

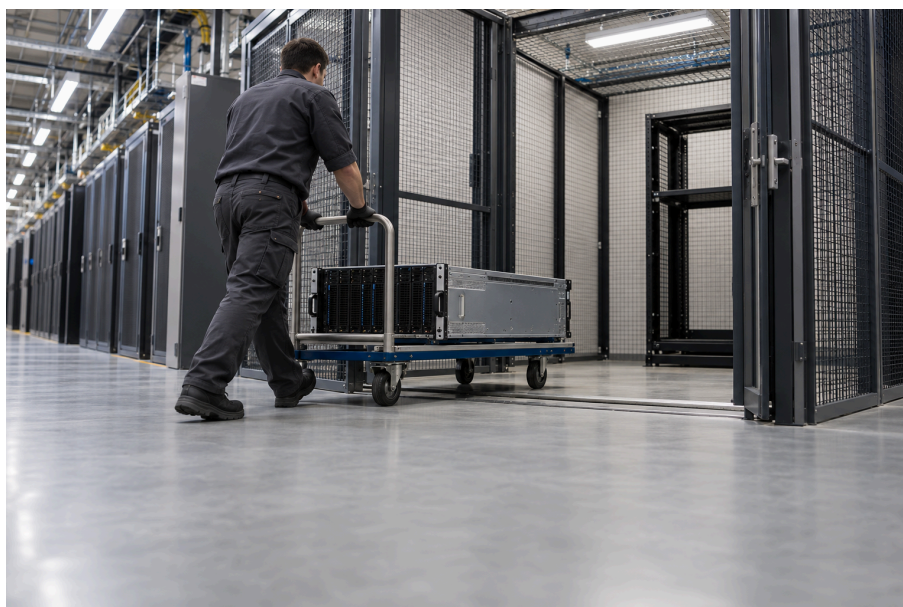


Silent Data Corruption in GPU Fleets: Detect and Screen

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 · 2026-10-29 · silent data corruption gpu · silent data corruption llm training · dcgmi diag level 4 memtest pulse · gpu burn-in testing before production · how to detect faulty gpu silent errors · dcgmi diag

Original article: <https://gpusmith.com/articles/en/silent-data-corruption-gpu-fleets-detection>



In this article

1. What silent data corruption is
 2. What the published studies show
 3. How SDC shows up in training and inference
 4. Screening with DCGM diagnostics
 5. Deterministic re-run and cross-check tests
 6. Telemetry that helps
 7. Screening cadence
 8. Quarantine and RMA policy
 9. Application-level defenses
 10. Cost model
 11. Runbook summary
 12. Sources
-

What silent data corruption is

Silent data corruption (SDC) is a hardware fault that returns a wrong answer and raises no error. Nothing appears in ECC counters, no Xid is logged, the kernel completes and the job keeps running. That makes it a different problem from the one covered in our [ECC and Xid triage](#) guide: there, the GPU reports the fault. Here, the only symptom is a bad number, and you find it only by checking output against something you trust.

The short version for GPU fleet operators: 2025 and 2026 production studies show that SDC-defective GPUs are real, often appear months after deployment, and are frequently missed by stress tests, including NVIDIA's own diagnostics. Run `dcgmi diag` anyway, but rely on deterministic replay and comparison, both offline and inside the training job, to actually catch SDC.

Meta's CPU paper gives a working definition: SDCs "are not captured by error reporting mechanisms" and "propagate across the stack and manifest as application-level problems" ([Dixit et al., 2021](#)). Its well-known example is a CPU core that computed `Int(1.1^53)` as 0 in a Spark decompression pipeline, so non-empty files were silently dropped.

What the published studies show

Read each rate with its population and definition. Most fleet-scale SDC data is about CPUs; the GPU numbers come from a few recent papers.

