Accelerating data movement between GPUs and storage or memory

S51142 | GTC S23 Mar 22, 2023, 1pm

Speakers

Chris J. Newburn (CJ), Distinguished Engineer, NVIDIA, cnewburn@nvidia.com
Magnum IO architect. Ph.D., Carnegie Mellon University 1997

Da Zheng, Sr. Applied Scientist, AWS, dzzhen@amazon.com
ML framework/algo leads; DGL/DGL-KE/DistDGL/TGL Ph.D. Johns Hopkins 2016

Harish Arora, Principal Product Manager, NVIDIA, hararora@nvidia.com
Magnum IO product mgr, Storage/Networking for DGX. MS IMT, MBA Cornell

Coming up…

New developments at NVIDIA related to accelerated storage IO

- Magnum IO
- GPUDirect
- Storage space innovation at NVIDIA
- GPUDirect Storage
- GPU-initiated storage
- Security

*

NVIDIA® Magnum IO

Optimized data movement, access, management
IO subsystem of the data center and edge

- Storage
  - Direct data path between storage and GPUs
  - Future: storage initiated from GPU
  - Future: increased security by moving the storage client from compute node to DPU
- Network
  - InfiniBand, Ethernet
  - MPI, NCCL, NVSHMEM, UCX, GDA KI Network
- Management
  - Network: UFM, NetQ
  - Future: data orchestration

NVIDIA GPUDirect™

Variants on who decides what to do and when to do it

Increasing autonomy for GPU

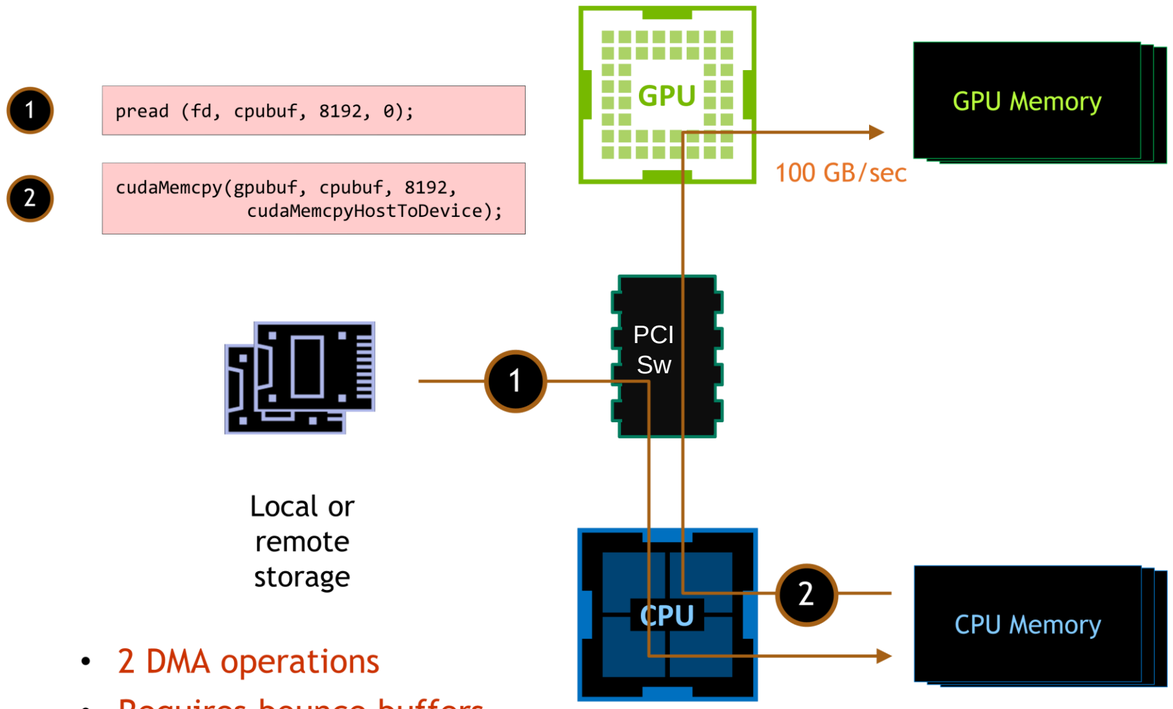
GPUDirect (non-Async)	GPUDirect Async	
CPU initiated (prepared, triggered)	**CPU prepared, GPU triggered**	**GPU kernel initiated**
Video: GPUDirect Video Local GPU: GPUDirect Peer-Peer P2P Remote master: GPUDirect RDMA GDR Storage: GPUDirect Storage GDS	Stream triggered GDA-ST Graph triggered GDA-GT Kernel triggered GDA-KT	GDA-KI Network NVSHMEM (blog) GPUNetIO (blog) GDA-KI Storage

- GPUDirect enables direct data movement to and from the GPU, without staging in CPU
  - As memory becomes migratable, the source/target may happen to be in GPU or CPU
- Network and storage IO involves
  - Preparation: create work request for an IO device, e.g. work queue entry on a cmd queue
  - Triggering: hand off work request to/sync with an IO device, e.g. ring a doorbell

NVIDIA Magnum IO GPUDirect™ Storage overview

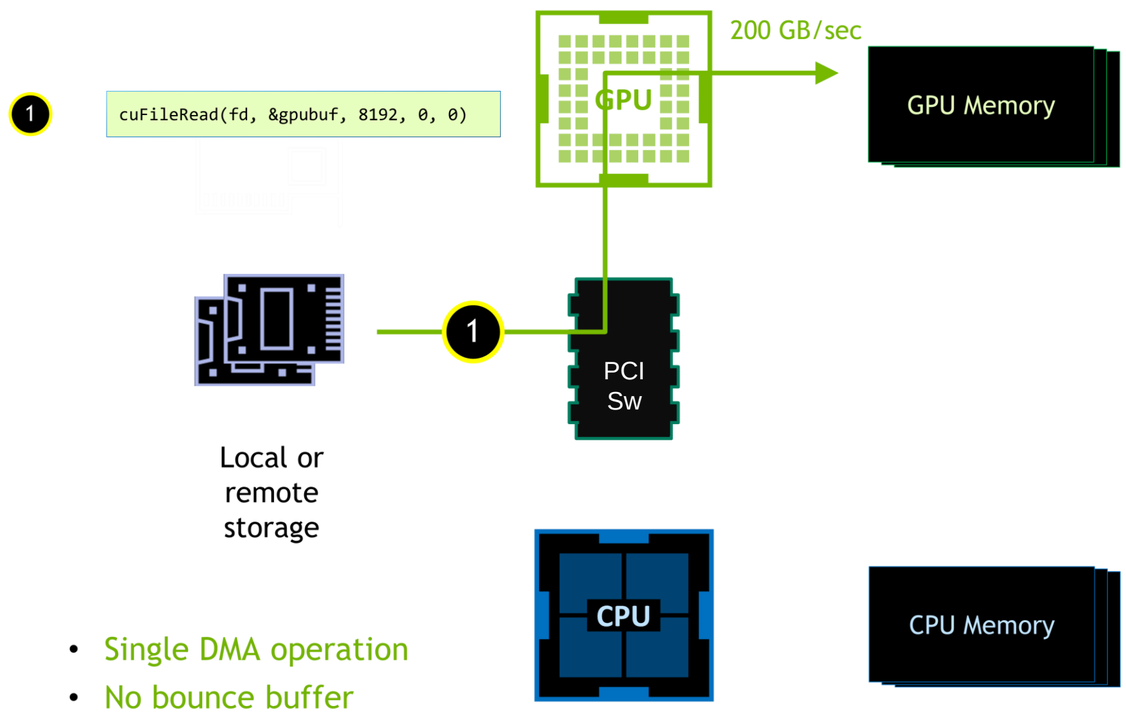
GPUDirect Storage adds File IO as part of CUDA

Traditional file system access



- 2 DMA operations
- Requires bounce buffers
- Halves CPU ⇔ PCI bandwidth

GPUDirect Storage access



- Single DMA operation
- No bounce buffer
- Independent of CPU ⇔ PCI
- Similar APIs as Posix
  - cuFileRead
  - cuFileWrite

NVIDIA Magnum IO GPUDirect Storage Partner Updates

The Partner ecosystem is expanding in response to customer demands

ALL GA PARTNERS

NVIDIA GPUDirect Storage integrated solution in production.

FILE SYSTEM PARTNERS

Partner company	Partner Product	GDS Version	Date
WekaIO	WekaFS 3.13	1.0	Jun-21
DellEMC	PowerScale 9.2	1.0	Oct-21
Hitachi Vantara	HCSF	1.0	Oct-21
IBM	Spectrum Scale 5.1.2	1.1 and higher	Nov-21
DDN	EXAScaler 6.0*	1.1 and higher	Nov-21
VAST	Universal Storage 4.1	1.1 and higher	Nov-21
NetApp	ONTAP 9.10.1	1.0 and higher	Jan-22
NetApp ThinkParQ System Fabrics Works	BeeGFS Tech Preview	1.1 and higher	Mar-22
IBM	Spectrum Scale 5.1.5	1.5 and higher	Dec-22
HPE	HPE Ezmeral 5.5**	1.5 and higher	Feb-23
PureStorage	FlashBlade®	1.7 and higher	Soon

*Open source Lustre 2.15 supports GPUDirect Storage

** Learn more about HPE Ezmeral by attending session S52331

NVIDIA Magnum IO GPUDirect Storage for Reverse Time Migration

Reverse Time Migration (RTM) use case summary

- A seismic migration technique
- Try NVIDIA's CPU accelerated RTM Implementation - Energy SDK Samples
- Workflow
  - Generate seismic images by computing forward & backward wave propagations on GPUs
  - Key Metric = Single shot migration time (forward propagation + backward propagation + imaging)
  - Single shot migration generates TBs of data that must be saved/written fast
    - Read back in reverse order during backward propagation

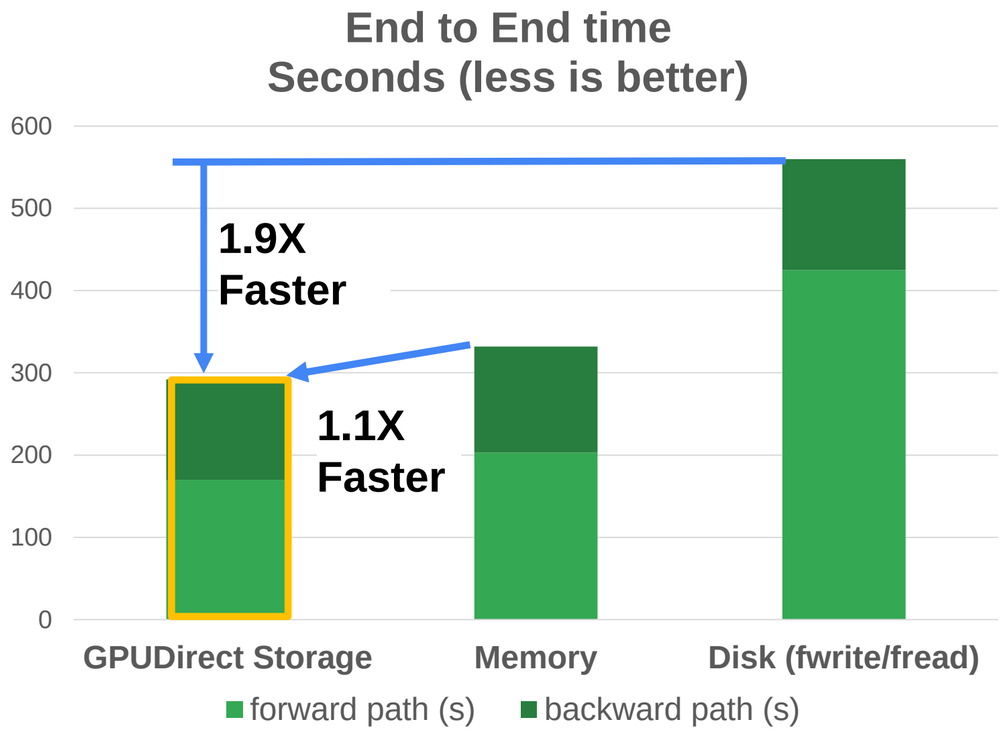
- Backends for storing/saving snapshots
  - System DRAM
  - On NVMe (local FS) via GPUDirect Storage
  - On NVMe (local FS) via POSIX

