



CONFERENCE

—NORTH AMERICA 2026—

Where's My Training Job?

End to end training job observability with Ray and Datadog

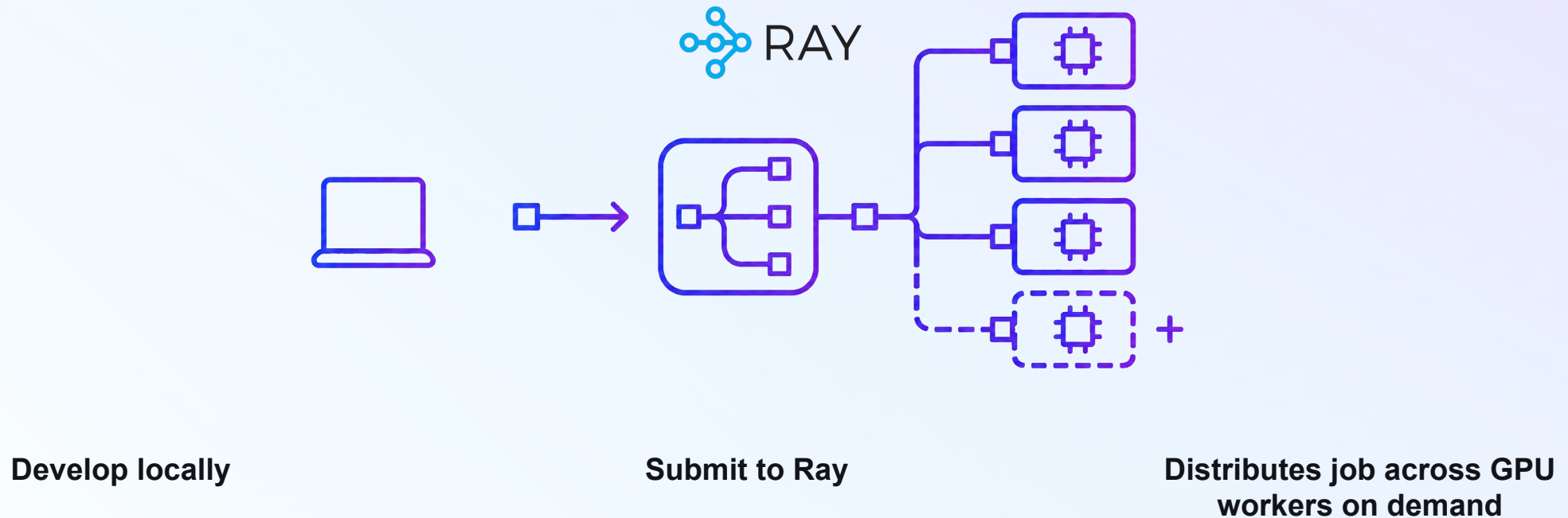
Ray takes training from one machine to an elastic cluster



Develop locally

Scaling solves the capacity problem, but creates a visibility problem.

Ray takes training from one machine to an elastic cluster



Scaling solves the capacity problem, but creates a visibility problem.

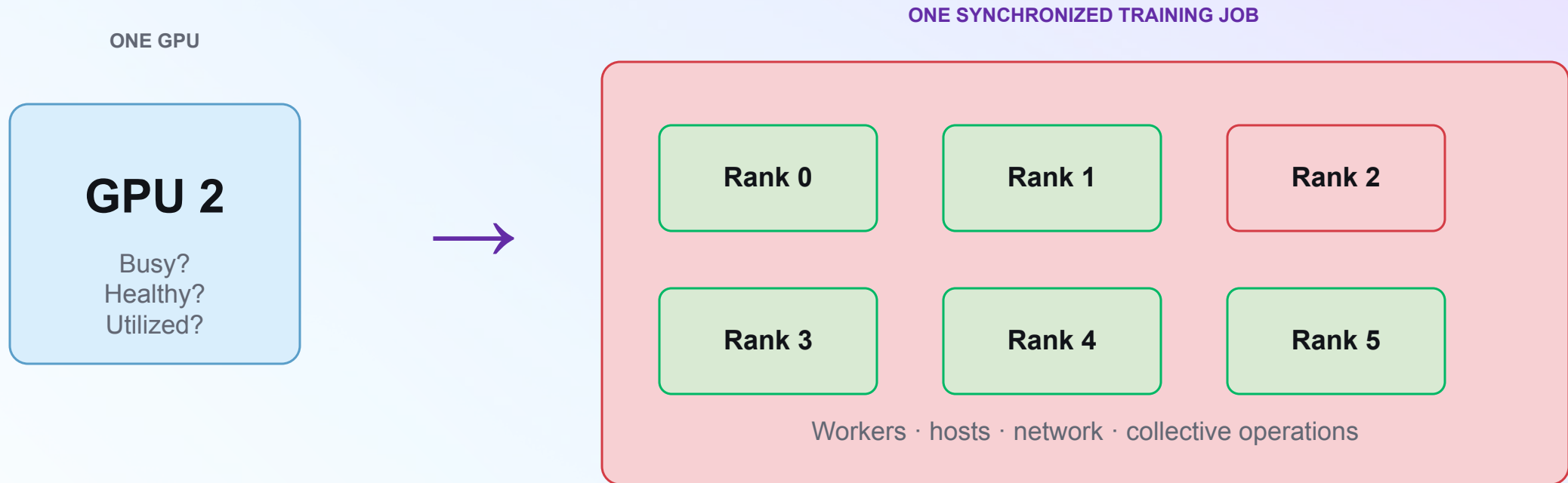
At cluster scale, the training job becomes the unit of performance

ONE GPU

GPU 2

Busy?
Healthy?
Utilized?

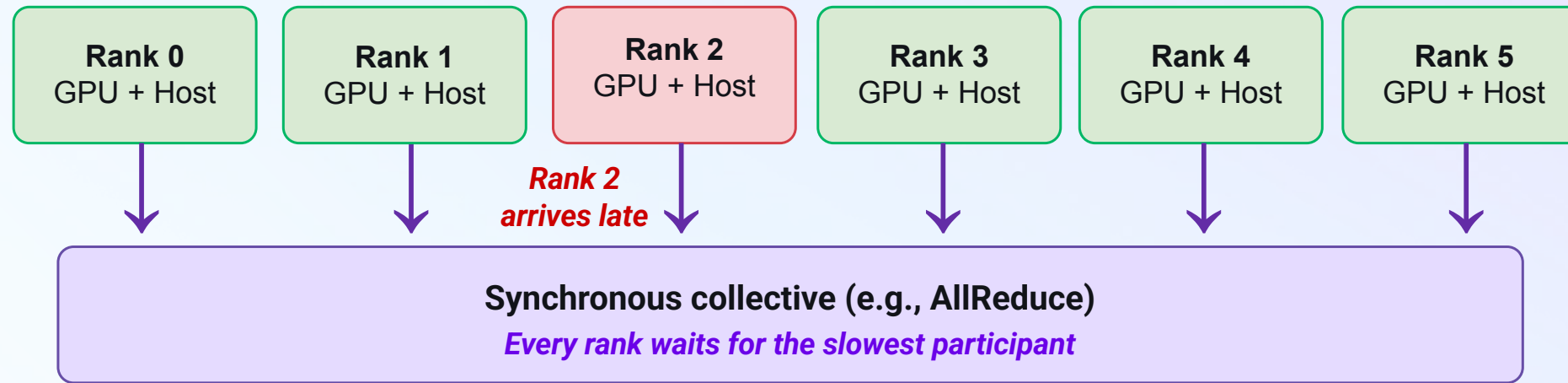
At cluster scale, the training job becomes the unit of performance



A healthy GPU does not necessarily mean a healthy training job.

