Rillor

EXPLAINER

Storage for model training: datasets, checkpoints and weights

Storage requirements for model training and checkpointing, from published sizes and measurements: datasets read on every pass, checkpoints written in bursts and weights kept by version.



Contents

01 Summary

02 Background

03 Method

04 Findings

05 Implications

06 Limits of this analysis

07 Sources

Summary

Storage for model training holds datasets, checkpoints and weights, and each sets its own requirement: datasets set the read rate, checkpoints set the write bursts and restores, and weights set the versions kept.

- A checkpoint, the saved training state a run resumes from, holds the model weights and the optimizer state, the values an optimizer such as Adam keeps for every parameter. In the rules of MLCommons' MLPerf Storage benchmark, a 70-billion-parameter model's checkpoint is 912 GiB, about 14 bytes per parameter and roughly seven times its 16-bit weights (computed for this report from [1, 9]).
- Training stalls while a checkpoint is written synchronously, and the write takes about its size divided by the storage bandwidth [7]. At the 20 GBps and 62 GBps write guidelines NVIDIA publishes for 32 DGX B200 systems, a 70-billion-parameter model's 912 GiB checkpoint takes about 49 and 16 seconds (computed for this report from [9, 24, 25]).
- Failure rates set how often to checkpoint. A 405-billion-parameter pre-training run on up to 16K H100 GPUs had 419 unexpected interruptions in 54 days [5], and a 2025 study of one operator's research clusters projects a mean time to failure of 1.8 hours for jobs on 16,384 GPUs [10].
- Asynchronous checkpointing shortens the pause without shrinking the write. PyTorch cut a 7-billion-parameter model's checkpoint pause from 148.8 to 6.3 seconds by copying state to host memory and writing it to storage in the background [14].
- Datasets are read again on every pass over the data [24]. In a 2021 study on V100 servers, models spent 10% to 70% of each pass over the data waiting on I/O when the dataset did not fit in memory [18]. In MLPerf Storage, a workload of large files needs about 38 times the read bandwidth of a workload of small files for each simulated B200, the benchmark's software stand-in for one GPU [7].
- Dataset sizes depend on the source and the format. The FineWeb card lists more than 18.5 trillion tokens of cleaned English web text in 54.8 TB of files, viewed 7 November 2026 [20], and one company's 2022 study reported compressed datasets of 2.93 PB to 29.18 PB for three recommendation models [19].
- Object storage reaches training code through the S3 API. MLPerf Storage v3.0 added an S3 access layer in September 2026, and about one-sixth of submissions used it [8]. Ceph serves S3-compatible buckets with versioning, a POSIX file system and block devices from one cluster [29, 30, 32, 33].

Background

Three kinds of data

Model training reads and writes three kinds of data. A dataset is the collection of samples a model learns from, such as tokens of text, images or records, stored as files. Training reads it again on every epoch, a full pass over the data, in randomly ordered batches [7, 24]. A checkpoint is the saved state of a training run, written at intervals so the run can resume after a failure. A checkpoint that can resume training holds the weights, the optimizer state and progress records such as the epoch [3]. The optimizer state is the set of values the optimizer keeps for each parameter. With mixed-precision Adam, it is a 32-bit copy of the weights, the momentum and the variance [1]. The weights are the trained parameters alone, which evaluation jobs and later training read [16, 17].

Distributed training splits this state across many processes, called ranks, that save and load in parallel [4]. A data stall is time spent waiting for data to be fetched and pre-processed [18].

Why storage belongs in the training plan

Infrastructure engineers at one model developer wrote in March 2024 that storage "plays an important role in AI training, and yet is one of the least talked-about aspects", and that storage needs grow as training jobs consume more image, video and text data [6]. In one company's recommendation model training, storage and online preprocessing can consume more power than the GPU trainers themselves [19]. For the 405-billion-parameter run, its developers aimed to minimize GPU pause time during checkpointing and to checkpoint more often, so that less work is lost after a recovery [5].

This report is for ML engineers, platform teams and buyers who size storage for a training cluster and want the arithmetic and the published measurements behind each requirement.

Method

What was examined. What datasets, checkpoints and weights ask of storage during model training: size, access pattern, read and write rates, how often checkpoints are written and restored, and the storage types that published clusters use.

