SPDK ACTIVITIES IN NVIDIA

2021 SPDK VIRTUAL FORUM

ALEXEY MARCHUK, SPDK CORE MAINTAINER

AGENDA

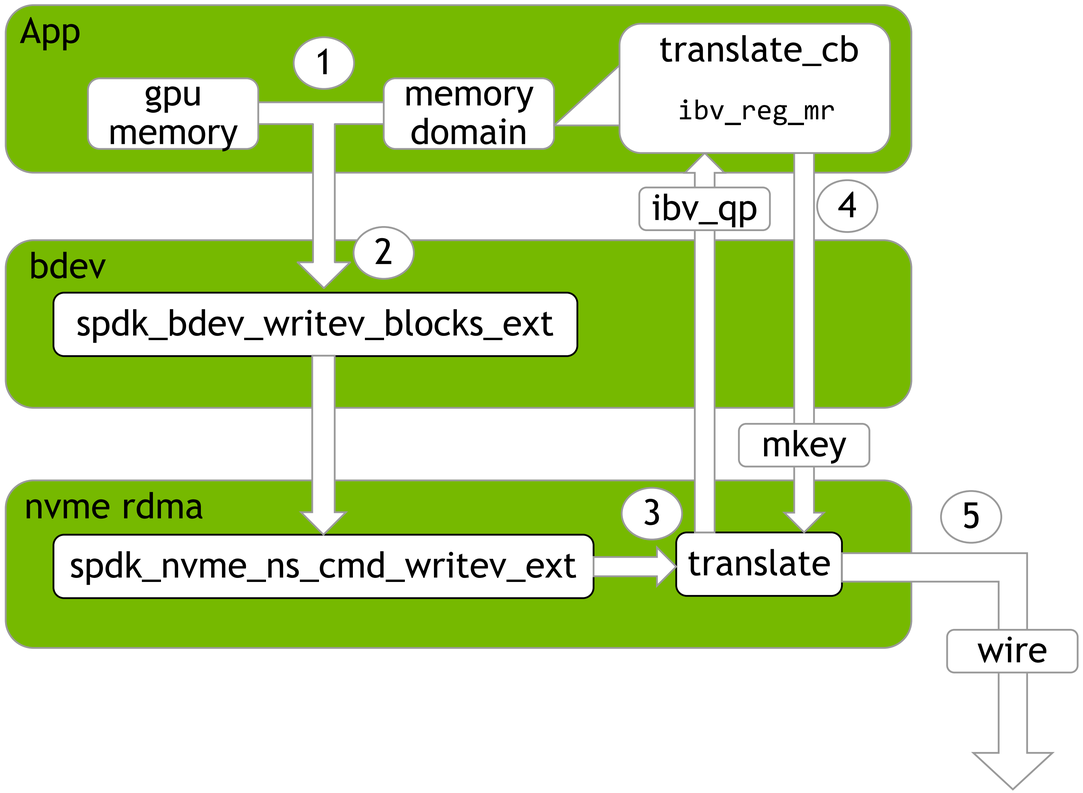
- SPDK activities in NVIDIA
- Memory domains
- NVMe-oF TCP performance improvements
- SPDK in large scale
- HW accelerations and offloads

SPDK ACTIVITIES AT NVIDIA

- Contribution: 137 patches merged in 2021
- Verification system:
  - Per patch set verification – focus on ARM platform and network protocols
  - Nightly verification – extended scope, covers more features and SPDK components
  - Performance regression
- SPDK is part of SW stack in BlueField DPU - DOCA
  - SPDK allows to customize IO processing in BlueField DPU

MEMORY DOMAINS

- Allows to work with IO buffers that are not accessible on CPU
- Query memory domains supported by bdev
- Attached to bdev IO request (optional)
- Translate data from app memory domain to SPDK memory domain
- Provides interfaces to implement pull/push data functionality between memory domain and local memory
- Suitable for different use cases:
  - GPUDirect RDMA
  - SNAP NVME-oF RDMA