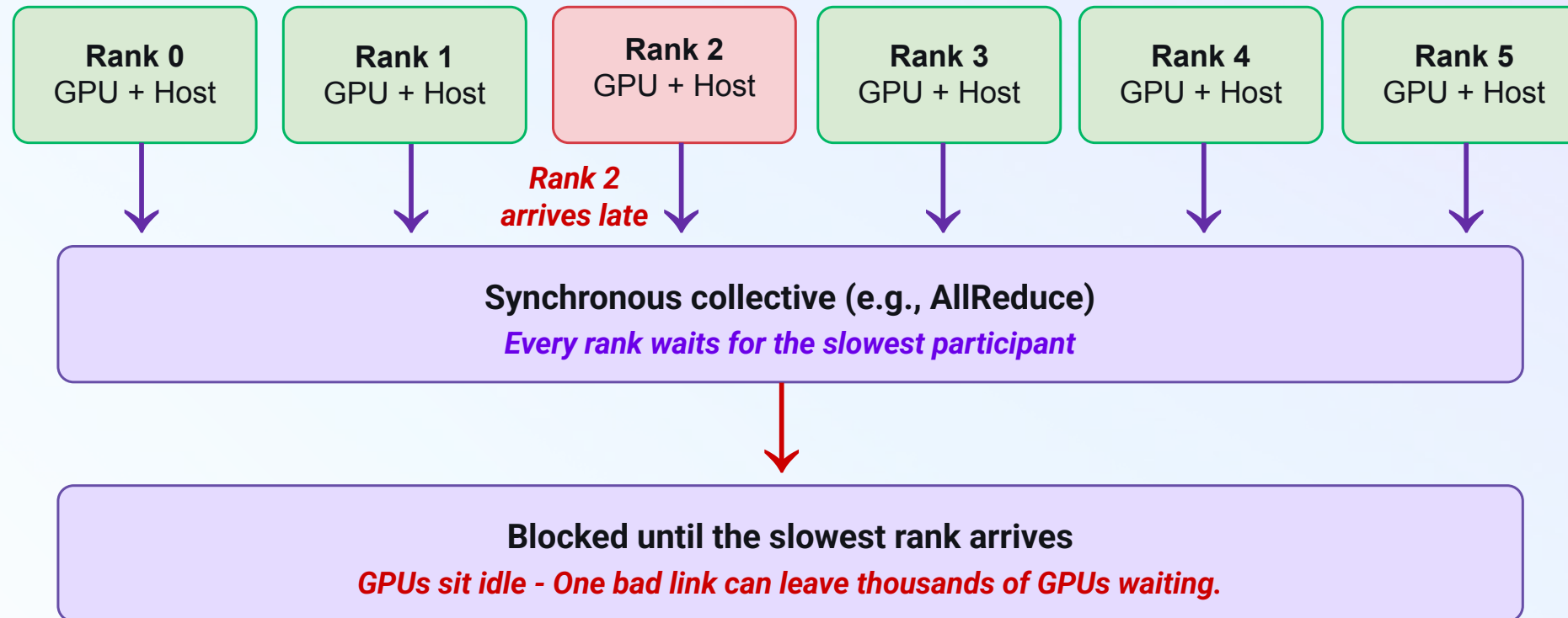
One slow worker stalls every healthy GPU

Training is synchronized, not merely distributed - Hive mind!



One slow worker stalls every healthy GPU

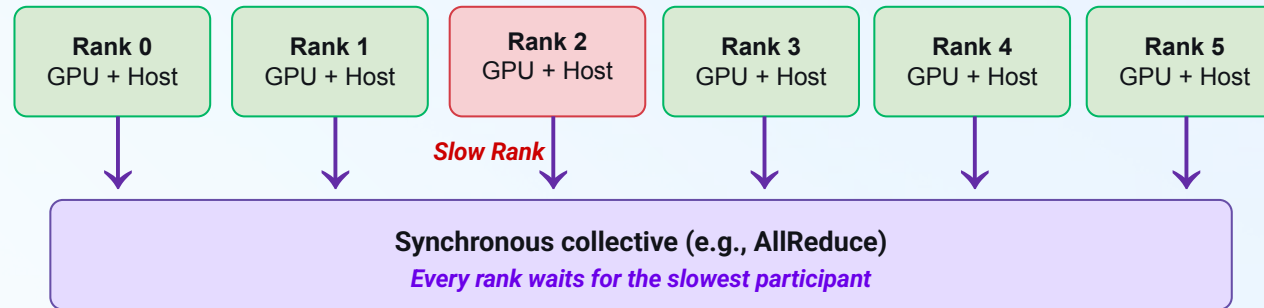
Training is synchronized, not merely distributed - Hive mind!