Source	Hardware	Population and definition	Finding
Meta, 2021	Server CPUs	Tests "across hundreds of thousands of machines", monitored "longer than 18 months"	"Hundreds of CPUs detected"; "a systemic issue across generations"
Meta, 2022	Server CPUs	Fleet detection since 2019	Cites an SDC occurrence rate of "one in thousand silicon devices"

Source	Hardware	Population and definition	Finding
Google, 2021	Server CPUs	Google fleet; exact rates withheld	"A few mercurial cores per several thousand machines"
Gemini Team, 2023	TPUv4	Gemini training at full scale	SDC events expected "to impact training every week or two"
Llama Team, 2024	Up to 16K H100	419 unexpected job interruptions in a 54-day snapshot	6 interruptions (1.4%) attributed to SDC
Lei et al., 2026	GPUs (ByteDance; model not named)	Online detection over 3.5×10^7 GPU-hours of LLM training	18 SDC incidents, 13 faulty GPUs
Zheng et al., 2026	GPUs (ByteDance; model not named)	23 defective GPUs characterized, 40 detected in production	Only 25% caught at pre-deployment burn-in

Three cautions. First, the Meta and Google figures describe CPU cores and say nothing about GPUs; we found no vendor-published SDC rate for any data center GPU as of October 2026. Second, Llama 3 counts only SDCs that interrupted the job, roughly one every 9 days ($54 / 6$) at that scale; corruption that quietly bent the loss curve would not appear in that table. Third, 3.5×10^7 GPU-hours / 18 incidents is about one *detected* incident per 1.9 million GPU-hours for one detector in one fleet. That is a floor, not a per-device rate.

The [AWS and Harvard study](#) covers training impact. It paired 15 nodes swept from a cloud fleet for exhibiting SDC with healthy nodes (the accelerator is not named). Under deterministic XLA execution, healthy nodes produced identical results, so every difference was SDC. Perturbations were usually small, but SDCs "can lead models to converge to different optima" and "even cause spikes in the training loss". Some of the unhealthy nodes "were initially healthy and began to fail the pre-checks after being used in training."

ByteDance's lifecycle data makes the case for screening over the GPU's whole life: "only 25% of SDCs were detected during pre-deployment burn-in", with "25% within the first two months, 10% within six months, and a substantial 40% around one year post-deployment" ([Zheng et al., 2026](#)). In GEMM cases, FP64 and FP32 CUDA cores failed more often than Tensor Core paths, and first failures also appeared in MoE, FlashAttention and RMSNorm kernels.

How SDC shows up in training and inference

The [OCP white paper "Silent Data Corruption in AI"](#) (v1.0, August 2025; authors from NVIDIA, Meta, Google, AMD, Intel, Arm and Microsoft) describes two training failure modes:

- **NaN propagation.** A corrupted value becomes NaN and spreads through collectives to the whole cluster, hiding its source. Restarting from a checkpoint without removing the bad accelerator just produces NaNs again.
- **Corrupted gradient variance.** The corrupted value is still a valid number, so every replica averages it in as correct. Training appears to progress; detection "can take weeks or even months", and NaN trapping does not catch it.

The GPU studies add three properties that shape screening:

1. **Non-deterministic manifestation.** On one faulty GPU, ByteDance reran nanoGPT 50 times under identical settings and saw divergent loss once. Re-running the faulty matmul 9.2×10^5 times gave three wrong results, an estimated error rate of 2×10^{-9} (Lei et al., 2026).
2. **Data dependence.** That fault appeared only for certain input value ranges. Across defective GPUs, replay occurrence rates ranged "from 100% down to 10^{-6} %" (Zheng et al., 2026).
3. **Persistence.** All incidents AEGIS detected "were reproducible on the corresponding GPUs." One GPU passed every offline suite after a reboot and showed SDC again a month later.

For inference, a defective device keeps returning wrong outputs for every request it serves. The OCP paper notes that large deviations are easier to catch than small ones. For an LLM endpoint, the realistic symptom is one replica disagreeing with its peers on the same deterministic request.

Screening with DCGM diagnostics

NVIDIA's DCGM Diagnostics is meant "to assess cluster readiness levels before a workload is deployed". It is explicitly not designed to "provide comprehensive hardware diagnostics" or to replace the field diagnostic used for RMA (NVIDIA DCGM Diagnostics). NVIDIA does not present it as an SDC detector.

