Data Communication for Deep Learning Systems

Ji Liu

Outline

- High Performance Computing (HPC)
- Systems for Deep Learning
- Problems in terms of System

Outline

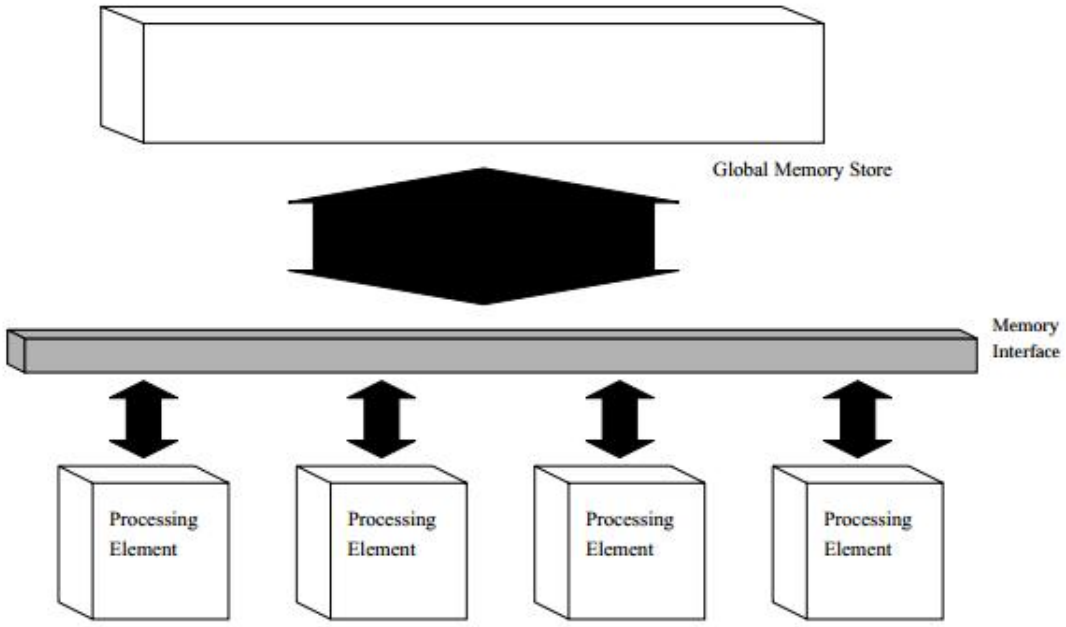
- **High Performance Computing (HPC)**
- Systems for Deep Learning
- Problems in terms of System

Introduction of Parallelization

- **SIMD:** Single Instruction, Multiple Threads
- Parallelization of multiple cores
  - Open Multi-Processing (OpenMP): parallelization among different CPU cores
  - Graphics Processing Unit (GPU): parallelization among different GPU cores (Cuda)
- Parallelization of multiple computers in a cluster
  - **Message Passing Interface (MPI): parallelization among different computing nodes**
  - Hadoop: parallelization among different computers
  - Spark: memory based parallelization among different computers
- High Performance Computing (HPC): OpenMP, GPU and MPI

OpenMP

- Architecture



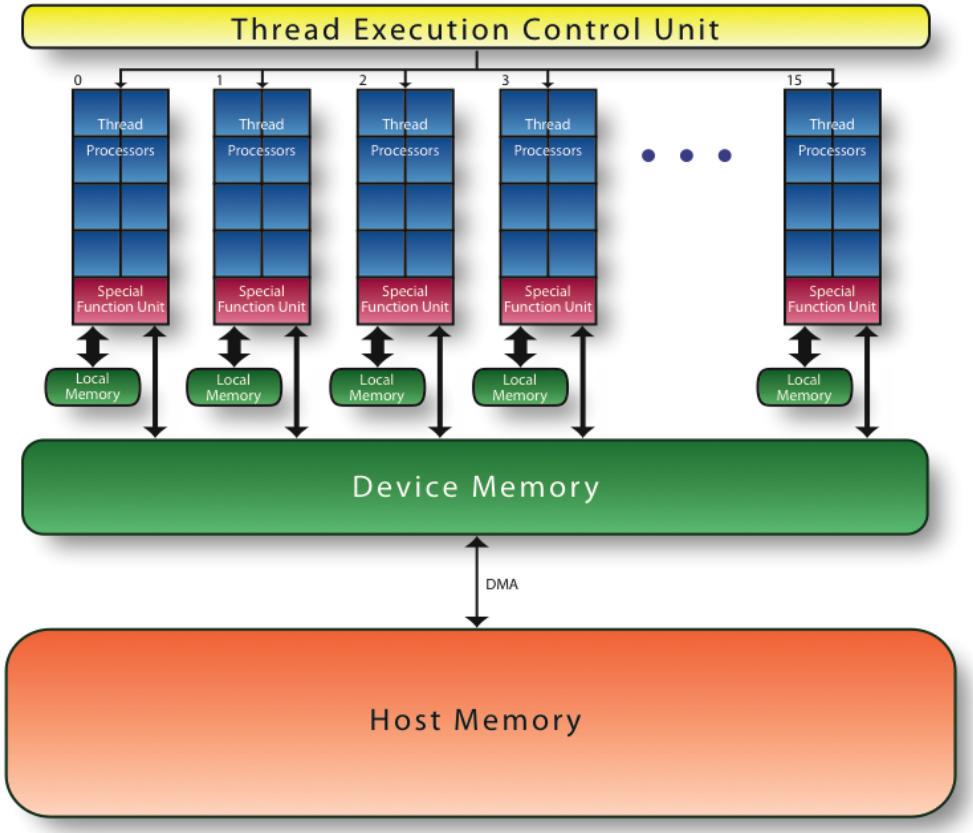
Schematic of a shared memory architecture

https://www.cs.rutgers.edu/~venugopa/parallel_summer2012/openMP.html

- Number of CPU cores: 1 – 32 in a computer
  - Max speedup: 32
- Average desktop CPU speed: 3.2 GHz

