GPU SMITH
COMPUTE ENGINEERING

GPU SMITH / RESEARCH & PRACTICAL GUIDANCE

GPU Cluster Reliability: Spares, Failure Domains and SLAs

> 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 · 2026-09-19 · gpu cluster reliability · gpu failure domains · spare capacity · reliability engineering · repair sla · mttf · mttr · effective training time · ai infrastructure

Original article: https://gpusmith.com/articles/en/gpu-cluster-reliability-budget

In this article

Executive Summary

GPU cluster reliability is not one percentage. It is a chain from a component event, through detection and node quarantine, to workload interruption, repair, validation, and return to service. The most useful public study covers **11 months**, **two research environments**, **4 million jobs**, and more than **150 million A100 GPU-hours** (ai.meta.com). That is substantial evidence, but it is evidence about those environments. It is not a universal GPU failure rate. Its taxonomy also cautions that symptoms can map to several domains (arxiv.org). A provider-authored architecture discussion likewise describes reliability as four interdependent operational layers (wf.coreweave.com). Buyers should therefore require a local event ledger with explicit denominators, attribution confidence, timestamps, topology, workload exposure, and repair outcomes.

The operating budget should separate **component availability**, **schedulable capacity**, **job interruption**, **unavailable GPU-hours**, and **Effective Training Time Ratio (ETTR)**. ETTR is productive runtime divided by available wall-clock time (arxiv.org). Availability alone misses lost work after a checkpoint. Job success alone gives equal weight to a brief single-GPU attempt and a long distributed run. Mean time to repair hides the difference between detection, isolation, physical replacement, validation, and queue delay. NIST's definition of reliability itself is conditional on stated conditions and a specified period (csrc.nist.gov). AWS separately defines strict availability to include scheduled and unscheduled interruption (docs.aws.amazon.com), showing why the eligible-time rule must be stated.

Published results demonstrate why workload size matters. In the Meta study, the reported MTTF for a **1,024-GPU job was 7.9 hours** (arxiv.org), while more than **90% of jobs** were smaller than one eight-GPU server (arxiv.org). The lesson is not to import either number. It is to calculate each workload class against the buyer's own exposure and to model independence only as a declared scenario. Microsoft's reliability guidance notes that multiplying component service levels assumes independent failures, which rarely holds (learn.microsoft.com).

Spare capacity should consequently be a policy output, not a fixed percentage. The worksheet in this report sizes hot spares, quarantined capacity, replacement pools, and optional overflow against observed demand, repair-time distributions, topology, and a service objective. Formal recovery guidance also calls for enough spare replica capacity to absorb failover demand (docs.aws.amazon.com). Contracts should define clocks and evidence, including incident start and end with time zones (learn.microsoft.com) and logs sufficient to corroborate an outage (aws.amazon.com). A sound SLA governs measurement, evidence, replacement, validation, exclusions, and remedy. It does not promise that one headline uptime figure will protect useful training time.

Introduction and Background

A GPU fleet can look healthy in inventory while delivering poor training continuity. A device may answer telemetry but fail a collective operation. A node may be powered yet drained by the scheduler. A job may restart successfully but lose hours of post-checkpoint work. A repaired server may remain unavailable until it passes validation and rejoins a compatible topology. These are different states, owned by different teams and measured with different clocks. Even a request-based availability measure needs an explicit success or failure status for each unit (sre.google).

This report treats reliability as a **buyer-operated reliability budget**. The objective is to connect fleet events to useful workload output without inventing a universal hardware rate. NIST defines mean time to recovery as the average time needed to recover from a product or system failure (csrc.nist.gov). For a GPU platform, that average is meaningful only when the starting event, restored state, population, and observation window are explicit.

The intended users are enterprise fleet operators, reliability engineers, machine learning platform owners, procurement teams, and infrastructure investors. The method begins after handoff and acceptance. It governs recurring detection, quarantine, repair, spares, and service-level evidence. GPU Smith's documented scope includes capacity planning, telemetry, failure-mode analysis, and operating procedures (gpusmith.com). That adjacent engineering perspective is relevant to defining the evidence package, but GPU Smith is not treated as a cloud or hardware provider.

