Rillor

METHODS NOTE

Burn-in and acceptance tests for GPU servers

A staged GPU burn-in test for servers, built from dated vendor documentation: baseline records, DCGM diagnostics, sustained load, NCCL and RDMA bandwidth, error signatures and a written acceptance record.



Contents

01 Summary

02 Background

03 Method

04 Findings

05 Implications

06 Limits of this analysis

07 Sources

Summary

Public documentation from NVIDIA, open-source test projects and published fleet studies is enough to write a staged GPU burn-in test and acceptance procedure into purchase terms, and it shows where each tool stops and which numbers it leaves to the buyer.

- NVIDIA's Data Center GPU Manager (DCGM) diagnostics run at four levels. NVIDIA measured level 3 at under 35 minutes and level 4 at under 2.25 hours on 8-GPU Hopper systems, and its guide frames the levels as readiness checks, follow-up after a failure and administrator post-mortems, without calling any level a burn-in or acceptance test [1].
- NVIDIA states that its Data Center GPU Manager diagnostics are not designed to provide comprehensive hardware diagnostics or to replace its field diagnostic, which it calls the authoritative tool for GPU health and which is usually required before a return (RMA) can start [1, 11].
- None of the vendor documents reviewed here sets a burn-in duration for a GPU server. The durations they print are measured run times, defaults and example arguments, such as the 60-second default run of the open-source gpu-burn container [1, 16, 19].
- The NCCL tests, NVIDIA's benchmarks for its collective communication library, and the perftest benchmarks for RDMA (remote direct memory access) networks report bandwidth without pass thresholds, and NVIDIA advises recording network results before production as a baseline [4, 6, 11].
- NVIDIA's Data Center GPU Manager tests collective communication on a single node only, and its multi-node diagnostic, available from version 4.3.0, supports GB200 NVL and GB300 NVL systems, with Vera Rubin devices added in version 4.8.0 [1, 2, 3].
- NVIDIA expects HPL, the double-precision Linpack benchmark, to run low on B300 and GB300 systems because their native FP64 throughput is low, so a low HPL score there is no sign of a fault on its own [7, 31].
- Xid codes, the GPU error codes NVIDIA catalogs with an immediate and an investigatory action for each, point a failed test to its cause. On two research clusters, 43% and 63% of PCI errors co-occurred with Xid 79, a GPU that has fallen off the bus [8, 24].
- Routine health checks miss some faulty servers. In two research clusters, finding faulty servers, which the authors call lemon nodes, from historic data "led to 10% reduction in large job failures (512+ GPUs), from 14% to 4%" [24].

Background

Reliability engineering explains why burn-in exists. The NIST/SEMATECH e-Handbook describes the start of the bathtub curve as an early failure period with "a high but rapidly decreasing failure rate", typically lasting several weeks to a few months, and applies the curve to repairable systems too [29]. A burn-in runs a unit through part of that period under load before it carries work.

On a GPU cluster a lost job can be large. The authors of a paper on pre-training a 405-billion-parameter model on up to 16K H100 GPUs report 466 job interruptions in a 54-day period, 419 of them unexpected, and note that "a single GPU failure may require a restart of the entire job" [25]. About 78% of the unexpected interruptions were attributed to confirmed or suspected hardware issues, and the paper gives GPU issues, the largest category, as 58.7% of them [25].

Some faults leave no log entry. The author of gpu-burn, an open-source stress tool, wrote in 2012 that failing GPUs "often fail silently and produce incorrect results" when slightly unstable [36]. A 2021 paper reports that silent data corruptions in CPUs are not captured by the processor's error reporting mechanisms [28]. A useful test checks its own results.

Acceptance is also a contract event. The Uniform Commercial Code, as published by Cornell's Legal Information Institute, gives a buyer the right to inspect goods before payment or acceptance, and presumes a place or method of inspection fixed by the parties to be exclusive [34]. Acceptance occurs when the buyer, after a reasonable opportunity to inspect, signifies that the goods conform or that it takes or keeps them in spite of a non-conformity, or fails to make an effective rejection, or does any act inconsistent with the seller's ownership [35]. Section 2-513 opens with "Unless otherwise agreed", so a contract can set its own terms [34]. This note is general information, not legal advice.

This methods note sets out a protocol in nine stages, built from public tools, for buyers and engineers who write acceptance terms for GPU servers. It then compares what published operators run.

Method

What was examined. The documentation of the tools a GPU server acceptance draws on, and published accounts of how operators of GPU fleets validate nodes: what each tool measures, how it runs, how long it runs where a document says so, and what pass criteria it prints.