Sources. Systems papers [1, 2, 10, 11, 12, 13, 15, 16, 17, 18, 19], reports and posts from teams that train large models [5, 6, 14], MLPerf Storage's page, release and rules [7, 8, 9], PyTorch and NVIDIA documentation [3, 4, 24, 25, 26, 27], dataset and format documentation [20, 21, 22, 23, 28] and Ceph's Umbrella documentation [29, 30, 31, 32, 33, 34, 35].

How. Each figure is taken from the passage, table or page section cited. Papers were read in full text, and web pages were viewed on 7 November 2026. Figures marked "computed for this report" are arithmetic on the cited numbers: unit conversions, divisions and the checkpoint interval formula restated in [10]. Write and read times are size divided by bandwidth, as MLPerf Storage defines duration [7], and leave out metadata work, contention and the cost of each fsync. Models are described by size and kind, never by family or vendor, and papers by title or subject.

Units. GB and TB are decimal units (1 GB is 1,000,000,000 bytes). GiB and TiB are binary units (1 GiB is 1,073,741,824 bytes). MLPerf Storage's checkpoint sizes are binary, as its rules state [9]. NVIDIA states its guidelines in GBps [24], read here as decimal gigabytes per second.

Findings

1. Three kinds of data, three access patterns

The three kinds of data reach storage in different ways.

Data	Written	Read	Size depends on	Sources
Datasets	Once per version	Again on every epoch, in randomly ordered batches	Samples or tokens, and the file format	[7, 20, 22, 24]
Checkpoints	At intervals, as files from every rank in parallel, each forced to durable media	After a failure by the whole job, and by evaluation and post-training jobs	Parameters, optimizer state and the parallel layout	[4, 7, 16]
Weights	As files, in the safetensors format with a header that lists each tensor	By evaluation and later training	Parameters times bytes per parameter	[1, 2, 16, 28]

NVIDIA's DGX SuperPOD storage guide says the key I/O operation in deep learning training is re-read, because the same data is reused on every pass, and expects most deep learning workloads to be read-dominant [24]. Checkpoint files can be terabytes in size and, while not written often, are typically written synchronously, which blocks training [24]. For LLM training the guide recommends write performance of at least half the read performance [24].

MLPerf Storage measures the two sides separately: a training benchmark of repeated random reads, and a checkpointing benchmark that writes the entire training state and reads it back as after a failure [7].

2. Dataset sizes depend on the source and the format

A cleaned text dataset. The FineWeb card describes more than 18.5 trillion tokens of cleaned and deduplicated English web data from Common Crawl, up from 15 trillion at first release, in 52,453,695,892 rows and 54.8 TB of files [20]. That is about 3 bytes of published files per token (computed for this report from [20]). The figures are as viewed on 7 November 2026, since the card grows with new crawls [20]. The 405-billion-parameter run pre-trained on 15.6 trillion text tokens [5].

A raw web capture. Common Crawl's October 2026 archive, crawled from 5 to 18 October, holds 2.16 billion web pages and 371.19 TiB of uncompressed content [21]. Its file list gives 100,000 WARC files of raw captures at 86.19 TiB compressed and 5.85 TiB of WET text files [21]. The compressed raw captures are about 23% of the uncompressed content, and the compressed text about 1.6% (computed for this report from [21]).

Recommendation datasets. A 2022 study of recommendation model training at one company describes a pipeline that stores exabytes of training data and serves it at tens of terabytes per second [19]. Its compressed datasets for three models were 13.45 PB, 29.18 PB and 2.93 PB, too large for local storage, so training jobs read samples from a central data warehouse [19].

Shards. WebDataset stores samples in tar files numbered as shards, and its authors write that purely sequential reads reach 3 to 10 times the rate of random access on local drives and suit object stores and cloud storage [22].

3. Reads stall GPUs when a dataset outgrows memory

A 2021 study of data stalls found that in many cases training time was dominated by time spent waiting for data to be fetched and pre-processed [18]. On servers with 500 GB of DRAM, only 35% of ImageNet-22K, 45% of the FMA dataset or 65% of OpenImages could be cached, although each fit on local storage [18]. On V100 servers with SSDs, models spent 10% to 70% of epoch time on blocking I/O despite pipelining and prefetching, because the GPUs computed faster than data arrived [18]. The page cache thrashed: for a 146 GiB dataset, each epoch fetched 85% of the data from storage where 65% should have been enough [18].

