The Future of AI Networks:

Advancing TCP with Device Memory & Collective Communication

Shaopeng He, Anjali Singhai, Sridhar Samudrala
Netdev 0x18, July 2024

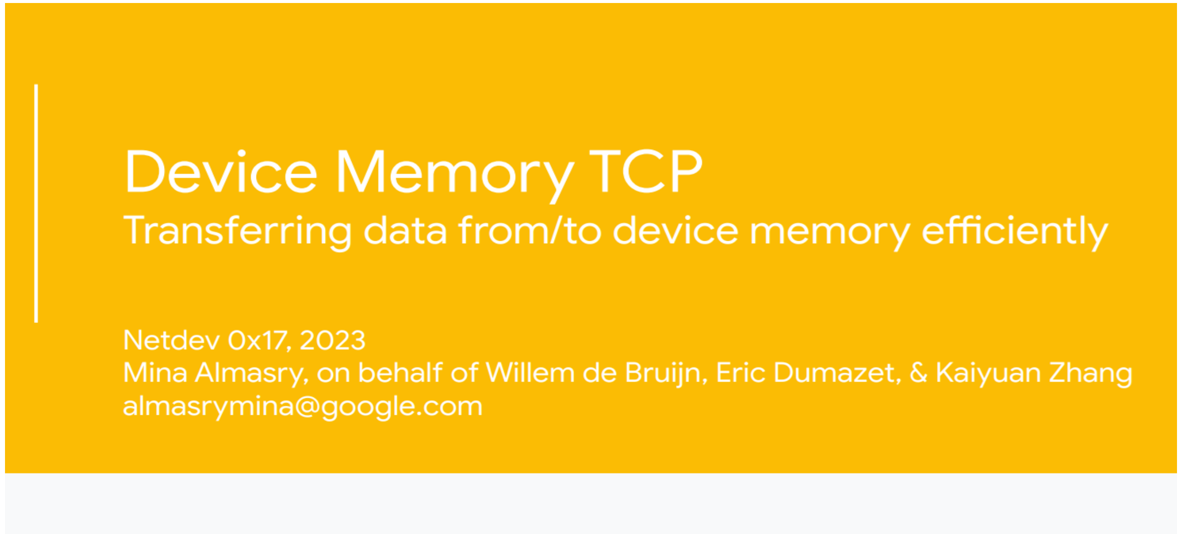
intel.

