GPUs as Data Access Engines

Thursday Aug 8, 2024, 8:30-9:35am session

CJ Newburn, Distinguished Engineer, NVIDIA GPU Cloud

Trends

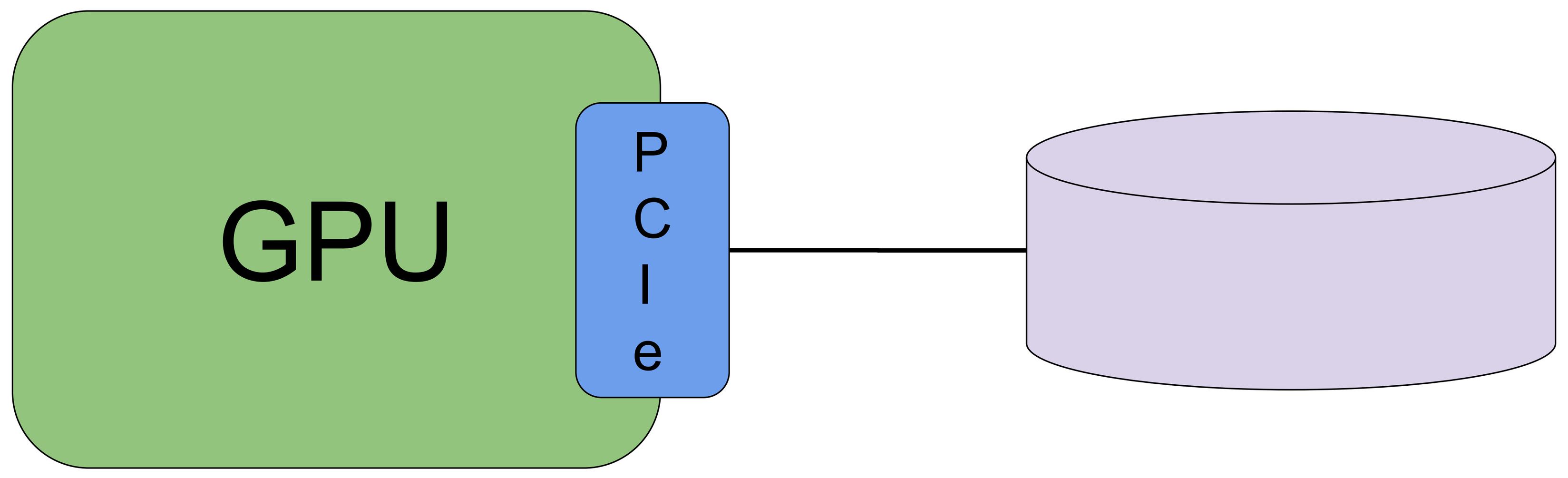
New considerations as we scale up and out

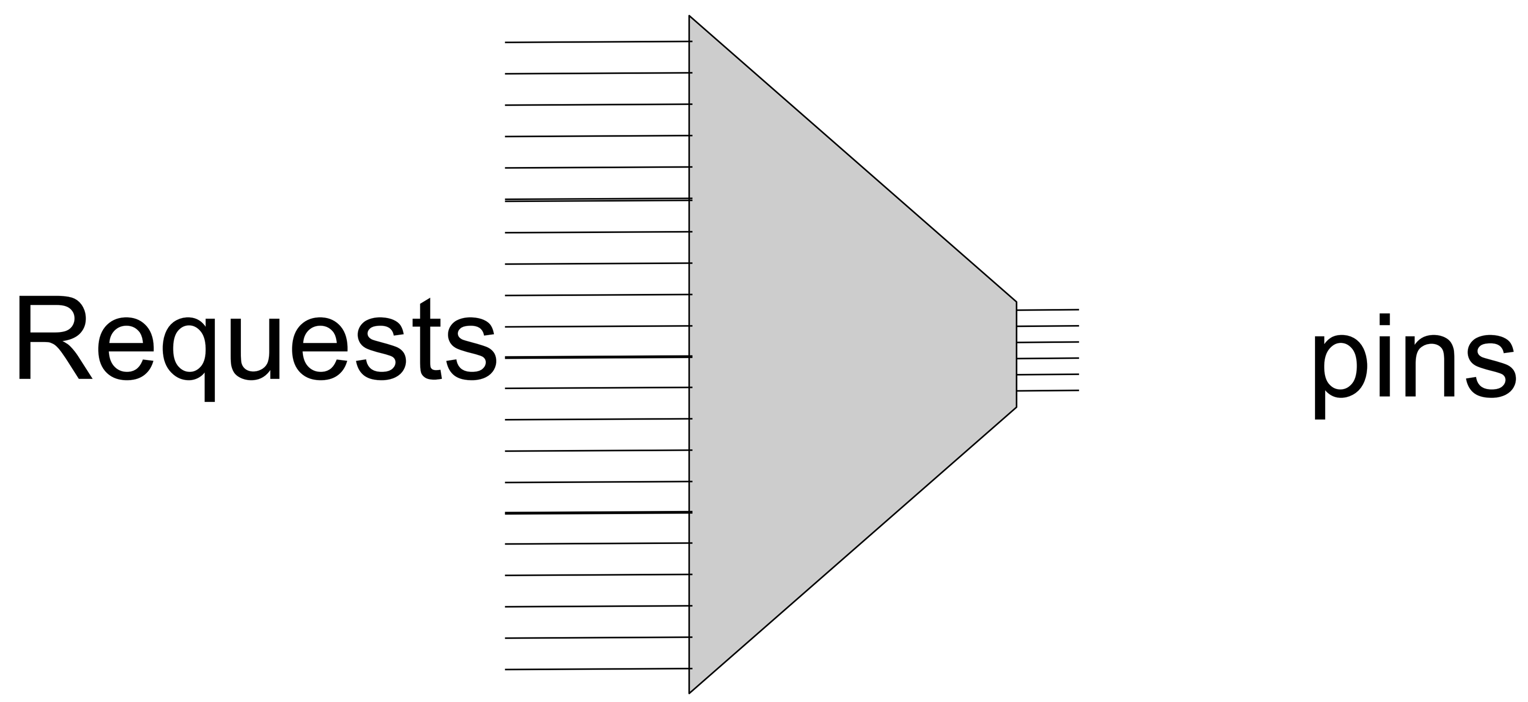
- Scaled data sets won't fit in the memory of one GPU or even of many nodes → use NVMe
- Can't reach all data via loads and stores → need new API
- New workloads that are bottlenecked on data access vs. compute
- Key-value/object stores are gaining traction as a way to access data → custom APIs for objects
- Too much data for apps to track → serverless, with dataset services, orchestration