Three principles govern the analysis:

- **State the unit.** A GPU event, node incident, job attempt, logical job, and unavailable GPU-hour are not interchangeable.
- **State the boundary.** Compute, host, interconnect, storage, facility, software, and user-code domains need separate fields even when attribution remains unknown.
- **State the assumption.** Independent exponential failures, immediate failover, and fixed repair times are modeling cases, not facts about a fleet.

Key Changes

Change 1: Replace headline uptime with a metric dictionary

An availability percentage is useful only after defining its eligible time and unavailable state. AWS's strict availability framing includes scheduled and unscheduled interruption (docs.aws.amazon.com). A contract may exclude planned maintenance, while an internal capacity budget should still count its effect. Both figures can coexist, but they must not share an unlabeled numerator.

Google's Site Reliability Engineering guidance defines a service level indicator (SLI) as a quantitative measure of delivered service (sre.google) and a service level objective (SLO) as the target or range for that service level (sre.google). A GPU platform should use several SLIs rather than force every effect into uptime.

Table 1 distinguishes the core measures and, critically, what each one cannot establish.

Metric	Definition and denominator	Decision supported	What it cannot prove
Component event rate	Confirmed events divided by device-hours, node-hours, or port-hours, stated separately.	Replacement forecasting and domain trend analysis.	Job impact or root cause from a symptom alone. NVIDIA notes that an Xid value may not establish the issue by itself (docs.nvidia.com).
Job interruption rate	Infrastructure-attributed interrupted attempts divided by eligible attempts, with workload class.	Scheduler and workload reliability.	Lost work magnitude, because attempts differ in size and duration.

Metric	Definition and denominator	Decision supported	What it cannot prove
MTTF	Total exposed operating time divided by observed failures for a defined population, or a fitted survival-model result.	Comparative exposure modeling within the same population.	A universal device constant, correlated events, or censored units unless modeled.
MTTR	Mean elapsed time between a declared start and defined recovery state.	Staffing, spares, and supplier response planning.	Which portion was detection, logistics, repair, validation, or queueing. AWS separates detection from actual repair or mitigation (docs.aws.amazon.com).
Availability	Eligible time less unavailable time, divided by eligible time. NASA's inherent form is MTTF divided by MTTF plus MTTR (standards.nasa.gov).	Capacity and service-level tracking.	Productive work, job success, or user waiting time.
Unavailable GPU-hours	Sum of affected GPU count multiplied by each unavailable interval.	Economic impact and reserve demand.	Whether GPUs are interchangeable across topology or configuration.
ETTR	Productive training runtime divided by available wall-clock time for a logical run.	Checkpoint, retry, and restart-policy optimization.	General service availability or workload quality.

The table shows why one executive dashboard needs linked measures. For request-like services, Google defines availability as successful responses divided by requests (sre.google). A batch training platform instead needs an explicit success state for each unit of work, plus GPU-hours and productive runtime. Otherwise, short jobs dominate the count while long jobs dominate the cost.

Change 2: Make the event ledger the system of record

The event ledger should join scheduler, hardware, network, storage, facility, and service-desk records under one incident identifier. ISO 14224 is formally scoped to process-industry equipment, not GPU clusters, but it defines a standard reliability and maintenance data format (iso.org). Its data categories provide a useful schema pattern: maintenance action, resources, consequence, and downtime (iso.org). The GPU-specific ledger should record:

- **Identity:** incident ID, asset serial, GPU, host, rack, fabric plane, power domain, and service owner.
- **Exposure:** affected GPUs and nodes, workload ID, requested topology, start time, and accumulated device-hours.
- **Clock sequence:** first symptom, detection, acknowledgement, quarantine, workload restart, hands-on repair, validation start, validation pass, and return to schedule.
- **Impact:** interrupted attempts, unavailable GPU-hours, queue delay, checkpoint age, recompute time, and productive time lost.
- **Attribution:** observed symptom, suspected domain, confirming test, root-cause confidence, and unknown or ambiguous status.
- **Disposition:** no-fault-found, reset, software change, cable or part replacement, vendor escalation, and return-to-service result.