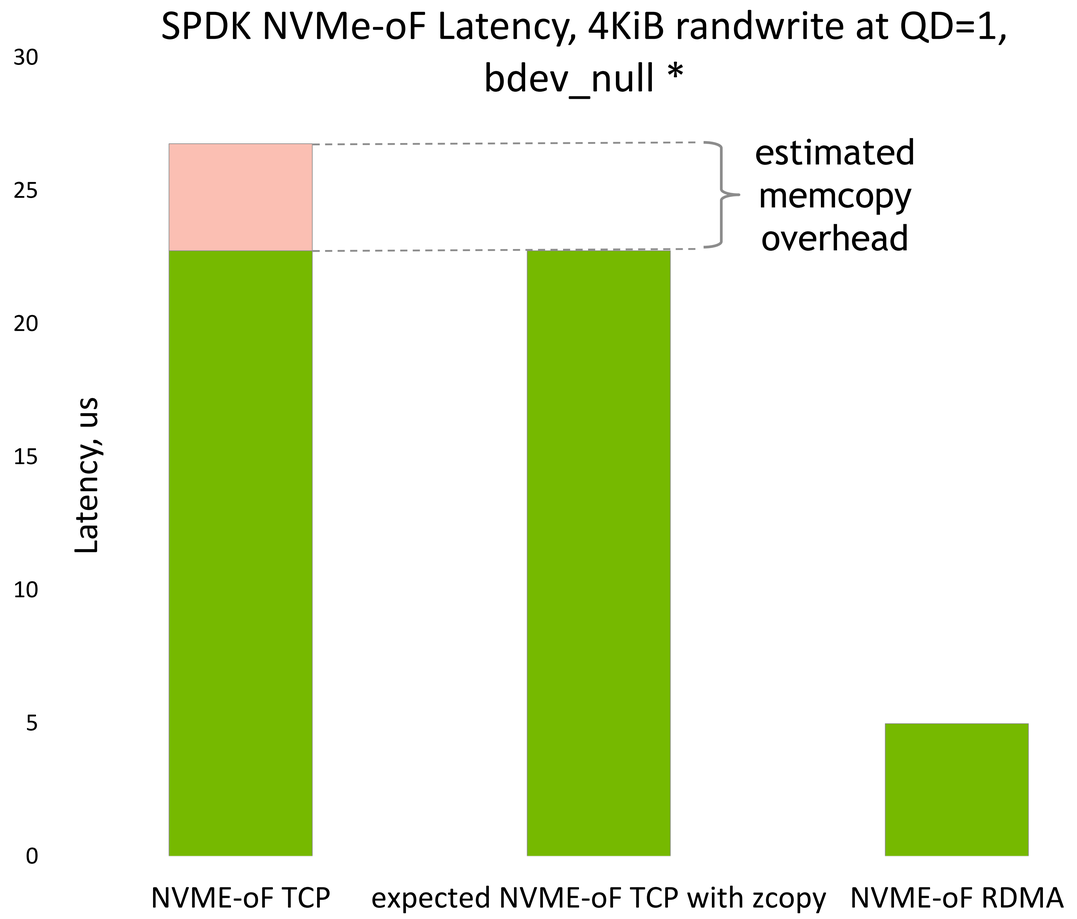


NVME-OF TCP IMPROVEMENTS

NVME-OF TCP IMPROVEMENTS

Latency challenge

- NVME-oF RDMA overperforms NVME-oF TCP: latency, CPU usage
- Memory copy overhead increases with payload size
- Focus on NVME-oF TCP performance improvements: SW optimizations, HW offloads
- XLIO (next generation of VMA) – user space TCP/IP stack

