GPUDirect Storage: A Direct Path Between Storage and GPU Memory | NVIDIA Developer Blog

https://developer.nvidia.com/blog/gpudirect-storage/

Adam Thompson Wed Jun, 09 03:30

Keeping GPUs Busy

As AI and HPC datasets continue to increase in size, the time spent loading data for a given application begins to place a strain on the total application's performance. When considering end-to-end application performance, fast GPUs are increasingly starved by slow I/O.

I/O, the process of loading data from storage to GPUs for processing, has historically been controlled by the CPU. As computation shifts from slower CPUs to faster GPUs, I/O becomes more of a bottleneck to overall application performance.

Just as GPUDirect RDMA (Remote Direct Memory Address) improved bandwidth and latency when moving data directly between a network interface card (NIC) and GPU memory, a new technology called GPUDirect Storage enables a direct data path between local or remote storage, like NVMe or NVMe over Fabric (NVMe-oF), and GPU memory. Both GPUDirect RDMA and GPUDirect Storage avoid extra copies through a bounce buffer in the CPU's memory and enable a direct memory access (DMA) engine near the NIC or storage to move data on a direct path into or out of GPU memory – all without burdening the CPU or GPU. This is illustrated in Figure 1. For GPUDirect Storage, storage location doesn't matter; it could be inside an enclosure, within the rack, or connected over the network. Whereas the bandwidth from CPU system memory (SysMem) to GPUs in an NVIDIA DGX-2 is limited to 50 GB/s, the bandwidth from SysMem, from many local drives and from many NICs can be combined to achieve an upper bandwidth limit of nearly 200 GB/s in a DGX-2.

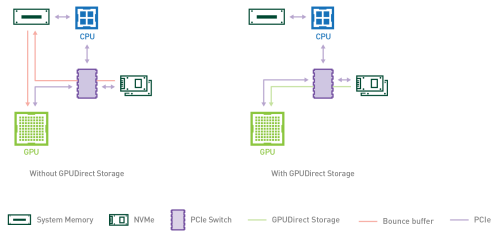

Figure 1: The standard path between GPU memory and NVMe drives uses a bounce buffer in system memory that hangs off of the CPU. The direct data path from storage gets higher bandwidth by skipping the CPU altogether.

In this blog, we expand on a previous post demonstrating GPUDirect Storage: a proof of concept enabling direct memory access (DMA) from storage that is either local to a given server or outside of the enclosure via NVMe-oF. We demonstrate that direct memory access from storage to GPU relieves the CPU I/O bottleneck and enables increased I/O bandwidth and capacity. Further, we provide initial performance metrics presented at GTC19 in San Jose, based on the RAPIDS project's GPU-accelerated CSV reader housed within the cuDF library. Lastly, we will provide suggestions on key applications that can make use of faster and increased bandwidth, lower latency, and increased capacity between storage and GPUs.

In future postings, as this feature approaches productization, we'll describe how to program it. A new set of cuFile APIs will be added to CUDA to support this feature along with native integration into RAPIDS' cuDF library.

How does Direct Memory Access Work, Anyway?

The PCI Express (PCIe) interface connects high-speed peripherals such as networking cards, RAID/NVMe storage, and GPUs to CPUs. PCIe Gen3, the system interface for Volta GPUs, delivers an aggregated maximum bandwidth of 16 GB/s. Once the protocol inefficiencies of headers and other overheads are factored out, the maximum achievable data rate is over 14 GB/s.

Direct memory access (DMA) uses a copy engine to asynchronously move large blocks of data over PCIe rather than loads and stores. It offloads computing elements, leaving them free for other work. There are DMA engines in GPUs and storage-related devices like NVMe drivers and storage controllers but generally not in CPUs. In some cases, the DMA engine cannot be programmed for a given destination; for example, GPU DMA engines cannot target storage. Storage DMA engines cannot target GPU memory through the file system without GPUDirect Storage.

DMA engines, however, need to be programmed by a driver on the CPU. When the CPU programs the GPU's DMA, the commands from the CPU to GPU can interfere with other commands to the GPU. If a DMA engine in an NVMe drive or elsewhere near storage can be used to move data instead of the GPU's DMA engine, then there's no interference in the path between the CPU and GPU. Our use of DMA engines on local NVMe drives vs. the GPU's DMA engines increased I/O bandwidth to 13.3 GB/s, which yielded around a 10% performance improvement relative to the CPU to GPU memory transfer rate of 12.0 GB/s shown in Table 1 below.