- **Governance:** evidence links, owner, change record, review date, and SLA eligibility decision.

Unknown attribution is a valid value. Meta's paper reports the most likely cause using heuristics (arxiv.org). Treating such an estimate as confirmed root cause would create false precision. Likewise, NIST calls a dataset censored when the observation period ends before every unit fails (www.itl.nist.gov). Its nonparametric guidance still requires exact failure times (itl.nist.gov). Retired, transferred, and still-running assets need censoring status rather than silent deletion.

Change 3: Treat failure domains as an evidence graph

A visible GPU error can originate in the device, host, link, fabric, storage path, power train, cooling system, system software, or application. NVIDIA explicitly notes that its diagnostic suite exercises memory, compute, PCIe, NVLink, power, thermal, and collective paths (docs.nvidia.com). A provider architecture account similarly frames distributed-training reliability across four interdependent layers (wf.coreweave.com). A diagnostic result narrows the graph. It does not make every neighboring dependency healthy.

Table 2 maps symptoms to evidence and disposition without pretending that symptoms are causes.

Domain	Example symptom	Confirming evidence	Confidence and disposition
GPU or HBM	Xid, double-bit error, repeated reset, or diagnostic memory failure.	Time-aligned device telemetry, offline test, firmware and driver state, and repeat on a controlled image.	Record confirmed, probable, or unknown. DCGM does not repair faults or determine RMA eligibility (docs.nvidia.com).
Host or PCIe	Device disappearance, link downgrade, or host reset.	Baseboard logs, PCIe counters, power state, kernel log, and slot-swap test.	Quarantine the node until the boundary is isolated.
NVLink or NVSwitch	Collective timeout or degraded peer path.	Topology map plus interval-based link counters. A cumulative total alone cannot time an event (docs.nvidia.com).	Compare a pre-event baseline, event window, and post-repair validation.
Fabric	Collective stall, packet loss, or route flap.	Switch, adapter, cable, port, queue, and routing telemetry correlated to job rank.	Identify whether the blast radius is port, leaf, spine, rail, or rack.
Storage	Checkpoint timeout, mount loss, or read latency.	Client and server logs, namespace health, metadata path, and throughput trace.	Separate checkpoint failure from compute failure and include lost-work age.
Power or cooling	Throttling, simultaneous node loss, or environmental alarm.	Rack PDU, branch circuit, inlet temperature, coolant, controller, and facility alarms.	Model as a shared domain rather than independent devices. NASA defines common-cause failure as several failures with one origin (ntrs.nasa.gov).
System software	Driver hang, scheduler drain, image regression, or health-check failure.	Version, configuration, deployment wave, canary outcome, and reproducible test.	Link to change records and preserve rollback evidence.

Domain	Example symptom	Confirming evidence	Confidence and disposition
User code	Deterministic exception, out-of-memory condition, or invalid collective use.	Application logs, input, environment, and reproducer.	Exclude from infrastructure SLI only under an explicit rule, but retain the attempt in workload accounting.

The evidence graph also defines the blast radius. Microsoft describes an availability zone as having distinct power, network, and cooling (learn.microsoft.com) and elsewhere calls each zone's power, cooling, and networking infrastructure independent (learn.microsoft.com). Within a private facility, equivalent boundaries may be a rack PDU, coolant distribution unit, top-of-rack switch, fabric plane, or storage namespace. The buyer should name these boundaries before calculating redundancy.

Change 4: Model workload exposure, not just fleet events

Under a deliberately simplified independent, constant-hazard scenario, a job using `g` identical exposure units for `t` hours has survival probability `S = exp(-lambda × g × t)`. Its interruption probability is `1 - S`. The formula is useful for sensitivity, but only after the buyer supplies `lambda`, chooses the exposure unit, and tests whether independence and constant hazard are plausible.

