GPU SMITH
COMPUTE ENGINEERING

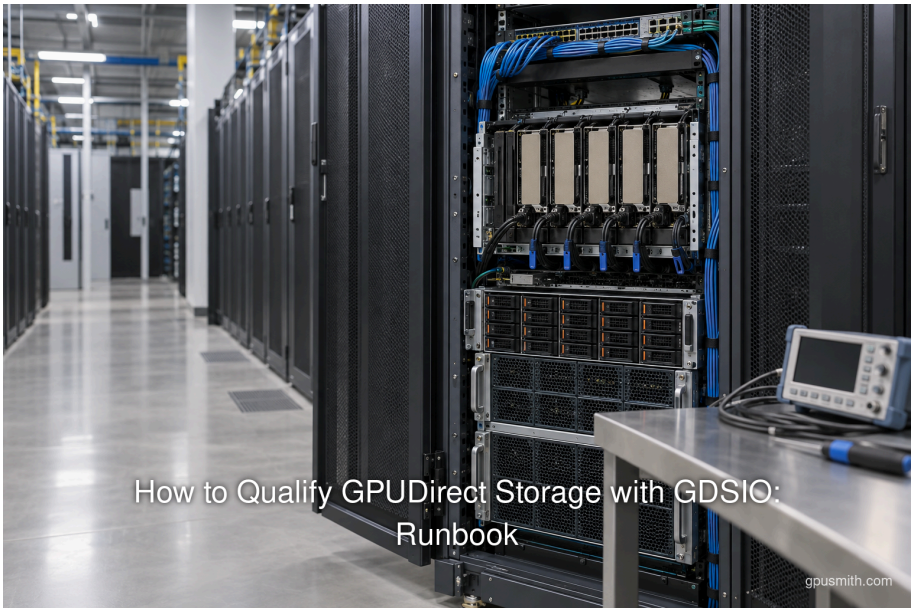
GPU SMITH / RESEARCH & PRACTICAL GUIDANCE

How to Qualify GPUDirect Storage with GDSIO: Runbook

> Plan and build private AI compute with GPU Smith. We connect workload requirements with GPU hardware, networking, deployment and operating decisions so your infrastructure fits the work it must perform.

Published by GPU Smith · 2026-09-23 · gpudirect storage · gdsio · gdscheck · nvidia gds · storage benchmarking · nvme · rdma · filesystem qualification · gpu infrastructure

Original article: https://gpusmith.com/articles/en/gpudirect-storage-gdsio-qualification-runbook



In this article

Executive Summary

To qualify **NVIDIA GPUDirect Storage (GDS)**, an operator must establish three separate facts: that the installed software and the exact file path are eligible, that functional graphics processing unit (GPU) reads and writes succeed, and that the measured run uses the intended data path. GDS is a direct memory access (DMA) path between storage and GPU memory that can avoid a CPU bounce buffer. (docs.nvidia.com) (github.com) An installed `libcufile` or a successful `gdsio` process is insufficient evidence by itself, because a compatible POSIX path can still complete I/O through system memory. (docs.nvidia.com) (github.com) A qualification report should identify the path as **P2PDMA**, **NVFS**, or **compatibility**, using the modes exposed by the installed GDS release, then attach counters or diagnostic logs for the run. NVIDIA added explicit `p2pdma`, `nvfs`, and `compat` reporting to `gdscheck` in GDS 1.16. (docs.nvidia.com) (docs.weka.io)

The practical sequence is: inventory GPU, kernel, CUDA, storage client and mount; map the GPU to the nonvolatile memory express (NVMe) controller or remote direct memory access (RDMA) network interface card (NIC); run `gdscheck`; perform read and write checks on a disposable test file; compare matched `gdsio` workloads; and preserve the logs. For local block storage, NVIDIA currently documents **EXT4 and XFS** as the supported filesystems. (docs.nvidia.com) (man7.org) For a distributed filesystem, the filesystem vendor's version and mount requirements are an additional gate: BeeGFS calls for RDMA across its nodes, WEKA can revert to standard I/O when requirements are unmet, and VAST calls for an RDMA-enabled mount. (doc.beegfs.io) (docs.weka.io) (kb.vastdata.com) The matrix below makes those checks path-specific rather than treating a cluster as globally enabled.