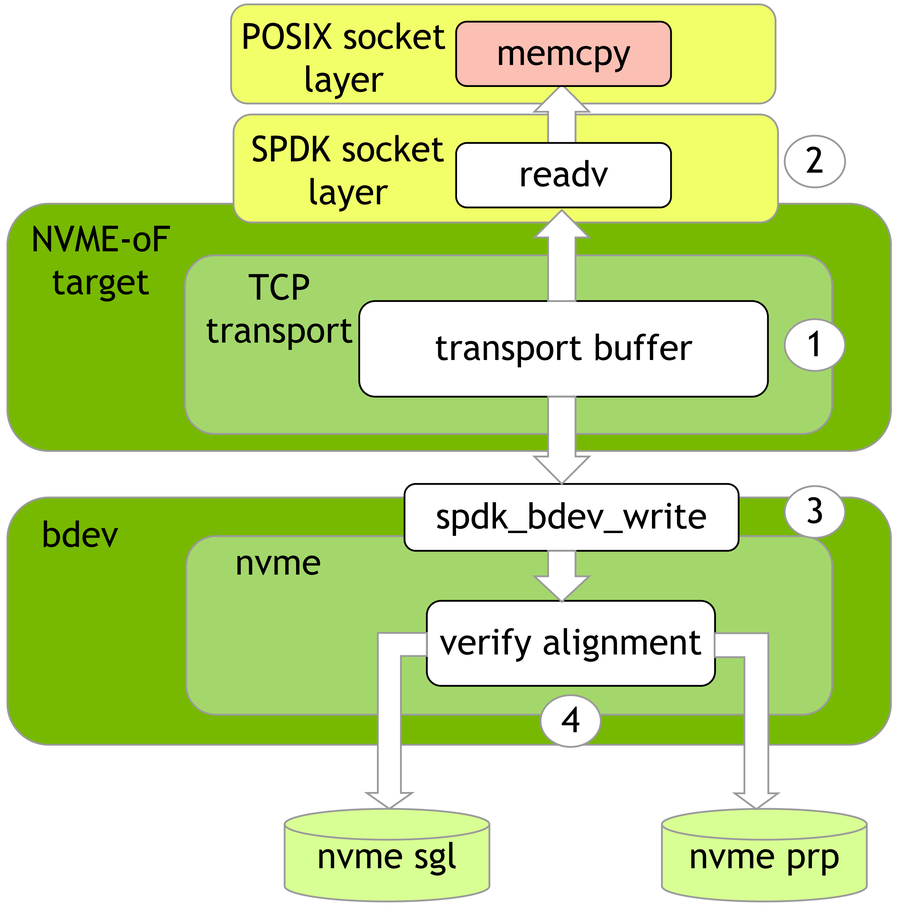


* Based on SPDK 21.10 performance reports

TCP IMPROVEMENTS

NVME-oF TCP target “Write” flow current implementation

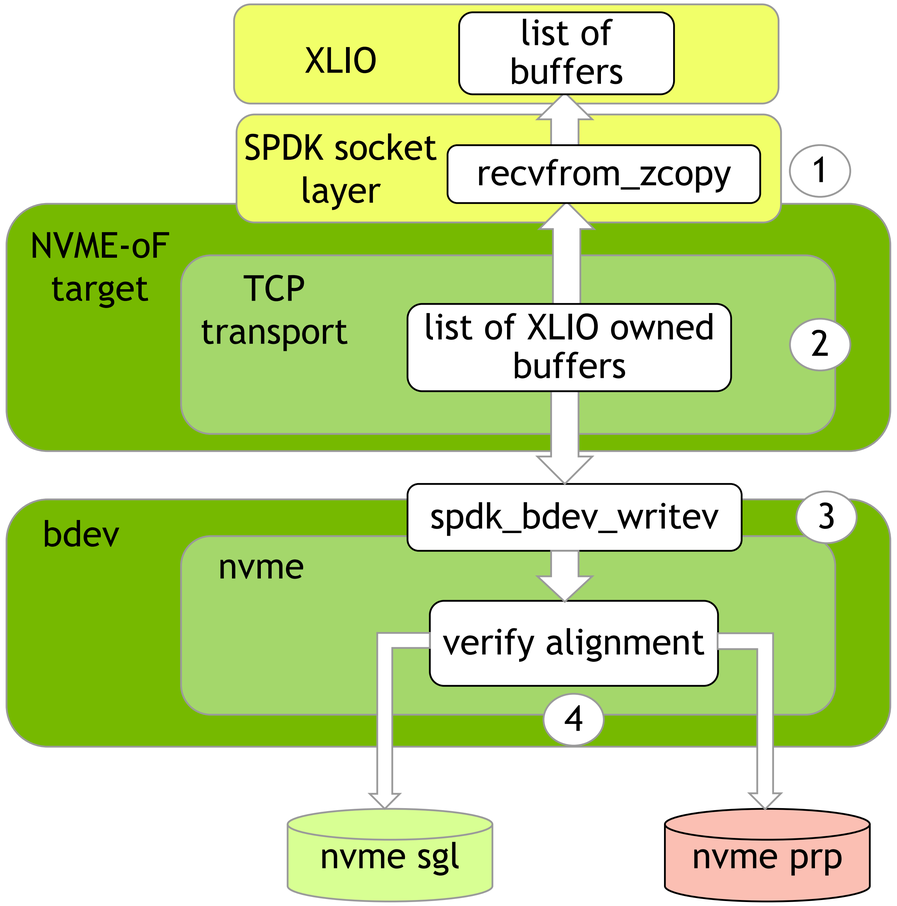
- SPDK NVME-oF transport manages pool of buffers
  - In TCP transports buffers are contiguous and 4KiB aligned
- TCP/IP socket layer (POSIX) copies data received from the network to transport buffer
- NVME disks that don’t support SGL have strict requirements on buffers alignment



