A Demand-Side Approach to Modeling Data Path Requirements for Training

Glenn K. Lockwood, Ph.D.
Principal Technical Strategist
VAST Data

The problem:

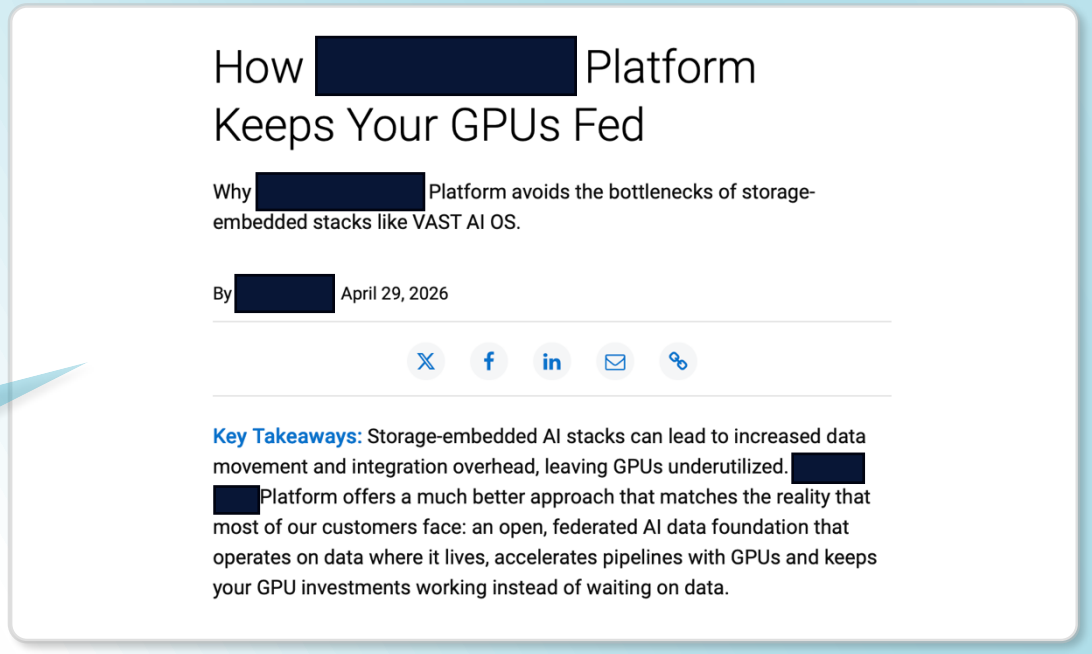
In the absence of data, marketing hyperbole guides decisions

"the real bottleneck isn't the GPU itself—it's the data layer that feeds it"

-- Some storage vendor, April 29, 2026

"GPUs are sitting idle 40% of the time while waiting for data to trickle in"

-- Some other storage vendor, January 6, 2026



How [redacted] Platform Keeps Your GPUs Fed

Why [redacted] Platform avoids the bottlenecks of storage-embedded stacks like VAST AI OS.

By [redacted] April 29, 2026

Key Takeaways: Storage-embedded AI stacks can lead to increased data movement and integration overhead, leaving GPUs underutilized. [redacted] Platform offers a much better approach that matches the reality that most of our customers face: an open, federated AI data foundation that operates on data where it lives, accelerates pipelines with GPUs and keeps your GPU investments working instead of waiting on data.

How the [redacted] Platform Solves AI's Million-dollar GPU Problem

Organizations face a multimillion-dollar problem as legacy, siloed storage starves expensive GPUs of data and crushes AI ROI. See how a unified, always-on data platform keeps GPUs fully utilized and accelerates enterprise AI.

But claims of I/O insufficiency are not warrantless

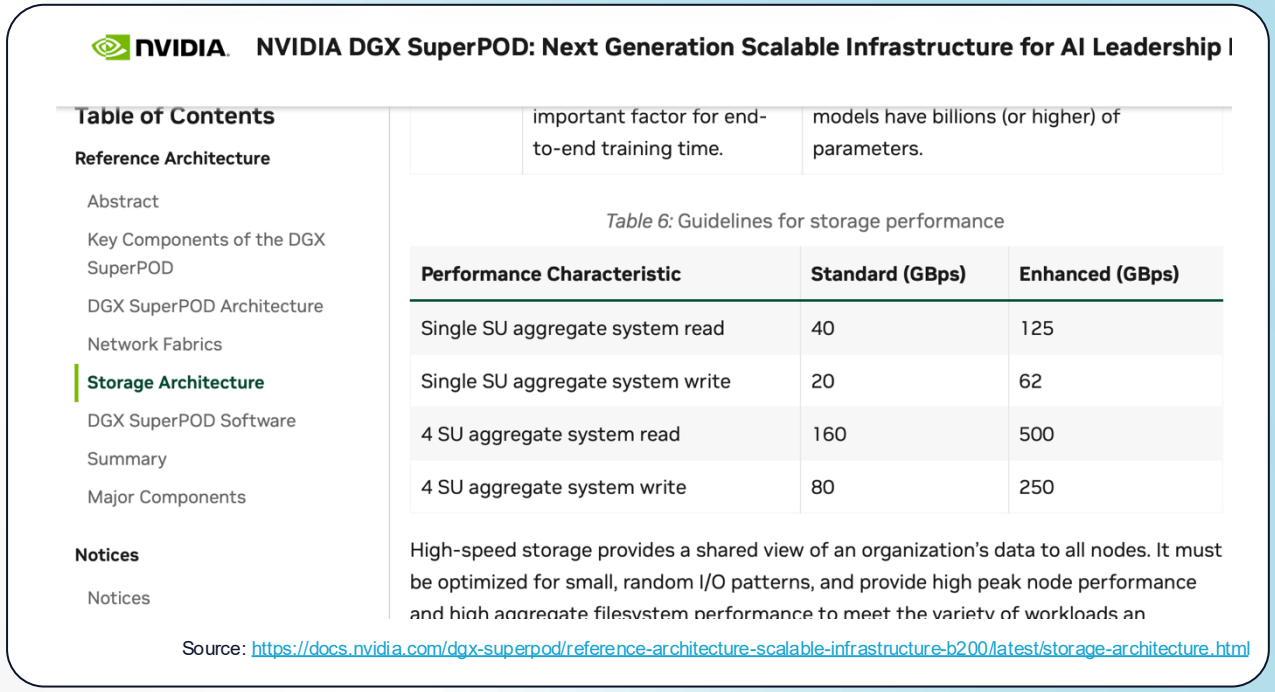
GPUs process tokens (data) at a finite rate