Benchmark numbers are evidence about the tested workload, not portable promises. Keep block size, file set, read/write mix, GPU, placement, threads and test duration fixed between paths. `gdsio` uses `O_DIRECT` file descriptors by default; Linux alignment rules still vary by filesystem and kernel, and a network filesystem may cache data at the server even when client page cache is bypassed. (docs.nvidia.com) (github.com) (man7.org) (man7.org) Record throughput together with process CPU utilization, errors, POSIX/bounce counters and the mode declaration. NVIDIA documents POSIX pool counters as system-memory bounce-buffer statistics for

compatibility mode. (docs.nvidia.com) (docs.weka.io) A path passes production qualification only when the observed mode and integrity results meet the operator's written acceptance criteria at the required workload read rate; this report does not assign a universal throughput threshold.

Introduction and Background

A GPU pipeline can be stalled by data delivery even when the GPU itself has adequate arithmetic capacity. The useful question is narrower than whether a node has GDS installed: **can this particular GPU read this particular mounted path through the intended data path, at the needed rate, without an unobserved fallback?** This report is a runbook for storage, platform and machine-learning infrastructure engineers evaluating local NVMe and distributed filesystems. It treats eligibility, functional correctness and performance as separate gates. GDS moves data between storage and GPU memory through DMA and is designed to avoid the extra CPU-memory staging of a conventional path.

The site publishing this guide, GPU Smith, describes its private AI infrastructure engagements as governed by written acceptance criteria and an as-built documentation set. (gpusmith.com) That advisor perspective belongs in the acceptance method: a storage path should be signed off with a reproducible record, not with a vendor slide or an isolated peak number. Neither a GPU model name nor a filesystem brand establishes the mode used by an individual file. NVIDIA's current tool reporting distinguishes `p2pdma`, `nvfs`, and `compat`; that distinction must be recorded for the installed release and target path.

The runbook dates version-sensitive claims to **September 23, 2026**. NVIDIA's docs span several GDS and CUDA generations, and the installed package's help output is the final authority for executable name and switches. Filesystem claims in this article are scoped to the vendor documentation version named in each row. The controlled test should use a scratch mount and a capacity limit agreed with storage operators, because directory-based `gdsio` writes create files before a read test can consume them. (docs.nvidia.com) The method applies to a single test node or a larger cluster rollout. Subsequent sections explain what to collect, what to run, and how to interpret a result without turning a single benchmark into a procurement forecast.

GDS Paths and the Qualification Decision

Decide whether the workload needs a direct path

Start with the pipeline's required **sustained read rate**, not an interface's advertised peak. In a **(Hypothetical Example)**, a training job that consumes eight input streams at 0.5 GB/s each requires 4 GB/s of delivered reads by simple multiplication; the stream rate is a planning input, not a measured GDS result. Add the job's burst pattern, data transforms, checkpoint writes and acceptable CPU headroom. If the ordinary path already exceeds demand and CPU use is acceptable, GDS may still be useful for scaling, but the immediate production decision can be based on the measured margin. If CPU staging is the suspected constraint, compare both throughput and CPU time under the same data set. `pidstat -u` provides per-task CPU utilization, whereas fio's grouped CPU values are averages across jobs. (man7.org) (github.com)

The data path has three operational categories. **Direct GDS** uses a supported PCI peer-to-peer DMA path or NVIDIA's filesystem integration. **Compatibility mode** permits cuFile calls to complete through POSIX I/O and CPU memory staging. **Ordinary application I/O** uses the application's normal POSIX path and its explicit transfer to GPU memory. The latter is a useful comparison, but it is a different software path. A change in throughput between these categories cannot alone prove which layer caused it. NVIDIA notes that its GDS statistics do not separately distinguish PCI P2PDMA from `nvidia-fs`; pair them with `gdscheck`, topology and device records. (docs.nvidia.com)

Local NVMe is a PCIe device, while networked NVMe can use RDMA or TCP transports; the name NVMe alone does not specify the GPU path. The NVM Express specification lists PCIe, RDMA and TCP transports separately. (nvmexpress.org) CUDA 12.8 introduced an NVMe P2PDMA configuration that can operate without `nvidia-fs.ko` when its other requirements are met. (docs.nvidia.com) (nvmexpress.org) Consequently, an unloaded `nvidia-fs` module is **not** a universal failure signal on current local NVMe nodes. For a distributed filesystem, the client's GDS integration, mount protocol, RDMA configuration and storage-side version must all be checked on the actual path. BeeGFS, WEKA and VAST each publish distinct requirements. (doc.beegfs.io) (docs.weka.io) (kb.vastdata.com)

Classification before timing

Before any performance run, write down the intended mode and the evidence that could disprove it:

- **Direct path candidate:** supported GPU and software, supported filesystem and mount, suitable topology, direct-I/O capable target file, and `gdscheck` showing the relevant capability. (docs.nvidia.com) (docs.nvidia.com) (man7.org)
- **Compatibility candidate:** cuFile calls work, but the I/O is served through a POSIX path or a vendor-documented standard-I/O fallback. (docs.weka.io)
- **Unqualified path:** any prerequisite is unknown, functional verification fails, the result changes without a known cause, or the expected mode cannot be corroborated.