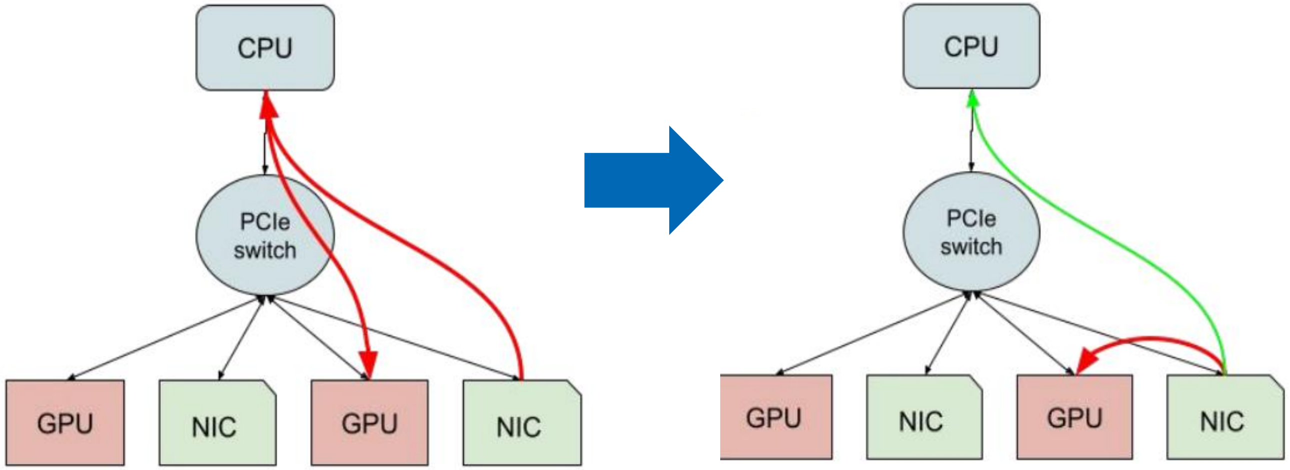
Agenda

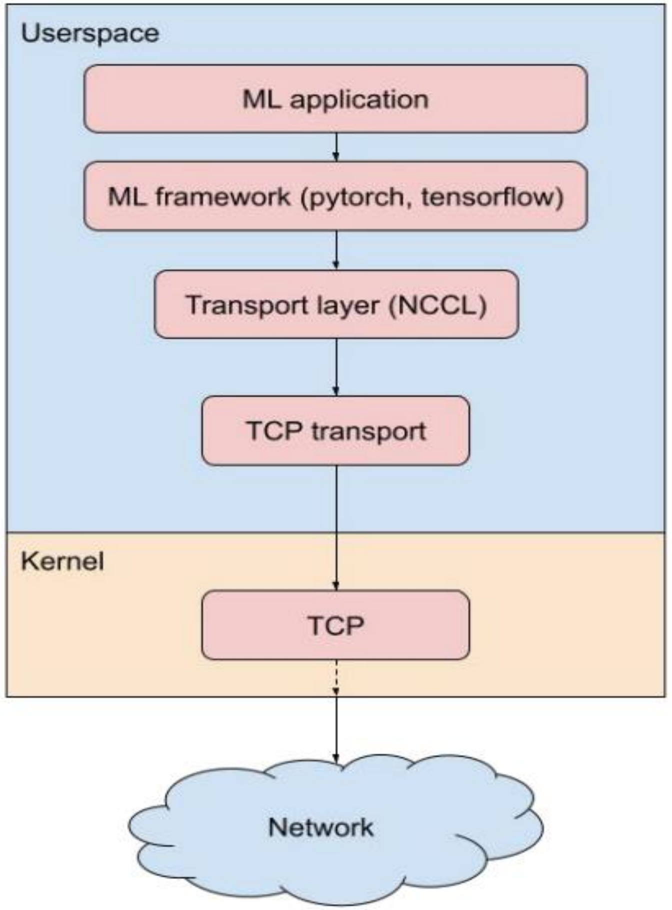
- Device Memory (DevMem) TCP Recap
- PoC Topology & Results
- One Page of Status & Plan
- Why is TCP Still Relevant for AI Networks
  - Challenges with the TCP Protocol
  - Another Perspective on Network: Semantics (fancy name for API)
  - Key Semantics for AI Networks
  - Contributions of DevMem TCP to Semantics
- Looking into “The Future of AI Networks”

