TECHNICAL PAPERS

GPU Faults Across Cloud Providers: Characterization and Operational Insights at Scale

Kenneth Sebastian, Devendra Ayalasomayajula, Annu Jolly, Rashi Motwani, Abby Kuang,
Giorgio Di Guardia, Aman Miryala, Sivakumar Arayandi Thottakara, Ranjitha Prasanna, Satvir Kooner

NVIDIA



SC26 hpc unites | Chicago

GPU faults are routine at scale

466

job interruptions

in 54 days of Llama 3 training
on 16K H100 GPUs

Llama 3 [Meta, 2024]

...but each study sees one context at a time

- One operator's ML clusters Kokolis et al., HPCA '25
- One system's A100 and H100 GPUs Cui et al., 2025
- One supercomputer's V100s Oles et al., ICS '24

Open questions

1. What fails, and does it vary across providers?
2. Is it fixed, and how fast on bare metal vs. VM?
3. Does it come back after repair?

OUR SYSTEM

How we manage GPU faults today

In service

Node in the Slurm pool, running jobs

Detect & drain

Telemetry, a health check or a failed job flags the node; it leaves the scheduler

Classify

1

1 of 26 drain categories

Remediate

2

Automated **83%**: reboot or power cycle, then health check

Engineer **17%**

back in service

3

Same Slurm stack and automated triage system (REMEDI8) on every cluster in this study.
Fleet: ~1B GPU-hours per year, 18 clusters, 7 cloud providers.

Terminate & replace

persistent defects, repeat offenders

1 **What fails?**

2 **Is it fixed, and how fast?**

3 **Does it come back?**

OUR SYSTEM

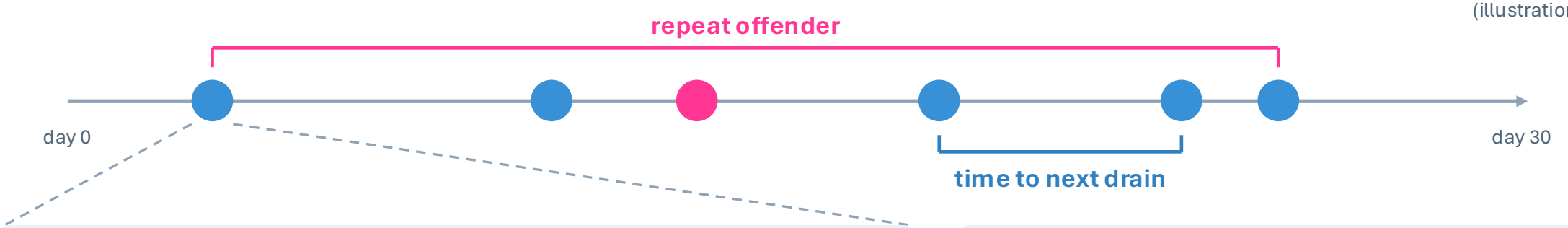
Three ways we catch a faulty node



Whichever path fires, the node is drained and triage starts

Five key terms, on one node's 30-day timeline

(illustration)



one drain, zoomed in



Drain event

Node pulled from the scheduler when a fault is detected: one node, one fault, one triage cycle

ATSR: automated triage success rate

drains resolved with no human ÷ all drains

- resolved automatically
- needed an engineer

Time to next drain: gap between a node's consecutive drains

Repeat offender: more than 4 drains in any rolling 30-day window

THE STUDY

Five production clusters, chosen for controlled comparisons

Production workloads
Operationally mature clusters running live training jobs

≥98% node availability
Sustained for 3+ months, with median repair under 35 min

Controlled comparison
Same Slurm stack and triage pipeline on every cluster

	A100	H100	B200
Bare metal direct hardware	**Cluster-A · Cluster-D** Provider-1	**Cluster-E** Provider-1	–
		same GPU: bare metal vs. VM	
VM GPUs via a hypervisor	–	**Cluster-B** Provider-2	**Cluster-C** Provider-3

3,990 nodes · 17,421 drain events · January–March 2026

Caveats: bare metal runs only on Provider-1, so provider and deployment model co-vary (patterns, not provider rankings). The clusters are mature by design (selection bias), Cluster-E is small (59 nodes), and we measure node drains, not job outcomes.