This classification deliberately keeps “functional” apart from “direct.” NVIDIA documents managed CUDA memory as an indirect, unregistered-buffer case rather than a normal registered GDS buffer; a successful application read using managed memory therefore does not establish the same path as a registered device buffer. (docs.nvidia.com) Capture the memory allocation type used by the production application, and use the same type in a representative functional test.

Support Matrix and Topology Gate

Version, filesystem and mount prerequisites

Table 1 is a **September 23, 2026** evidence matrix, not a universal compatibility guarantee. The installation's CUDA, driver, GDS and filesystem-client versions still need comparison with current vendor release notes. NVIDIA documents EXT4 and XFS for local block device filesystems and gives platform-specific minimum kernels for NVMe P2PDMA in its release notes. (docs.nvidia.com) (man7.org)

Candidate path	Documented gate	Local evidence to capture	Decision if missing
Local NVMe on EXT4	NVIDIA documents EXT4 among supported local block filesystems; inspect the actual mount, direct-I/O behavior and P2PDMA or NVFS mode.	`findmnt` target, controller PCI address, `gdscheck`, file alignment, GPU ID.	Hold direct-path sign-off until the exact target file passes.
Local NVMe on XFS	NVIDIA documents XFS among supported local block filesystems.	Same evidence, plus XFS and kernel versions.	Treat a passing cuFile call without path evidence as functional only.
BeeGFS 8.4 documentation	RDMA is required on BeeGFS nodes, and its guide calls for RDMA connections to storage and metadata. (doc.beegfs.io) (doc.beegfs.io)	Client/server versions, `beegfs-net`, mount, 4 KB alignment check. (doc.beegfs.io)	Qualify the mounted client path only after RDMA and alignment checks.
WEKA 5.0 documentation	RDMA and GDS must both be enabled; if requirements are unmet, WEKA says it reverts to standard I/O. (docs.weka.io) (docs.weka.io)	Client and server versions, RDMA interface, mount options, encryption state. (docs.weka.io)	Record the fallback state rather than calling the path direct.
VAST NFS documentation tagged 5.3	VAST says GDS needs an RDMA-capable network and RDMA-enabled mount. (kb.vastdata.com)	Mounted protocol/options and RDMA NIC, then test file and mode.	Do not infer GDS from NFS reachability alone.
Other advertised integrations	DDN has publicly described EXAScaler GDS support; Dell describes a PowerScale integration using NFS over RDMA. (www.ddn.com) (www.dell.com)	Obtain the current vendor version and deployment-specific support statement.	Keep the row pending until the exact version is documented.

The matrix distinguishes a vendor's feature statement from an operator's result. In particular, a distributed filesystem's RDMA fabric may be healthy while the GPU data path falls back. WEKA explicitly describes such a reversion, and BeeGFS supplies `beegfs-net` to verify its storage and metadata connections. (docs.weka.io) (doc.beegfs.io) Keep separate rows for every mount, because client configuration and storage target can differ even within one cluster.

Topology and Evidence Gate

Draw **GPU to PCIe switch to root complex to NVMe controller or RDMA NIC** before selecting a test GPU. NVIDIA lists root complex, switches and physical device placement as factors in delivered GDS performance. (docs.nvidia.com) (man7.org) The upstream `lspci -t` view shows buses, bridges and their connections; `nvidia-smi topo -m` is called out by BeeGFS for GPU-to-RDMA-NIC topology. (man7.org) (doc.beegfs.io) Capture PCI bus addresses rather than relying only on friendly GPU ordinals, which can change across systems. A PCI device's sysfs `numa_node` file reports its associated nonuniform memory access (NUMA) node or `-1` if unknown. (docs.kernel.org) `numactl --hardware` shows the system NUMA layout, and remote memory access can carry a performance penalty. (docs.redhat.com) (docs.redhat.com)

For each candidate, record the GPU UUID and PCI address, storage device or mount source, NIC address if remote, shared switch/root complex and NUMA node. `findmnt --target` resolves the filesystem containing the actual test path, avoiding an assumption based on a parent directory. (man7.org) Do not change BIOS

PCIe settings or unload modules merely to improve a score during qualification; record the as-found configuration, make changes through normal change control, and repeat the identical matrix afterward. The goal is to measure the deployable state.

