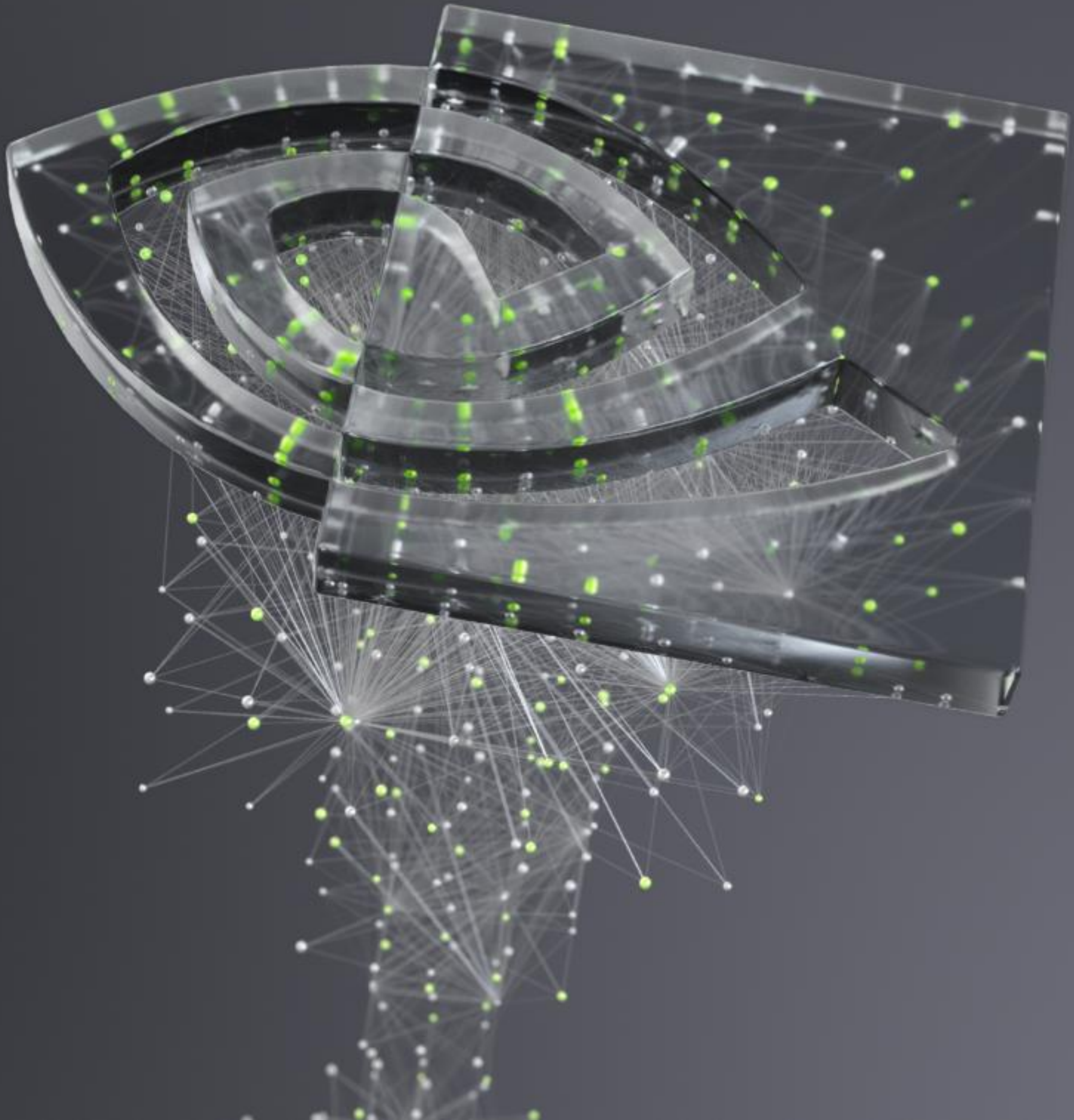
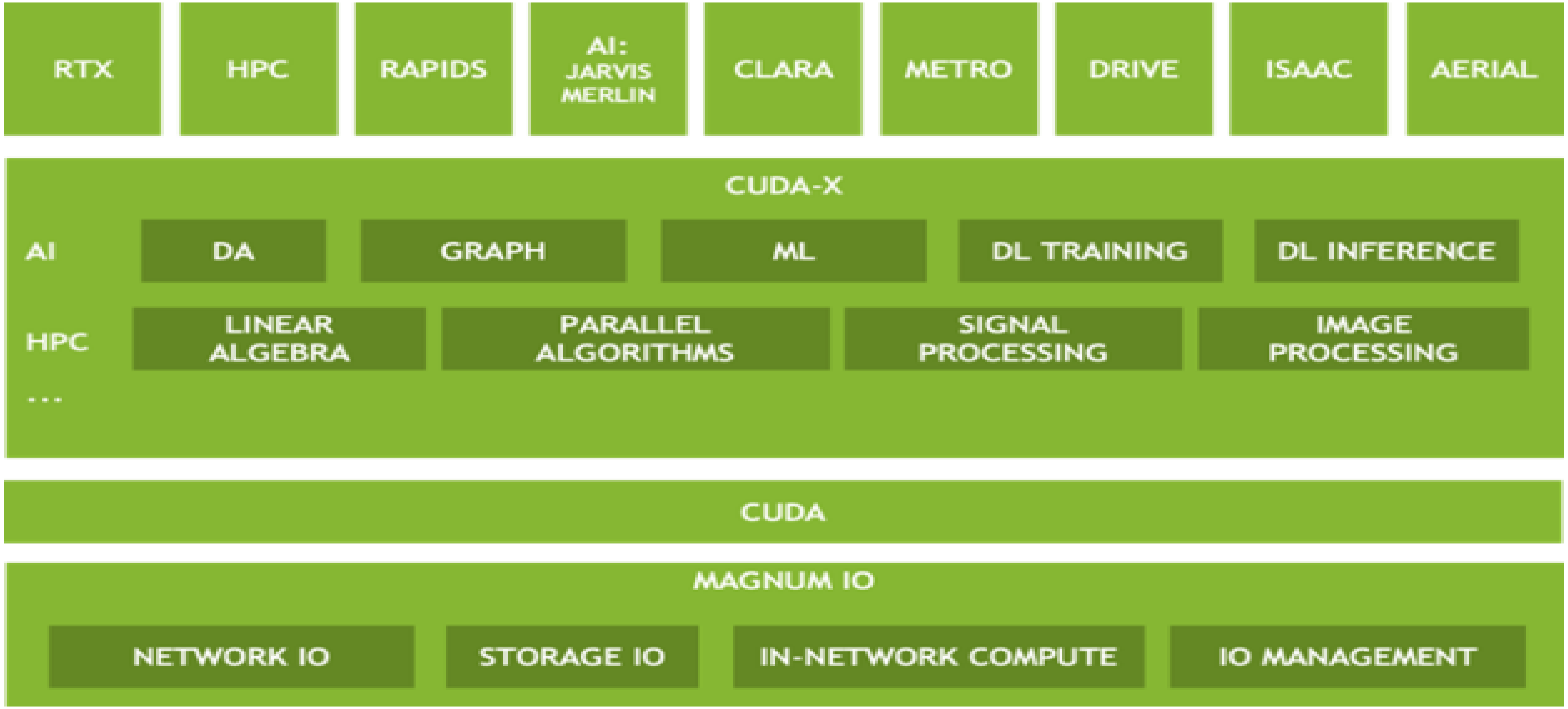


NVIDIA GDS

Sungta Tsai, Staff Field Application Engineer

MAGNUM IO



RTX

HPC

RAPIDS

AI:
JARVIS
MERLIN

CLARA

METRO

DRIVE

ISAAC

AERIAL

CUDA-X

AI

DA

GRAPH

ML

DL TRAINING

DL INFERENCE

HPC

LINEAR ALGEBRA

PARALLEL ALGORITHMS

SIGNAL PROCESSING

IMAGE PROCESSING

...