A new class of problems on scaled data

GPU becomes not only a compute monster, but also a fine-grained data access engine
Both use O(100K) threads to accelerate, compute or IO

- Huge data that are too big to reach with loads and stores
  - Partitioning, caching, communication complexity
  - Error handling at scale is problematic
  - NVSHMEM for memory; something more for mem+storage
    → new API family that covers data anywhere
- Accesses are initiated from the GPU (or CPU)
  - GPUDirect Async Kernel Initiated Storage, not just GDS
  - Example: graph traversal based on reading node data
- Vast volume of accesses, 1+ per GPU thread
  → greatest benefit with fine granularity





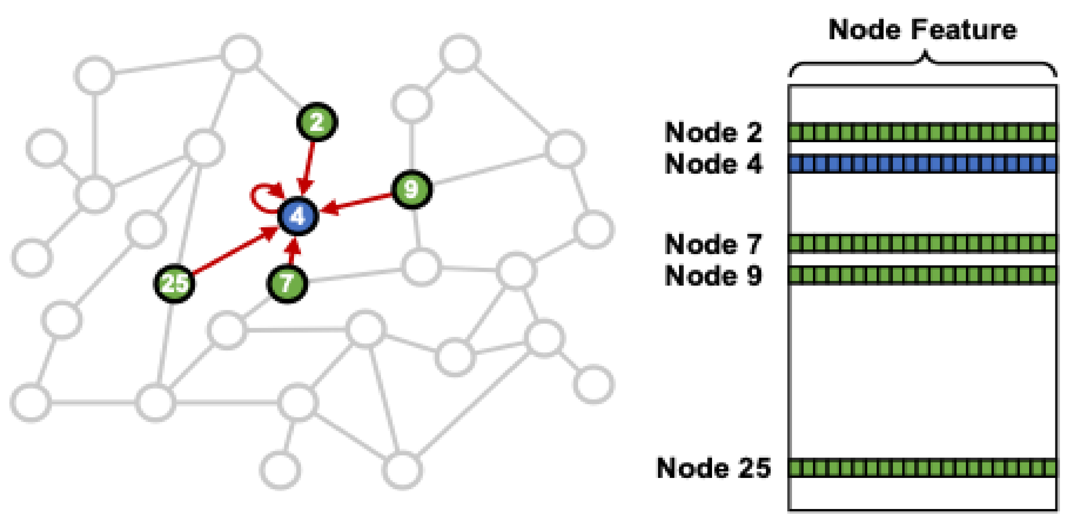
How do we most efficiently squirt
O(100K) requests/responses through the PCIe pins?

New class of applications → new programming model

Fast, sparse access to massive data

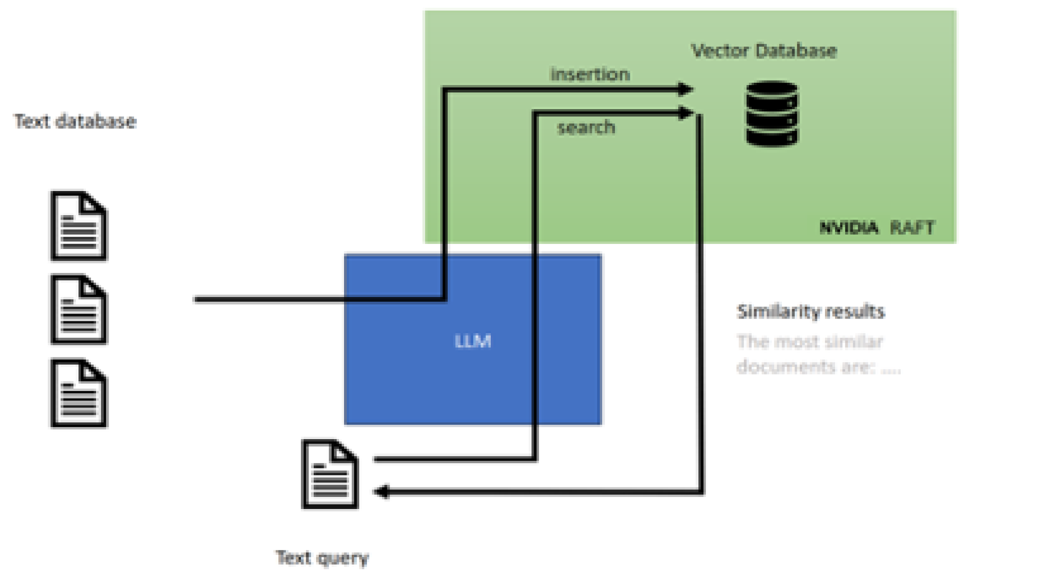
- **GPU becomes not only a compute monster but also a data access monster**
- Huge volume of fine-grained accesses from each of O(100K) GPU threads