NVIDIA Magnum IO GPUDirect Storage for Reverse Time Migration

Reverse Time Migration (RTM) results

Small Problem Size

imageGrid.size.y=201 on 2 GPUs + 2 NVMe
Total snapshot data size = 1.3TB

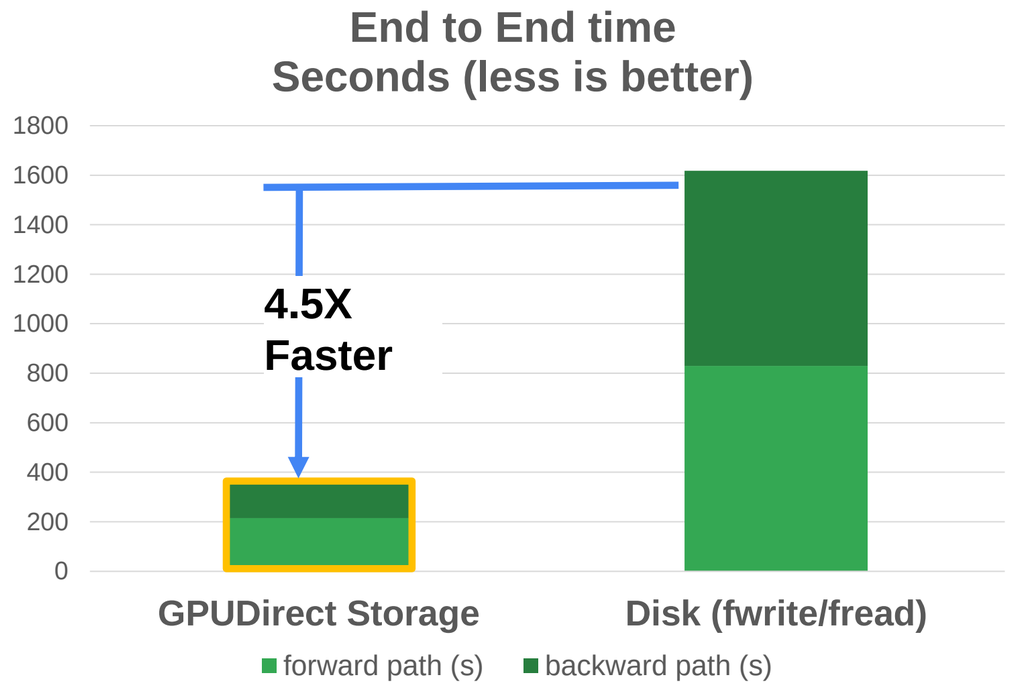


Takeaways

- GPUDirect Storage is better than System Memory
- GPUDirect Storage is cheaper than System Memory

Large Problem Size

imageGrid.size.y=960 on 8 GPUs + 8 NVMe
Total snapshot data size = 5.9TB



Takeaways

- System Memory (>2TB) not an option for larger problems
- Data compression will improve performance

NVIDIA Magnum IO GPUDirect Storage recent evaluation

PSTM benchmark at KAUST

MISSION

KAUST Supercomputing Lab (KSL)'s mission is to inspire and enable scientific, economic and social advances through the development and application of HPC solutions, through collaboration with KAUST researchers and partners, and through the provision of world-class computational systems and services.

Learn more about KSL - https://www.hpc.kaust.edu.sa/

KAUST Supercomputers

- CPU-based Shaheen II supercomputer
- IBEX cluster with GPUs and Weka IO storage for scientific computing and AI applications.
- Shaheen III planned in end of 2023 with 100 PFlops/s based on NVIDIA Grace Hopper

Achievements

KAUST in collaboration with oil and gas industry broke 3 World Records in seismic imaging, reservoir simulation, and engineering

NVIDIA Magnum IO GPUDirect Storage for Pre-Stack Time Migration

Use case highlights – PSTM benchmark at KAUST

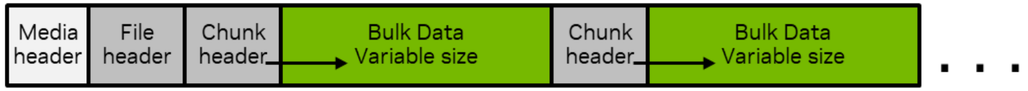
Pre-Stack Time Migration (PSTM)

Seismic migration is a set of techniques for transforming recorded seismic reflection data into an image of reflecting boundaries in the earth's interior.

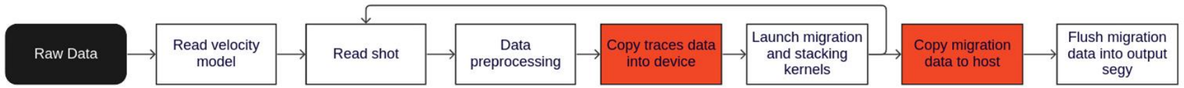
PSTM is based on the computation of the travel-time surface over which energy from a subsurface point is scattered.

PSTM workflow characteristics

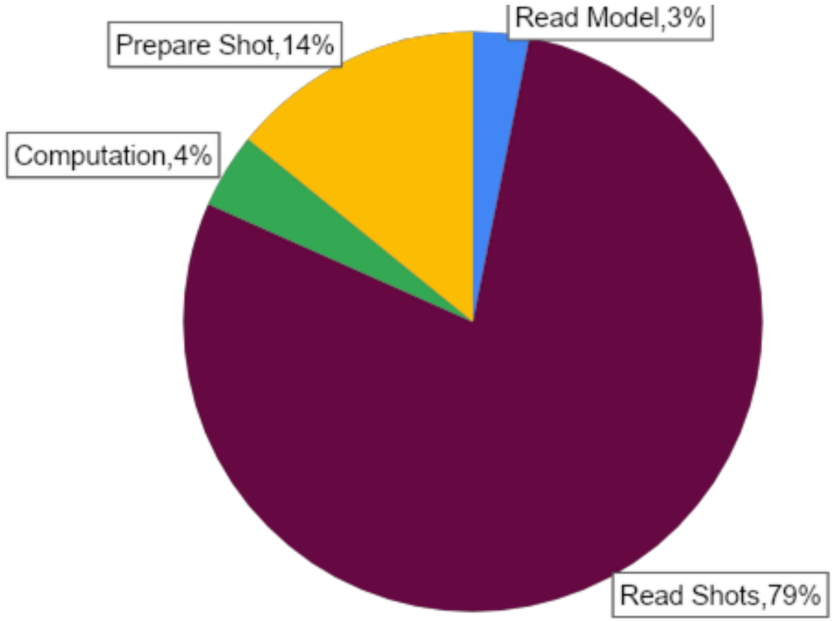
- Simple yet parallel compute on GPUs
- Large Dataset size → IO bound
- Industry Standard data file format → SegY



PSTM workflow analysis



IO Bottlenecks

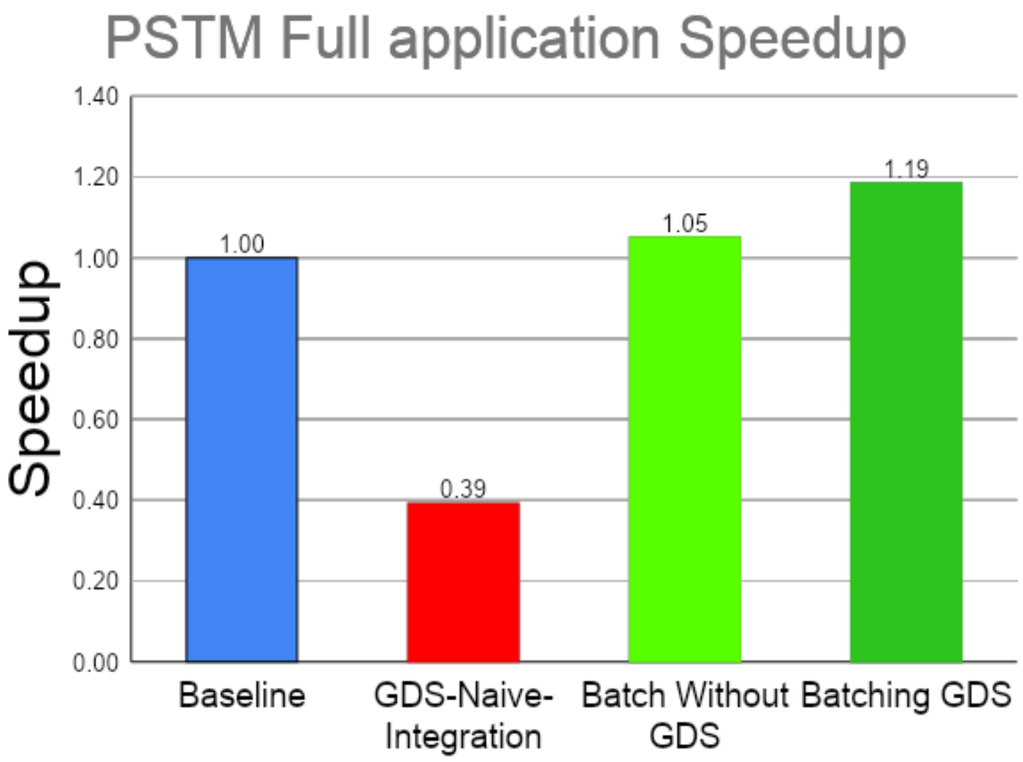


Reading shot data takes 79% of the end-to-end time

NVIDIA Magnum IO GPUDirect Storage for Pre-Stack Time Migration

Results and optimization opportunities

GPUDirect Storage acceleration



1.2X gain by using GPUDirect Storage Batch APIs

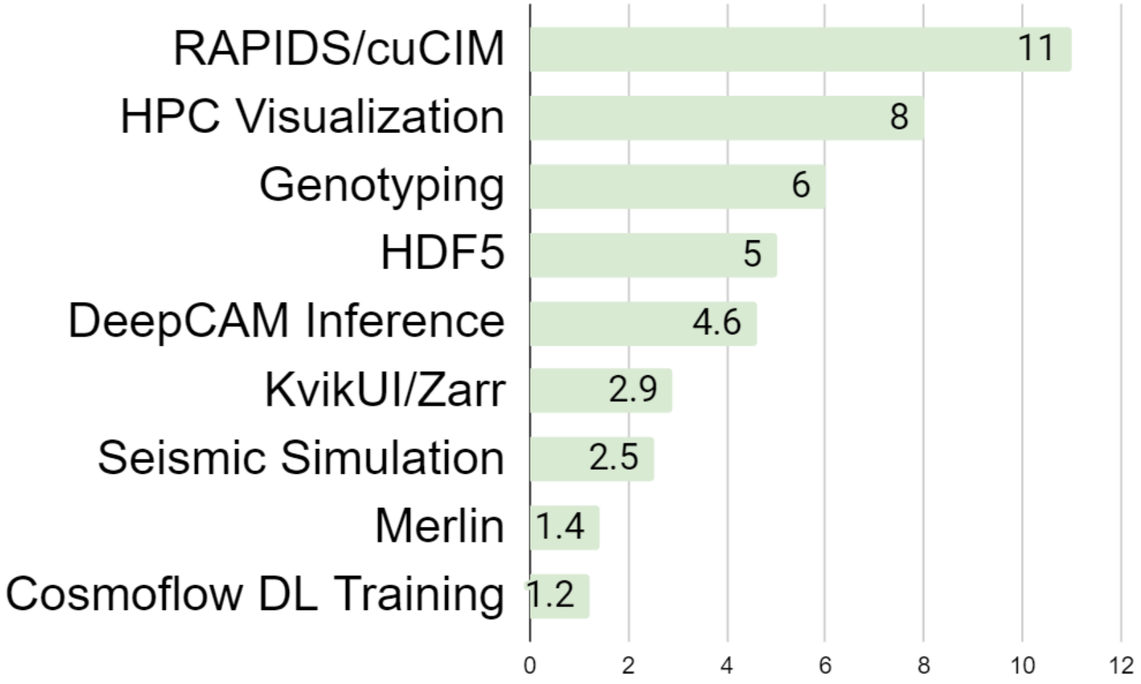
Further Optimization Opportunities

- Large IO requests
- 4KB Aligned IOs
- Use GDS with WekaFS
- Optimize Write path
- Evaluate New GPUDirect Storage Features
  - SysMem Support
  - Async APIs
  - Batch APIs

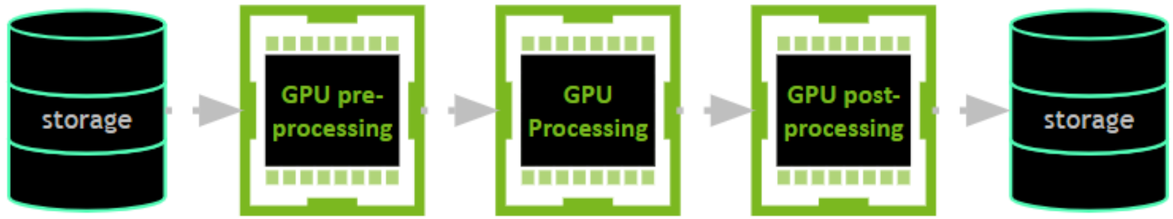
NVIDIA Magnum IO GPUDirect Storage – other use cases

Proven success for many use cases and application profile

IO performance boost



Keep workflow & data on GPU



BENEFITS	REQUIREMENTS
File IO in CUDA. No special Hardware	Use CTK.
2x-8x Bandwidth improvement	System Topology (PCIe Switch)
Faster IO to GPU memory	GPUDirect / Vidmem Only
Up to 3x lower CPU utilization	O_DIRECT only

Many GPU applications are IO sensitive and process large amounts of data.

