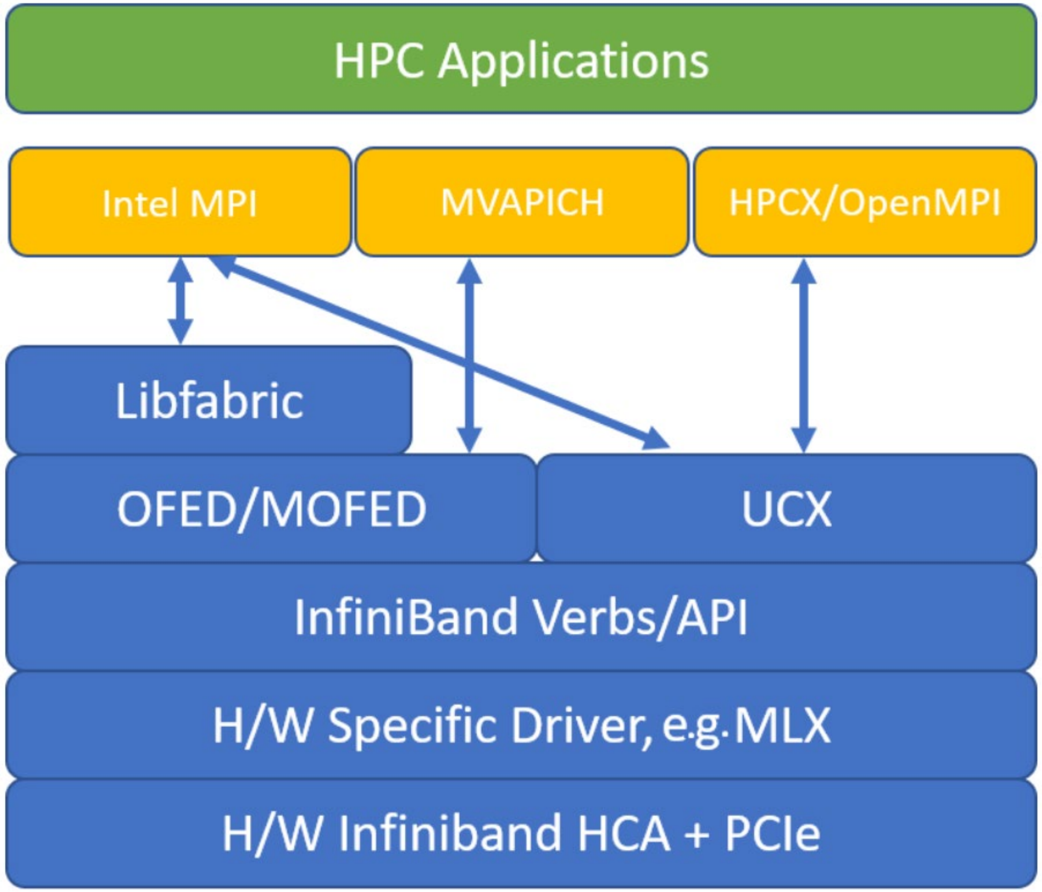
國立清華大學叢集電腦競賽團隊
Student Cluster Competition Team of NTHU