- For huge data sets, you eventually can't do load/store → this is the new API for data of unbounded size in memory/storage
- NVMe brings compelling TCO vs. HBM/DRAM
  - Open 1T edge graph problems to those with only 1 GPU or 1 node
- Relieve the "out of memory" management for greater productivity



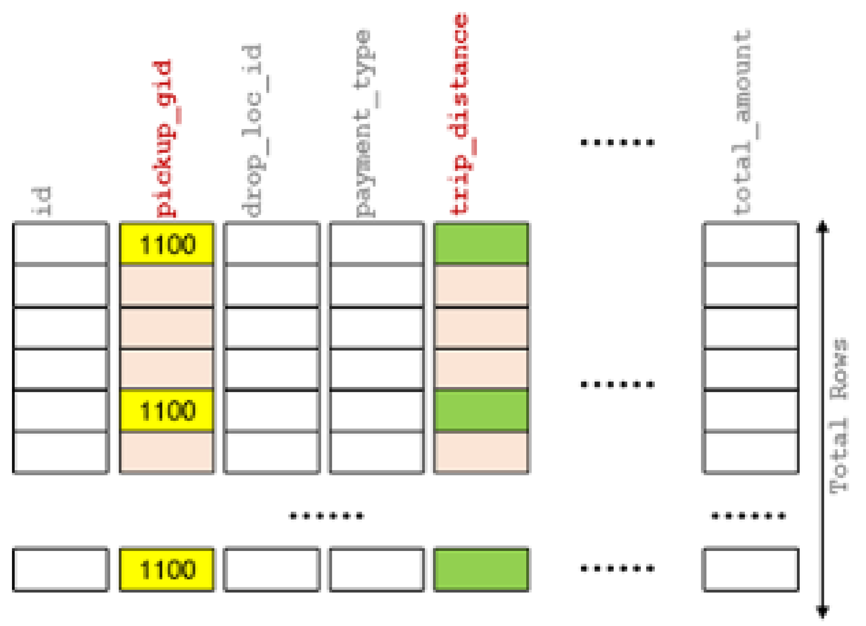
Graph analytics and Graph neural networks (100GB-100TB) Nodes/edges/embeddings

WholeGraph, cuGraph, …



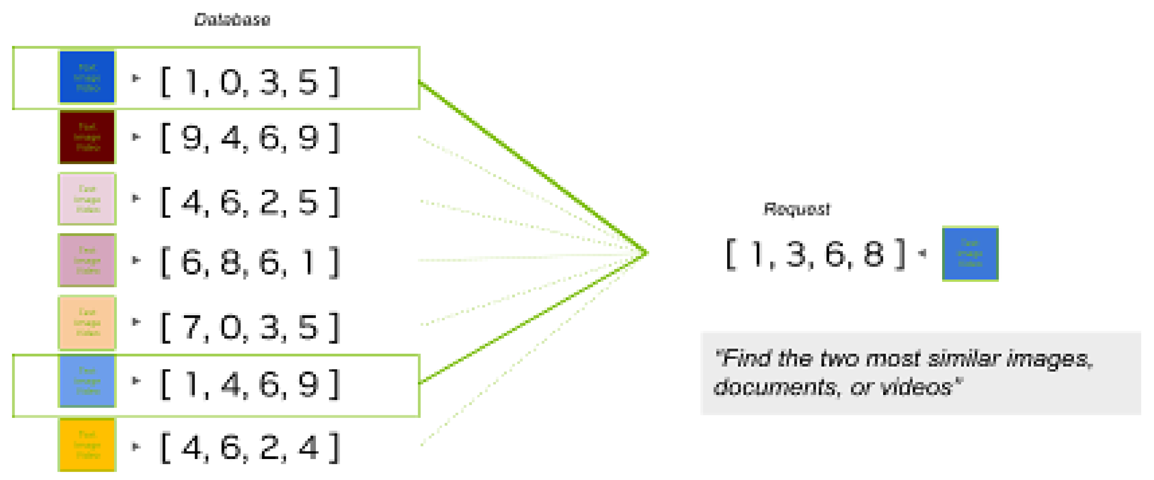
RAG/VectorDB (>600GB) ANN algos on embeddings

cuVS



Data analytics (100GB-1PB) select row/column based on compute

RAPIDS



Vector Search (up to 40PB) specialized algos on embeddings and files

cuVS

Emerging application domains that motivate a new programming model

Applications

- Graph Neural Networks (GNNs) – graph + feature store
  - Neither the graph nor the data fit into a GPU for 1T edges
  - High-value embeddings for entities and relationships
  - Key parts of recommendation and bad-actor detection systems
  - GNNs improve accuracy over other embedding types
- Vector search/vectorDB – vector store
  - NeMo Retriever, NVIDIA RAFT in RAG-LLM
  - Data deduplication to prep for foundational training of trillion-token LLMs
- LLM fine-tuning joint with GNN embeddings benefits from huge key value service
- Graph analytics available in cuGraph:
  - Personalized pagerank, community detection on huge graphs
  - Distributed sampling and partitioning for GNN models

Common need: simple management of data larger than physical memory of host + device

- Avoid OOM (Out of Memory) errors
- Typically requires caches, partitioning, multi-GPU/multi-node communication
- Needs to be re-created for each application unless we have a common solution