The entire job advances at pace of its slowest rank

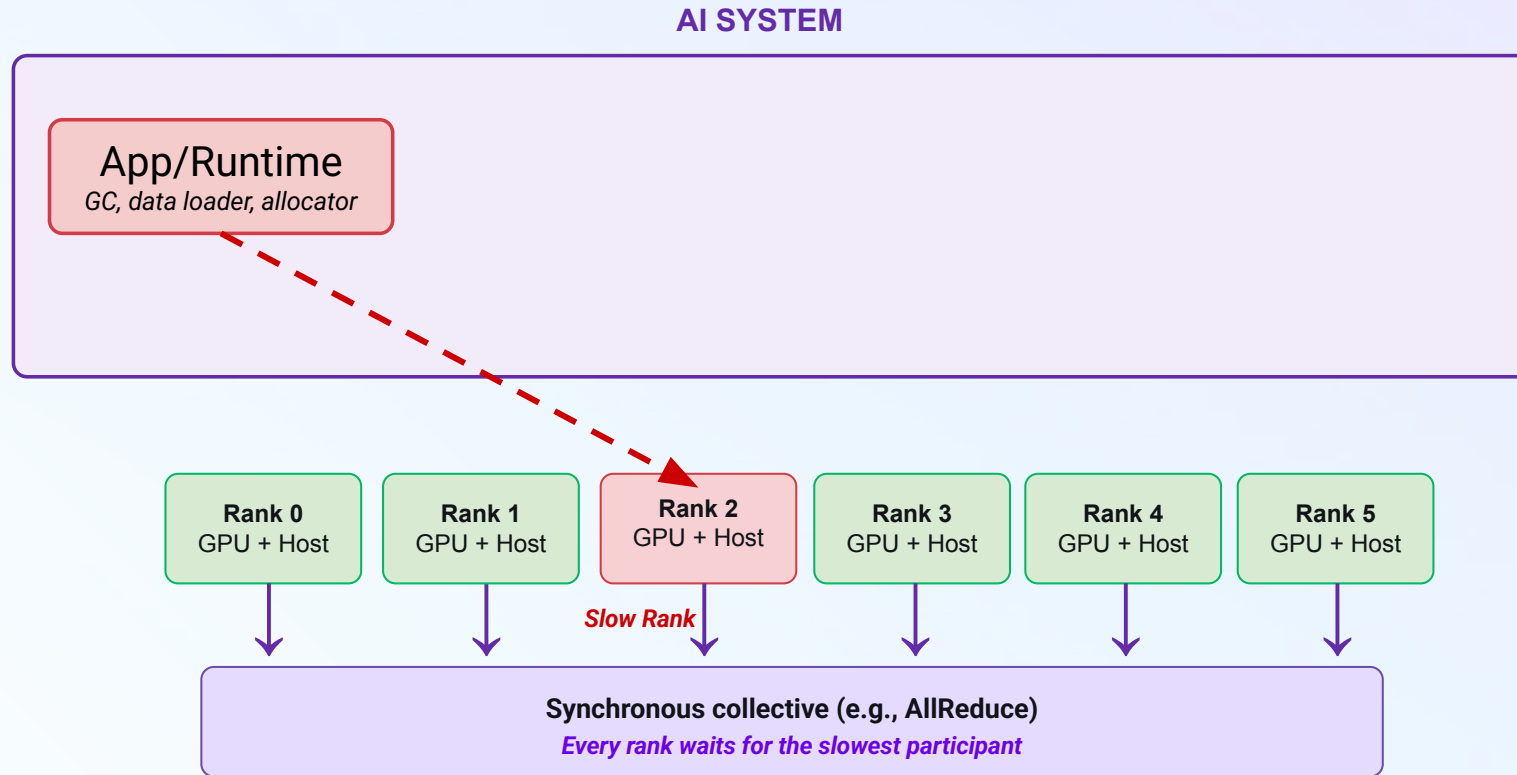
The visibly slow GPU may be innocent

The delay appears at one rank—but can begin elsewhere in the system.



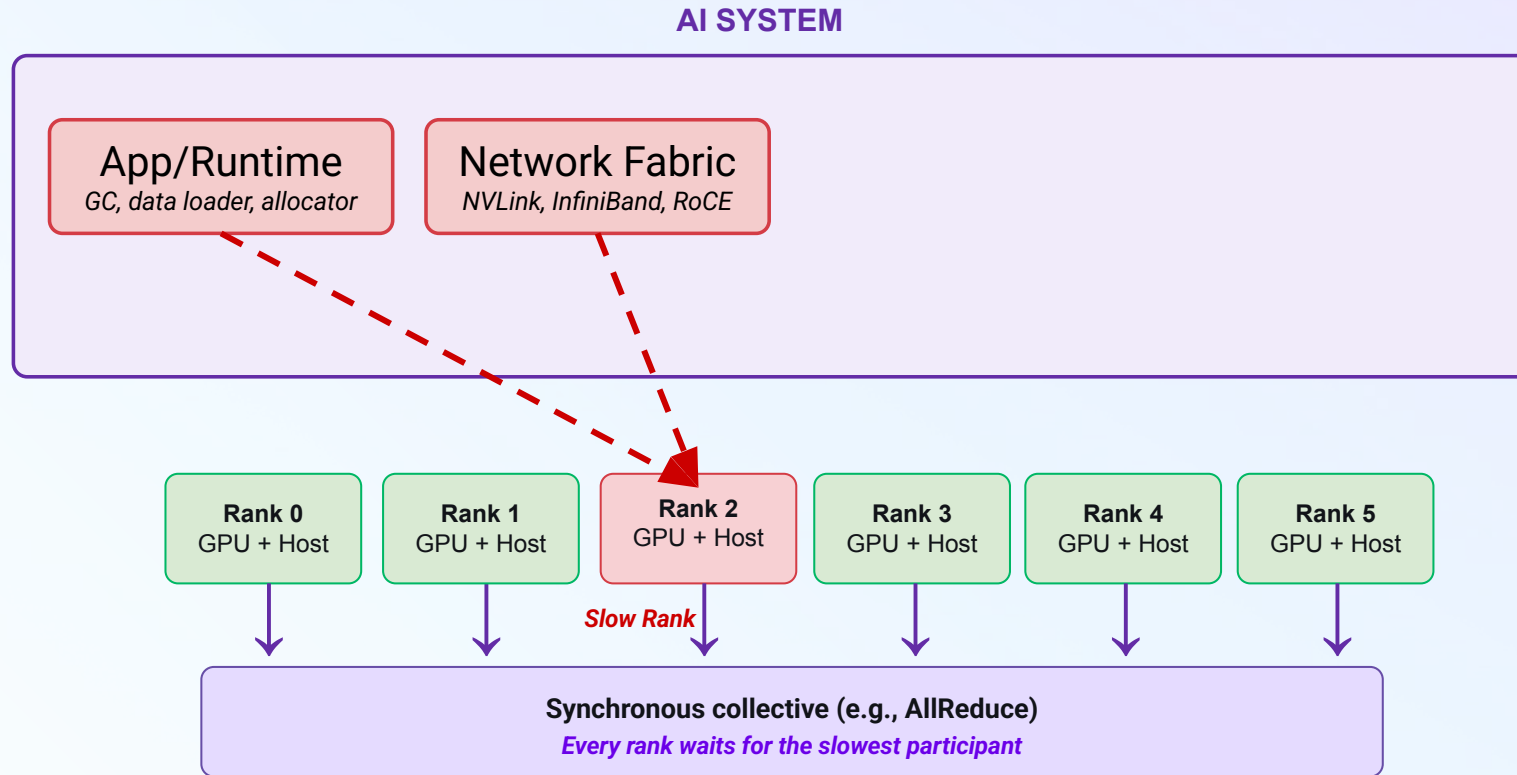
The visibly slow GPU may be innocent

The delay appears at one rank—but can begin elsewhere in the system.