Use this **support and topology checklist** before approving a benchmark slot:

- **GPU identity:** save UUID, model, driver and PCI address for each tested GPU.
- **Mode capability:** save the complete `gdscheck` output for the installed release.
- **Kernel and CUDA:** record exact versions, architecture and the GDS package revision.
- **Mount identity:** resolve the test file with `findmnt --target`; record source, type and options. (man7.org)
- **Block device route:** save controller PCI address and the `lspci -t` tree. (man7.org)
- **Remote route:** save GPU-to-NIC topology and NUMA placement. (doc.beegfs.io)
- **NUMA record:** retain PCI `numa_node` values, including `-1` when the node is unknown. (docs.kernel.org)
- **RDMA state:** for BeeGFS, verify storage and metadata connections, not just link status. (doc.beegfs.io)
- **Mount protocol:** for VAST, confirm that the tested NFS mount is RDMA-enabled. (kb.vastdata.com)
- **Feature exclusions:** for WEKA, record whether filesystem encryption is enabled. (docs.weka.io)
- **File alignment:** inspect direct-I/O alignment where `statx` supports it. (man7.org)
- **Scratch controls:** record available space, file ownership and a cleanup plan.

Functional Verification and GDSIO Protocol

Preflight and an integrity matrix

On each node, collect package versions, kernel, GPU identity, mounted source and options, `gdscheck` output and the installed `gdsio` help screen. NVIDIA recommends `gdscheck` as an installation verification step; current releases can expose supported modes and topology information. Interpret its report alongside the **specific file** being tested. For local NVMe, note that NVIDIA's current documentation allows P2PDMA without `nvidia-fs.ko` from CUDA 12.8 in qualifying configurations, while CUDA Toolkit 13.4 changes how `nvidia-fs` is packaged on non-DGX platforms. (docs.nvidia.com)

Use a scratch directory with known free capacity and permissions. `gdsio -D` writes one file per worker before directory reads, so a read-only invocation against an empty directory is not a valid integrity test. Choose a block size that meets the mount's direct-I/O alignment; Linux says `O_DIRECT` restrictions vary by filesystem and kernel, and misalignment may fail with `EINVAL` or be buffered. (man7.org) (man7.org) On Linux 6.1 and newer, `statx` can report file-specific direct-I/O alignment where the filesystem implements it. (man7.org) Preserve the exact alignment result if available.

Table 2 defines a functional matrix to run on **every candidate mount and selected GPU**, using the installed tool's documented switches. Its purpose is to isolate failure, not to rank throughput.

Test	Memory and I/O path	Expected evidence	Disposition
Sequential write	Device buffer and cuFile, `gdsio -I 1`	File appears, bytes and errors recorded; use `-V` where supported. (docs.nvidia.com) (github.com) (docs.nvidia.com)	A write error blocks qualification.

Test	Memory and I/O path	Expected evidence	Disposition
Sequential read	Same file, device buffer and cuFile, `gdsio -I 0`	Read succeeds with verification against data written with `-V`.	A mismatch blocks qualification.
Random write/read	Paired `-I 3` and `-I 2`, same seed and geometry	Errors, verification and mode recorded. (docs.nvidia.com)	Treat a different geometry as a separate test.
CPU-only comparison	Same file and workload through an ordinary I/O path	Output, CPU use and cache state recorded. (github.com) (man7.org)	Comparator only; it does not prove GDS mode.
Forced compatible comparison	Same cuFile workload with documented compatibility setting	POSIX/bounce counters or log state and CPU use. (docs.nvidia.com) (docs.weka.io)	Label it compatibility, never direct.
Application smoke test	Production allocation type and file API	Integrity, errors, GPU and mount identity.	Synthetic pass does not replace application test.

The paired read/write tests matter because a throughput result without verified data can mask a path or workload mistake. NVIDIA says `gdsio -V` read verification should use files written with the verification option. If an application uses managed memory, report that explicitly and avoid equating its result with a registered device-buffer test. Leave the test files intact until checksums, logs and counters are archived, then remove only the scratch data created for qualification.