Run levels per NVIDIA's DCGM documentation (page updated 2026-06-17; latest DCGM tag v4.7.0, 2026-10-21), runtimes measured on Hopper:

Level	Documented runtime	Tests added
-r 1	< 2.5 seconds	software
-r 2	< 2.5 min (4 GPU), < 10.5 min (8 GPU)	pcie (PCIe + NVLink), memory, memory_bandwidth
-r 3	< 10 min (4 GPU), < 35 min (8 GPU)	diagnostic, targeted_stress, targeted_power, nvbandwidth, nccl_tests
-r 4	< 45 min (4 GPU), < 2.25 hours (8 GPU)	memtest, pulse

The parts most relevant to SDC:

- **diagnostic** runs large matrix multiplies at 64-, 32- and 16-bit precision and walks the frame buffer verifying read-back. It fails on "unrecoverable memory errors, temperature violations, or XIDs"; the documentation does not say it checks the matmul results.
- **memory** and **memtest** write patterns over GPU memory and fail on mismatch. memtest (level 4 only) "will exercise GPU memory with various test patterns."
- **pulse** (level 4 only) fluctuates power "to create spikes in current flow on the board to ensure that the power supply is fully functional."
- **EUD** (Extended Utility Diagnostics, DCGM 3.1+) includes "confirmation of the numerical processing engines in the GPU." Once installed it runs inside level 3 (under 5 minutes) and level 4 (about 20 minutes), or alone with `dcgmi diag -r eud`. It needs a package matching the driver version and MIG

disabled. Its supported list covers V100, A100/A800, H100/H800, L40, L40S and L4; no Blackwell part was listed on 2026-10-28 ([NVIDIA EUD](#)).

```
dcgmi diag -r 3 -j          # JSON output for automation
dcgmi diag -r 4             # adds memtest and pulse
dcgmi diag -r "pcie,diagnostic" # named tests only
dcgmi diag -r eud -p "eud.passthrough_args='run_tests=compute,memory'"
```

How much SDC it catches

Two OSDI 2026 papers from ByteDance tested vendor diagnostics on GPUs known to be defective:

- AEGIS: "vendor-provided offline diagnostics tests" detected 2 of 8 previously faulty machines (25%). The authors attribute part of the gap to pre-screening bias: all eight had passed offline stress testing before entering the cluster ([Lei et al., 2026](#)).
- ByteRobust: "NVIDIA's EUD diagnostic tool achieves only 70% recall" in ByteDance's production environment ([Wan et al., SOSP '25](#)).
- SDCHunter: real-world coverage of 30.0% for NVIDIA hardware tools (EUD), 20.0% for GPU-Burn and 25.0% for loss and gradient-norm monitors ([Zheng et al., 2026](#)).

All three papers propose competing systems, so treat these recall figures as the authors' measurements, but the direction is consistent. A clean `-r 4` means no detectable memory, interconnect, power or thermal fault under synthetic load. It does not mean the GPU computes your workload correctly. In SDCHunter's words, "a fixed-duration stress test is not a reliable acceptance criterion."

Deterministic re-run and cross-check tests

Every effective method in these studies does the same thing: run identical work on different hardware in a way that should be bitwise identical, and compare. Gemini used "deterministic replay to isolate incorrect computations" ([Gemini Team, 2023](#)). The cloud fleet in the AWS study ran "a small LLM training run where training outputs are compared with pre-computed golden truth values" at node intake ([Ma et al., 2025](#)).

To build a golden-workload screen on NVIDIA GPUs:

1. **Pin the stack.** cuBLAS produces "the same bit-wise results at every run when executed on GPUs with the same architecture and the same number of SMs", but not across toolkit versions, and not with multiple active streams unless you set per-stream workspaces ([NVIDIA cuBLAS](#)). PyTorch does not guarantee reproducibility "across PyTorch releases, individual commits, or different platforms"; enable `torch.use_deterministic_algorithms(True)` and disable cuDNN benchmarking ([PyTorch](#)). Compare only within one image digest.
2. **Use your real kernels.** A few hundred steps of a scaled-down production model with fixed seed and data is a better probe than a GEMM loop, since first failures appeared in attention, MoE and normalization kernels too.
3. **Cover precisions and value ranges.** Include FP64/FP32 paths and inputs at several scales.

