

On 23 September 2026 SemiAnalysis, an industry research firm, published the third edition of ClusterMAX, its rating of the "neoclouds" that rent out AI accelerators. It covers 77 providers, and it finds that tests of the chips' own matrix arithmetic rarely tell one provider from another. What does separate them, on its account, is mostly what surrounds the chips — storage, networking, support, security and, above all for many large customers, reliability. Once a job outgrows a single chip, in memory or in time, the work has to be split, and there are three ways to split it — by copying the model, by cutting each layer, and by dealing the layers out along a pipeline. Each lets thousands of chips work on one job, and each obliges them to stop and exchange results at regular intervals. A synchronous job moves at the pace of its slowest participant, over two roads of very different speed: fast links inside a server or rack, and a network perhaps a ninth as fast between them. Published figures, each from the organisation that ran or recomputed the run, put the share of the chips' peak arithmetic that became model at 46.2% for Google's PaLM and 38% to 43% for Meta's Llama 3, with higher figures in some scaling experiments. On that record, the count of chips says little on its own.

1. A rating that does not start from the chip count

ClusterMAX 3.0 appeared on SemiAnalysis's newsletter on 23 September 2026 under the by-line of Jordan Nanos, Sam Harshe, Samuel Kruse "and 4 others". It rates 77 providers, tracks 323, and sets out criteria "across 10 categories". The clusters it tests by hand go through three phases: an audit of whether each is "setup properly", a performance phase, and a reliability phase. Only 19 providers earn what the firm calls a Medallion rating.

The performance phase includes the obvious test — how fast each accelerator multiplies matrices, the operation at the core of every neural network — and the firm's verdict on it is short:

Although GEMMs are the heart of modern AI workloads, these microbenchmarks very rarely distinguish providers—it would be hard for a provider to mess up this functionality

(GEMM, "general matrix multiply", is the name of the routine.) Among its storage benchmarks it calls one, the fio sweep, "the test that finds the most issues"; its subtitle lists "reliability, performance, support, pricing—and, of course, security"; hands-on testing, it adds, "is only a part of the overall ratings". On NVIDIA's 72-GPU racks the firm runs some tests "with NVLink disabled ... in order to isolate the performance of the scale-out network", and it describes that network as "the last big design decision a provider still owns". Reliability, it writes, "is the #1 most important criteria to many of the biggest customers in the world", and of the managed-cluster business it writes that "TCO comes down to goodput, and goodput comes down to reliability" — goodput being, roughly, the useful work a cluster delivers once failures, restarts and lost progress are subtracted.

The rating has limits that belong in the same breath as its findings. SemiAnalysis sells data and models to the industry it rates, and the post advertises them. Its test allocations are small; in its own words, "genuine hardware failures are rare in testing because our ClusterMAX allocations are too small and short-lived". So ClusterMAX measures set-up, networking and remediation on clusters of a few nodes, not the utilisation of a 16,000-GPU training run. Meta's published account of its Llama 3 run shows the same pressures from another side: confirmed or suspected hardware faults interrupted it more than three hundred times in 54 days, and its per-chip output fell as the job was spread wider.

2. Two readings that mislead

2.1 The count

The first misleading reading is arithmetic: that 16,384 accelerators do 16,384 times the work of one. Meta's paper on Llama 3 (arXiv 2407.21783, first posted 31 July 2024) reports the throughput of each GPU at three stages of training its largest model:

Figure 1. Llama 3 405B pre-training: useful arithmetic per GPU at three cluster configurations (BF16 model FLOPs utilisation)



Source: Meta, 'The Llama 3 Herd of Models', arXiv 2407.21783, Table 4 and section 3.3.2 (version 3, 23 November 2024, read 25 September 2026). Model FLOPs utilisation (MFU) is useful model arithmetic as a share of the chips' peak. The paper attributes the fall from 43% to 41% to 'the lower batch size per DP group needed to keep the global tokens per batch constant'. The figures are Meta's own; no outside party measured the run.