Characteristics and usages across scale

system scale	1 GPU discrete 1-10TB, tabular data Local NVMe	8 GPU HGX 20-40TB, 3D proteins Local or TOR NVMes	256 GPU SuperPOD 100+TB, transaction graph TOR NVMes or RDMA filers
app domain	Data science	Molecular generative AI BioNemo, Pharma	Anomaly detection, RecSys FSI, cybersecurity, retail
usage model	Exploratory data analysis Model creation Train a couple of models overnight	Input knowledge graph Build hetero molecular graphs Molecular diffusion inference Docking analysis	Load and build graph Sample and train Inference to create embeddings

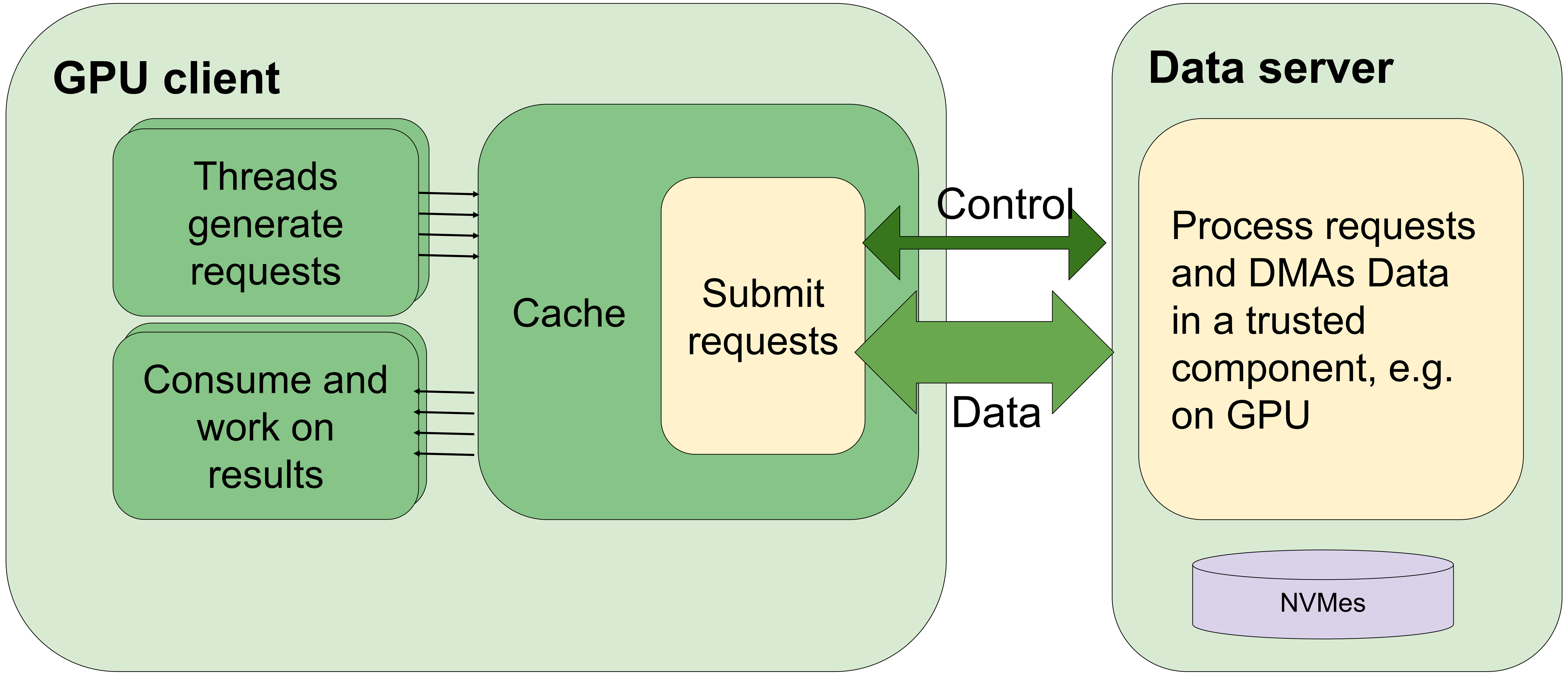
Application layering example

Delivering new capabilities through existing stack

- cuGraph service implementation changes, application does not
- Now training can proceed independent from data size
- No need to manage memory system, caching, partitioning, big improvement in maintainability for GNN training at scale

GPU-initiated scaled data architecture

GPU becomes an autonomous highly parallel data access engine



- Request, initiation, service, and consumption all happen within a GPU kernel
- GDA KI Storage enables data IO accesses that are both initiated and triggered by GPU
- Requests are processed in a trusted, privileged server with access to storage
- Features a key pillar of Magnum IO: flexible abstraction

Simplified architecture

3 views: user, tiered, backing storage

User view

Tiered view = SCADA

Backing storage view

Simplified architecture

3 views: user, tiered, backing storage

User view

Tiered view = SCADA

Backing storage view

Implications

- Huge, distributed data
  - ↑ separation of data from work → tiered/hierarchical locality with data orchestration
- Abstraction over complexity
  - Provide easy onramp in addition to near-full control
  - Handle fragility of unreliable storage and its error conditions
  - Retain cost transparency - queriable cost, if not directly implied by API
- User interfaces
  - Could be new for SOL but must also support legacy
  - Set of non-owning views; vernacular data collections vs. just mdspan
  - Provide usage hints
  - Dev/tuner/DC admin specify preprocessing and transformation down in data network
    - Both config - this might happen, between whom and how
    - And on demand - get me this