Text models need 156 MB/s/GPU

- Every token requires lots of compute
- Lots of time to stage next batch

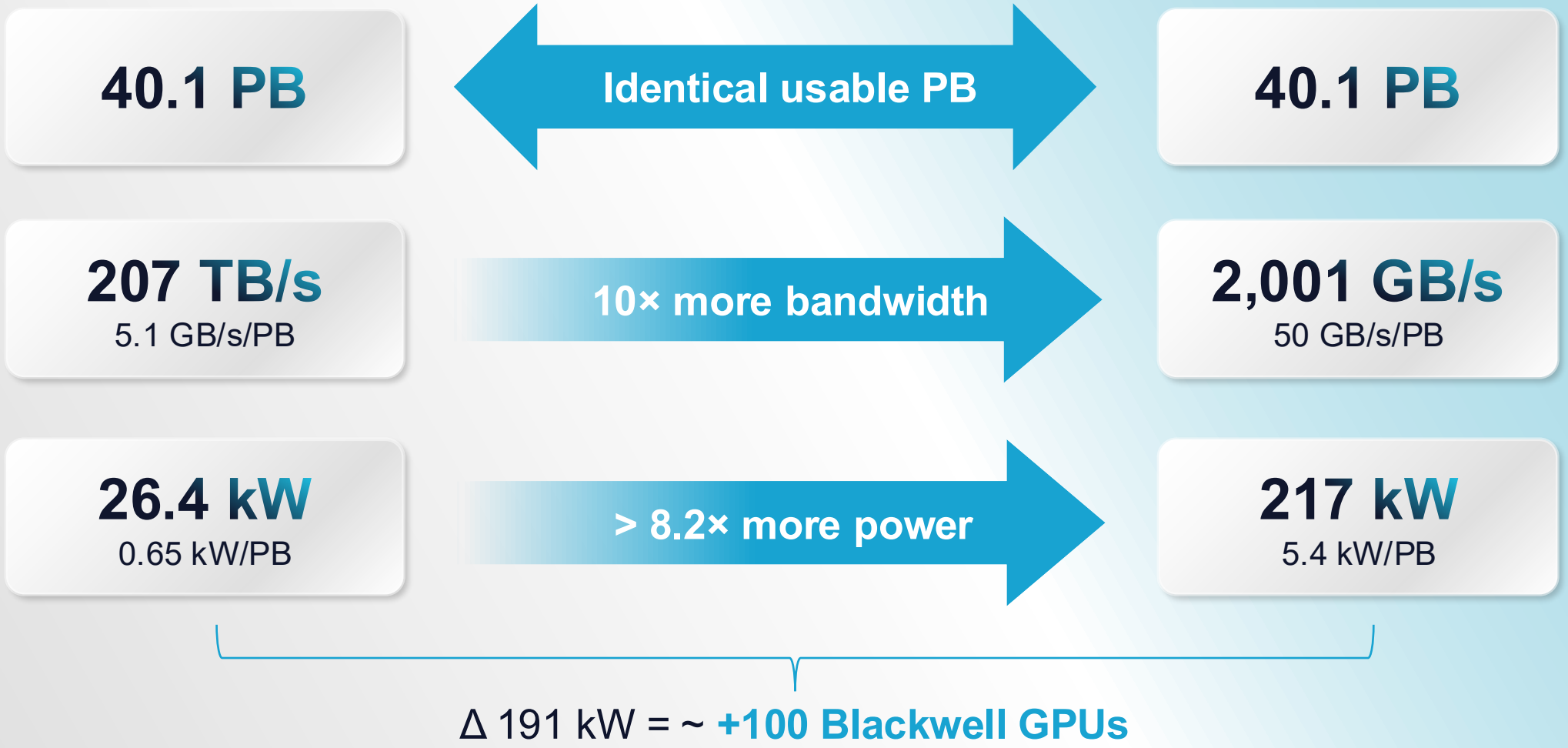
Visual models need 488 MB/s/GPU

- Image encoders require little compute
- Less time to stage next batch

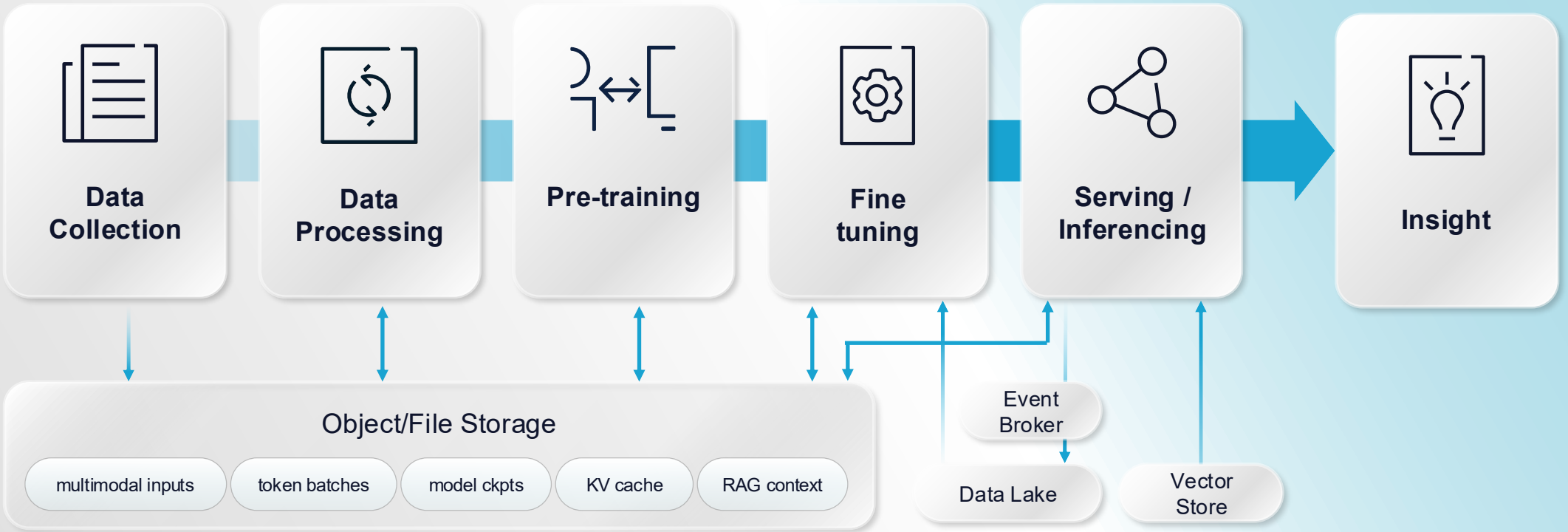


But these specify scalar I/O rates into HBM, *not* I/O rates from storage

Storage is cheap relative to compute. Why not maxx it?



Understanding the demand side of I/O performance



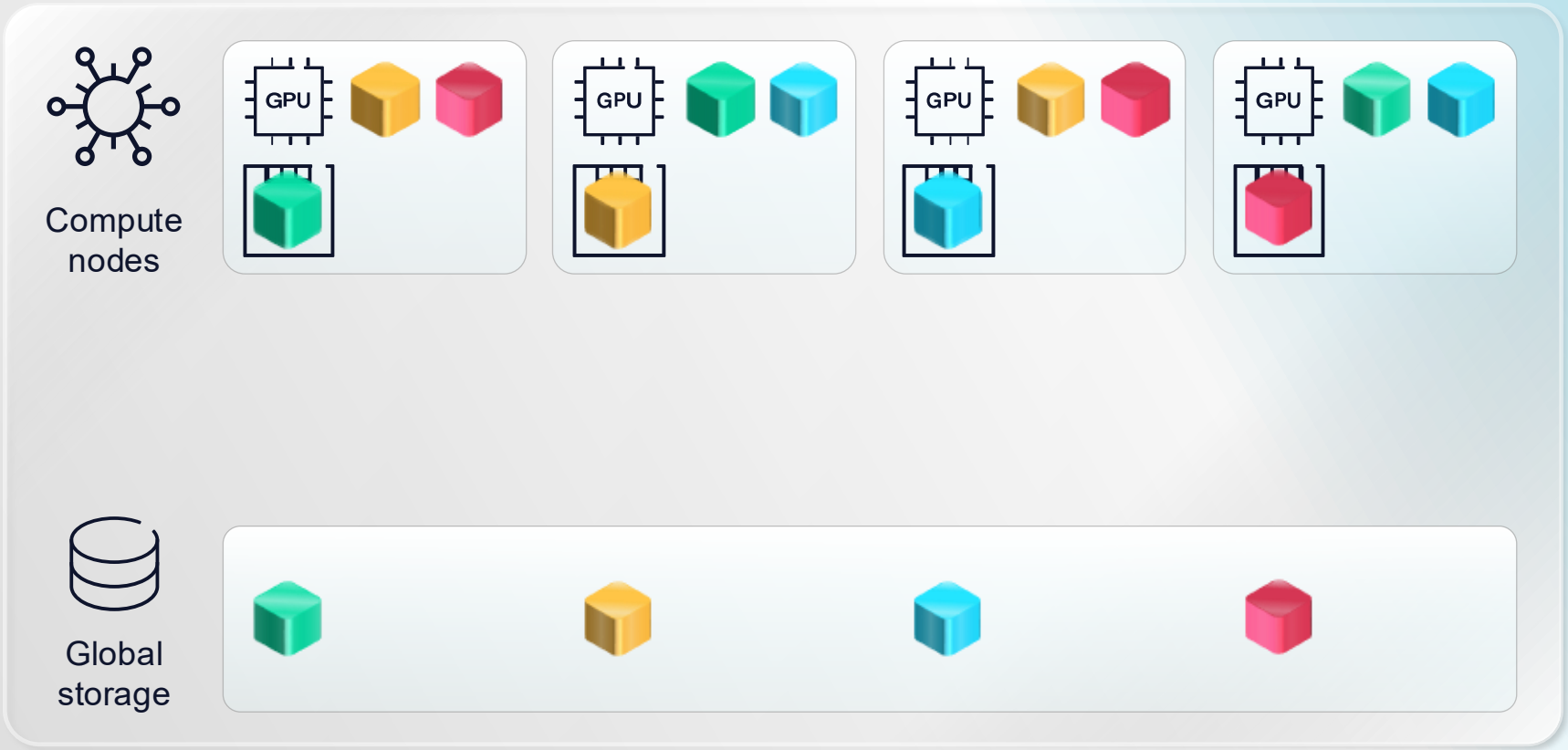
The demand side of training, in principle

What drives I/O during pretraining?

