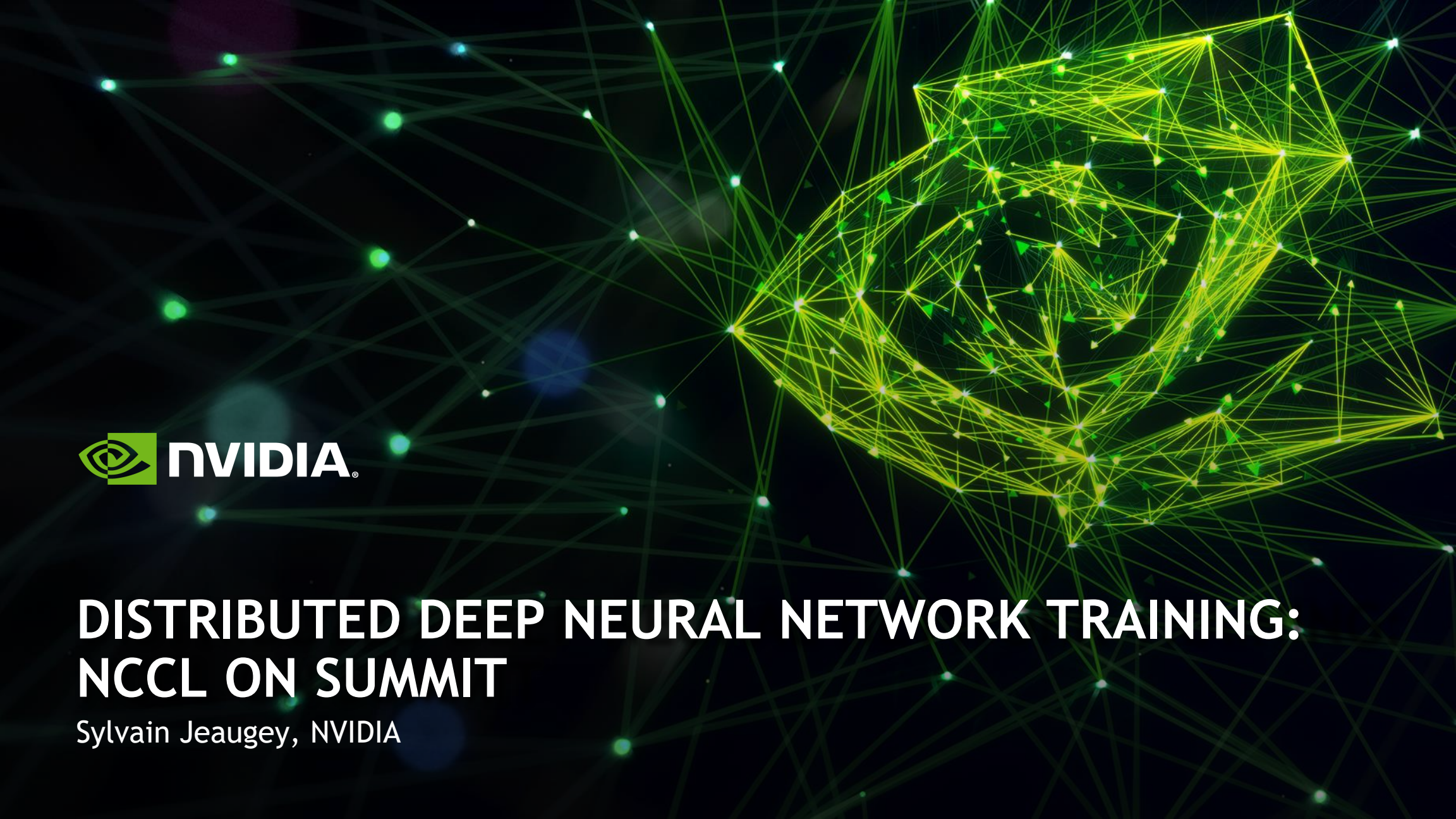


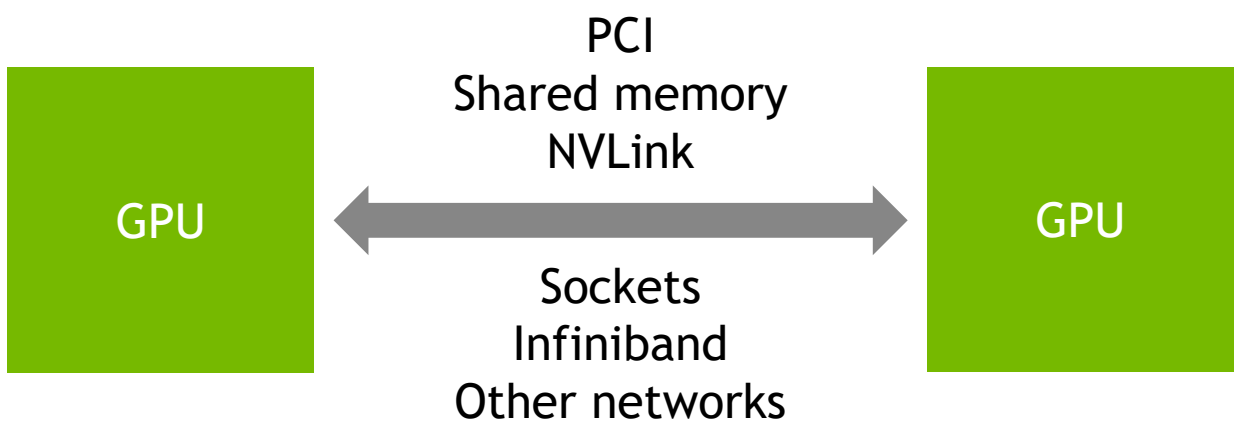
DISTRIBUTED DEEP NEURAL NETWORK TRAINING: NCCL ON SUMMIT