Sources. Thirty-six sources, each opened and read for the passages cited: 17 NVIDIA documents (DCGM, the NCCL tests, nvbandwidth, HPC-Benchmarks, the Xid catalog, the row-remapping return policy, nvidia-smi, the GPU Debug Guidelines, Fabric Manager, the DGX B300 user guide, NVSM, a DGX BasePOD guide and Base Command Manager), six open-source tool and benchmark references, the Linux kernel's PCIe error guide, the NIST/SEMATECH e-Handbook, three operator documents, six arXiv papers and two sections of the Uniform Commercial Code. The organizations named appear as the publishers of these documents. Most NVIDIA pages carry their own "last updated" date. The nvidia-smi page and the Base Command Manager chapter show none, so they are dated by version and by the modification date their server or page metadata reports. GitHub files carry the date their last change reached the default branch, and arXiv papers the date of the version read.

How. Each stage uses tools whose documentation says what they measure, with commands quoted from it. A duration or a pass criterion appears only as a document states it, with its scope. Durations were sought in the DCGM, NVSM, Base Command Manager, BasePOD, HPC-Benchmarks, NCCL tests, perftest, nvbandwidth and gpu-burn documents and the three operator pages. "Derived" marks arithmetic on a source's own numbers. No hardware was tested for this note, and the protocol describes no company's internal procedure.

Where sources disagree. The DCGM user guide lists four run levels and measures level 3 at under 35 minutes on 8-GPU Hopper systems [1]. NVIDIA's GPU Node Triage page lists three levels and says the long test takes approximately 30 minutes [11]. This note uses the user guide. The pre-training paper's Table 5 prints 72 memory interruptions as 17.2%, which matches a base of 419, and 148 faulty-GPU interruptions as 30.1%, where 148 of 419 is 35.3% (derived) [25]. This note uses the counts. The paper's 58.7% for GPU issues is the sum of the table's printed GPU percentages. By its counts, the GPU rows hold 268 of 419 interruptions, 64.0% (derived) [25].

Findings

1. Inventory and baseline capture before any load

The first record is the server as received. nvidia-smi reports each board's serial number, which matches the number printed on the board and is immutable, the VBIOS version, and the version of the InfoROM, a small persistent store of configuration and state data for the GPU [10]. It reports PCIe generation and width as current, which may be reduced when the GPU is idle, and as the maximum the GPU and system allow [10]. The counters later stages compare against sit in the same output [10]:

Field	What nvidia-smi documents	Why it goes in the baseline
Aggregate error-correcting code (ECC) counts	A lifetime counter that cannot be cleared on Ampere and later	Errors under load show as a change from it
Row remapper	Remaps that take effect only after a GPU reset, a remapping failure field and a bank availability histogram	Remaps during the burn-in show here
PCIe replays	Replays since reset, and a rollover after 4 consecutive replays retrains the link	A marginal link shows as replays
Clocks event reasons	The factors reducing clock frequency	Explains a slow result under load
Power draw	Board power averaged over the last second	Shows whether load reached its target
NVLink errors	`nvidia-smi nvlink -e` for error counters, `-ec` for per-lane CRC and ECC counters	Separates link faults from GPU faults

NVIDIA's GPU Debug Guidelines list checks to run before jobs [12]. They confirm that the GPU driver, nv-peer-mem, gdrdrv and, on HGX systems, Fabric Manager are loaded, that GPU count, PCIe link speed and VBIOS are as expected, and on NVSwitch systems that Fabric Manager is running and the NVLink topology is correct. They also check for recent Xid errors, inactive network links and missing filesystem mounts [12].

On NVSwitch systems without ALI, CUDA initialization fails with `cudaErrorSystemNotReady` if an application starts before Fabric Manager has initialized the system [14]. From driver release 620 Fabric Manager is versioned on its own, where earlier it had to match the driver, so the baseline names both versions [14]. On DGX systems, `sudo nvsm show health` should report every check and the overall status as Healthy [15].

2. Single-node diagnostics with dcgmi diag levels 1 to 4

DCGM Diagnostics, formerly the NVIDIA Validation Suite, is designed to "assess cluster readiness levels before a workload is deployed", in the words of its user guide, last updated 17 June 2026 [1]. The guide sets level 1 as a readiness metric, level 2 as an epilogue on failure, and levels 3 and 4 as post-mortem tools for an administrator, and it does not call any level a burn-in or acceptance test [1]. Run times below are NVIDIA's, measured on Hopper systems, and each level also runs the tests of the levels below it [1].