Relieving I/O Bottlenecks and Relevant Applications

As researchers apply data analytics, AI, and other GPU-accelerated applications to increasingly-large datasets, some of which will not fit entirely in CPU memory or even local storage, it will become increasingly important to relieve the I/O bottleneck on the data path between storage and GPU memory. Data analytics applications operate on large quantities of data that tends to be streamed in from storage. In many cases the ratio of computation to communication, perhaps expressed in flops per byte, is very low, making them IO bound. In order for deep learning to successfully train a neural network, for example, many sets of files are accessed per day, each on the order of 10MB and read multiple times. Optimization of data transfer to GPU, in this case, could have a significant and beneficial impact on the total time to train an AI model. In addition to data ingest optimizations, deep learning training often involves the process of checkpointing – where trained network weights are saved to disk at various stages of the model training process. By definition, checkpointing is in the critical I/O path and reduction of the associated overhead can yield shorter checkpointing periods and faster model recovery.

In addition to data analytics and deep learning, graph analytics, the study of network interactions, have high I/O demands. When traversing a graph to find influential nodes or the shortest path from here to there, compute is a small fraction of the total time to solution. From a current node, identification of where to travel next can involve I/O queries in the range of one to hundreds of files originating from a petabyte-sized data lake. While local caching assists in keeping track of directly actionable data, graph traversal is both universally latency and bandwidth sensitive. As NVIDIA expands GPU acceleration of graph analytics via the RAPIDS cuGraph library, eliminating the file I/O overhead will be critical in continuing to provide speed-of-light solutions.

Scaled Storage and Bandwidth Options to GPUs

One common theme between data analytics and AI is that the datasets used to derive insights are often massive. The NVIDIA DGX-2, consisting of 16 Tesla V100s contains a stock configuration of 30TB of NVMe SSD memory (8x 3.84TB) and 1.5TB of system memory. Enablement of DMA operations from drives allows for fast memory access with increased bandwidth, lower latency, and potentially unlimited storage capacity.

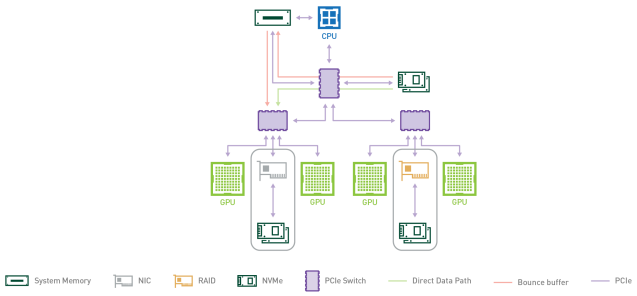

Figure 2: More bandwidth and more storage from outside the enclosure. The NIC sitting off the PCI switch enables NVMe-oF remote storage scaling while the RAID card enables nearby storage. The RAID card shown is a prototype only and is not indicative of current or future DGX-2 product offerings.

The DGX-2 enclosure contains two CPUs, and each CPU has two instances of the PCIe subtree shown in Figure 2. The multiple PCIe paths from storage or SysMem supported by two levels of PCIe switches into GPUs make the DGX-2 a good test vehicle for prototyping GPUDirect Storage. The left column in Table 1 lists various origins of data transfers to the GPU, the second column lists the measured bandwidth from that source, the third column identifies the number of such paths, and the final column is the product of the middle two columns showing the total bandwidth available from that kind of source. There's one path from the CPU's system memory (SysMem) for each of 4 PCIe trees, at 12-12.5 GB/s and another one from ¼ of the drives that hang off of each PCIe tree, at 13.3 GB/s. DGX-2s have one PCIe slot per pair of GPUs. That slot can be occupied either by a NIC which has been measured at 10.5 GB/s or, in the case of our prototype used for this blog, a RAID card, which has been measured at 14 GB/s. NVMe-oF (over fabric) is a common protocol that uses a NIC to reach remote storage, e.g. over an Infiniband network. The right hand column of PCIe bandwidth added across all sources can be summed to 215 GB/s if RAID cards are used in the 8 PCIe slots (2 per PCIe subtree in Figure 2); the sum would be lower if NICs are used in those slots instead.

Origin to GPU	Bandwidth per PCIe Tree, GB/s	PCIe paths per DGX-2	Total Bandwidth per DGX-2, GB/s per direction
Inside Enclosure – SysMem	12.0-12.5	4	48.0-50.0
Inside Enclosure – NVMe	13.3	4	53.3
Outside Enclosure – NIC	10.5	8	84.0
Outside Enclosure – RAID	14.0	8	112.0

Max: SysMem+NVMe+RAID			215
GPUs	>14.4	16	>230

Table 1: DGX-2 Bandwidth Options to GPUs. 4 PCIe subtrees inside the enclosure and 8 NICs or RAID cards.