The Meta model reflects the same scale pressure in its definition of job-level MTTF through node count and per-node failure rate (arxiv.org). Yet its workload was heterogeneous: more than 90% of jobs used less than one server. An earlier primary study separately characterized a two-month trace from a shared Microsoft training cluster (usenix.org). These bounded studies reinforce that the correct planning distribution is the buyer's mix of small, medium, and large jobs, not the single largest configuration.

For each workload class, collect:

- **Requested scale:** GPUs, nodes, racks, fabric rails, and storage paths.
- **Duration distribution:** wall-clock runtime, not only the mean.
- **Checkpoint policy:** interval, write duration, retention, and successful restore evidence.
- **Retry path:** detection delay, scheduler delay, reallocation delay, restore time, and catch-up work.
- **Completion definition:** attempt success, logical-job completion, deadline compliance, and acceptable output.

AWS defines checkpointing as saving state periodically so work can resume from the last saved point after interruption (docs.aws.amazon.com). That makes checkpoint age an economic variable. The ETTR model classifies catching up from the last checkpoint as unproductive time (arxiv.org). A reliable restart can therefore coexist with poor effective training time.

Implementation Considerations and Process Changes

Quarantine, diagnose, validate, return

Reliability operations need a state machine that prevents a suspect node from cycling silently back into production:

1. **Detect:** bind the symptom to synchronized telemetry and an incident ID.
2. **Contain:** stop new work and capture affected jobs without erasing evidence.
3. **Quarantine:** mark the node unavailable to scheduling. Kubernetes marks a node unschedulable to prevent new pod placement (kubernetes.io) and automatically taints nodes under defined conditions (kubernetes.io). Slurm can run a health check that drains a node (slurm.schedmd.com).
4. **Diagnose:** test the narrowest suspected domain, then adjacent dependencies.
5. **Repair or mitigate:** record reset, reconfiguration, component replacement, or workload reroute separately.
6. **Validate:** run the agreed short test, then a longer administrator-led validation when risk warrants it (docs.nvidia.com).
7. **Return:** restore scheduling eligibility only after evidence is attached and configuration drift is checked.

The repair SLA should name each transition. A promise to replace hardware within a period does not specify detection time, logistics start, validation duration, or time until the scheduler can place a topology-compatible workload. Slurm's maintenance guidance explicitly says to set nodes to draining before suspension (slurm.schedmd.com), giving the ledger a schedulable-state timestamp distinct from power state.

Spare capacity as a portfolio

No defensible universal spare percentage exists. Reserve demand depends on concurrent quarantine, failure correlation, job placement, parts lead time, and the service objective. The reserve portfolio can combine:

- **Hot schedulable spares:** compatible nodes powered, validated, and ready for placement.
- **Warm local spares:** installed capacity requiring a short commissioning or image step.
- **Cold parts:** on-site field-replaceable inventory that still incurs repair and validation time.
- **Supplier pool:** contractual replacement stock with a defined request, dispatch, delivery, and acceptance clock.
- **Cloud overflow:** prequalified capacity with tested data, network, security, image, and checkpoint portability.
- **Deferred capacity:** low-priority work that can be preempted to protect a higher-priority SLO.

AWS notes that spare subsystems can reduce experienced recovery time almost to zero when work can be rerouted (docs.aws.amazon.com). That does not mean the failed asset is repaired. It means the service can recover before repair. Conversely, reserved cloud capacity carries cost even when idle, as AWS documents for On-Demand Capacity Reservations (docs.aws.amazon.com).

The planning method should use an empirical distribution or simulation. Multiplying nominal service levels is inadequate because it assumes independent failure, an assumption Microsoft says rarely holds (learn.microsoft.com). The simulation should include:

- **Independent case:** resample observed device or node events and repair intervals.
- **Correlated case:** inject rack, rail, fabric, storage, cooling, and maintenance-domain outages separately.
- **Topology case:** count only spares compatible with the workload's placement and interconnect constraints.

