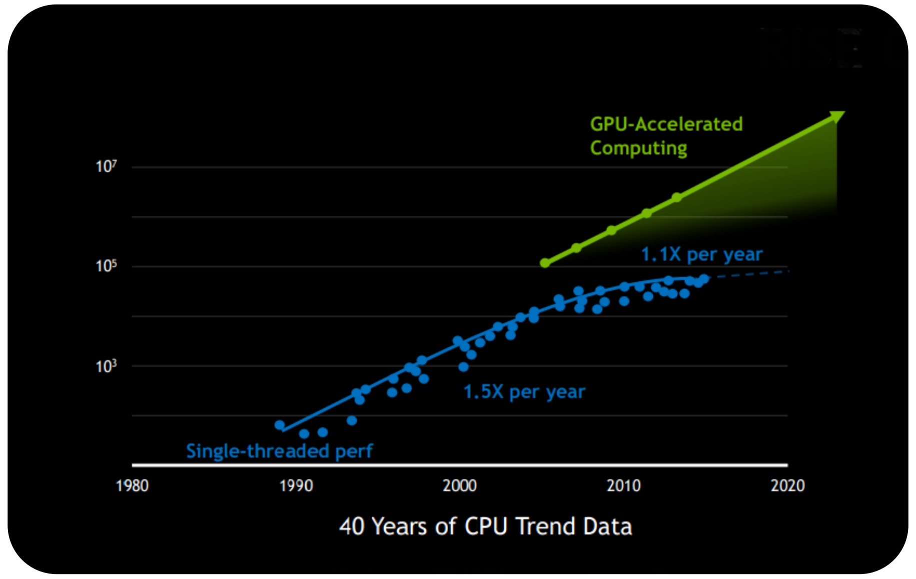
SHARP: IN-NETWORK SCALABLE STREAMING HIERARCHICAL AGGREGATION AND REDUCTION PROTOCOL

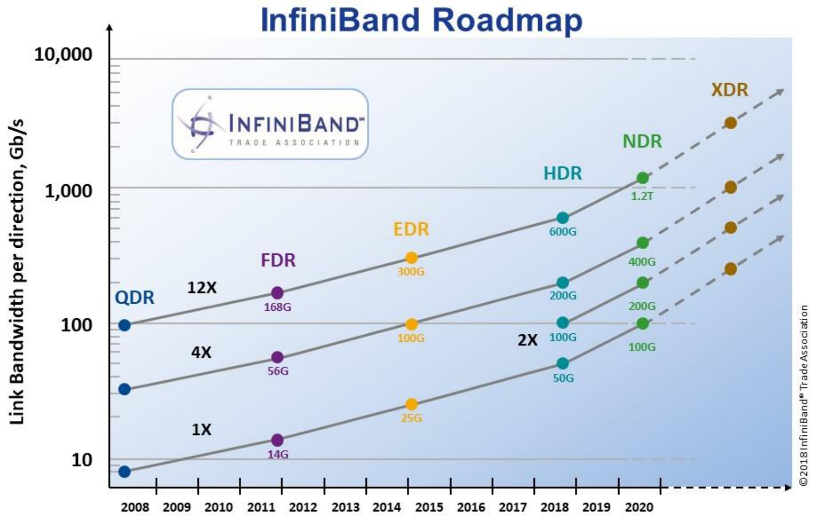
Devendar Bureddy, Aug 2020

IN NETWORK COMPUTING

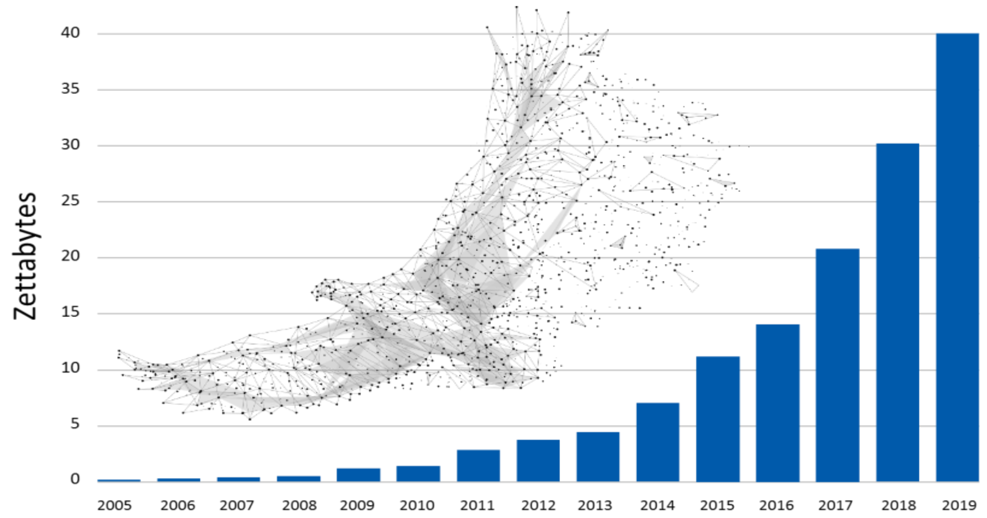
Datacenter Trends



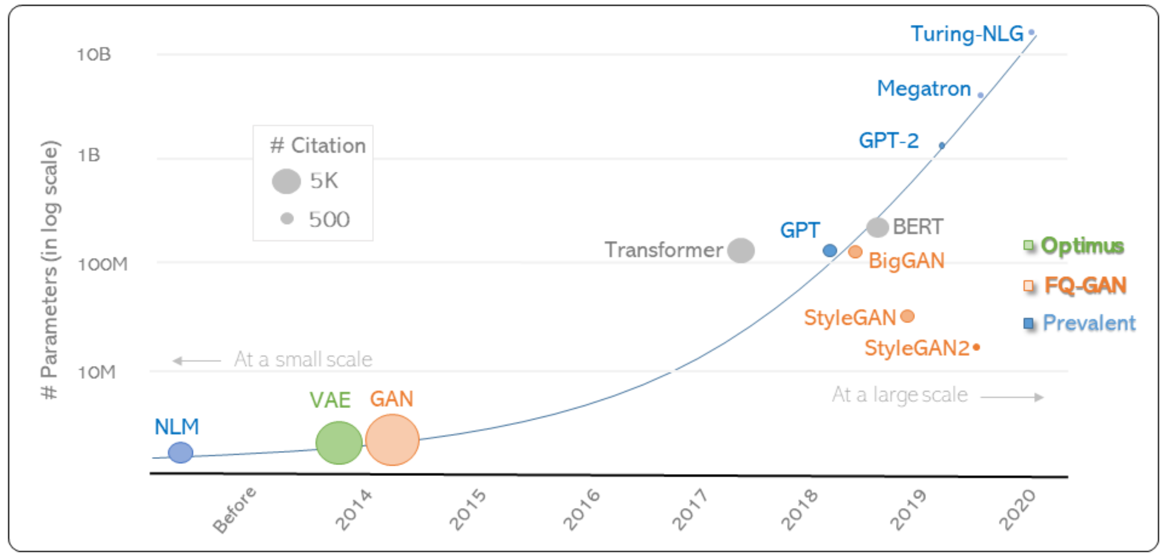
Moore's Law



Networking Moore's Law



Exponential Data Growth



Better DL Model

Efficient Interconnect

IN NETWORK COMPUTING

Offload, Co-design and In-network Computing

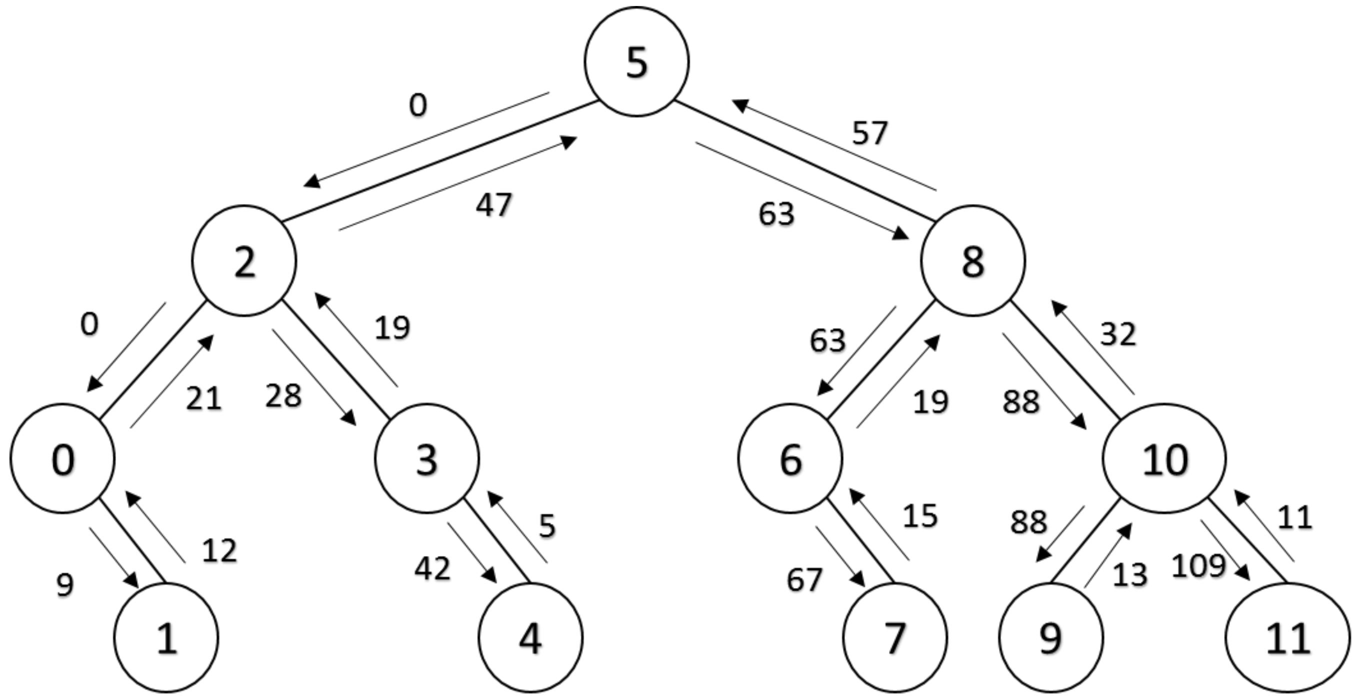
- Offload – Have someone else do the work
  - Move functionality from the CPU to the network
- Co-Design – Re-thinking the boundaries between different components
  - Move functionality from SW to HW / end node to switches
- In-Network Computing – Move traditionally compute operations to the network
  - A type of Co-Design