CUDA

MAGNUM IO

NETWORK IO

STORAGE IO

IN-NETWORK COMPUTE

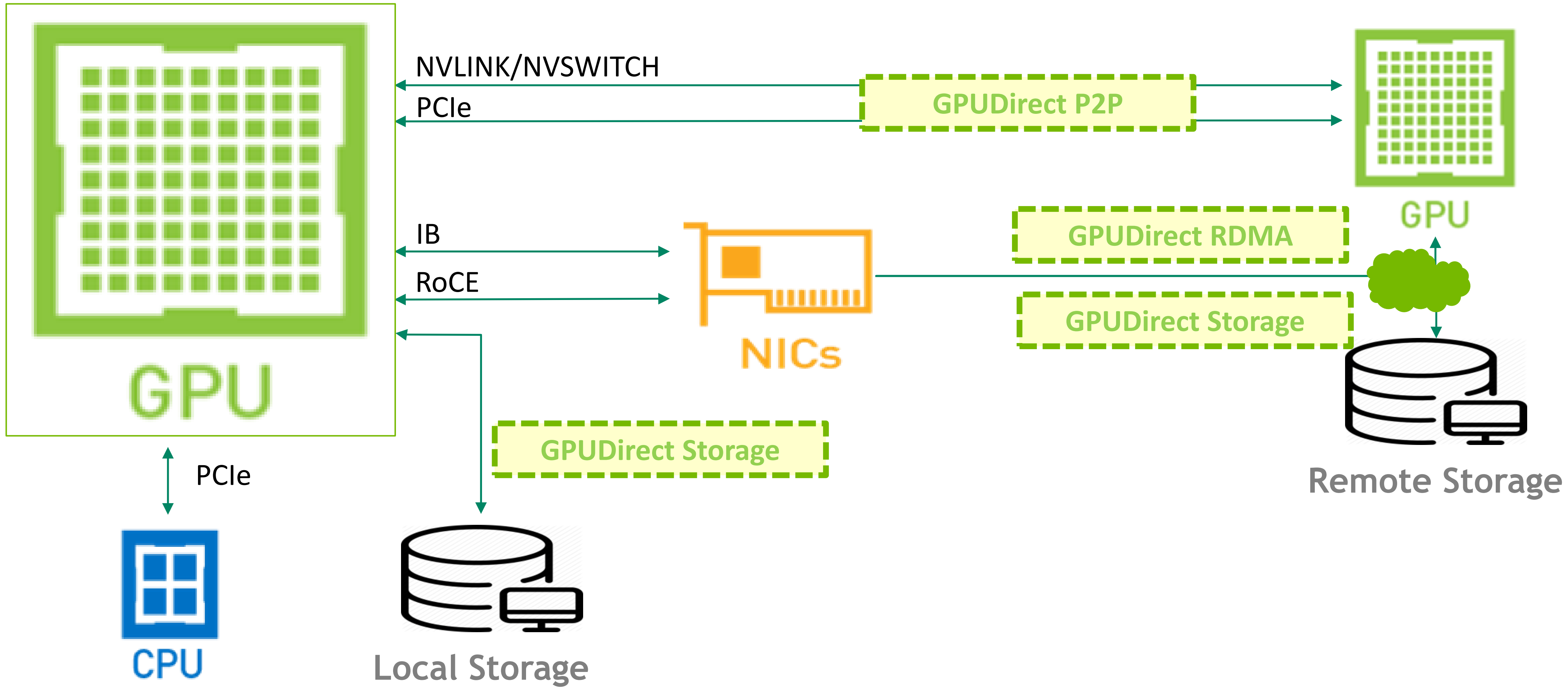
IO MANAGEMENT

MAGNUM IO

NVIDIA's Multi-GPU, Multi-Node Networking and Storage IO Optimization Stack

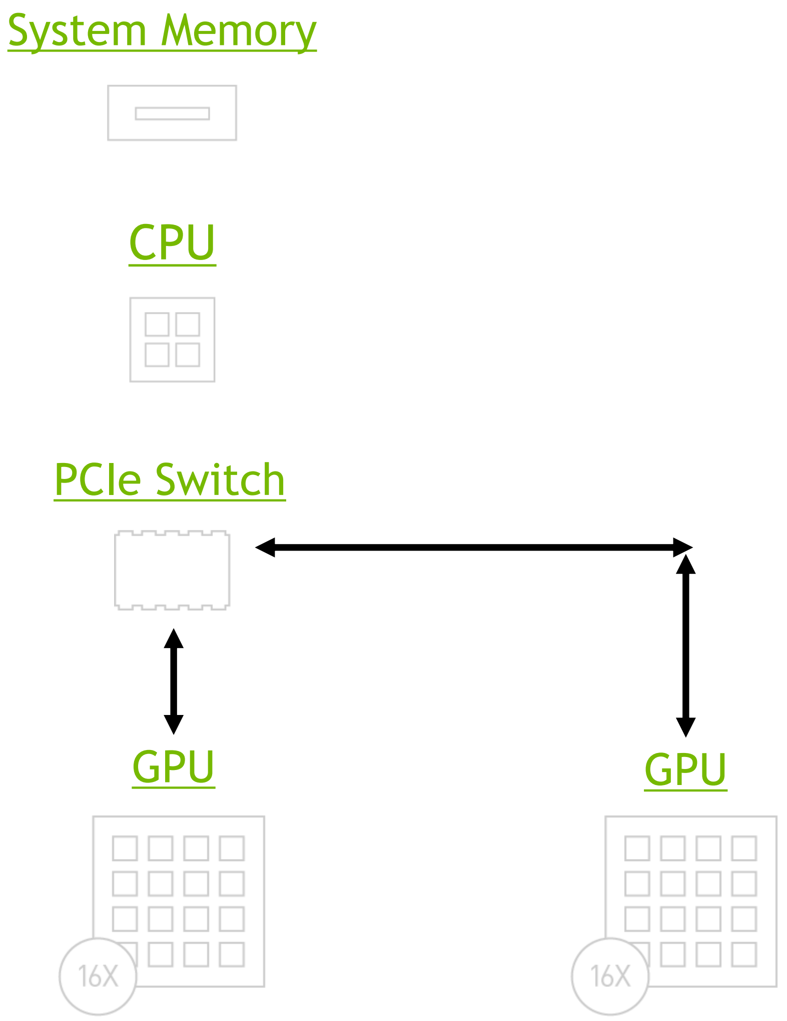
Magnum IO

Point-to-Point MPI, UCX, NVSHMEM…	Collectives MPI, NCCL,…	Storage cuFile



DATA I/O ACCELERATION

GPU Direct - P2P (Multi-GPU)



System Memory

CPU

PCIe Switch

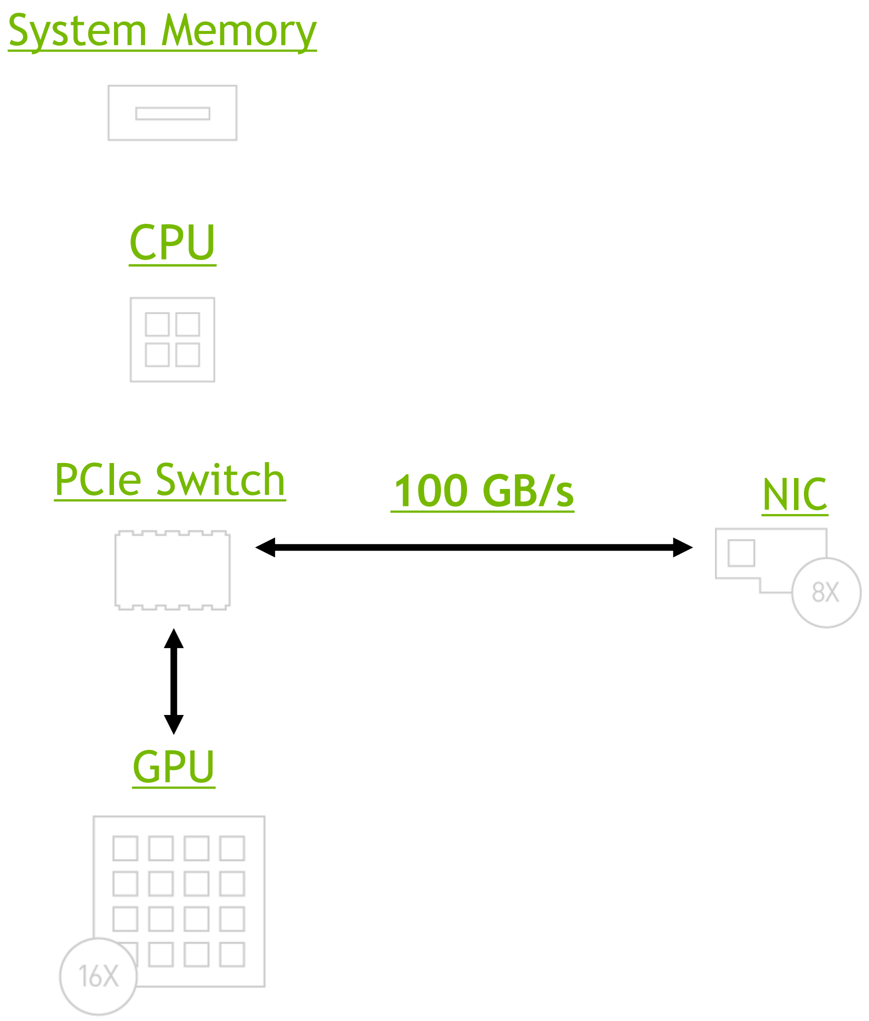
GPU

GPU

16X

16X

GPU Direct RDMA (Multi-Node)