▼ Table view

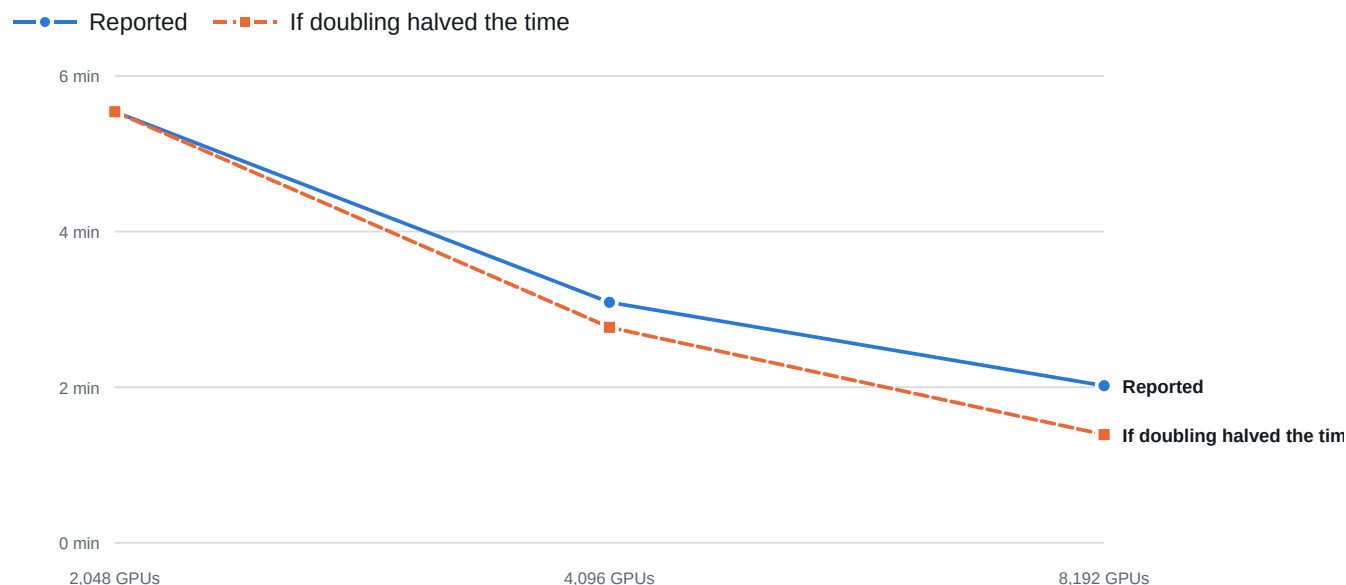
Figure 1. Llama 3 405B pre-training: useful arithmetic per GPU at three cluster configurations (BF16 model FLOPs utilisation)

Configuration	MFU
8,192 H100s, 8K-token sequences (430 TFLOPS per GPU)	43%
16,384 H100s, 8K-token sequences (400 TFLOPS per GPU)	41%
16,384 H100s, 131K-token sequences (380 TFLOPS per GPU)	38%

Doubling the cluster from 8,192 to 16,384 chips lowered each chip's useful output from 430 to 400 trillion operations a second. Nothing had broken. The paper traces the fall to how the work had to be divided once there were twice as many copies of the model sharing a fixed batch. A second illustration comes from the MLPerf Training benchmark of June 2026, where CoreWeave, a cloud provider, reported its time on one task falling from 5.54 minutes on 2,048 GPUs to 2.02 minutes on 8,192. CoreWeave called that "near-linear

scaling efficiency". Four times the chips bought 2.7 times the speed, and the statement does not say where the shortfall went. The numbers are CoreWeave's own, published by MLCommons with the disclaimer that such statements "do not reflect the opinions or views of MLCommons".

Figure 2. Time to train one MLPerf benchmark as the cluster grows: CoreWeave's reported times against perfect scaling



Source: CoreWeave's submitter statement in MLCommons, MLPerf Training v6.0 supplemental statements, 16 June 2026 (DeepSeek-V3 671B benchmark on NVIDIA GB300 NVL72); 'near-linear' is CoreWeave's word. Each doubling bought less: 1.79x, then 1.53x; 2.74x over the whole fourfold increase. MLCommons publishes these statements without endorsing them. MLPerf times measure the time to a benchmark's quality target, not a full training run: 2.02 minutes on 8,192 GPUs is about 276 GPU-hours, against the 2.788 million H800 GPU-hours DeepSeek reported for training the model.