Controlled GDSIO Baselines

The installed `gdsio` help output should be saved in the evidence bundle. NVIDIA's guide defines `-I 0/1/2/3` as sequential read/write and random read/write, and documents `-w`, `-s`, `-i` and `-T` for workers, size, I/O size and duration. (docs.nvidia.com) Record the selected GPU, transfer mode, file or directory target, block size, workers, file size and duration for every run. Do not claim a queue depth that the tool did not expose or measure. If using fio for a supplementary test, inspect achieved I/O-depth distribution rather than trusting the requested depth; fio's own documentation warns that the two can differ. (github.com)

The following is a **(Hypothetical Example)** command sequence, with an existing scratch directory and GPU index 0. Confirm every switch against the installed `--help` before execution, choose a file size within the scratch allocation, and save the outputs. NVIDIA's documented `-x` modes distinguish storage-to-GPU, storage-to-CPU, and storage-to-CPU-to-GPU transfers; `-d` chooses the GPU index, while `-D` creates a file per worker on the write pass.

```
gdscheck.py -p
gdsio -D /mnt/qual-scratch -d 0 -x 0 -I 1 -w 4 -s 8G -i 1M -T 120 -V
gdscheck.py -f /mnt/qual-scratch/testfile
gdsio -D /mnt/qual-scratch -d 0 -x 0 -I 0 -w 4 -s 8G -i 1M -T 120 -V
gdsio -D /mnt/qual-scratch -d 0 -x 2 -I 0 -w 4 -s 8G -i 1M -T 120
```

The file check needs an existing target file. The `-D` write creates one file per worker, so replace `testfile` above with one of the generated filenames shown in the write output. The final `-x 2` comparison represents CPU staging and GPU transfer; it is not a forced cuFile compatibility-mode test. Keep a separate compatibility run only when the installed configuration offers a documented, reversible way to force that mode. `gdsio` exposes worker threads, but the published help does not specify a dedicated queue-depth switch; record worker count and any observed concurrency instead. (github.com)

A minimal sequence is to create data with a verified write, run a verified read on the same geometry, then rerun matched direct, compatible and CPU comparisons. Use **three independent repeats** as a local protocol choice, not as an NVIDIA standard, and preserve each result rather than only the best. Stabilize concurrent cluster load, storage-side cache state and placement as far as operations permit. fio's `ramp_time` illustrates the distinction between a warm-up interval and a measured interval; if used, record both. (github.com) `gdsio` normally opens files with `O_DIRECT`, which helps avoid a client page-cache benchmark, but `O_DIRECT` on NFS does not bypass the server's cache. (man7.org)

Avoid a global cache drop on shared production nodes. Linux documents that `drop_caches` discards clean caches and can impose performance costs; it is a testing/debugging control, not a routine performance setting. (kernel.org) (kernel.org) Use a fresh file set, a clearly labeled warm-cache repeat or a maintenance window instead. A cold versus warm comparison is useful only if the state is known and repeatable. Capture storage-side cache policy when available and avoid calling a client-cache bypass “end-to-end cold.”

Fallback Detection, Results and Troubleshooting

Prove the mode with independent signals

Treat the requested `gdsio` transfer type as an input, not a verdict. The direct path should have the expected `gdscheck` capability, a supported target file and mount, functional integrity, and counters consistent with direct I/O. NVIDIA's `gds_stats` documentation identifies POSIX pool buffer statistics as system-memory bounce-buffer evidence for compatibility mode. For a short diagnostic run, NVIDIA describes a TRACE-level `cufile.log` message, `cufile IO mode: POSIX`. TRACE logging itself can perturb performance, so the diagnostic run should be separate from the timing run. A filesystem vendor's own fallback statement is another clue: WEKA says a requirements mismatch can return its path to standard I/O. (docs.weka.io)

Table 3 is a blank **local results record**. One row is required for each mount, GPU, transfer mode, block size and placement; do not pool different paths into a single average.

Path and GPU	Topology and mount	I/O geometry	Mode evidence	Measured result	Integrity and fallback
Local row A	PCI addresses; NUMA; root complex; filesystem and options	Block size; workers; achieved depth if available; read/write; duration; direct-I/O state	`gdscheck`; cuFile counters; diagnostic log	Throughput; latency if measured; process CPU utilization	Verify result; errors; POSIX/bounce counters; pass/hold

Path and GPU	Topology and mount	I/O geometry	Mode evidence	Measured result	Integrity and fallback
Remote row B	GPU to NIC route; RDMA state; client and server versions	Same fields, with cache state and file set	Same fields plus vendor client status	Same metrics and repeats	Same disposition
Comparator row C	Identical storage target and GPU placement	Matched geometry and duration	CPU-only or forced compatibility explicitly marked	Same metrics and repeats	Differences recorded without causal attribution

This table deliberately separates **requested mode**, **reported mode** and **measured outcome**. A high throughput number does not override a POSIX fallback counter or a verification error. PCIe link rates are capacity context, not target GDS rates: Intel's PCIe table lists 32 GT/s for PCIe 5.0 and a theoretical x16 example near 63 GB/s, while the actual storage route includes other devices and software. (www.intel.com) (www.intel.com)