Sylvain Jeaugey, NVIDIA

OPTIMIZED INTER-GPU COMMUNICATION

NCCL : NVIDIA Collective Communication Library

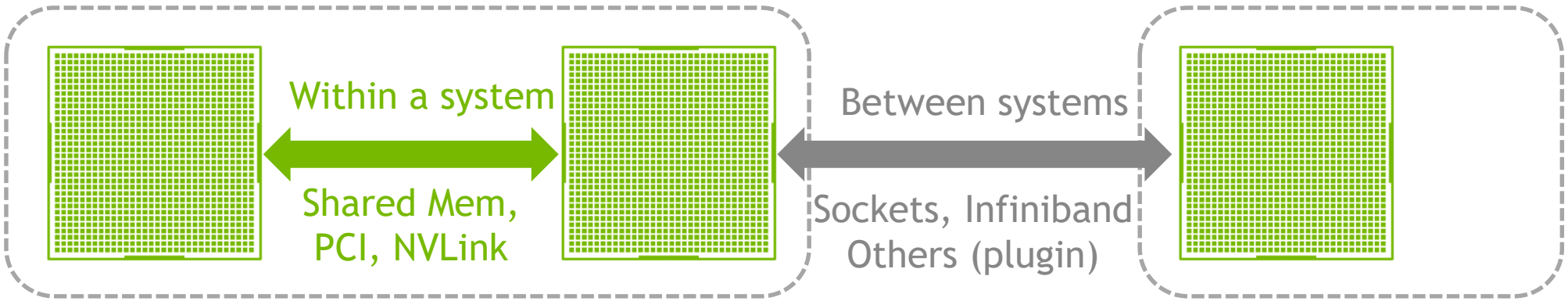
Communication library running on GPUs, for GPU buffers.



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

INTER-GPU COMMUNICATION

Intra-node and Inter-node



6-9	QPI (shared memory)
9-12	PCI Express Gen3 x16 (P2P)
42	NVLink/P9, V100
62	NVLink, P100 (P2P)
132	NVLink, V100 (P2P)

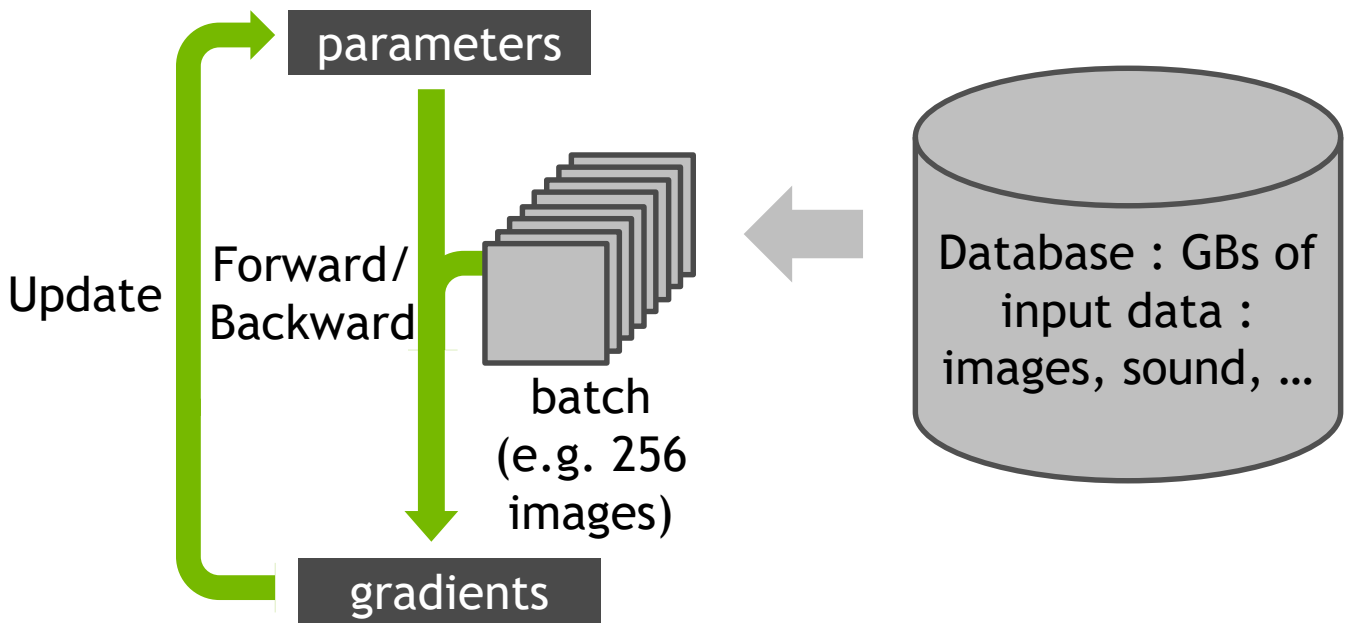
1.2	10GbE, TCP/IP Sockets
12	100Gb IB or RoCE, RDMA (IB verbs)
24	2x 100Gb (Summit)
47	4x 100Gb (DGX1)
82	8x 100Gb (DGX2)