- **Service case:** select the reserve level that meets the chosen percentile of unmet demand or deadline breach.
- **Stress case:** lengthen supplier and validation delays and report the marginal reserve required.

Contract schedule and audit evidence

Table 3 turns the reliability budget into a procurement and operating schedule.

Control block	Buyer-supplied inputs	Computed or auditable output	Contract evidence
Population and window	Asset IDs, topology, commissioning and retirement dates, observation start and end.	Device-hours, node-hours, censored units, and eligible service time.	Serialized inventory, as-built topology, and change history.
Event demand	Ledger events, affected assets, confidence, correlation group, and exclusions.	Event rate by domain, concurrent quarantine distribution, and unknown share.	Raw telemetry, scheduler records, ticket ID, and preserved logs.
Repair path	Detection, acknowledgement, quarantine, dispatch, arrival, repair, validation, and return times.	Stage percentiles, total MTTR, backlog, and unavailable GPU-hours.	Timestamp source and time zone. Microsoft specifically recommends incident start and end with time zones (learn.microsoft.com).
Workload exposure	GPU count, duration, priority, deadline, checkpoint interval, restore and catch-up time.	Interruption probability by declared model, expected lost work, and ETTR distribution.	Job attempts linked into one logical run and checkpoint restore proof.
Reserve policy	Compatible hot, warm, cold, supplier, overflow, and deferrable capacity.	Unmet-demand percentile, reserve utilization, cost, and correlated stress result.	Monthly capacity snapshot and failover exercise.
SLA remedy	Eligible services, exclusions, threshold, measurement window, claim deadline, and remedy.	Pass or fail with a reproducible calculation.	Customer and provider logs. Amazon requires logs that corroborate the claimed outage (aws.amazon.com).
Repeat-event control	Organization-defined threshold, exposure basis, sample size, statistical method, and owner.	Watchlist, confidence interval, action, and scheduled review.	Diagnostic history and disposition, without copying another fleet's threshold.

The table makes a crucial distinction: a provider may satisfy a replacement clock while the buyer misses its workload objective. The SLA schedule should therefore cover both service evidence and operational handoffs. It should also state which clock pauses for buyer access, security clearance, maintenance windows, parts approval, or missing logs.

Repeat-event and quarterly review workflow

A “lemon node” label should be the result of a declared repeat-event rule, not an intuition. The rule should specify event class, minimum exposure, lookback period, treatment of no-fault-found returns, and statistical review. A node with three events in light exposure may warrant more attention than one with the same count over much greater exposure, but the threshold must be local.

Every quarter, the owner should issue:

- **Exposure pack:** active assets, additions, removals, device-hours, node-hours, and censoring changes.
- **Event pack:** counts and rates by domain, confidence, workload class, and unknown attribution.
- **Impact pack:** interruptions, unavailable GPU-hours, deadline misses, lost work, and ETTR.
- **Repair pack:** median and tail times for detection, quarantine, dispatch, repair, validation, and return.
- **Repeat-event pack:** watchlist, statistical basis, sample size, action, and outcome.
- **Capacity pack:** reserve inventory, quarantine demand, overflow tests, and correlated stress scenarios.
- **Change pack:** firmware, driver, scheduler, fabric, storage, facility, and operating-procedure changes.

The review should preserve denominators when the fleet changes. Kaplan-Meier estimation can accommodate censored observations without assuming a lifetime distribution, but it still requires exact failure times (nist.gov). If those timestamps are absent, the report should say so rather than manufacture precision.

Data Analysis and Evidence

The Meta study is the richest public reference for this framework, but its numbers must retain their scope. Its official summary records 4 million jobs and more than 150 million A100 GPU-hours (ai.meta.com). It analyzed two multi-tenant research clusters over 11 months, not every GPU generation, provider, workload, or operating model. In one scheduler-status analysis, infrastructure failures affected **0.2% of jobs** (arxiv.org) yet touched **18.7% of runtime** (arxiv.org). This contrast is exactly why job count cannot stand in for economic impact.