Rethinking interfaces for the modern data center

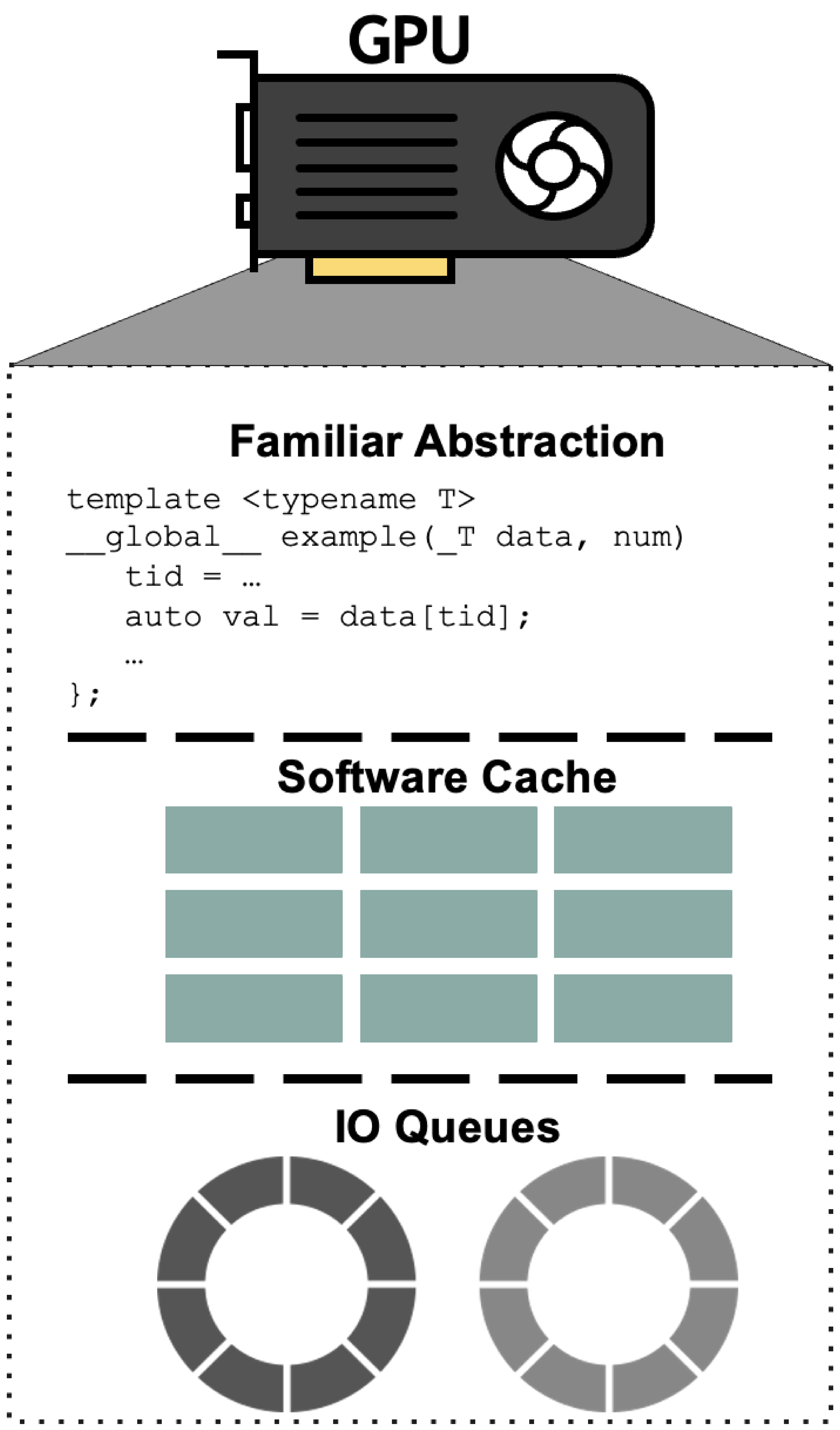
Internal name: **SCADA for scaled accelerated data access**

- **Scale:** Single API for data access independent of scale
  - Fit where you couldn't before, e.g. 10 TB in one node, avoid OOM worries
  - Transparently scale both data set size and size of compute cluster
- **Higher abstraction:** "Serverless access" is the way of the modern data center
  - Front end: handle caching, avoid partitioning, communicate among multi-GPU/multi-node
  - Back end: app accesses dataset X, relegates details of where/how data is stored
  - Data platform tools could manage curation, locality, sharding, staging
  - Acceleration with best use of GPU threads, memory management, and topology-tuned communication
- **Easy enablement:** Low-level interface that leaves application layers unchanged
- **Fundamentally-low TCO:** Reduce the cost of storage data
  - Huge data → huge memory → huge cost
  - Applications of low computational complexity use HBM only for memory vs. compute
  - Cheap NVMes make datacenters more efficient