Failure signatures and rollback

- **`gdscheck` lacks the intended mode:** record CUDA, GDS, GPU and kernel versions; compare them to current release notes. NVIDIA's P2PDMA support is conditional, and the `nvidia-fs` requirement differs by release and platform.
- **`EINVAL` or verification mismatch:** inspect file, memory-buffer and offset alignment before changing performance parameters. Linux warns that direct-I/O alignment behavior varies across filesystems and kernels; BeeGFS documents 4 KB alignment for its GDS I/O. (man7.org) (doc.beegfs.io)
- **CuFile call succeeds but POSIX counters move:** label the result compatible; inspect mount support, `O_DIRECT`, client RDMA state and `cufile.log`.
- **Remote path is unexpectedly slow:** compare GPU-to-NIC topology, NUMA placement, client RDMA and mount options. VAST requires an RDMA-enabled mount for its path, and BeeGFS provides a way to check storage and metadata RDMA connections. (kb.vastdata.com) (doc.beegfs.io)
- **Read appears implausibly fast:** check the file set, duration and client/server cache state. `O_DIRECT` on NFS bypasses only the client page cache.
- **Results vary between repeats:** record background jobs, storage occupancy and achieved concurrency before changing software. A requested fio depth may differ from its achieved depth. (github.com)

The rollback is operational: restore the known working application path and configuration, preserve the failed test's logs and version snapshot, and leave direct-path sign-off on hold. Do not convert a forced compatibility result into a production “GDS pass.” If a vendor-specific setting is changed, rerun the full functional matrix for that mount and compare the exact same geometry. This makes the decision auditable even when the final disposition is that ordinary I/O is adequate.

Data Analysis and Evidence

Required rate and useful comparisons

The most useful number is the ratio **measured sustained read rate divided by workload-required read rate** for the actual placement. In a **(Hypothetical Example)** with eight simultaneous readers at 0.5 GB/s each, required rate is 8 × 0.5 = **4 GB/s**. If the locally measured GDS run delivers 5 GB/s under the matched file set, the observed ratio is 5 / 4 = **1.25**. Those inputs are illustrative assumptions, not published vendor benchmarks or a universal acceptance threshold. A production decision should replace them with a trace of the actual pipeline, include variance across repeats, and state the minimum headroom the operator requires for concurrent activity. The arithmetic makes a capacity decision explicit while keeping it separate from the question of whether the direct path is active.

For every mode, report median and range of repeat throughput, transferred bytes, elapsed time, process CPU utilization, verification result and errors. `pidstat -u` can supply process CPU values; fio's own CPU utilization is averaged across jobs in a reporting group, so the aggregation must be identified. (man7.org) (github.com) Keep reads and writes separate: an application that reads training data and writes checkpoints has different service demands. Keep sequential and random workloads separate as well, because `gdsio` provides distinct selectors for them. If the distributed filesystem uses RDMA, capture its client and network evidence alongside throughput. BeeGFS asks for RDMA to both storage and metadata services; VAST's GDS documentation requires an RDMA-enabled mount. (doc.beegfs.io) (kb.vastdata.com)

Several published numbers are **specifications or vendor examples**, not local targets. PCI-SIG identifies PCIe 5.0 as 32 GT/s, and Intel's x16 example reaches about 63 GB/s at the link level. (pcisig.com) (www.intel.com) That number is not an expected `gdsio` output: the storage device, path, PCIe topology, data size and software can all be limiting. WEKA's documentation says RDMA typically becomes advantageous for reads of 32 KB or larger and writes of 256 KB or larger in its architecture; those are vendor-scoped guidance, not a universal GDS threshold. (docs.weka.io) BeeGFS specifies 4 KB alignment for its GDS requests, which is an eligibility condition rather than a performance forecast. (doc.beegfs.io) NVM Express released its 2.4 specifications on August 4, 2026, showing why transport and version records should be dated, but a specification revision itself does not qualify a mount. (nvmexpress.org)

A sound analysis also tests whether the comparator is fair. fio documents `direct=1` as usually selecting `O_DIRECT`, and its CUDA engine distinguishes a `cufile` path from a POSIX path that copies via a RAM buffer. (github.com) (github.com) (github.com) This can be useful as a supplementary cross-check, but do not blend fio and `gdsio` results as if they were the same workload generator. State each tool's version, engine, direct-I/O mode and achieved depth. The outcome is a local performance envelope with an integrity and mode verdict, not a universal vendor ranking.