CORE DIRECT

CORE DIRECT

Cross-Channel

- Offload complex communication patterns
  - Send to multiple destinations
  - Receive from multiple sources
  - Dependencies between the operations
- Define communication graph
  - Two new operations (WQE Opcodes):
  - WAIT / ENABLE
- HW Execution (including progress)



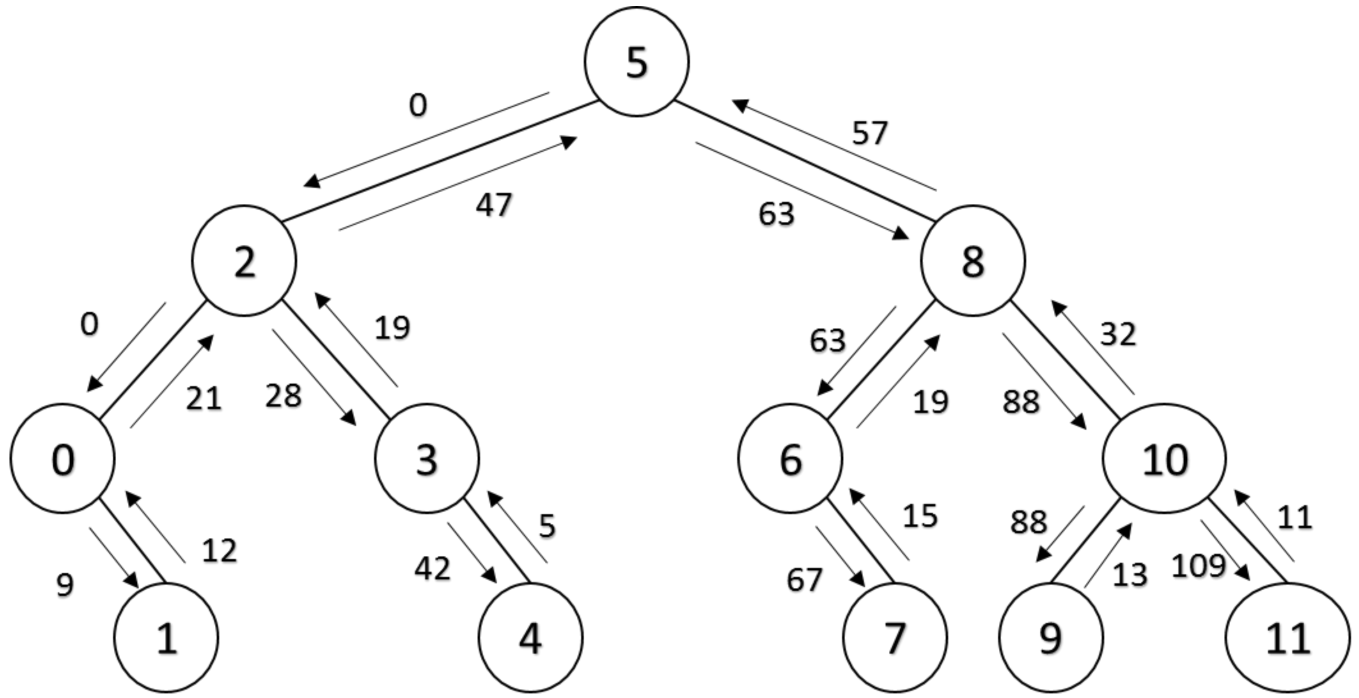
Rank	0	1	2	3	4	5	6	7	8	9	10	11
n	9	12	7	14	5	16	4	15	6	13	8	11
S	9	21	28	42	47	63	67	82	88	101	109	120

CORE DIRECT

Cross-Channel

- Example: Tree based reduce algorithm
  - Focus on Rank=2
- Two main flows:

1. Receive data from Ranks: 0, 3
   Calculate reduction $\sum D_{0,3,2}$
   Send to Rank 5
2. Receive data from Rank=5
   Forward data down the tree



Rank	0	1	2	3	4	5	6	7	8	9	10	11
n	9	12	7	14	5	16	4	15	6	13	8	11
S	9	21	28	42	47	63	67	82	88	101	109	120

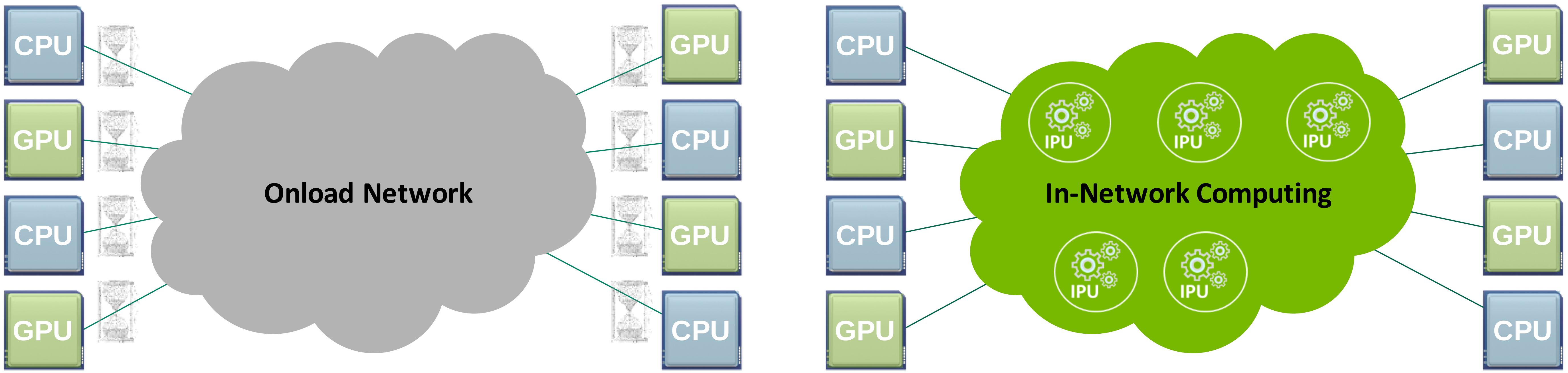
SCALABLE HIERARCHICAL AGGREGATION AND REDUCTION PROTOCOL (SHARP)

THE NEED FOR INTELLIGENT AND FASTER INTERCONNECT

Faster Data Speeds and In-Network Computing Enable Higher Performance and Scale

CPU-Centric (Onload)

Data-Centric (Offload)



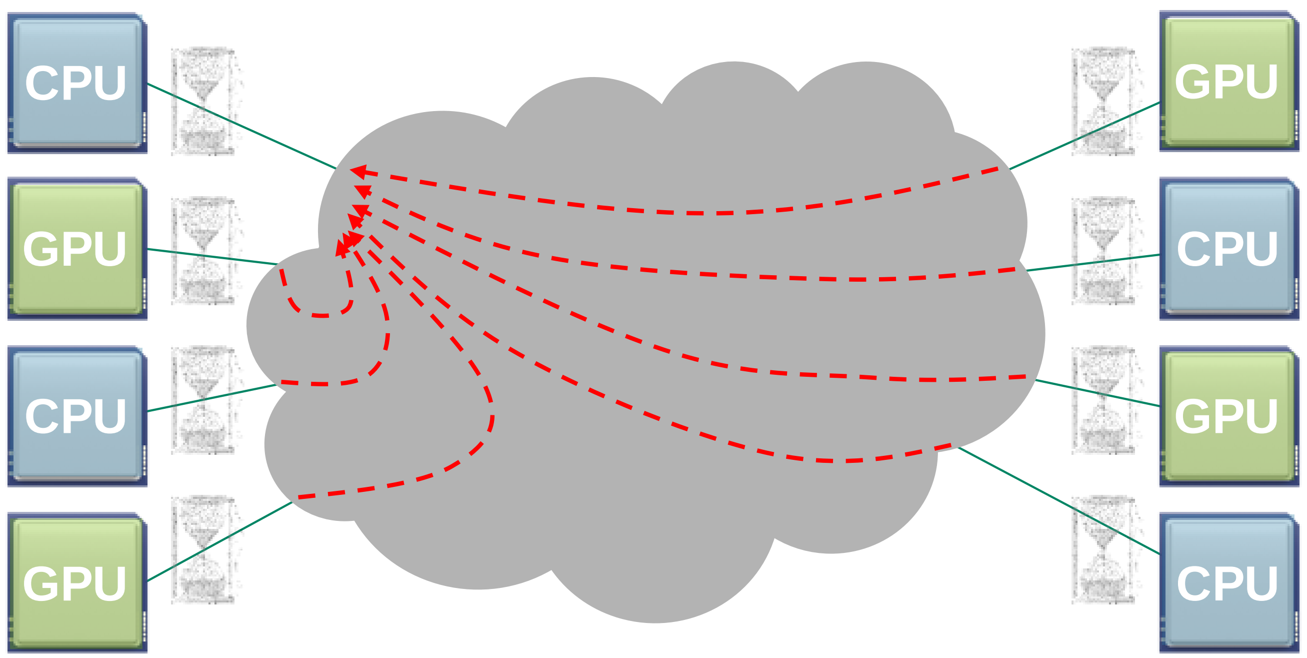
Must Wait for the Data
Creates Performance Bottlenecks