System Memory

CPU

PCIe Switch

100 GB/s

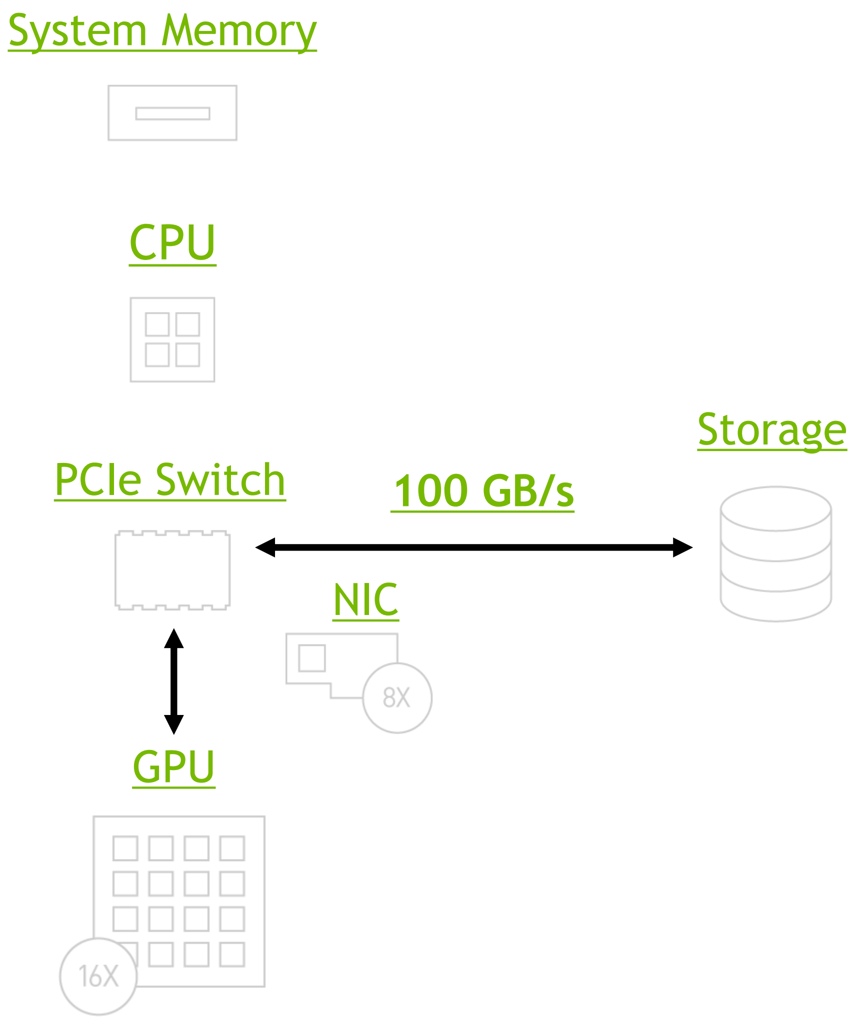
NIC

8X

GPU

16X

GPU Direct Storage (Storage)