1	**Checkpointing**	Write single model replica as fast as possible
2	**Restarting**	Read one model replica as fast as possible
3	**Data loading**	Read inputs into HBM for next forward + backward pass before current step finishes

1. Checkpointing: Asynchronous and hierarchical

Synchronous checkpoint performance absorbed by CPU DRAM or local SSD



Node-local storage

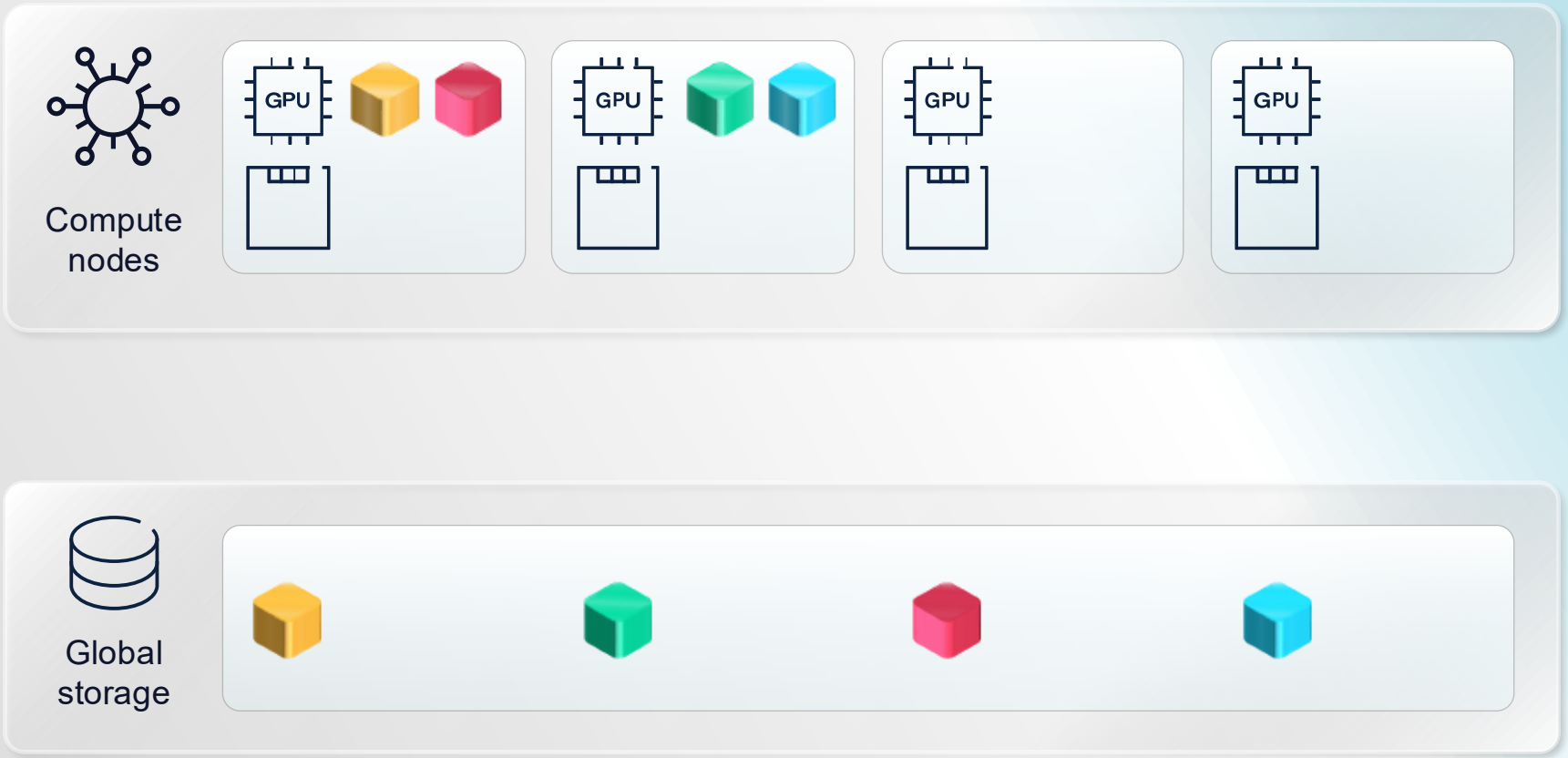
- Synchronous
- Blocks GPUs
- CPU RAM and/or local SSD
- Scalable