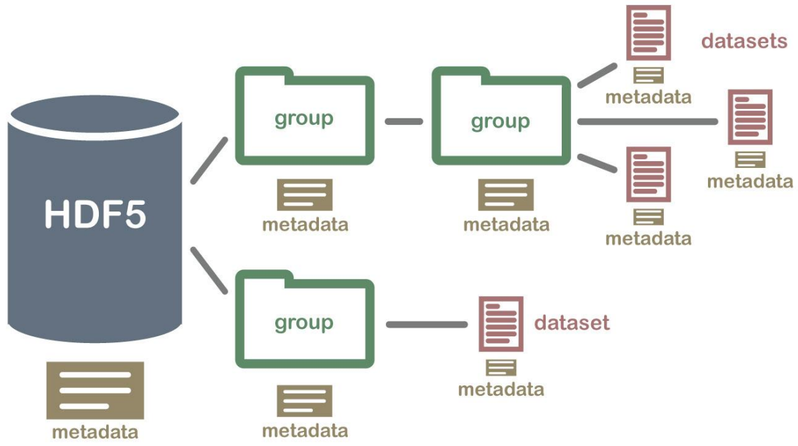
NVIDIA Magnum IO GPUDirect Storage lessons from users

Application developers like GPUDirect Storage performance but have additional requirements

- Hybrid data processing
  - Metadata in CPU, bulk data processing in GPU
  - Asynchronous IO to overlap with compute on GPUs
- Unified APIs are easier to adopt/use
  - POSIX for metadata, GDS API for bulk data
- CSP environment
  - Not ideal system topology, no RDMA for storage
- Industry standard file formats & readers
  - SegY, HDF5, Zarr, Tiff, others

SegY

HDF5



NVIDIA Magnum IO GPUDirect Storage new features

Coming soon

GPUDirect Storage

GPUDirect (GPU Memory)
- GPUDirect mode
- Compat mode

Available Now

GPUDirect Storage

GPUDirect (GPU Memory)
- GPUDirect mode
- Compat mode

+ System Memory
  - Page cache support
- Async API support

Enables
- Hybrid data processing
- Single IO API
- CSP environment
- Standard file formats
- Reader libraries

June 2023

New workloads

MPI-IO, PHDF5, PnetCDF

Many others…

2H 2023

Distribution

NVIDIA Virtual Machine Image (VMI)

NVIDIA DGX Pod

NVIDIA DGX Cloud

NVIDIA AI Enterprise

NVIDIA DGX Solutions

NVIDIA CUDA Toolkit

2H 2023

Please reach out - GPUDirectStorageExt@nvidia.com

Inference and training - DeepCAM and Cosmoflow

Inference over lunch vs. all day; significant impact on training for simpler models

	DeepCAM inference				DeepCAM training				CosmoFlow training			
IO pattern (S=storage IO, N=NCCL, C=compute)	CCCCC SSSS				CCCCCCCCCCCCCNNNN SSSSSS				CCCCNNN SSSSSS			
DNN Layers	O(100)				O(100)				O(10)			
GDS gain (speedup)	global batch size, 1 DGX A100 node				Number of DGX A100 nodes				Number of DGX A100 nodes			
	8	**16**	**32**	**64**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
	2.4x	3.3x	4.3x	4.6x	1x	1x	1x	1x	1.11x	1.16x	1.28x	1.28x

- GDS only helps when storage IO is on the critical path
  - Inference or few-layer training
  - Compute per sample decreases with larger batch sizes
  - Total data increases linearly with batch size
- Larger batch sizes get bandwidth bound (~50 GB/s) for inference

Training: DGX A100 80GB, GDS 1.5.0, NCCL 2.16.2, PyTorch 1.14.0, DDN AI400x. Page cache size reduced to 100 GB to avoid caching, to simulate lack of cacheability of really large data sets.
Inference; DGX A100 80GB, GDS 1.2.1, cuDNN 8.3.0, DALI 1.12.0, PyTorch 1.11.0, Torch-TensorRT: 1.1.0DDN AI400x. Since inference streams, caching doesn't matter.

Storage space innovation at NVIDIA

Solid foundation today for an exciting future

- Compatibility: driver presence, platform variations [CUDA 12.2+]
- Sets: instance → batch [CUDA 12.0.1]
- Synchrony: sync → async in stream/graph [CUDA 12.2, experimental]
- Control source: from CPU → from GPU, DPU [CUDA POC]

Compatibility

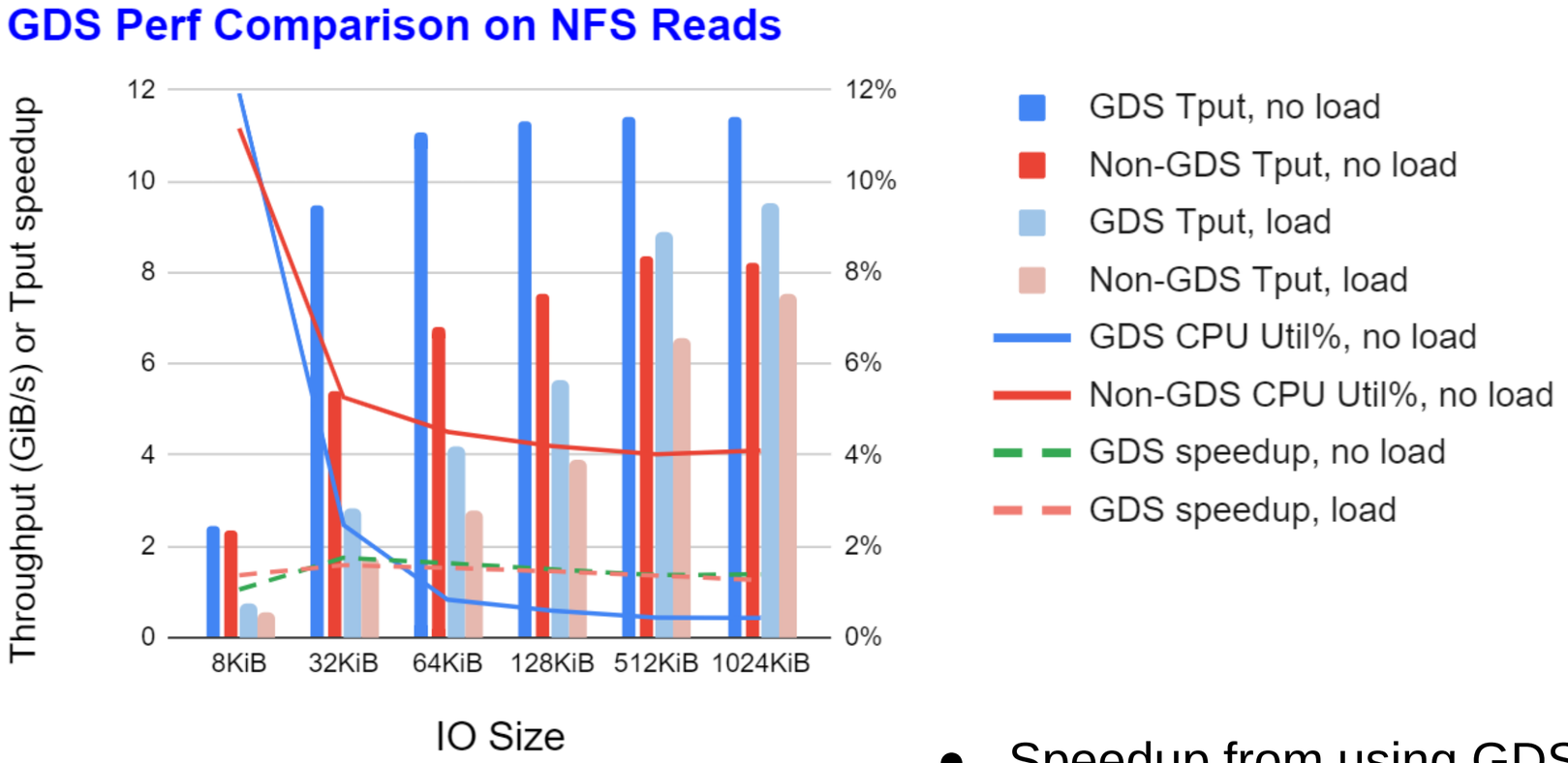
Synchronous submission, asynchronous completion; amortize user-kernel overhead

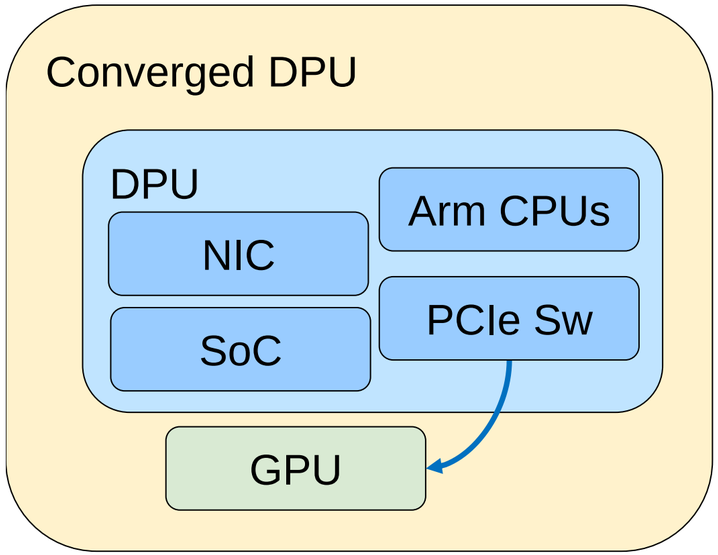
- Single and simple file IO API for any type of Memory
  - Similar to POSIX
  - No need for separate code paths for IO to different types of memory
    - GPU memory (e.g. cudaMalloc, cuMemMap) ✓
    - CPU memory (e.g. malloc, cudaHostMalloc) ✓
      - O_DIRECT or Page Cache ✓
    - Migratable memory
      - e.g. cudaMallocManaged/staged ✓
      - malloc + HMM, malloc on Grace-Hopper
- Implementation generality
  - Unaligned buffers and those too big to fit through the GPU BAR1 aperture ✓
  - Work in CSP instances (ACS , IOMMU)
  - Converged GPU-DPU cards ✓
  - Grace-Hopper unified memory

New in CUDA 12.2
Soon

Converged DPU-GPU

Remote storage to NIC to GPU never leaves the card





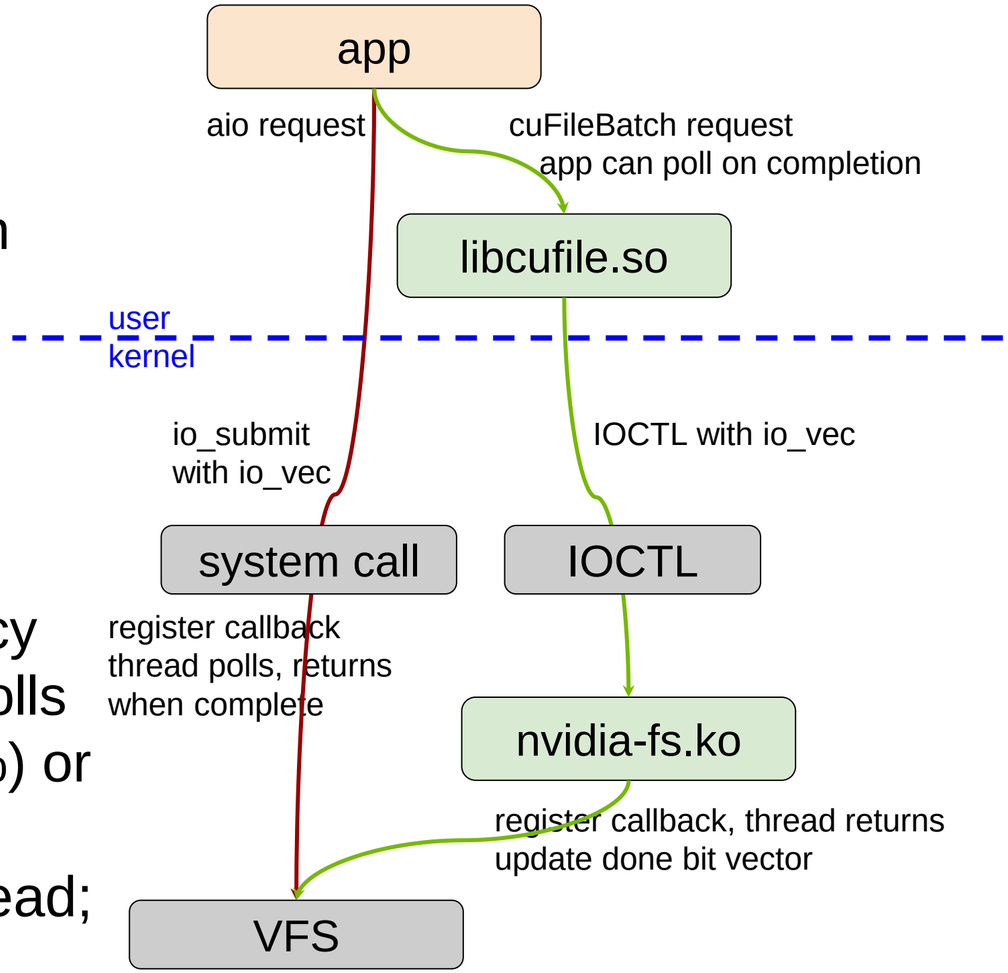
- Setup
  - A100T = BF2@200 Gbps+A100 GPU
  - Remote storage
- With and without load on Arm cores
  - Optional load on DPU = Stream