TCP IMPROVEMENTS

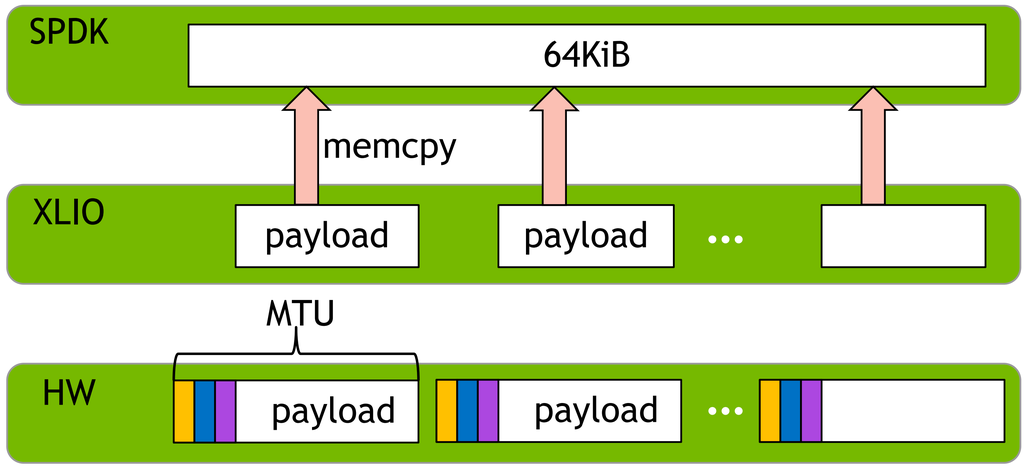
NVME-oF TCP target “Write” flow zero copy

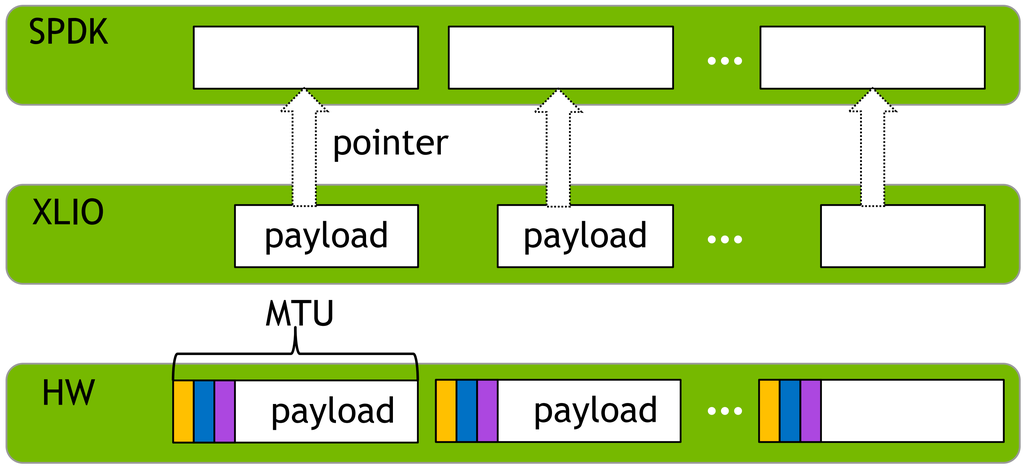
- SPDK NVME-oF TCP transport requests socket layer to provide buffers with data
- Socket layer provides 2 APIs – zero copy and regular readv
- Data is returned as a list of buffers
  - IO request payload can be fragmented
  - Dedicated socket layer API to release buffers