  --- Long Live the TCP Semantics

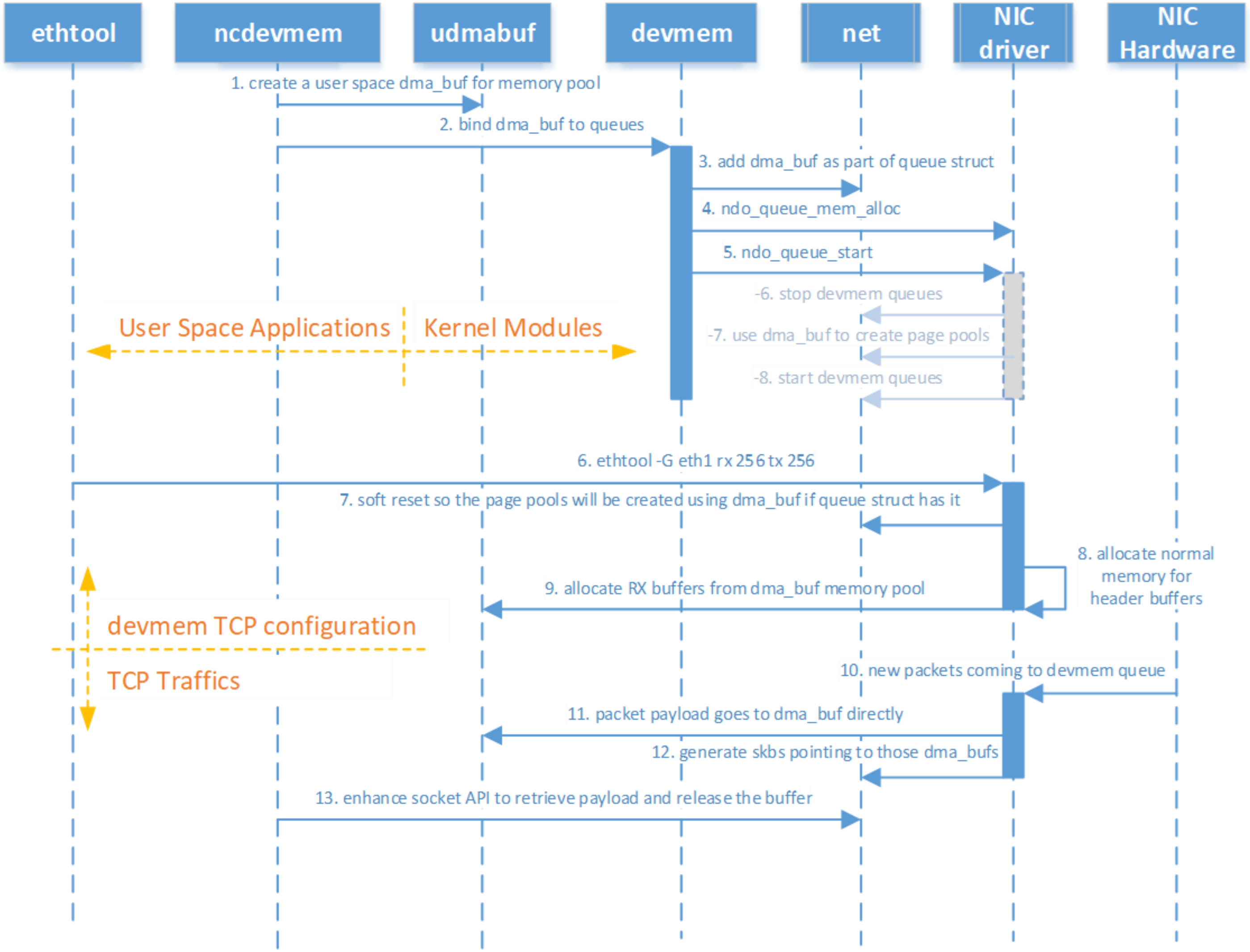
Recap: DevMem TCP framework from Google





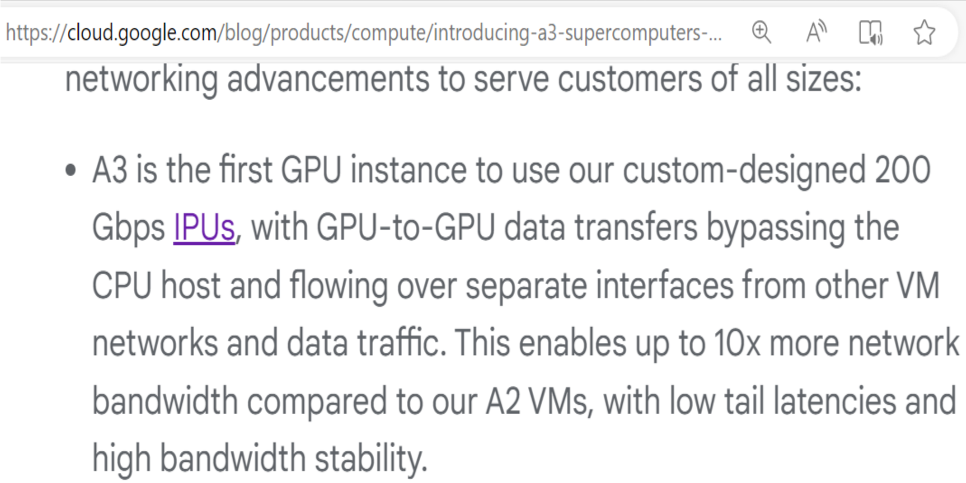


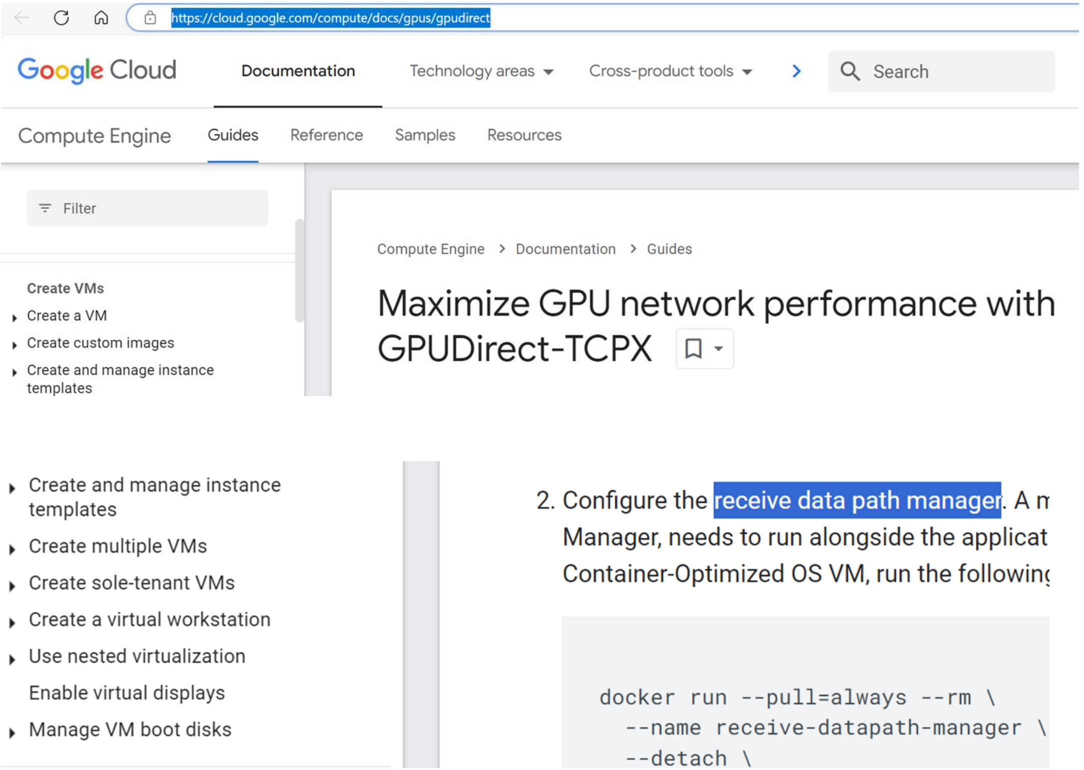
Recap: Upstream DevMem TCP RX Patches



- devmem kernel changes:
  - queue management API (Step 4, 5) which already merged
  - net modification (step 3), to be merged
- Depending on NIC driver capability, gve implementation uses steps -6, -7, -8), idpf PoC uses steps 6,7,8,9
- ncdevmem is a self testing tool for DevMem TCP from RFC patches, which includes socket API operations, dmabuf and queue management.