▼ Table view

Figure 2. Time to train one MLPerf benchmark as the cluster grows: CoreWeave's reported times against perfect scaling

Cluster size	Reported	If doubling halved the time
2,048 GPUs	5.5 min	5.5 min
4,096 GPUs	3.1 min	2.8 min
8,192 GPUs	2.0 min	1.4 min

2.2 The rack as one machine

The second misleading reading is architectural: that the chips in a rack, or a building, merge into one device. NVIDIA's page for its GB200 NVL72 rack says it "boasts a 72-GPU NVIDIA NVLink™ domain that acts as a single, massive GPU". Its NVLink page goes further, saying that NVLink connections "can be extended across nodes to create a seamless, high-bandwidth, multi-node GPU cluster—effectively forming a data-center-sized GPU". The same page's specification table lists NVLink domains of 8 and 72 GPUs.

Those are the vendor's descriptions of a rack. What the specifications describe is two roads. Inside a GB300 NVL72 rack each GPU's NVLink connections carry about 1.8 terabytes a second, counting both directions (130 TB/s across 72 GPUs). Beyond the rack, each GPU has a network card that NVIDIA rates at "800 gigabits per second (Gb/s) of network connectivity for each GPU" — if, as is conventional for network cards, that is per direction: 100 gigabytes a second each way, 200 counting both, roughly a ninth of each GPU's 1.8 TB/s NVLink figure. Anything that must cross from one rack to another takes the slower road. Microsoft's description of its "Fairwater" AI datacentres (12 November 2025) illustrates how easily the two get confused: it says "each rack provides 1.8 TB of GPU-to-GPU bandwidth", which is NVIDIA's per-GPU figure, not a rack's, and a quantity rather than a rate.

3. The idea: three ways to split the work, and why every chip waits

3.1 Room and time

There are two separate reasons to use more than one chip. The first is room. GLM-5.3, an open model with about 753 billion parameters, needs roughly 750 gigabytes merely to hold its weights at eight bits each; the largest accelerators in service carry a few hundred gigabytes of memory apiece. The second is time. Meta's paper puts the pre-training compute of Llama 3 405B at 3.8×10^{25} floating-point operations. An H100 running at its dense sixteen-bit peak of about 989 trillion operations a second would need roughly 1,200 years to do that much arithmetic; at the 400 trillion a second Meta's chips actually sustained, about 3,000. On 16,384 chips at that rate the arithmetic alone takes about 67 days — an estimate from the paper's own figures, since the paper does not state the run's total duration.

3.2 Data parallelism: many copies, one model

The simplest answer is to copy the model. Every chip holds a complete replica, reads a different slice of the training examples, and computes how its replica ought to change. Jeffrey Dean and colleagues at Google described the arrangement in "Large Scale Distributed Deep Networks" (NIPS 2012), whose DistBelief framework "supports data parallelism, where multiple replicas of a model are used to optimize a single objective". DistBelief ran on clusters of ordinary processors — "tens of thousands of CPU cores" — and the first of its two training methods, Downpour SGD, was "an asynchronous stochastic gradient descent procedure", in which replicas did not wait for each other.

The large language-model runs of recent years use the synchronous version. Alex Krizhevsky, then at Google, stated its cost in 2014 ("One weird trick for parallelizing convolutional neural networks", arXiv 1404.5997): in data parallelism "the workers must synchronize model parameters (or parameter gradients) to ensure that they are training a consistent model". Every replica finishes its share of a step, the replicas pool and average what they have computed, and only then does anyone begin the next step. Priya Goyal and colleagues at Facebook showed in 2017 that the synchronous form could be pushed a long way: their implementation "achieves ~90% scaling efficiency when moving from 8 to 256 GPUs" (arXiv 1706.02677). Krizhevsky also named the obvious escape and its price: data parallelism can be made "arbitrarily efficient" by enlarging the batch, "but very big batch sizes adversely affect the rate at which SGD converges as well as the quality of the final solution". The pooling itself is usually done with an "all-reduce", in which each chip

ends up holding the sum of everyone's contributions. Uber's Horovod paper (arXiv 1802.05799, February 2018) records that "in early 2017 Baidu published an article" promoting a ring version for deep learning, and cites earlier work by Patarasuk and Yuan that "suggest[s] that this algorithm is bandwidth-optimal".