The study also reported that similar-priority hardware events co-occurred in **3% of RSC-1** and **5% of RSC-2** hardware failures (arxiv.org). Those observations do not define another fleet’s correlation. They do show that an independence-only reserve model can omit real multi-asset demand.

Its ETTR examples illustrate sensitivity to checkpoint policy. In a hypothetical dedicated **16,000-GPU** setting, the paper estimated ETTR at **0.7** with a **60-minute** checkpoint interval (arxiv.org) and **0.93** with a **five-minute** interval (arxiv.org). Those are model outputs under the paper’s scenario, not promises for other clusters. Its simplified approximation is specifically framed for long-running, high-priority jobs with small queue time (arxiv.org).

Other research reinforces the need to preserve denominators and censoring. The Titan lifetime study covered more than **100,000 cumulative GPU-years** over a **six-year** production period (impact.ornl.gov). A later survival model retained **15,099 censored observations**, alongside **1,127 off-the-bus** and **3,093 double-bit-error** events in one cohort (arxiv.org). These studies address different hardware, time periods, and questions. Pooling their headline rates would destroy the population definitions that make them useful.

Reproducible worksheet

For a buyer's own data, the calculation sequence is:

1. **Choose the population:** one GPU model, node design, software baseline, and observation window per stratum.
2. **Compute exposure:** sum active device-hours or node-hours, preserving commissioned, removed, and censored units.
3. **Classify events:** confirmed, probable, unknown, planned maintenance, user-caused, and shared-domain event.
4. **Estimate event demand:** use counts divided by the matching exposure, plus confidence intervals or a survival model where appropriate.
5. **Fit repair demand:** retain the full detection-to-return distribution and each intermediate stage.
6. **Join workloads:** map each event to affected GPU count, duration, checkpoint age, retry, and deadline outcome.
7. **Simulate reserves:** draw independent events and separate correlated scenarios against compatible spare pools.
8. **Report sensitivity:** vary job size, duration, checkpoint interval, detection delay, repair tail, and reserve policy.

For a labeled scenario input, suppose the buyer supplies a fitted per-node hazard `lambda_n`. An `n`-node job of duration `t` has independent-case exposure `n × t` and modeled survival `exp(-lambda_n × n × t)`. Doubling either nodes or duration doubles the exponent, not necessarily the observed interruption rate. The buyer should run a second case with rack and fabric events sampled as shared shocks. The gap between the cases is a decision-relevant sensitivity, not an error bar to hide.

Case Studies and Real-World Examples

Meta research clusters: method, not benchmark

The Meta study offers four transferable practices: a workload-aware taxonomy, explicit heuristic attribution, job-scale exposure, and ETTR. It also demonstrates that the largest jobs and the most numerous jobs are different populations. Its extrapolated model projected a **1.8-hour MTTF for a 16,384-GPU job** (arxiv.org). That projection is valuable for testing architectural sensitivity, but it should not be entered into a private-cluster contract as an observed universal value.

A buyer can reproduce the method without copying the rate: fit local event or survival models, stratify by node and topology, map attempts into logical runs, and calculate productive time. The result will be auditable because each numerator and denominator points back to the event ledger.

Titan: survival analysis needs the non-events

The Titan work shows why reliability analysis must retain units that have not failed. Right-censored assets contribute exposure and constrain survival estimates. Deleting them leaves only eventful devices and biases the story toward failure. This is especially important for mixed-age fleets, phased expansions, and assets

transferred between service pools.

The practical implication is simple: inventory history belongs in the reliability dataset. Commission date, active intervals, configuration, retirement date, and reason for exit should be joined to incidents. The study's scale, more than 100,000 cumulative GPU-years across six production years, supports the survival-analysis method rather than transfer of its rates (impact.ornl.gov). The resulting cohort can be analyzed without implying that Titan's device behavior applies to current private AI clusters.