NVIDIA's guidance is to cache data on its first read, so later reads need not cross the network [24]. Shared file systems typically use RAM as the first cache layer, a cached read can be an order of magnitude faster than a read from remote storage, and the DGX B200's local NVMe can also cache or stage data [24]. The DGX B200 user guide lists two 1.92 TB NVMe drives in RAID 1 for the operating system and eight 3.84 TB NVMe drives in RAID 0 as a data cache [26]. That is 30.72 TB of local data cache per system (computed for this report from [26]). Storage traffic has its own cards, two dual-port BlueField-3 DPUs for storage and in-band management, apart from the cluster network's ConnectX-7 ports [26].

In the 2025 study's clusters, inputs, outputs and checkpoints are expected to be durable and decoupled from the lifetime of the job, and a read-only caching service backed by flash speeds up dataset access [10].

GPUDirect Storage moves data between GPU memory and storage by DMA without a bounce buffer in CPU memory, and on some systems that path offers at least twice the peak bandwidth of a path through the CPU [27]. It needs file system drivers that support it [27].

MLPerf Storage's rules require a dataset of at least five times the client nodes' total DRAM, and a result counts only if the simulated accelerators stay busy 90% of the time on Unet3D and 85% on RetinaNet [7, 9].

4. Published read and write rates per GPU differ widely

NVIDIA's DGX SuperPOD guide sets two performance levels. Standard covers multiple concurrent LLM or fine-tuning jobs with periodic checkpoints, where compute needs dominate data I/O. Enhanced covers multiple concurrent multimodal jobs with periodic checkpoints, where data I/O performance matters [24]. A scalable unit (SU) is 32 DGX B200 systems [25], 256 GPUs at eight per system (computed for this report from [25, 26]).

Scope	Standard read	Standard write	Enhanced read	Enhanced write
One SU, GBps [24]	40	20	125	62
Four SUs, GBps [24]	160	80	500	250
Per GPU in one SU, GBps (computed)	0.16	0.08	0.49	0.24

Per-GPU figures are computed for this report from [24, 25, 26]. For automotive and other computer vision training on high-resolution images, whose datasets easily exceed 30 TB, the same guide says 4 GBps of read performance per GPU is needed [24].

MLPerf Storage v3.0 shows the effect of file size. Its simulated accelerators compute by sleeping for the measured per-batch time of a B200 or an MI355 [7]. Unet3D reads files of about 140 MiB in full and in sequence, picking the next file effectively at random, and keeping one simulated B200 at full use takes about 5.9 GiB/s, a load set by bandwidth [7]. RetinaNet reads JPEG files of about 315 KiB, about 155 MiB/s or roughly 500 files a second per simulated B200, a load set by metadata and operations per second [7]. The Unet3D accelerator needs about 38 times the bandwidth of the RetinaNet one [7].

NVIDIA advises characterizing workloads before finalizing performance and capacity [24].

5. What a checkpoint holds and how big it is

PyTorch's tutorial says a checkpoint for resuming training must hold the optimizer's state as well as the model's, and may add the epoch, the last recorded training loss and other items, so it is often 2 to 3 times larger than the model alone [3]. CheckFreq adds the data loader's state through a resumable iterator [11].

The ZeRO paper counts the model states of mixed-precision training with Adam. The 16-bit weights and gradients take 2 bytes per parameter each, and the optimizer holds a 32-bit copy of the weights, the momentum and the variance at 4 bytes each, 16 bytes per parameter in total [1]. MLPerf Storage's checkpoints hold weights plus optimizer state [7], and their sizes match 2 bytes of weights plus 12 of optimizer state, without gradients (computed for this report):

Model size	Processes [7, 9]	Checkpoint, binary [9]	Decimal (computed)	Bytes per parameter (computed)	Files [7]
8 billion	8	105 GiB	112.7 GB	14.1	16

Model size	Processes [7, 9]	Checkpoint, binary [9]	Decimal (computed)	Bytes per parameter (computed)	Files [7]
70 billion	64	912 GiB	979 GB	14.0	128
405 billion	512	5.29 TiB	5.82 TB	14.4	1,544
1T in the rules, 1250B on the results page	1,024	18 TiB	19.8 TB	Not computed	2,072

Against 2 bytes per parameter for 16-bit weights, such a checkpoint is about seven times the weights (computed for this report from [1, 9]). PyTorch's 2 to 3 times describes a general checkpoint in a different setting [3].

Other published sizes span a wide range. In the 405-billion-parameter pre-training, each GPU's saved model state ranged from 1 MB to 4 GB [5]. CheckFreq put model state at a few hundred MB to a few hundred GB for the models of 2021 [11]. A 2024 datacenter study describes model states at terabyte scale across all GPUs [17], and NVIDIA's guide says checkpoint files can be terabytes in size [24].