- Speedup from using GDS
  - 1.1x-1.7x without load
  - 1.3x-1.6x with load
- CPU utilization reduction with GDS
  - 1-8x reduction without load
  - Fully loaded with Stream

Batch - new feature in CUDA 12.0.1

Synchronous submission, asynchronous completion; amortize user-kernel overhead

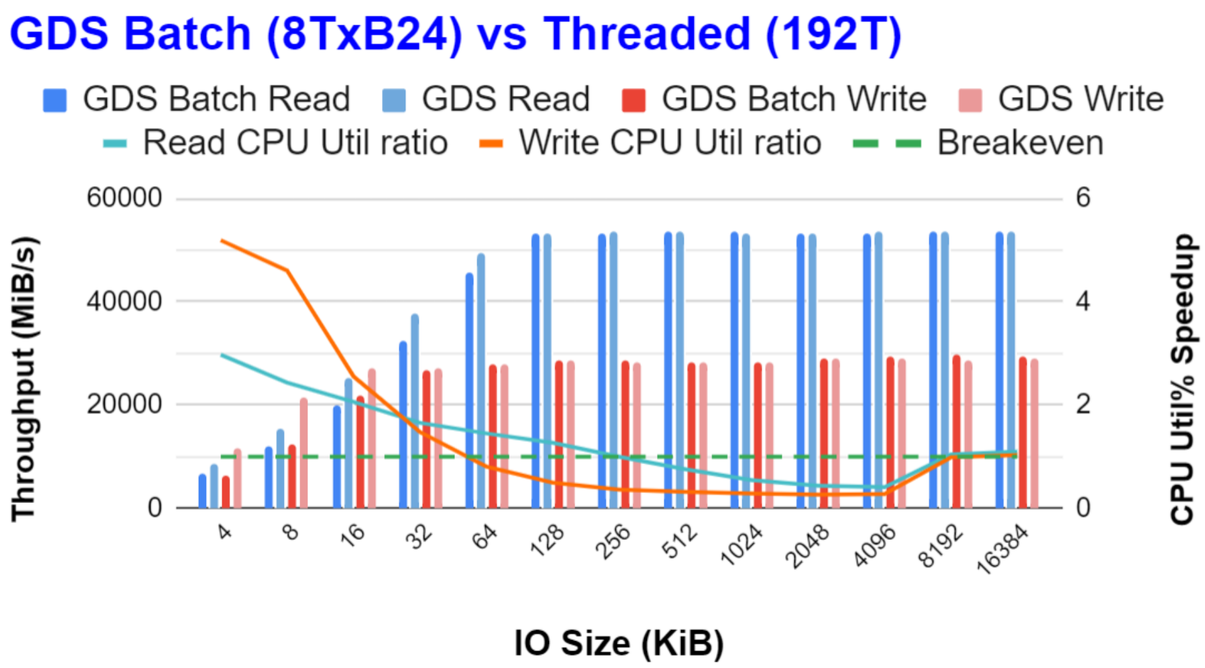
- Submit a read/write mix to 1+ open files at a time
- Async completion
  - app thread submits, returns, polls
  - io_vec of requests serially submitted from within kernel, but they complete OOO
- Performance advantages of batch
  - vs. many threads: lower perf overhead
    - 1 IOCTL per batch of requests
    - IOPS for batch of small IOs > many threads
  - vs. Linux aio: lower CPU utilization, lower latency
    - batch: callback updates done vector; app polls
    - aio: either kernel polls continuously (CPU %) or takes an interrupt (longer latency)
- Enables as-if threaded execution with a single thread; multiple threads can further boost throughput
- Call to action: try the new production APIs



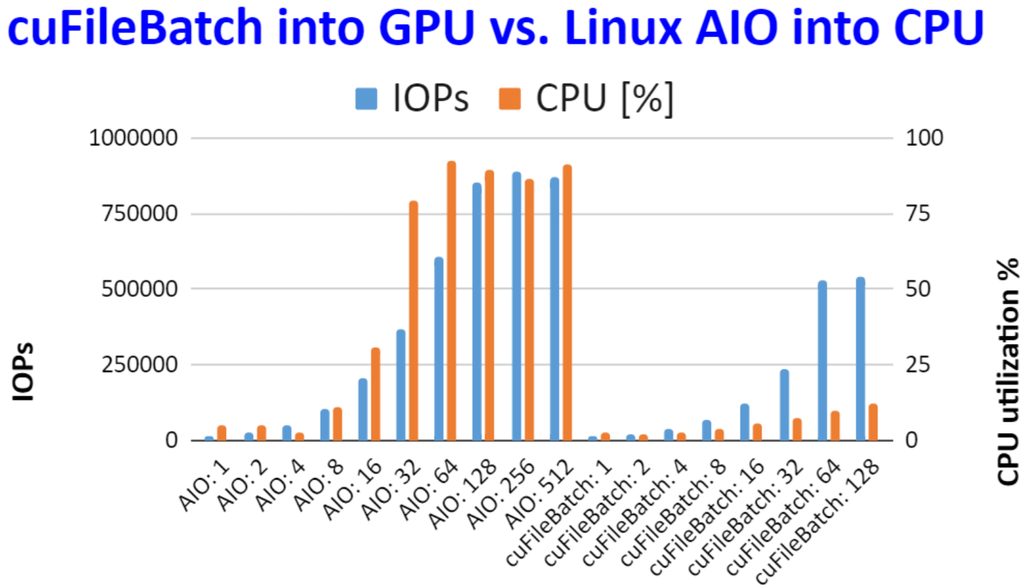
Batch results

Trading off a bit of throughput for drastically lower CPU utilization

- Batch is 75-90% of non-batch performance with 1 host thread
- But its CPU utilization lower, especially under 32KB IO size
  - Up to 5x lower on an enterprise-grade system, especially for small IO sizes
  - 9x lower on a consumer system



Zhen Zeng, NVIDIA, DGX A100, 8x Samsung P1733 3.5TB
Batch size of 24/thread, 8 host threads vs. 24*8 host threads for 8 GPUs

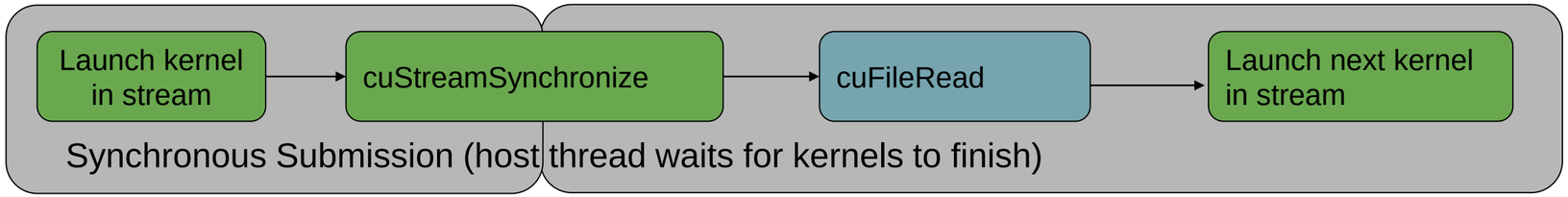


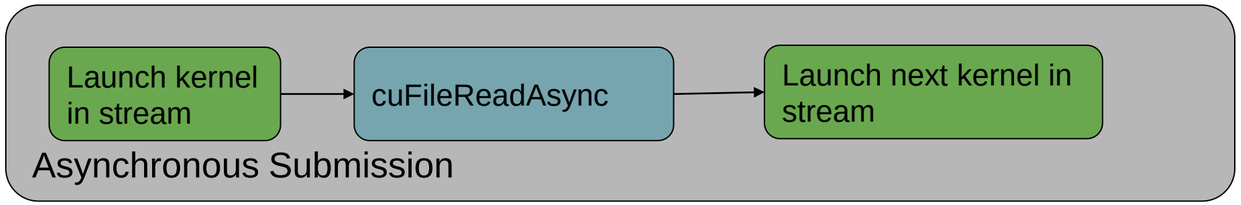
Kazuo Goda, U Tokyo, RTX A4000/16 GB, Intel Xeon W-2245/8C/3.90 GHz/64GB,
WD_BLACK AN1500 NVMe SSD/1.8TB

Async - future feature

Asynchronous submission in a stream, async completion

- Read/write/register/unregister APIs gain a stream parameters
- Enables stream-ordered submission for deferred execution
  - Can rapidly submit many IOs without waiting for completion with a blocking API
  - Synchronization in CPU-GPU handoff HW enables higher IOPs than SW on CPU
- Opens the door for codes that use CUDA Graphs and CUDA Streams

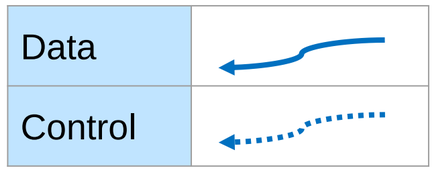




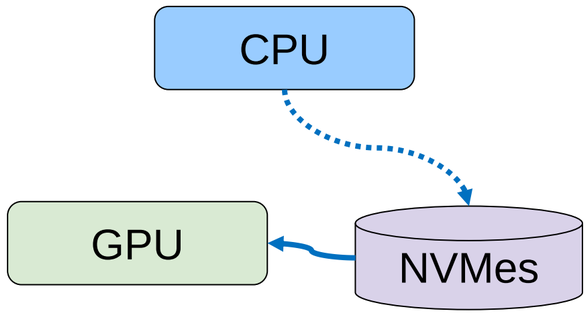
- Call to action
  - Offer feedback on the (public) experimental APIs
  - Try them out in a forthcoming release

Control sources

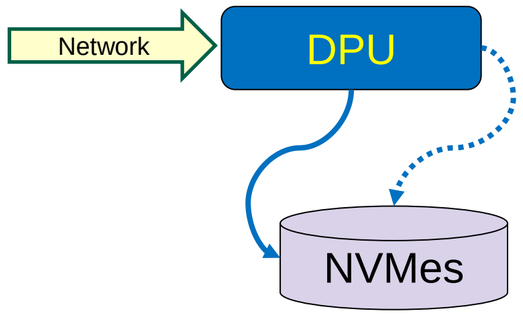
Each of CPU, DPU, GPU can control NVMe storage



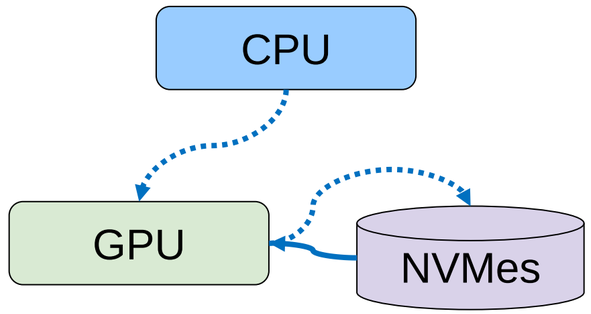
CPU



DPU



GPU



Usage	Read/compute/write	JBOF	RecSys, GML, GNN
Interface	cuFile	NVMf	GDA kernel-initiated
Coord	orchestrate DMAs	Stage data to/from media	Spcl staging
Data	bypass CPU	System cache or CMB	NVMe
Gain	↑bw, ↓CPU utilization	Max bandwidth, ↓cost	Concurrency

Problem: Large volume of random fine-grained accesses

GPU concurrency is key to throughput

- Large volume of random fine-grained accesses → large concurrency to maximize tput
  - GPU >>$_{concurrency}$ CPU; control and data path would be bottlenecked on CPU
- Data consumed on GPU; requests may also be generated there
  - Feeding data through CPU becomes a bottleneck
- Criticality assumptions about rates
  - tput = min(GPU request generation, $ bw, NVMe access for misses, data consumption on GPU)
- GPU advantages over CPU
  - Generating requests — more threads generating requests
  - Requests to local cache — more threads accessing in parallel, tolerant of latency
  - Making requests to NVMes — more threads generating NVMe requests
  - Consuming data — more threads and other acceleration features like tensor cores