System Memory

CPU

Storage

PCIe Switch

100 GB/s

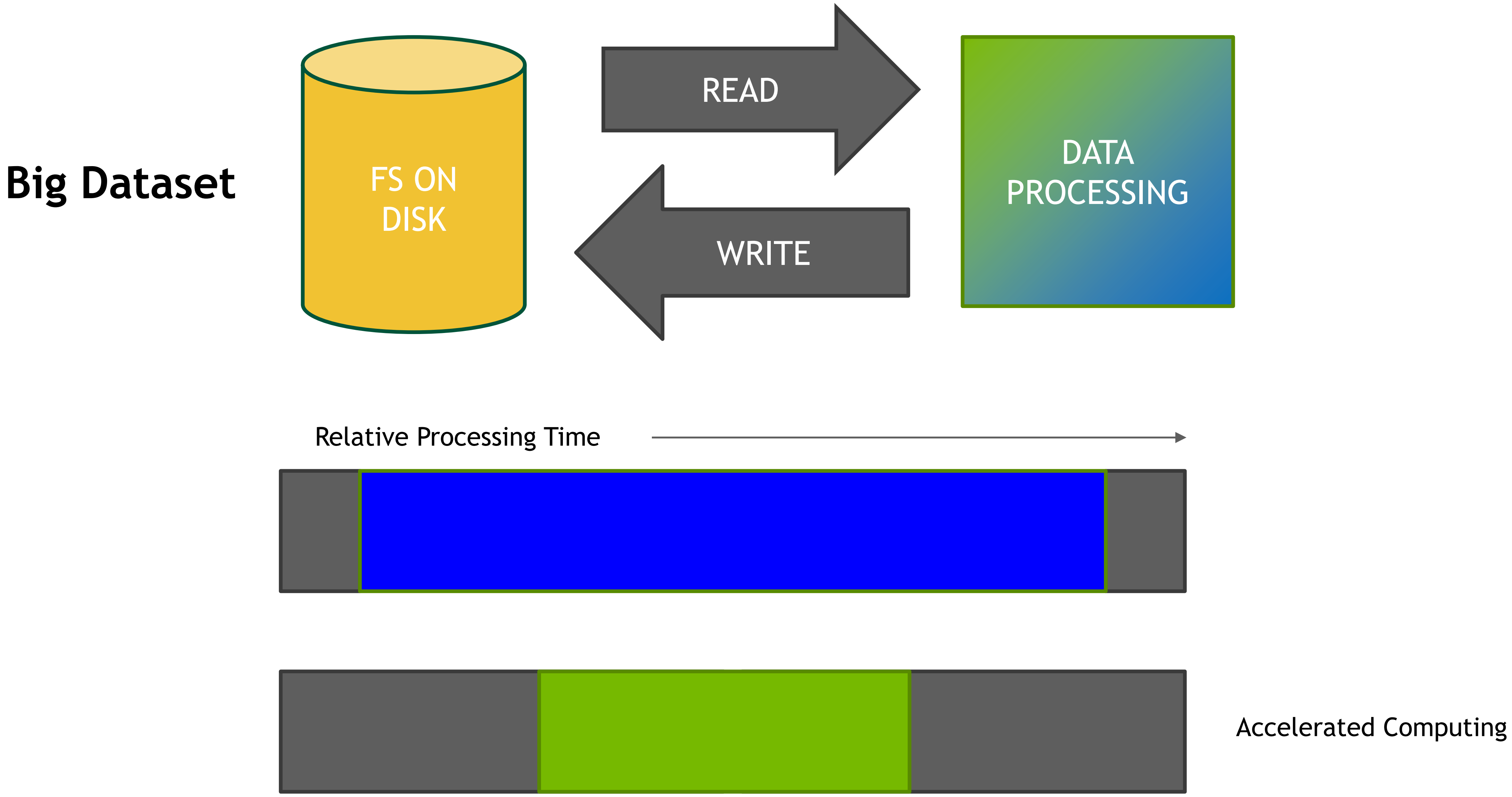
NIC

8X

GPU

16X

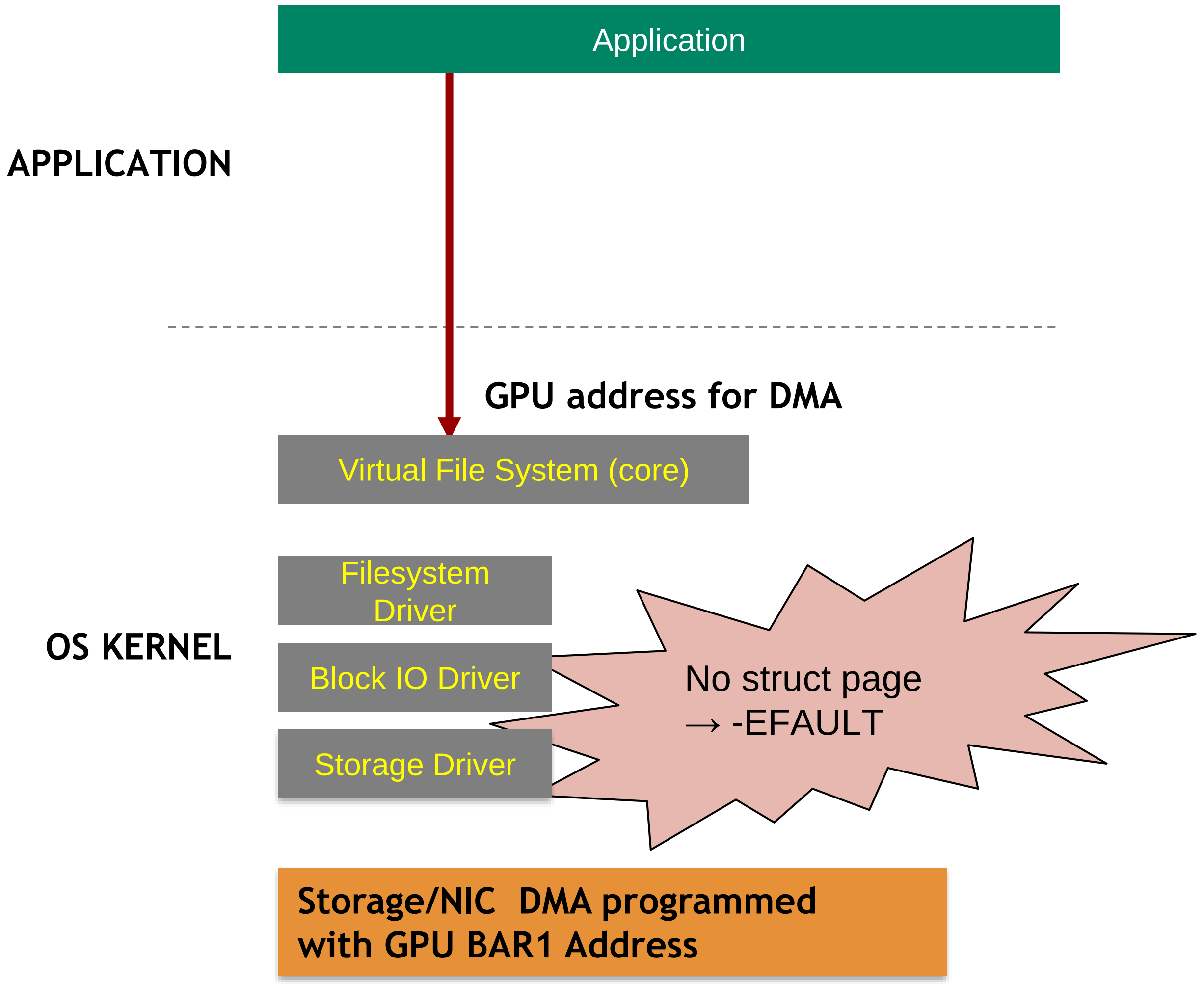
THE IO CHALLENGE

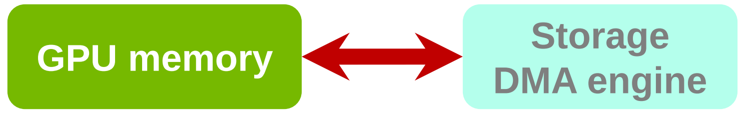


GPUDIRECT STORAGE SW ARCHITECTURE

User and kernel components

- Want to program DMA near storage to push/pull data in GPU memory
- Linux is not enabled to handle GPU Virtual Addresses needed for DMA
-

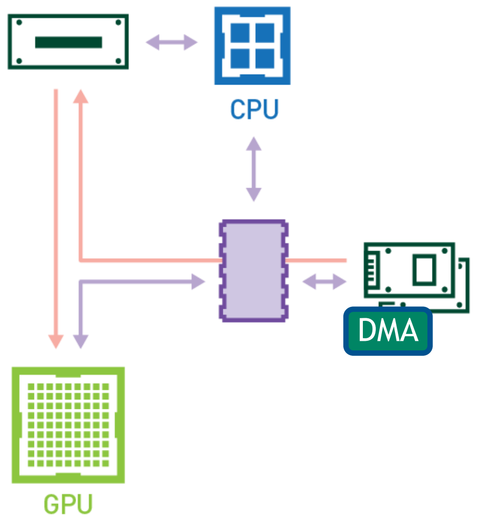


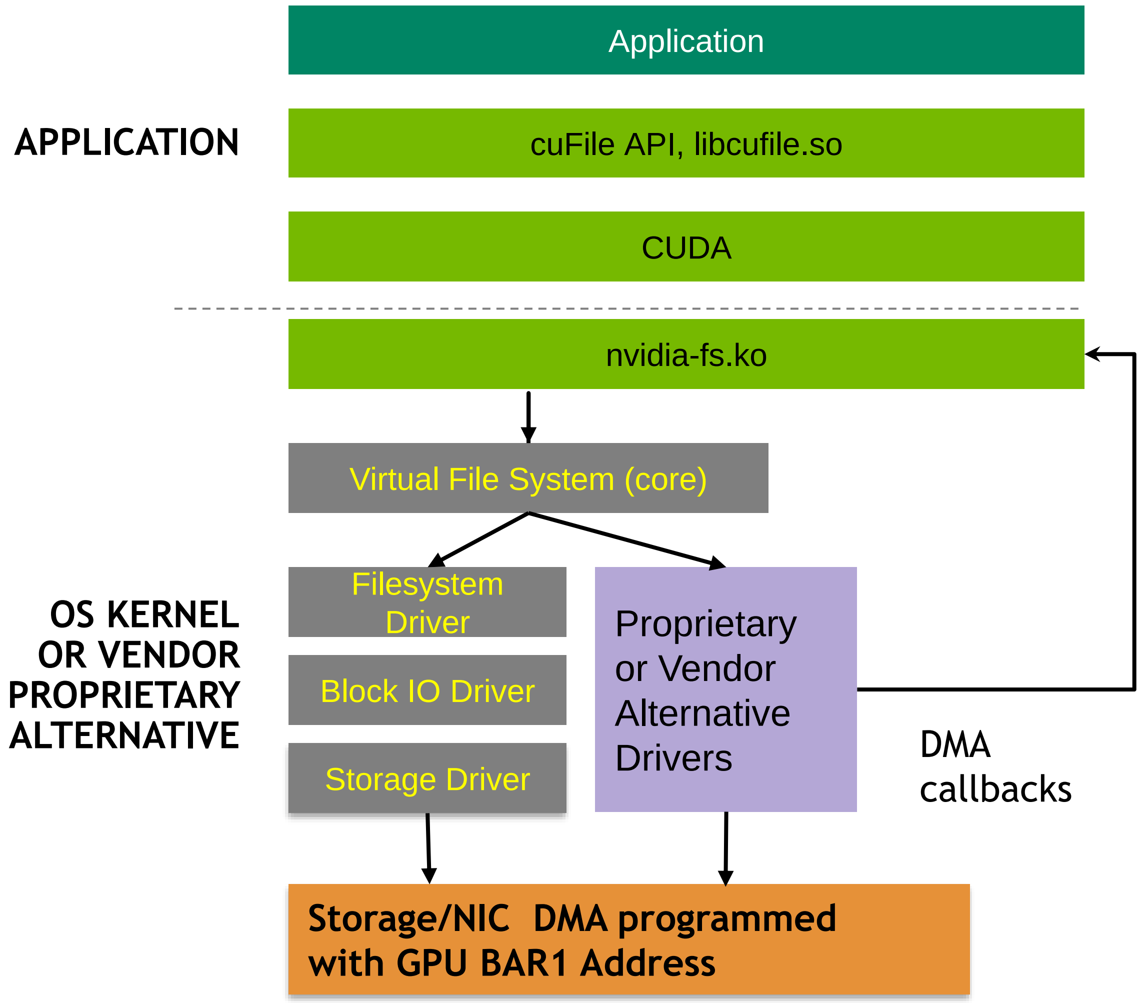


GPUDIRECT STORAGE SW ARCHITECTURE

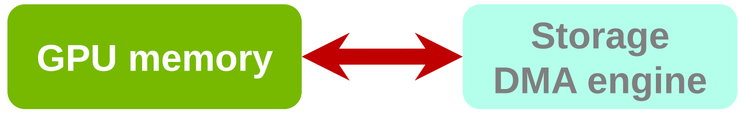
User and kernel components

- cuFile API
  Enduring API for applications and frameworks
- nvidia-fs Driver API
  For filesystem and block IO drivers
  Vendor-supported solutions: no patching
  avoid lack of Linux enabling
- NVIDIA is actively working with the community on upstream first to enable Linux to handle GPU VAs for DMA