Level	Tests that start at this level	4-GPU system	8-GPU system
r1 (short)	software	under 2.5 seconds	under 2.5 seconds
r2 (medium)	pcie (PCIe and NVLink), memory, memory_bandwidth	under 2.5 minutes	under 10.5 minutes
r3 (long)	diagnostic, targeted_stress, targeted_power, nvbandwidth, nccl_tests	under 10 minutes	under 35 minutes
r4 (extra long)	memtest, pulse	under 45 minutes	under 2.25 hours

The plugins carry their own pass rules [1]. The PCIe test checks peer-to-peer correctness, errors and replays, and accepts a minimum PCIe generation and width. The diagnostic test runs large matrix multiplies while writing frame buffer data and checking that it reads back correctly. Targeted stress fails below 95% of its target performance by default, or on unrecoverable memory errors, single-bit errors over a set maximum, temperature violations, Xids, or more than 160 PCIe replays by default [1]. The guide's example configuration sets pass parameters per GPU model, and its example values belong to the model shown [1].

`--iterations` repeats the run to lengthen it, `-j` prints JSON, logs go to `/var/log/nvidia-dcgm/nvvs.log`, and `--fail-early` checks for a failure every 5 seconds in targeted power, targeted stress and the diagnostic test [1]. From DCGM 4.7.1, results include each test's elapsed runtime with millisecond precision [3].

The guide states that DCGM Diagnostics are not designed to provide comprehensive hardware diagnostics, actively fix problems, replace the field diagnosis tools or facilitate any RMA process [1]. NVIDIA's GPU Node Triage page recommends level 3 for fully stressing a system [11]. It calls the field diagnostic "the authoritative and comprehensive NVIDIA tool" for GPU health, usually required before an RMA, and tells users to ask their system vendor when, if and how to run it [11].

3. Sustained compute, memory and power load

The level 3 and level 4 plugins load the GPUs [1]. Targeted power drives a GPU toward its thermal design power (TDP) and holds it there, for 120 seconds by default [1]. The level 4 pulse test swings power usage to create current spikes that check the board's power supply, with default parameters that NVIDIA says it verified, by measuring with oscilloscopes, to create failure conditions. Memtest, from DCGM 2.4.0, exercises GPU memory with test patterns in the manner of memtest86 [1]. Crusoe's published burn-in sustains rated TDP under continuous compute load and runs 20 ramp-up and ramp-down cycles, which Crusoe says catch power delivery faults that steady load misses [20].

The gpu-burn README describes a multi-GPU CUDA stress test whose container burns for 60 seconds by default, with options for a share of GPU memory (`-m N%`), doubles (`-d`) and Tensor Cores (`-tc`), and an example run of `gpu_burn -d 3600` [19]. Its author's page describes using all of the GPUs' memory and checking results while the GPUs keep running [36]. The README recommends no duration [19].

On DGX systems, NVIDIA provides NVSM, whose stress test loads the GPUs, PCIe, DIMMs, storage drives, CPUs and network cards at once and reports at the end whether each passed or failed, with an optional duration and the example `nvsm stress-test 60 --force` [15, 16]. Base Command Manager's burn framework is designed "to stress test newly built machines" [18]. Its default configuration erases the filesystem and partition data of every attached drive. It stops all tests at the first failure, retries the failed phase after a restart, keeps old logs, and needs power control through a BMC over IPMI, iLO or Redfish [18].

NVIDIA's HPC-Benchmarks container, version 26.02.01 (updated 26 August 2026), packages HPL, HPL-MxP, HPCG and STREAM with readiness microbenchmarks, among them NCCL tests and GEMM [7]. HPL solves a dense linear system in 64-bit arithmetic and checks the accuracy of the solution [31]. The container's notes say B300 and GB300 have low native FP64 throughput, so HPL performance there is expected to be low, and that NVIDIA's HPL can emulate FP64 on them with the Ozaki-II scheme [7]. STREAM measures sustainable memory bandwidth on datasets much larger than the cache [30].

NVIDIA's Best Practices page lists the key metrics to collect as time series on GPU systems, which a load test can record too: power at each supply and at the PDU, the PDU power factor, CPU and GPU power and temperature, GPU memory temperature and clocks, fan speed and inlet temperature, in-band with the DCGM Prometheus plugin or out-of-band with IPMI [12].

4. Bandwidth inside the server: nvbandwidth and all_reduce_perf

nvbandwidth measures memcpy bandwidth across a system's links with copy-engine or kernel copies, and its README notes that further tuning may be needed to reach peak [13]. DCGM runs it at level 3 [1].

The NCCL tests "check both the performance and the correctness of NCCL operations" [5]. The single-node example is `./build/all_reduce_perf -b 8 -e 128M -f 2 -g 8` on 8 GPUs [5]. `-c` sets how many iterations check results, and `-N 0` runs cycles without end [5]. Each test reports operation time, algorithm bandwidth and bus bandwidth [4]. For all-reduce, bus bandwidth multiplies algorithm bandwidth by 2(n-1)/n,

which is 1.75 on 8 ranks (derived), so it can be compared with the hardware peak at any number of ranks [4]. NVIDIA's document says bus bandwidth reflects the hardware bottleneck, whether NVLink, PCI, QPI or the network, and it sets no pass threshold [4].