GPU-initiated storage usage models

Large batches of small IOs to GPU memory requiring efficient KeyValue APIs

- Graph neural network
  - Widely used in fraud detection, fake reviews, tracking bot assaults, recommendation systems
  - Nodes (millions to billions) and edges (billions to trillions) graphs
  - Each node and edge has embeddings of size upto 4KB (>10TB)
- Recommender systems
  - Training pipeline - 10-100 TB embedding models requiring fine-grain access
  - Data ingestion pipeline - requiring efficient preprocessing such as filtering and reconstruction
- Data analytics (cuDF, Spark from RAPIDs)
  - spill management, shuffle management on billion rows
  - Cost reduction
- Omniverse
  - Low-latency persistent texture objects with multiple simultaneous clients
- Vector Search
  - Billions of documents represented as vectors (~20PB)
- Graph Analytics in ML - current cuGraph only supports if graph is in memory
  - Nodes (millions to billions) and edges (billions to trillions)
  - Require the graph in memory address space

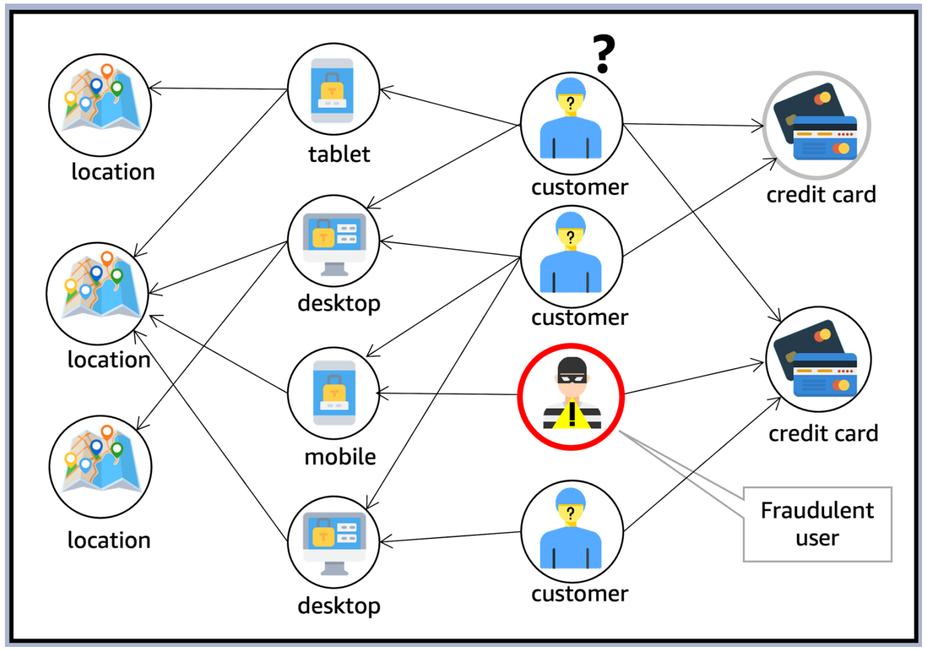
GPU-initiated storage usage models

Large batches of small IOs to GPU memory requiring efficient KeyValue APIs

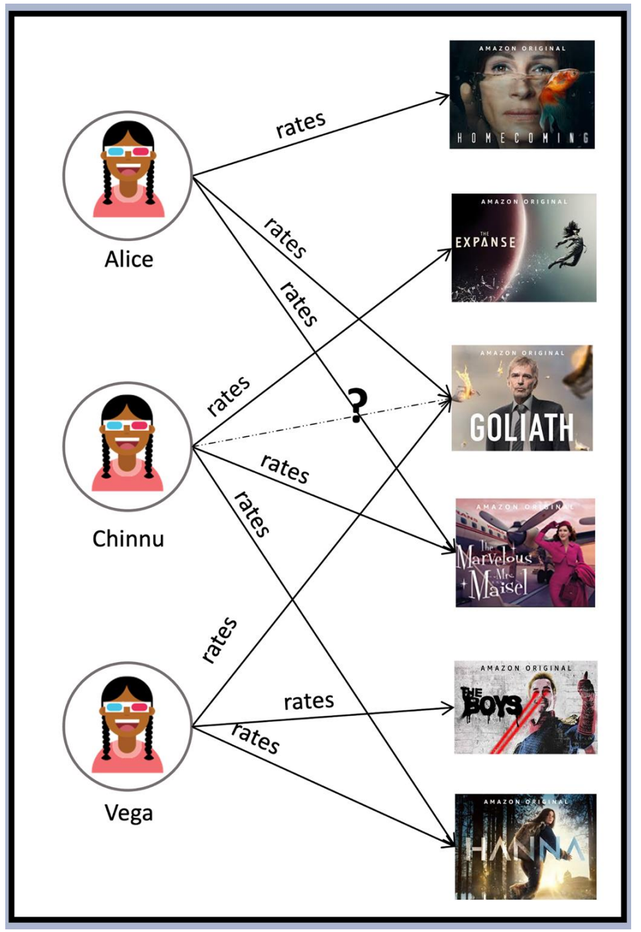
- Graph neural network
  - Widely used in fraud detection, fake reviews, tracking bot assaults, recommendation systems
  - Nodes (millions to billions) and edges (billions to trillions) graphs
  - Each node and edge has embeddings of size upto 4KB (>10TB)
- Recommender systems
  - Training pipeline - 10-100 TB embedding models requiring fine-grain access
  - Data ingestion pipeline - requiring efficient preprocessing such as filtering and reconstruction
- Data analytics (cuDF, Spark from RAPIDs)
  - spill management, shuffle management on billion rows
  - Cost reduction
- Omniverse
  - Low-latency persistent texture objects with multiple simultaneous clients
- Vector Search
  - Billions of documents represented as vectors (~20PB)
- Graph Analytics in ML - current cuGraph only supports if graph is in memory
  - Nodes (millions to billions) and edges (billions to trillions)
  - Require the graph in memory address space

Graph machine learning in industry

Node-level prediction

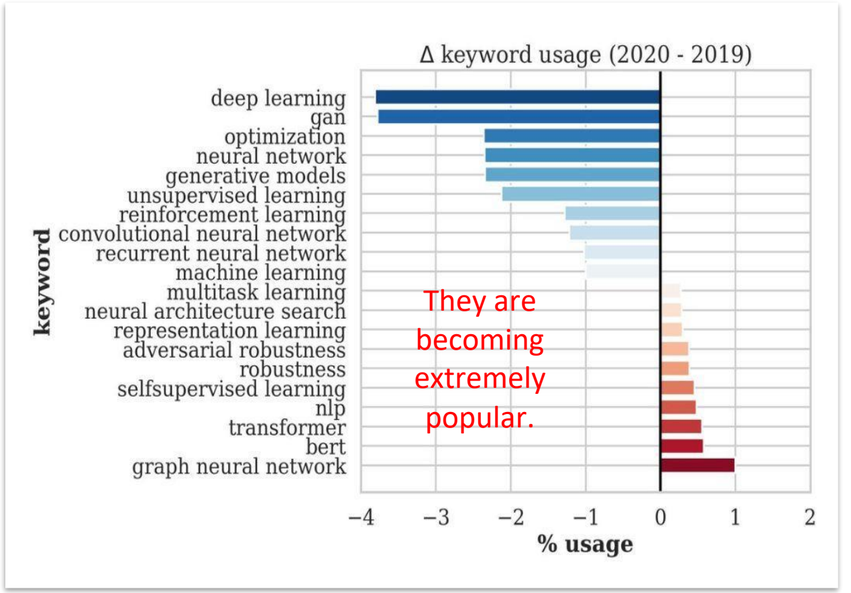


Edge-level prediction

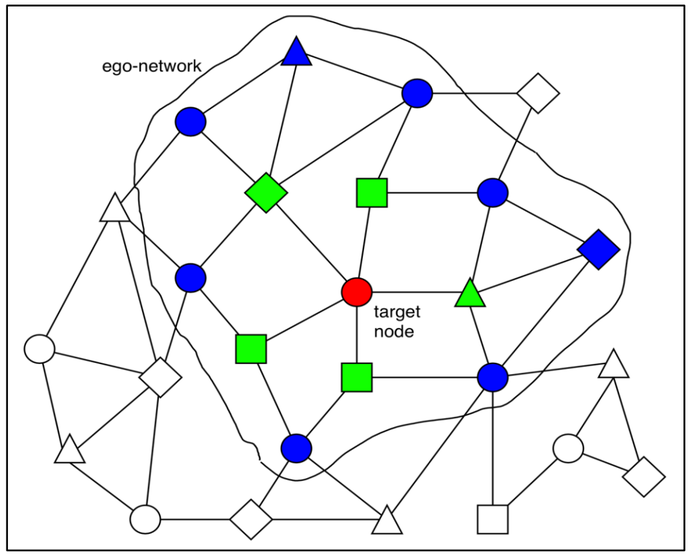


Graph neural network (GNN)

A family of (deep) neural networks that learn node, edge, and graph embeddings



How do GNNs work?



An ego-network around each node is used to learn an embedding that captures task-specific information.

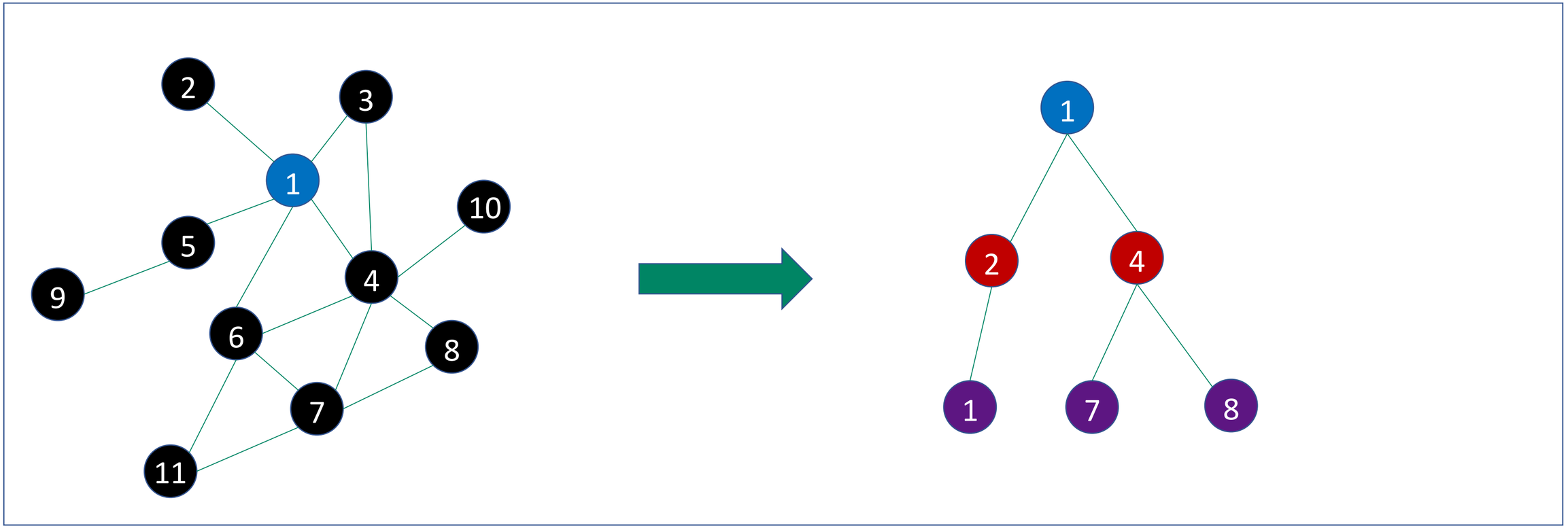
The embeddings use both the structure of the graph and the features of the nodes and edges.

The embeddings are learned in an end-to-end fashion; thus, the predictions are a function of the target node's ego-network.