3.3 Tensor parallelism: cutting each layer

When a model is too large for one chip, copying it is not enough; the model itself must be divided. Tensor parallelism divides each layer. A layer's weights form grids of numbers, and a matrix multiplication can be cut into pieces that different chips compute side by side. Mohammad Shoeybi and colleagues at NVIDIA set out a scheme for transformer language models in "Megatron-LM" (arXiv 1909.08053, September 2019), which for a layer's feed-forward block "requires only a single all-reduce operation in the forward pass ... and a single all-reduce in the backward pass". That is economical, but it is still an exchange inside every layer of every step, and so in practice it is kept on the fastest links available. The paper trained models of up to 8.3 billion parameters on 512 GPUs, sustaining "15.1 PetaFLOPs across the entire application with 76% scaling efficiency when compared to a strong single GPU baseline that sustains 39 TeraFLOPs, which is 30% of peak FLOPs". Its 32 DGX-2H servers held 16 GPUs each, joined inside the server at "300 GB/sec", with "100 GB/sec of interconnect bandwidth between servers".

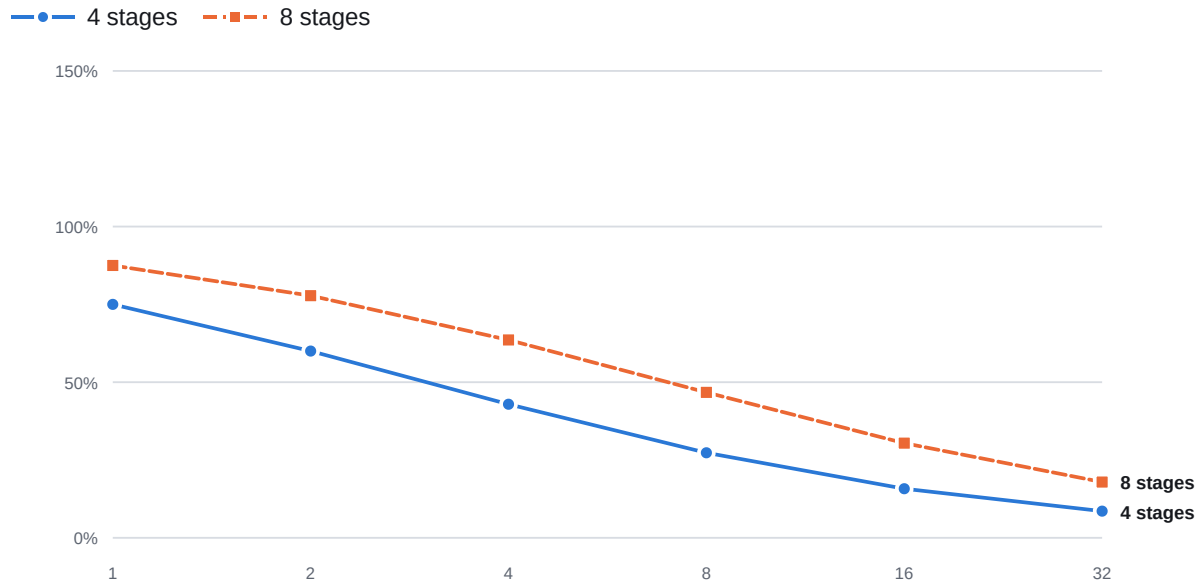
3.4 Pipeline parallelism, and the bubble

Pipeline parallelism divides the model the other way: the first layers on one group of chips, the next layers on the next, like stations on an assembly line. Yanping Huang and colleagues at Google described it in "GPipe" (arXiv 1811.06965, first posted in November 2018; fifth version, July 2019), and named its cost:

As illustrated in Figure 2c, partitioning introduces some idle time per accelerator, which we refer to as the bubble overhead.

At the start of each batch the later stations have nothing to do until work reaches them; at the end, the early ones have nothing left. GPipe's remedy is to cut each batch into many "micro-batches", which keeps the line fuller; the paper gives the bubble as being of the order of $(K-1)/(M+K-1)$ for K stages and M micro-batches, and reports: "In our experiments, we found the bubble overhead to be negligible when $M \geq 4 \times K$." The finding is empirical and, by the paper's own account, holds "partly because re-computation during the backward pass can be scheduled earlier". The formula alone does not reach zero:

Figure 3. The pipeline bubble: share of each step a simple pipeline spends idle, by number of micro-batches



Computed from the idle fraction $(K-1)/(M+K-1)$ for K stages and M micro-batches, the form GPipe (Huang et al., arXiv 1811.06965, section 2.3) gives for the bubble's order. Real schedules differ: GPipe reports the overhead negligible when M is at least $4K$, partly owing to recomputation overlap; Narayanan et al. (2021) express the bubble as $(p-1)/m$ of ideal time and cut it with interleaved schedules; DeepSeek's DualPipe (2024) reports fewer bubbles still. At 16 stages and $M = 4K = 64$ the formula still gives about 19%.