One of the major benefits of GPUDirect storage is that fast data access, whether resident inside or outside of the enclosure, on system memory or NVMe drives, is additive across the various sources. Use of internal NVMe and system memory does not exclude the use of NVMe-oF or RAID storage. Finally, these bandwidths are bidirectional, enabling complex choreographies where data may be brought in from distributed storage, cached in local disks, and there may be collaboration with the CPU via data structures that are shared in CPU system memory for a total bandwidth at over 90% of GPU's peak IO. Reads and writes to each of these three sources may occur concurrently. Figure 3 color codes the various sources and shows additive combinations as stacked bars. In the column labels below, the number of source instances are in parentheses, e.g. 16 NVMe drives or 8 NICs doing NVMe-oF. The rough capacity available with each option is shown last in the column labels.

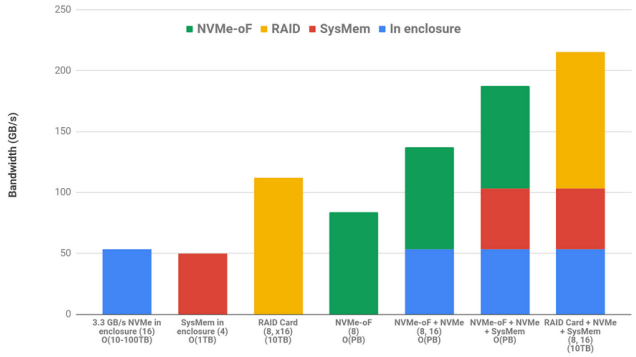

Figure 3: Bandwidth limits from various sources are additive

GPU Accelerated CSV Reader Case Study

The RAPIDS open-source software supported by NVIDIA focuses on end-to-end GPU-accelerated data science. One library, cuDF, provides a Pandas-like experience and allows users to load, filter, join, sort, and explore datasets on the GPU. NVIDIA engineers were able to leverage GPUDirect Storage to GPU to provide a 8.8x throughput speedup over the naive cuDF CSV reader and 1.5x speedup over the current, best-effort implementation that the cuDF library has been updated to use. These improvements are illustrated in Figure 4.

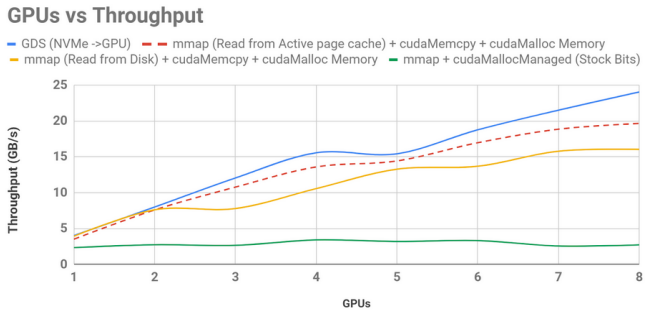

Figure 4: The original (0.7) cuDF csv_reader implementation, shown in green at the bottom, didn't scale with GPU concurrency because it incurred faulting from SysMem to the GPU, faulting from storage into SysMem, and unpinned data movement through a CPU bounce buffer. The revised bounce buffer implementation now shipping with RAPIDS uses best-available memory management and explicit data movement is shown in yellow. Reading data from a warmed-up page cache is shown in dotted red, and GPUDirect Storage, in blue, beats all of these, limited only by NVMe drive speeds. These measurements happened to be made with only 8 GPUs and 8 NVMe drives.

Further, a direct data path decreased end-to-end latency by 3.8x on 80 GB of data. In another cuDF CSV reader study on 16 GPUs, shown in Figure 5, read bandwidth with smoother, more predictable, and had lower latency with the direct, non-faulting data path in blue than either the improved, direct cuDF behavior that still uses a bounce buffer in red, or the original behavior in yellow.