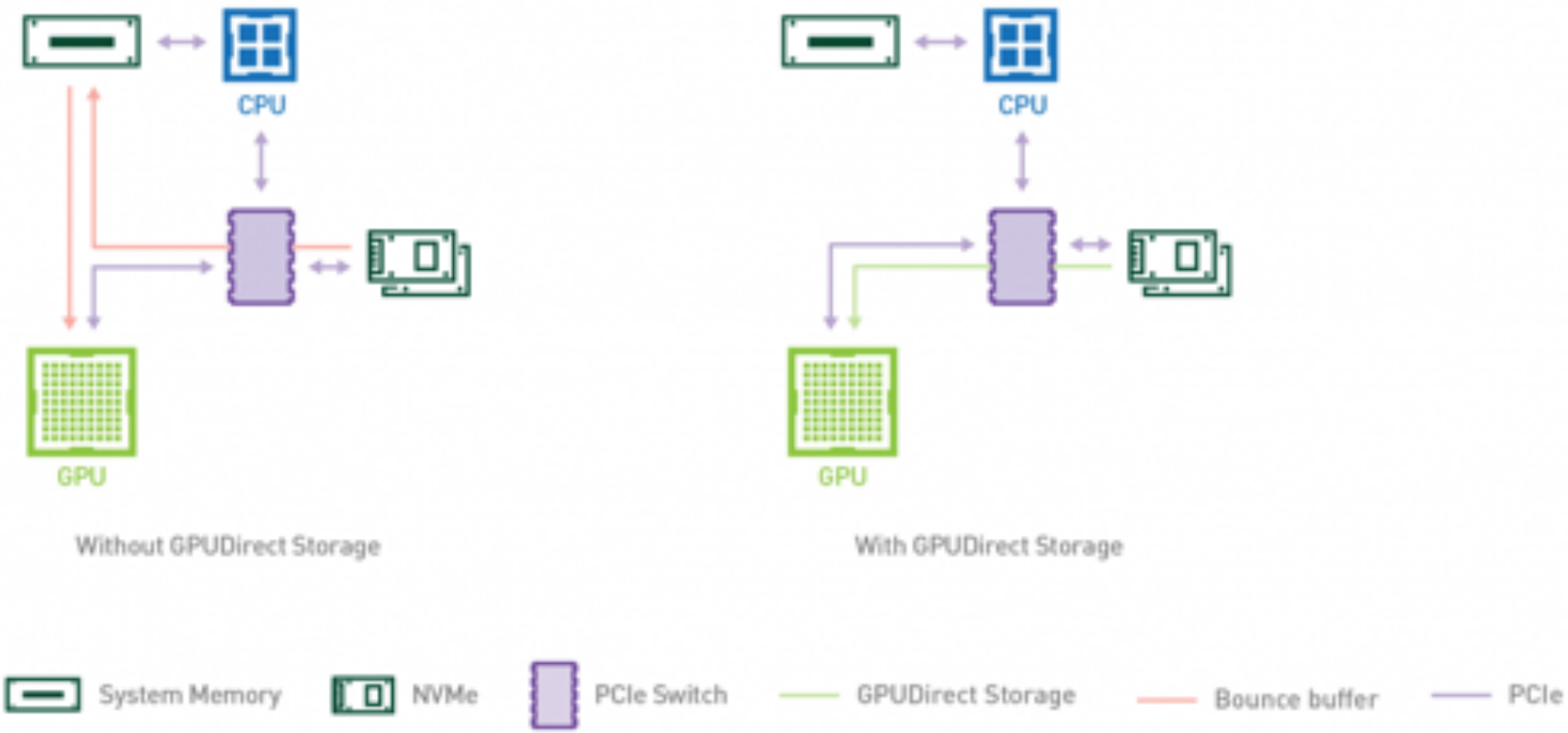
GPU

- Architecture
  - Block
  - Grid
- Number of cores:
  - One card:
    - 700 – 4000
  - Max speedup: 4000
- Average GPU speed: 3 GHz
  - 1GHz - 1.4GHz
- GPU Direct:
  - Directly send data in GPU memory to NIC



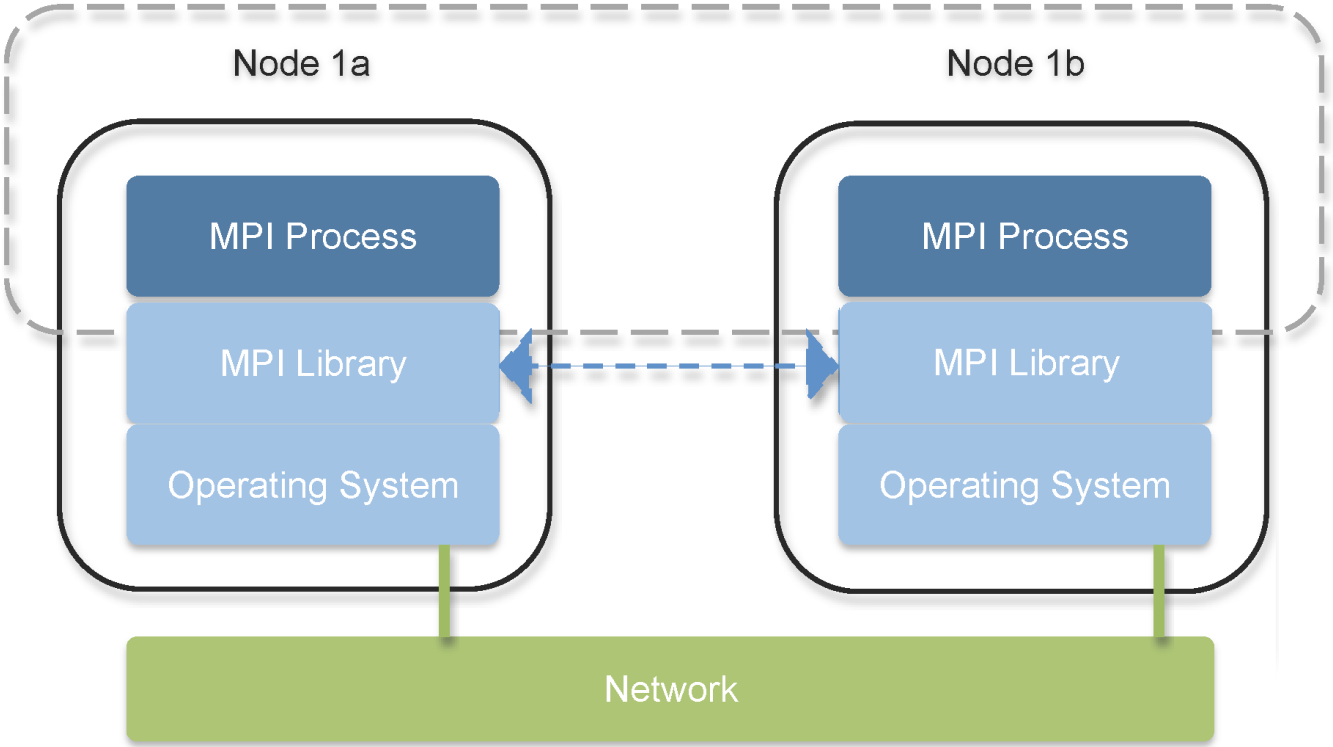
https://www.hpcwire.com/2008/09/10/compilers_and_more_gpu_architecture_and_applications/

GPU Direct


CPU
GPU
Without GPUDirect Storage
CPU
GPU
With GPUDirect Storage
System Memory
NVMe
PCIe Switch
GPUDirect Storage
Bounce buffer
PCIe

MPI

- Communication among different computing nodes
  - Master slave architecture
- Max speedup: number of computers
- Ca
  - I
  - I
- Ca
  sto



GPUs

mited

MPI Functions

- Synchronization: MPI_Barrier
- Broadcast:
  - Synchronization among different nodes
  - One to many:MPI_Bcast, MPI_Scatter
  - Many to one: MPI_Gather, MPI_Reduce
  - Many to many: MPI_Allgather, MPI_Allreduce
- Send: MPI_Send, Receive: MPI_Recv
  - Can be synchronized or asynchronized

MPI Library	Where?	Compilers
MVAPICH	Linux clusters	GNU, Intel, PGI, Clang
Open MPI	Linux clusters	GNU, Intel, PGI, Clang
Intel MPI	Linux clusters	Intel, GNU
IBM BG/Q MPI	BG/Q clusters	IBM, GNU
IBM Spectrum MPI	Coral Early Access and Sierra clusters	IBM, GNU, PGI, Clang