Analyze Data as it Moves!
Higher Performance and Scale

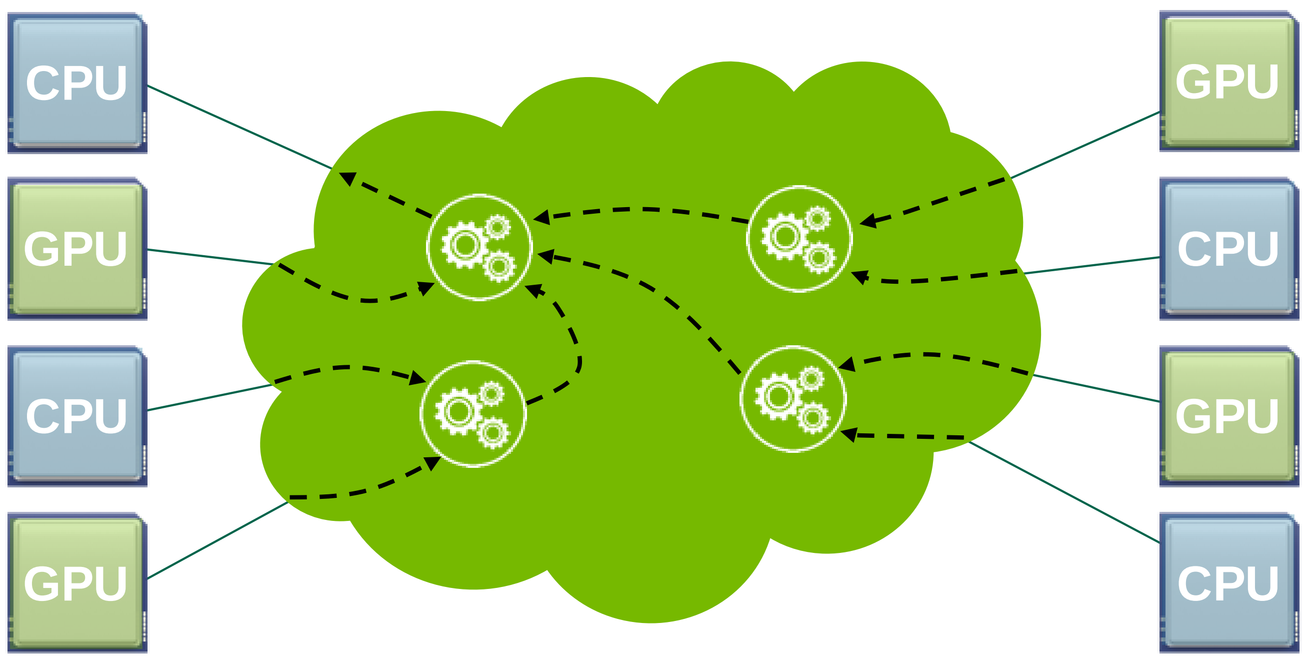
THE NEED FOR INTELLIGENT AND FASTER INTERCONNECT

Faster Data Speeds and In-Network Computing Enable Higher Performance and Scale

CPU-Centric (Onload)



Data-Centric (Offload)



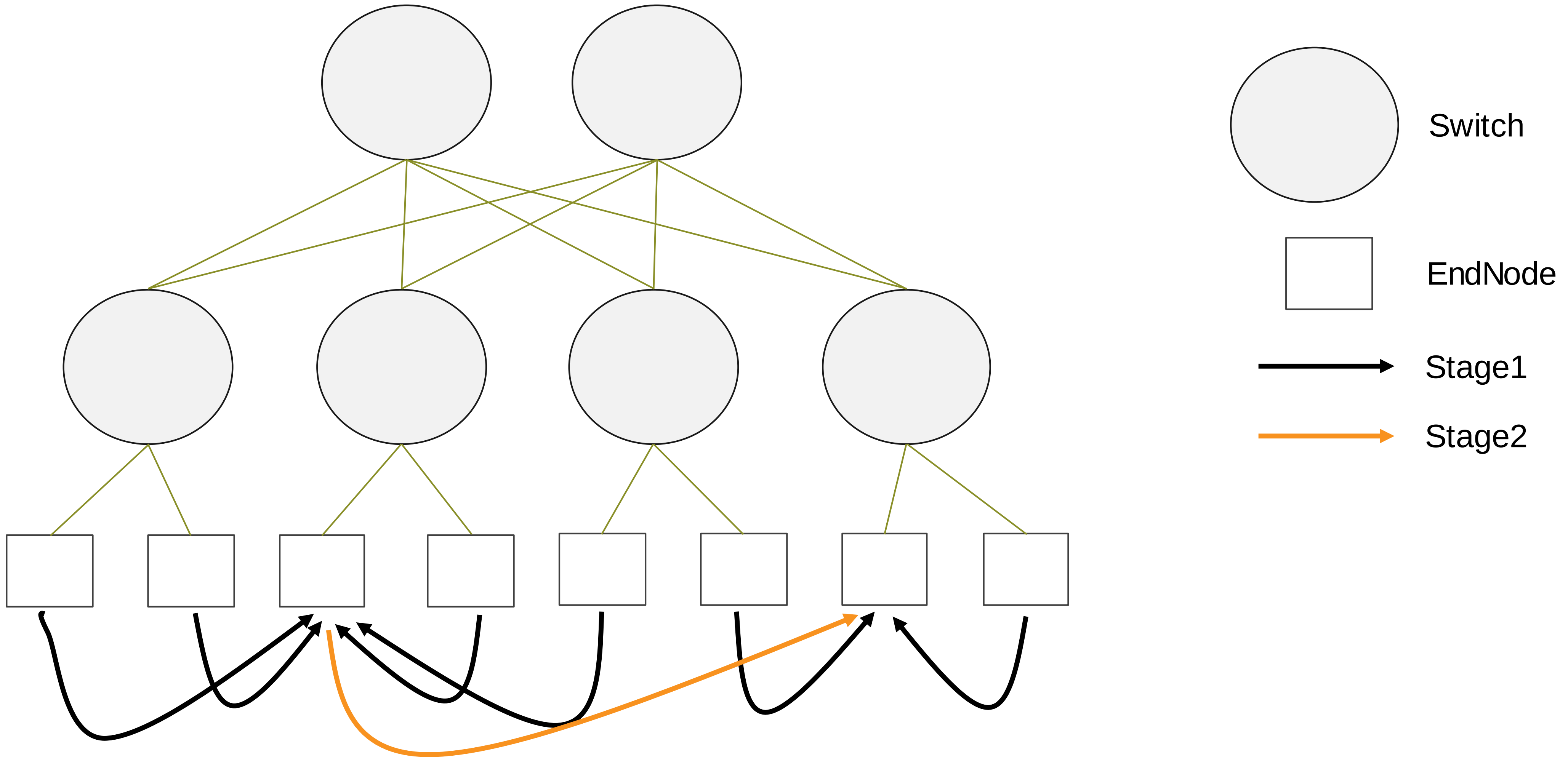
Communications Latencies
of 30-40us

Communications Latencies
of 3-4us

COLLECTIVE OPERATIONS

Tress

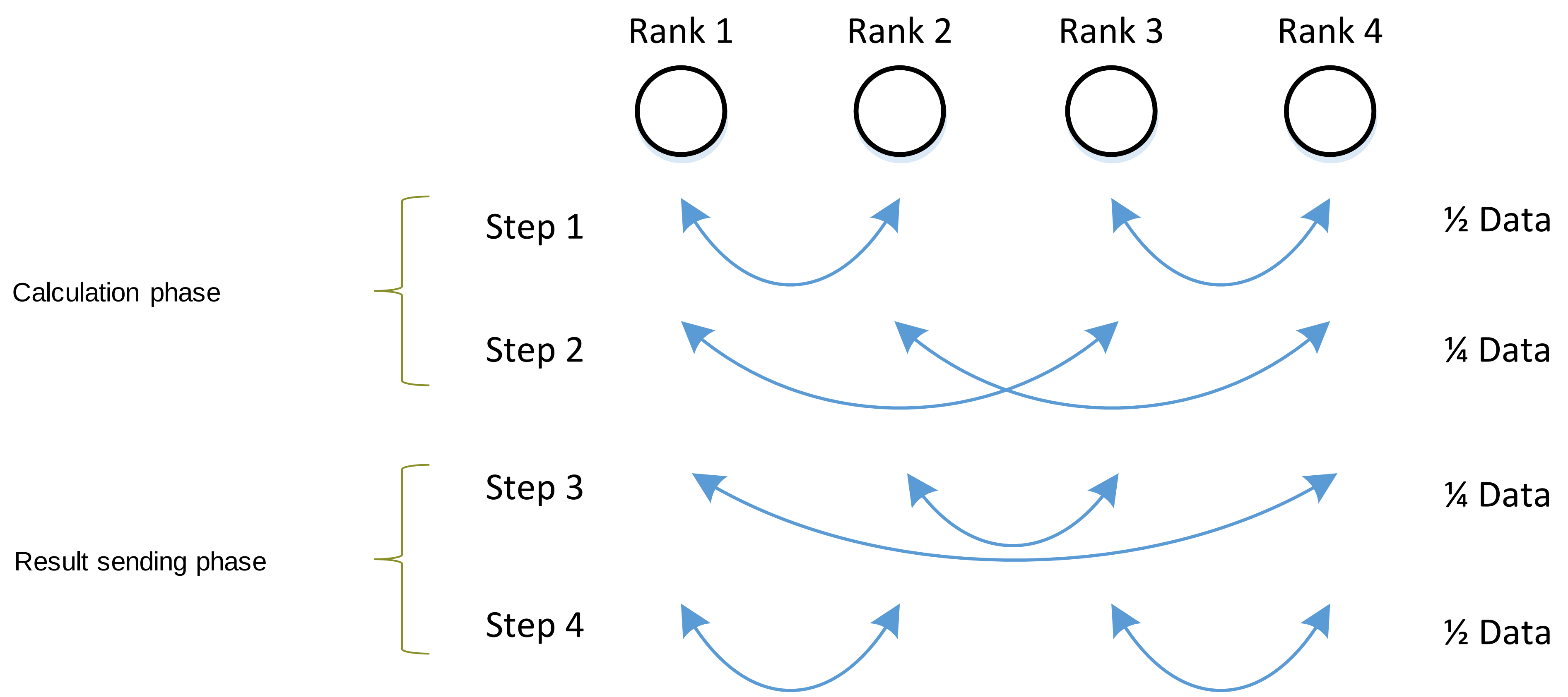
- Many2One and One2Many traffic patterns – possible network congestion
- Probably not a good solution for large data
- Large scale requires higher tree / larger radix
- Result distribution – over the tree / MC



COLLECTIVE OPERATIONS

Recursive Doubling

- Many2One and One2Many traffic patterns – possible network congestion
- Probably not a good solution for large data
- Large scale requires higher tree / larger radix
- Result distribution – over the tree / MC



SCALABLE HIERARCHICAL AGGREGATION AND REDUCTION PROTOCOL (SHARP)