4. **Compare against peers.** Hash per-step loss, gradient norms and selected tensors on every GPU and compare across same-SKU peers. No single GPU is assumed to be correct, so agreement among several peers decides.
5. **Repeat.** At the reported manifestation rates, a single pass proves little. Repetition raises the odds of a catch; it never proves a GPU clean.

Replay also localizes faults during an incident. SDCHunter splits the data-parallel dimension into two replicas fed the same batch and hashes tensors at pipeline-parallel boundaries, with about 3% runtime overhead, then pins down the GPU offline in under an hour. ByteDance measured deterministic training at "no measurable throughput loss (within 0.01% step-time difference)" in their environment; your cost will depend on model and kernels.

Telemetry that helps

Telemetry does not detect SDC. ByteDance found that "the ECC error count shows no correlation with the SDC occurrences", and clocks and thermals on defective GPUs matched healthy ones. Use telemetry to rule causes in or out:

- **Row remapping and ECC.** NVML and nvidia-smi expose remapped-row counts, a pending flag and a failure flag. Xid 48, 94 and 95 mark uncorrectable, contained and uncontained ECC errors ([NVIDIA](#)). Memory faults and SDC can overlap: one of AEGIS's eight faulty machines was an HBM single-bit-flip case.
- **Per-GPU outliers.** Power, clock and temperature deltas against same-node peers separate stragglers and cooling problems from corruption. See our [DCGM exporter alerts](#) guide.
- **Job signals.** Loss spikes, NaN/Inf counts and gradient-norm excursions are cheap. They should trigger a replay, not decide the outcome.

Screening cadence

Trigger	What to run	Why
Acceptance / burn-in	-r 4 plus EUD, then golden workload across all nodes	About a quarter of ByteDance's SDC cases surfaced here; see the acceptance testing runbook
After repair, reseal, firmware or driver change	-r 3 plus golden workload	Re-establish peer agreement on the new baseline
Periodic, rolling	Golden workload on drained nodes; -r 3 or -r 4	Most cases surfaced after deployment. The OCP paper describes periodic screening covering a fleet within 45 to 60 days
Before a large job	-r 2 or -r 3 plus a short golden run on allocated nodes	Cheap insurance
Unexplained loss spike or NaN	Deterministic replay from last good checkpoint on split replicas	Localizes the device

NVIDIA positions level 2 as "an epilogue on failure" and levels 3 and 4 for administrators "as post-mortem." Meta's CPU experience favors frequent short tests alongside occasional long ones: in-production Ripple tests matched 70% of out-of-production Fleetscanner's detections within 15 days, against about 6 months for Fleetscanner, and each found defects the other missed (Dixit et al., 2022). For GPUs, the equivalent is checks inside the training job.

Quarantine and RMA policy

Keep two decisions separate: get suspect hardware out of the job fast, then confirm the device offline.

SDCHunter's first deployment lesson is to "remove suspicious machines from the training path quickly and defer device-level confirmation to offline analysis."

1. **Suspect.** Loss spike, NaN, shadow-replica disagreement or golden mismatch. Cordon the node group, resume from a checkpoint validated elsewhere, and keep the triggering trace.
2. **Localize.** Replay deterministically on split replicas down to one GPU. Run `-r 4` and EUD in parallel and preserve row-remap, ECC and Xid state.
3. **Decide.** A reproducible mismatch on one GPU means permanent quarantine, even if the GPU passes diagnostics after a reboot. No reproduction but a clear trigger means more replays, because low-rate faults need many iterations. A mismatch that follows the job to different hardware points at software: AEGIS traced one "SDC-like" alarm to an out-of-bounds write from a framework bug.
4. **Escalate.** NVIDIA's DRAM RMA criterion "is met when the row-remapping failure flag is set and validated by the field diagnostic." The flag trips on a remap attempt in a bank that already has eight uncorrectable-error rows remapped, a repeat remap of the same row, or "after 512 total remappings" (NVIDIA RMA policy). There is no published equivalent for compute SDC, so your evidence package (reproducer, input reference, expected vs observed tensors, software digests, diagnostic logs) is what you bring to the OEM.