Global storage

- Asynchronous
- Drains from node-local storage

2. Restart: Read once and broadcast

Low-intensity read followed by high-intensity peer-to-peer transfer



Node-local storage

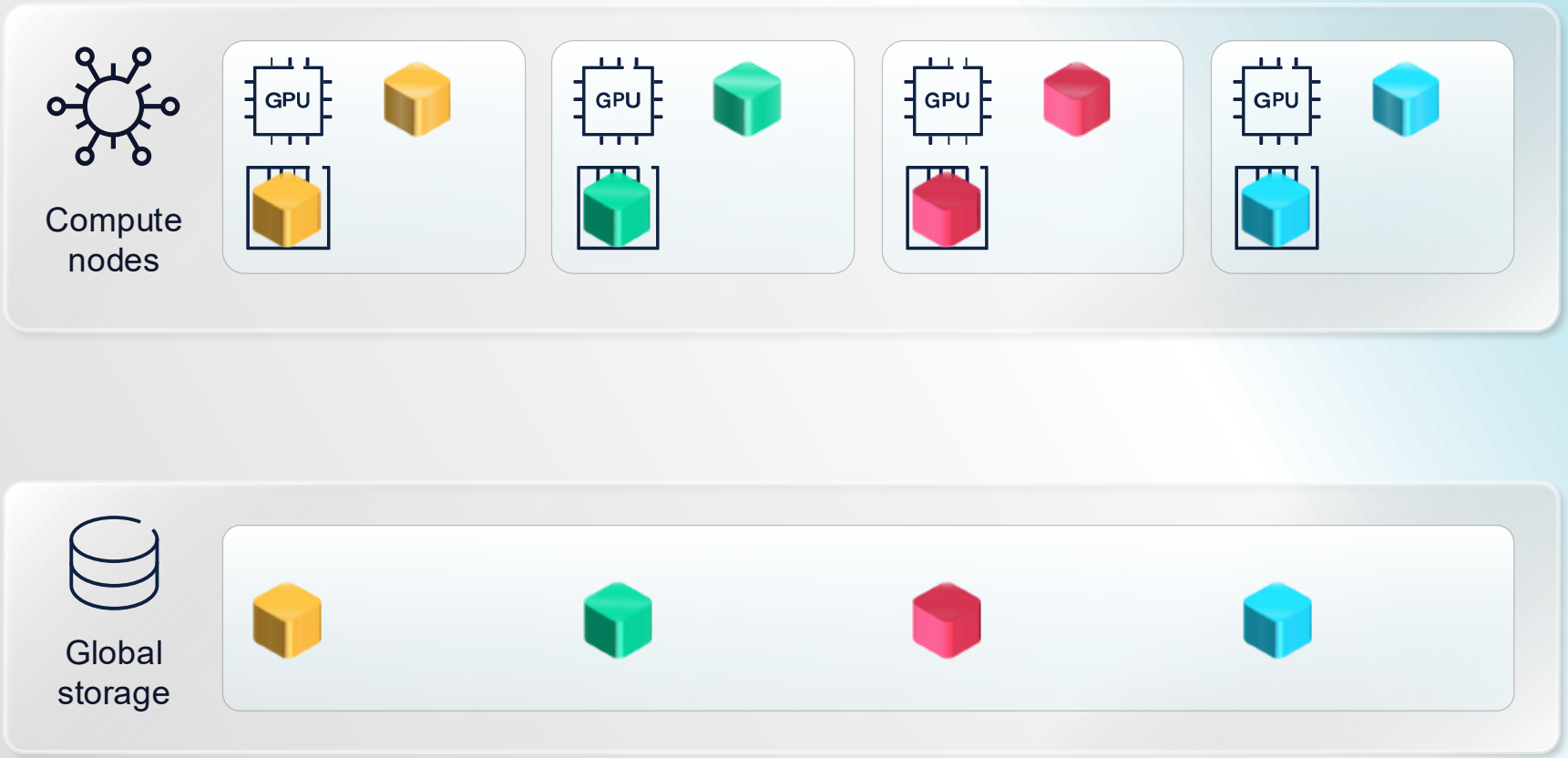
- May contain usable checkpoint shards

Global storage

- Single copy of checkpoint read to a subset of nodes
- Single copy is broadcast to other nodes

3. Data loading: Data-parallel or sharded data-parallel

Distribute training data across node-local SSDs, then synchronously load data from there



Node-local storage

- Dataset sharded over all nodes
- Data loading I/O served by local SSDs
- Scalable

Global storage

- Data read by all nodes at job start
- Never in sync training loop

The demand side of training, in practice

Write performance

Analyze 140K GPU-years of telemetry from the VAST fleet

Algorithmically identify training jobs from production traces

Fancy stats[1] identify training jobs

Filter on high-quality job signals

- Single-tenant / single-user jobs
- Periodic checkpoint drains

Limitations:

- Only global storage (not node-local)
- Only clusters with > 1024 GPUs
- Only clusters known to be for training

22	Clusters from four cloud providers
183	Wall-days of LLM training
931	Model training jobs
36B - 1.3T	Parameters per model

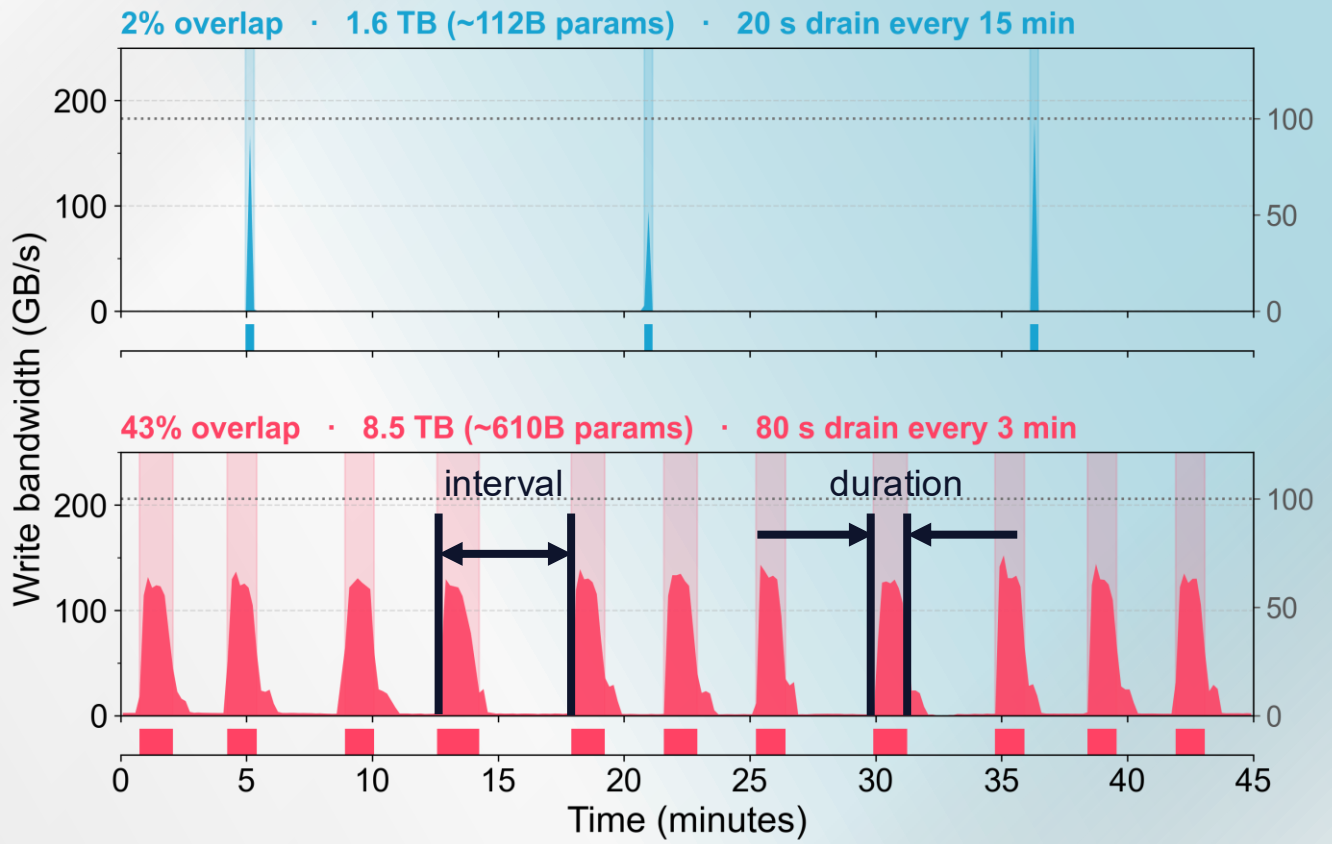
[1]: Details:

- Detect concurrent peaks in write bandwidth and capacity growth
- Group peaks into runs of similar (within 1.5x) cadence, filter on noise
- Group unbroken (< 7 day) chains of runs with similar (2x) cadence into jobs

Checkpoint drain characterization

- **Duration**: time spent bursting
- **Interval**: time between bursts
- **Overlap**: duration / interval
  - Dictates minimum required write bandwidth
  - Training not impacted until overlap > 100%