Operational control planes: quarantine is measurable

Modern schedulers expose a practical boundary between "powered" and "available." Kubernetes supports condition-driven node taints (kubernetes.io), while Slurm advises draining nodes before maintenance suspension (slurm.schedmd.com). Google separately documents proactive node replacement in a managed clustered-GPU service (docs.cloud.google.com). These controls make quarantine time directly measurable. They also allow the capacity budget to count a drained node even when a hardware inventory system reports it as online.

Implications and Future Directions

The next improvement in GPU reliability reporting is likely to be **state linkage**, not another global rate. Device telemetry, topology, scheduler state, checkpoints, service tickets, parts logistics, and validation results need a common time model and incident key. Interval baselines matter: NVIDIA notes that topology output does not prove current link activity or application performance (docs.nvidia.com).

Providers are also exposing more managed reliability mechanisms. Google documents automated hardware health monitoring and proactive node replacement for AI Hypercomputer (docs.cloud.google.com). It also states that GPU-attached instances cannot be live migrated for host maintenance (docs.cloud.google.com) and that attached Local SSD data is unrecoverable when such an instance stops (docs.cloud.google.com). Such features and constraints should be evaluated through evidence fields: what signal triggers replacement, which timestamp starts the clock, what capacity is quarantined, how the workload resumes, and what logs the customer receives.

Architecture will continue to complicate arithmetic. Google separates GPU communication traffic from storage and system paths in its documented networking design (docs.cloud.google.com). Azure documents NVLink 4.0 within ND H100 v5 virtual machines (learn.microsoft.com). NVIDIA cautions that topology data alone does not prove every link is active or predict application performance (docs.nvidia.com). Buyers should represent these as distinct paths and failure domains, not as a generic "network."

Procurement should demand machine-readable ledgers and audit rights before the first quarterly review. GPU Smith states that its engagements are delivered against written acceptance criteria and an as-built documentation set (gpusmith.com). Ongoing reliability extends that discipline: preserve the accepted baseline, record each change, and make service evidence exportable. The result is a budget that can be recalculated as workload mix, topology, repair lead time, or business priority changes.

Frequently Asked Questions (FAQs)

What is a good GPU cluster availability target?

There is no context-free target. Define the eligible service, measurement window, maintenance treatment, capacity unit, and workload objective first. An internal target may count all lost capacity while a contract excludes approved maintenance. Report both without blending them.

How should GPU cluster spare capacity be calculated?

Use the local distribution of concurrent quarantines and repair durations, add explicit correlated rack and fabric scenarios, then test the result against the selected SLO. Count only topology-compatible capacity. AWS reliability guidance similarly says replicas need sufficient spare capacity to absorb failover demand (docs.aws.amazon.com). The required amount remains fleet-specific.

Is MTTF the same as availability?

No. MTTF describes time to failure for a defined population and model. Availability also depends on recovery or repair time. Neither measures checkpoint loss, queueing, or useful training time by itself.

What should a GPU repair SLA measure?

It should timestamp detection, acknowledgement, quarantine, provider acceptance, dispatch, arrival, repair, validation, and return to scheduling. It should define evidence sources, time zones, exclusions, claim procedure, and remedy. It should also distinguish service recovery through a spare from physical repair of the failed asset.

How should repeated node failures be handled?

Create a local rule based on event class, exposure, lookback period, sample size, and statistical review. Preserve unknown and no-fault-found outcomes. Do not copy another operator’s count threshold without matching its fleet, workload, and observation method.

Can cloud availability be copied into an on-premises model?

No. Service boundaries differ. For example, Google states that GPU-attached Compute Engine instances cannot be live migrated during host maintenance (docs.cloud.google.com). Microsoft defines availability zones through separate failure-domain infrastructure (learn.microsoft.com). A private cluster has different maintenance, spares, topology, and staffing. Use provider documentation to define that service, not to supply a private fleet’s rate.

Conclusion

GPU cluster reliability becomes actionable when every metric is tied to a unit, boundary, observation window, and workload consequence. This follows the core definition of reliability as functioning under stated conditions for a specified period (csrc.nist.gov). A component event is not automatically a node incident. A quarantined

node is not schedulable capacity. A successful retry is not zero lost work. A fast service recovery through a spare is not the same as a completed repair.