Effective bandwidth in GB/s


DEEP NEURAL
NETWORK TRAINING

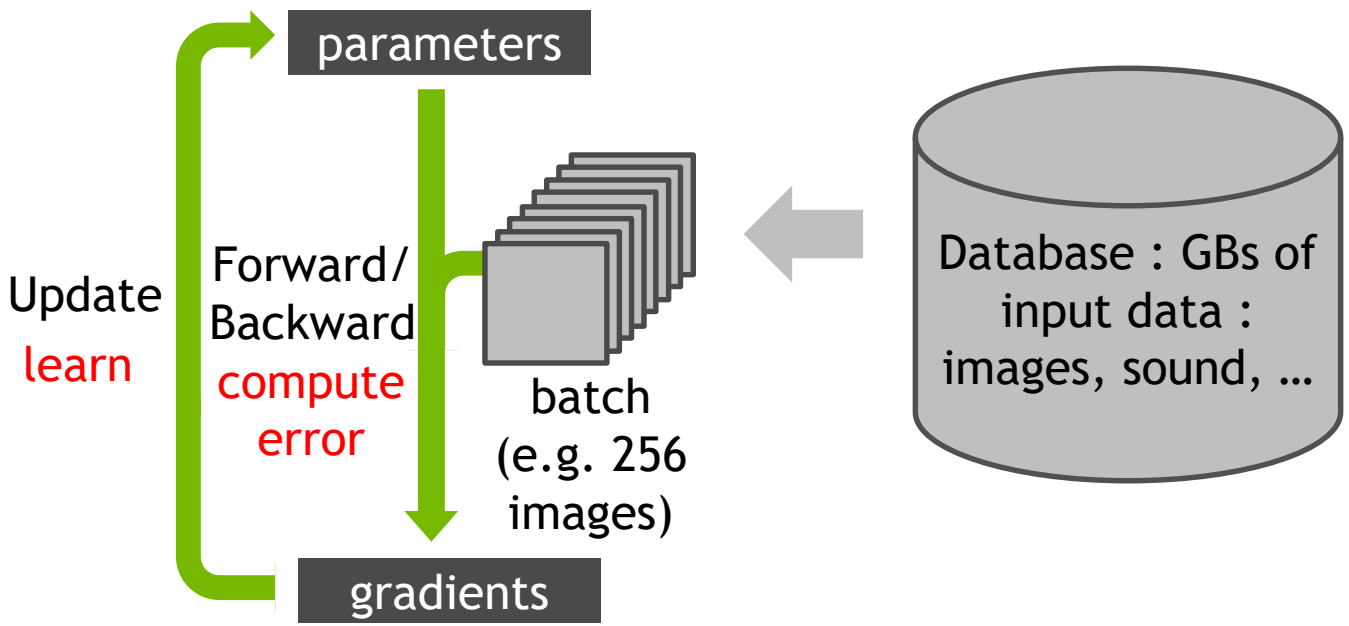
MULTI-GPU TRAINING

Single-GPU



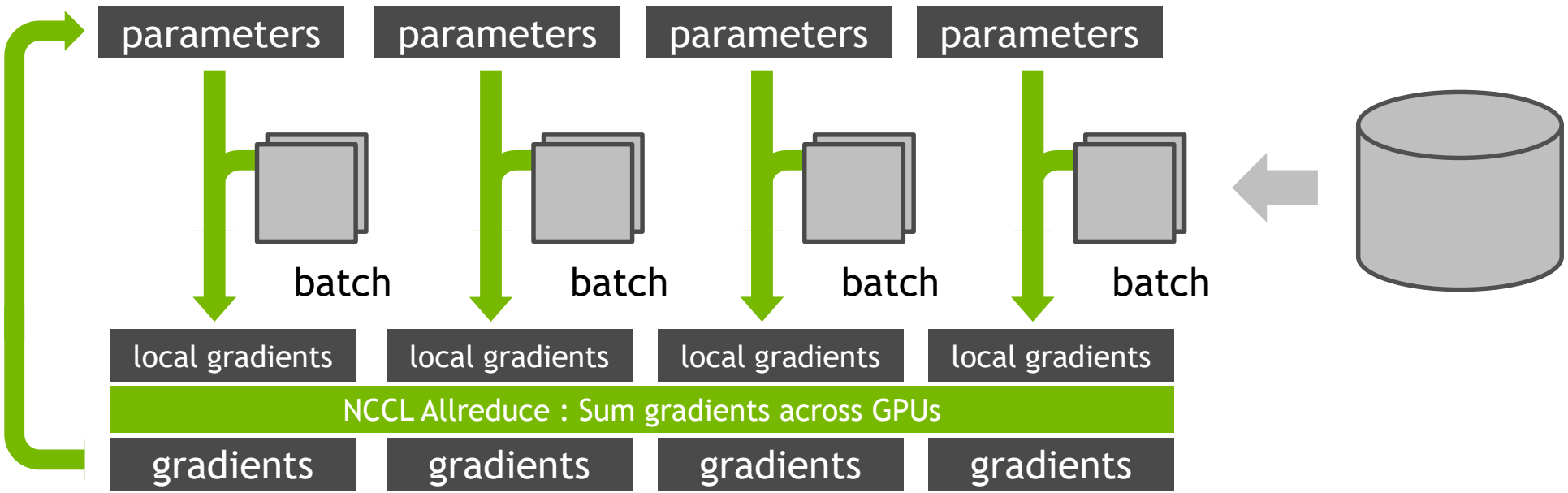
MULTI-GPU TRAINING

Single-GPU



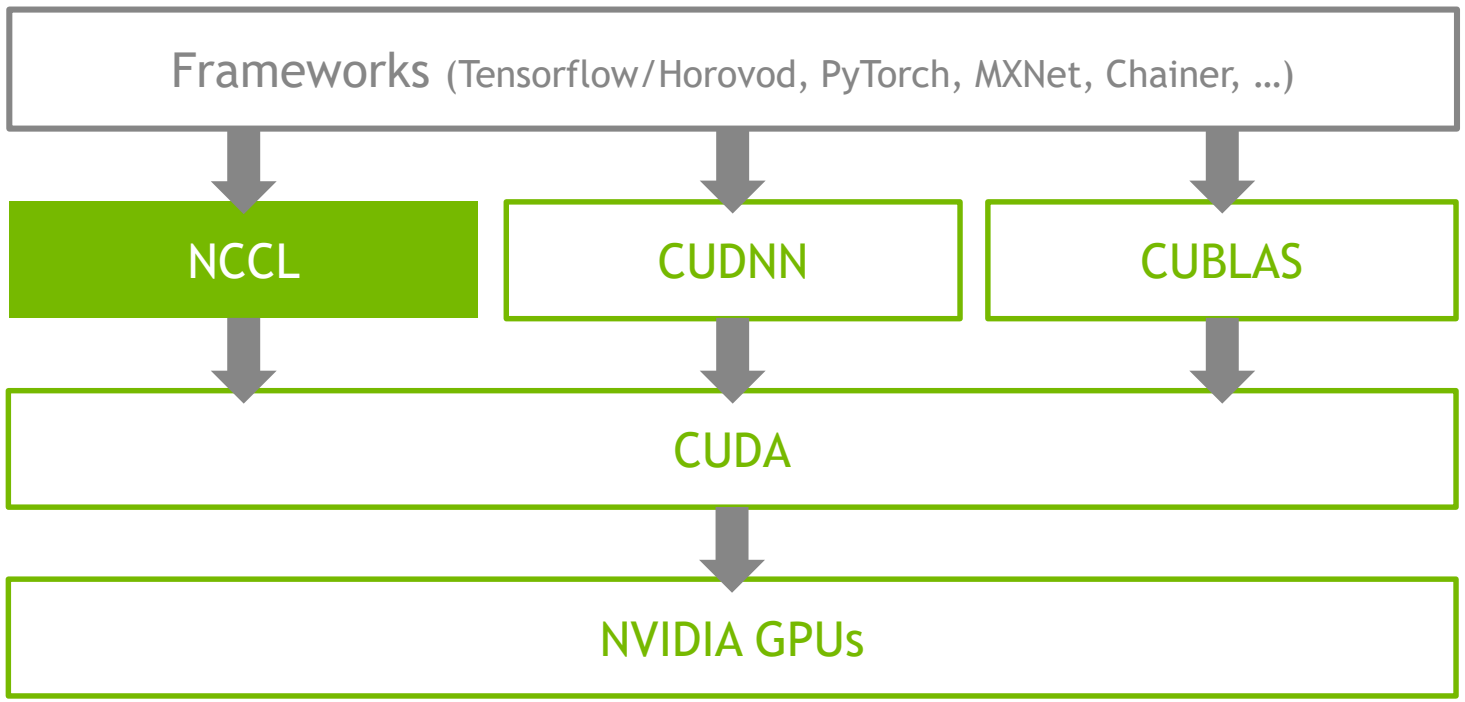
MULTI-GPU TRAINING

Data parallel



NCCL

DL stack




USER INTERFACE

NCCL API

Aims to cover most of MPI-1, plus fault tolerance

```
// Communicator creation
ncclGetUniqueId(ncclUniqueId* commId);
ncclCommInitRank(ncclComm_t* comm, int nranks, ncclUniqueId commId, int rank);
ncclCommSplit(ncclComm_t* newcomm, int group, ncclComm_t comm);

// Communicator destruction / fault tolerance
ncclCommDestroy(ncclComm_t comm);
ncclCommAbort(ncclComm_t comm);
ncclCommGetAsyncError(ncclComm_t comm, ncclResult_t* asyncError);

// Collectives communication
ncclAllReduce(void* sbuff, void* rbuff, size_t count, ncclDataType_t type, ncclRedOp_t op,           ncclComm_t comm, cudaStream_t stream);
ncclBroadcast(void* sbuff, void* rbuff, size_t count, ncclDataType_t type,                 int root, ncclComm_t comm, cudaStream_t stream);