Checkpoints are written as many files. PyTorch's distributed checkpoint saves and loads from multiple ranks in parallel, produces at least one file per rank and can reshard at load time, so a checkpoint saved in one cluster layout loads into another [4]. In MLPerf Storage, each process streams one or two large files of its shard, roughly 9 to 20 GiB depending on the model [7].

6. Failure rates set the checkpoint interval

The 405-billion-parameter pre-training had 466 job interruptions in a 54-day period, 419 of them unexpected [5]. Its authors reached higher than 90% effective training time, and reduced job startup and checkpointing time to do so [5].

The 2025 research-cluster study, with eight A100 GPUs per node, measured failure rates of 6.50 per thousand node-days on its cluster RSC-1 and 2.34 on RSC-2 [10]. It puts the mean time to failure (MTTF) at around 10 hours already at 4,000-GPU scale and projects 1.8 hours for 16,384-GPU jobs and 0.23 hours for 131,072-GPU jobs [10]. It restates the Daly and Young rule for the optimal checkpoint interval:

Interval ≈ √(2 × checkpoint write overhead ÷ (nodes in the job × failures per node-day))

The job's MTTF is the inverse of nodes times failure rate [10]. For a job on about 12,000 GPUs, 1,500 nodes of eight, at RSC-1's rate, the job sees 9.75 failures a day, an MTTF of about 2.5 hours. A 5-minute write overhead gives an optimal interval of about 38 minutes, and a 10-second overhead about 7 minutes (computed for this

report from [10]). The study's own conclusion for runs of that size: to reach an effective training time ratio (ETTR, productive runtime over available wallclock time, which includes time waiting in the queue) of 0.9, checkpoint write overhead needs to be on the order of 10 seconds, or the failure rate must fall from 6.50 to about 1 [10].

A long interval costs lost work. Check-N-Run's authors note that a checkpoint after 5,000 batches can waste five times the work of one after 1,000 batches in the worst case [12]. MegaScale reports catching up to the pre-crash training progress within 15 minutes from its checkpoints, with over 90% effective training time [13].

Published intervals vary with the method. GEMINI, a 2023 system that keeps its failure-recovery checkpoints in CPU memory, reaches a checkpoint every iteration [15]. The 2024 datacenter study tested asynchronous checkpointing at a 30-minute interval [17].

7. Checkpoint writes arrive as bursts

For the 405-billion-parameter pre-training, its developers built a storage fabric on their own distributed file system: 240 PB on 7,500 servers with SSDs, 2 TB/s sustained and 7 TB/s peak [5]. The authors call the highly bursty checkpoint writes, which saturate the fabric for short durations, a major challenge [5]. The same company's engineers describe thousands of GPUs saving and loading checkpoints in a synchronized fashion as "a challenge for any storage solution", and write that new storage solutions provided opportunities to optimize checkpointing across thousands of ranks to run in hundreds of milliseconds [6].

MLPerf Storage's checkpointing benchmark reproduces the burst. All processes stream their shards at once, with an fsync that forces every file to durable media before it counts, then repeat five seconds later, for 10 writes and then 10 reads of the same data [7, 9]. Write duration, from first byte to last fsync, is the time training would stall per checkpoint, and read duration is the storage part of recovery after a failure [7]. MLCommons notes that sustained write bandwidth bounded by fsync is harder to scale than read bandwidth [7].

At NVIDIA's write guidelines, MLPerf's checkpoint sizes take these times (computed for this report from [9, 24, 25], illustrative only):

Checkpoint	One SU Standard, 20 GBps	One SU Enhanced, 62 GBps	Four SUs Standard, 80 GBps	Four SUs Enhanced, 250 GBps
8 billion, 112.7 GB	5.6 s	1.8 s	1.4 s	0.45 s
70 billion, 979 GB	49 s	16 s	12 s	3.9 s

Checkpoint	One SU Standard, 20 GBps	One SU Enhanced, 62 GBps	Four SUs Standard, 80 GBps	Four SUs Enhanced, 250 GBps
405 billion, 5.82 TB	291 s	94 s	73 s	23 s

Write bandwidth and capacity bound checkpoint frequency, and when checkpoints sit on remote storage, as in many industrial settings, network bandwidth bounds it too [12]. Many files also load the metadata service. ByteCheckpoint's optimized HDFS reaches hundreds of petabytes, 10 TB/s of read and write bandwidth and about 100,000 metadata queries per second, and its authors address the metadata bottleneck that a high volume of distributed checkpoints intensifies [16].