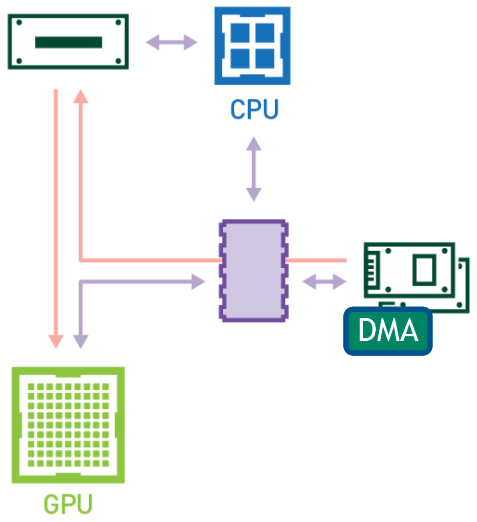
APPLICATION
Application
cuFile API, libcufile.so
CUDA
nvidia-fs.ko
Virtual File System (core)
OS KERNEL OR VENDOR PROPRIETARY ALTERNATIVE
Filesystem Driver
Block IO Driver
Storage Driver
Proprietary or Vendor Alternative Drivers
DMA callbacks
Storage/NIC DMA programmed with GPU BAR1 Address

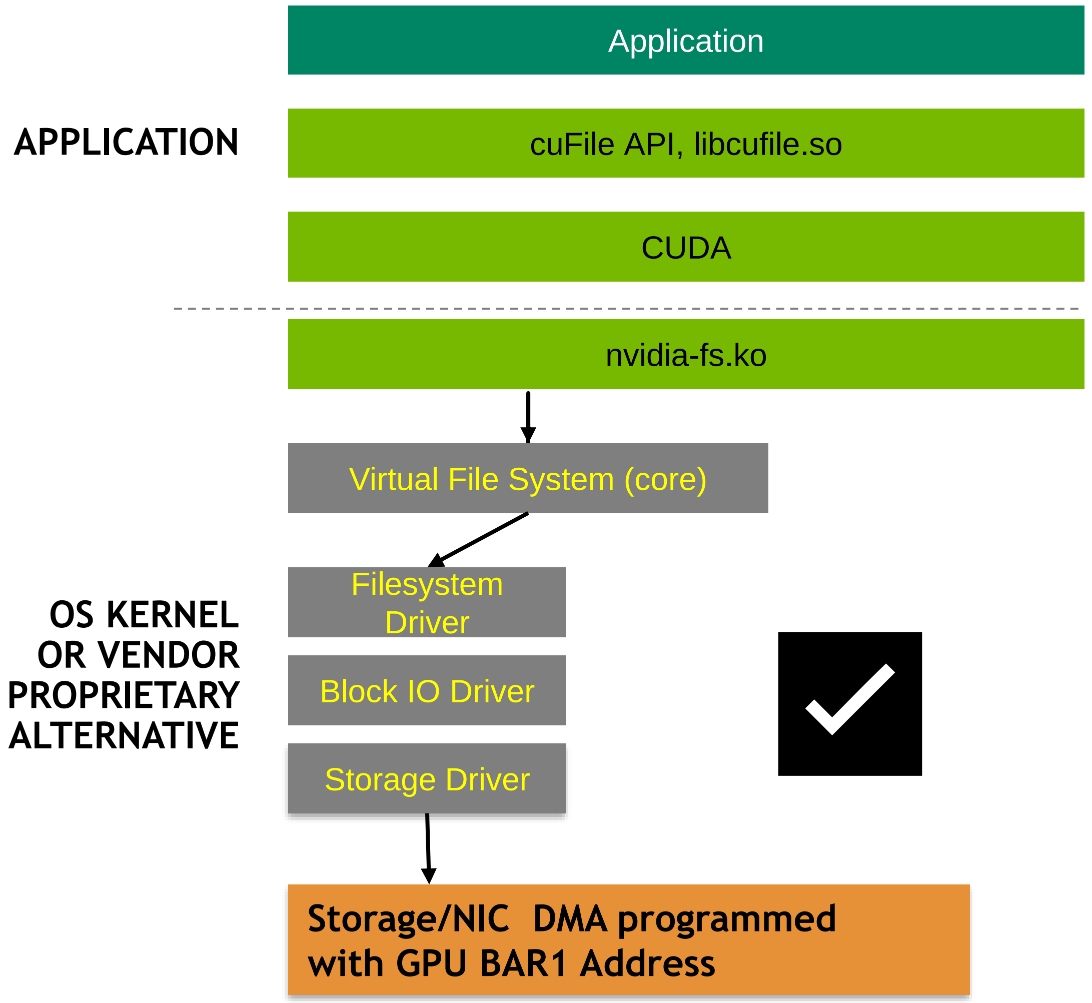


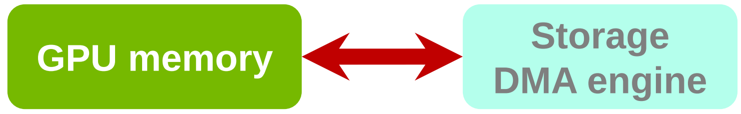
GPUDIRECT STORAGE SW ARCHITECTURE

User and kernel components

- cuFile interfaces will endure
- NVIDIA is actively working with the community to enable Linux to handle GPU Virtual Addresses needed for DMA
- We are increasingly open sourced and are partnering with MLNX in upstreamed efforts



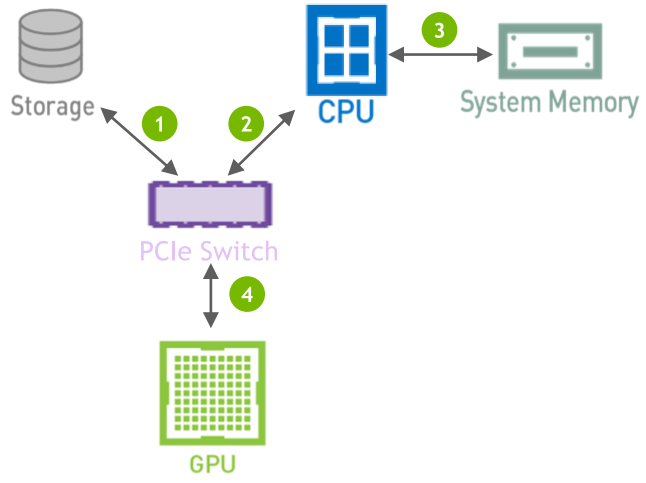

APPLICATION
Application
cuFile API, libcufile.so
CUDA
nvidia-fs.ko
Virtual File System (core)
OS KERNEL OR VENDOR PROPRIETARY ALTERNATIVE
Filesystem Driver
Block IO Driver
Storage Driver
✓
Storage/NIC DMA programmed with GPU BAR1 Address



DATA TRANSFER WITH GDS

Data Transfer without GPUDirect Storage

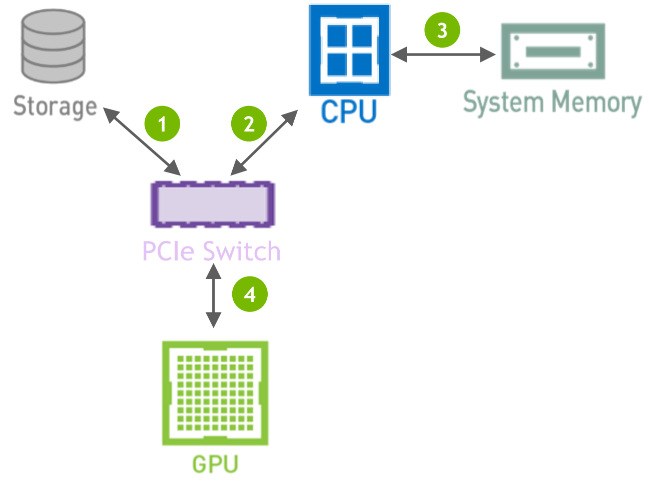
- fd = open(“file.txt”, O_RDONLY);
- buf = malloc(size);
- pread(fd, buf, size, 0);
- cudaMalloc(d_buf, size);
- acudaMemcpy(d_buf, buf, size, cudaMemcpyHostToDevice);