▼ Table view

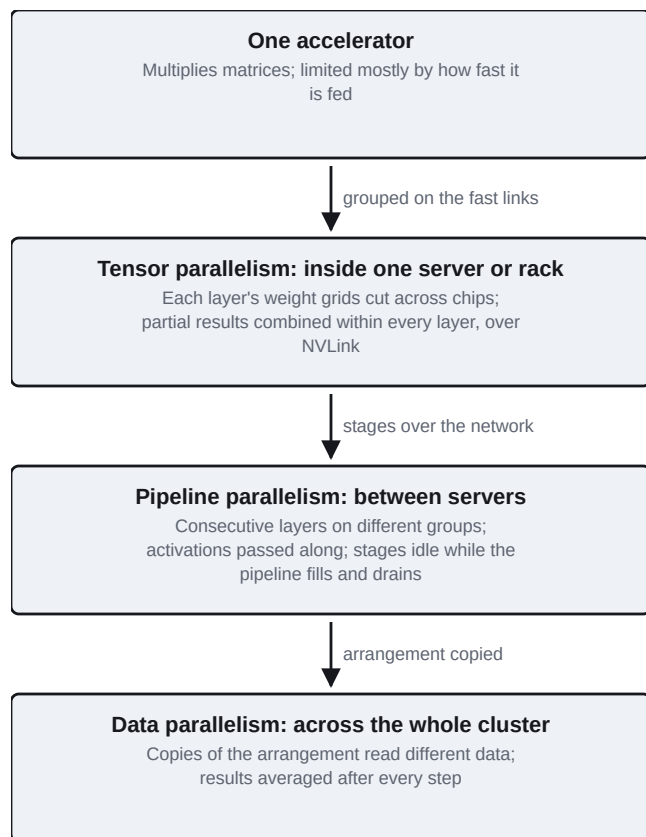
Figure 3. The pipeline bubble: share of each step a simple pipeline spends idle, by number of micro-batches

Micro-batches per batch (M)	4 stages	8 stages
1	75%	87.5%
2	60%	77.8%
4	42.9%	63.6%
8	27.3%	46.7%
16	15.8%	30.4%
32	8.6%	17.9%

3.5 How the three are combined

The three methods are not rivals. Deepak Narayanan and colleagues at NVIDIA, Stanford and Microsoft Research ("Efficient Large-Scale Language Model Training on GPU Clusters Using Megatron-LM", arXiv 2104.04473, SC21) described how to layer them, leveraging "the combination of pipeline parallelism across multi-GPU servers, tensor parallelism within a multi-GPU server, and data parallelism". Their rule of thumb follows directly from the two roads: "tensor model parallelism should generally be used up to degree g when using g -GPU servers, and then pipeline model parallelism can be used to scale up to larger models across servers". The arrangement ran "training iterations on a model with 1 trillion parameters at 502 petaFLOP/s on 3072 GPUs (per-GPU throughput of 52% of theoretical peak)"; the authors estimated that

training it to completion would take about three months, and the paper reports iteration throughput rather than a completed run. The 52% rests on the paper's count of floating-point operations, which, it notes, "assumes activation recomputation", and so includes arithmetic a stricter measure would leave out. Meta's Llama 3 paper states the rule as engineering practice — "The innermost parallelism requires the highest network bandwidth and lowest latency, and hence is usually constrained to within the same server" — and follows it: tensor parallelism eight ways inside each eight-GPU server, where "the eight GPUs are connected via NVLink", pipeline parallelism sixteen ways, and data parallelism across the rest.

Figure 4. Where each kind of split lives: the fastest exchanges on the shortest road

Schematic of the layering Narayanan et al. (arXiv 2104.04473, 2021) recommend: tensor parallelism within a multi-GPU server, pipeline parallelism across servers, data parallelism on top. Llama 3 405B used tensor parallelism of 8 within eight-GPU NVLink servers, pipeline parallelism of 16 and data parallelism of 8 to 128 (Meta, arXiv 2407.21783, Table 4), plus context parallelism for long sequences.