8. Asynchronous, in-memory and smaller checkpoints

Several systems take the write off the training path. CheckFreq splits checkpointing into a snapshot phase and a persist phase, with runtime overhead within 3.5% [11]. PyTorch's asynchronous checkpointing first copies each rank's data from GPU to CPU, the downtime the user sees, 6 to 14 seconds for models of 7 to 13 billion parameters, then copies it from CPU memory to disk in the background to persist it [14]. A 7-billion-parameter model's checkpoint downtime fell from an average of 148.8 seconds to 6.3 seconds [14]. The authors state that the method does not compress the serialization time and moves the final step to CPU threads [14].

MegaScale also works in two stages. Each GPU worker writes its state to host memory in several seconds and continues training, and a background process then copies the state to HDFS [13]. GEMINI checkpoints to the CPU memory of the host machines, whose aggregate bandwidth exceeds that of remote storage, and recovers more than 13 times faster than existing solutions [15]. Its authors add that checkpoints in CPU memory need placement, since their availability cannot be guaranteed when failures occur, and that checkpoint traffic can interfere with training traffic [15].

Against existing open-source checkpointing systems, ByteCheckpoint reports an average 54.20-times reduction in runtime checkpoint stalls and up to 9.96 times faster saving and 8.80 times faster loading [16]. The 2024 datacenter study reduced checkpoint time and overhead for 7-billion and 123-billion-parameter models by 3.6 to 58.7 times at a 30-minute interval, a figure that leaves out the time to persist to storage [17].

Smaller checkpoints are another route. For recommendation models at one company, Check-N-Run's differential checkpointing and quantization cut the write bandwidth needed by 6 to 17 times and the capacity by 2.5 to 8 times [12]. Its method relies on only part of such a model changing between checkpoints, which is the paper's setting [12].

In PyTorch's asynchronous checkpointing and in MegaScale, the full state still reaches storage. The pause shrinks, and the storage write moves to the background [13, 14]. GEMINI keeps its failure-recovery checkpoints in CPU memory and puts checkpoints for other uses, such as transfer learning and model debugging, in remote persistent storage [15].

9. Restores and the other readers of checkpoints

A restore is a read burst. While a checkpoint is restored, the whole cluster waits [7]. MegaScale names the distributed file system's bandwidth as the bottleneck of recovery when every GPU worker reads its own partition, so one worker per data-parallel group reads the shared partition and broadcasts it to the rest, which reduces the load linearly [13].

Checkpoints are read for other reasons too. ByteCheckpoint's authors write that checkpoints serve concurrent evaluation tasks and move from pre-training to downstream post-training, and that resharding during training is common [16]. They observe that evaluation tasks download new checkpoint files shortly after creation, that access frequency then falls when no training anomaly such as a loss spike occurs, and that all checkpoint files must be preserved for traceability [16]. Their answer is a hot and cold tier of SSD and hard drive servers, with files migrating from SSD to HDD [16]. In the 2024 datacenter study, an evaluation job on HumanEval spent over a minute loading model checkpoints and preprocessing data before GPU inference began, 29.5% of its duration [17].

10. Weights: size by precision, format and versions

Weights alone take 2 bytes per parameter in 16-bit formats [1]. The 8-bit floating point formats proposed in 2022, E4M3 and E5M2, take one byte [2]. A 70-billion-parameter model is therefore about 140 GB at 16 bits and about 70 GB in FP8 (computed for this report from [1, 2]).

The safetensors format opens with an 8-byte header length and a JSON header listing each tensor's data type, shape and data offsets [28]. Its authors compare formats on safety, whether a downloaded file can be used without running arbitrary code, and on lazy loading, whether a file can be inspected without loading everything [28].

Versions keep runs traceable. FineWeb's changelog starts at v1.0.0 on 21 April 2024 and keeps previous versions available in branches [20]. In Ceph's S3 API, enabling versioning on a bucket gives every object added a unique version ID, and suspending it gives new objects the version ID null [31].