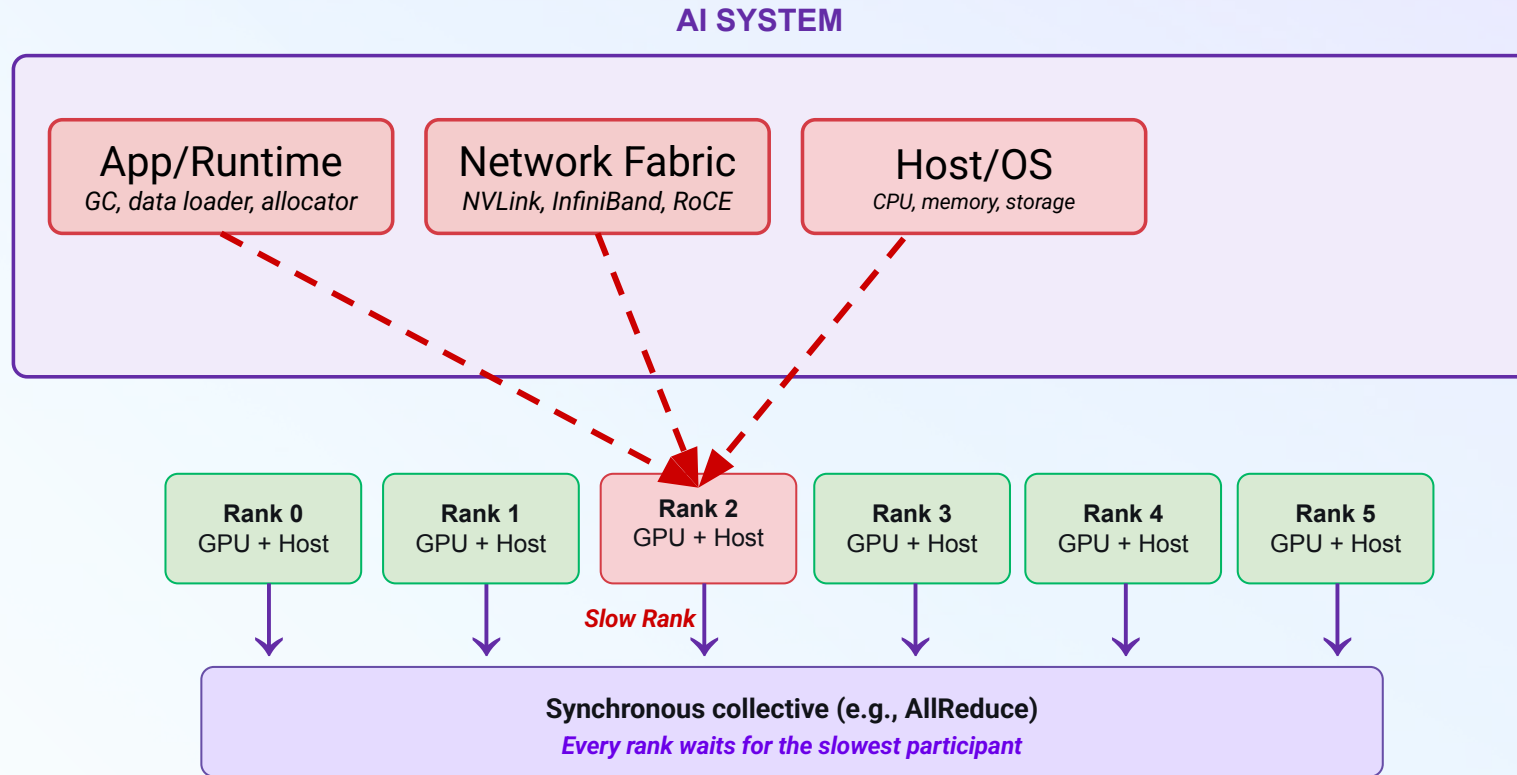
The visibly slow GPU may be innocent

The delay appears at one rank—but can begin elsewhere in the system.



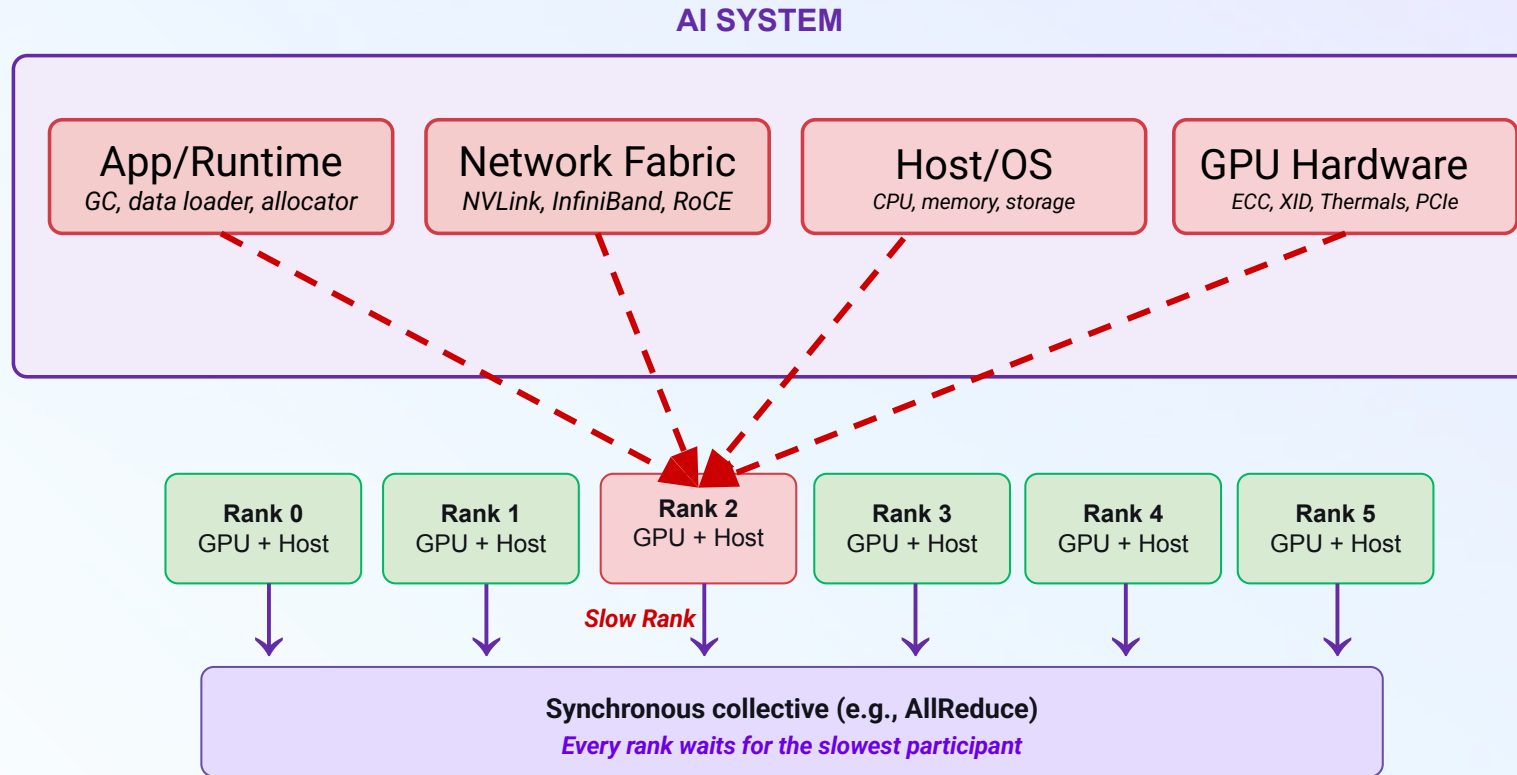
The visibly slow GPU may be innocent

The delay appears at one rank—but can begin elsewhere in the system.