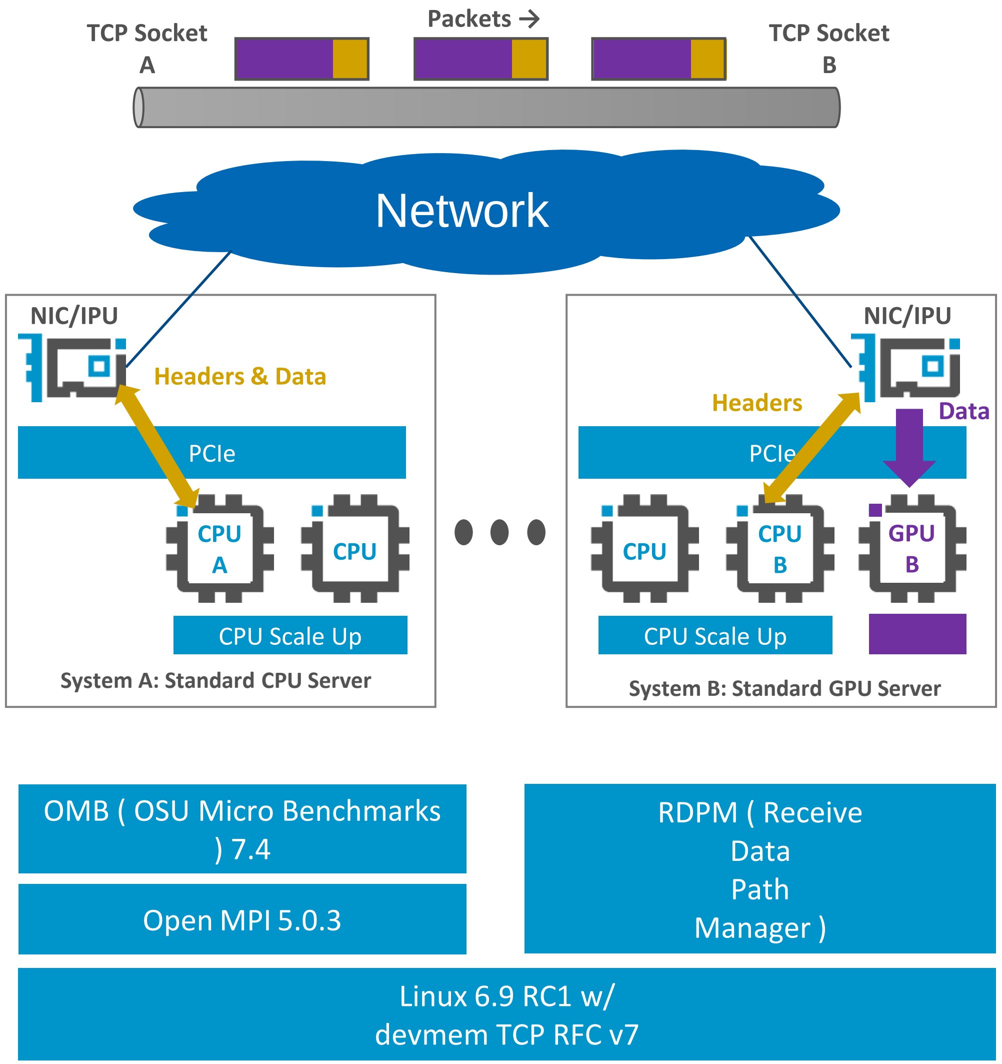
Recap: Real-World Usage of DevMem TCP





- Google A3 with Intel IPU: [Introducing A3 supercomputers with NVIDIA H100 GPUs | Google Cloud Blog]
- GPU Direct TCPx: https://cloud.google.com/compute/docs/gpus/gpudirect
- Intel IPU has advanced features like ATE (Address Translation Engine) which acts as device side MMU (Memory Management Unit) solution for TCP, similar to RDMA's private MMU design.

PoC Topology & Design Overview



- OMB and Open MPI were enhanced with devmem TCP capability. GPU memory and IPU queue management logics are in RDPM.
- As TX path not ready, System A doesn't use GPU and devmem TCP, which may affect overall performance comparing to RDMA because of the memory copy between user space and kernel for normal TCP.
- Open MPI devmem TCP PoC reuses normal TCP implementation, not optimized as RDMA which has a separate GPU Direct RDMA module in addition to normal RDMA, which also affects the performance.

PoC MPI Benchmark Results

- Normal TCP to GPU Command: mpirun --mca pml ob1 --mca btl tcp,sm,self --mca btl_tcp_if_include ens9f0d1 -n 2 --host vb,va /opt/osu_bw -m 512:512 D H
- DevMem TCP to GPU Command: mpirun --mca pml ob1 --mca btl tcp,sm,self --mca btl_tcp_if_include ens9f0d1 -n 2 --host vb,va /opt/osu_bw -m 512:512 H H
- DevMem TCP PoC has ~3x performance over normal TCP, larger packet size tests are more restricted by current single queue design and other kernel parameters.

One-Page DevMem TCP Status for TL;TR

- Google
  - open-sourced RX side kernel patches with gve driver.
  - related product performance on par with RDMA for large packets.
  - published product design for NCCL and RDPM, not open-sourced yet.
- Intel implemented RDPM (Receive Data Path Manager) PoC (Proof of Concept) with IDPF/IPU driver and integrated into MPI systems, both are ready to open-source soon.
- Product level open-source complete implementation based on IDPF (standard driver) and open-sourced user space plugins (from Google, Intel etc.) to be available no later than end of year.

DevMem TCP Future Plan

- Kernel Tx side support and improve the MPI benchmarks performance
- User NCCL (DevMem TCP) plugin Open sourcing and integration for usage outside of GCP
- User RDPM Open Sourcing and integration for usage outside of GCP
- Support on Other Intel NICs
- Innovations for reordering design
- Direct Signal: send signal to NIC from GPU
- Application Direct (Meta and Google collaborating)

Agenda

- ✓ DevMem TCP Recap
- ✓ PoC Topology & Results
- ✓ One-Page of Status & Plan
- ➢ Why is TCP Still Relevant for AI Networks
  - Challenges with the TCP Protocol
  - Another Perspective on Network: Semantics (fancy name for API)
  - Key Semantics for AI Networks
  - Contributions of DevMem TCP to Semantics
- Looking into “The Future of AI Networks”

  --- Long Live the TCP Semantics

Challenges with the TCP Protocol

It's Time to Replace TCP in the Datacenter

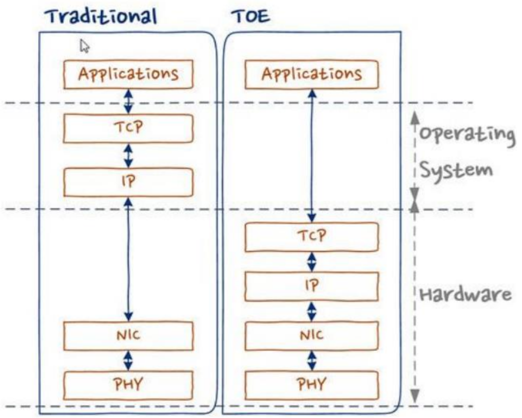
John Ousterhout
Stanford University

PLATFORMLAB

https://netdevconf.info/0x16/keynote/netdev0x16-keynote.pdf

The brief history of TCP Offload

- TOE: TCP Offload Engines, offload TCP to specialized hardware
- Promise was great performance
- TOE was popular 20 years ago
- TOE startups, Windows Chimney
- Open Onload, run TCP stack in userspace, socket intercept using LD_PRELOAD



https://netdevconf.info/0x17/docs/netdev-0x17-paper42-talk-slides/netdev-0x17-paper42.pdf