The recommended operating system is an evidence chain: serialized inventory and topology, a timestamped event ledger, confidence-labeled attribution, explicit quarantine and validation states, workload exposure, repair-stage distributions, and a reserve simulation that includes correlated domains. The SLA should mirror that chain and give the buyer access to the records needed to reproduce a result.

Published cluster studies are most valuable as methodological templates. Their rates remain attached to their hardware, software, workloads, and observation windows. A buyer that preserves this boundary can use public research without turning it into a false benchmark. The resulting reliability budget can answer the decision that matters: how much compatible capacity, repair capability, and evidence are required to protect the organization’s own useful training time.

Source links

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

1. https://ai.meta.com/research/publications/revisiting-reliability-in-large-scale-machine-learning-research-clusters/
2. https://arxiv.org/html/2410.21680v1
3. https://wf.coreweave.com/blog/what-a-reference-architecture-for-distributed-ai-training-actually-looks-like
4. https://csrc.nist.gov/glossary/term/reliability
5. https://docs.aws.amazon.com/wellarchitected/latest/reliability-pillar/availability.html
6. https://learn.microsoft.com/en-us/azure/reliability/concept-service-level-agreements
7. https://docs.aws.amazon.com/r53recovery/latest/dg/readiness-what-is.html
8. https://aws.amazon.com/compute/sla/
9. https://sre.google/workbook/implementing-slos/
10. https://csrc.nist.gov/glossary/term/mean_time_to_recovery
11. https://gpusmith.com/
12. https://sre.google/sre-book/service-level-objectives/
13. https://docs.nvidia.com/deploy/xid-errors/introduction.html
14. https://docs.aws.amazon.com/whitepapers/latest/availability-and-beyond-improving-resilience/reducing-mttr.html
15. https://standards.nasa.gov/sites/default/files/standards/NASA/A/0/nasa-std-87291a.pdf
16. https://www.iso.org/standard/64076.html
17. https://www.itl.nist.gov/div898/handbook/apr/section1/apr131.htm
18. https://itl.nist.gov/div898/handbook/apr/section2/apr215.htm
19. https://docs.nvidia.com/datacenter/dcgm/latest/learn/modules/dcgm-diagnostics.html
20. https://docs.nvidia.com/datacenter/dcgm/latest/learn/core-services/topology-and-links.html
21. https://ntrs.nasa.gov/api/citations/20160005837/downloads/20160005837.pdf
22. https://learn.microsoft.com/en-us/azure/virtual-machines/availability
23. https://learn.microsoft.com/en-us/azure/reliability/availability-zones-overview

24. https://www.usenix.org/conference/atc19/presentation/jeon
25. https://docs.aws.amazon.com/eks/latest/best-practices/aiml-compute.html
26. https://kubernetes.io/docs/concepts/architecture/nodes/
27. https://kubernetes.io/docs/concepts/scheduling-eviction/taint-and-toleration/
28. https://slurm.schedmd.com/slurm.conf.html
29. https://gpusmith.com/articles/en/gpu-cluster-acceptance-testing-runbook
30. https://slurm.schedmd.com/scontrol.html
31. https://gpusmith.com/articles/en/choose-gpu-cloud-provider-ai-training
32. https://gpusmith.com/articles/en/infiniband-vs-spectrum-x-vs-roce-vs-ethernet-ai-clusters
33. https://impact.ornl.gov/en/publications/gpu-lifetimes-on-titan-supercomputer-survival-analysis-and-reliab/
34. https://arxiv.org/html/2303.16369
35. https://docs.cloud.google.com/ai-hypercomputer/docs/overview
36. https://docs.cloud.google.com/compute/docs/gpus/gpu-host-maintenance
37. https://docs.cloud.google.com/ai-hypercomputer/docs/networking-overview
38. https://learn.microsoft.com/en-us/azure/virtual-machines/sizes/gpu-accelerated/ndh100v5-series

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