In-network Tree based aggregation mechanism

Multiple simultaneous outstanding operations

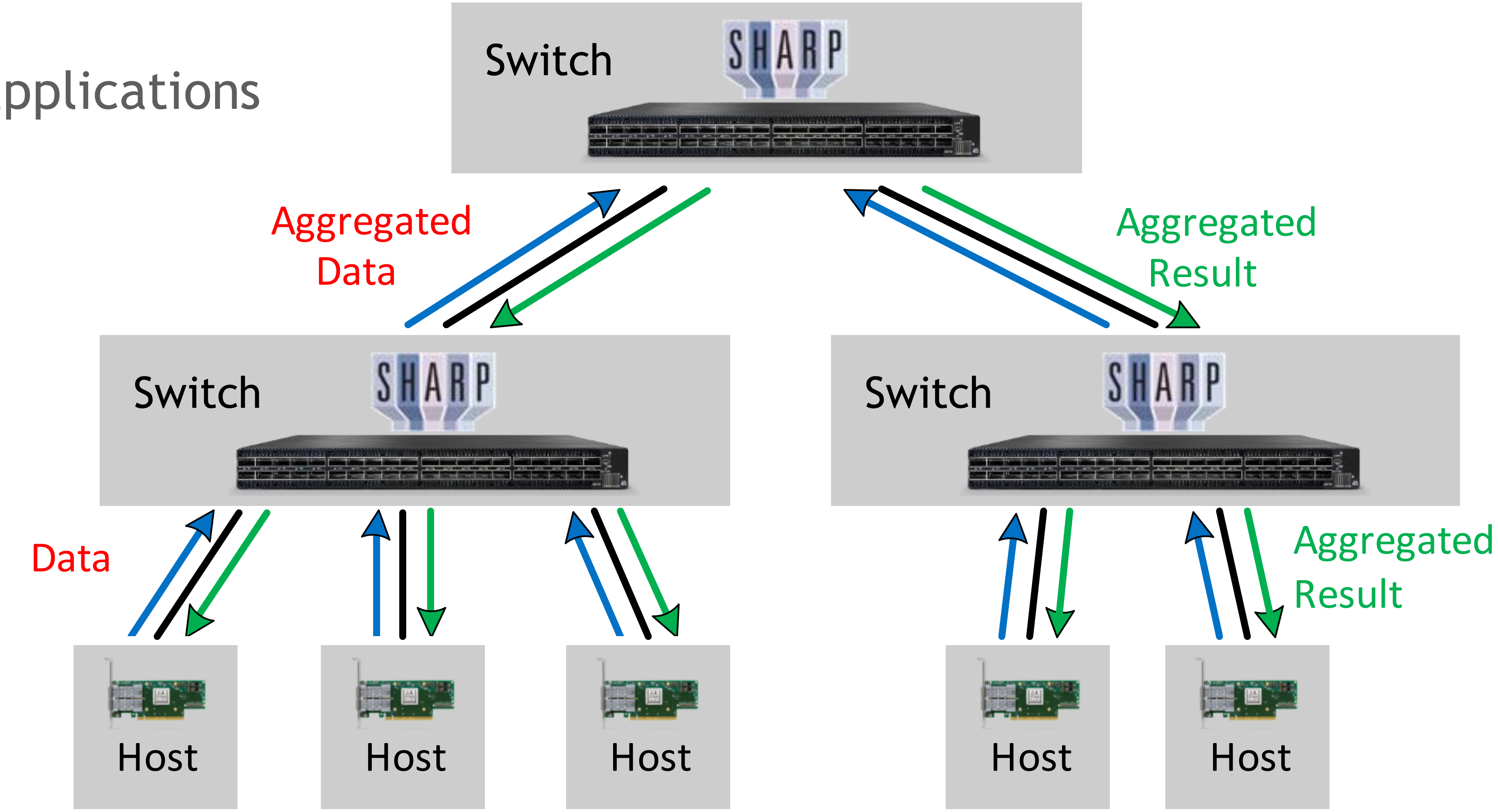
For HPC (MPI / SHMEM) and Distributed Machine Learning applications

Scalable High Performance Collective Offload

Barrier, Reduce, All-Reduce, Broadcast and more

Sum, Min, Max, Min-loc, max-loc, OR, XOR, AND

Integer and Floating-Point, 16/32/64 bits



SHARP

Switch hardware-based network- level reduction supporting for the full range of message sizes

EDR InfiniBand Switch-IB-2 switch introduced support for short message reductions

- Referred to as Low Latency Transmission (LLT) SHARP
- Latency optimized, fully offloaded to the switches – asynchronous

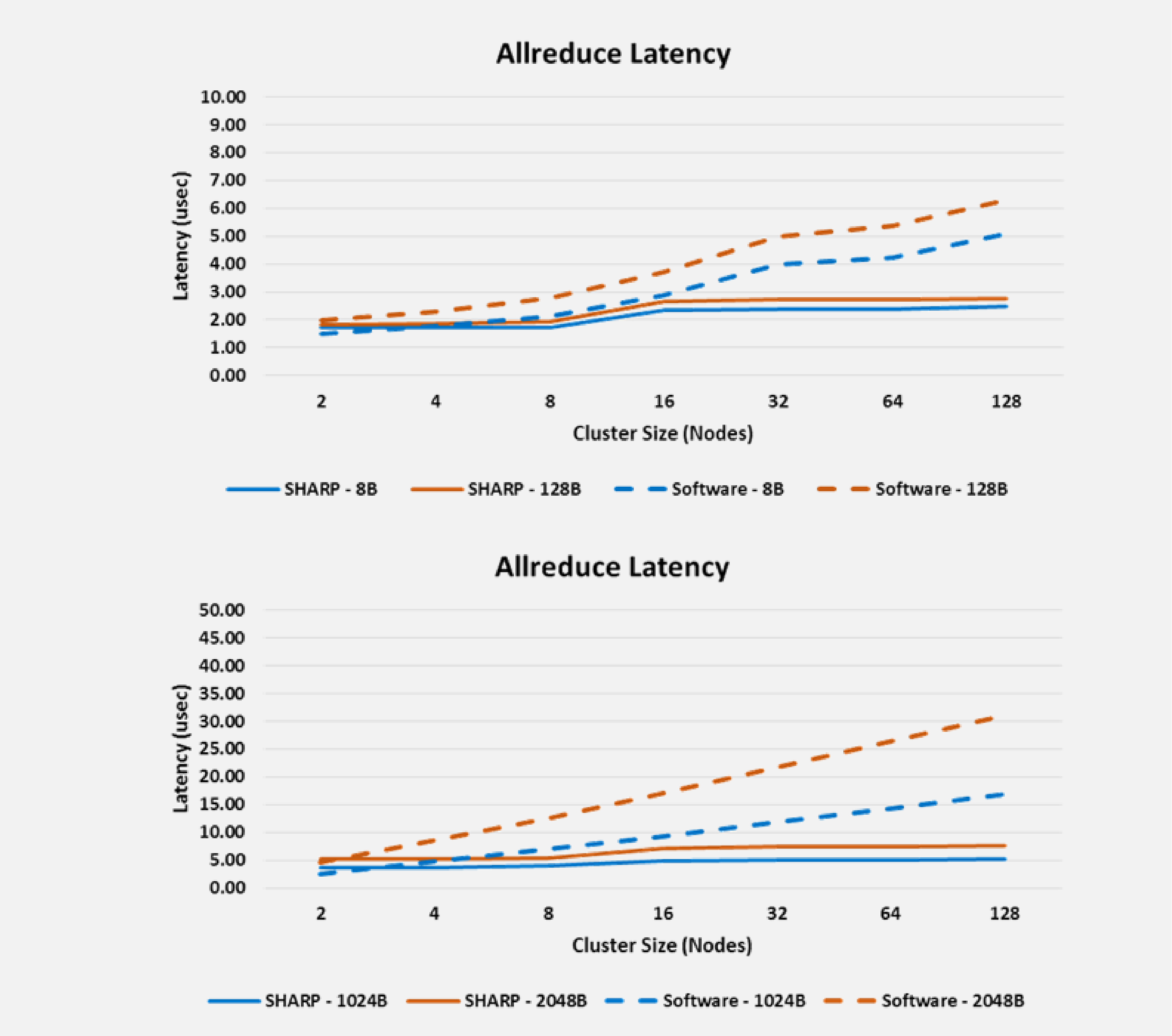
SHARP
Scalable Hierarchical Aggregation and Reduction Protocol

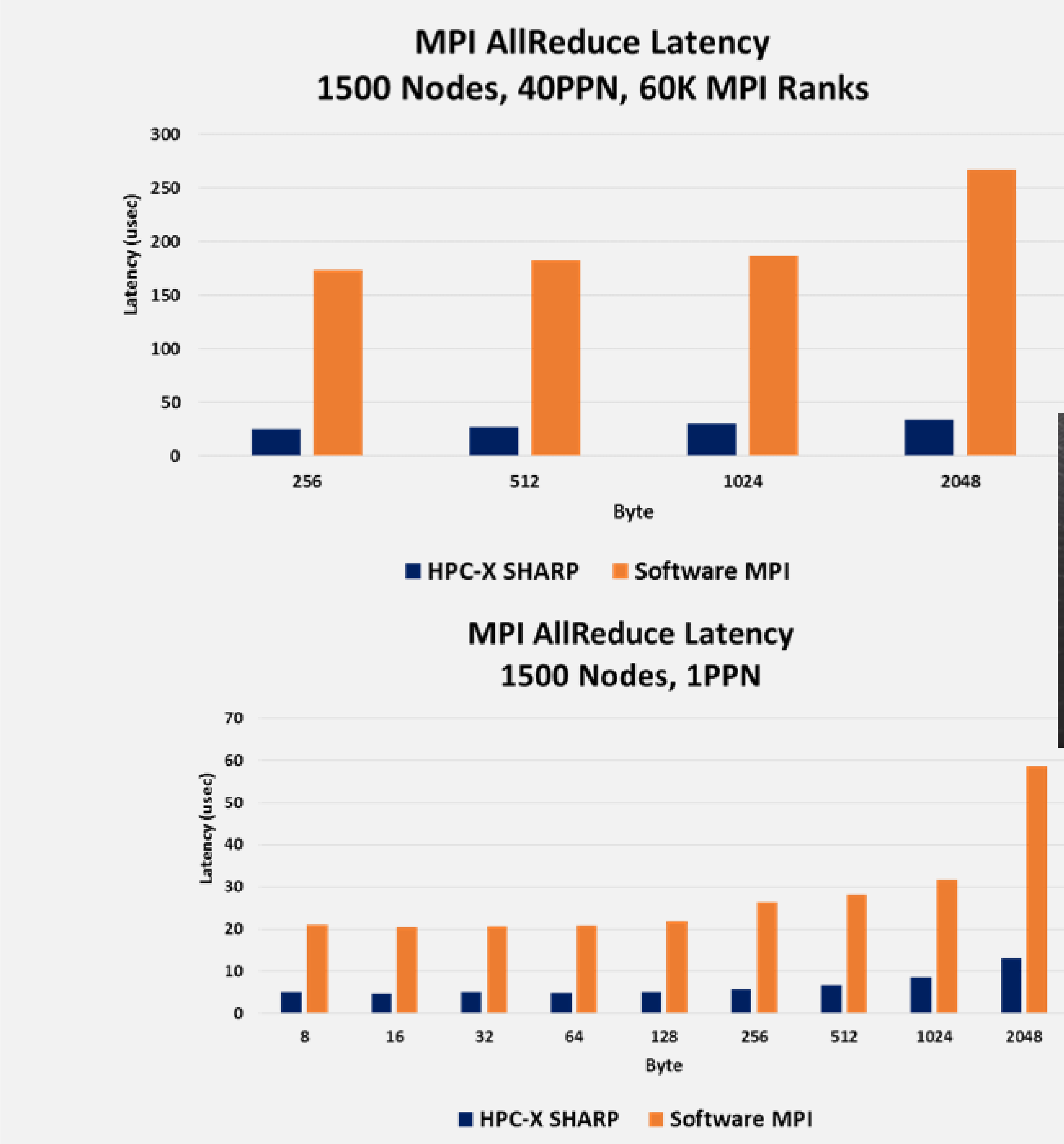
HDR InfiniBand Quantum switch added support for long vector reduction