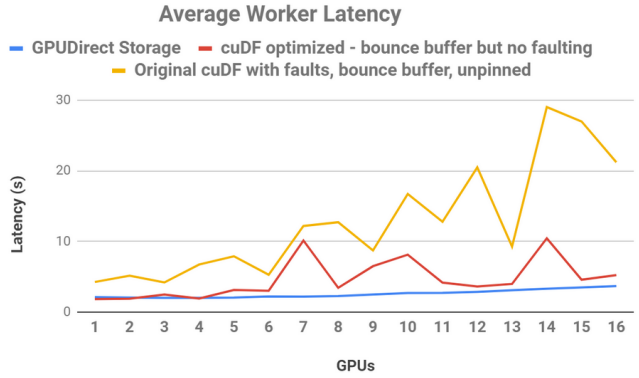

Figure 5: Latency comparison for cuDF read_csv. When the CPU bounce buffer was used in the original cuDF version with faulting (yellow), the latency as a function of GPUs was choppy and jittery. cuDF has since been optimized to eliminate faulting with direct transfers (red), leading to improved performance and stability. GPUDirect Storage (blue), provides smooth and predictable latency as processing scales to additional GPUs.

Bandwidth and CPU Load Study

Figure 6 highlights the relative bandwidth achievable with different transfer methods. Data can be transferred from storage to CPU memory using buffered I/O and retained using the file system's page cache (yellow line). Using the page cache has some overheads, such as an extra copy within CPU memory, but it's a win relative to DMA'ing data from storage and using a bounce buffer (red line)

until the transfer sizes become large enough to amortize the DMA programming. Because the bandwidth between storage and the GPU using GPUDirect Storage (blue line) is much higher than between the CPU and GPU, it wins at any transfer size.

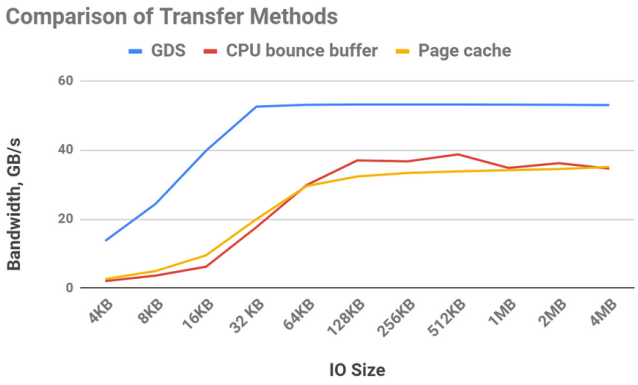

Figure 6: GPUDirect Storage (GDS) has significantly better bandwidth than either using a bounce buffer (CPU_GPU) or than enabling the file system's page cache with buffered IO. 16 NVMe drives were used with 16 GPUs.

Getting higher bandwidth is one thing, but some applications are sensitive to CPU load. If we examine the bandwidth for these three approaches when divided by CPU utilization, the results are much more dramatic, as shown in Figure 7.

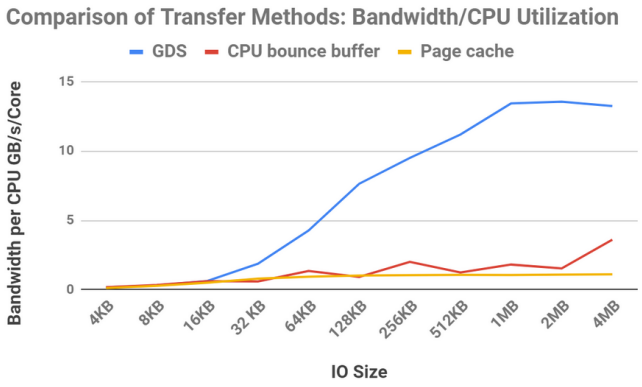

Figure 7: Bandwidth divided by the fractional utilization of the CPU core. GPUDirect Storage incurs far less burden on the CPU at larger transfer sizes.

TPC-H Case Study

TPC-H is a decision support benchmark. There are many queries for this benchmark, and we've focused on query four (Q4) which streams in large amounts of data and does some processing on the GPU of that data. The size of the data is determined by the scale factor (SF). A scale factor of 1K implies a data set size of nearly 1TB (82.4GB of binary data); 10K implies 10 times that, which

cannot entirely fit into CPU memory. In the non-GPUDirect Storage case, space in CPU memory must be allocated, filled from disk, and later freed, all of which take time that ends up being irrelevant if the data can be directly transferred to GPU memory on demand as it is consumed. Figure 8 shows GPUDirect Storage with large performance speedups over not using GPUDirect Storage: 6.7x for SF 1K and 32.8x for SF 10K.