11. Storage types in published clusters, the S3 API and Ceph

Cluster or system	Storage reported	Sources
One operator's research clusters, A100 nodes	A POSIX file system on flash over NFS, a read-only dataset cache on flash and a high-capacity object store	[10]
The same operator's two 24,576-GPU clusters	A flash-optimized distributed store behind a FUSE API for checkpoints and data loading, at exabyte scale, plus a parallel NFS so code changes reach every node	[6]
A 2024 datacenter study	An all-NVMe shared parallel file system, and an extra network adapter for storage on one cluster's nodes	[17]
MegaScale	HDFS for checkpoints	[13]
ByteCheckpoint	HDFS, with SSD and hard drive tiers	[16]

The S3 API in training code. Amazon's S3 connector for PyTorch supports map-style datasets for random access and iterable-style datasets for streaming, and saves and loads checkpoints directly to Amazon S3 without writing to local storage first [23]. MLPerf Storage v3.0, released on 1 September 2026, added an S3 access layer beside its POSIX layer for training, checkpointing and some vector database tests, and about one-sixth of the submissions from 19 organizations used it [8]. In that release, co-chair Curtis Anderson said "we expect object-based storage systems to emerge as a viable" alternative to file system storage, possibly the preferred one [8].

Ceph's three interfaces. The Ceph Object Gateway is a RESTful gateway with an interface compatible with a large subset of the Amazon S3 API [29]. Its feature table lists bucket object versions, multipart uploads, bucket lifecycle and bucket and object policies as supported, and bucket replication as partial, permitted only across zones [30]. CephFS is a POSIX-compliant file system for uses such as shared home directories, HPC scratch space and shared storage for distributed workflows [32]. Its metadata sits in a separate pool served by metadata servers, and clients read and write file data directly, with no gateway in the data path [32]. Ceph block devices are thin-provisioned and resizable, stripe their data across the cluster and support snapshots, replication and strong consistency [33].

Protection and hardware. Ceph pools are replicated by default, with every object copied to several disks, and erasure coding splits data into data and parity blocks and saves space at scale [34]. The default erasure-code profile needs 2 TB of raw space to store 1 TB, where a replicated pool of size three needs 3 TB [34]. Ceph's hardware guide asks for at least 10 Gb/s networking, suggests 25 Gb/s for substantial workloads and notes that dense nodes often warrant 100 Gb/s [35].

Implications

For training teams. Size the checkpoint from parameters and bytes per parameter. With mixed-precision Adam and no gradients saved, MLPerf's sizes come to about 14 bytes per parameter [1, 9]. Count files per rank as well as bytes, since a high volume of checkpoint files strains the metadata service [4, 16]. Measure the write time, then set the interval from the failure rate expected at the job's scale with the Daly and Young rule [10]. Save the data loader's state with the weights and optimizer state, as CheckFreq does [11]. Plan the restore as a read burst, and share reads within data-parallel groups where the framework allows it [13].

For storage design. Treat each kind of data as its own workload. Dataset reads depend on file size: large files need bandwidth, small files need metadata and operations per second [7]. Sharding small samples into large files helps [22], and a cache on first read or local NVMe staging takes repeated reads off the network [24]. Checkpoint writes need burst bandwidth that holds under fsync [7]. Capacity has to hold the checkpoints kept for evaluation and traceability, with fast and capacity tiers as ByteCheckpoint describes [16]. Versioned buckets give each dataset or weights file a version ID that a run can record [31], and a shared file system makes code changes reach every node at once, as one operator's parallel NFS does [6].

For buyers comparing storage. Ask which MLPerf Storage workload a result covers, training or checkpointing, over POSIX or S3, and how many simulated accelerators it kept busy [7, 8, 9]. Ask for write bandwidth measured with fsync [7], and match the guideline level to the work, LLM training or multimodal training on large datasets [24].

Limits of this analysis

- The data stall figures come from 2020-era V100 servers [18]. Check-N-Run's figures are for recommendation models at one company [12], and the 2022 study's dataset sizes are for recommendation models [19].
- No MLPerf Storage result rows for specific systems were read, and this report measures and compares no storage system, Ceph included.
- The fourth MLPerf model size is 1T in the rules and 1250B on the results page [7, 9], so no per-parameter figure is computed for it.
- The 405B checkpoint is 5.29 TB in the rules, stated there as binary (5.29 TiB), and about 5,290 GiB on the results page [7, 9]. This report uses the rules' figure (14.4 bytes per parameter, or 14.0 from the results page).
- FineWeb's figures are as viewed on 7 November 2026 and grow with new crawls [20].
- Weight sizes below 8 bits are outside this report's sources.
- The Daly and Young rule is taken as restated by the 2025 research-cluster study [10]. The original papers were not reviewed.
- The S3 connector for PyTorch is documented for Amazon S3 [23]. No source reviewed tests it against another S3-compatible store.
- No source reviewed compares the speed of object storage and file systems for training. The MLCommons view is an expectation stated on 1 September 2026 [8].
- How long checkpoints are kept is reported only for ByteCheckpoint's setting [16].