Application-level defenses

- **Shadow replicas and hot standbys.** Gemini ran "proactive SDC scanners on idle machines and hot standbys." The AWS study proposes a shadow data-parallel replica that recomputes a random rank's step and compares gradients; OCP describes a similar shadow-pool scheme.
- **Checksums plus replay.** Classic checksummed matmul "fails to reliably detect SDCs with high precision and recall" in the AWS experiments, because SDC effects can fall inside floating-point error bounds. AEGIS uses the higher-precision accumulators in modern GPUs for a checksum sensing stage, confirms by recomputation, and reports 0.86% overhead.
- **Cheap guards.** NaN/Inf traps and gradient clipping contain NaN propagation (OCP) but miss representable corruption; in SDCHunter's fault injection, loss and gradient-norm monitors caught 16% and 17% of SDC GPUs.
- **Checkpoint hygiene.** Implicit anomalies "may continue to produce invalid checkpoints" (SDCHunter). Keep enough history to go back past the suspect window, and mark a checkpoint good only after validation on separate hardware.

Cost model

Worked example: 1,024 GPUs (128 nodes $\times$ 8). Rates are assumptions.

Screening. Level 4 upper bound 2.25 h per 8-GPU node: $128 \times 8 \times 2.25 = 2,304$ GPU-hours per fleet pass. Quarterly: $4 \times 2,304 = 9,216$ GPU-hours a year. Add a monthly 30-minute golden run (assumption): $1,024 \times 0.5 \times 12 = 6,144$. Total 15,360 GPU-hours, against $1,024 \times 8,760 = 8,970,240$ GPU-hours of capacity, or about 0.17%.

Exposure. Assume the AEGIS detected rate transfers (order of magnitude only): $8,970,240 / 1,940,000 \approx 4.6$ incidents a year. Assume each undetected incident costs a 3-day full-cluster rollback: $1,024 \times 72 = 73,728$ GPU-hours, before engineering time.

Comparison. If offline screens catch 25% to 30% of incidents, as ByteDance measured, they save roughly $0.25 \times 4.6 \times 73,728 \approx 84,800$ to $0.30 \times 4.6 \times 73,728 \approx 101,700$ GPU-hours a year for 15,360 spent. In-job detection at 0.86% overhead costs $0.0086 \times 8,970,240 \approx 77,100$ GPU-hours a year, about one rollback, for much higher coverage.

Under these assumptions offline screening is cheap enough to run routinely, but most protection for long synchronous training comes from online detection. Our [reliability budget](#) article shows how to fold this into goodput targets.

Runbook summary

1. **Intake:** -r 4 plus EUD, then golden workload with bitwise peer comparison; record the image digest.
2. **Telemetry:** row-remap flags, ECC/Xid alerts, per-GPU outliers, used to exclude known faults, never to clear a GPU.
3. **Rolling screen:** drained-node golden runs and -r 3 on a fixed cycle; -r 2/-r 3 before long jobs.
4. **In-job:** NaN/Inf traps, loss and gradient monitors, plus checksum-and-replay or shadow replicas on large runs.
5. **Incident:** cordon, resume elsewhere from a validated checkpoint, replay to find the GPU, run diagnostics in parallel.
6. **Disposition:** reproducible mismatch means permanent quarantine and a vendor evidence package; a post-reboot pass is not a release criterion.
7. **Review:** record which method caught each incident and shift effort toward what works in your fleet.

GPU Smith helps teams design acceptance, screening and quarantine processes for private GPU clusters; [contact us](#) if that is useful.

Sources

1. Dixit, H. D. et al. "Silent Data Corruptions at Scale." arXiv:2102.11245, Facebook, 2021-02-22. <https://arxiv.org/abs/2102.11245>
2. Dixit, H. D. et al. "Detecting silent data corruptions in the wild." arXiv:2203.08989, Meta Platforms, 2022-03-16. <https://arxiv.org/abs/2203.08989>