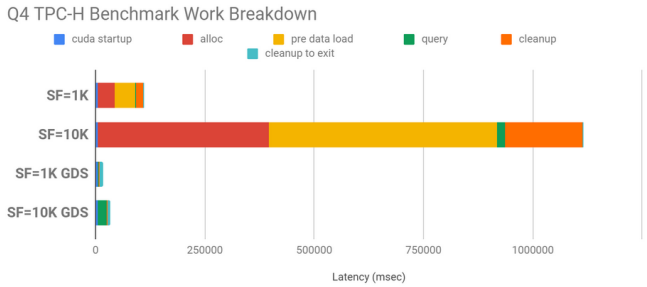

Figure 8: TPC-H Query 4, with and without GPUDirect Storage (GDS). Scale factors (SF) of 1K (~1TB) and 10K (~10TB) were used, with speedups of 4.9x and 19.6x respectively. Repeated allocation in CPU memory, loading of data into memory, and freeing of memory on the CPU side are big bottlenecks which are avoided in the GPUDirect Storage case.

Data Manipulation Case Study

A direct path from storage to GPU is also applicable to datasets that don't entirely fit into the GPU's framebuffer. In one experiment, NVIDIA used a 1TB input dataset and the 512GB of aggregate GPU memory of the DGX-2 to prove with GPUDirect Storage that even in the event of memory oversubscription, data I/O is faster to the 16 GPUs than to host memory. Direct reads and writes to data yields a 2x speed improvement, but chunking, using smaller batches, and other optimizations increase speedup further. In total, GPUDirect Storage improved data manipulation speed by 4.3x.

The Value of GPUDirect Storage

The key functionality provided by GPUDirect Storage is that it enables DMA from storage to GPU memory through this file system. It provides value in several ways:

- 2x-8x higher bandwidth with data transfers directly between storage and GPU.
- Explicit data transfers that don't fault and don't go through a bounce buffer are also lower latency; we demonstrated examples with 3.8x lower end-to-end latency.
- Avoiding faulting with explicit and direct transfers enables latency to remain stable and flat as GPU concurrency increases.
- Use of DMA engines near storage is less invasive to CPU load and does not interfere with GPU load. The ratio of bandwidth to fractional CPU utilization is much higher with GPUDirect

Storage at larger sizes. We observed (but did not graphically show in this blog) that GPU utilization remains near zero when other DMA engines push or pull data into GPU memory.
- The GPU becomes not only the highest-bandwidth computing engine but also the computing element with the highest IO bandwidth, e.g. 215 GB/s vs. the CPU's 50 GB/s.
- All of these benefits are achievable regardless of where the data is stored – enabling very fast access to petabytes of remote storage faster than even the page cache in CPU memory.
- Bandwidth into GPU memory from CPU memory, local storage, and remote storage can be additively combined to nearly saturate the bandwidth into and out of the GPUs. This becomes increasingly important and data from large, distributed data sets is cached in local storage, and working tables may be cached in CPU system memory and used in collaboration with the CPU.

In addition to the benefits of speeding up computation with GPUs instead of CPUs, GPUDirect Storage acts as a force multiplier once whole data processing pipelines shift to GPU execution. This becomes especially important as dataset sizes no longer fit into system memory, and data I/O to the GPUs grows to be the defining bottleneck in processing time. Enabling a direct path can reduce, if not totally alleviate, this bottleneck as AI and data science continues to redefine the art of the possible.

Acknowledgements

We are deeply indebted to the outstanding contributions of the GPUDirect Storage team to the architecture, design, evaluation, and tuning of this work that provided the basis of this content, including Kiran Modukuri, Akilesh Kailash, Saptarshi Sen, and Sandeep Joshi. Special thanks are due to Nikolay Sakharnykh and Rui Lan for their assistance with TPC-H.

We'd like to thank the following fellow travelers for help with prototyping and initial data collection for GPUDirect Storage:

- We used Micron 9200 MTFDHAL3T8TCT 3.84TB and Samsung MZ1LW960HMJP 960GB NVMe drives for local storage.
- There may be a storage controller used for storage inside or outside of the enclosure to perform RAID 0, RAID 5, and to localize controls. We used SmartROC 3200PE RAID controllers from MicroChip, anHPE Smart Array P408i-a SR Gen10 12G SAS modular controller and an HPE Smart Array P408i-p SR Gen10 12G SAS PCIe plug-in controller.
- Storage may be enabled over fabric. We used Mellanox MCX455A-ECAT ConnectX4 NICs and E8 Storage solutions with 4 Mellanox ConnectX5s.

We also acknowledge related work from Mark Silberstein's SPIN project.

Interested in Learning More?

If you'd like to be kept apprised of updates on GPUDirect Storage, please sign-up for the interest list (you must be a member of the NVIDIA Registered Developer Program to access).