Slide courtesy of ORNL UCX Team

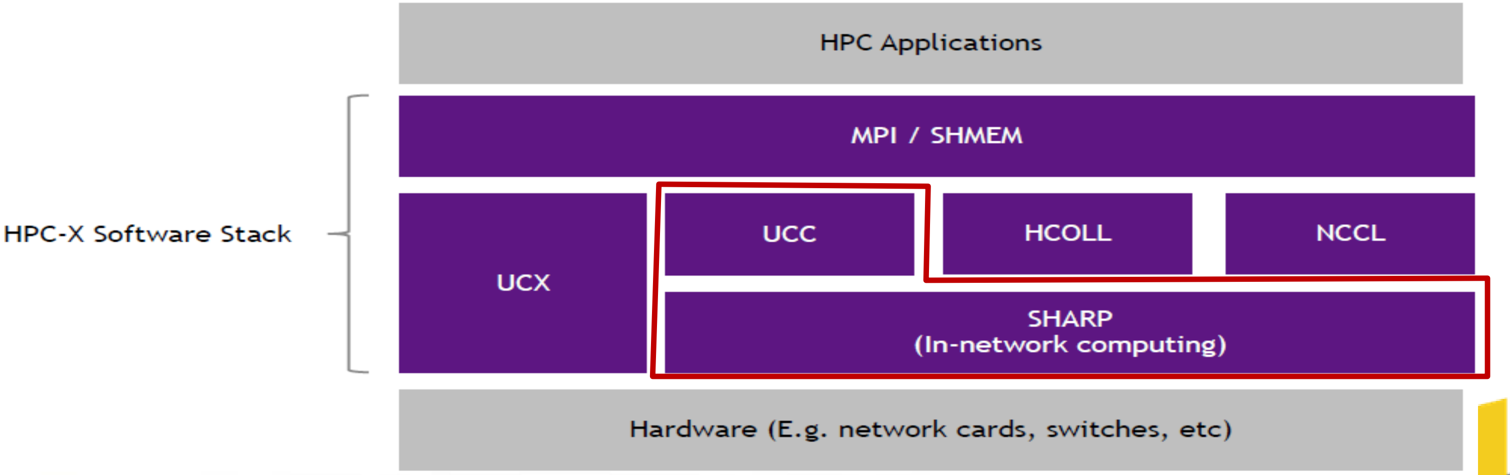
Outline

- RDMA Technology
- Verbs
- UCX
- HPC-X

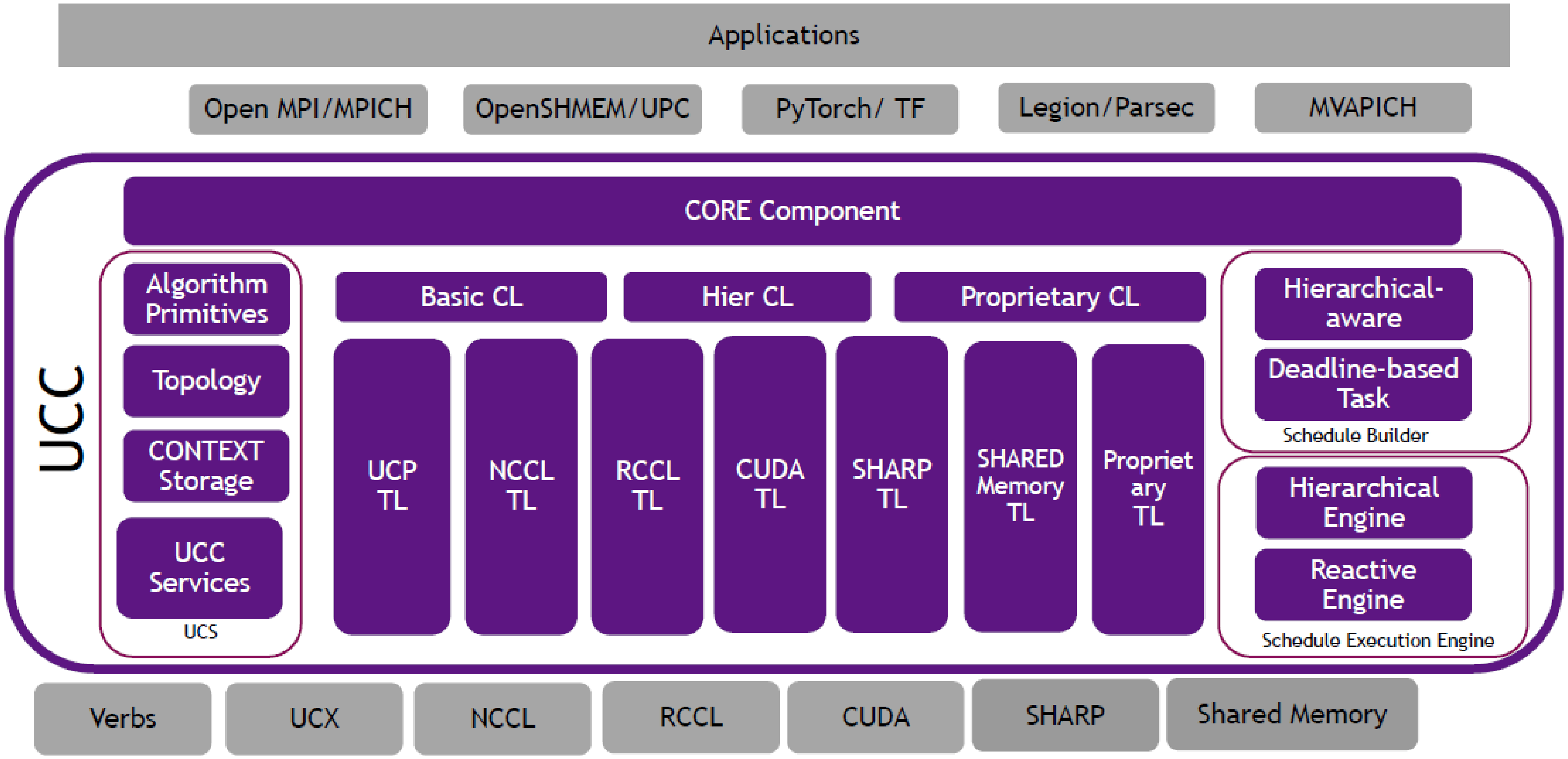


HPC-X Software Stack

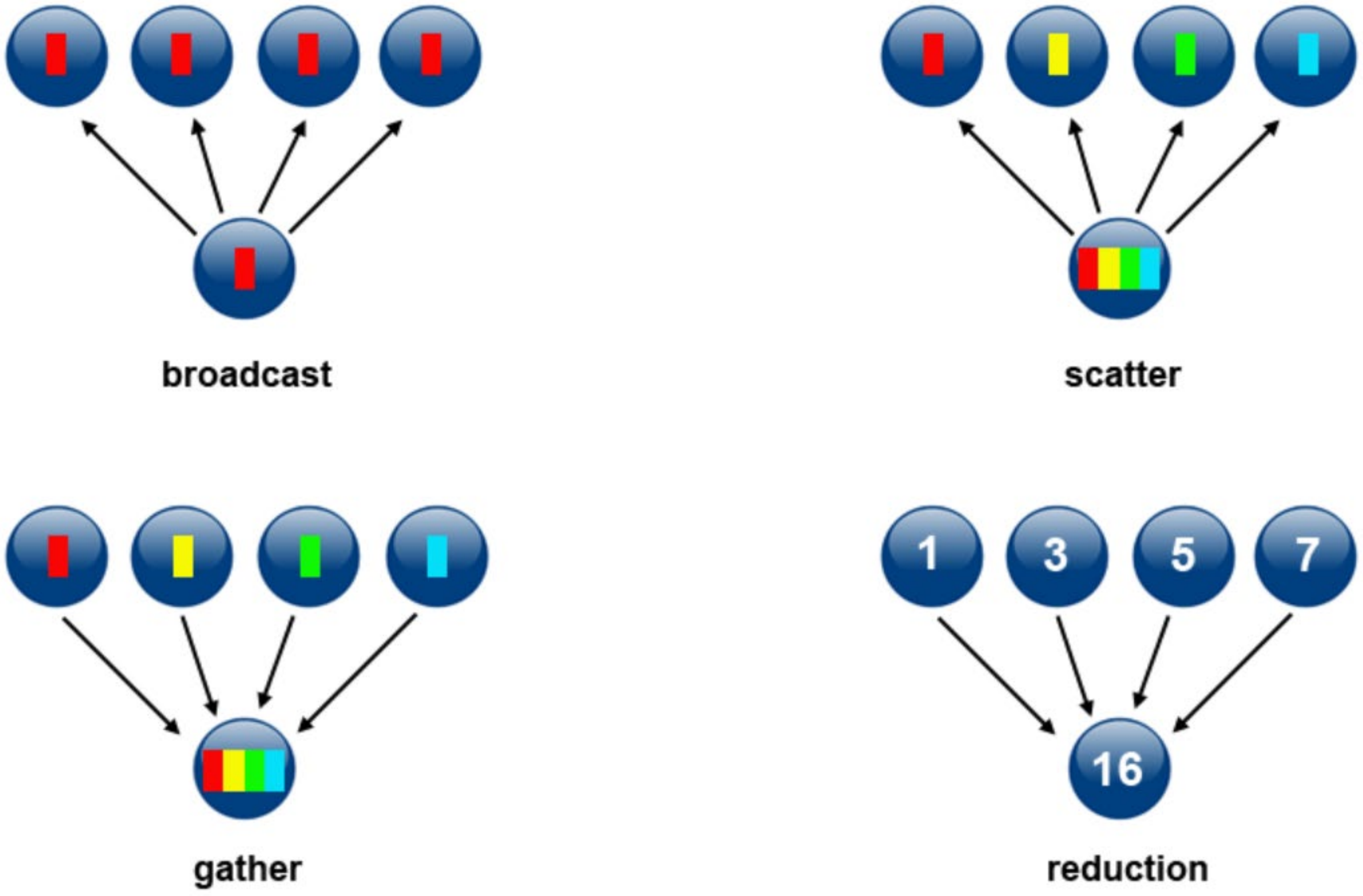
- HPC-X is the Mellanox solution for HPC communication libraries to improve the performance and scalability of your HPC cluster
  - MPI /SHMEM implementation
  - UCX –Unified Communication X
  - UCC –Unified Collective Communication
  - HCOLL –Hierarchical Collectives (Note: UCC will replace this in the future)
  - NCCL/SHARP hardware collectives
  - In-network computing infrastructure with SHARP



UNIFIED COLLECTIVE COMMUNICATION (UCC) Architecture



Collective Communication Routines



Types of Collective Operations:

Open MPI with UCX & UCC

- https://github.com/openucx/ucc
  - Compile UCX
  - Compile UCC
  - Compile Open MPI

```
$ git clone https://github.com/open-mpi/ompi
$ cd ompi
$ ./autogen.pl; ./configure --prefix=<ompi-install-path> --with-ucx=<ucx-install-path>
--with-ucc=<ucc-install-path>; make -j install
```

  - Run MPI programs

```
$ mpirun -np 2 --mca coll_ucc_enable 1 --mca coll_ucc_priority 100 ./my_mpi_app
```

  - SUPPORTED Transports
    - UCX/UCP: InfiniBand, ROCE, Cray Gemini and Aries, Shared Memory
    - SHARP、CUDA、NCCL、RCCL

Reference

- https://github.com/gt-crnch-rg/ucx-tutorial-hot-interconnects
- https://mug.mvapich.cse.ohio-state.edu/static/media/mug/presentations/21/gorentla_bureddy_ucc_sharp_mug21.pdf
- https://openucx.github.io/ucc/
- https://ucfconsortium.org/wp-content/uploads/2020/02/Manjunath_GV_gorentla_ucx_collectives.pdf
- https://mug.mvapich.cse.ohio-state.edu/static/media/mug/presentations/20/bureddy-mug-20.pdf