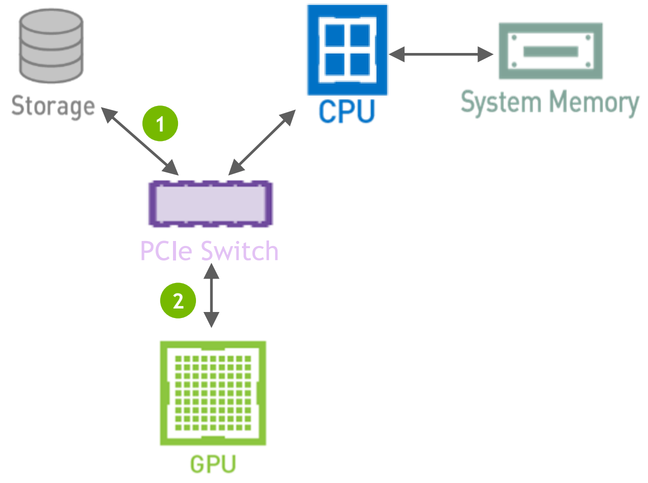
DATA TRANSFER WITH GDS

Data Transfer GPUDirect Storage

- fd = open(“file.txt”, O_RDONLY);
- ~~buf = malloc(size);~~ Don’t need to have “bounce buffer”
- pread(fd, buf, size, 0);
- cudaMalloc(d_buf, size);
- acudaMemcpy(d_buf, buf, size, cudaMemcpyHostToDevice);



- fd = open(“file.txt”, O_RDONLY | O_DIRECT, …);
- cudaMalloc(d_buf, size);
- cuFileRead(fhandle, d_buf, size, 0, 0);



cuFile APIs

```
/* cuFile File Registration APIs */
int cuFileImportExternalFile(CUFileHandle_t *fh, CUFileDescr_t *descr);
void cuFileDestroyFile(CUFileHandle_t fh);

/* core IO APIs */
ssize_t cuFileRead(CUFileHandle_t fh, void *devPtr, size_t size, off_t offset);
ssize_t cuFileWrite(CUFileHandle_t fh, void *devPtr, size_t size, off_t offset);

/* APIs to Register user specified buffer for direct BAR1 mapping */
CUfileError_t cuFileBufRegister(void *devPtr, size_t size, int flags);
CUfileError_t cuFileBufDeregister(void *devPtr);

/*APIs to control the Driver and cuFile resource lifecycle */
CUfileError_t cuFileDriverOpen();
void cuFileDriverClose();

/*APIs to tune the Driver and cuFile resource usage*/
CUfileError_t cuFileGetDriverProperties(CUfileDrvProps_t *props);
CUfileError_t cuFileDriverSetPollMode(bool poll, size_t poll_threshold_size);
CUfileError_t cuFileDriverSetMaxDirectIOSize(size_t max_direct_io_size);
CUfileError_t cuFileDriverSetMaxCacheSize(size_t max_cache_size);
CUfileError_t cuFileDriverSetBAR1Size(size_t max_bar1_size);
```

USAGE EXAMPLE

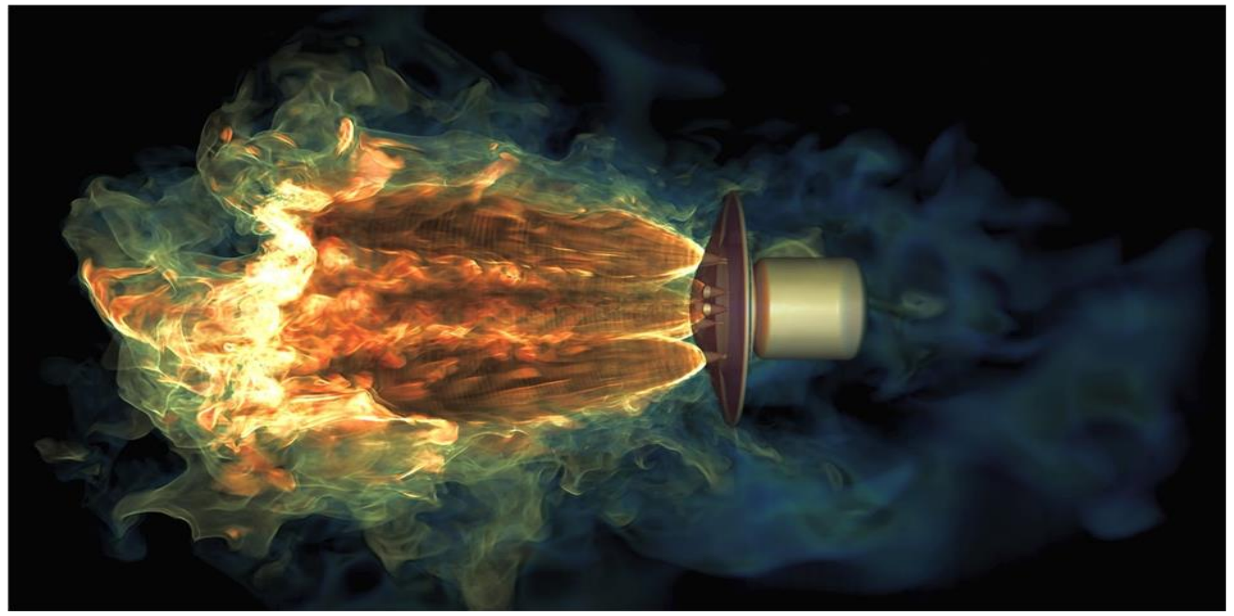
Write from GPU memory to local or remote storage

```
int main(void) {
 …
        CUfileError_t status;
        CUFileDescr_t cf_descr;                                    // general enough to support Windows
        CUFileHandle_t cf_handle;

        status = cuFileDriverOpen();
        fd = open(TESTFILE, O_WRONLY|O_DIRECT, 0644);              // interop with normal file IO
        cf_descr.handle.fd = fd;
        status = cuFileImportExternalFile(&cf_handle, &cf_descr);
        cuda_result = cuMemAlloc(&devPtr, size);
        status = cuFileBufRegister(devPtr, size);                  // performance optimization
        assert(cudaMemset((void *) devPtr, 0xab, size) == cudaSuccess);
        ret = cuFileWrite(cf_handle, devPtr, size, 0);             // ~pwrite: file handle, GPU address, size, offset

        status = cuFileBufDeregister(devPtr);                      // optional cleanup for good hygiene
        cuFree(devPtr);
        cuFileDestroyFile(cf_handle);
        close(fd);
        cuFileDriverClose();
        return 0;
}
```

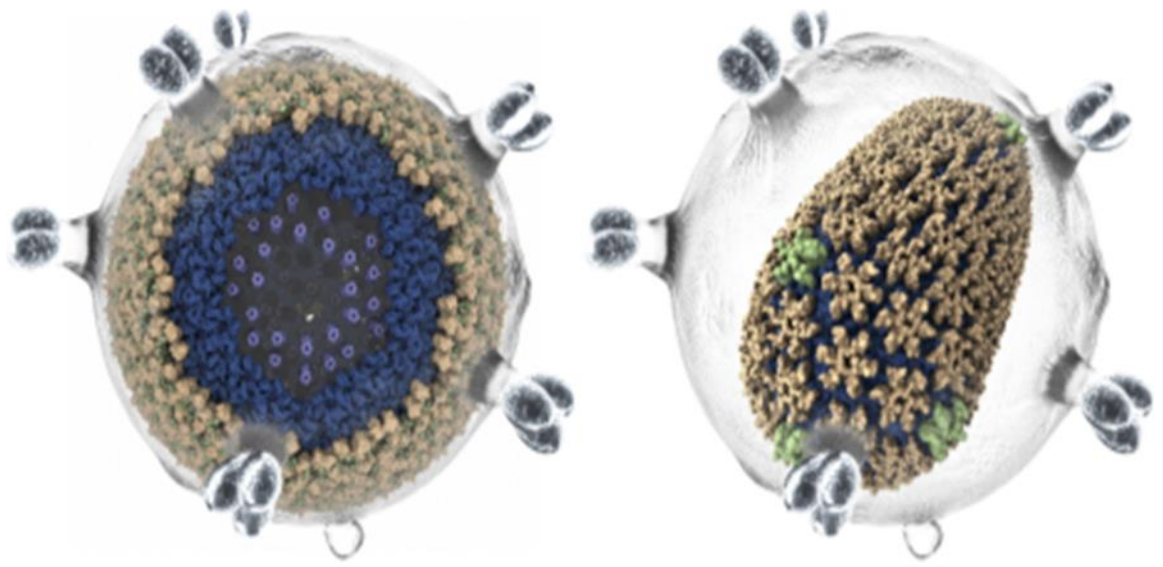
GPUDIRECT STORAGE – USE CASES



<u>NASA Mars Lander</u>

Simulation → visualization
128TB data, must stream in from remote
Part render, part IO; not quite linear in IO