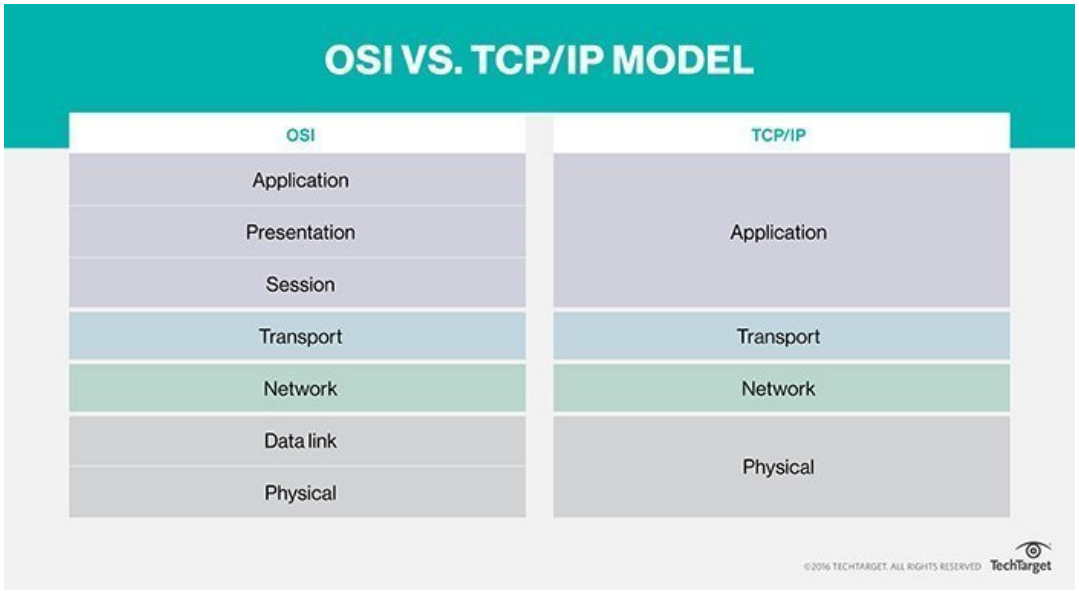
Implementations

- Without RDMA:
  - MPI usually exploit TCP/IP to transfer data [1]
  - Other protocol can be used, e.g. SCTP [1]
  - Main memory: Data in Application memory or GPU -> data in OS kernel -> data in network interface card (NIC) buffer -> network
- With RDMA:
  - Data in Application memory or GPU -> NIC -> network -> another NIC -> data in application memory
  - GPUDirect for GPU
  - Open MPI supports RDMA
  - Requires hardware support and system identification
  - RDMA implementation:
    - InfiniBand: sockets direct protocol (SDP)
    - iWARP (TCP/IP): internet Wide Area RDMA Protocol
    - RoCE: based on TCP (v1.5), UDP (v2) or (FCoE): Fibre Channel over Ethernet (generally for storage)

[1] Kamal et al. SCTP versus TCP for MPI, SC, 2005

TCP/IP

- A suite of communication protocols used to interconnect network devices on the internet.
- Application layer, e.g. MPI
- Transport layer: TCP, FCoE, SDP
- Network layer: IP (Ethernet), Ethertype (for FCoE), SDP
- Physical layer:
  - LANs, ARP

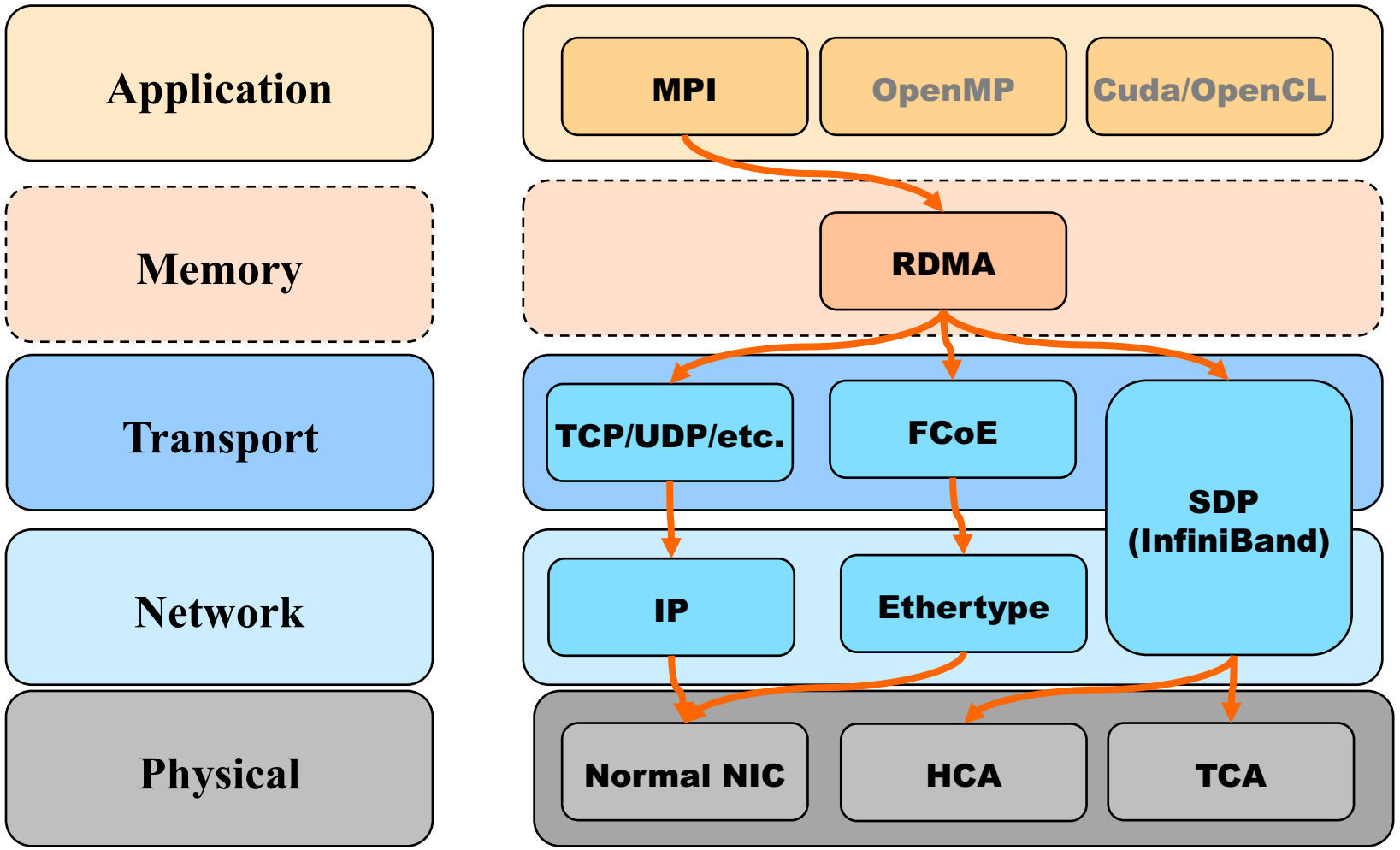


InfiniBand (IB)

- Hardware support: InfiniBand Switches, host channel adapters (HCAs), and target channel adapter (TCA)
- Communication features:
  - Data transfer without OS kernel
  - Handle network protocol of breaking a large message into packets without CPU
- Works on Network Layer and physical layer
- IPoIB (for non RDMA):
  - Send IP packets over IB
  - TCP can work on it
  - Uses CPU to generate IP sockets