Two SCADA research prototypes: BaM, GIDS

Preparing for trial integration into production stacks

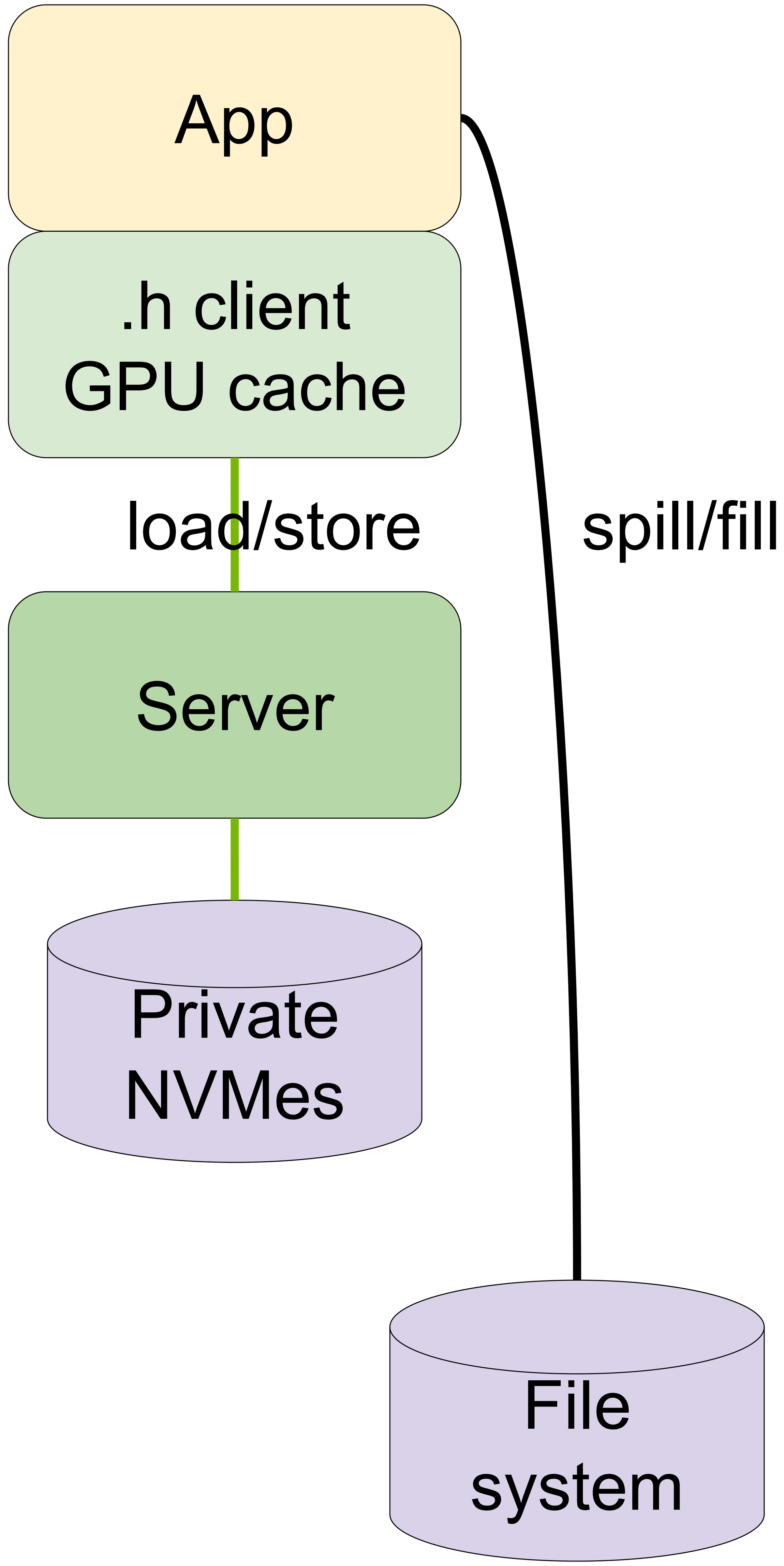
- Follow-on to an earlier OSS academic prototype
  - **Big accelerator memory**, **BaM**: “GPU-Initiated On-Demand High-Throughput Storage Access in the BaM System Architecture”, ASPLOS 2023: https://doi.org/10.1145/3575693.3575748
  - **GIDS**: “Accelerating Sampling and Aggregation Operations in GNN Frameworl with GPU-Initiated Direct Storage Accesses”, VLDB’24: https://arxiv.org/abs/2306.16384
- Currently a functional prototype, first used by cuGraph
  - Easy integration into widely used package manager
- Templated C++ header library, specialized for app objects
  - Familiar programmer abstractions
- GPU cache aggregates to a smaller number of IOs
- Optimizing IO queue interactions for O(100K) GPU threads



Progression of SCADA services

Start with simpler cases, grow over time with your input

- Common infrastructure
  - Header-centric client library in front of opaque implementation
  - Memory managed by app, SCADA provides APIs to allocate and free
- Start with a simple but critical service like swap, then extend
  - App on CPU reads all data from storage into GPU, as it always has
  - GPU threads write data into SCADA and read it back later
  - Relieve out of memory (OOM) avoidance with unbounded capacity
  - API for contiguous arrays
- We'd love your feedback for the next APIs and services