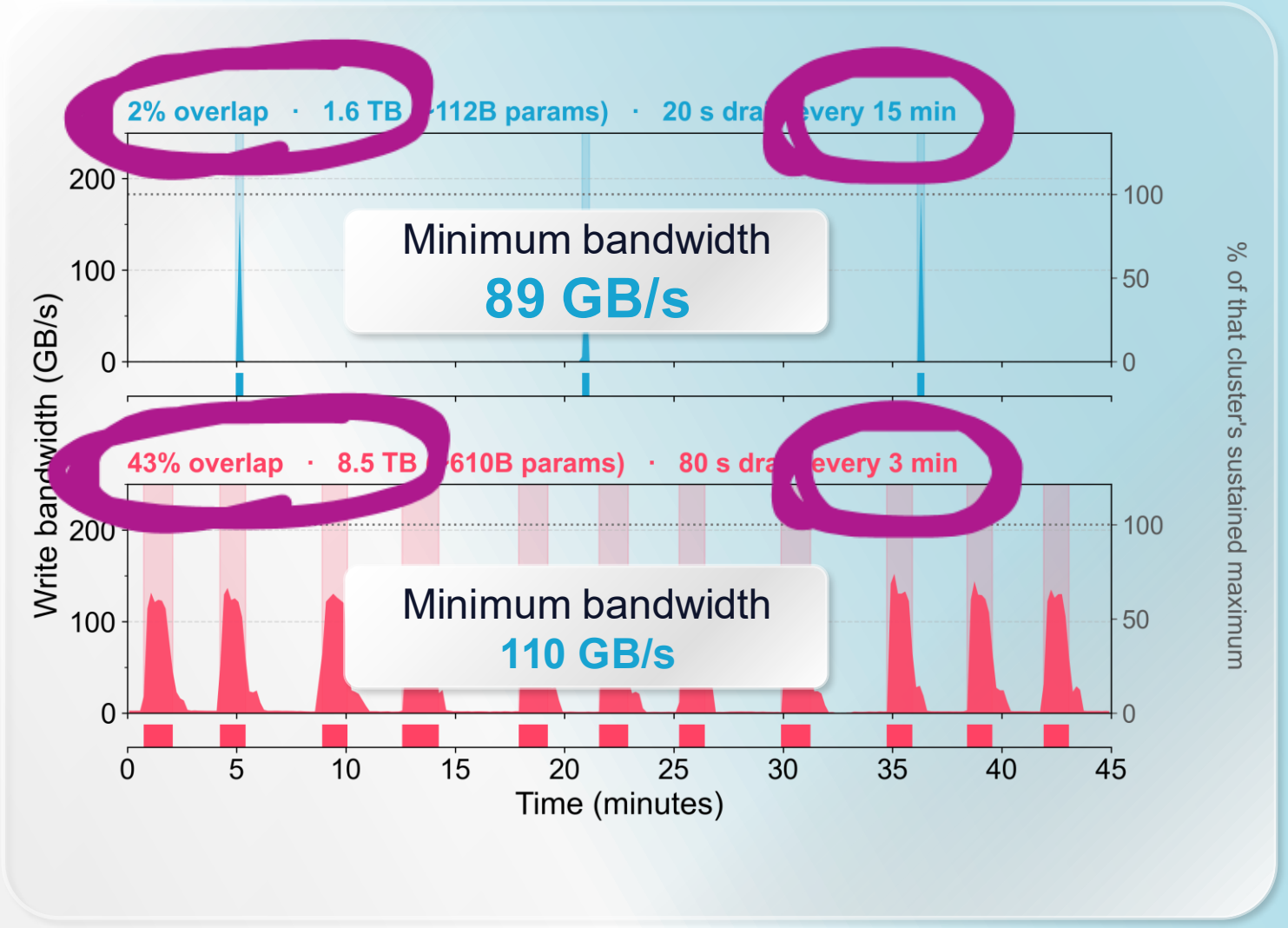
$$\text{Min write bandwidth} = \frac{\text{checkpoint size}}{\text{interval} \times \text{overlap}}$$



Checkpoint drain characterization

- **Duration**: time spent bursting
- **Interval**: time between bursts
- **Overlap**: duration / interval
  - Dictates minimum required write bandwidth
  - Training not impacted until overlap > 100%

$$\text{Min write bandwidth} = \frac{\text{checkpoint size}}{\text{interval} \times \text{overlap}}$$



What governs checkpoint behavior?

$$\text{Min write bandwidth} = \frac{\text{checkpoint size}}{\text{interval} \times \text{overlap}}$$

Checkpoint Size	**User defined**. Parameter count. O(14) bytes per parameter.
Checkpoint Interval	**User defined**. Probably related to MTBF, but not Young/Daly due to asynchronous nature.
Checkpoint Overlap	**System designer defined**. But what is reasonable?

Checkpoint interval isn't intrinsic to the workload

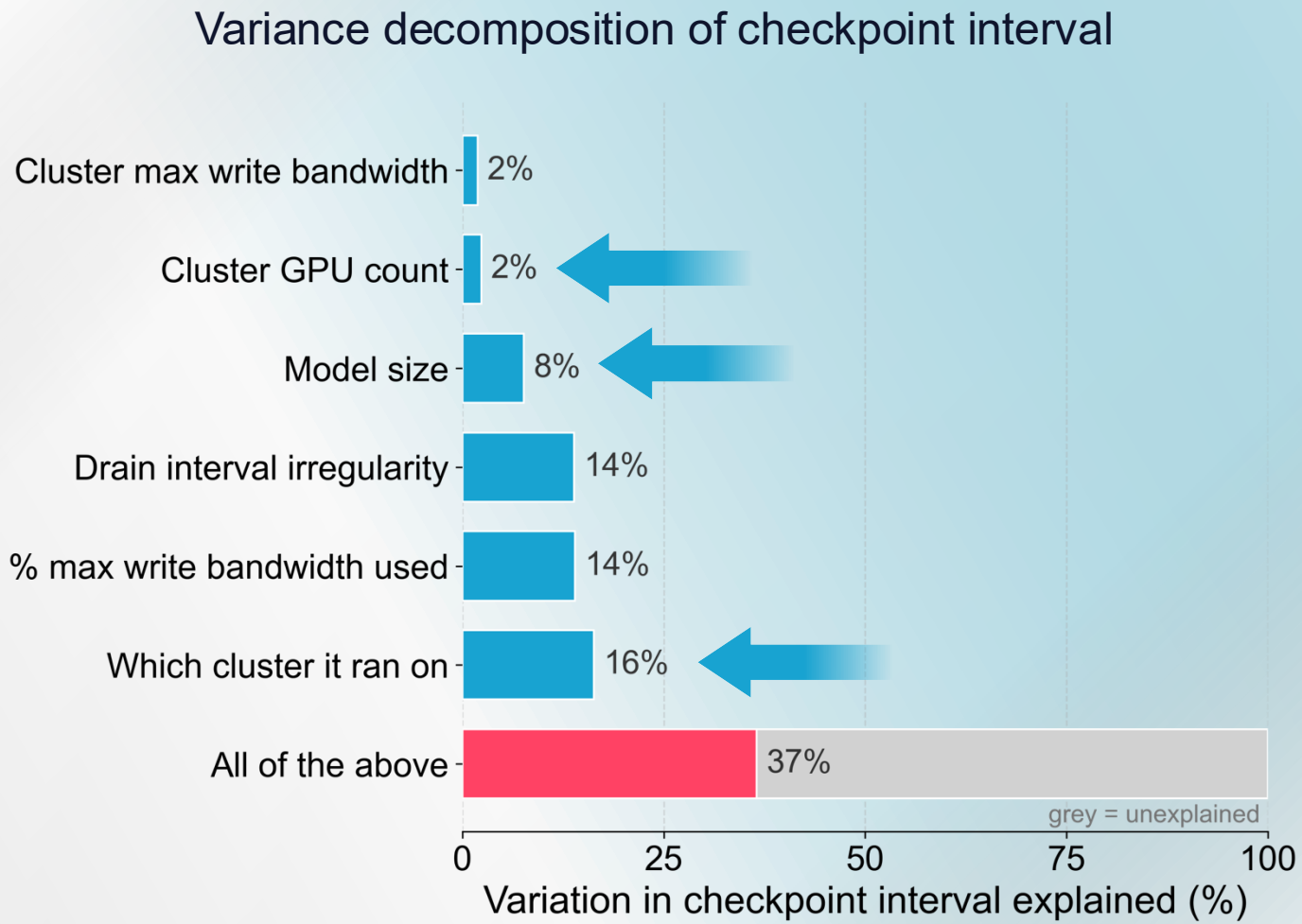
Nothing strongly predicts checkpoint interval:

- **63%** of variance is **unexplainable**
- GPU cluster size (~MTBF) is 2%
- Model size is 8%

Strongest predictor (16%): which GPU cluster is running job?