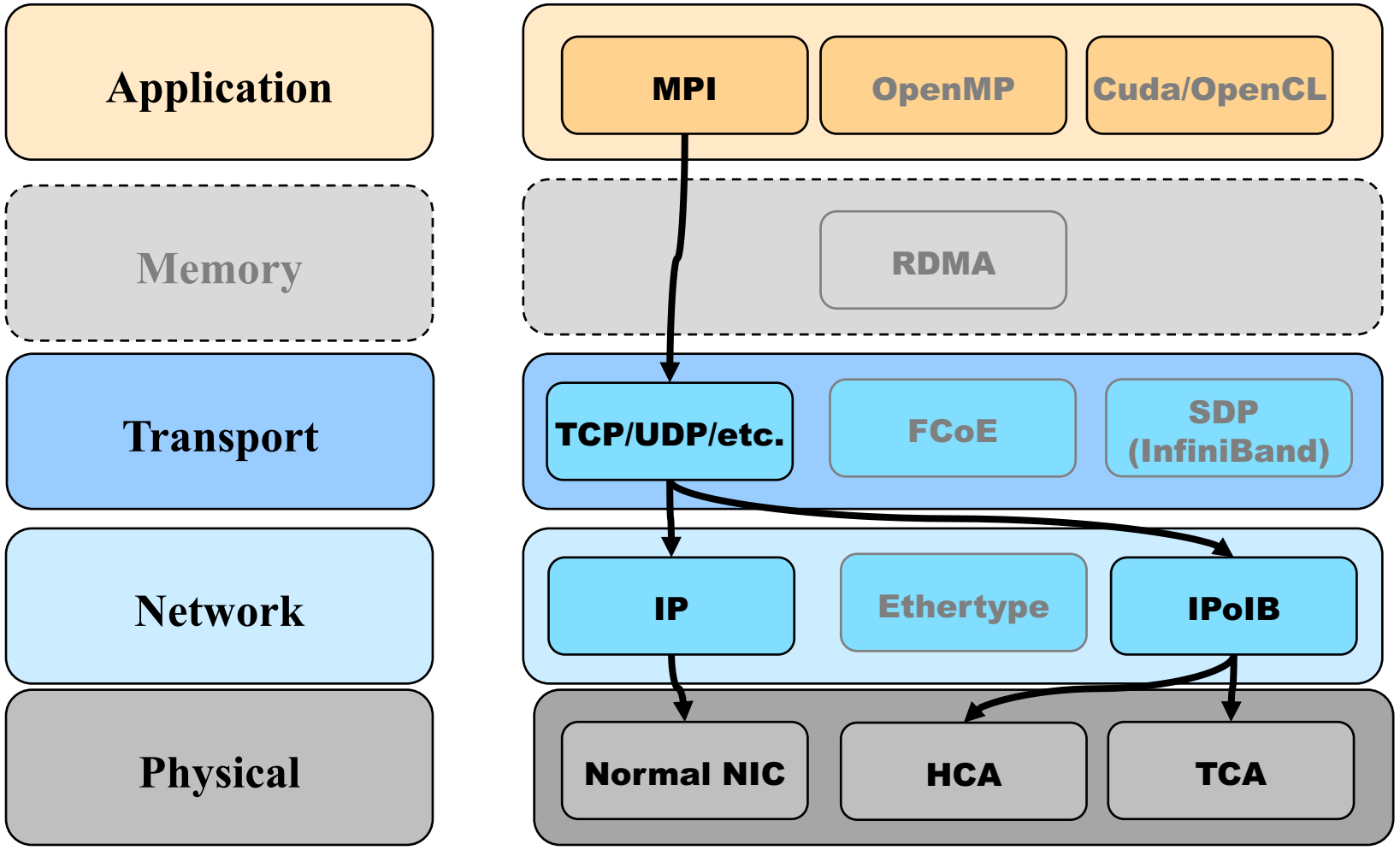
Data Transfer (With RDMA) for MPI

- No/few CPU interaction and no data copy for data transfer



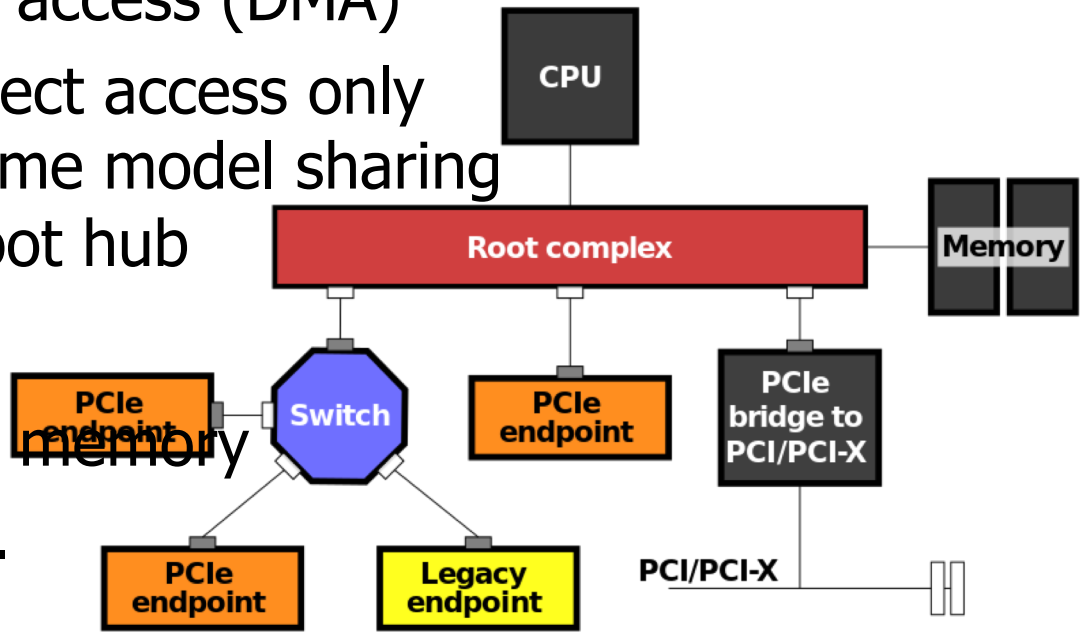
Data Transfer (Without RDMA) for MPI

- CPU interaction and data copy for data transfer



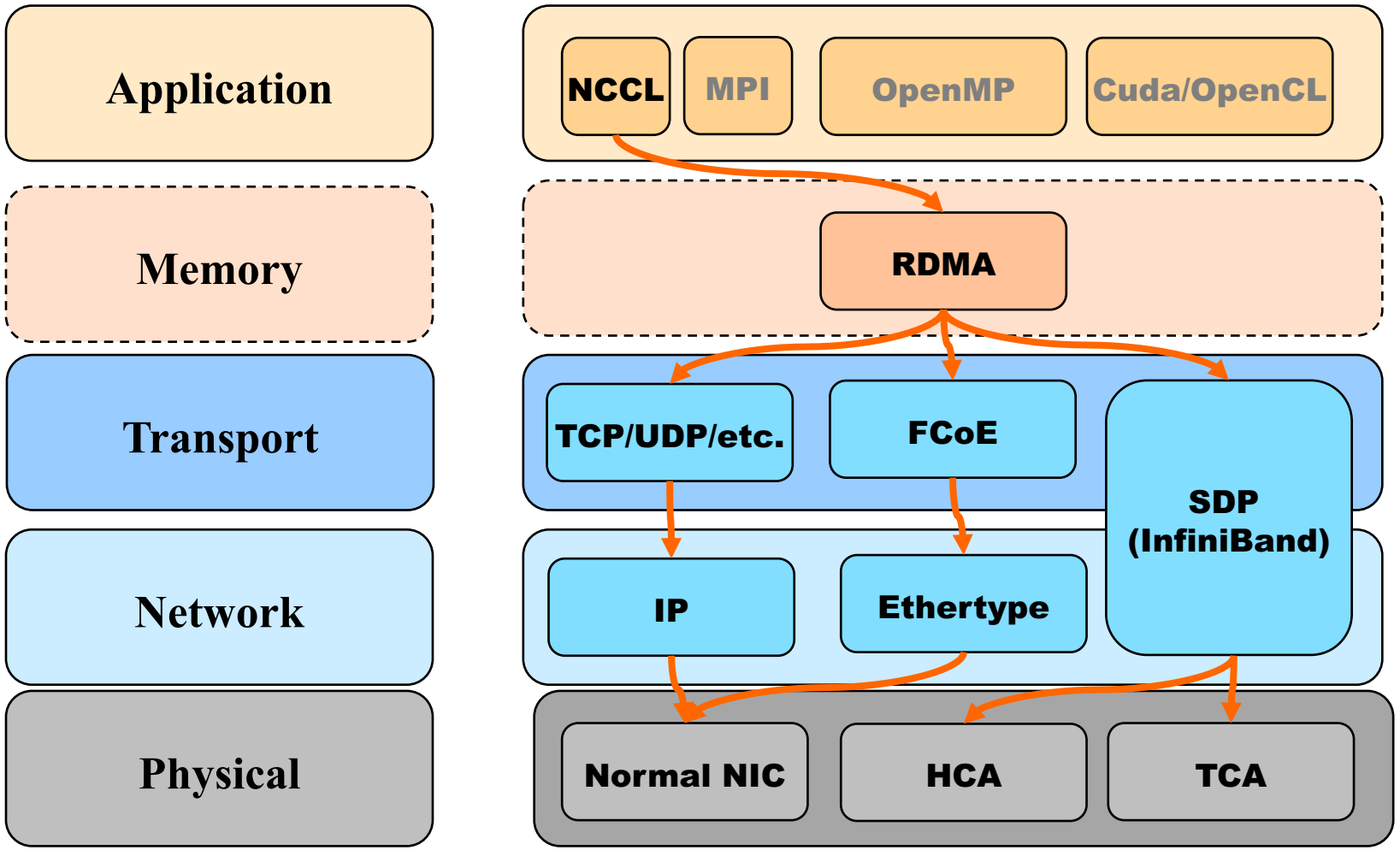
NCCL (Data Transfer)