- Discussions about replacing TCP often focus on substituting the TCP Protocol with innovative new protocols like Homa.
- A primary challenge for TCP Protocol is the difficulty of Offloading; still lag behind the full hardware protocol like RDMA over RoCE and Falcon especially for smaller packets.
- Datacenters primarily operate two types of networks; today, we will focus on the first:
  - AI: Collective Communication
  - others: gRPC etc. Point to Point Communication

Another Angle for Network: Semantics (API)

https://www.mpi-forum.org/docs/mpi-4.1/mpi41-report.pdf

Classification of Collective Operations

MPI_BARRIER:	Synchronisation
MPI_BCAST:	Send from one process to all processes
MPI_GATHER:	gather data from all processes on one process
MPI_SCATTER:	scatter data from one process to all processes
MPI_ALLGATHER:	gather data from all processes, broadcast them to all processes
MPI_ALLTOALL:	exchange data between all processes
MPI_REDUCE:	reduction over all processes, result goes to one process
MPI_ALLREDUCE:	reduction over all processes, result is broadcasted to all processes
MPI_REDUCE_SCATTER:	reduction over all processes, result is scattered to all processes
MPI_SCAN, MPI_EXSCAN:	process i receives result from reduction over processes with j<= i, j<i

https://hps.vi4io.org/_media/teaching/summer_term_2022/pchpc_mpi_collective_slides.pdf

- Semantics here refers to a programming paradigm, including API and the meaning of related concepts like Operations, Procedures and Datatypes in MPI SPEC Chapter 2.4
- Message Passing Interface (MPI) is the default communication Semantics (API) for HPC (High Performance Computing) using distributed computing engines like CPU and GPU.
- Message Passing Interface (MPI) v1 SPEC was released in 1994, defined Point-to-Point and Collective Communication (CC). **No** definition for protocol, only Semantics (API)
- AI/ML could be seen as one type of HPC. First NCCL GitHub release v1.2.1 at 2016, implemented most of the CC operations defined in MPI.

Another Angle for Network: RDMA Semantics

10.7.2.2 RDMA

There are two types of RDMA: RDMA Read and RDMA Write.

InfiniBand^{SM} Trade Association   Page 526   Proprietary and Confidential

InfiniBand^{TM} Architecture Release 1.3   **Software Transport Interface**   March 3rd 2015
VOLUME 1 - GENERAL SPECIFICATIONS   FINAL

RDMA Read Operations are supported only on the three reliable Transport Service Types—Reliable Connection, Extended Reliable Connected and Reliable Datagram. RDMA Write Operations are supported on the three reliable Service Types plus the Unreliable Connection Service Type.

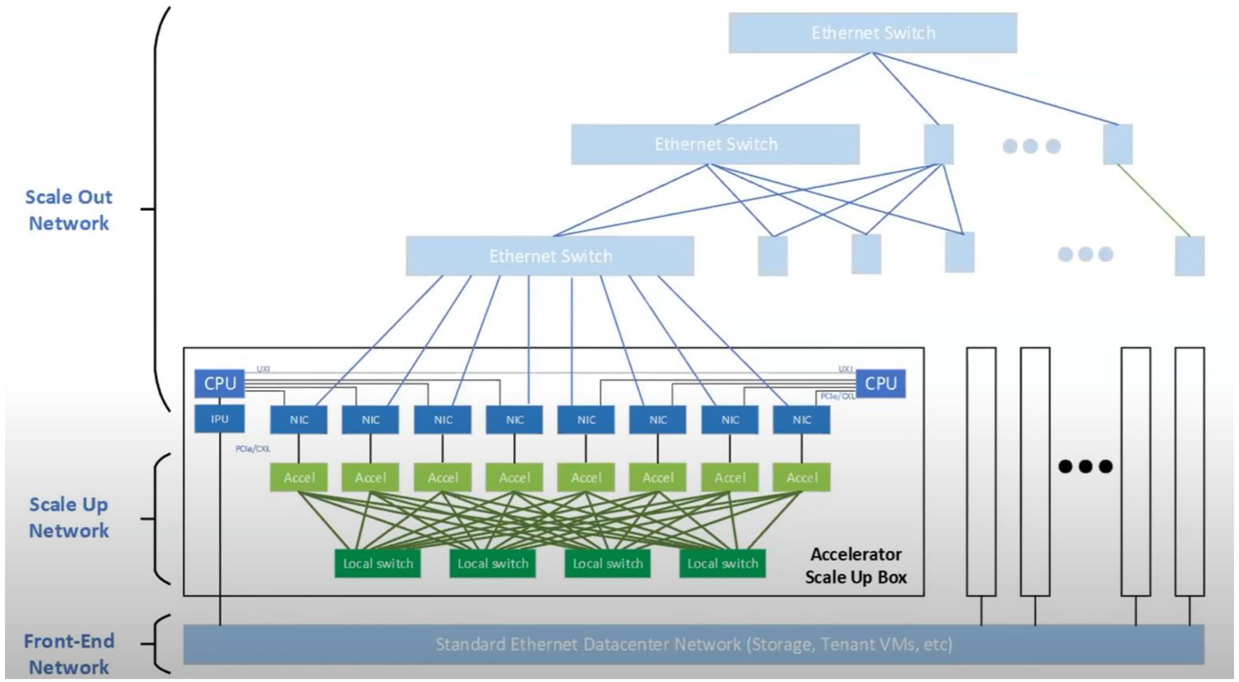
C10-83: The CI **shall** support RDMA Read Operations on the RC Transport Service Type.