NVIDIA's DGX BasePOD guide for DGX H200 and H100 systems, dated 16 June 2025, validates a node-level NCCL test on 8 GPUs [17]. Its example output counts wrong results, reports zero out-of-bounds values and gives an average bus bandwidth, and the guide states no threshold [17]. DCGM's NCCL plugin checks correctness on single-node systems only and stays disabled unless `DCGM_NCCL_TESTS_BIN_PATH` is set [1]. The MegaScale paper's authors run a full-mesh loopback test from every RDMA network card to the host's memory and GPUs, then an all-to-all NCCL test inside the node [26].

5. Fabric tests between servers: ib_write_bw and multi-node all_reduce_perf

perftest is a set of RDMA bandwidth and latency benchmarks, including `ib_write_bw` [6]. Built with `CUDA_H_PATH` set, it offers a `--use_cuda=<gpu index>` flag for GPUDirect use. Bandwidth tests run for a set number of iterations or a fixed duration, and latency tests report minimum, median and maximum [6].

The NCCL tests cross nodes through MPI, as in `mpirun -np 64 -N 8 ./build/all_reduce_perf -b 8 -e 8G -f 2 -g 1` for 64 GPUs on 8 nodes [5]. NVIDIA's BasePOD guide follows its node test with a cluster-level test on 4 nodes and 32 GPUs [17]. MegaScale adds an all-reduce with neighbouring machines under the same top-of-rack switch [26].

NVIDIA's triage page uses all_reduce_perf between groups of nodes, says node pairs should give the same performance, and retests a slow pair against different nodes to isolate the fault [11]. It advises running network tests before production and recording the results as "a baseline for future comparisons" [11]. One operator reports as much as 50% to 75% bandwidth loss during the bring-up of one of its research clusters [24].

DCGM's multi-node diagnostic, `dcgmi mndiag`, runs from DCGM 4.3.0, and its user guide page lists GB200 NVL and GB300 NVL systems and one test, mnubergemm, an MPI application that stresses GPUs across hosts at once [2]. It needs OpenMPI, passwordless SSH and the same driver version on each node. Its run time, `mnubergemm.time_to_run`, defaults to 3600 seconds, its example sets 600 seconds, and the page prints no pass threshold [2]. DCGM's release notes add the nvloom and mnnvbandwidth diagnostics in 4.7.0 and Vera Rubin devices in 4.8.0 [3]. nvbandwidth's multi-node pairwise test runs with exactly 2 MPI ranks, one per GPU, and its full multi-node run uses one rank per GPU, as in its example of 8 ranks on 2 nodes of 4 GPUs [13].

6. Storage and host checks

Crusoe's node tests cover PCIe enumeration of the expected GPUs, network cards and NVMe controllers, CPU and host memory throughput with sysbench, block and local NVMe bandwidth, IOPS and latency, and sustained CPU stress [20]. fio runs a job file written to match the I/O load to simulate, and its `verify` option

checks file contents after each iteration [32].

The Linux kernel's Advanced Error Reporting guide separates correctable PCIe errors, from which the protocol recovers without loss of data, from uncorrectable ones, which impact the interface [33]. NVIDIA's entry for Xid 79 points to system event logs and kernel PCI event logs for further signs [8].

Host processors need result checks too. The silent data corruption paper reports that such errors are not traceable at the hardware level, and that its authors' tests across their fleet found hundreds of affected CPUs across generations [28]. Its figures concern CPUs. The pre-training paper separately lists 6 silent data corruption interruptions in its 54 days [25].

7. Reading error signatures: Xid codes, ECC counters and remapped rows

NVIDIA's Xid catalog, last updated 11 November 2026, gives each Xid an immediate action, needed to recover the system, and an investigatory one [8]. NVIDIA ties Xids 48, 63, 64, 92, 94 and 95 to GPU memory errors [11]. The catalog lists these actions in its own codes [8]:

Xid	Catalog description	Immediate, then investigatory action
48	Double Bit ECC Error	RESET_GPU when alone, DRAIN_AND_RESET with 63 or 64, then RUN_FIELDDIAG when alone
63	GPU memory remapping event	IGNORE, IGNORE
64	GPU memory remapping failure	RESET_GPU, CONTACT_SUPPORT
79	GPU has fallen off the bus	RESTART_BM, CONTACT_SUPPORT
92	High single-bit ECC error rate	IGNORE, CONTACT_SUPPORT
94	Contained memory error	RESTART_APP, IGNORE (sympathetic)
95	Uncontained memory error	RESET_GPU, IGNORE (sympathetic)
119	GSP RPC Timeout	RESET_GPU, INVESTIGATE_SW

Xid 79 is often caused by hardware failures on the PCI Express link [8], and on two research clusters 43% and 63% of PCI errors co-occurred with it [24]. Xid 74, an NVLink error, can be reported by a GPU because the GPU at the other end failed and the link went down, so a 74 found during a burn-in is read at both ends of the link [8]. The catalog's mechanical check asks that device seating and all connections be secure [8].