Mini-batch training on graphs

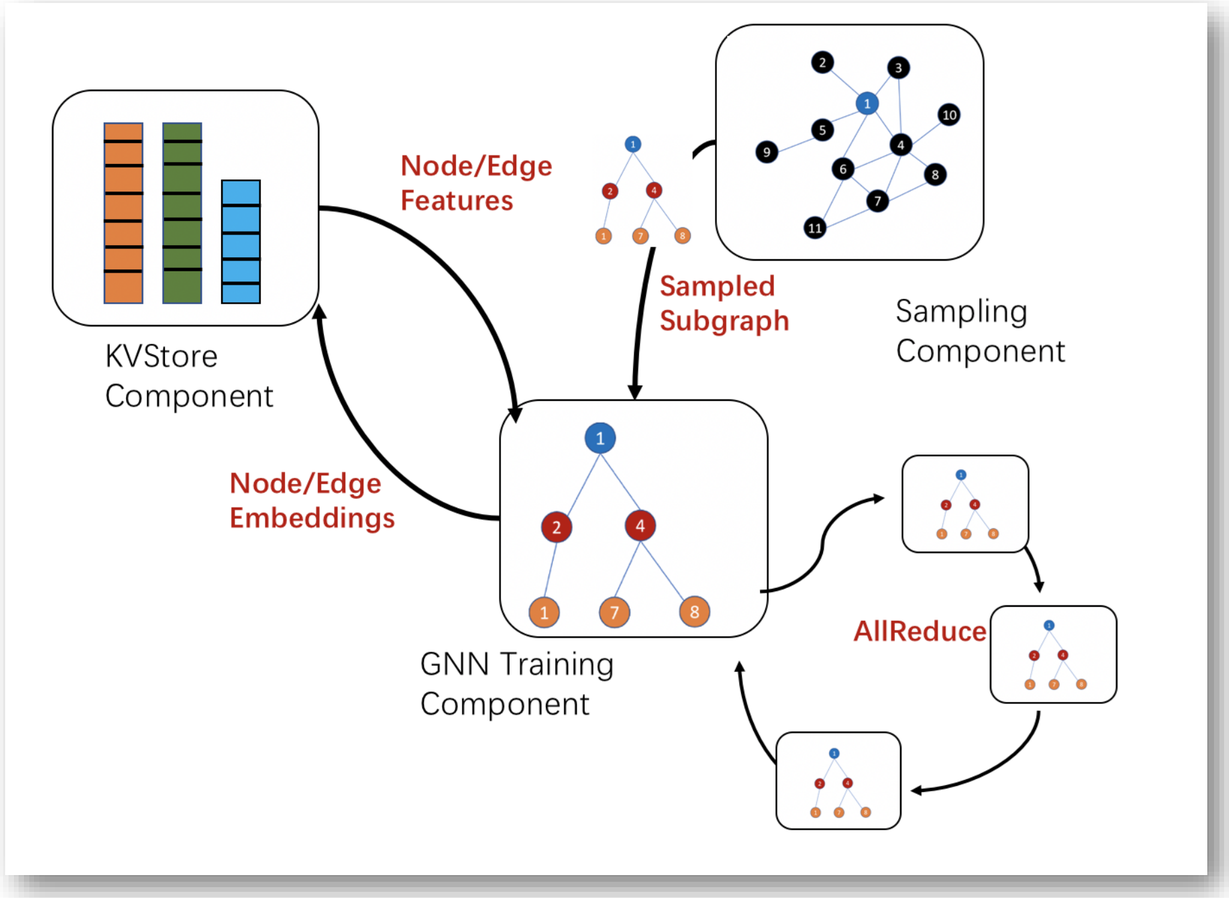
A mini batch represents the computation graph for some target nodes.

- Sample target nodes for prediction
- Sample neighbors of the target nodes

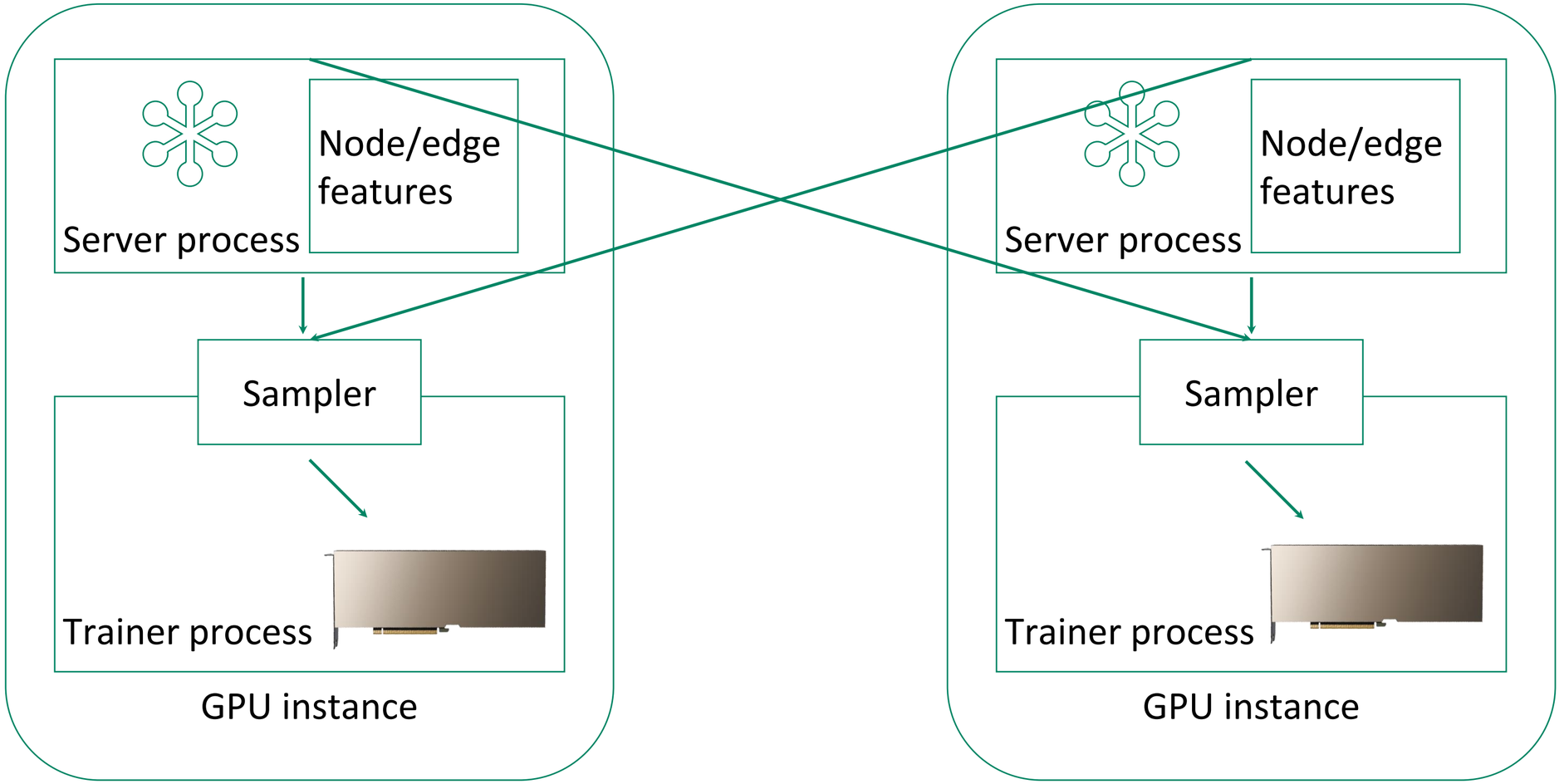


Multi-machine distributed training with DistDGL

- Design
  - Partitions the graph across the machines.
  - Data parallel mini-batch training.
  - Synchronous GNN model parameter updates.
  - Asynchronous sparse embedding updates.

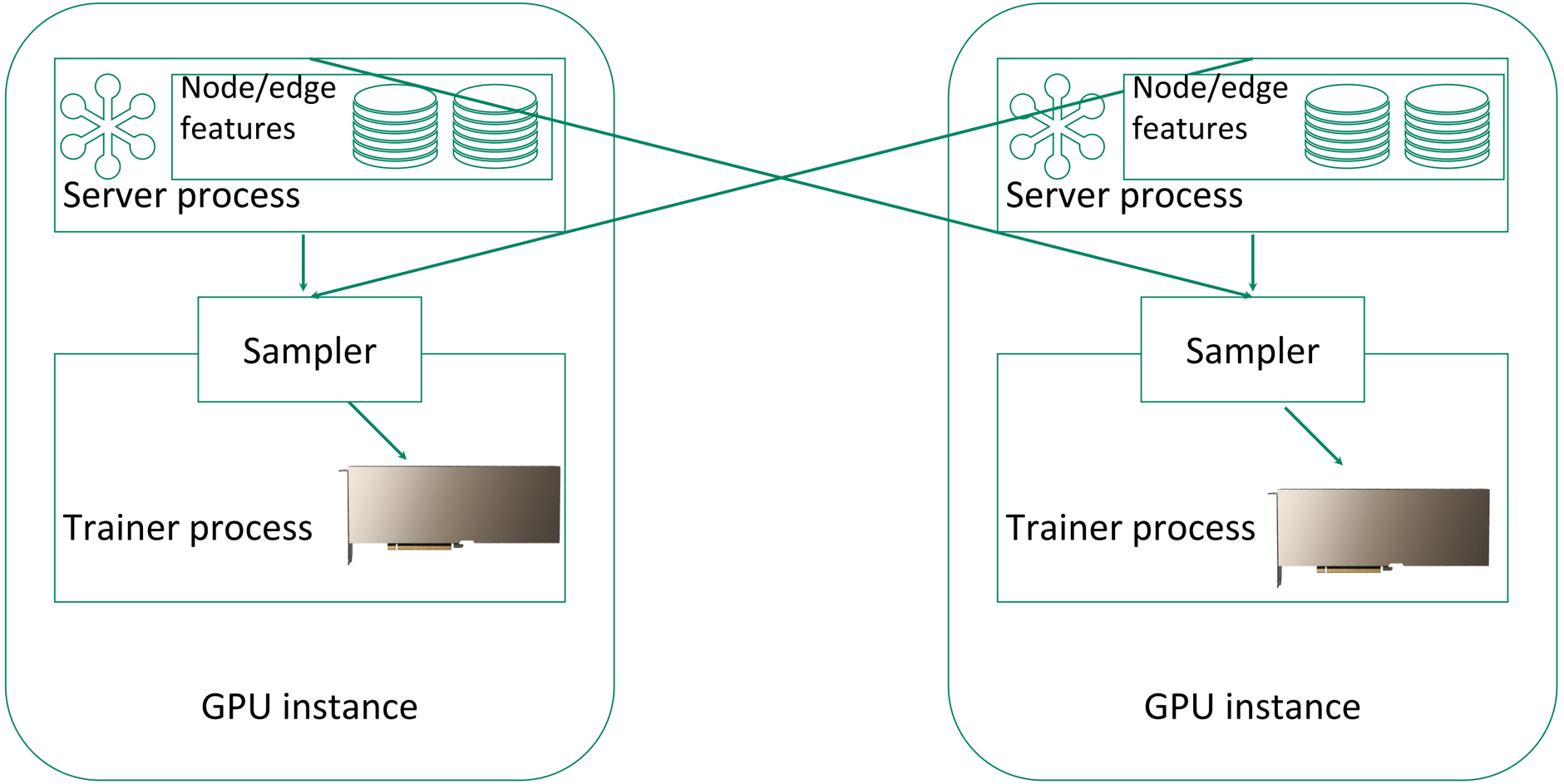


CPU-based distributed training



Co-locates data with computations to reduce network communication.

NVMe-based distributed training



Scale GNN training and memory requirements

	Graph Size	Total memory size	Node feature size	Reduce CPU memory consumption with NVMe by (%)
OGBN-papers100M	#100M nodes #1.6B edges 128 node features	100 GB	52 GB	52%
MAG-LSC	#240M nodes #3.4B edges 768 node features	224 GB	174 GB	78%
Future target	O(10-100B) nodes O(100B-1T) edges	100 TB to >1 PB	100 TB to >1 PB > feature richness	improves with > nodes, compression

- Storing node/edge features on NVMe can significantly reduce CPU memory consumption.
- NVMes are much cheaper than CPU and GPU memory and have much larger capacity.
- If NVMe performance can keep up with GPU training speed, this is much more cost effective to scale GNN training.

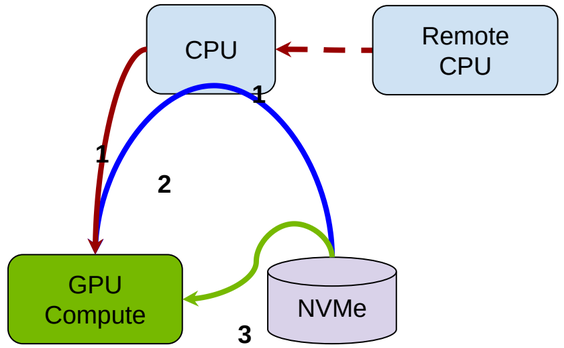
Distributed GNN training pipeline

Mini-batch training steps:

- Sample mini-batch
- Copy node/edge features
- Mini-batch computations

The data copy throughput required in each step.