Implications and Future Directions

Handover evidence

The **evidence bundle** for each result row should contain:

- **Run identity:** node, operator, date, start and end times.
- **Software manifest:** kernel, CUDA, driver, GDS and filesystem-client versions.
- **GPU map:** UUIDs and `nvidia-smi topo -m` output. (doc.beegfs.io)
- **PCI map:** `lspci -t`, storage controller and NIC addresses.
- **Mount proof:** `findmnt --target` output for every test file. (man7.org)
- **Tool syntax:** installed `gdscheck` and `gdsio` help output.
- **I/O geometry:** file list, size, alignment, block size, workers and duration. (man7.org)
- **Cache state:** record client and known server cache conditions separately.
- **CPU trace:** collect process-level utilization during each timed run.
- **Mode proof:** retain counters, compatible-path diagnostics and errors. (docs.weka.io)
- **Integrity proof:** retain verification output and any checksum record.
- **Disposition:** write pass, hold or fallback with the specific acceptance rule.

Requalification after change

The version gate will keep changing. NVIDIA's release notes added explicit mode reporting in GDS 1.16 and describe newer POSIX behavior when required cuFile libraries are absent. (docs.nvidia.com) (docs.nvidia.com) (docs.weka.io) The current local NVMe path can use P2PDMA without `nvidia-fs.ko` under the documented CUDA 12.8 conditions, while other configurations still depend on the driver and filesystem integration. A static checklist therefore needs an attached version snapshot and a dated support-matrix review, not an undated “GDS enabled” box.

For platform owners, the durable unit of acceptance is **GPU plus file path plus mount plus software release**. A second GPU behind a different switch, a changed NIC or a new filesystem-client build merits its own row. The Linux PCI tree and NUMA metadata provide a reproducible map of those differences. (docs.kernel.org) For procurement, the evidence bundle should travel with the proposed storage design: required workload rate, topology, support statements, integrity results, direct and compatible comparisons, CPU use and fallback counters. A vendor's feature announcement can start a conversation, but a dated client/mount support statement and local test complete it. DDN and Dell both describe GDS-related integrations publicly, yet their cited pages alone do not specify this article's tested path. (www.ddn.com) (www.dell.com)

GPU Smith states that its engineering recommendation is independent of reseller quota or a cloud service of its own. (gpusmith.com) In that advisor role, the practical recommendation is to set acceptance criteria before timing, use a reproducible scratch workload and report an honest hold when mode evidence is missing. This is especially important for distributed filesystems that can serve data correctly after a mode change. WEKA explicitly documents reversion to standard I/O when RDMA or GDS prerequisites are unmet. (docs.weka.io) Future revisions of a site runbook should update the versioned filesystem rows, rerun the same test geometry and archive both the prior and new mode evidence; otherwise a performance trend can be mistaken for a hardware change.

Frequently Asked Questions (FAQs)

How do engineers verify that GPUDirect Storage is enabled?

Run the installed `gdscheck` tool, capture its supported modes and topology, then test the **actual target file** with verified reads and writes. NVIDIA recommends `gdscheck` for installation verification and documents mode reporting. Add cuFile statistics or a short TRACE diagnostic to detect a POSIX path; the documented `cufile IO mode: POSIX` message is explicit. A successful benchmark alone establishes only that I/O completed.

What is a defensible GDSIO baseline command protocol?

Save `gdsio --help` from the installed package; choose a scratch directory, GPU and supported aligned block size; write verified files; read the same geometry; then run matched direct, compatible and CPU comparators. NVIDIA's guide maps `-I 1` to write and `-I 0` to read, and warns that `-D` directory tests need a write first. Record workers, size, I/O size, duration, direct-I/O state, repeats and cache state. (github.com) Treat queue depth as observed or tool-specific rather than inventing a `gdsio` flag. (github.com)

Does an unloaded `nvidia-fs` module mean GDS is disabled?

No blanket conclusion follows. NVIDIA documents an NVMe P2PDMA mode from CUDA 12.8 that can omit the module when its requirements are met. (docs.nvidia.com) Check the actual release, GPU, kernel, target file and `gdscheck` modes. Other paths can still require NVIDIA's filesystem integration; the package and driver state belong in the evidence record.

How should compatibility mode be interpreted?

It is a functional fallback path for cuFile I/O that stages through CPU system memory, so label its results separately from direct GDS. Monitor POSIX pool or bounce-buffer counters and, when necessary, a short TRACE run; neither a high score nor a successful cuFile call overrides those signals. Compare CPU load and integrity under matched conditions before deciding whether the ordinary path is sufficient.