- Based on Cuda (parallel computing platform and API model)
- Similar to MPI (MPI functions)
- Within one machine:
  - Without data copy to main memory, i.e. direct memory access (DMA)
  - CUDA supports direct access only for GPUs of the same model sharing a common PCIe root hub
- Within a cluster:
  - Data copy to main memory
  - RDMA, TCP/IP etc.
  - GPU Direct (V2.3)



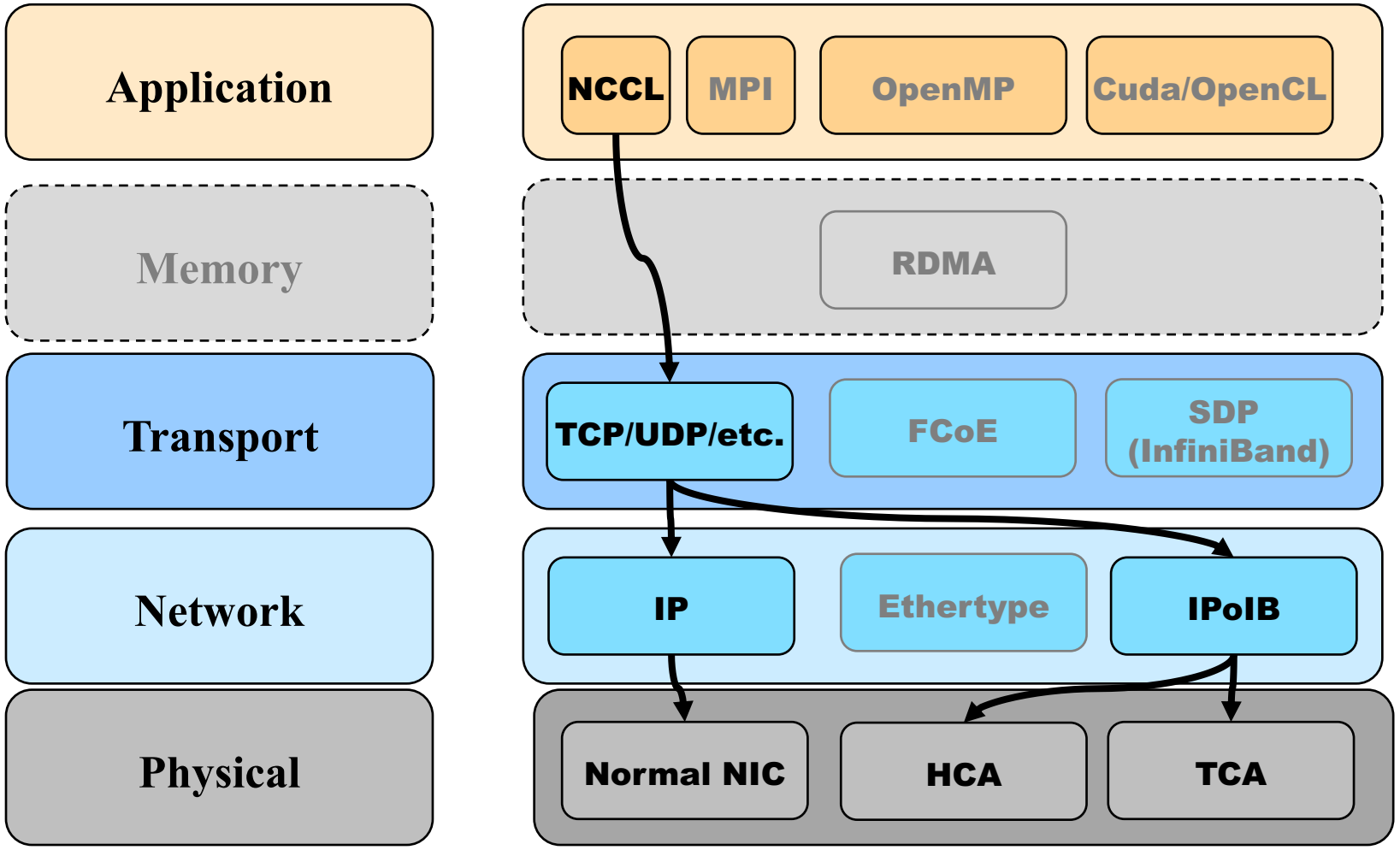
Data Transfer (With RDMA) for NCCL

- No/few CPU interaction and no data copy for data transfer



Data Transfer (Without RDMA) for NCCL

- CPU interaction and data copy for data transfer

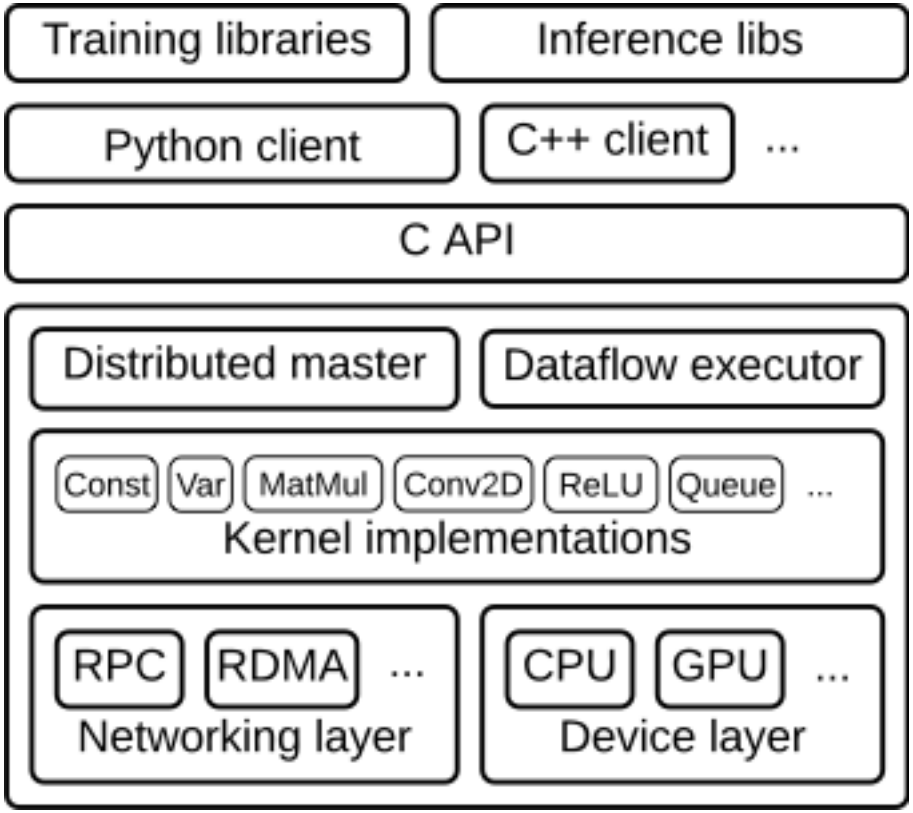


Outline

- High Performance Computing (HPC)
- **Systems for Deep Learning**
- Problems in terms of System