Sources

1. ZeRO: Memory Optimizations Toward Training Trillion Parameter Models. Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase and Yuxiong He, arXiv:1910.02054, version 3, 13 May 2020 (first version 4 October 2019), viewed 7 November 2026. https://arxiv.org/abs/1910.02054

2. FP8 Formats for Deep Learning. Paulius Micikevicius and others, arXiv:2209.05433, version 2, 29 September 2022 (first version 12 September 2022), viewed 7 November 2026. https://arxiv.org/abs/2209.05433

3. Saving and Loading Models. PyTorch Tutorials, PyTorch, last updated 26 June 2025, viewed 7 November 2026. https://docs.pytorch.org/tutorials/beginner/saving_loading_models.html

4. Distributed Checkpoint: torch.distributed.checkpoint. PyTorch 2.15 documentation, PyTorch, last updated 8 July 2026, viewed 7 November 2026. https://docs.pytorch.org/docs/2.15/distributed.checkpoint.html

5. The Llama 3 Herd of Models. Llama Team, AI @ Meta, arXiv:2407.21783, version 3, 23 November 2024 (paper dated 23 July 2024), viewed 7 November 2026. https://arxiv.org/abs/2407.21783

6. Building Meta's GenAI Infrastructure. Kevin Lee, Adi Gangidi and Mathew Oldham, Meta Engineering, 12 March 2024, viewed 7 November 2026. https://engineering.fb.com/2024/03/12/data-center-engineering/building-metas-genai-infrastructure/

7. Benchmark MLPerf Storage (v3.0 results page). MLCommons, viewed 7 November 2026. https://mlcommons.org/benchmarks/storage/

8. MLCommons Releases New MLPerf Storage v3.0 Benchmark Results. MLCommons, 1 September 2026, viewed 7 November 2026. https://mlcommons.org/2026/09/mlperf-storage-v3-0-results/

9. MLPerf Storage Rules (Rules.md, rules edition 3.0). MLCommons, GitHub repository mlcommons/storage, viewed 7 November 2026. https://github.com/mlcommons/storage/blob/main/Rules.md

10. Revisiting Reliability in Large-Scale Machine Learning Research Clusters. Apostolos Kokolis and others, 2025 IEEE International Symposium on High-Performance Computer Architecture (HPCA), arXiv:2410.21680, version 2, 6 February 2025, viewed 7 November 2026. https://arxiv.org/abs/2410.21680

11. CheckFreq: Frequent, Fine-Grained DNN Checkpointing. Jayashree Mohan, Amar Phanishayee and Vijay Chidambaram, 19th USENIX Conference on File and Storage Technologies (FAST 21), February 2021, viewed 7 November 2026. https://www.usenix.org/system/files/fast21-mohan.pdf

12. Check-N-Run: a Checkpointing System for Training Deep Learning Recommendation Models. Assaf Eisenman and others, 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22), April 2022, viewed 7 November 2026. https://www.usenix.org/system/files/nsdi22-paper-eisenman.pdf

13. MegaScale: Scaling Large Language Model Training to More Than 10,000 GPUs. Ziheng Jiang and others, arXiv:2402.15627, 23 February 2024 (NSDI 2024), viewed 7 November 2026. https://arxiv.org/abs/2402.15627

14. Reducing Model Checkpointing Times by Over 10x with PyTorch Distributed Asynchronous Checkpointing. Lucas Pasqualin, Less Wright, Iris Zhang, Chien-Chin Huang, Swaminathan Sundararaman, Saransh Gupta and Raghu Ganti, PyTorch blog, 12 June 2024, viewed 7 November 2026. https://pytorch.org/blog/reducing-checkpointing-times/

15. GEMINI: Fast Failure Recovery in Distributed Training with In-Memory Checkpoints. Zhuang Wang, Zhen Jia, Shuai Zheng, Zhen Zhang, Xinwei Fu, T. S. Eugene Ng and Yida Wang, ACM SOSP '23, October 2023, author copy at Rice University, viewed 7 November 2026. https://www.cs.rice.edu/~eugeneng/papers/SOSP23.pdf