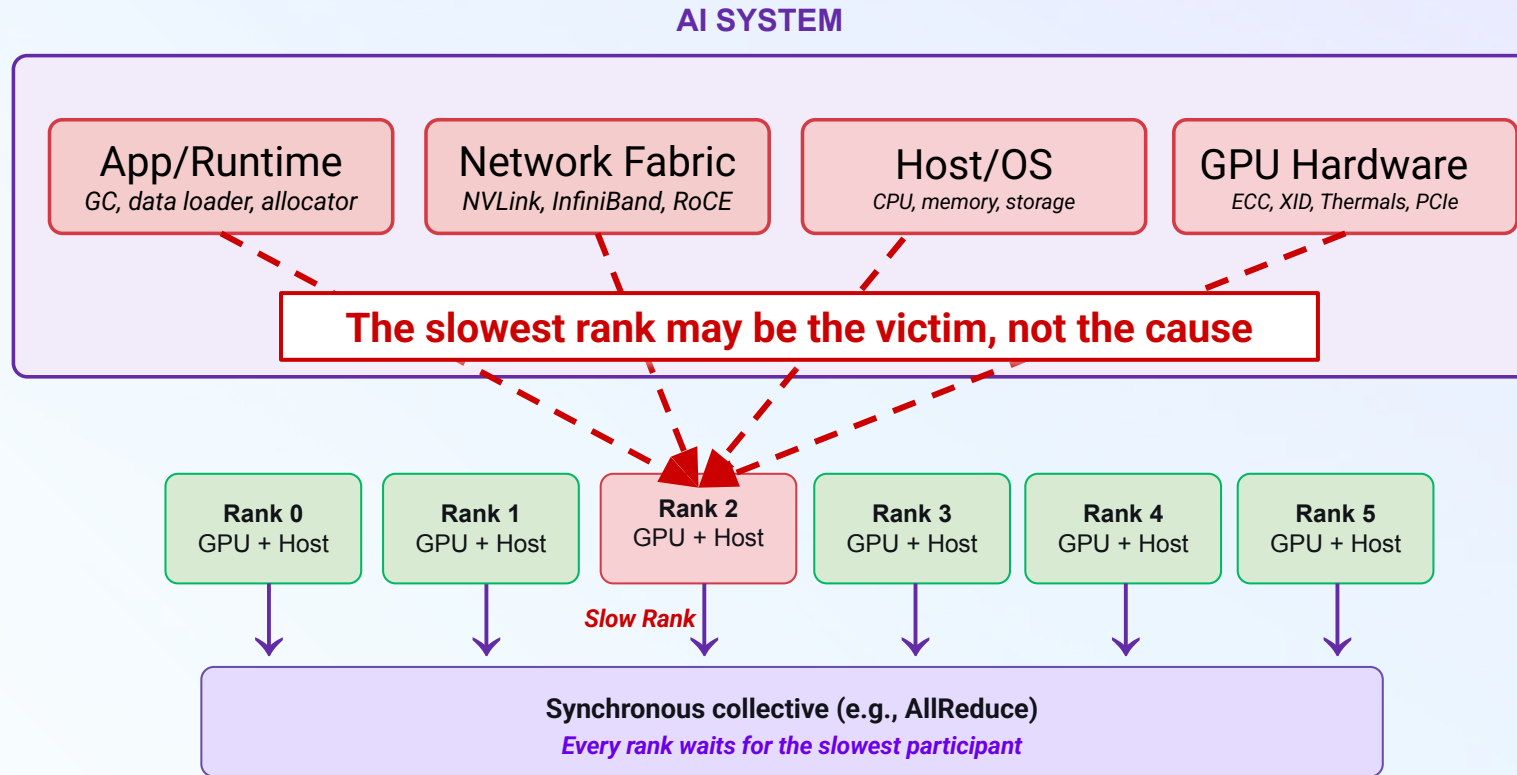
The visibly slow GPU may be innocent

The delay appears at one rank—but can begin elsewhere in the system.



The visibly slow GPU may be innocent

The delay appears at one rank—but can begin elsewhere in the system.



Finding the straggler is not the same as finding the cause.

Failures are increasingly common as the steady state

Meta, Llama 3 405B: 54-day pre-training run on 16,384 H100 GPUs

419

unexpected failures in 54
days

Every 3 hrs

average failure cadence

58.7%

of failures were
GPU-related

>90%

effective training time
held

Source: The Llama 3 Herd of Models, Meta AI (2024), Table 5.

Failures are increasingly common as the steady state

Meta, Llama 3 405B: 54-day pre-training run on 16,384 H100 GPUs

419

unexpected failures in 54
days

Every 3 hrs

average failure cadence

58.7%

of failures were
GPU-related

>90%

effective training time
held

At ~\$3 / GPU-hour, 16,384 H100s cost ≈ \$36M every month.

A 1% efficiency gap wastes ~\$360K / month (~\$4.3M a year). Our customer's ask, "save me 1% of my fleet cost" is real money.

Because of tight synchronization, one GPU failure can stall the entire run. Failures stop being events and become a constant

Source: The Llama 3 Herd of Models, Meta AI (2024), Table 5.

From unexplained MFU drop to product roadmap

We handled a live frontier MFU drop and turned the investigation into product

*A customer handed us an unexplained MFU drop.
We used their incident to learn what it actually takes to investigate one.*

We drove the team to respond in days, not months. Every gap we hit by hand becomes a product capability.

From unexplained MFU drop to product roadmap

We handled a live frontier MFU drop and turned the investigation into product

A customer handed us an unexplained MFU drop.

We used their incident to learn what it actually takes to investigate one.

1

MFU degradation reported

Slowdown flagged on pretrain run

We drove the team to respond in days, not months. Every gap we hit by hand becomes a product capability.

From unexplained MFU drop to product roadmap

We handled a live frontier MFU drop and turned the investigation into product

*A customer handed us an unexplained MFU drop.
We used their incident to learn what it actually takes to investigate one.*

1

MFU degradation reported

Slowdown flagged on pretrain run



2

Red herring ruled out

Agent restarts and Memory errors looked correlated

We drove the team to respond in days, not months. Every gap we hit by hand becomes a product capability.

From unexplained MFU drop to product roadmap

We handled a live frontier MFU drop and turned the investigation into product

*A customer handed us an unexplained MFU drop.
We used their incident to learn what it actually takes to investigate one.*

1

MFU degradation reported

Slowdown flagged on pretrain run

2

Red herring ruled out

Agent restarts and Memory errors looked correlated

3

Localization Real cause: NVLink fabric

XID 145 / 149. NVLink physical-layer retransmission as a leading indicator

We drove the team to respond in days, not months. Every gap we hit by hand becomes a product capability.