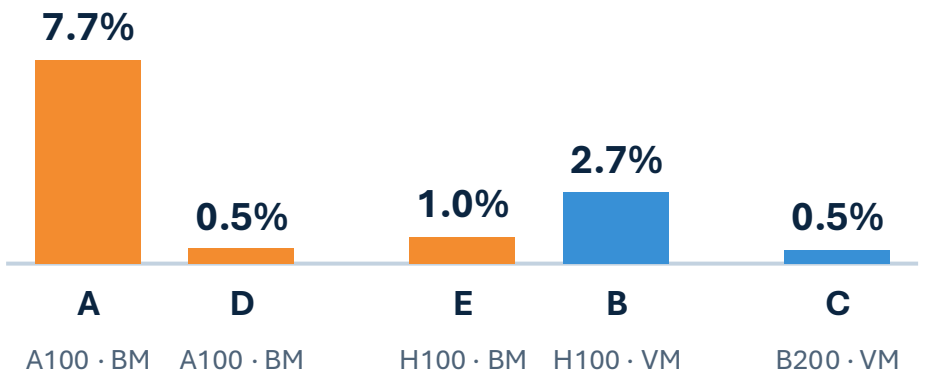
What we found

1 What fails? 2 Is it fixed, and how fast? 3 Does it come back?

Same GPU, different faults: each cluster has its own profile

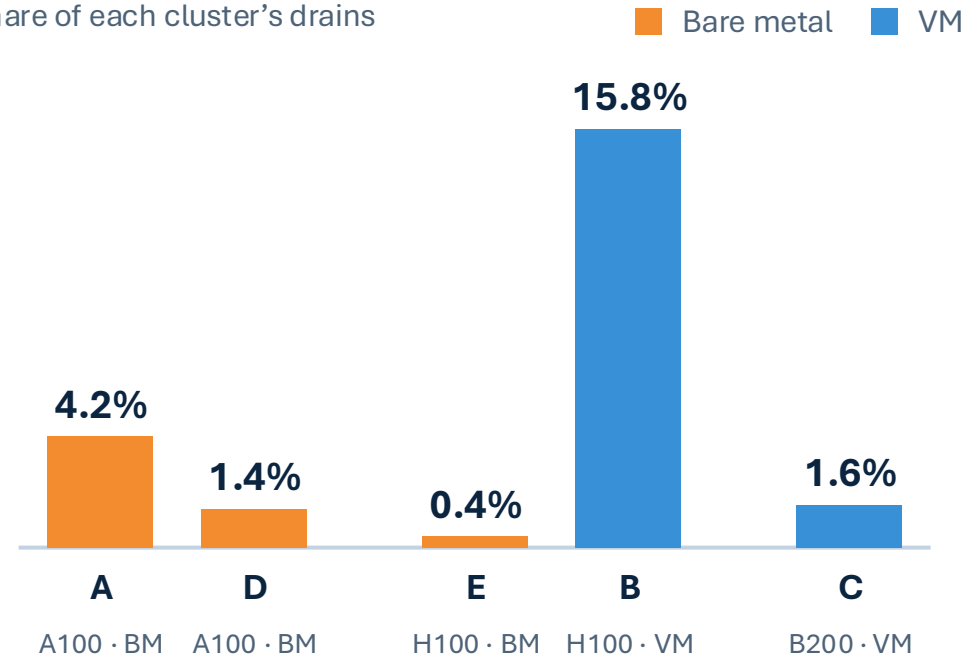
GPU fell off the bus (Xid 79)