TCP IMPROVEMENTS

Buffers fragmentation challenge



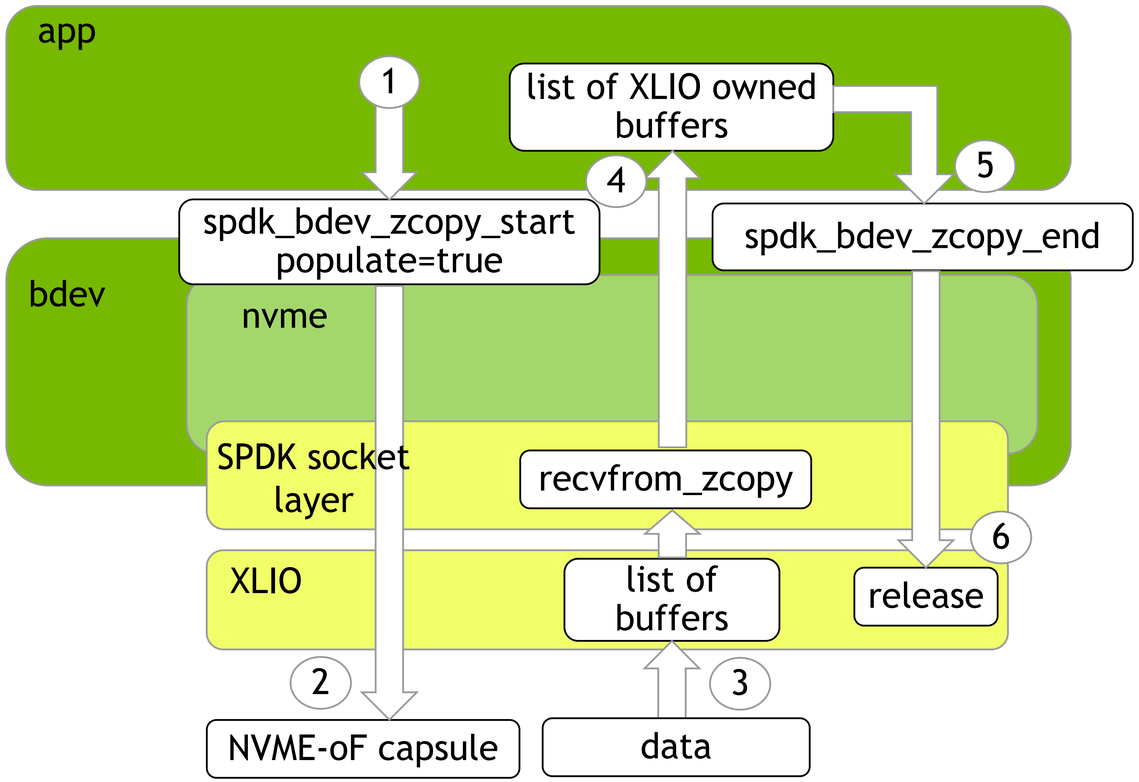


- Problem: received data buffers are fragmented by (MTU – (TCP + IP + NVME-oF headers)) bytes
  - On the target side such buffers are not suitable for most of NVME disks (which don’t support SGL)
  - LRO (Large Receive Offload) reduces fragmentation, but data buffers still don’t suit PRP requirements
- Solution: ConnectX7 and BlueField3 will support NVME-oF TCP HW offload
  - In receive path it allows to store received NVME-oF headers and data to separate contiguous buffers