From unexplained MFU drop to product roadmap

We handled a live frontier MFU drop and turned the investigation into product

*A customer handed us an unexplained MFU drop.
We used their incident to learn what it actually takes to investigate one.*

1

MFU degradation reported

Slowdown flagged on pretrain run

2

Red herring ruled out

Agent restarts and Memory errors looked correlated

3

Localization Real cause: NVLink fabric

XID 145 / 149. NVLink physical-layer retransmission as a leading indicator

4

RCA + Product Learning

Topology, tracing, and workload-aware diagnosis

We drove the team to respond in days, not months. Every gap we hit by hand becomes a product capability.

GPU Monitoring with Datadog

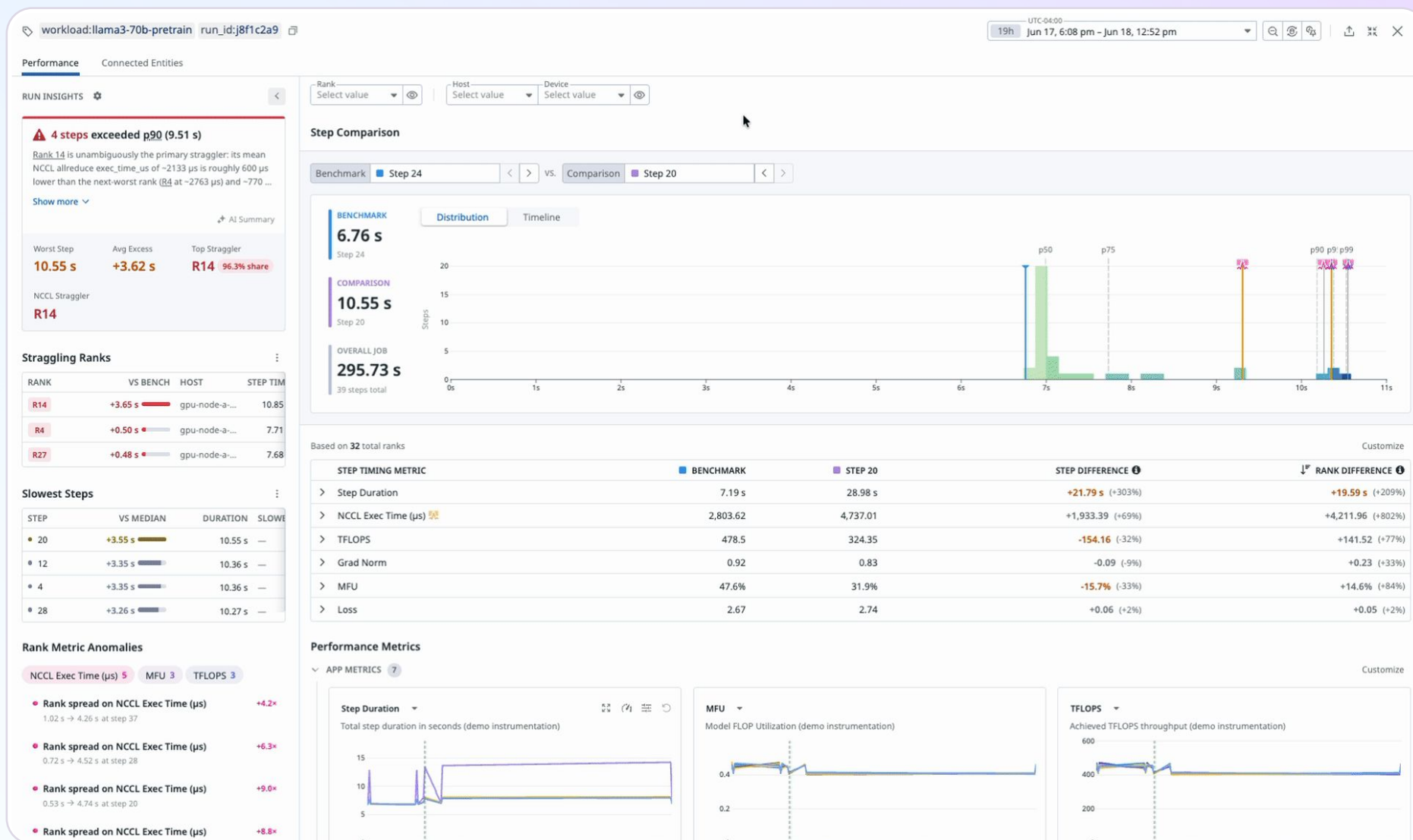
Debugging slow training jobs in GPU Monitoring - which rank?

1 Step 24 slowed

2 10.55 s vs. 6.76 s

3 Rank 14 identified

Now we know where to investigate. But why is it slowing down?



Training View: Root Cause Analysis

Agentically-powered investigations
directly connected via MCP and Bits
AI

llama3-70b-pretrain (j8f1c2a9) — RCA Summary

Hardware Fault — Straggler Causing Fleet-Wide Step Slowdown

Run llama3-70b-pretrain / j8f1c2a9 · July 15, 2026 · 2:21–2:24 PM EDT

Peak step time (a-003)
14.3s
vs ~7.0s baseline (+2×)

Uncorrectable ECC (peak)
6.4/min
host:gpu-node-a-003 only

MFU at fault peak
0.376
vs 0.45 baseline (–16%)

STEP TIME PER NODE · 2:19–2:26 PM EDT (1-MIN BUCKETS)

Step ≥12s

ECC uncorrected

Xid errors

Normal

Host	2:19	2:20	2:21 ⚡	2:22 🔥	2:23	2:24	2:25	2:26
gpu-node-a-003 ⚠️	6.9s	7.1s	9.3s	14.3s	7.7s	6.8s	7.0s	6.2s
gpu-node-a-001	6.9s	7.2s	7.9s	9.9s	7.2s	6.8s	7.1s	6.2s
gpu-node-a-002,004,005,006,007	~7.0s	~7.1s	~7.9s	~9.9s	~7.3s	~6.9s	~7.1s	~6.3s
gpu-node-b-* (all 8)	~7.0s	~7.0s	~7.0s	~7.0s	~6.9s	~6.9s	~7.1s	~6.3s

FAULT SIGNALS — GPU-NODE-A-003 ONLY (ALL OTHER NODES: ZERO)

Signal	2:19	2:20	2:21 ⚡	2:22 🔥	2:23	2:24+
ECC uncorrected/min	0	0	1.07	6.4	1.07	0
Xid errors/min	0	0	0.27	1.6	0.27	0
NVLink recovery/min	0	0	0.5	3.0	0.5	0
HBM row remaps	0	0	0.5	3.0	0.5	0

ROOT CAUSE CHAIN

⚡ TRIGGER

A single GPU on gpu-node-a-003 — gpu-uuid: --0002-0003-...000e — suffered uncorrectable HBM/ECC errors starting at 2:21 PM EDT. Xid errors, NVLink recovery events, and active row remapping all corroborate a hardware memory fault on this device. The 3 other GPUs on the same host stayed clean.