	Mini-batch sampling (CPU)	Node feature copy (1: CPU-CPU-GPU)	Node feature copy (2: NVMe-mmap)	Mini-batch computation (GPU)
OGBN-papers100M	3407 MB/s	1020 MB/s	40.2 MB/s	6813 MB/s
MAG-LSC	4733 MB/s	1241 MB/s	41.2 MB/s	4730 MB/s



AWS system info:
g4dn.metal with T4 GPUs
(2560 CUDA cores @ 585MHz)

- CPU, NVMe can’t keep up, need a better solution
- But fast-enough NVMe reduces memory consumption, enables lower cost

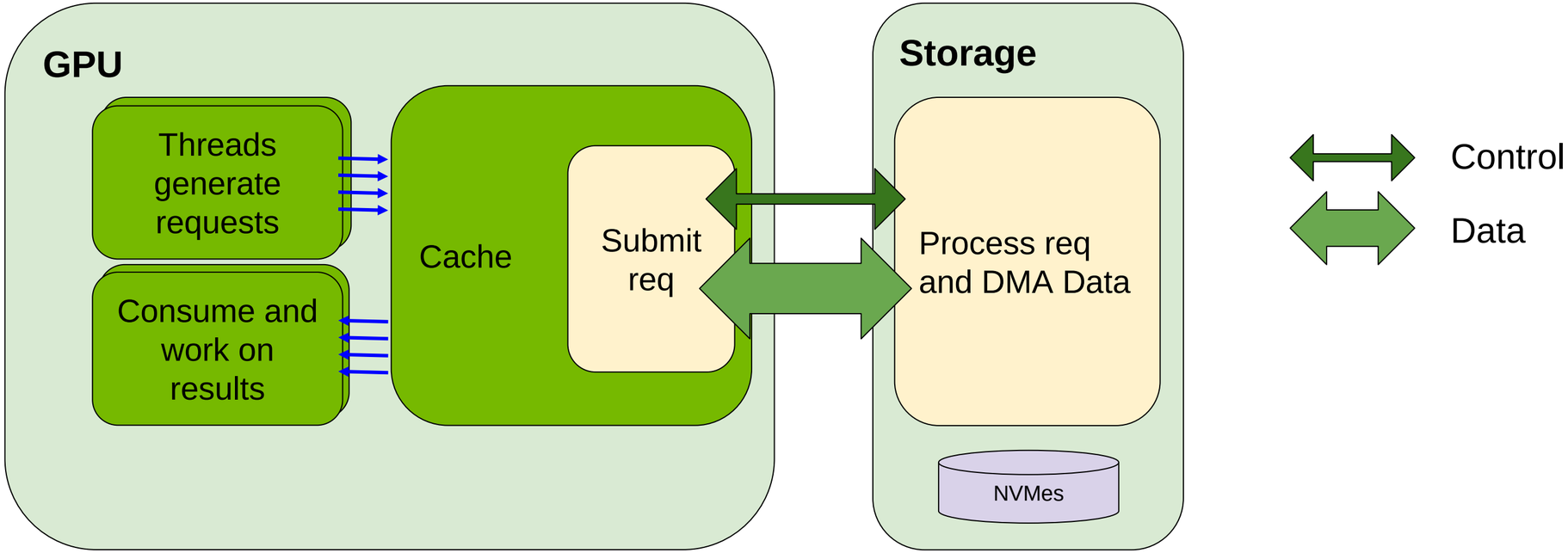
GPU-initiated storage design goals and requirements

POC in research, not a committed product yet

- Design goals
  - New APIs to access storage IO from GPU
  - Maximize throughput for large batches of small data accesses
  - Scale to problem sizes too big to fit into GPU HBM or CPU DDR with cheaper NVMe
  - Relieve NVMe IOPs bottleneck with GPUs vs. CPUs
- Design choices to fulfill requirements
  - In case there's any temporal or spatial locality to the data
    - Bandwidth out of the cache in the GPU >> PCIe bandwidth into the GPU
    - Making the cache line size match the block size enabling aggregation into block accesses
  - Storage capacity provided by NVMes
  - NVMe bandwidth is maximized by issuing concurrent accesses on abundant GPU threads

GPU-initiated storage architecture

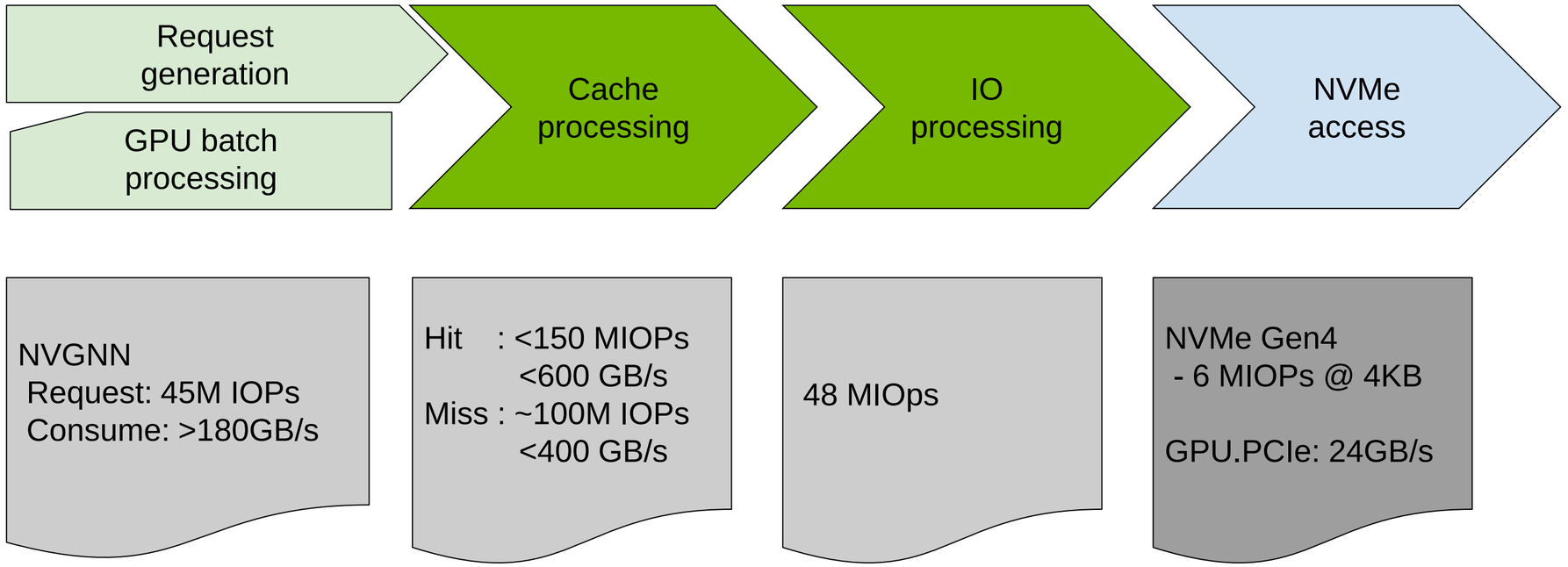
Ultimate removal of the CPU as a bottleneck for storage



- Request, initiation, service, consumption all happen on the GPU
- GDA KI Storage enables storage IO accesses that are both initiated and triggered by GPU
- Features a key pillar of Magnum IO: flexible abstraction

NVIDIA GNN performance results with our POC

Balanced end to end system matches best-available throughput

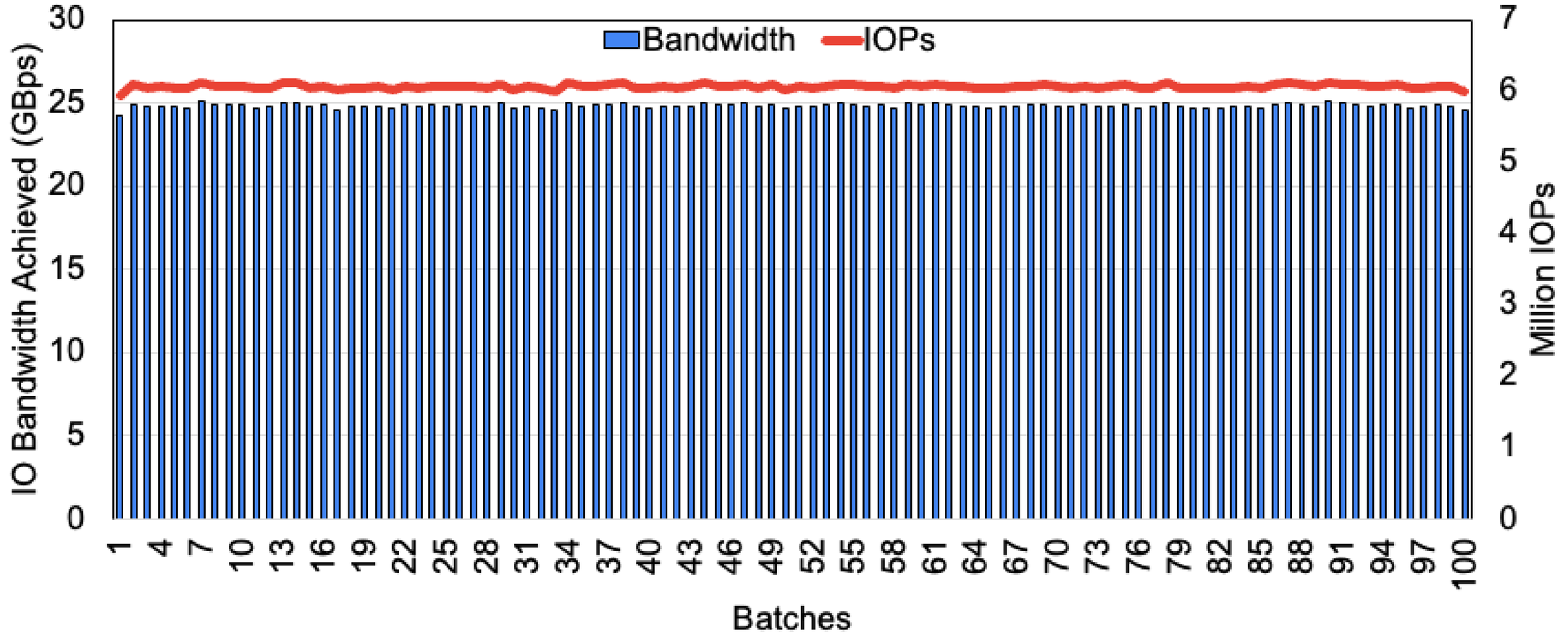


Per GPU tput on A100
6910 CUDA cores @1.41GHz
Transfer size = 4KB

- Data lookup acceleration enables higher throughput by reducing the IO bottleneck to (feature) data
  - transparent data reuse benefit: cache bw (400-600 GB/s) >> PCIe into the GPU (24 GB/s)
  - IO processing (48 MIOPs) keeps up with PCIe-saturating NVMe IOPs rates (6-48 MIOPs)
- GPUs are latency tolerant - HW context switching covers miss latency

GNN Performance

GPU-initiated IO can saturate storage throughput easily



MAG240M Dataset, GraphSAGE Model running NVIDIA A100 GPU with 8 Samsung 17XX SSDs.

*

GPU-initiated storage: Current POC limitations

Effective proof of concept with extensible architecture

- Scale
  - Single node with CPU/GPU/NVMe
- Access APIs
  - Memory array abstraction
- Loading NVMe
  - NVMe is preloaded
  - Requests always hit in NVMe tier
- Not integrated into AWS’s E2E system

*

AWS applicability

	Mini-batch sampling @AWS (CPU)	Node feature copy			Mini-batch computation @AWS (GPU)
		1: CPU-CPU-GPU @AWS	2: NVMe-mmap @AWS	3: GPU-initiated storage @NV subsystem	
OGBN-papers100M	3407MB/s	1020MB/s	40.2MB/s	up to 24 GB/s	6813MB/s
MAG-LSC	4733MB/s	1241MB/s	41.2MB/s	up to 24 GB/s	4730MB/s

- Relevance of prototype
  - Bandwidth significantly exceeds current alternatives of CPU and NVMe via CPU
  - Much lower training cost.
  - Provides massive memory address space and scales with the graph size
  - Simplifies the programming to the storage
  - Minimizes the cost of the graph partitioning
- Priorities among GPU-initiated storage future directions
  - Distributed support.
  - Unified data access API to hide all complexity
  - Prefetching data to overlap computation and communication.