C10-84: The CI **shall** support RDMA Write Operations on the RC and UC Transport Service Types.

o10-39: If the CI supports RD Service, the CI **shall** support both RDMA Read and Write Operations on the RD Transport Service Type.

- RDMA was first defined as an InfiniBand high level protocol, then as binding protocol over Ethernet (RoCE v1) and UDP (RoCE v2).
- When binding to TCP as iWARP, and AWS EFA, it’s largely becoming a Semantics (API)
- It has several key Semantics listed in next page and becomes most popular for AI Networks

key Semantics for AI Networks



HOTI 2023 - Day 1: Session 4: Panel - EtherNET vs EtherNOT … again? (youtube.com)

- Reliable Transport
  - Scale Up: shared memory for CPU or CXL, NVLink, UALink
  - Scale Out: TCP or RDMA
  - Front-End: TCP
- Device Direct
  - GPU Direct RDMA
  - devmem TCP
- Application Direct
  - zero copy (similar to RDMA Memory Semantics)
  - direct signal from GPU to network

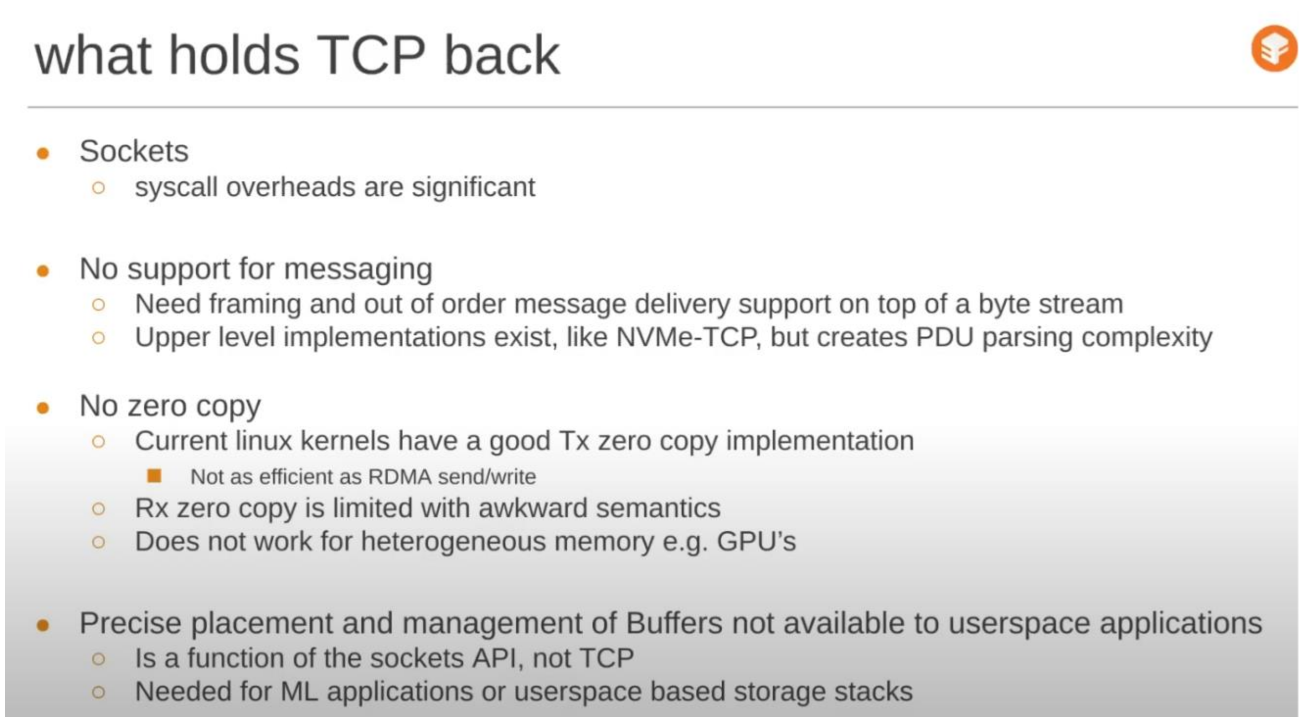
What’s devmem TCP contributed for Semantics?

```
msg.msg_iov = &iov;
msg.msg_iovlen = 1;
msg.msg_control = ctrl_data;
msg.msg_controllen = sizeof(ctrl_data);
ret = recvmsg(client_fd, &msg, MSG_SOCK_DEVMEM);
printf("recvmsg ret=%ld\n", ret);
for (cm = CMSG_FIRSTHDR(&msg); cm; cm = CMSG_NXTHDR(&msg, cm))
    if (cm->cmsg_level != SOL_SOCKET ||
        (cm->cmsg_type != SCM_DEVMEM_DMABUF &&
         cm->cmsg_type != SCM_DEVMEM_LINEAR)) {
        fprintf(stdout, "skipping non-devmem cmsg\n");
        continue;
    }

    dmabuf_cmsg = (struct dmabuf_cmsg *)CMSG_DATA(cm);
    is_devmem = true;
```