Conclusion

GPUDirect Storage qualification is a sequence of **eligibility, integrity, mode proof and workload fit**. Begin with the required read or write rate. Inventory the exact GPU, software, filesystem and mount, then map the path through the PCIe and RDMA topology. Run `gdscheck`, perform verified reads and writes on a disposable file set, and compare controlled `gdsio` modes with the same geometry. Store throughput beside CPU utilization, errors, cache state and fallback evidence. The support matrix and blank result table make the decision reviewable per path, rather than per product name.

A direct-path pass means that the configured path is supported, the functional data is correct, the measured run has corroborating direct-mode evidence, and the local rate satisfies written workload criteria. A compatibility result can still be an acceptable production choice if the workload meets its own criteria, but it should be named accurately. When evidence is incomplete, the appropriate outcome is a documented hold with the ordinary path preserved and a narrow retest plan. That record remains useful as CUDA, GDS, kernel

and filesystem versions change. The archive gives the next software upgrade a fixed comparison point: the same file path, placement and workload can be retested while the former mode, integrity outcome and CPU cost remain visible. A change in a peak score alone is not sufficient sign-off.

Source links

Links cited in this article, in order of first appearance. Full addresses are provided for readers using extracted text.

1. https://docs.nvidia.com/gpudirect-storage/getting-started/
2. https://github.com/axboe/fio/blob/master/HOWTO.rst
3. https://docs.nvidia.com/gpudirect-storage/overview-guide/
4. https://docs.nvidia.com/gpudirect-storage/release-notes/index.html
5. https://docs.weka.io/5.0/weka-system-overview/networking-in-wekaio
6. https://docs.nvidia.com/gpudirect-storage/troubleshooting-guide/
7. https://man7.org/linux/man-pages/man2/open.2.html
8. https://doc.beegfs.io/latest/advanced_topics/gds_support.html
9. https://kb.vastdata.com/documentation/docs/nfs-features-for-enhanced-performance-3
10. https://docs.nvidia.com/gpudirect-storage/configuration-guide/
11. https://gpusmith.com/
12. https://man7.org/linux/man-pages/man1/pidstat.1.html
13. https://nvmexpress.org/specification/nvm-express-base-specification/
14. https://man7.org/linux/man-pages/man8/findmnt.8.html
15. https://www.ddn.com/press-releases/ddn-boosts-ai-leadership-with-exascaler-6/
16. https://www.dell.com/en-us/blog/powerscale-the-architectural-backbone-for-genai-workloads/
17. https://man7.org/linux/man-pages/man8/lspci.8.html
18. https://docs.kernel.org/5.17/admin-guide/abi-testing.html
19. https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/10/html-single/monitoring_and_managing_system_status_and_performance/index
20. https://gpusmith.com/articles/en/gpu-cpu-nic-affinity-rank-binding
21. https://man7.org/linux/man-pages/man2/statx.2.html
22. https://kernel.org/doc/html/v6.7/admin-guide/sysctl/vm.html
23. https://www.intel.com/content/www/us/en/docs/oneapi/optimization-guide-gpu/2023-0/asynchronous-and-overlapping-data-transfers.html
24. https://pcisig.com/sites/default/files/files/PCI-SIG-Seamless_Transition_to_PCIe_5.0_in_System_Implementations_FINAL.pdf

THE PUBLISHER

About GPU Smith

GPU Smith provides independent engineering and research for private AI infrastructure. We help technical buyers and operators reason about GPU workload sizing, hardware selection, procurement, deployment and inference operations, from the compute system to the networking, power and cooling requirements around it.

Start with the workload

Useful infrastructure decisions begin with the models, data, throughput, latency and operating constraints a team actually has. GPU Smith helps connect those requirements with system design and deployment choices. The goal is a privately controlled AI environment whose capacity and operational demands are understood before a purchasing decision.

Hardware and supplier research

Our public reference library covers GPUs, complete systems and networking components. The server-vendor directory and manufacturing research help teams investigate suppliers, compare options and follow sources behind technical claims. We distinguish manufacturer specifications from measured benchmarks and advertised capabilities from tested configurations. Reference pages are research resources, not a live inventory listing or a binding equipment quote.

Deployment and operations

GPU Smith's engineering scope includes infrastructure integration, networking, power, cooling and the practical operation of inference workloads. Our published articles explain the tradeoffs behind deployment, procurement and ongoing operations so teams can ask better questions and document decisions.

Work with GPU Smith

Explore the hardware library, GPU references, networking references and vendor research. Contact GPU Smith with your workloads and deployment constraints to discuss a project and confirm current scope, pricing and availability.

Inclusion of a manufacturer or product in our research does not imply a partnership, certification or endorsement.

Website: https://gpusmith.com