- Referred to as Streaming Aggregation (SAT) SHARP
- Bandwidth optimized, fully offloaded to the network hardware - asynchronous

SHARP ALLREDUCE PERFORMANCE ADVANTAGES

Providing Flat Latency, 7X Higher Performance





SHARP NEW FEATURES

- SHARP v2.0 HDR Quantum switch
  - Support for small vector reductions
  - Improved latency reduction for small vectors (LLT – low latency trees)
  - Support for large vector reductions – perform reductions at line rate (SAT – streaming aggregation trees)
    - Support for two simultaneous streaming operations per switch (limited resource)
  - **Works together with GPUDirect RDMA**
  - **SAT killer app is distributed, synchronous deep learning workloads**
    - Distributed stochastic gradient descent
    - Limiter is large vector allreduce / bandwidth – gradient averaging between nodes
  - Initial Support for BCAST

SHARP NEW FEATURES

- Support for using SHARP on virtual ports
- Support for using UCX for message communication between SHARPD & SHARP_AM
- Non-default PKEY support
- PCI Relaxed Ordering

SHARP DESIGN CHARACTERISTICS – RELIABILITY MODEL

Transport provides data transfer reliability

Tree failure

revoke resources

Notify the user of failure

SHARRP DESIGN CHARACTERISTICS

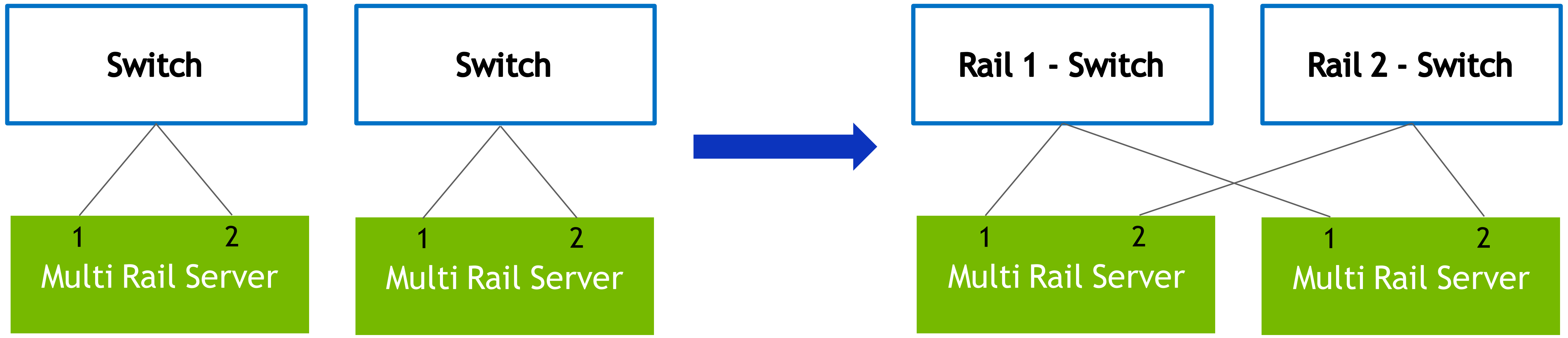
- New tree type defined which supports the Streaming-Aggregation
  - Same layout as that for the low-latency reduction trees
- Tree is locked for a specified duration before use
  - Scarce resource
  - Mirror low-latency reduction tree is used to lock the tree
- Transport selected: RC
  - Reliable
  - Sparse connectivity
- Data is pipelined through the tree

NETWORK TOPOLOGY DESIGN

Rail-optimization for SHARPv2

Rail optimized topology design properties:

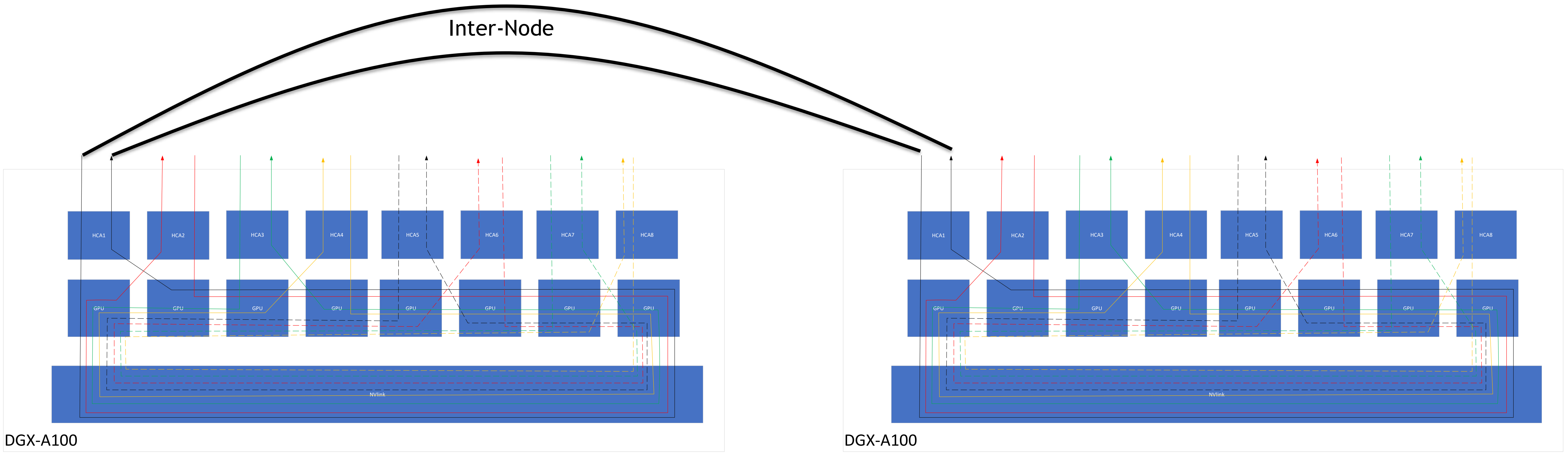
- Keep rail affinity connectivity
- Maximize the number of servers that are reachable on each switch hop (or less switch hops)
- Maximize SHARP reduction capabilities for multi-rail configurations when framework can utilize it