TensorFlow

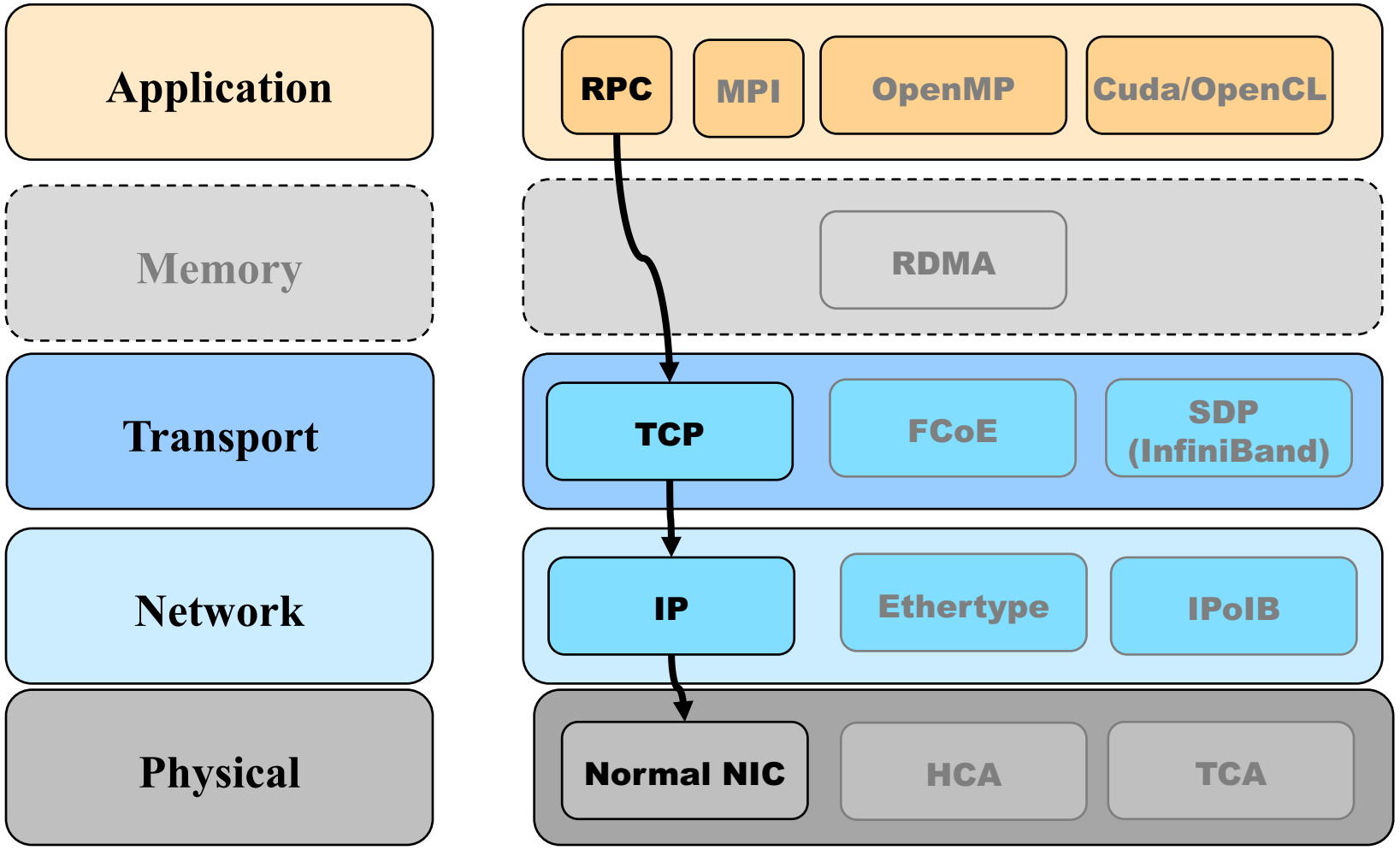
- At application layer
- Based on RPC or RDMA etc.
- MPI-enabled TensorFlow*
- Master slave architecture
- Strategies
  - Synchronized
  - Asynchronized



* https://github.com/tensorflow/tensorflow/tree/master/tensorflow/contrib/mpi

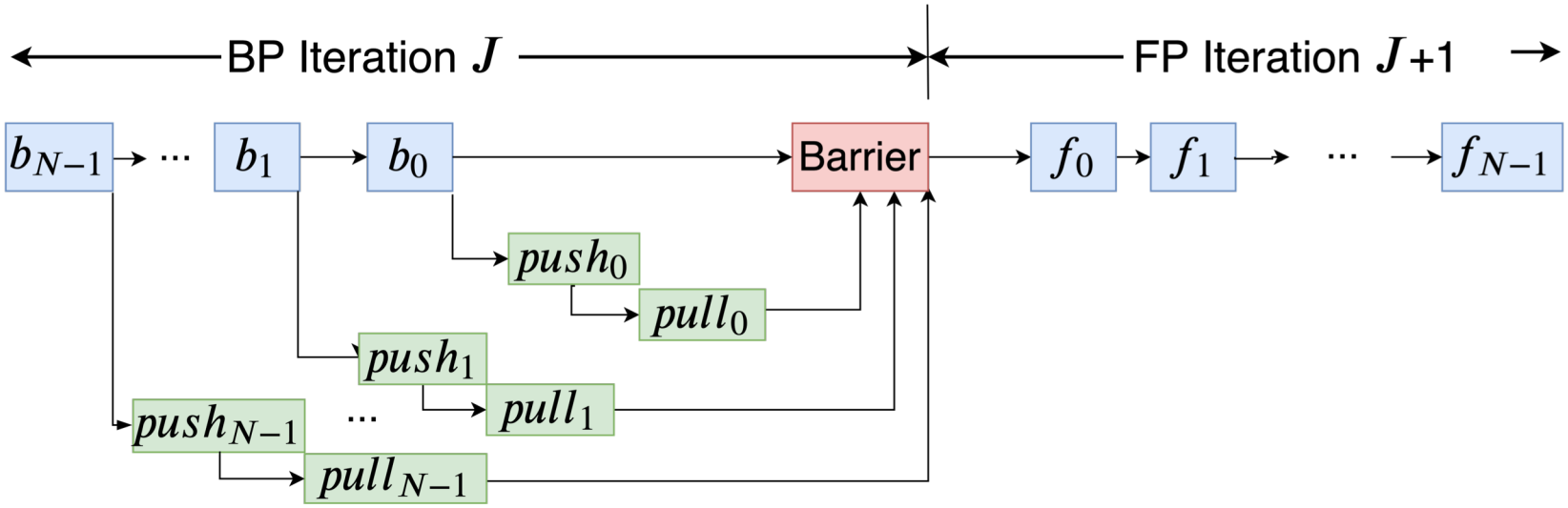
RPC Data Transfer

- CPU interaction and data copy for data transfer



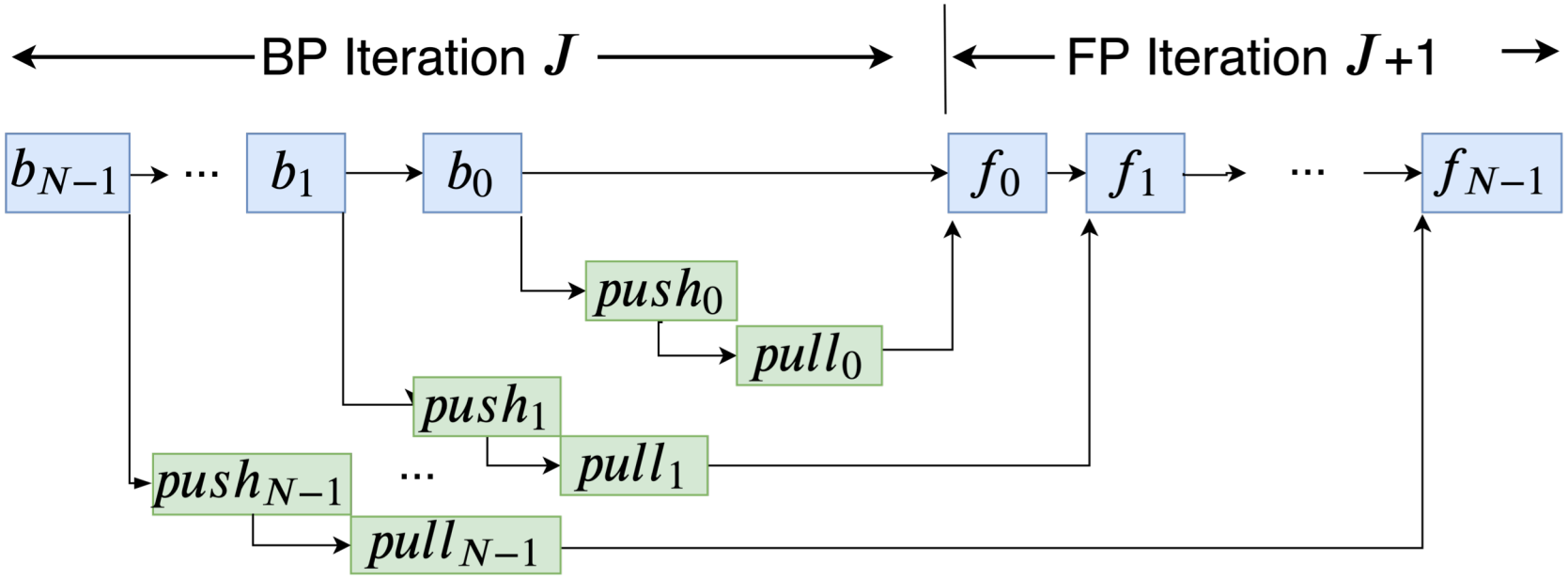
Strategies in TensorFlow

- Synchronous training
  - All workers train over different slices of input data
  - Aggregate gradients at each step
  - Via all-reduce