NVIDIA's field diagnostic decides whether a GPU qualifies for return, and the criterion is met when the row-remapping failure flag is set and validated by that diagnostic [9]. The flag is set by an uncorrectable-error remap attempt in a bank that already has eight such rows remapped, by an attempt on a row already remapped, or after 512 uncorrectable-error remaps in total [9]. On Blackwell, a third such attempt in one bank invokes channel repair in the GPU's high-bandwidth memory (HBM) when a spare channel exists [9]. The flag reads in-band through NVML or nvidia-smi and out-of-band through SMBPBI [9]. These are return criteria, and a stricter limit at delivery is for the purchase terms to state.

A study of the Delta system's 1,056 A100 and H100 GPUs, over 2.5 years and 11.7 million GPU hours, found 3.2 times lower per-GPU mean time between memory errors on H100 than on A100 [27]. The pre-training paper's Table 5 lists 148 interruptions from faulty GPUs, 72 from GPU HBM3 memory and 35 from network switches and cables [25].

8. Duration, pass, fail and retest rules

None of the documents reviewed sets a burn-in duration for a GPU server. The durations they do print fall into four kinds:

Kind	Example	Source
Measured run time	DCGM level 3 under 35 minutes and level 4 under 2.25 hours on 8-GPU Hopper systems	[1]
Stated estimate	Level 3 "approximately 30 minutes"	[11]
Default	Targeted power 120 seconds, gpu-burn container 60 seconds, mndiag (mnubergemm) 3600 seconds	[1, 2, 19]
Example argument	`gpu_burn -d 3600`, `nvsm stress-test 60`, mndiag 600 seconds	[2, 16, 19]

The bathtub curve's "several weeks to a few months" is the typical length of the early failure period itself, and the page gives no test length [29]. The parties set a length with DCGM's `--iterations`, endless NCCL test cycles or perftest's fixed duration [1, 5, 6]. MegaScale's authors name the trade-off: longer diagnostics cost training time, and high false positive rates exclude machines that work [26]. SuperBench, a proactive validation system for cloud GPU infrastructure, balances validation time against issue-related penalties [23].

Published pass criteria are scoped to their own systems. DCGM prints plugin defaults and accepts parameters per GPU model [1], and the NCCL tests check their results [5]. Bus bandwidth is compared with the bottleneck link [4], and node pairs with each other [11]. Microsoft Azure's open-source health checks publish expected values for its own virtual machine types: for `nd96isr_h100_v5`, 8 GPUs, no inactive NVLinks, no Xid errors, 52 GB/s between GPU and host, 460 GB/s of NCCL all-reduce bandwidth and 380 GB/s of InfiniBand bandwidth with GPUDirect, and for `nd96isr_h200_v5`, 55, 480 and 400 GB/s [22]. Amazon's HyperPod short training run fails an instance whose training loss becomes NaN or infinite, rises above a threshold relative to its starting average, or misses its time limit [21].

Fail and retest rules follow from the same documents. Base Command Manager stops at the first failure and retries the failed phase [18], and Crusoe states that a node that fails a test does not enter its fleet [20]. A slow pair is retested against other nodes [11], an Xid 74 is checked at the far GPU [8], a remap waiting on a GPU reset needs that reset before the counters are read again [10], and a return goes through the field diagnostic and the system vendor [9, 11].

9. The acceptance record

For a GPU issue, NVIDIA's triage page tells users to give their system vendor the system configuration, key log messages, the debug steps taken, the full log, nvidia-bug-report.sh output, Fabric Manager logs on HGX systems and DCGM diagnostics logs [11]. On DGX systems, `nvsm dump health` writes a health report for support tickets [16]. A record that holds these from the start serves acceptance and any later claim.