Share of each cluster's drains



GPU memory (ECC) errors

Share of each cluster's drains

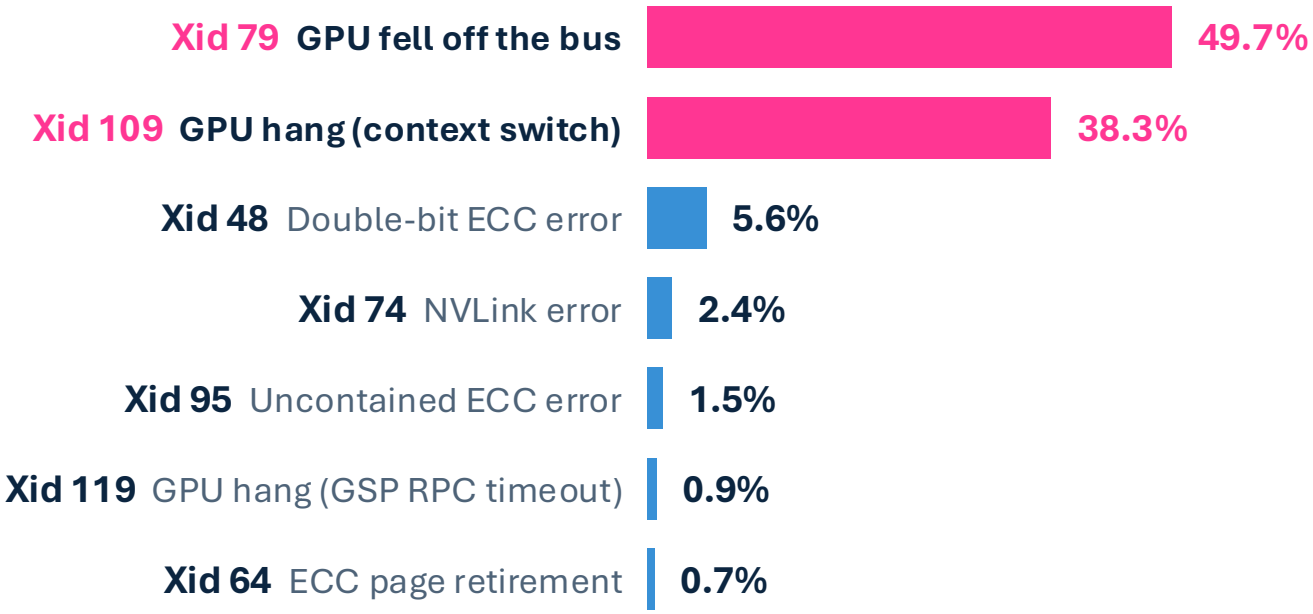


→ Cover each cluster's fault types before you automate

QUESTION 1 · WHAT FAILS?

Most GPU driver errors: the GPU fell off the bus, or it hung

Share of the 1,380 drains with a GPU driver error code (Xid)