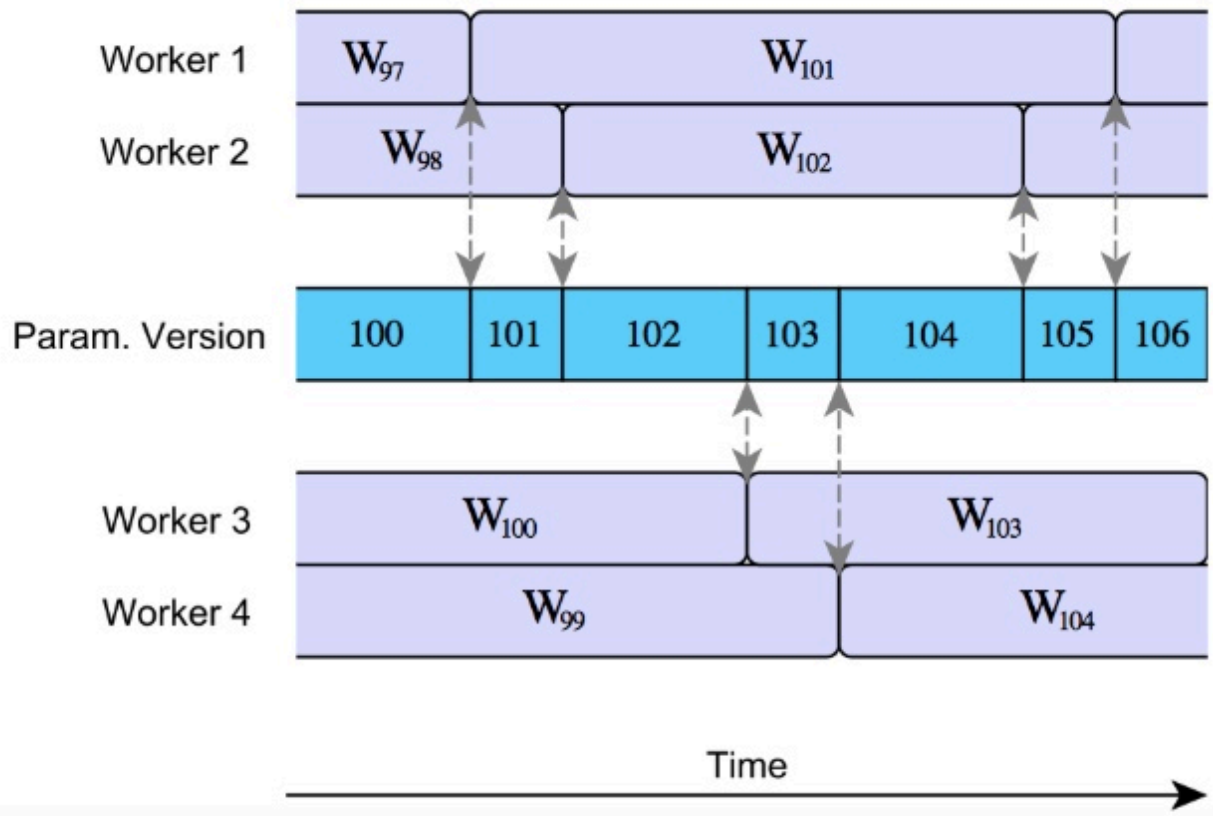


Strategies in TensorFlow (Continue)

- Asynchronous training
  - All workers are independently training over the input data
  - Updating variables asynchronously
  - Parameter server



Stale gradient problem



$$\vdots$$

$$W_{101} = W_{100} - \lambda \Delta W_{97}$$

$$W_{102} = W_{101} - \lambda \Delta W_{98}$$

$$W_{103} = W_{102} - \lambda \Delta W_{100}$$

$$W_{104} = W_{103} - \lambda \Delta W_{99}$$

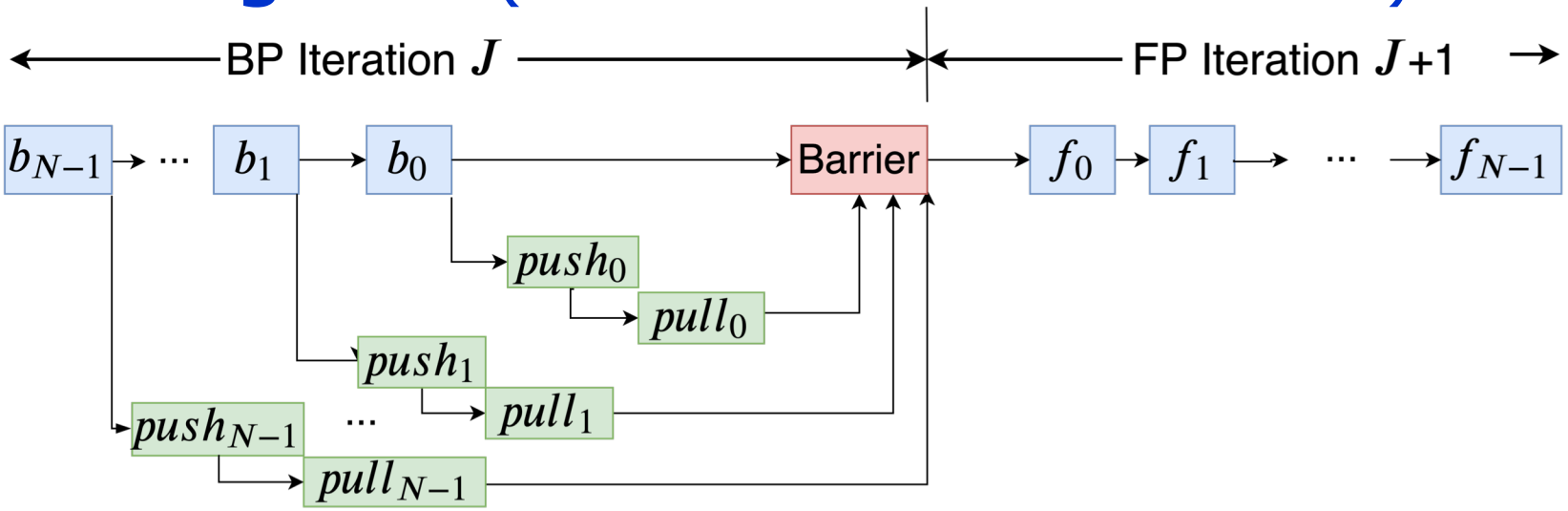
$$\vdots$$

Soft Synchronization Protocols

- Combination of synchronized and asynchronized training
- Resolves stale gradient problem
- Asynchronized update within one or several iterations
- Synchronization based on threshold