TCP IMPROVEMENTS

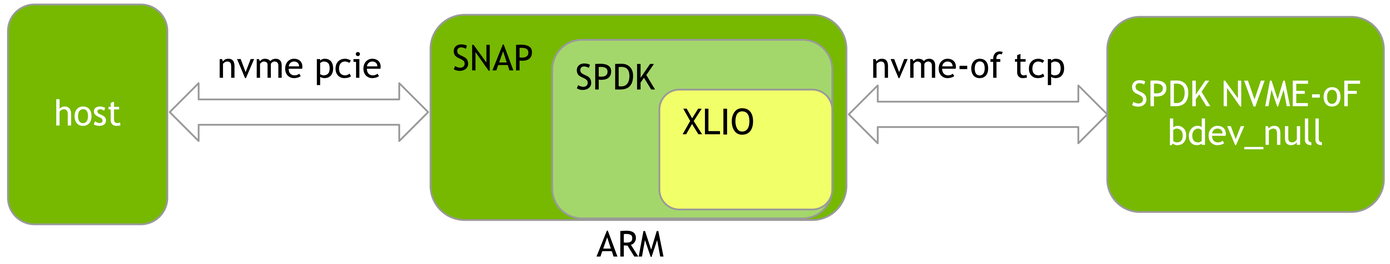
NVME-oF TCP initiator “Read” flow zero copy

- Implemented zero copy in receive in NVME-oF initiator
- Utilizes existing SPDK bdev zero copy start/end API
- SPDK API changes:
  - socket layer: new ops “recvfrom_zcopy” and “buffer_release”
  - nvme: new payload type “zero_copy”
  - bdev_nvme: implementation of zero_copy_start, zero_copy_end



TCP IMPROVEMENTS

NVME-oF TCP initiator “Read” flow zero copy results



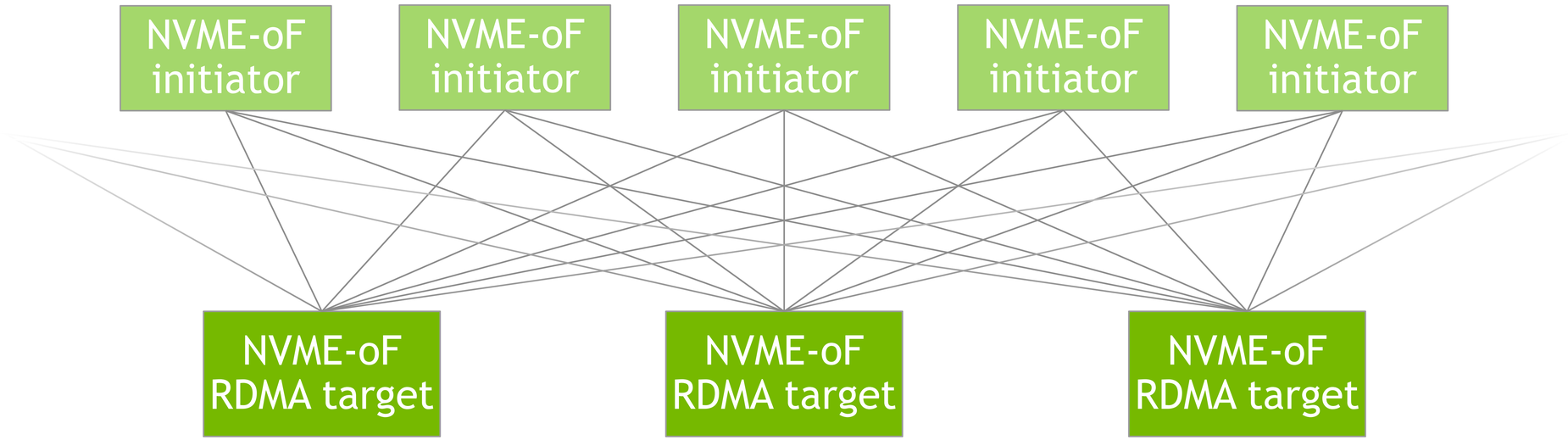
- 4KiB, QD=1: about **10%** lower latency
- 128KiB, QD=1: about **25%** lower latency
- Plan: start upstreaming in 2022