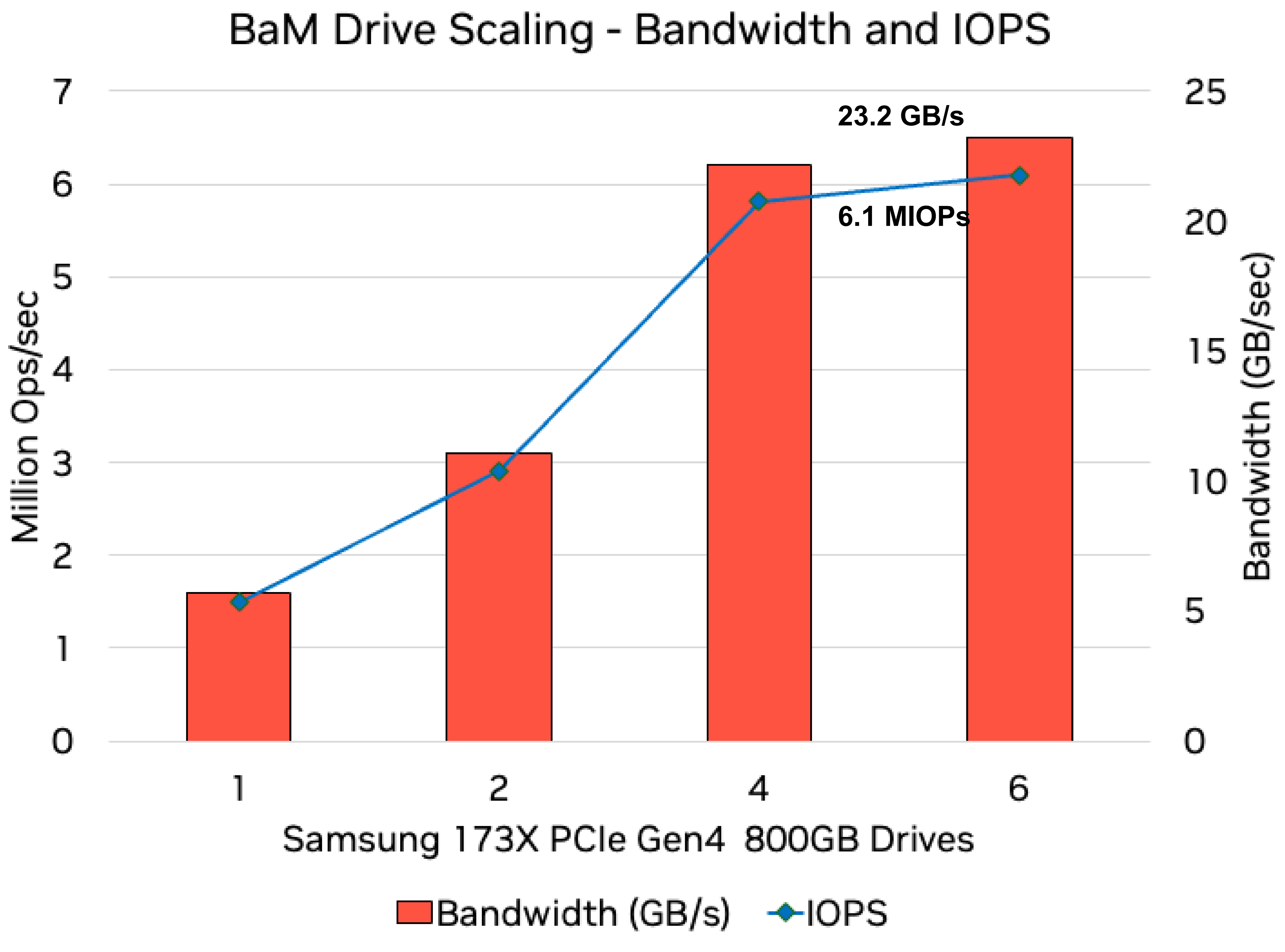
Performance results: the GPU as a data access monster

Bottleneck is NVMe and pin bandwidth, not GPU code

- 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 for Gen4)
  - IO processing (16 MIOPs) keeps up with PCIe-saturating NVMe IOPs rates (6 MIOPs for Gen4)
- GPUs are latency tolerant - HW context switching covers miss latency

Random reads mostly saturate PCIe Gen4 with 4 Gen4 drives

Initial Big Accelerator Memory (BaM)* research prototype validates perf trends



DGX A100 GPU

- UIUC-NVIDIA BaM replaces the NVMe driver to enable GPU-initiated IO transfers to/from NVMe
  - 6+ Million IOPs, 23+ GB/s on 4KB random reads
  - 0% CPU utilization
- BaM NVMe Block Bench
  - Microbenchmark to stress storage through BaM
  - Scales GPU requests at 4KB against storage devices and measures operations/s and GB/s in 4KB xfers.
  - 6 drives vs. 4 bumps up MIOPs and GB/s slightly

BaM and GIDS are UIUC-NVIDIA Research prototype projects and not intended for general release.

Explicit storage IO is 25x of mmap, faster media is better

Direct GPU access vs. faulting through CPU to storage with BaM and high-performance Gen5 NVMe™ brings 25+x for GNN Training

Workload Execution Time (seconds)	Baseline (mmap) Gen4/A100	BaM Enabled Micron Gen5/H100	BaM Enabled Micron Gen4	Gen5 v Gen4 Performance
Feature Aggregation	1,130 (99%)	18.6 (83%)	35.0	**2x**
Training	2.1 (0.2%)	0.73 (3%)	3.6	**5x**
End-to-End	1,137	22.4	43.2	**2x**
E2E Improvement over Baseline		**50x**	**26x**	
Feature Aggregation Improvement		**61x**	**32x**	

Feature Aggregation depends on SSD performance
It's 99% of execution time in the baseline, 80% of tuned
Sampling and training depend on GPU performance