16. ByteCheckpoint: A Unified Checkpointing System for Large Foundation Model Development. Borui Wan and others, arXiv:2407.20143, version 4, 2 April 2025 (accepted by NSDI '25), viewed 7 November 2026. https://arxiv.org/abs/2407.20143

17. Characterization of Large Language Model Development in the Datacenter. Qinghao Hu and others, arXiv:2403.07648, version 2, 3 April 2024, viewed 7 November 2026. https://arxiv.org/abs/2403.07648

18. Analyzing and Mitigating Data Stalls in DNN Training. Jayashree Mohan, Amar Phanishayee, Ashish Raniwala and Vijay Chidambaram, PVLDB 14(5), 2021, arXiv:2007.06775, version 3, 19 January 2021, viewed 7 November 2026. https://arxiv.org/abs/2007.06775

19. Understanding Data Storage and Ingestion for Large-Scale Deep Recommendation Model Training. Mark Zhao and others, ISCA '22 Industry Track, arXiv:2108.09373, version 4, 22 April 2022, viewed 7 November 2026. https://arxiv.org/abs/2108.09373

20. FineWeb dataset card (HuggingFaceFW/fineweb). Hugging Face, viewed 7 November 2026. https://huggingface.co/datasets/HuggingFaceFW/fineweb

21. October 2026 Crawl Archive Now Available. Common Crawl, 21 October 2026, viewed 7 November 2026. https://commoncrawl.org/blog/october-2026-crawl-archive-now-available

22. WebDataset (README). WebDataset project, GitHub, viewed 7 November 2026. https://github.com/webdataset/webdataset

23. Amazon S3 Connector for PyTorch (README). Amazon Web Services Labs, GitHub, viewed 7 November 2026. https://github.com/awslabs/s3-connector-for-pytorch

24. Storage Architecture. NVIDIA DGX SuperPOD Reference Architecture Featuring NVIDIA DGX B200, NVIDIA, viewed 7 November 2026. https://docs.nvidia.com/dgx-superpod/reference-architecture-scalable-infrastructure-b200/latest/storage-architecture.html

25. Key Components of the DGX SuperPOD. NVIDIA DGX SuperPOD Reference Architecture Featuring NVIDIA DGX B200, NVIDIA, viewed 7 November 2026. https://docs.nvidia.com/dgx-superpod/reference-architecture-scalable-infrastructure-b200/latest/dgx-superpod-components.html

26. Introduction to the NVIDIA DGX B200 System. NVIDIA DGX B200 User Guide, NVIDIA, viewed 7 November 2026. https://docs.nvidia.com/dgx/dgxb200-user-guide/introduction-to-dgxb200.html

27. Magnum IO GPUDirect Storage Overview Guide. NVIDIA, viewed 7 November 2026. https://docs.nvidia.com/gpudirect-storage/overview-guide/index.html

28. safetensors (README). Hugging Face, GitHub, viewed 7 November 2026. https://github.com/huggingface/safetensors

29. Ceph Object Gateway. Ceph documentation (Umbrella), viewed 7 November 2026. https://docs.ceph.com/en/umbrella/radosgw/

30. Ceph Object Gateway S3 API. Ceph documentation (Umbrella), viewed 7 November 2026. https://docs.ceph.com/en/umbrella/radosgw/s3/

31. Ceph Object Gateway S3 API: Bucket operations. Ceph documentation (Umbrella), viewed 7 November 2026. https://docs.ceph.com/en/umbrella/radosgw/s3/bucketops/

32. Ceph File System. Ceph documentation (Umbrella), viewed 7 November 2026. https://docs.ceph.com/en/umbrella/cephfs/

33. Ceph Block Device. Ceph documentation (Umbrella), viewed 7 November 2026. https://docs.ceph.com/en/umbrella/rbd/

34. Erasure code. Ceph documentation (Umbrella), viewed 7 November 2026. https://docs.ceph.com/en/umbrella/rados/operations/erasure-code/

35. Hardware Recommendations. Ceph documentation (Umbrella), viewed 7 November 2026. https://docs.ceph.com/en/umbrella/start/hardware-recommendations/

About this report

Published by Rillor on 8 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. Storage for model training: datasets, checkpoints and weights. 8 November 2026. https://rillor.com/insights/storage-for-model-training

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 what the storage must hold, how fast it grows, how it is read and written, and where it runs. Contact us at rillor.com/contact?type=storage or contact@rillor.com. More on distributed storage: rillor.com/compute/storage