SPDK IN LARGE SCALE

SPDK IN LARGE SCALE

Challenge

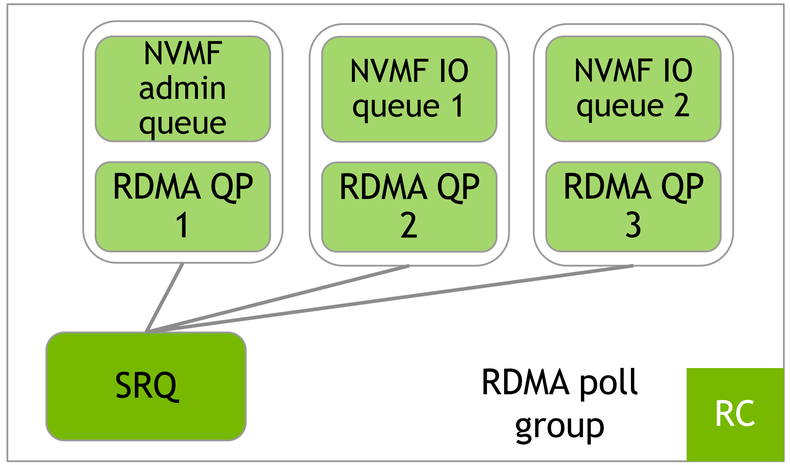
- Storage applications in HPC and ML require big scale
  - Direct communication of compute nodes with storage nodes via NVMe-oF RDMA
  - Layered storage solution - JBOF and storage controller. All-to-all connection pattern
- Challenge: thousands of NVME-oF connections per node

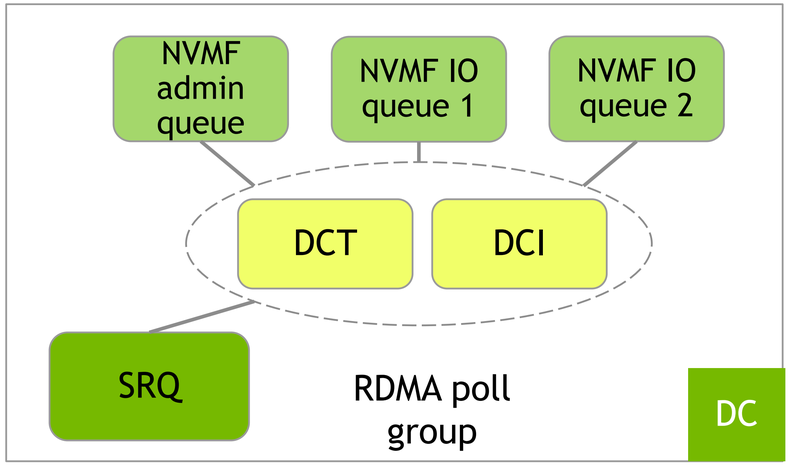


SPDK IN LARGE SCALE

NVMe-oF RDMA DC

- DC (Dynamically Connected) is another type of RDMA transport, combination of RC (Reliable Connection) and UD (Unreliable Datagram)
  https://docs.mellanox.com/display/rdmacore50/Dynamically+Connected+(DC)+QPs
- DC in SPDK: Uses mlx5_dv API, available as part of rdma-core or Mellanox OFED
- Target with 2K+ connections: DC shows 5% more IOPS than RC
  - 30 % lower memory consumption – constant memory footprint in DC case
- Plan: start upstreaming in 2022