88%

are Xid 79 or 109: the GPU fell off the bus, or it hung and needed a reboot

+1,264

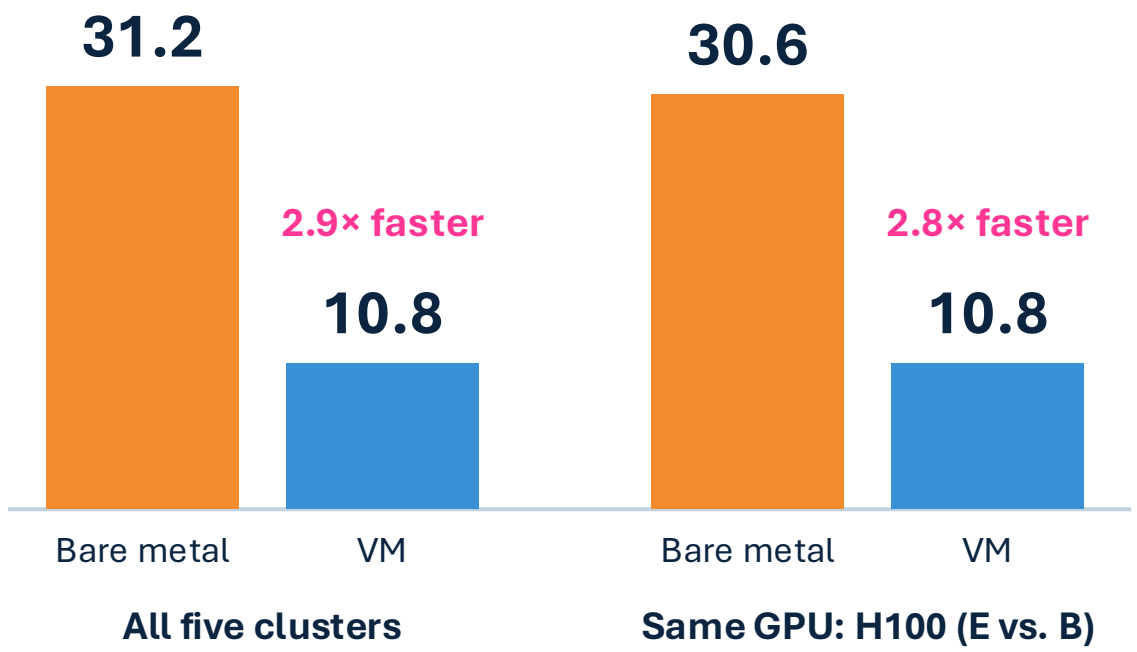
ECC drains caught by health checks, counted separately

→ Use field data to pick which GPU faults to model and inject

Application Xids excluded; other codes: 0.8%.

Repairs take minutes; VMs are 3× faster, even on the same GPU

Median time to repair (minutes)



→ Repair speed is part of the bare-metal vs. VM trade-off

WHY VMs REPAIR FASTER

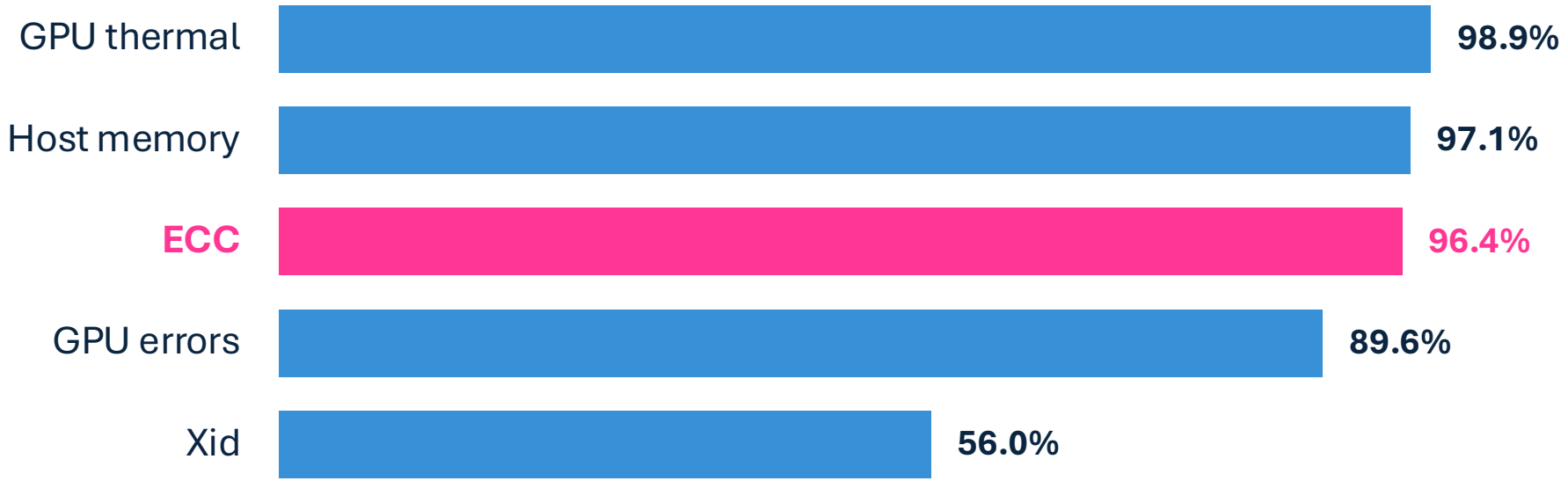
VM: typically a hypervisor warm reset, which re-attaches the GPUs without power-cycling the host.