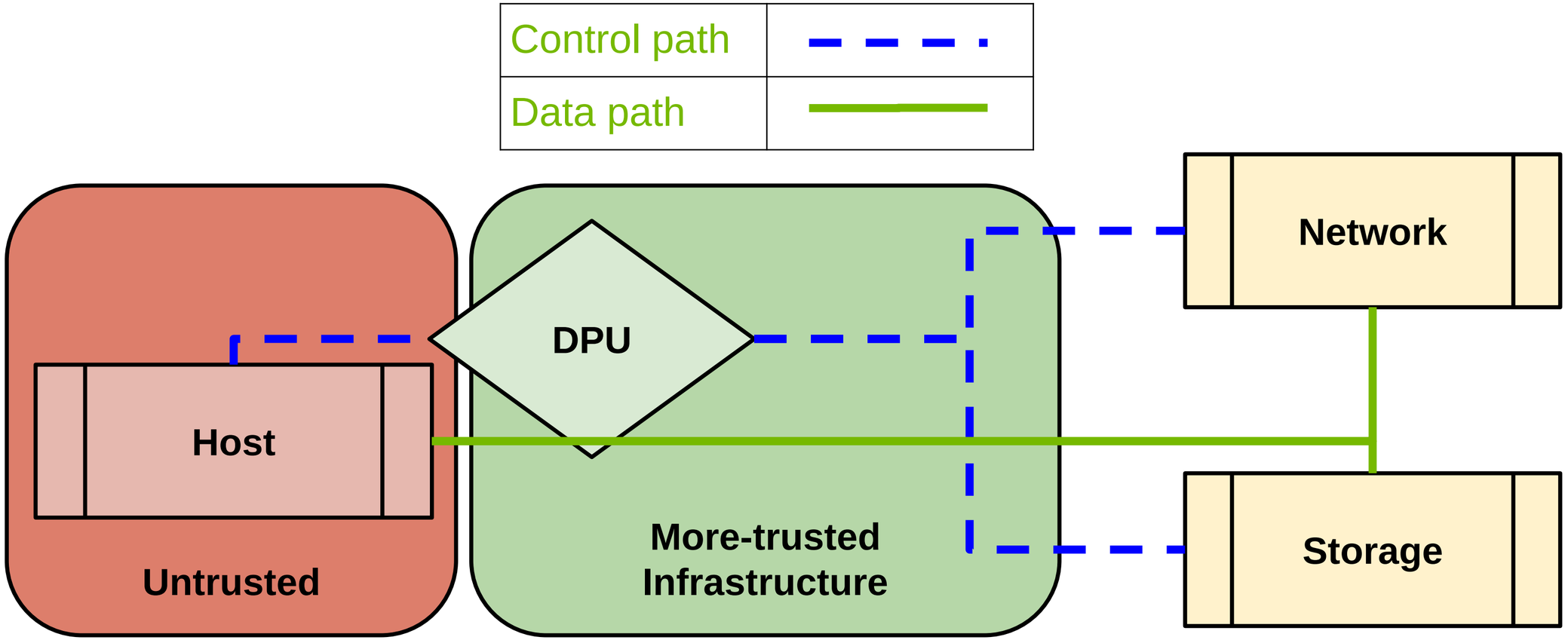
*

Security

- Attacks
- Separation, least privilege
- Trusted intermediaries

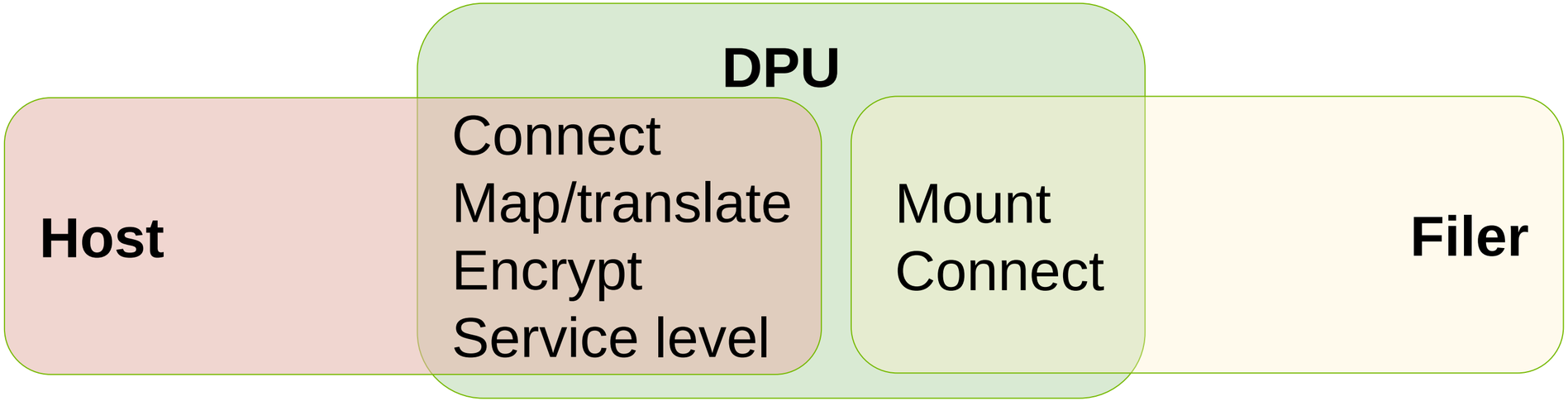
Trust zones

Control path enabled by trusted infrastructure
Uninhibited data path (RDMA)



Trade-offs

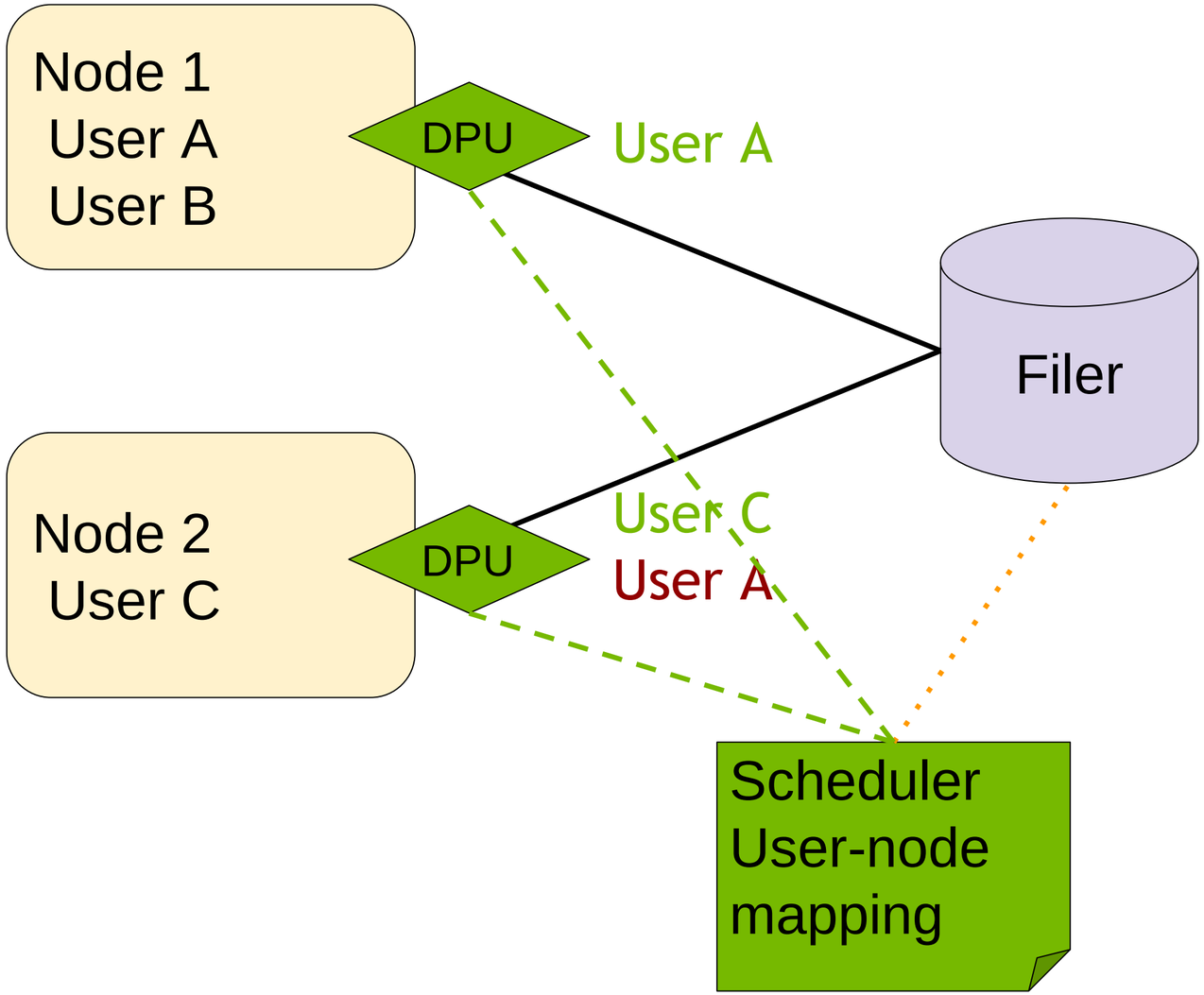
Frame problems, explore technologies as a community



- Choices in where to implement a solution
  - Outsource client work to DPU (left)
  - Proxy actions of the filer at threshold of compute host (right)
- Trade-offs regarding the effort, robustness, generality of the solution

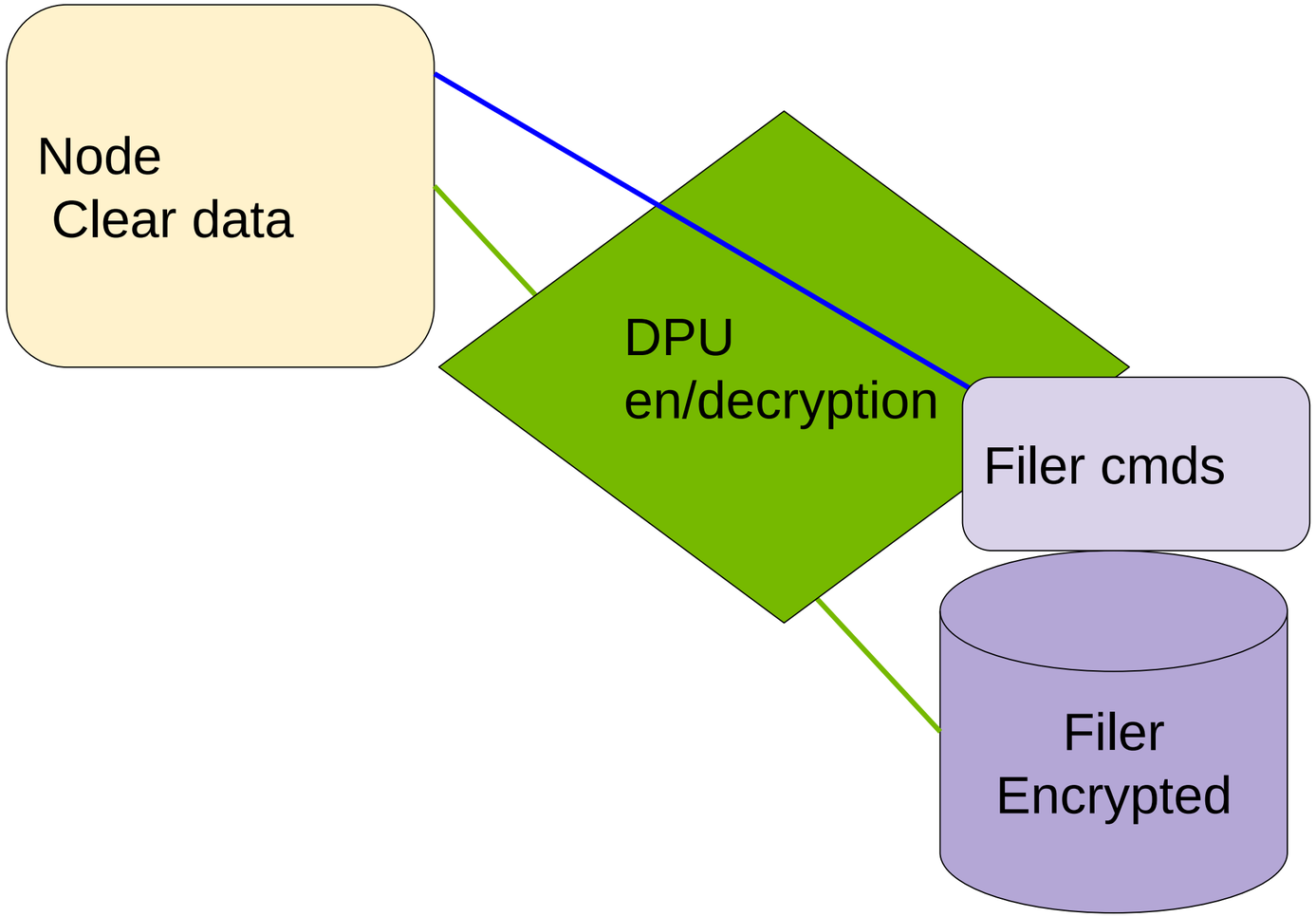
DPU as a gate to authorization

Credentials sent to and used by more-trusted DPU vs. compute node



Data in motion and at rest

Keys distributed to DPU vs. to untrusted compute node



Credits

GPUDirect Storage team

- Kiran Kumar Modukuri, Zhen Zeng, Sourab Gupta, Rebanta Mitra, Prashant Prabhu, Aniket Borkar, Vahid Noormofidi, Sandeep Joshi, Salah Chaou

NVIDIA Research team

- Wen-mei Hwu, Isaac Gelado, Vikram Sharma Mailthody, Zaid Qureshi

NVIDIA GNN team

- Kyle Kranen, Nicolas Castet, Onur Yılmaz, Joe Eaton

NVIDIA GPU Communications team

- GPUDirect technologies: Davide Rosetti, Pak Markthub
- Libraries: Jim Dinan, Sreeram Potluri

UIUC

- Arpandeep Khatua, Jeongmin Park

U Tokyo

- Kazuo Goda, Tsuyoshi Ozawa, Yuya Miura

Call to action

- Try CUDA 12.2 for new features
  - Update: Batch APIs - Better performance and lower CPU utilization. Good for small IOs.
  - New: Versatile cuFile APIs for all file IO needs that can be used in CSP environments
  - New: Async APIs - Aligned with CUDA streams and CUDA graphs, apps can continue working without waiting for IOs to complete.
- HPC reader library developers - Please reach out to enable GPU IO.
- Engage with us to share your usage models and feedback
  - GPU-initiated storage
  - Storage security partner implementations