Part of the record	Contents	Basis
Identity	Board serials, VBIOS, InfoROM, driver and Fabric Manager versions, GPU count, topology	[10, 12, 14]
Baseline counters	ECC aggregates, remapped rows and flags, PCIe replays, NVLink errors	[10]
Diagnostics	Level, iterations, JSON output, nvvs.log, per-test runtimes	[1, 3]
Load	Tool, duration or iterations, telemetry, clocks event reasons	[10, 12]
Bandwidth	NCCL tests tables with wrong-result counts and bus bandwidth, perftest results, node pairs	[4, 5, 6, 11]
Storage and host	fio jobs with verification, kernel AER log	[32, 33]
Errors and retests	Xids seen, actions taken, retest results	[8]

Part of the record	Contents	Basis
Support bundle	nvidia-bug-report.sh output, Fabric Manager and DCGM logs	[11]

The research cluster study defines lemon nodes as servers that cause repeated job failures but escape health checks and repair workflows [24]. Its detector found 40 faulty nodes across two clusters with more than 85% accuracy, and its observation 11 reads "Historic data is necessary to find defective nodes" [24]. Root causes were GPUs in 28.2% of cases, DIMMs in 20.5% and PCIe in 15.4%, and those clusters ran health checks every five minutes [24].

Under the model code, acceptance of a part of a commercial unit is acceptance of the whole unit [35], so terms that name the unit (a server or a rack) and keep a record per unit fit together. That is a point for counsel.

10. What published operators run

Published operator practice runs from the node outward, and two of the publishers end with a training workload [20, 21].

Publisher	Stages	Tools named	Pass values published
Crusoe	Up to 30 tests per node across node, GPU and fabric, in the order sanity, validation, performance, stress, then multi-node PyTorch jobs before handoff	DCGM level 2 and level 4, sysbench, NCCL over InfiniBand	None [20]
Amazon Web Services (SageMaker HyperPod)	Instance stress, connectivity across instances and multi-node training validation, during cluster creation and update	DCGM level 4 with extra memory tests, NCCL, EFA latency and bandwidth tests	Training run fail rules [21]
Microsoft Azure (open-source health checks)	Node checks per virtual machine type, an extension of LBNL Node Health Checks	NCCL all-reduce, InfiniBand with GPUDirect, host-device bandwidth	Expected values for its own types [22]
NVIDIA (DGX BasePOD guide)	Node-level NCCL on 8 GPUs, then 4 nodes and 32 GPUs	NCCL tests	None [17]

Publisher	Stages	Tools named	Pass values published
MegaScale paper	Intra-host loopback, all-to-all inside the node, all-reduce with top-of-rack neighbours	RDMA loopback, NCCL	None in the passages read [26]

Crusoe states that its tests are layered to isolate local hardware failures from fabric failures [20]. SuperBench reports raising the mean time between incidents by up to 22.61 times and validating hundreds of thousands of GPUs in Azure production over two years, and its authors note that redundancy can hide degraded components, which they call gray failure [23].

Implications

For buyers writing purchase terms. "Burn-in" needs a definition in the contract, because the diagnostic suite NVIDIA documents uses no such term [1]. Terms can name each stage, its tool and version, the DCGM level, the iterations or duration and the pass parameters for the specific GPU model [1]. Bandwidth targets come from the system's own specification and its first accepted baseline, since the NCCL tests and perftest set none [4, 6]. Terms can say who runs the field diagnostic, since NVIDIA routes it through the system vendor [11], and whether drives may be erased, since one documented burn default erases them [18]. Terms can also fix the place and method of inspection, which the model code presumes to be exclusive once the parties fix them [34].

For engineers running the tests. The published order isolates faults: counters and topology, single-node diagnostics, load with result checks, bandwidth inside the server, then the fabric and a workload [11, 20, 26]. HPL on B300 and GB300 systems is read against NVIDIA's own expectation of low FP64 results [7]. An NVLink error is read at both ends [8]. Tools that check results catch faults that logs miss [28, 36].

For both. The record outlasts the delivery. NVIDIA's triage guide treats recorded network results as the baseline for later comparisons [11], and the research cluster study found historic data necessary to find defective nodes [24].

Limits of this analysis

No hardware was tested for this note. DCGM's run times were measured by NVIDIA on Hopper systems and may differ on other generations [1]. Vendor documentation changes, and each date here is the one the page, its metadata or its server gives, or the day it was viewed. The DCGM user guide was last updated 17 June 2026, while its release notes, updated 12 November 2026, list version 4.8.0 [1, 3].

The operator pages describe their own fleets. The Crusoe and Amazon pages show no date, and the Azure values come from a repository file last changed 21 March 2025 [20, 21, 22]. The fleet studies cover A100 and H100 clusters, and the research cluster results hold for those two clusters [24, 27].

None of the sources gives a share of servers that fail burn-in. The field diagnostic itself is outside this note, since NVIDIA directs users to the system vendor for it [11]. The Uniform Commercial Code text is model text as published by the Legal Information Institute, contracts can change it, and nothing here is legal advice [34, 35].