Bare metal: a full power cycle re-enumerates PCIe and re-initializes the GPUs, NVLink and network fabric.

Automation rate is similar: ATSR 80% (bare metal) vs. 86% (VM)

Mann–Whitney U test, effect size r = 0.41. All bare-metal clusters run on Provider-1 and Cluster-E is small (59 nodes): a pattern, not a causal effect.

Automation resolves 82% of drains, and ECC looks solved

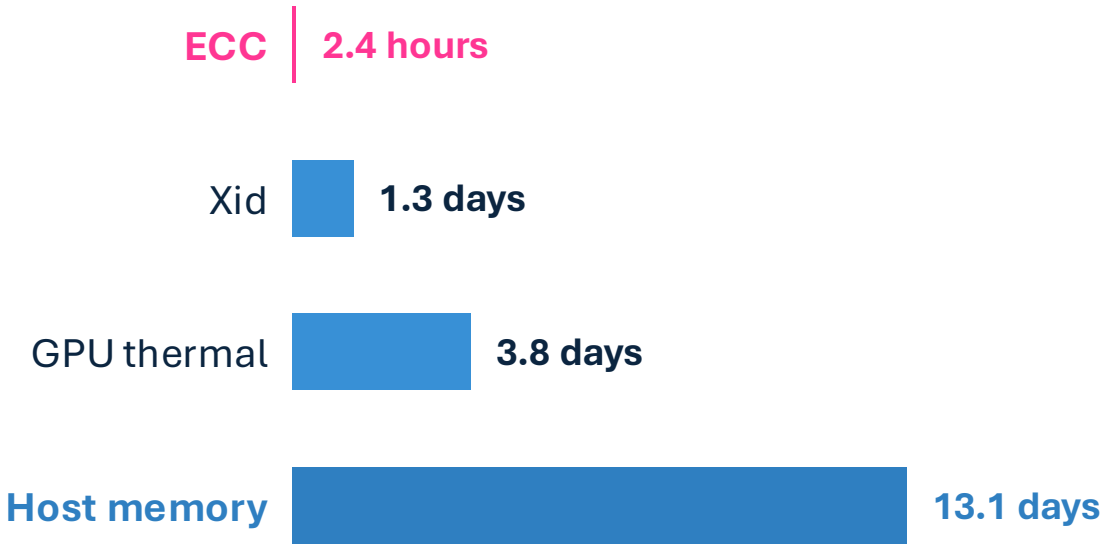


Xid = GPU driver error code, mostly the GPU falling off the bus or hanging. GPU errors = other GPU hardware faults.

But does “resolved” mean healthy?

ECC looks solved, but nodes drain again within hours

Median time to next drain, by fault type



→ Remediate by fault type: a reboot fixes host memory, not ECC

2.4 h

after an ECC drain, the median time to that node's next drain (ECC auto-resolves 96%)