   ncclReduce(void* sbuff, void* rbuff, size_t count, ncclDataType_t type, ncclRedOp_t op, int root, ncclComm_t comm, cudaStream_t stream);
ncclReduceScatter(void* sbuff, void* rbuff, size_t count, ncclDataType_t type, ncclRedOp_t op,           ncclComm_t comm, cudaStream_t stream);
ncclAllGather(void* sbuff, void* rbuff, size_t count, ncclDataType_t type,                               ncclComm_t comm, cudaStream_t stream);

// Point-to-point communication
ncclSend(void* sbuff, size_t count, ncclDataType_t type, int peer, ncclComm_t comm, cudaStream_t stream);
ncclRecv(void* rbuff, size_t count, ncclDataType_t type, int peer, ncclComm_t comm, cudaStream_t stream);

// Aggregation/Composition
ncclGroupStart();
ncclGroupEnd();
```

(functions in grey coming in future versions)

NCCL USAGE

Communicators and GPUs

One ncclComm_t handle = one NCCL rank = one GPU.

Fits any parallel model : multi-process, multi-thread, multi-GPU and any combination.

Creating multiple communicators is discouraged

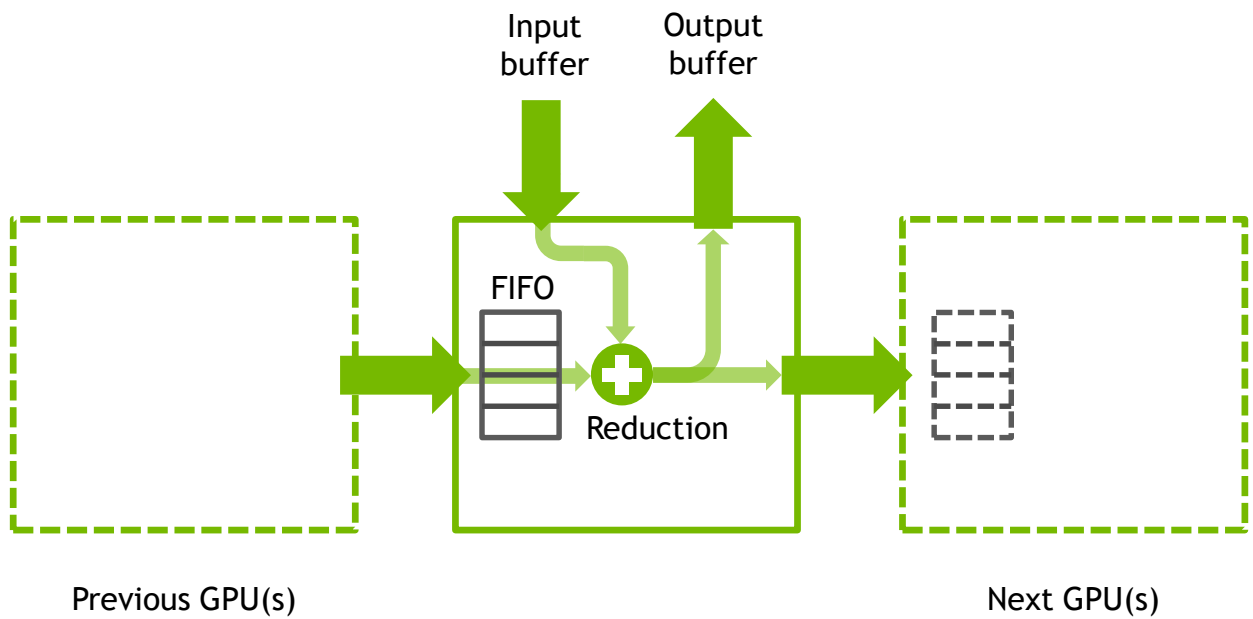
No guaranteed concurrent progress of operations in CUDA, might lead to deadlocks. See ncclCommSplit.