Sources

1. DCGM Diagnostics. NVIDIA DCGM Documentation (User Guide). NVIDIA, last updated 17 June 2026, viewed 8 November 2026. https://docs.nvidia.com/datacenter/dcgm/latest/user-guide/dcgm-diagnostics.html

2. DCGM Multi-Node Diagnostics. NVIDIA DCGM Documentation. NVIDIA, last updated 17 June 2026, viewed 8 November 2026. https://docs.nvidia.com/datacenter/dcgm/latest/user-guide/dcgm-multinode-diagnostics.html

3. DCGM Release Notes. NVIDIA DCGM Documentation. NVIDIA, last updated 12 November 2026, viewed 13 November 2026. https://docs.nvidia.com/datacenter/dcgm/latest/release-notes/changelog.html

4. NCCL Tests: Performance reported (doc/PERFORMANCE.md). NVIDIA, GitHub, file last changed 25 July 2024, viewed 8 November 2026. https://github.com/NVIDIA/nccl-tests/blob/master/doc/PERFORMANCE.md

5. NCCL Tests (README). NVIDIA, GitHub, file last changed 19 August 2026, viewed 8 November 2026. https://github.com/NVIDIA/nccl-tests

6. perftest (README). linux-rdma project, GitHub, file last changed 1 September 2026, viewed 8 November 2026. https://github.com/linux-rdma/perftest

7. NVIDIA HPC-Benchmarks (container), version 26.02.01. NVIDIA NGC Catalog, repository updated 26 August 2026, viewed 8 November 2026. https://catalog.ngc.nvidia.com/orgs/nvidia/containers/hpc-benchmarks

8. Analyzing Xid Errors with the Xid Catalog. NVIDIA Xid Errors. NVIDIA, last updated 11 November 2026, viewed 13 November 2026. https://docs.nvidia.com/deploy/xid-errors/analyzing-xid-catalog.html

9. RMA Policy (RMA policy thresholds for row remapping). NVIDIA GPU Memory Error Management. NVIDIA, last updated 4 November 2026, viewed 8 November 2026. https://docs.nvidia.com/deploy/a100-gpu-mem-error-mgmt/rma-policy-thresholds-for-row-remapping.html

10. NVIDIA System Management Interface (nvidia-smi) documentation, covering nvidia-smi v620. NVIDIA, page modified 4 November 2026, viewed 8 November 2026. https://docs.nvidia.com/deploy/nvidia-smi/index.html

11. GPU Node Triage. NVIDIA GPU Debug Guidelines. NVIDIA, last updated 4 November 2026, viewed 8 November 2026. https://docs.nvidia.com/deploy/gpu-debug-guidelines/gpu-node-triage.html

12. Best Practices. NVIDIA GPU Debug Guidelines. NVIDIA, last updated 4 November 2026, viewed 8 November 2026. https://docs.nvidia.com/deploy/gpu-debug-guidelines/best-practices.html

13. nvbandwidth (README). NVIDIA, GitHub, file last changed 21 July 2026, viewed 8 November 2026. https://github.com/NVIDIA/nvbandwidth

14. Fabric Manager User Guide. NVIDIA, last updated 7 October 2026, viewed 8 November 2026. https://docs.nvidia.com/datacenter/tesla/fabric-manager-user-guide/index.html

15. Quickstart and Basic Operation. NVIDIA DGX B300 User Guide. NVIDIA, last updated 4 November 2026, viewed 8

November 2026. https://docs.nvidia.com/dgx/dgxb300-user-guide/quickstart-basics.html

16. Using the NVSM CLI. NVIDIA System Management User Guide. NVIDIA, last updated 13 July 2026, viewed 8 November 2026. https://docs.nvidia.com/datacenter/nvsm/nvsm-user-guide/latest/using-nvsm-cli.html

17. NVIDIA DGX BasePOD: Deployment Guide Featuring NVIDIA DGX H200/H100 Systems (DG11192005). NVIDIA, 16 June 2025, viewed 8 November 2026. https://docs.nvidia.com/dgx-basepod/deployment-guide-dgx-basepod/latest/_downloads/c632d64dbf70bf7b63ad34d5017b3bf3/DG11192005-Deployment-Guide.pdf

18. Burning Nodes. NVIDIA Base Command Manager 11 Installation Manual. NVIDIA, page modified 4 November 2026, viewed 8 November 2026. https://docs.nvidia.com/base-command-manager/product-manuals/installation/burn

19. gpu-burn (README). Ville Timonen, GitHub, last commit 2 October 2026, viewed 8 November 2026. https://github.com/wilicc/gpu-burn