⚙️ MECHANISM

The faulty GPU stalled its rank's computation during the collective communication phase. Because allreduce barriers require all ranks to rendezvous, the entire 16-node job was forced to wait. Step time on the culprit host peaked at 14.3s (+100%); even the fastest unaffected nodes recorded ~9.9s (+40%) as they waited at the barrier.

👥 CASCADE

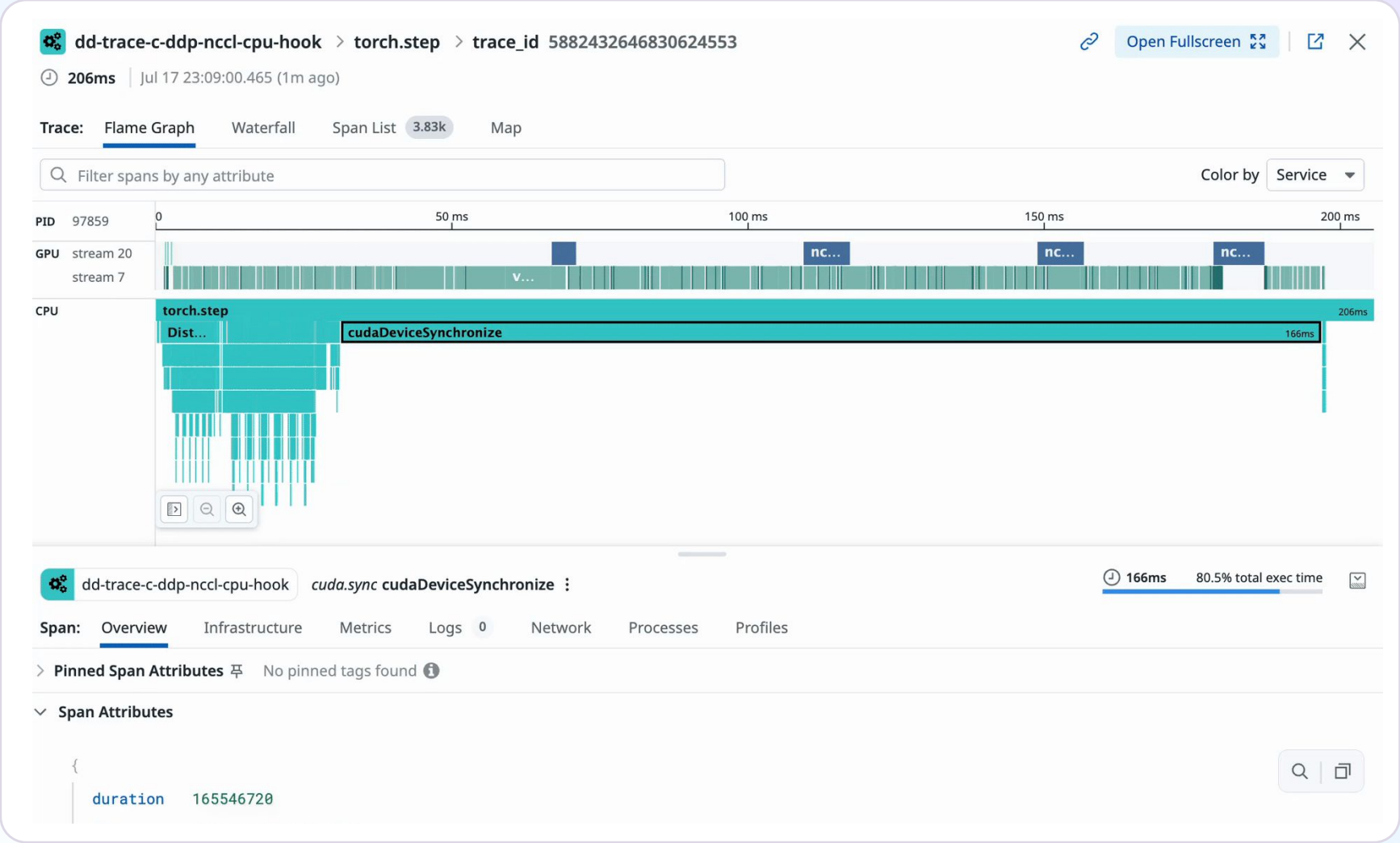
All a-side nodes (a-000 through a-007) experienced the step-time inflation. b-side nodes were largely unaffected, consistent with pipeline or tensor-parallel group topology where a-003 and some a-side peers share the same synchronization group. Fleet MFU dropped from 0.45 → 0.376 during the 2-minute fault window. The run appeared to recover — but the uncorrectable ECC errors mean **gradients computed during 2:21–2:24 PM may be silently corrupted.**

REQUIRED ACTIONS

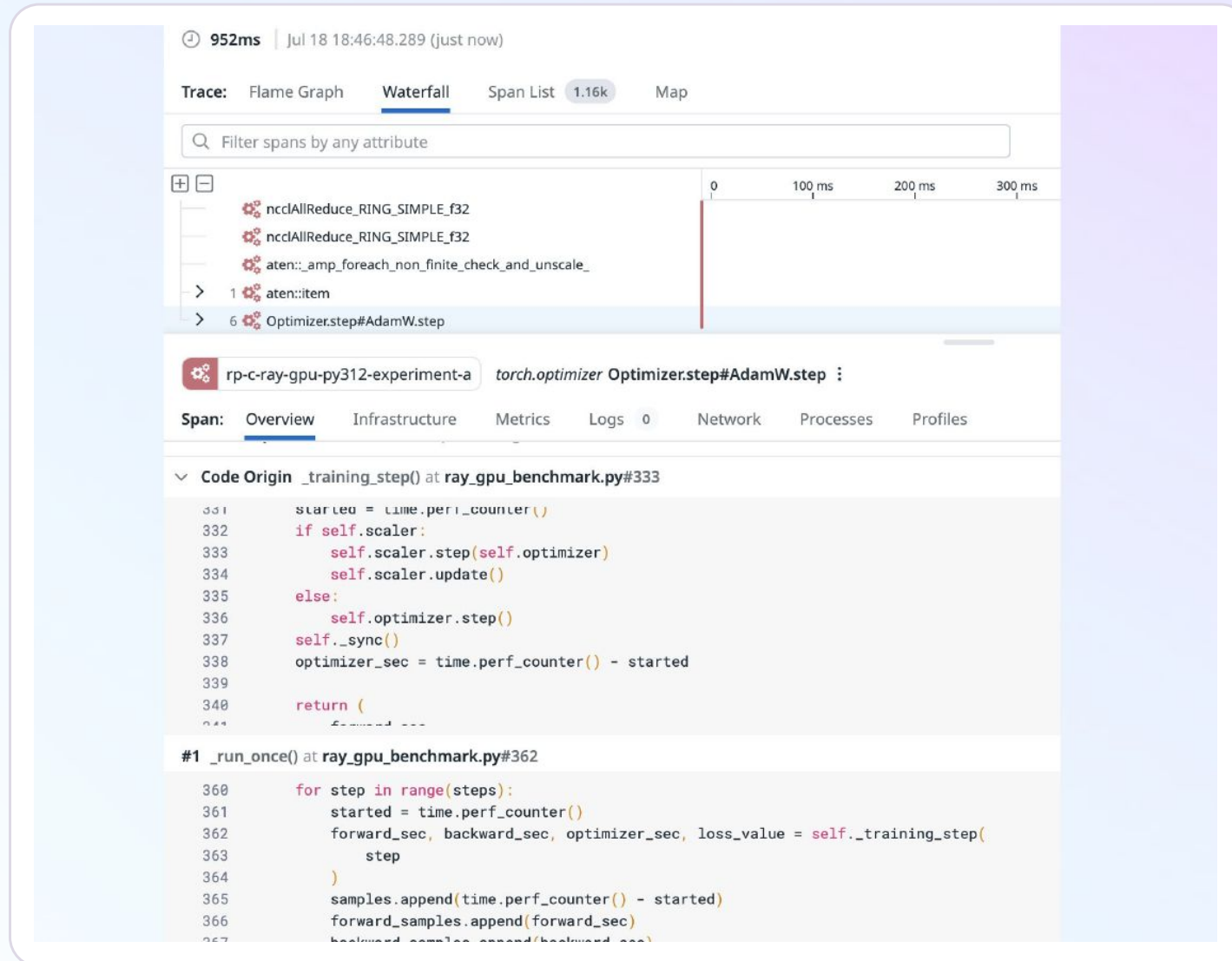
1 Drain gpu-node-a-003 now. Uncorrectable ECC + active row remapping = faulty HBM. The specific GPU is gpu-uuid --0002-0003-...000e. Even if the node appears recovered, it has proven hardware damage that will reoccur.