- TCP Semantics is defined by Socket API.
- new TCP Semantics for devmem TCP RX side includes
  - new MSG_SOCK_DEVMEM flag for recvmsg
  - new SCM_DEVMEM_DMABUF and LINEAR message type
- TX side, Direct Signal etc. to be expected

Why it works now (, not before)?



what holds TCP back

- Sockets
  - syscall overheads are significant
- No support for messaging
  - Need framing and out of order message delivery support on top of a byte stream
  - Upper level implementations exist, like NVMe-TCP, but creates PDU parsing complexity
- No zero copy
  - Current linux kernels have a good Tx zero copy implementation
    - Not as efficient as RDMA send/write
  - Rx zero copy is limited with awkward semantics
  - Does not work for heterogeneous memory e.g. GPU's
- Precise placement and management of Buffers not available to userspace applications
  - Is a function of the sockets API, not TCP
  - Needed for ML applications or userspace based storage stacks

[2023 OFA Virtual Workshop: RDMA and Linux TCP (youtube.com)]

- With the header split feature, the network payload of the kernel protocol (controlled by the Socket API) can be directed to heterogeneous memory, such as GPU memory.
- This has been proven to deliver incredibly good performance, comparable to GPU Direct RDMA, according to Google's real product deployment.
- Although the sockets syscall still incurs significant overheads, it is separated from the data path by the header split feature, and does not affect overall performance, especially for larger packets.

AI Networking: Two paths, teasing out the pillars

Key pillars for AI Networking	RDMA	DevMem TCP	Comments
User visible Application APIs			
Application Semantics	RDMA verbs	TCP/Socket API	For AI applications, memory semantics needed which RDMA has, but TCP is way more popular and got small enhancements to acquire the same.
Infrastructure and Transport Invisible to the user			
Device (GPU) Direct	PCIe P2P	PCIe P2P	TCP semantics also need a Device MMU or ATS like RDMA
Reliable Transport	Falcon	TCP	Transport of choice depending on Edge AI vs Cloud Front End vs Backend Network
Congestion Control	Swift/HPCC++ etc	Swift/HPCC++ etc	Inband Telemetry CSIG/IFA is independent from Transport Protocol
Crypto Offload of choice	PSP	PSP	
Transport offload	Specialize Protocol Engines	Embedded generalized Cores	
Multipathing and out of order handling	ECMP, packet spraying, relaxed ordering with Falcon	ECMP and out of order support	
Very very tight Tail latency for scale out	Yes	Yes, using Application/device Directed queues and flow director	

Long Live the TCP Semantics

- The current landscape of AI networking is too limited with only one RDMA Semantics, which is predominantly controlled by a single entity and originates outside of the Ethernet world.
- With robust TCP Semantics for AI networks, Cloud Service Providers (CSPs) can modify the underlying protocol without users noticing.
- Other Potential Protocols for such TCP Semantics include:
  - Innovative new protocols like Homa.
  - Long-standing vendor protocols like AWS's EFA.
  - Existing public protocols like Falcon.
  - newly proposed public protocols like UEA.