INSIDE NCCL

GPU COMMUNICATION KERNEL

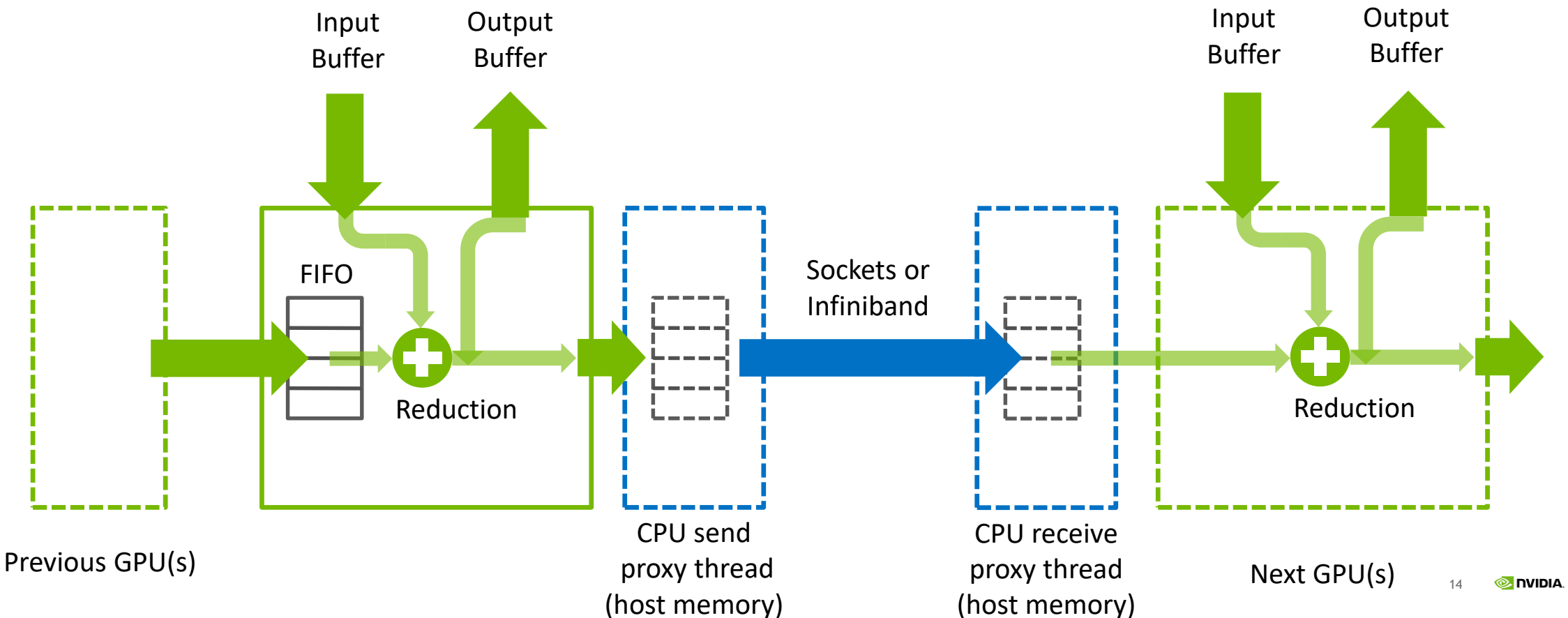
Principle

NCCL CUDA Kernel
- runs on the GPU
- receives and sends from other peers through internal FIFOs
- perform reductions with local buffers, or copies