▼ Table view

Figure 4. Where each kind of split lives: the fastest exchanges on the shortest road — stages

#	Stage	Note
1	One accelerator	Multiplies matrices; limited mostly by how fast it is fed
2	Tensor parallelism: inside one server or rack	Each layer's weight grids cut across chips; partial results combined within every layer, over NVLink
3	Pipeline parallelism: between servers	Consecutive layers on different groups; activations passed along; stages idle while the pipeline fills and drains
4	Data parallelism: across the whole cluster	Copies of the arrangement read different data; results averaged after every step

Figure 4. Where each kind of split lives: the fastest exchanges on the shortest road — connections

From	To	Label
One accelerator	Tensor parallelism: inside one server or rack	grouped on the fast links
Tensor parallelism: inside one server or rack	Pipeline parallelism: between servers	stages over the network
Pipeline parallelism: between servers	Data parallelism: across the whole cluster	arrangement copied

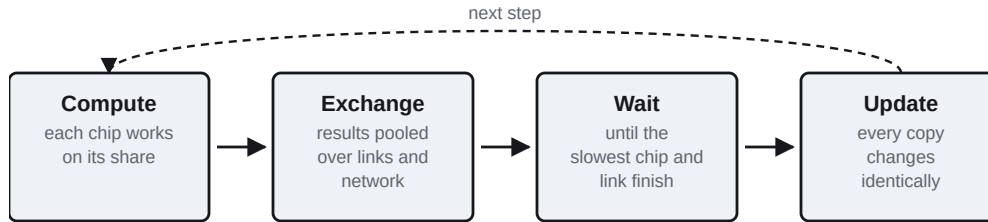
3.6 Waiting for the slowest

Every one of these arrangements ends each step in an exchange, and an exchange cannot finish before its slowest member. That parallel machines have limits is an old observation. Gene Amdahl, making the case for faster single processors in April 1967, argued that the serial part of a job caps what parallel hardware can buy: "the effort expended on achieving high parallel processing rates is wasted unless it is accompanied by achievements in sequential processing rates of very nearly the same magnitude"; the paper, its reprint's editors note, "has no equations", and the formula now called Amdahl's law came later. The barrier itself — every component finishing before the next step — was formalised in Leslie Valiant's bulk-synchronous parallel model (Communications of the ACM, August 1990), whose superstep rule is, in effect, the rule synchronous training follows: "After each period of L time units, a global check is made to determine whether the superstep has been completed by all the components." Dean and colleagues gave the machine-learning version in 2012, in their discussion of model parallelism:

The typical cause of less-than-ideal speedups is variance in processing times across the different machines, leading to many machines waiting for the single slowest machine to finish a given phase of computation.

A year later Dean and Luiz André Barroso ("The Tail at Scale", Communications of the ACM, February 2013) used a hypothetical web service to show how scale amplifies rare slowness: if each server is slow one time in a hundred and a request must wait for 100 of them, "63% of user requests will take more than one second". Training is the extreme case, because a synchronous job needs every participant at every step. Meta's paper puts it plainly:

Moreover, the synchronous nature of training makes it less fault-tolerant—a single GPU failure may require a restart of the entire job.

Figure 5. One synchronous training step: every chip waits at the exchange

Schematic. In synchronous data-parallel training the step ends only when every participant has contributed; a slow chip or link delays all of them, and a failed one can force a restart from the last checkpoint (Meta, arXiv 2407.21783, section 3.3.4).

▼ **Table view**

Figure 5. One synchronous training step: every chip waits at the exchange — stages

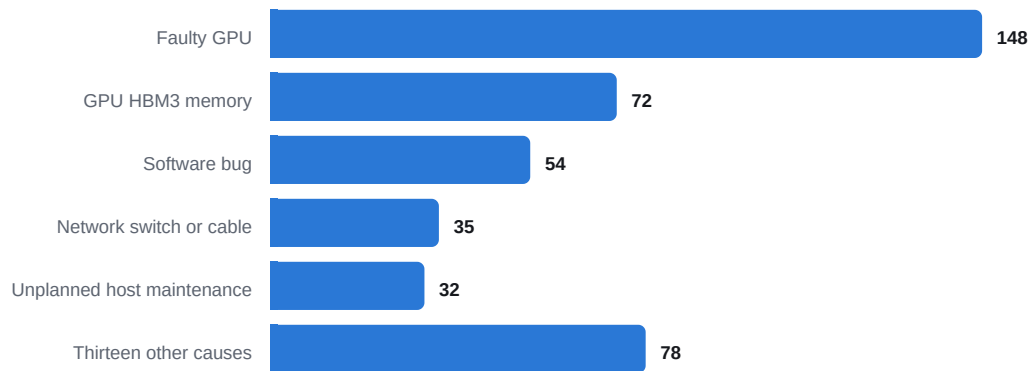
#	Stage	Note
1	Compute	each chip works on its share
2	Exchange	results pooled over links and network
3	Wait	until the slowest chip and link finish
4	Update	every copy changes identically