- Hardware factors
- Users, usage policies, human elements

Users seem to pick checkpoint intervals arbitrarily



What governs checkpoint behavior?

$$\text{Min write bandwidth} = \frac{\text{checkpoint size}}{\text{interval} \times \text{overlap}}$$

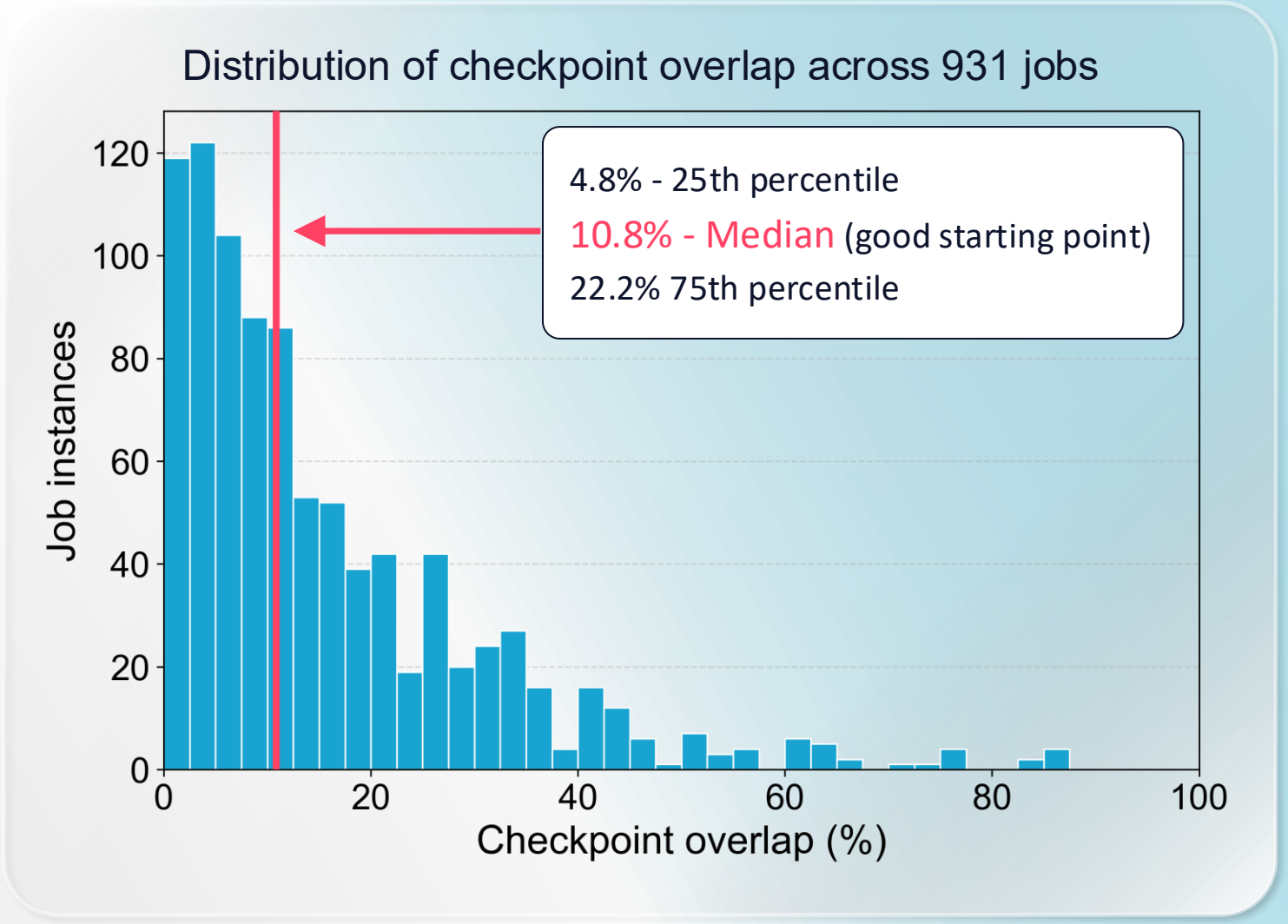
Checkpoint Size	**User defined**. Parameter count. O(14) bytes per parameter.
Checkpoint Interval	**User defined**. **Mostly unpredictable.**
Checkpoint Overlap	**System designer defined**. But what is reasonable?

Overlap is distributed widely and unpredictable

Variance analysis fails for overlap too:

- 61% unexplainable
- Falls out of model size and cluster bandwidth

Just pick a reasonable value



A model for write performance requirements

User requirements:

- Model size
- Checkpoint interval

You decide:

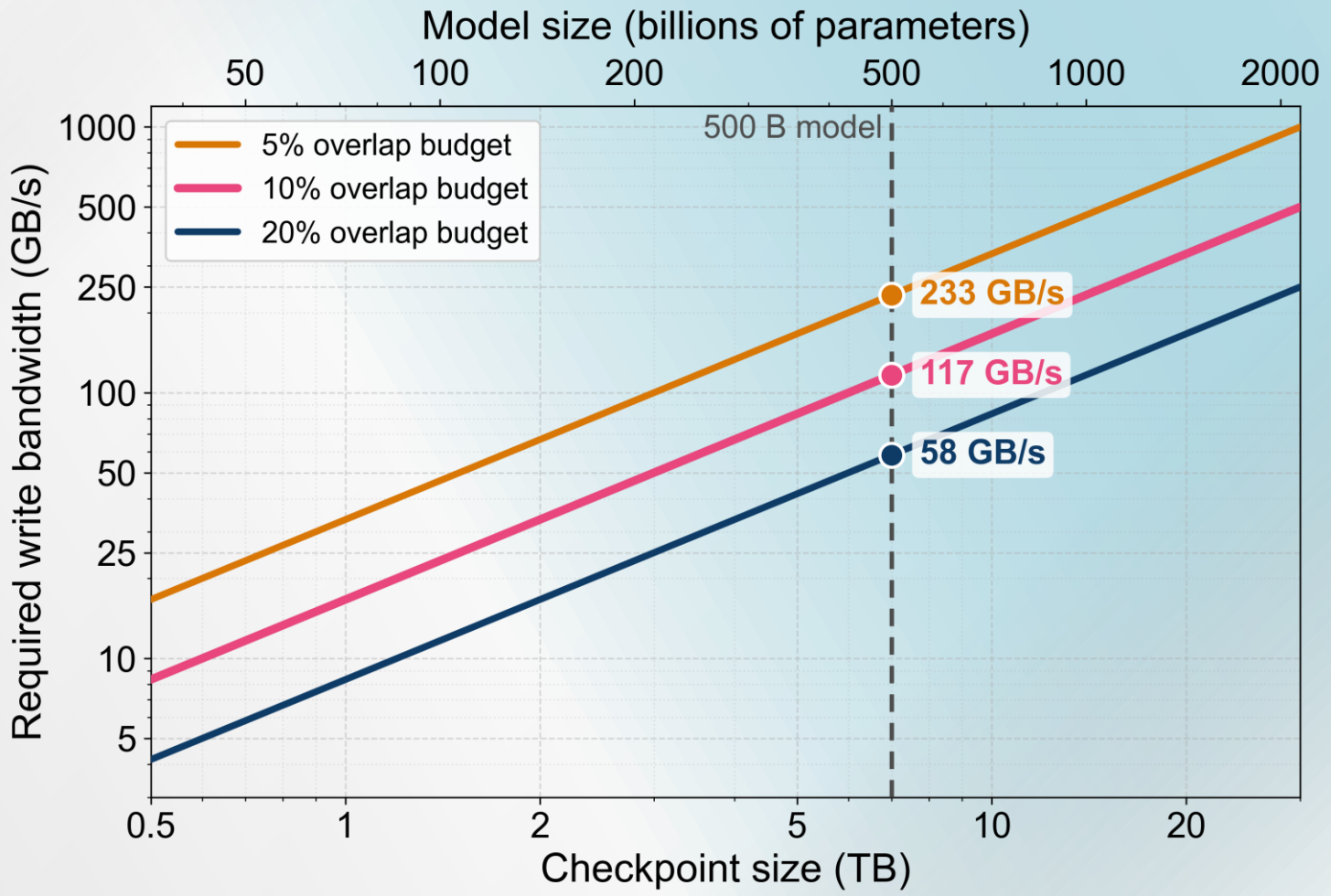
- What overlap will I support?