GNN on GPU induces queue depths 10-100x of CPU

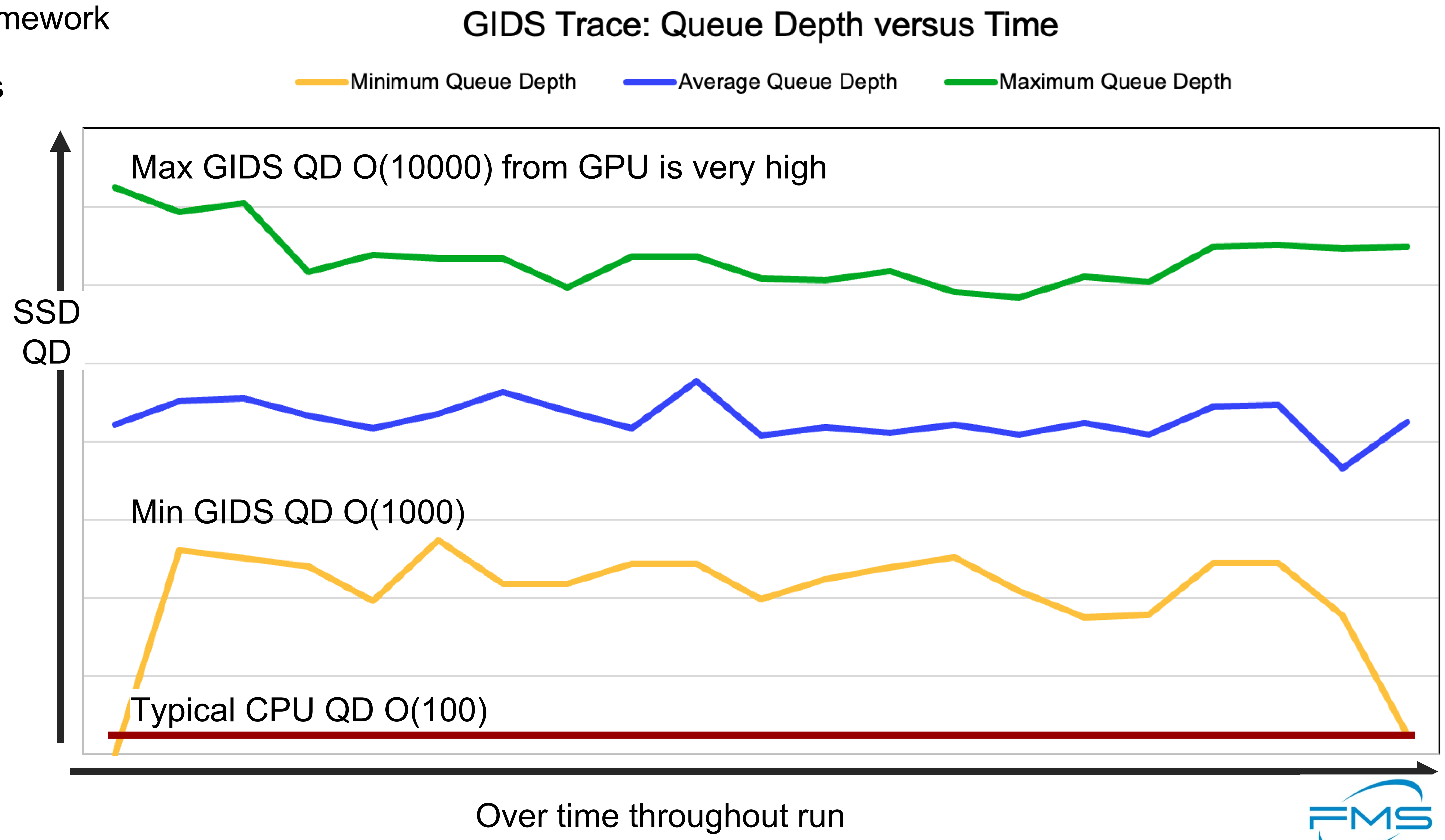
Investigated with Micron NVMe™ IO Trace tool

Using **GPU-initiated direct storage (GIDS)** framework

A trace of the IO pattern at the SSD level shows interesting behavior:

- Near drive's max IO performance
- 10-100x SSD queue depth wrt CPU
- 99% small block reads

GIDS with BaM presents a challenging SSD workload:
High-Performance NVMe is Required



GIDS with IGBH-Full training. NVMe IO trace measured by Micron's Data Center Workload Engineering team.
System under test: 2x AMD EPYC 7713, 64-core, 1TB DDR4, 4 Micron 9400 PRO 8TB, 1x NVIDIA A100-80GB GPU, Ubuntu 20.04 LTS (5.4.0-144), NVIDIA Driver 535.113.01, CUDA 12.2, DGL 1.1.2
Work based on paper "GPU-Initiated On-Demand High-Throughput Storage Access in the BaM System Architecture" https://arxiv.org/abs/2203.04910 and using https://github.com/ZaidQureshi/bam

Suitability

GPU-initiated, dynamic, per thread

- Each GPU thread dynamically forms and makes its own request
- If you know the batch ahead of time, use GPUDirect Storage

Fine-grained, high throughput

- Could be 4B-4KB
- If coarse, you can saturate pins with very few threads

Unbounded data size

- Can't reliably fit in GPU high-bandwidth memory
- If it could, you could just stick with load/store

Data access bound

- Focus is on data access as a bottleneck
- If compute bound, HBM in very many nodes is free

Benefits

Scale

- Fit problems into a small number of GPUs by spilling into NVMe storage
- Even if somewhat slower (e.g. 1 NVMe) – what's your slowdown threshold?

TCO

- For a given perf level, offer greater cost effectiveness with NVMes vs. HBM or DDR

Performance

- As we tune over time, potentially improve perf
- Limited by PCIe bandwidth into GPUs and # of drives

Call to action for SCADA

Come help chart the future of turning the GPU into a data access engine

- Storage technologists
  - Give us lots of IOPs!
  - Pack in fine-grained transfers across PCIe
- App developers and users
  - Share need for more data capacity than will fit in GPU-CPU memory for compute
  - Specify kinds of services of interest, e.g. array, swap, key-value, VectorDB, dataframe?
  - Specify details on product stack support, deployment models
- Infrastructure developers
  - Layer on SCADA as has been done for NVSHMEM, e.g. Kokkos perf-portable framework
  - Look at new venues for fine-grained interleaving of compute and communication, e.g. LLNL