20. Burn-in validation. Crusoe Cloud documentation. Crusoe, viewed 8 November 2026. https://docs.crusoe.ai/compute/virtual-machines/burn-in-validation/index.html

21. Deep health checks. Amazon SageMaker Developer Guide. Amazon Web Services, viewed 8 November 2026. https://docs.aws.amazon.com/sagemaker/latest/dg/sagemaker-hyperpod-eks-resiliency-deep-health-checks.html

22. AzureHPC Node Health Checks (README). Microsoft Azure, GitHub, file last changed 21 March 2025, viewed 8 November 2026. https://github.com/Azure/azurehpc-health-checks

23. SuperBench: Improving Cloud AI Infrastructure Reliability with Proactive Validation. Xiong and others, arXiv 2402.06194, version 2, 8 June 2024, viewed 8 November 2026. https://arxiv.org/abs/2402.06194

24. Revisiting Reliability in Large-Scale Machine Learning Research Clusters. Kokolis, Kuchnik and others, Meta, arXiv 2410.21680, version 2, 6 February 2025, viewed 8 November 2026. https://arxiv.org/abs/2410.21680

25. The Llama 3 Herd of Models. Llama Team, Meta, arXiv 2407.21783, version 3, 23 November 2024, viewed 8 November 2026. https://arxiv.org/abs/2407.21783

26. MegaScale: Scaling Large Language Model Training to More Than 10,000 GPUs. Jiang and others, arXiv 2402.15627, version 1, 23 February 2024, viewed 8 November 2026. https://arxiv.org/abs/2402.15627

27. Story of Two GPUs: Characterizing the Resilience of Hopper H100 and Ampere A100 GPUs. Cui and others, arXiv 2503.11901, version 4, 10 December 2025, viewed 8 November 2026. https://arxiv.org/abs/2503.11901

28. Silent Data Corruptions at Scale. Dixit and others, Facebook, Inc. (now Meta), arXiv 2102.11245, version 1, 22 February 2021, viewed 8 November 2026. https://arxiv.org/abs/2102.11245

29. 8.1.2.4. "Bathtub" curve. NIST/SEMATECH e-Handbook of Statistical Methods. National Institute of Standards and Technology, viewed 8 November 2026. https://www.itl.nist.gov/div898/handbook/apr/section1/apr124.htm

30. STREAM Benchmark Reference Information. John D. McCalpin, Department of Computer Science, University of Virginia, viewed 8 November 2026. https://www.cs.virginia.edu/stream/ref.html

31. HPL: A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers, version 2.3. Antoine Petitet, R. Clint Whaley, Jack Dongarra and Andy Cleary, Netlib, 2 December 2018, viewed 8 November 2026. https://www.netlib.org/benchmark/hpl/

32. fio: Flexible I/O tester documentation. fio project, Read the Docs, viewed 8 November 2026. https://fio.readthedocs.io/en/latest/fio_doc.html

33. The PCI Express Advanced Error Reporting Driver Guide HOWTO. The Linux Kernel documentation, kernel.org, viewed 8 November 2026. https://docs.kernel.org/PCI/pcieaer-howto.html

34. U.C.C. § 2-513. Buyer's Right to Inspection of Goods. Legal Information Institute, Cornell Law School, viewed 8 November 2026. https://www.law.cornell.edu/ucc/2/2-513

35. U.C.C. § 2-606. What Constitutes Acceptance of Goods. Legal Information Institute, Cornell Law School, viewed 8 November 2026. https://www.law.cornell.edu/ucc/2/2-606

36. GPU Burn. Ville Timonen, wili.cc, 5 January 2012, viewed 8 November 2026. http://wili.cc/blog/gpu-burn.html

About this report

Published by Rillor on 14 November 2026. Superintelligence (SI) tools assist research and drafting, and every figure cites its source. Rillor builds and secures SI systems, supplies the data they learn from and delivers the infrastructure they run on. We apply SI to financial and compute markets.

Cite as: Rillor. Burn-in and acceptance tests for GPU servers. 14 November 2026. https://rillor.com/insights/gpu-server-burn-in-and-acceptance-tests

Notices

Rillor is not a registered investment adviser or commodity trading advisor and does not provide investment or trading advice. Research, forecasts, data and software described on this site are for research and engineering use. Nothing here is an offer or recommendation to buy or sell any security, commodity interest or digital asset. Past or simulated results do not indicate future results.

NVIDIA, AMD, Ceph, Kubernetes and related product names are trademarks of their owners and are used only to identify products. No affiliation or endorsement is implied.

Work with Rillor

Send the workload, system, quantity, delivery window and destination. Contact us at rillor.com/contact?type=gpu-systems or contact@rillor.com. More on GPU systems: rillor.com/compute/gpu-systems