NETWORK TOPOLOGY DESIGN

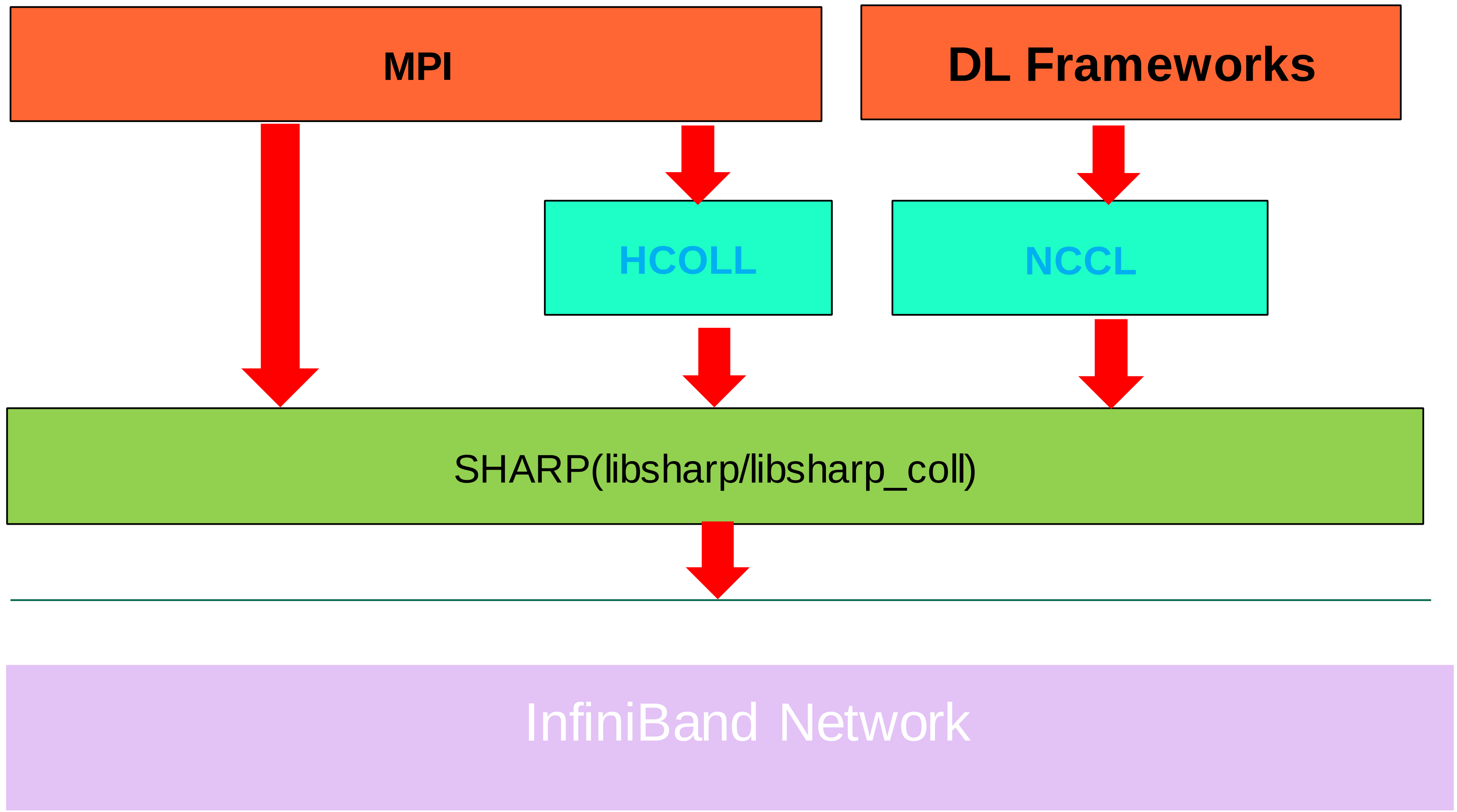
Optimal rail-optimization for SHARPv2

Fit rail-aware workloads – such as NCCL

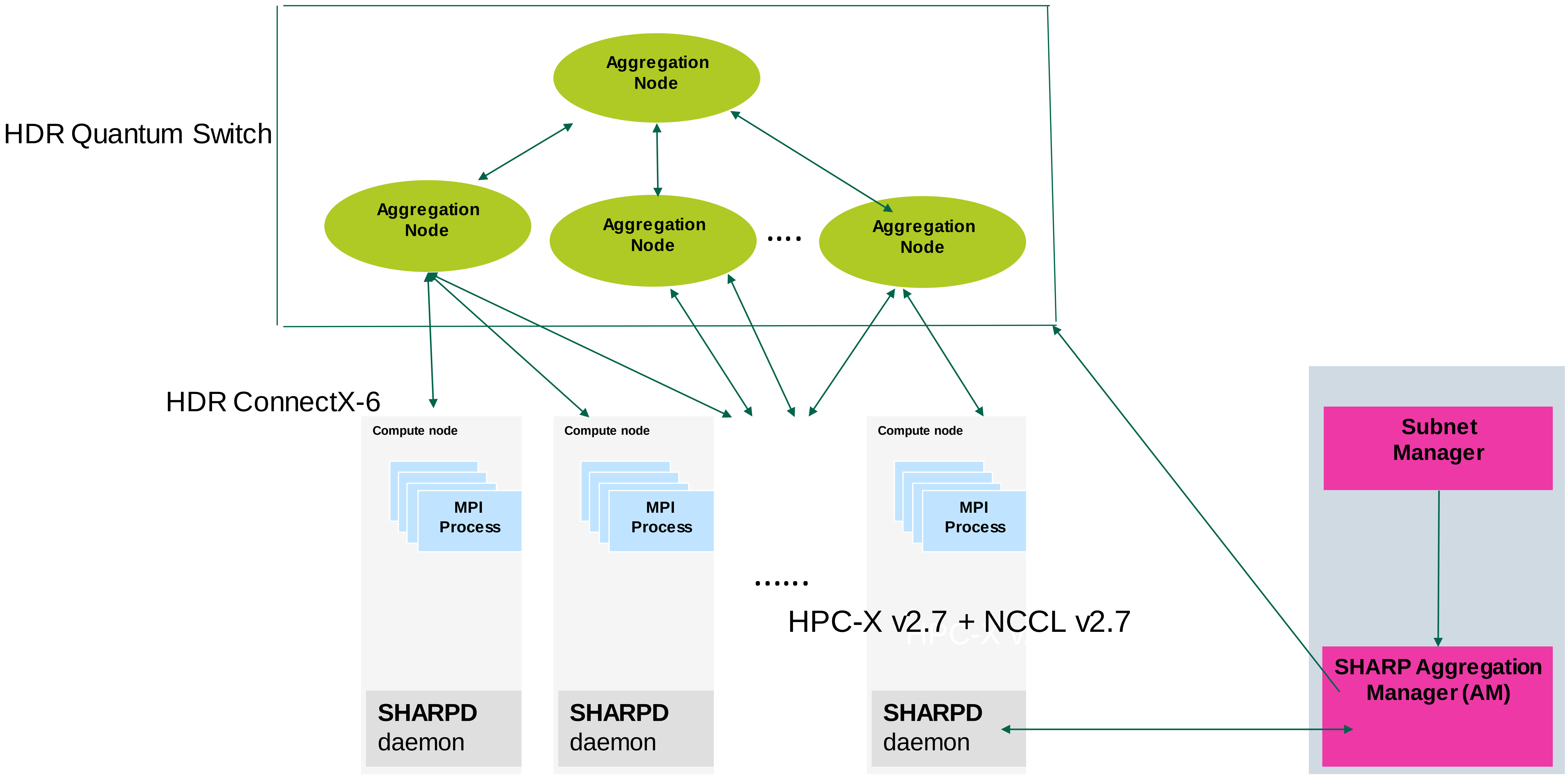


SHARP SW ARCHITECTURE

- MPI
    - Open MPI/Specturm MPI, MVAPICH
- HCOLL
    - Optimized collective library.
- NCCL
    - Optimized GPU collective library
- SHARP
    - Easy to use high level API



SHARP S/W ARCHITECTURE



USING SHARP WITH MPI

- Integrated with multiple MPI libraries
  - MVAPICH2
    - MV2_ENABLE_SHARP
  - OMPI (HPC-X, Spectrum MPI)
    - HCOLL_ENABLE_SHARP
    - SHARP_COLL_ENABLE_SAT (for streaming aggregation)

MPI COLLECTIVE OFFLOADS USING SHARP

HCOLL_ENABLE_SHARP

Enable SHARP

HCOLL_SHARP_NP (default: 2)

- Number of nodes(node leaders) threshold in communicator to create SHARP group and use SHARP collectives

SHARP_COLL_LOG_LEVEL

0 – fatal , 1 – error, 2 – warn, 3 – info, 4 – debug, 5 – trace

SHARP_COLL_ENABLE_SAT=1

Enables SHARP Streaming aggregation

SHARP_COLL_SAT_THRESHOLD=16386

Message size threshold to switch from LLT(Local latency Tree) to SAT (Streaming Aggregation Tree)

MPI COLLECTIVE OFFLOADS USING SHARP

Resources (quota)

- SHARP_COLL_JOB_QUOTA_MAX_GROUPS
  - #communicators
- SHARP_COLL_JOB_QUOTA_OSTS
  - Parallelism on communicator
- SHARP_COLL_JOB_QUOTA_PAYLOAD_PER_OST
  - Payload/OST

For complete list of SHARP COLL tuning options

$HPCX_SHARP_DIR/bin/sharp_coll_dump_config -f

USING SHARP WITH MPI

```
$ mpirun -np 128 -map-by ppr:1:node -x HCOLL_ENABLE_SHARP=3  -x SHARP_COLL_ENABLE_SAT=1 ./osu_allreduce
# OSU MPI Allreduce Latency Test v5.6.2
4         5.61
8         5.53
16        5.55
32        5.76
64        5.78
128       5.86
256       6.51
512       6.92
1024      7.64
2048      8.91
4096      10.63
8192      14.54
16384     13.63
32768     16.73
65536     19.16
131072    25.71
262144    33.29
524288    57.17
1048576   96.53
```

LLT

SAT

GPU DIRECT RDMA

Network adapter can directly read data from GPU device memory

Avoids copies through the host

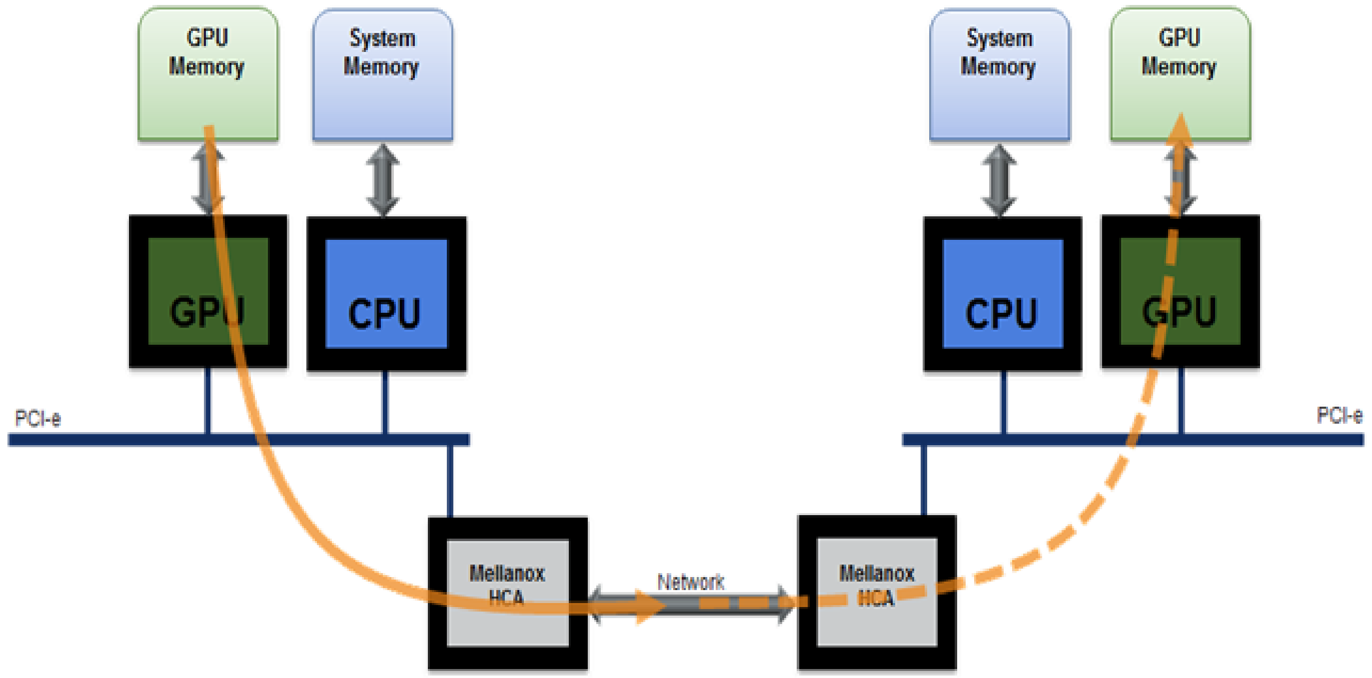
Eliminates CPU bandwidth and latency bottlenecks

Uses remote direct memory access (RDMA) transfers between GPUs

Resulting in significantly improved MPISendRecv efficiency between GPUs in remote nodes