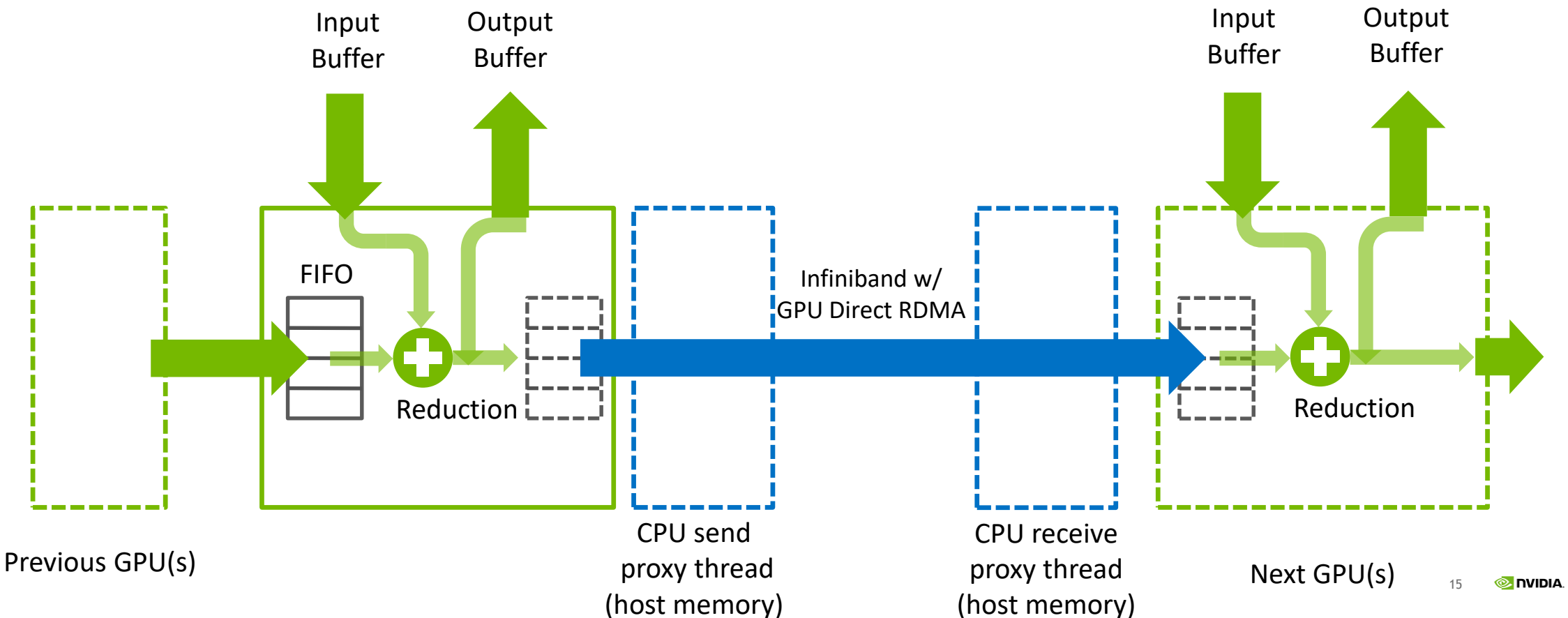
INTER-GPU COMMUNICATION

Inter-node



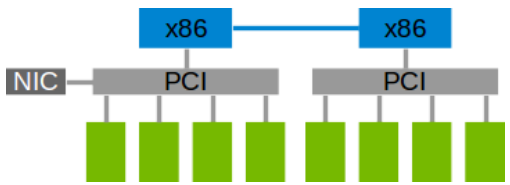
INTER-GPU COMMUNICATION

Inter-node, GPU Direct RDMA

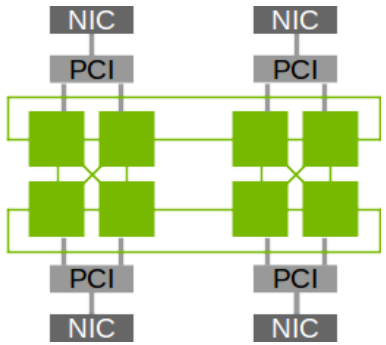


TOPOLOGY AWARENESS

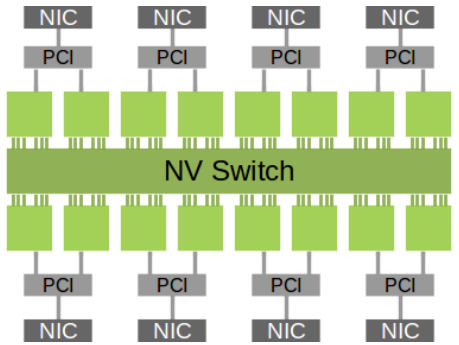
Extensive support for all platforms



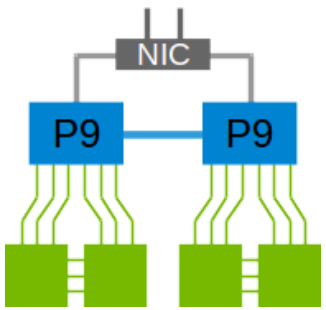
PCI only



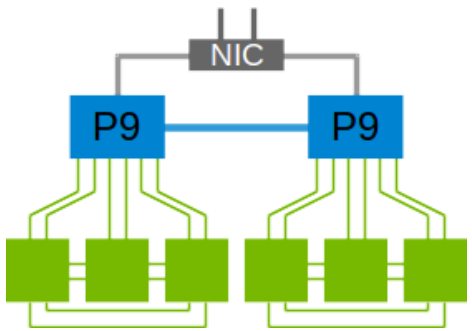
DGX-1



DGX-2



P9/4 V100

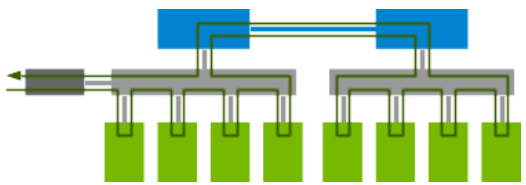


P9/6 V100

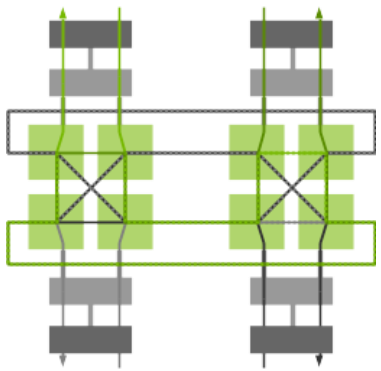
And many others :
ARM, Intel/AMD,
Single RC,
…

TOPOLOGY AWARENESS

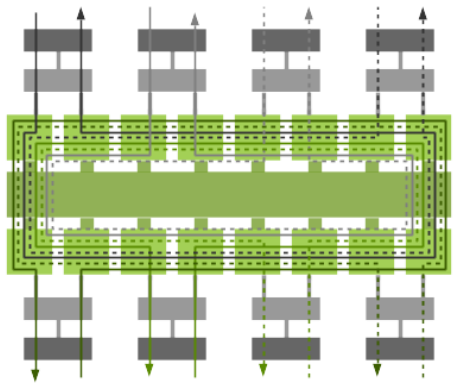
Extensive support for all platforms



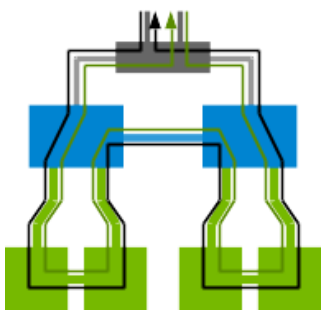
PCI only : 6-12 GB/s



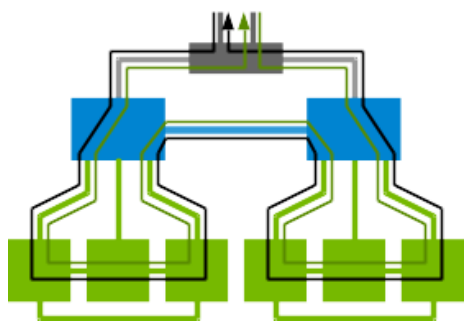
DGX-1 : 48 GB/s



DGX-2 : 85 GB/s



P9/4 V100 : 24 GB/s



P9/6 V100 : 24 GB/s

Extensive topology detection system, multi-path search maximizing performance on all platforms

ALGORITHMS AND PROTOCOLS

For best latency and bandwidth

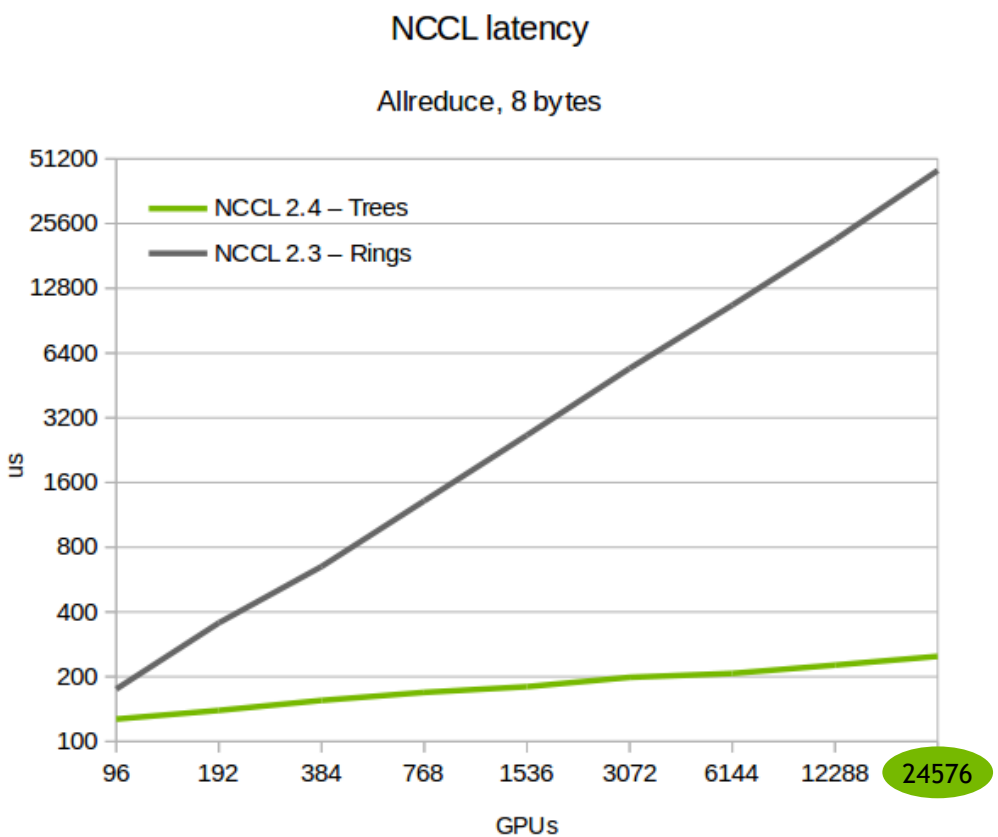
Point-to-point communication protocols
- LL (1): 25-50% bandwidth, 1us latency (since 2.1)
- LL128 (1) : 95% bandwidth, 2us latency (since 2.5)
- Simple : 100% bandwidth, 6us latency (since 2.0)