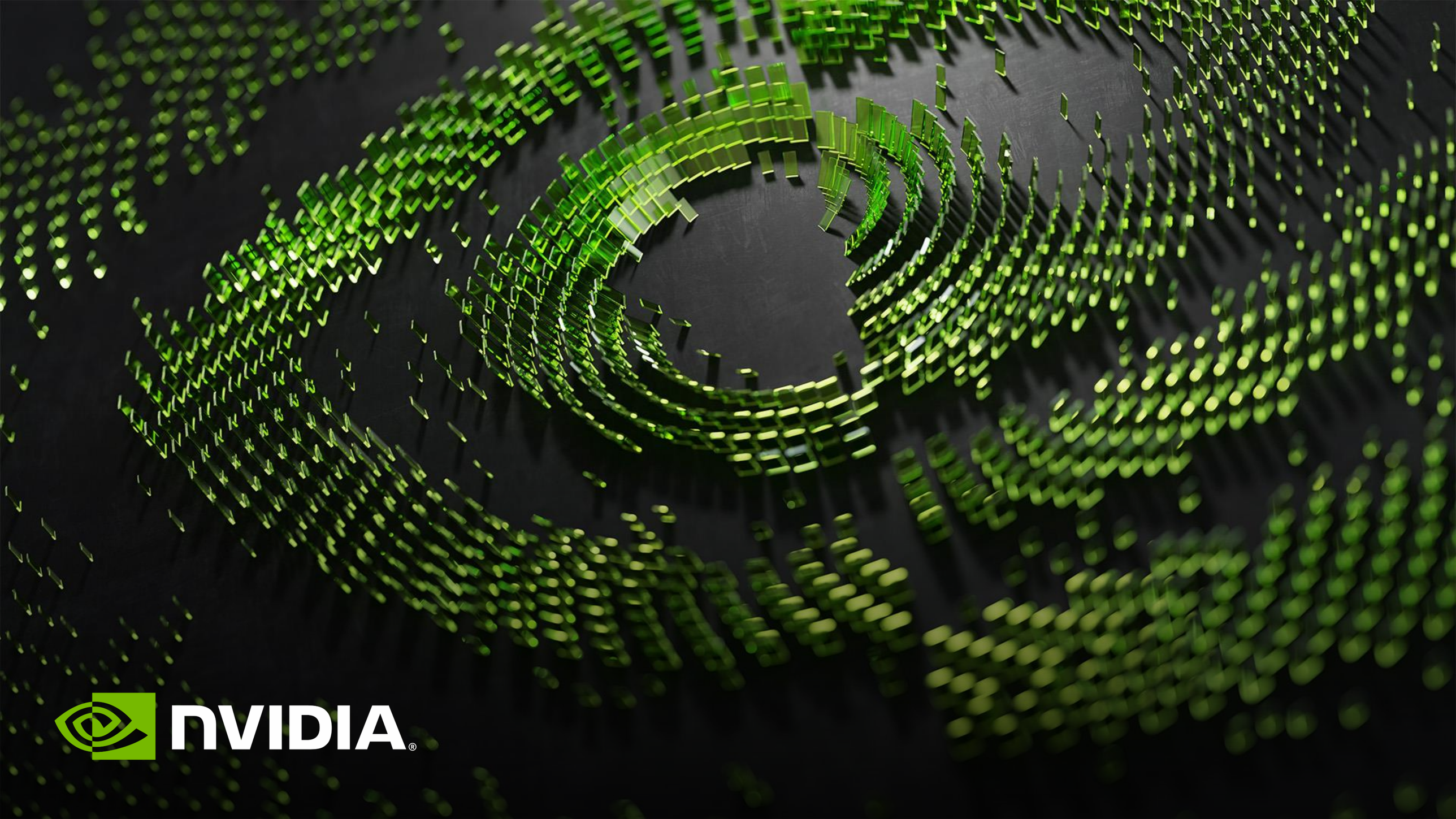


HW ACCELERATIONS AND OFFLOADS

- Two approaches to utilize HW acceleration:
  - “Look aside” – memory to memory operation in loopback qpair
  - “Inline” – memory-to-wire or wire-to-memory operation
- Acceleration types:
  - Compression/decompression. Available on ARM cores in BlueField. Available in SPDK 21.10 as “look aside” via DPDK
  - AES/XTS encrypt/decrypt (look aside) via DPDK driver, coming soon
  - AES/XTS encrypt/decrypt (inline), work in progress
  - T10DIF (inline) merged to rdma-core, work in progress

HW ACCELERATIONS AND OFFLOADS

- HW inline accelerations - set of properties on top of User Memory Region (UMR)
  - These properties are applied when data is processed by the NIC as part of RDMA operation
  - UMR registration – post special Work Request to ibv qpair. Lightweight operation, can be chained with Data WRs
  - UMRs can be created on top of each other, this allows to combine various HW accelerations
- NVME-oF RDMA target HW offload:
  - Fully offloads NVMe-oF target IO operations, reduces CPU utilization
  - Admin qpairs and IO qpair establishment are handled by SW
  - Connects NVME PCIe disk’s IO qpair (physical addresses of CQ/SQ doorbells) with NVME-oF IO qpair in HW


NVIDIA