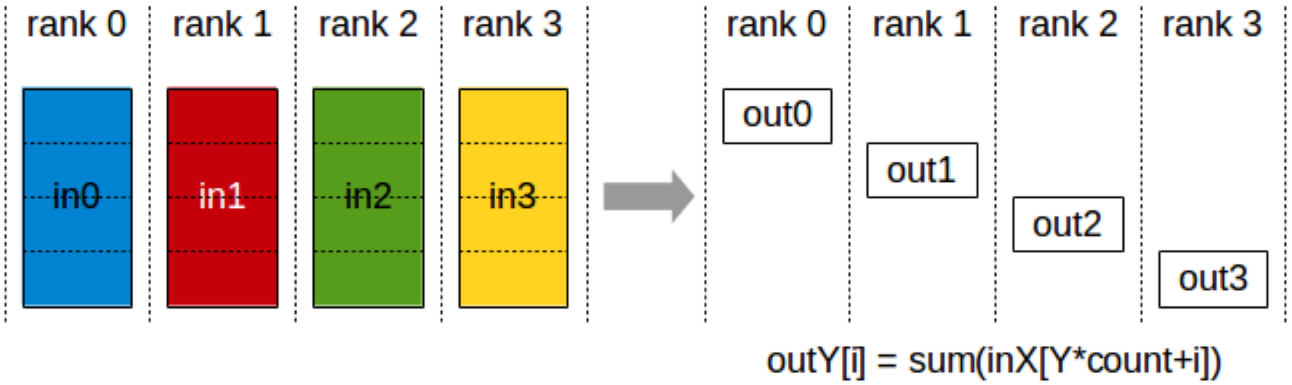
Horovod

- At application layer
- Simplify the development in TensorFlow
- Based on MPI, NCCL (depends on implementation) or Gloo (TCP/IP, optional: NCCL, MPI)
- Synchronous distributed training using **allreduce** and **allgather** (in Horovod and TensorFlow)

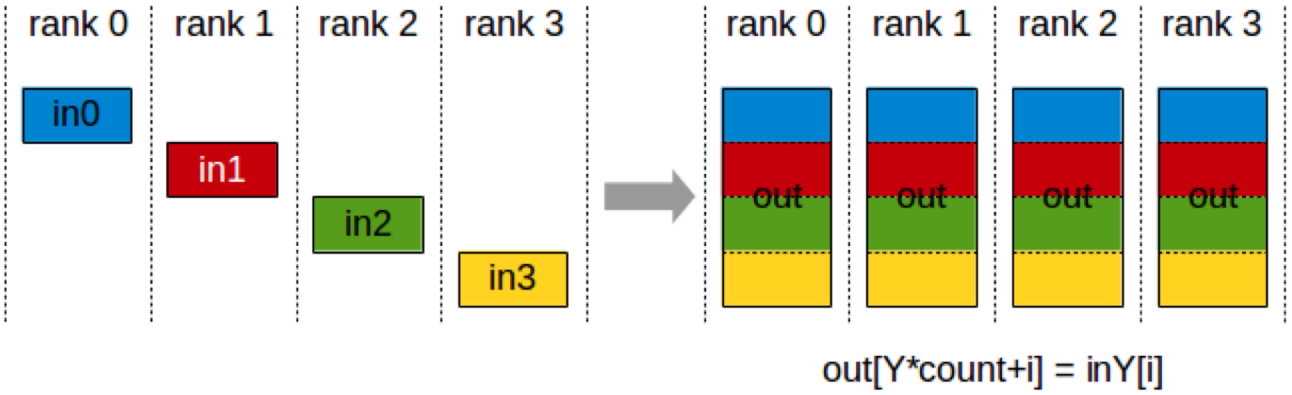


Byteps

- Local communication
  - NC
    -
  - Lo
    -
  - Lo
    - AllGather (in NCCL)



- Bas
  wor
- Asy



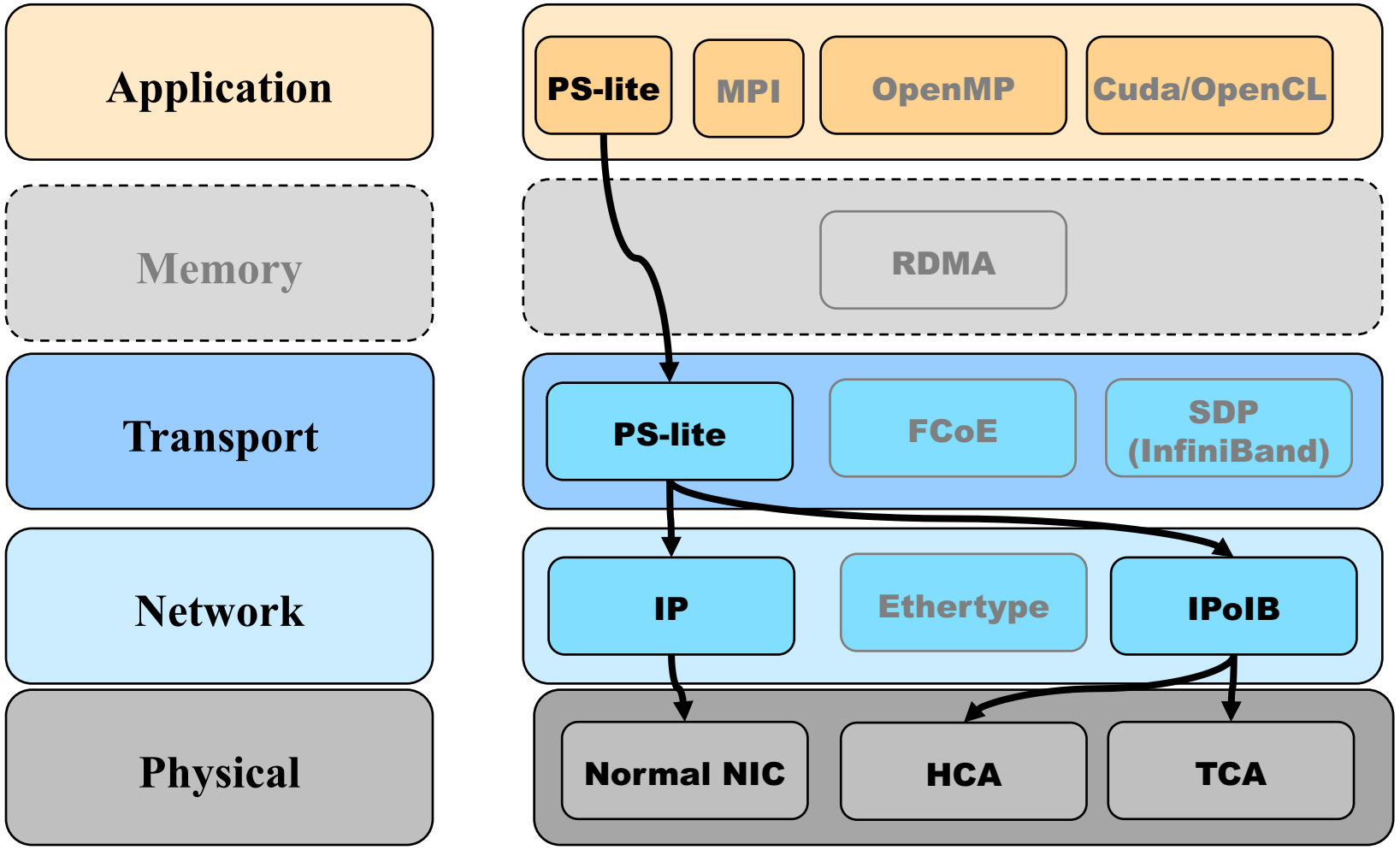
PS-Lite *

- Light and efficient implement of parameter server framework
- Push(keys, values): push a list of (key, value) pairs to the server nodes
- Pull(keys): pull the values from servers for a list of keys
- Wait: wait until a push or pull finished
- Works above the network layer
- Based on: IP, Supports infiniBand

*** Li et al. Scaling Distributed Machine Learning with the Parameter Server. OSDI, 2014**
Li et al. Communication Efficient Distributed Machine Learning with the Parameter Server. NIPS, 2014

PS-lite Data Transfer

- CPU interaction and data copy for data transfer



Outline

- High Performance Computing (HPC)
- Systems for Deep Learning
- **Problems in terms of System**

Problems in Terms of System

- Decentralized architecture
- Fault-tolerance?
- Check point?
- CAP? How to ensure
  - Consistence, availability and network Partition
  - Eventual consistency?