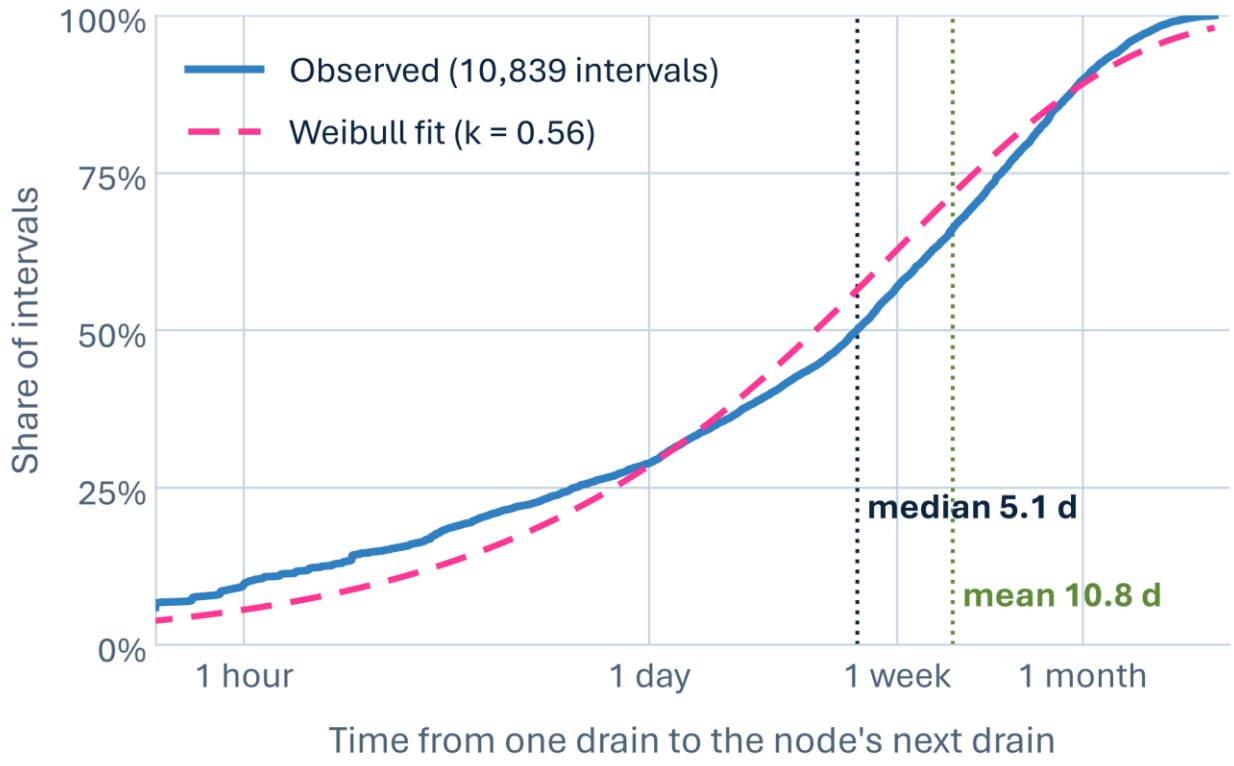
3 in 4

of those next drains are ECC again, a median of 42 minutes later

13.1 days

for host-memory errors: here a reboot appears to work

Nodes are most failure-prone right after repair



→ **Watch nodes most closely right after repair**

5 of 5

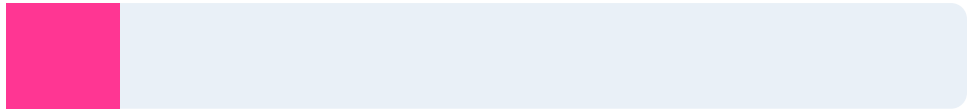
clusters show risk falling with time since repair: Weibull shape k = 0.43–0.69, all below 1, as in HPC systems (Schroeder & Gibson)

63.5%

of nodes that drain more than once drain again within 7 days

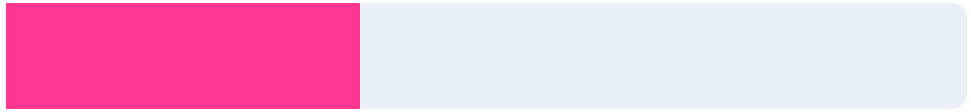
The same nodes keep failing: 12% of nodes cause 37% of drains