Fastest possible communication between GPU and other PCI-E devices

Allows for better asynchronous communication



SHARP + GPU DIRECT RDMA

Supports CUDA buffers in SHARP API

GDR copy optimizations for smaller messages

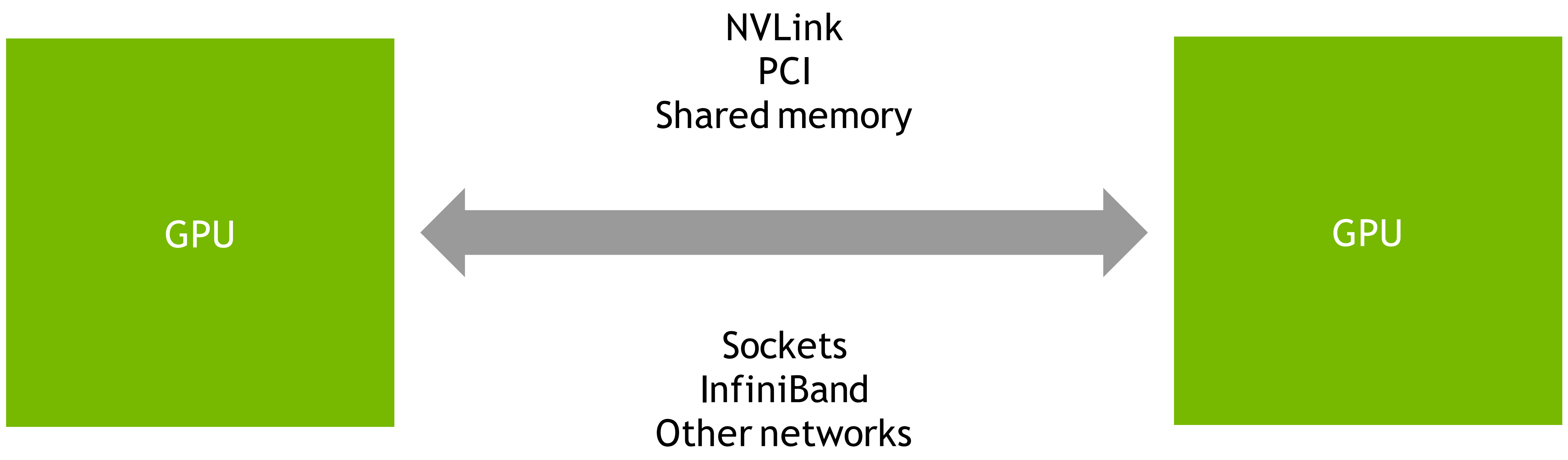
GPUDirect RDMA Streaming Aggregation

NVLINK + GPUDirect RDMA + SHARP

OPTIMIZED INTER-GPU COMMUNICATION

NCCL : NVIDIA Collective Communication Library

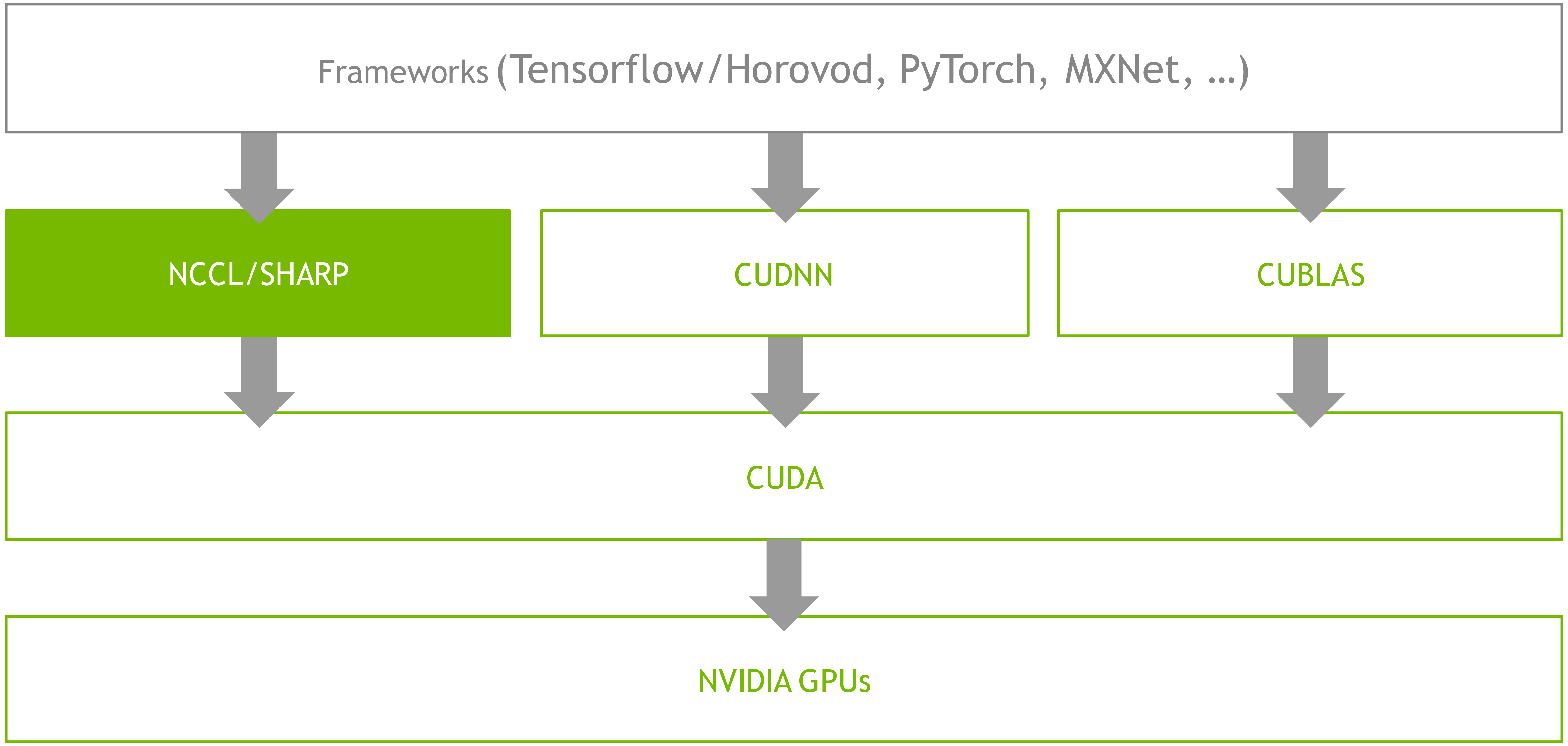
Communication library running on GPUs, for GPU buffers.



Binaries : https://developer.nvidia.com/nccl and in NGC containers
Source code : https://github.com/nvidia/nccl
Perf tests : https://github.com/nvidia/nccl-tests