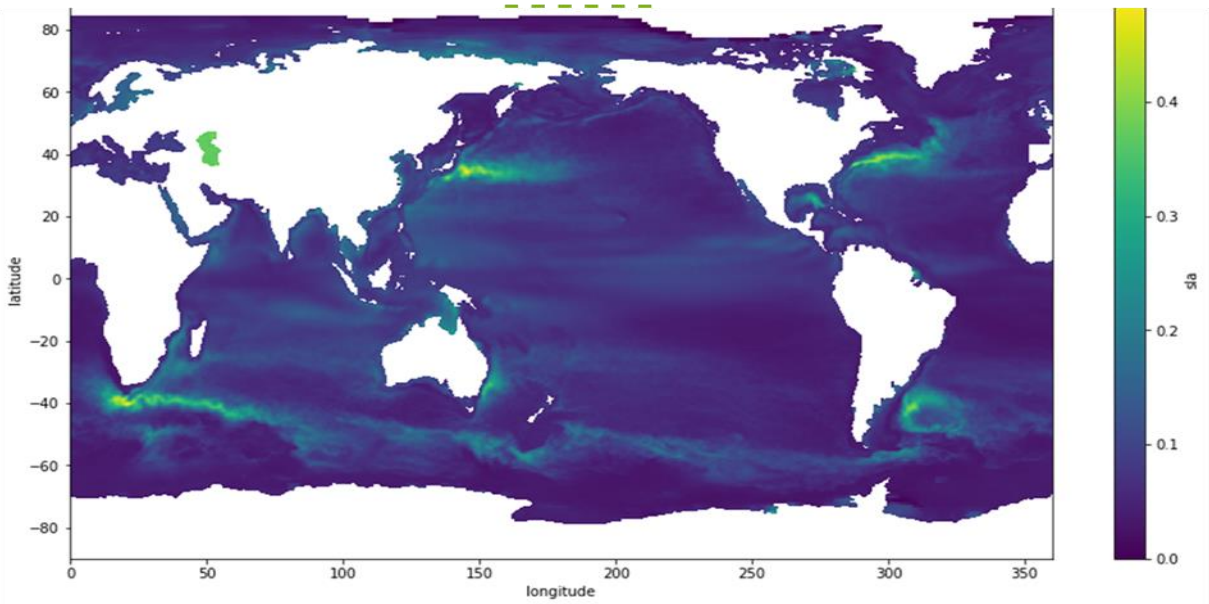
5 GB/s (1 DGX-2) → 160-180 GB/s (4 DGX-2s) with GPUDirect Storage



<u>Molecular Dynamics</u>

Simulation → analytics → visualization
30TB data, can be remote + local
O(N^2) IO problem to build dissimilarity matrix of macromolecule poses across time steps so can find stable configs

7 GB/s → 22 GB/s local, 60 GB/s remote
3x from GDS vs. heroic effort, 4x threads



<u>Pangeo Earth Science</u>

Simulation → DA, DL → visualization
100 TB–PB data, streamed from remote
Coming to GPUs because of faster IO
Increasing richness: DA, DL