3. Hochschild, P. H. et al. "Cores that don't count." HotOS '21, Google, 2021.
<https://sigops.org/s/conferences/hotos/2021/papers/hotos21-s01-hochschild.pdf>
4. Gemini Team. "Gemini: A Family of Highly Capable Multimodal Models." arXiv:2312.11805, Google, 2023. <https://arxiv.org/abs/2312.11805>
5. Llama Team. "The Llama 3 Herd of Models." arXiv:2407.21783, Meta, 2024.
<https://arxiv.org/abs/2407.21783>
6. Ma, J., Pei, H., Lausen, L., Karypis, G. "Understanding Silent Data Corruption in LLM Training." arXiv:2502.12340, Harvard University and Amazon Web Services, 2025-02-17.
<https://arxiv.org/abs/2502.12340>
7. Zheng, W. et al. "SDCs in the Wild: Characterizing and Diagnosing SDC-defective GPUs in Production LLM Training." OSDI '26, USENIX, July 2026.
<https://www.usenix.org/conference/osdi26/presentation/zheng>
8. Lei, K. et al. "Safeguarding LLM Training at Scale: Online SDC Detection and Insights from 35 Million GPU Hours." OSDI '26, USENIX, July 2026. <https://www.usenix.org/conference/osdi26/presentation/lei>
9. George, N. et al. "Silent Data Corruption in AI." Open Compute Project white paper v1.0, 2025-08-26.
<https://www.opencompute.org/documents/sdc-in-ai-ocp-whitepaper-final-pdf>
10. NVIDIA. "DCGM Diagnostics." DCGM User Guide, accessed 2026-10-28.
<https://docs.nvidia.com/datacenter/dcgmlatest/user-guide/dcgm-diagnostics.html>
11. NVIDIA. "Extended Utility Diagnostics (EUD)." DCGM User Guide, accessed 2026-10-28.
<https://docs.nvidia.com/datacenter/dcgmlatest/user-guide/dcgm-eud.html>
12. NVIDIA. "RMA Policy: Thresholds for Row Remapping." GPU Memory Error Management, accessed 2026-10-28. <https://docs.nvidia.com/deploy/a100-gpu-mem-error-mgmt/rma-policy-thresholds-for-row-remapping.html>
13. NVIDIA. "User Visible Statistics." GPU Memory Error Management, accessed 2026-10-28.
<https://docs.nvidia.com/deploy/a100-gpu-mem-error-mgmt/latest/user-visible-statistics.html>
14. NVIDIA. "cuBLAS Library: Results Reproducibility." CUDA Toolkit documentation, accessed 2026-10-28.
<https://docs.nvidia.com/cuda/cublas/index.html>
15. PyTorch. "Reproducibility." PyTorch documentation, accessed 2026-10-28.
<https://docs.pytorch.org/docs/stable/notes/randomness.html>
16. Wan, B. et al. "Robust LLM Training Infrastructure at ByteDance" (ByteRobust). SOSP '25, arXiv:2509.16293, 2025. <https://arxiv.org/abs/2509.16293>

Source links

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

1. <https://arxiv.org/abs/2102.11245>
2. <https://arxiv.org/abs/2203.08989>
3. <https://sigops.org/s/conferences/hotos/2021/papers/hotos21-s01-hochschild.pdf>
4. <https://arxiv.org/abs/2312.11805>
5. <https://arxiv.org/abs/2407.21783>
6. <https://www.usenix.org/conference/osdi26/presentation/lei>

7. <https://www.usenix.org/conference/osdi26/presentation/zheng>
8. <https://arxiv.org/abs/2502.12340>
9. <https://www.opencompute.org/documents/sdc-in-ai-ocp-whitepaper-final-pdf>
10. <https://docs.nvidia.com/datacenter/dcgmlatest/user-guide/dcgml-diagnostics.html>
11. <https://docs.nvidia.com/datacenter/dcgmlatest/user-guide/dcgml-eud.html>
12. <https://arxiv.org/abs/2509.16293>
13. <https://docs.nvidia.com/cuda/cublas/index.html>
14. <https://docs.pytorch.org/docs/stable/notes/randomness.html>
15. <https://docs.nvidia.com/deploy/a100-gpu-mem-error-mgmt/latest/user-visible-statistics.html>
16. <https://docs.nvidia.com/deploy/a100-gpu-mem-error-mgmt/rma-policy-thresholds-for-row-remapping.html>

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>

This article is educational. Verify consequential medical, legal, financial, regulatory and technical decisions with appropriate professionals and the cited primary sources. Company services are described separately from the article's research findings.