2 Roll back to the last checkpoint before 2:21 PM EDT on July 15 and restart the run on a clean replacement node. Any checkpoint saved after 2:21 PM may contain corrupted data.

Tracing View: Where did the missing time go?



Tracing View: Code Origins



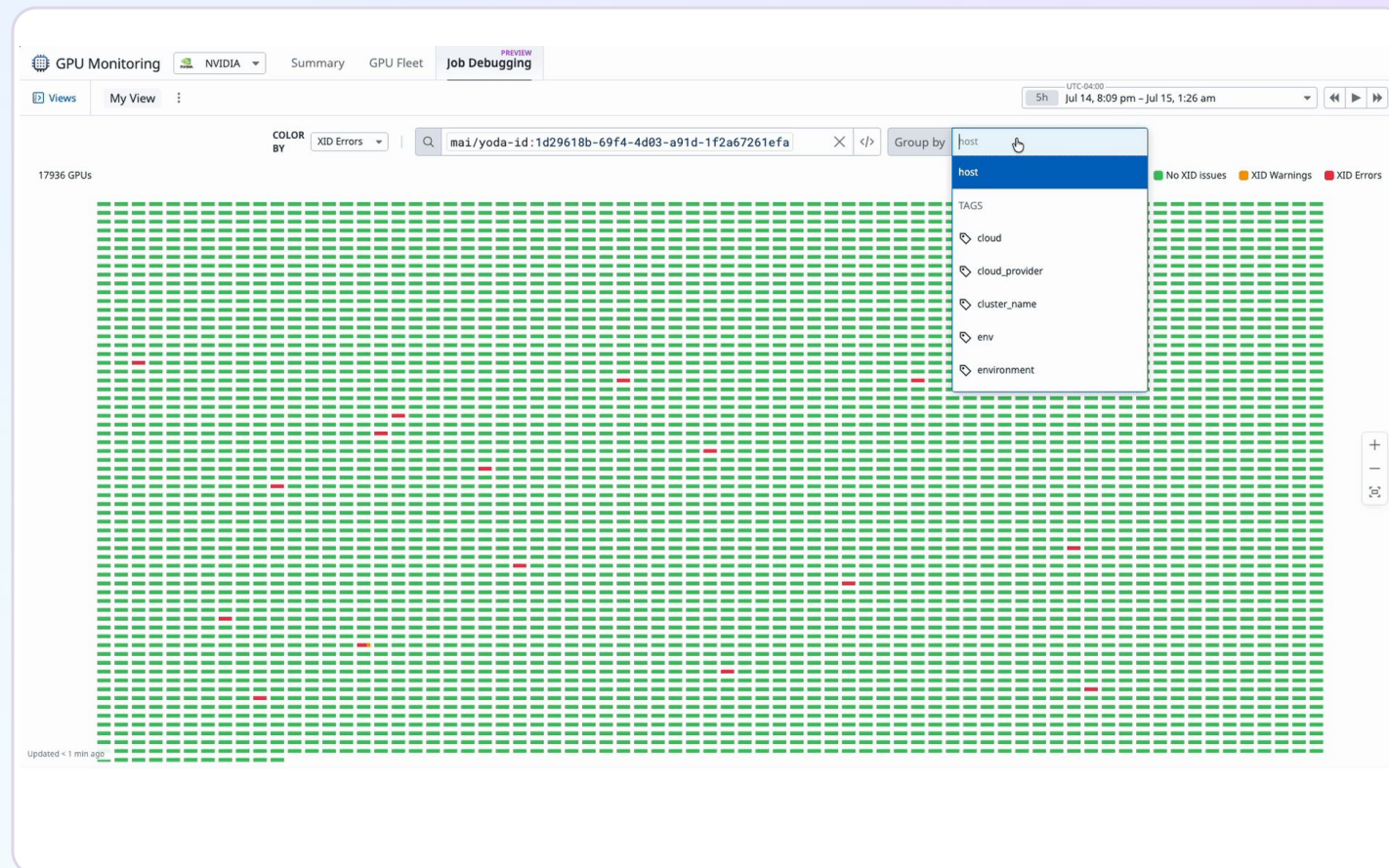
Topology View: Where did the pattern concentrate?

Do affected workers share physical infrastructure?

HOST RACK GPU LINK

FINDING

Errors were related to full rack of GPUs



A stalled Ray job becomes a diagnosable system with Datadog's GPU Monitoring product

1 Training

Find the slow rank

Scope the symptom

A stalled Ray job becomes a diagnosable system with Datadog's GPU Monitoring product

1 Training

Find the slow rank

Scope the symptom

2 Tracing

Explain the missing time

Working from waiting

A stalled Ray job becomes a diagnosable system with Datadog's GPU Monitoring product

1 Training

Find the slow rank

Scope the symptom

2 Tracing

Explain the missing time

Working from waiting

3 Topology

Localize the cause

Shared dependency

So, where's your training job?

**Connected context turns a stalled job
into a diagnosable system.**

Ray helps us scale the work. Datadog helps us understand the system that work becomes.

GPU Monitoring



Get more value from each GPU.

Start your free trial – the first
14 days are on us.

Datadog for Startups



Startup new to Datadog?

Get up to \$100K in Datadog
credits for eligible startups.

Thank You!

Contact us

gpu-monitoring-product@datadoghq.com