Moving from 1 per day to 2-3 per day
What ifs vs. safe bets

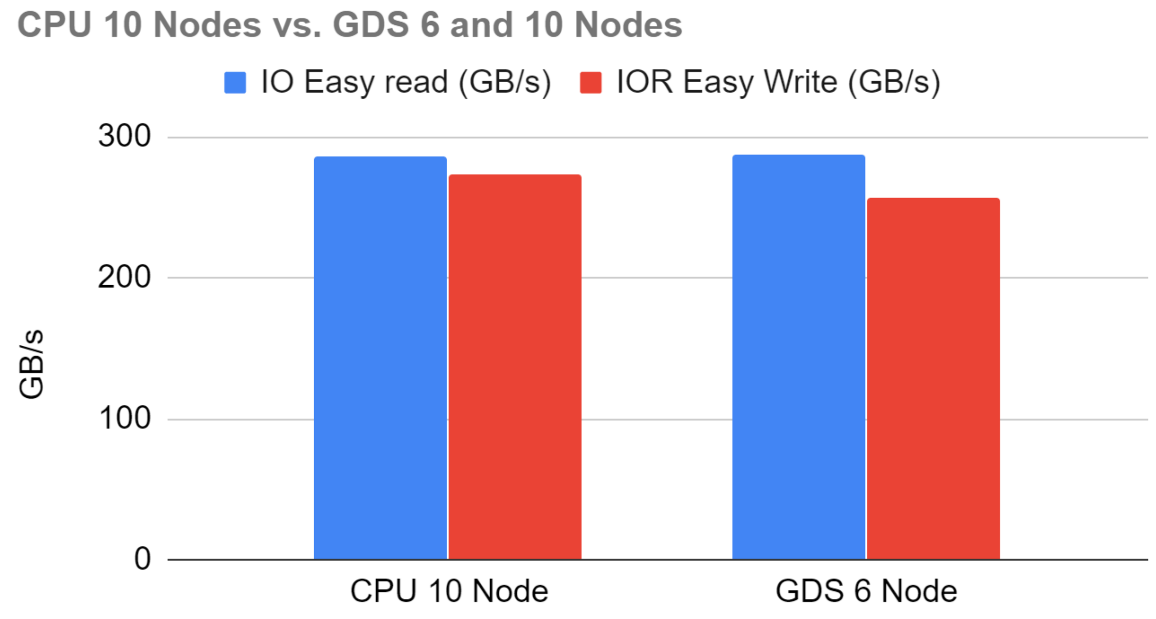
IO500

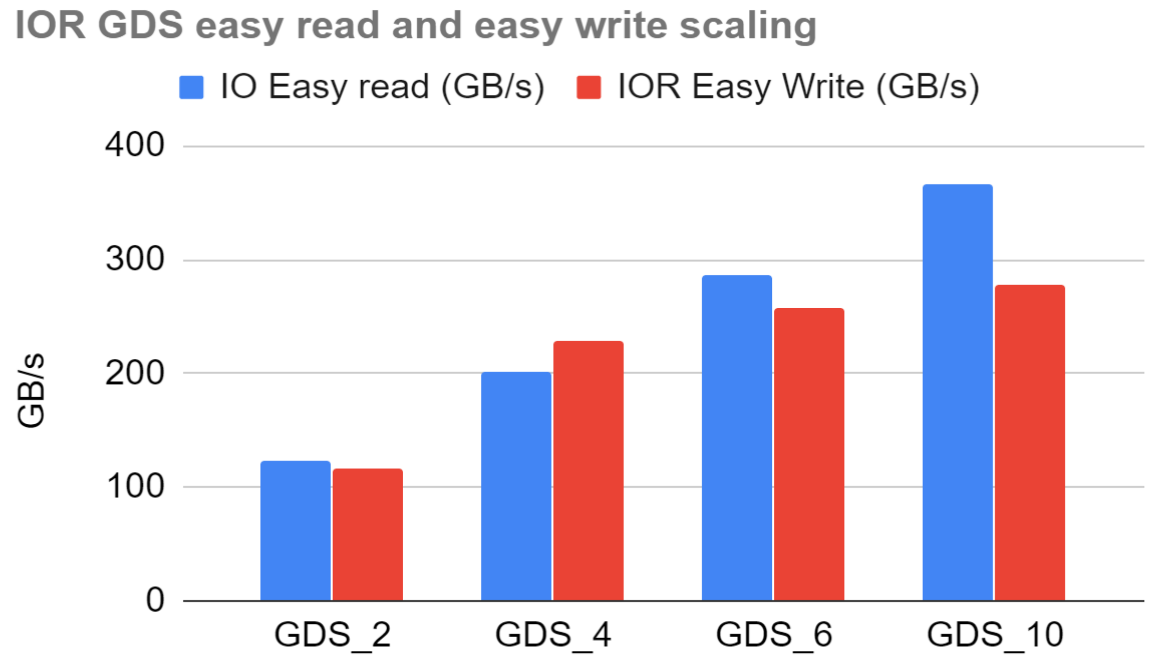
IO to compute dominates

DGX-2: 16 GPUs, 8 NICs, 2 CPUs

Move data directly to GPUs

Relieve CPU bottleneck

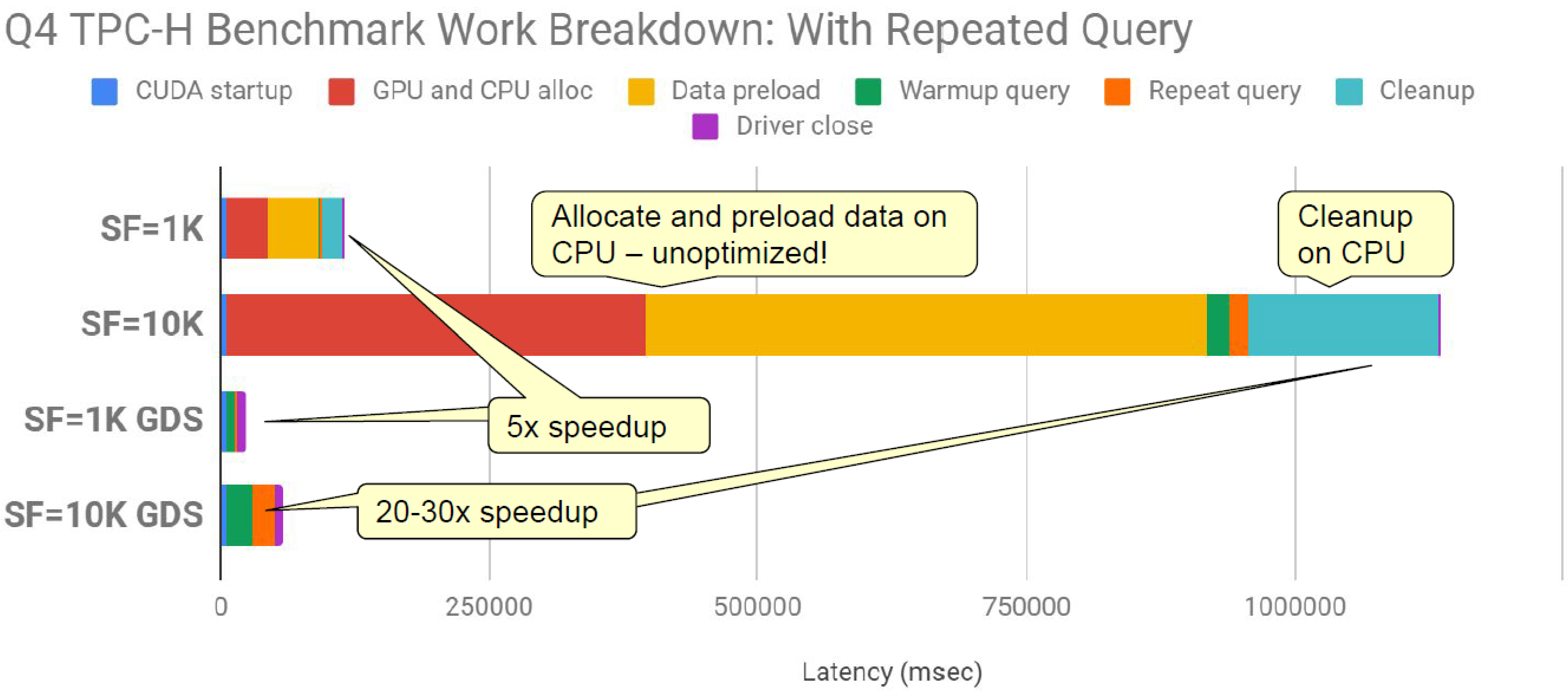




6 or 10 DGX-2,
10 DDN EXA5 on A3I AI400X
GPUDIrect Storage used for IOR easy

TPC-H (REAL BUT EXTREME CASE)

Speedups from both IO and savings in CPU memory management

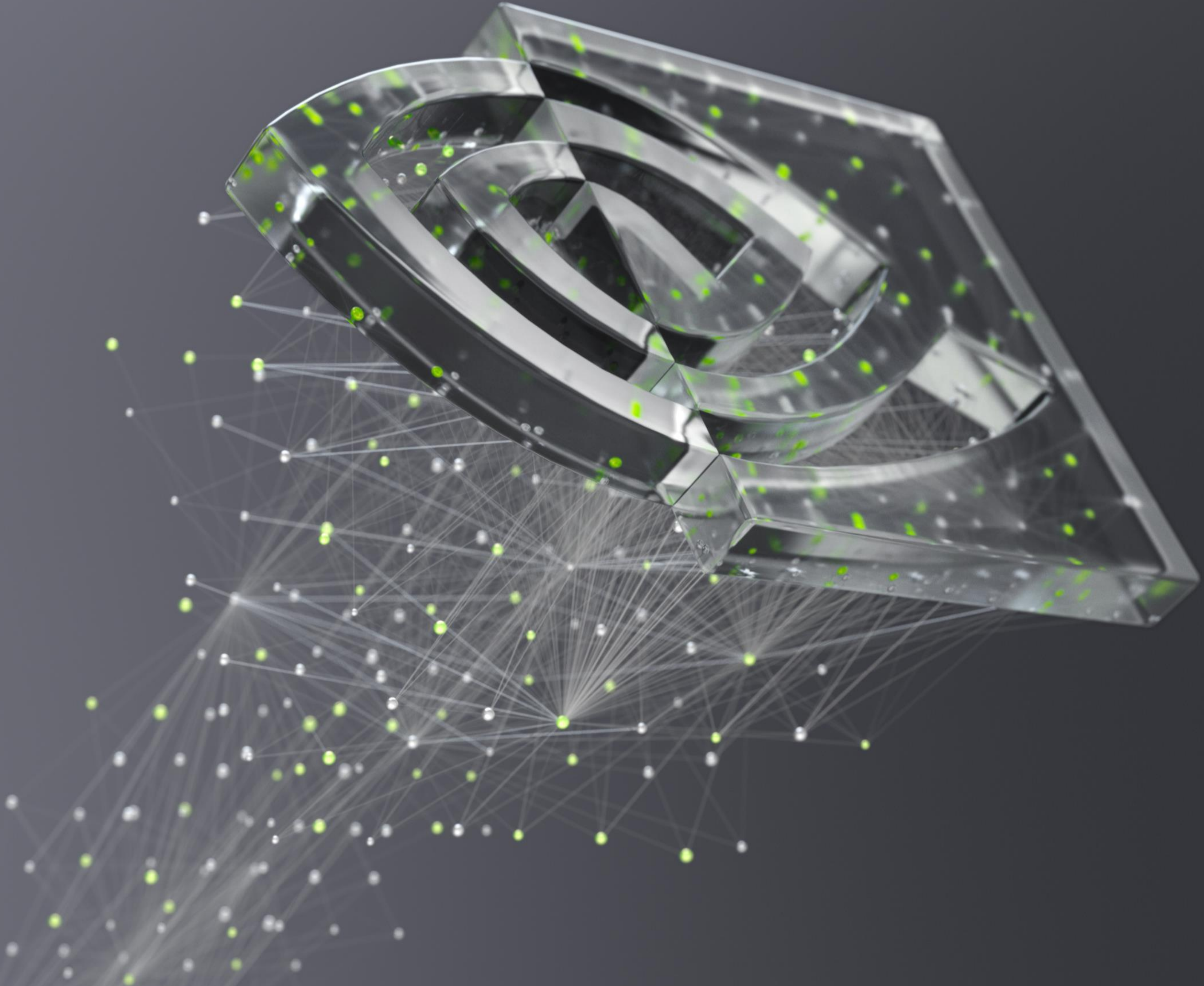


Allocate and preload data on CPU – unoptimized!

Cleanup on CPU

5x speedup

20-30x speedup



NVIDIA