Median interval: 10 minutes
Median overlap: 10%

17 GB/s per TB checkpointed

23 GB/s per 100B params

$$\text{Min write bandwidth} = \frac{\text{checkpoint size}}{\text{interval} \times \text{overlap}}$$

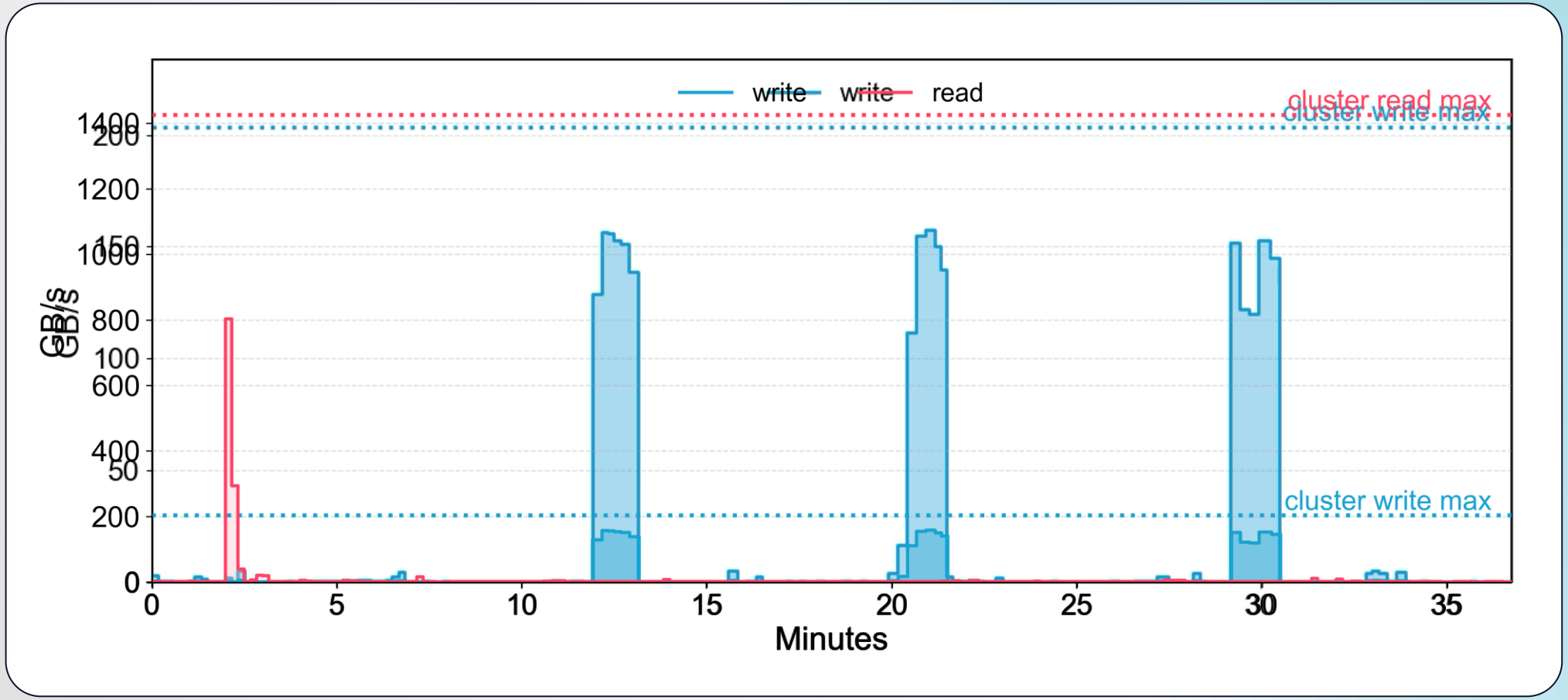


The demand side of training, in practice

Read performance

Adding read bandwidth to the equation

Anecdotally, write bursts and read bursts are asymmetric during training



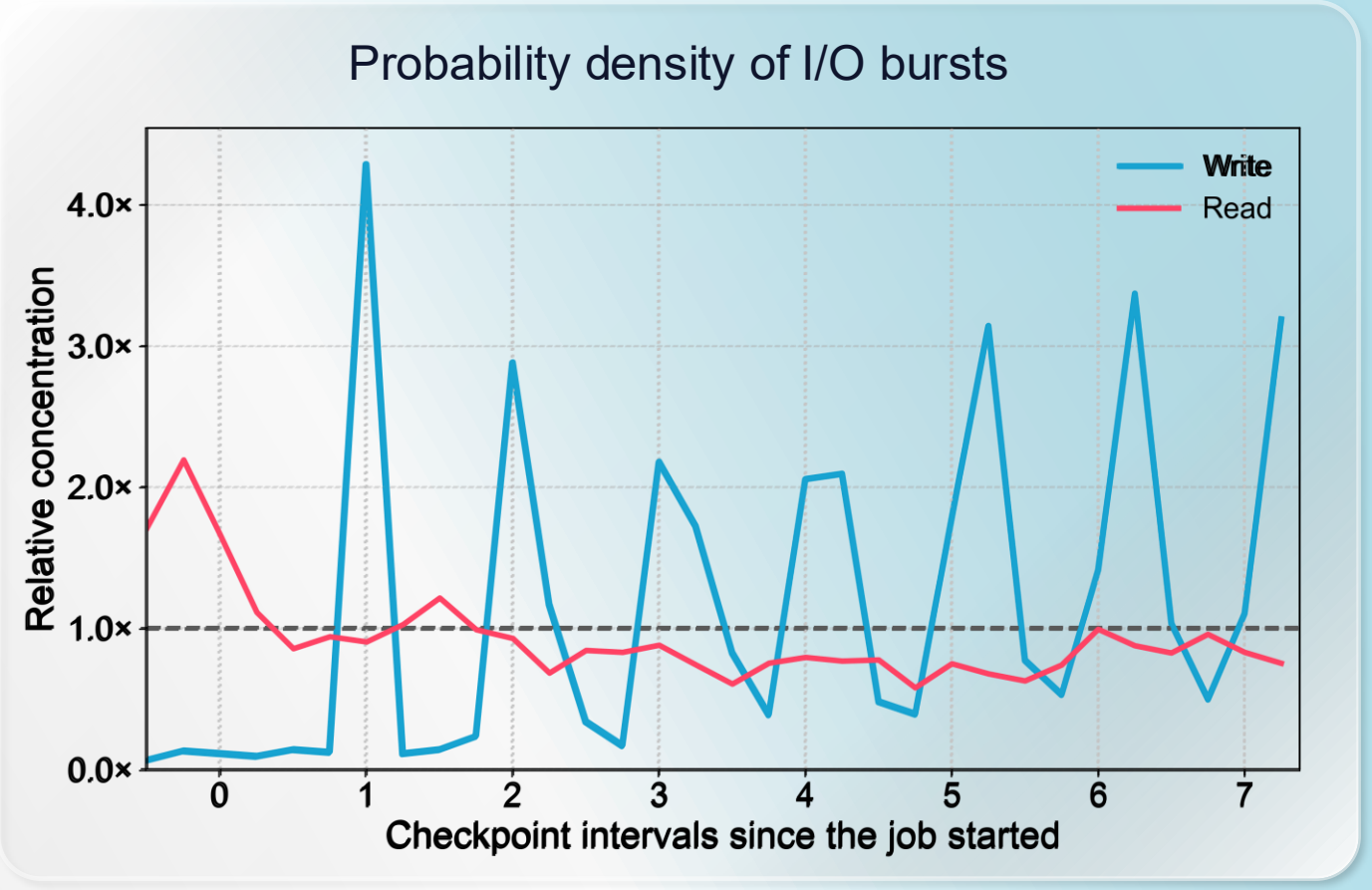
Read bursts concentrate at the start of jobs