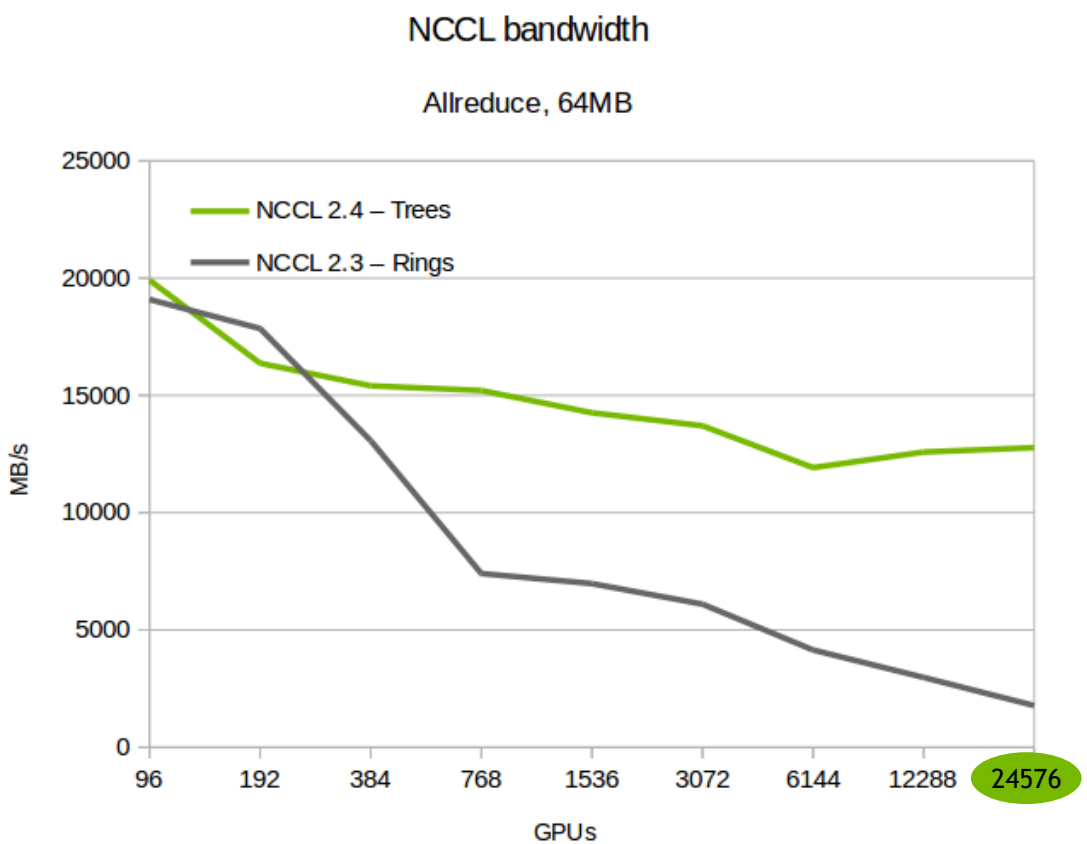
AllReduce algorithms
- Tree (2) : 95% Bandwidth, log latency (since 2.4)
- Ring : 100% bandwidth, linear latency (since 2.0)

(1) LL : low-latency protocol, sending self-flagged data. Receiver polls on data instead of a separate flag
(2) Tree : dual binary tree, arranged in a complementary way to achieve full bandwidth.

SUMMIT

Tree performance





nccl-tests allreduce - 4096 nodes, 6x V100, 2x IB EDR (Summit/Oak Ridge National Laboratory)