Nodes



469 repeat-offender nodes: 11.8% of all nodes

Drains

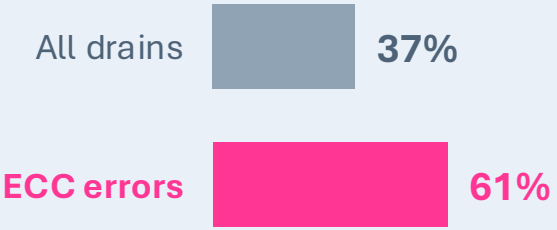


...generate 36.9% of all drain events

Repeat offender: more than 4 drains in any rolling 30-day window, tracked by hardware serial number.

WHERE REPEAT OFFENDERS SHOW UP

Share of drains on repeat-offender nodes



WHY QUARANTINE

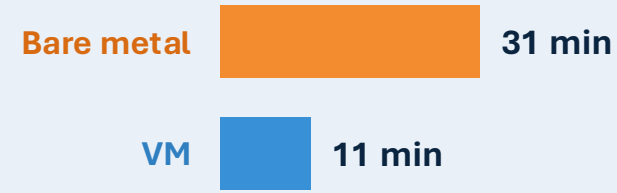
Each drain past the threshold costs triage time and node availability

→ Quarantine at the threshold to stop futile repair cycles

VMs repair 3× faster; automation and recurrence barely differ

Median time to repair

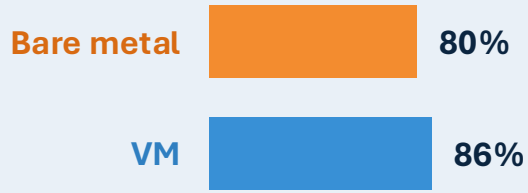
minutes



2.9× faster on VMs

Resolved automatically

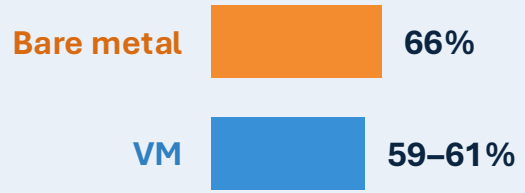
share of drains



About the same

Drain again within 7 days

repeat-drain nodes, larger clusters



Modest gap

→ **Repair speed varies with deployment; recurrence needs quarantine either way**

FOR RESILIENCE MODELS

Three failure-model assumptions that don’t hold in our fleet

ASSUMPTION

“Failures are memoryless”

OUR FLEET

k = 0.56

Weibull shape below 1 on all clusters: risk peaks after repair

INSTEAD

Model risk that falls with time since repair

ASSUMPTION

“The mean time between failures describes a node”

OUR FLEET

2.1×

Typical gap 5.1 days vs. mean 10.8 days between a node’s drains

INSTEAD

Plan checkpoints with the median, not the mean

ASSUMPTION

“All nodes are alike”

OUR FLEET

12% → 37%

12% of nodes (the repeat offenders) cause 37% of drains

INSTEAD

Account for which nodes a job gets, not just how many

Per-node results: a job’s interruption rate also depends on its size and on whether faults are recovered transparently.

TAKEAWAYS

The biggest threat: the same node failing again

Question	What we found	What to do
1 **What fails?**	Each cluster has its own fault profile; no clear GPU or deployment pattern	**Cover each cluster's fault types, then tune automation per cluster**
2 **Is it fixed?**	82% resolved automatically; VMs repair about 3× faster than bare metal	**Treat "resolved" as provisional; weigh the repair-time trade-off**
3 **Does it come back?**	Often, soon after repair, and on the same nodes: 12% cause 37%	**Quarantine by category, watch the post-repair window, recalibrate monthly**

Next: job-level impact, predictive quarantine, rack-level correlation

THANK YOU

Questions?

GPU Faults Across Cloud Providers: Characterization and Operational Insights at Scale

Artifact: doi.org/10.5281/zenodo.22104097



SC26 hpc unites | Chicago