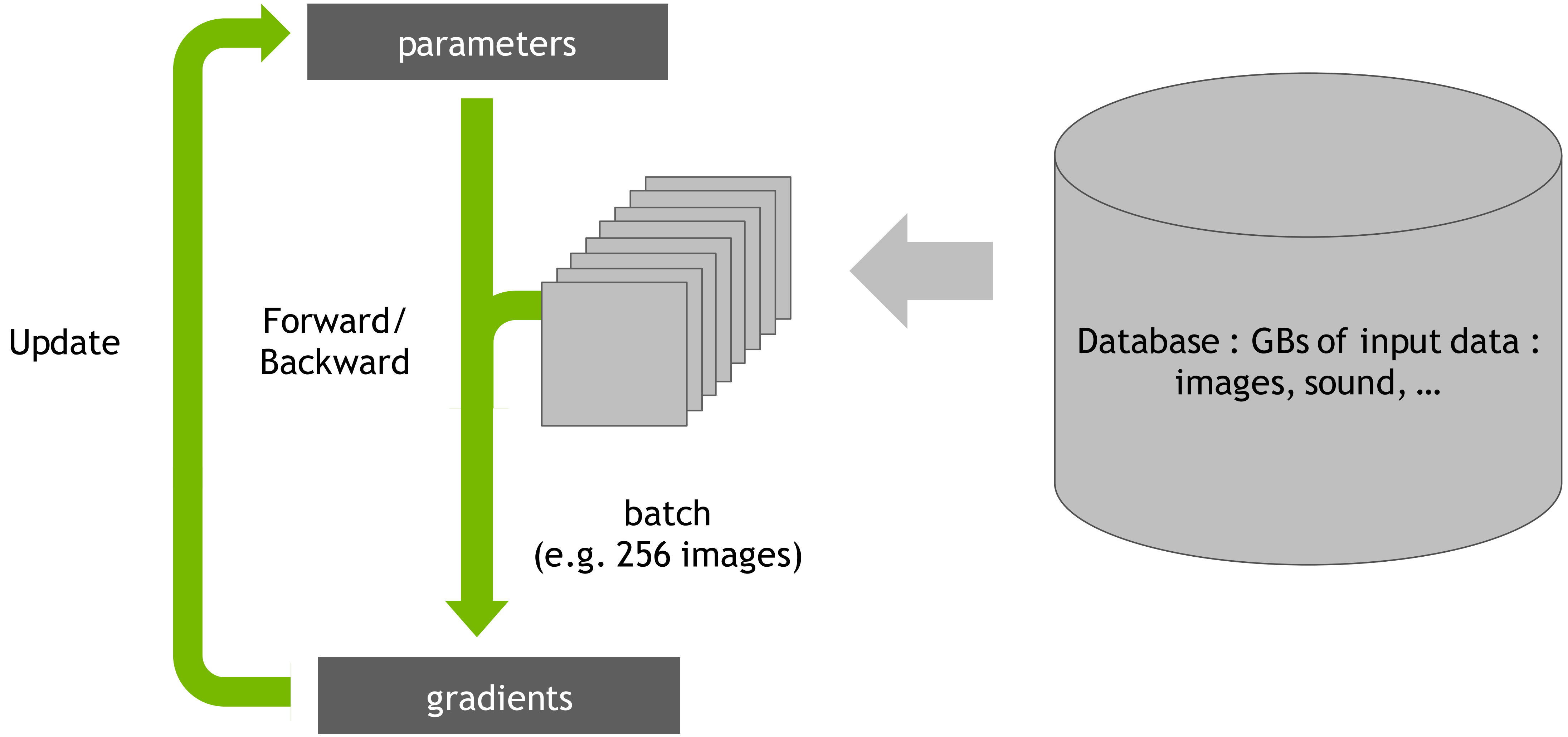
USING SHARP WITH DL

DL stack



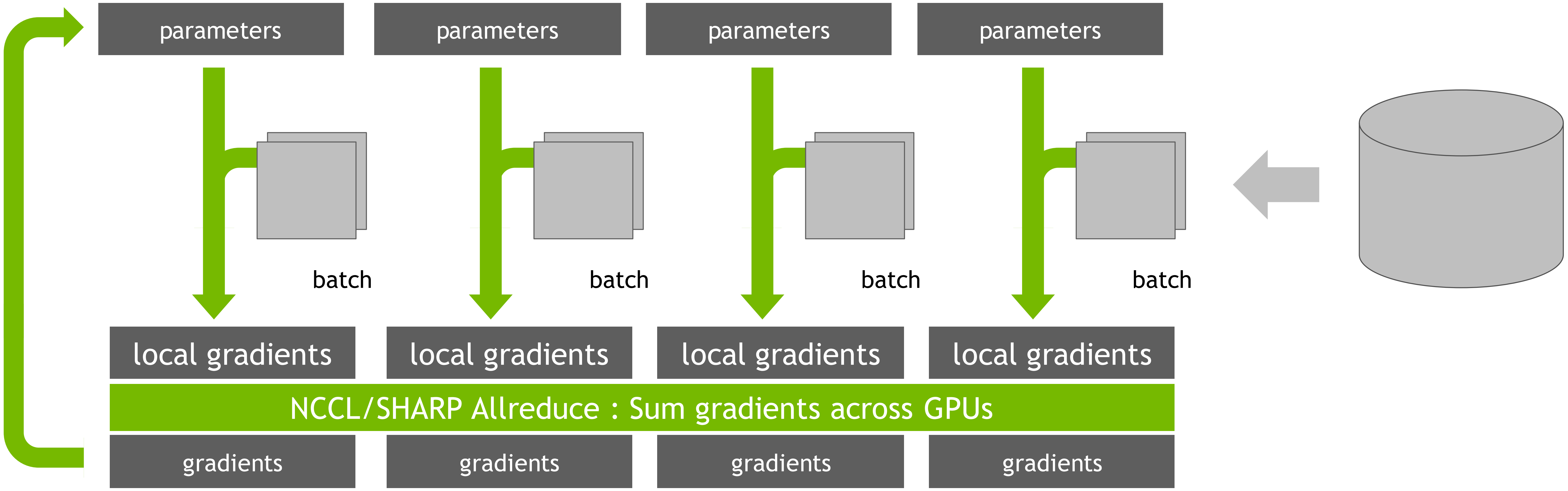
MULTI-GPU TRAINING

Single-GPU



USING SHARP WITH DL

Data parallel

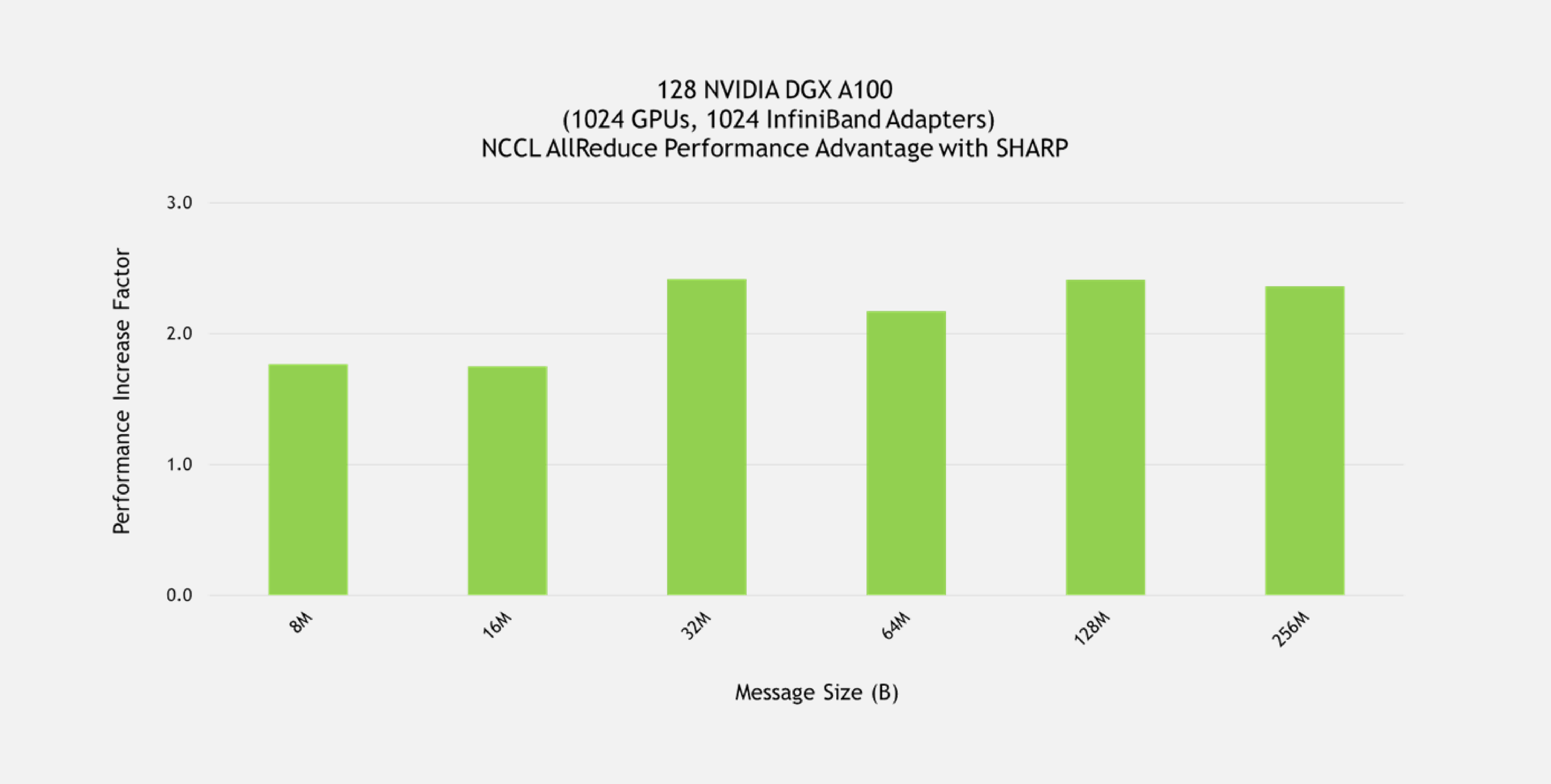


USING SHARP WITH NCCL

- NCCL Plugin
  - Source : https://github.com/Mellanox/nccl-rdma-sharp-plugins
  - Binary distribution with HPC-X
- Simple to use
  - Set plugin lib path with LD_LIBRARY_PATH
  - **NCCL variables:**
    - NCCL_COLLNET_ENABLE=1
    - NCCL_ALGO=CollNet (< NCCL-2.7.8)

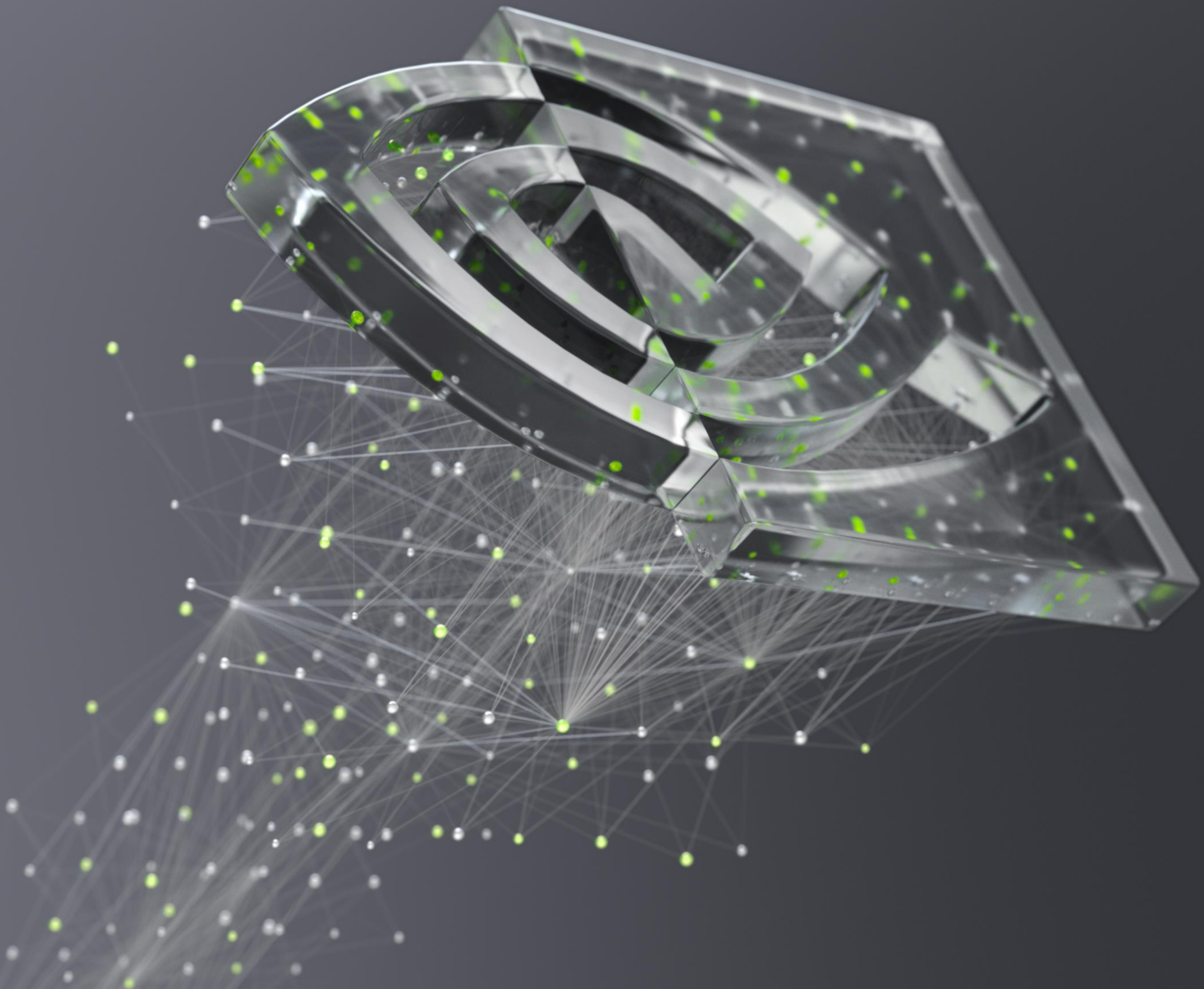
INFINIBAND SHARP AI PERFORMANCE ADVANTAGE

2.5X Higher Performance



INFINIBAND SHARP AGGREGATION - SUMMARY

- Low latency Trees for smaller messages to accelerate MPI application
- High Bandwidth Streaming trees to achieves high network utilization to accelerate DL applications
- Higher efficiency with in-network computing than any host based end-point based algorithms



NVIDIA