FUTURE

NCCL 2.6

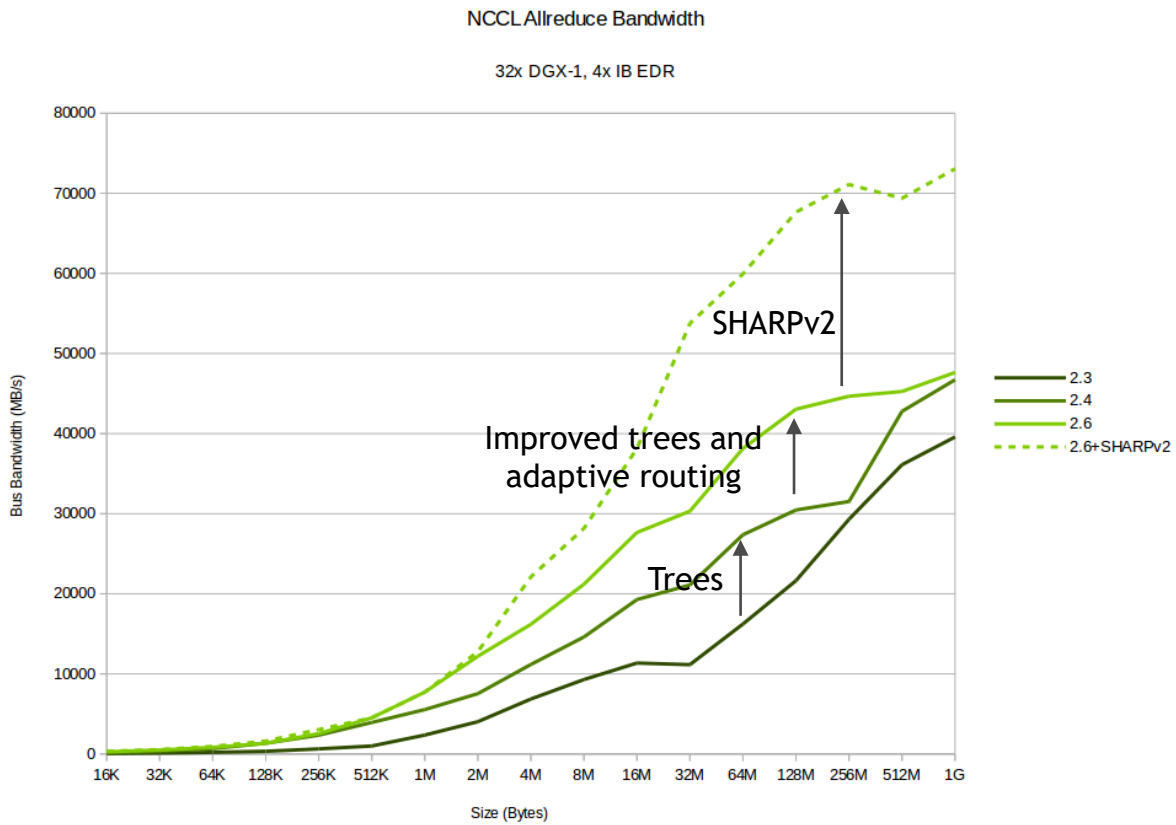
SHARPv2 and Adaptive routing

Continuous performance improvement

Add support for accelerated network collectives e.g. SHARPv2
(not available on Summit)

Add support for adaptive routing
(might need to set NCCL_IB_SL on Summit)

Preview available :
https://github.com/nvidia/nccl/tree/v2.6

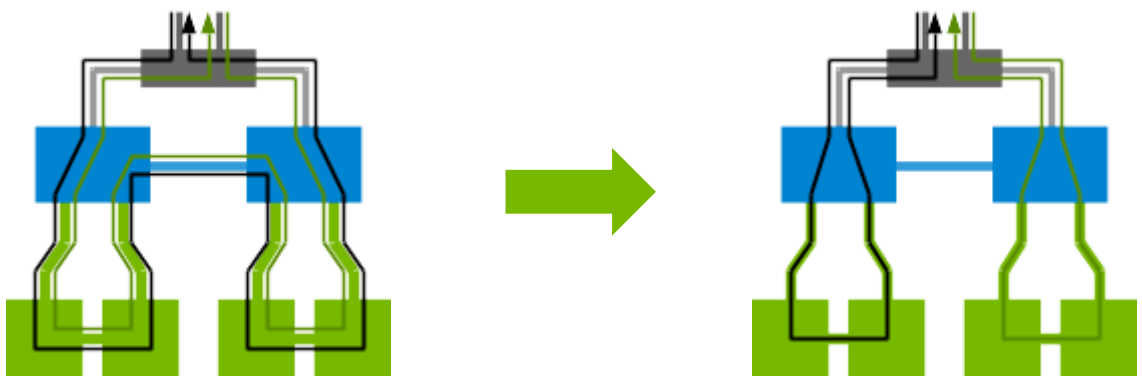


NCCL 2.7

ncclCommSplit

ncclCommSplit creates sub-communicators for :
- Hierarchical collectives
- Model parallelism

Multi-communicator topology search to ensure maximum concurrent bandwidth on all sub-communicators.



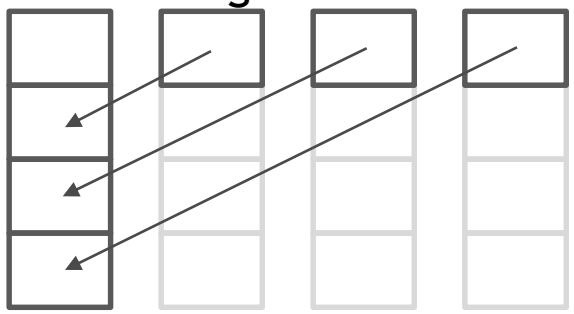
Shares the same resources with the parent communicator and can detect programming errors leading to deadlocks

NCCL 2.8

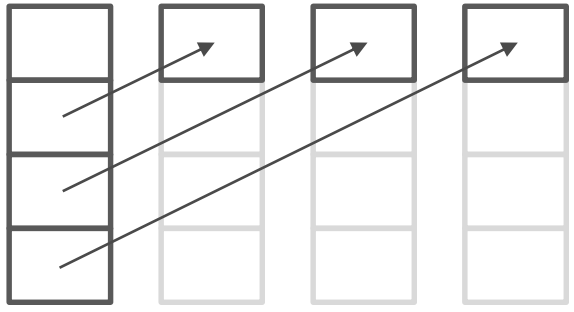
Point-to-point communication

Add **ncclSend** and **ncclRecv** functions to enable Send / Receive, Scatter[v],Gather[v],Alltoall[v,w], neighbor collectives, …

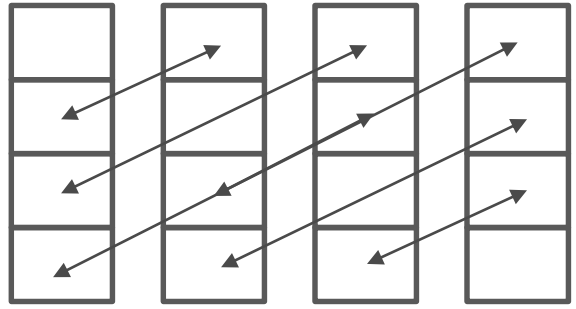
gather



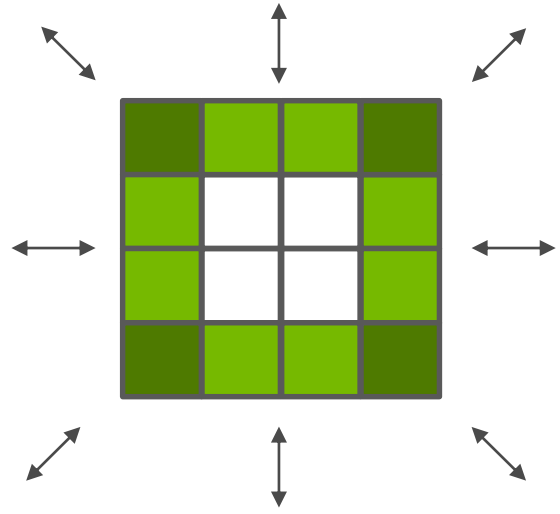
scatter



alltoall



Neighbor



NCCL
Summary

Optimized inter-GPU communication for DL and HPC
- Optimized for all NVIDIA platforms, most OEMs and Cloud
- Scales to 10,000s of GPUs.

Aims at covering all communication needs for multi-GPU computing.

Only relies on CUDA. No dependency on MPI or any parallel environment.

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


NVIDIA