733 job restarts, stacked

Writes show periodicity: checkpoint drains

Reads show one burst at job start: restart

No periodic data loading

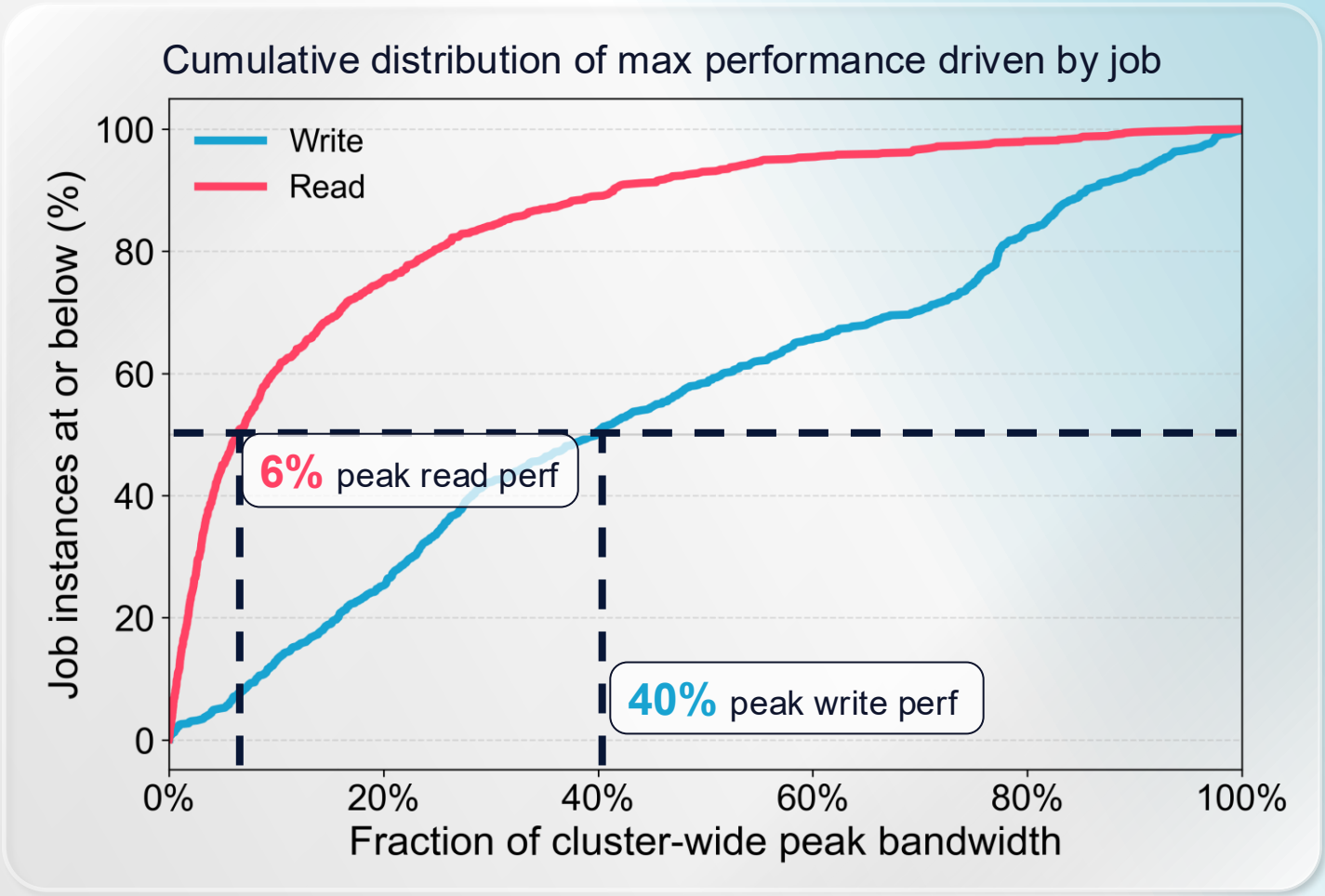


Details:

- Project jobs on to a common timescale (unit checkpoint time)
- Probability density of an I/O burst at each point in time
- Allows us to find statistical structure across many different jobs

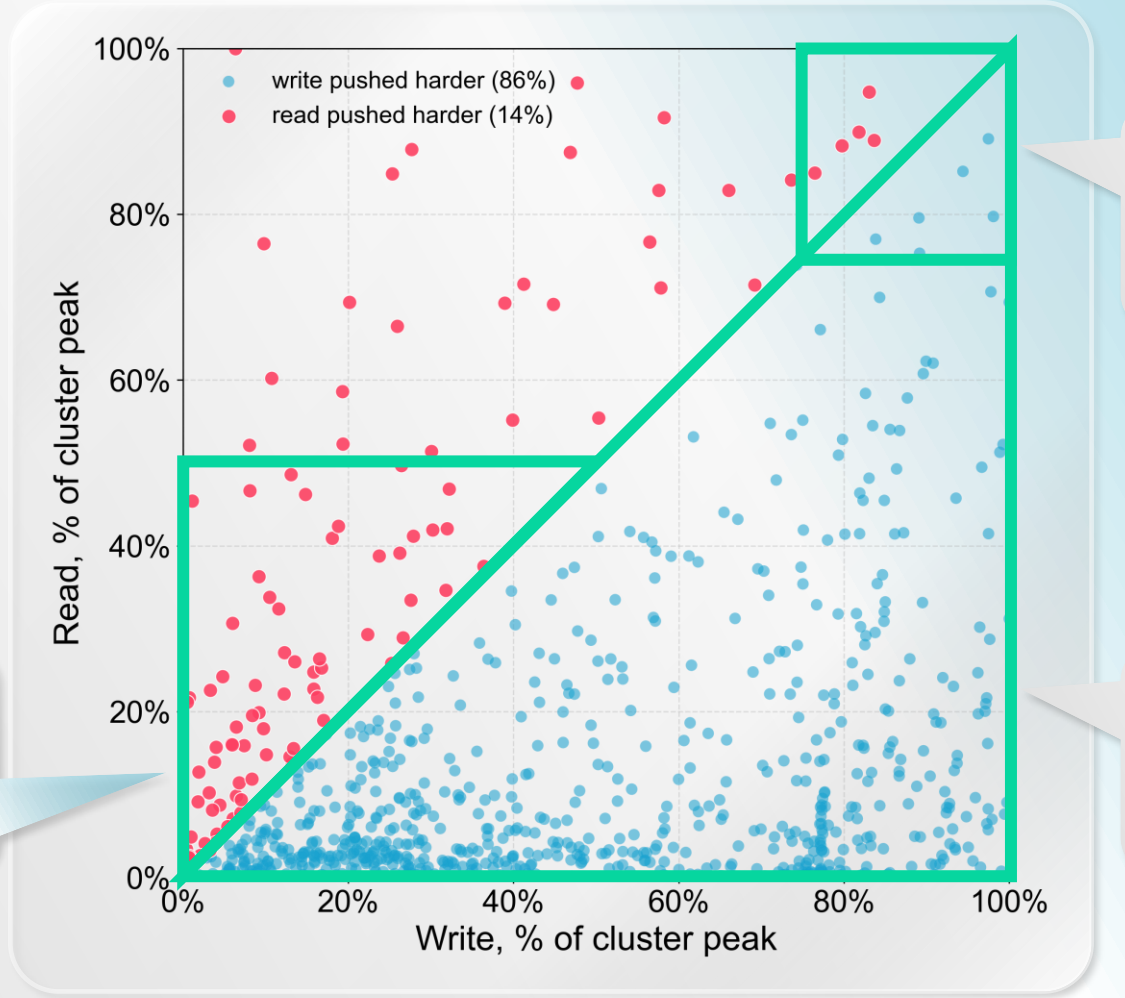
Read intensity is lower than write intensity across jobs

Jobs tend to saturate more of clusters' write bandwidth than read bandwidth



Jobs will be fine if you just get write bandwidth correct

Writes, not reads, more closely approach storage cluster limits within individual jobs



1.3% of jobs drive more than 75% of clusters' read *and* write max

86% of jobs write harder than reads

Read-intensive jobs are small and do not saturate the cluster

Worry about the writes, and reads come free

Feeding the GPUs is not a problem in real-world training

- Writes demand more than reads
- Read bursts occur at job start
- Data loading is a trickle at best

Defining storage performance for training is simple

1. Decide the correct write bandwidth
   - User defines model size (e.g., 100B parameters)
   - User makes up an interval (e.g., 10 minutes)
   - System designer defines a supported overlap (e.g., 10%)
2. Whatever read bandwidth results will be good enough

$$\text{Min write bandwidth} = \frac{\text{checkpoint size}}{\text{interval} \times \text{overlap}}$$