Figure 5. One synchronous training step: every chip waits at the exchange — connections

From	To	Label
Compute	Exchange	
Exchange	Wait	
Wait	Update	
Update	Compute	next step

During "a 54-day snapshot period of pre-training", Meta counted 466 job interruptions: 47 planned, for maintenance, and 419 unexpected — about one every three hours. It attributes "approximately 78%" of the unexpected ones to "confirmed hardware issues ... or suspected hardware-related issues", and writes that "GPU issues are the largest category, accounting for 58.7% of all unexpected issues". Averaged over the chips, the rate is modest: if all 16,384 GPUs ran for the whole 54 days, that is about 21 million GPU-hours, or roughly 50,700 GPU-hours per unexpected interruption. Sixteen thousand chips turn a rare event per chip into a frequent event per run, and the synchronous job turns each of those events into a stop for everyone. For Llama 3, Meta says it "achieved higher than 90% effective training time"; of the 54-day period it adds that "significant manual intervention was required only three times during this period".

Figure 6. Why a 16,384-GPU job stopped: Llama 3's 419 unexpected interruptions in a 54-day period, by cause



Source: Meta, 'The Llama 3 Herd of Models', arXiv 2407.21783, Table 5 and section 3.3.4. The table prints 30.1% beside the 148 faulty-GPU interruptions, although 148 of 419 is 35.3%; its other shares match their counts (72 is 17.2%, 54 is 12.9%). The paper's 58.7% for all GPU causes is the sum of the printed shares, 30.1% included; on the counts, GPU causes are 268 of 419, about 64%. The remaining causes include GPU SRAM (19), the GPU system processor (17), network cards (7), NCCL watchdog timeouts (7) and silent data corruption (6).

▼ Table view

Figure 6. Why a 16,384-GPU job stopped: Llama 3's 419 unexpected interruptions in a 54-day period, by cause

Cause	Interruptions
Faulty GPU	148
GPU HBM3 memory	72
Software bug	54
Network switch or cable	35
Unplanned host maintenance	32
Thirteen other causes	78

Failures are the visible part. Meta also reports:

Even a single straggler can slow down thousands of other GPUs, often appearing as functioning but slow communications.

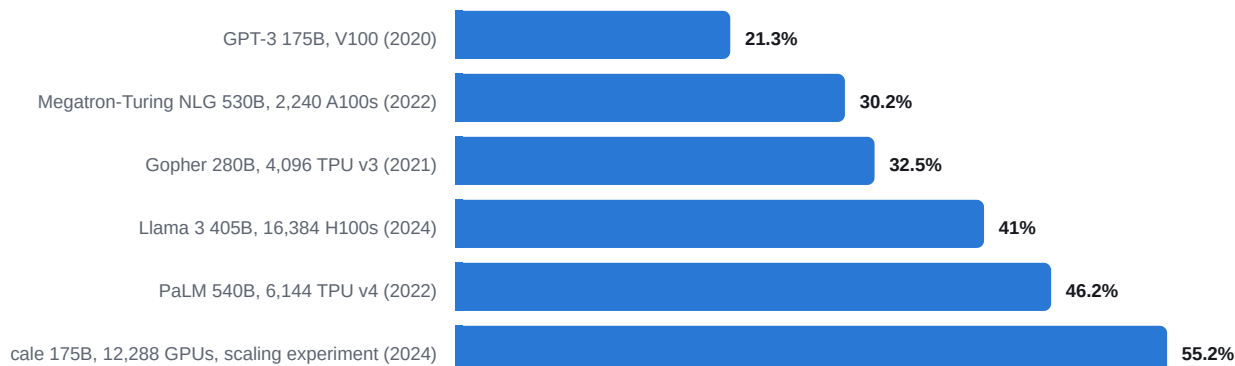
Its throughput also showed "a diurnal 1-2% throughput variation based on time-of-day", which the paper attributes to "higher mid-day temperatures impacting GPU dynamic voltage and frequency scaling".

3.7 Utilisation: how much arithmetic becomes model

Google's PaLM paper (arXiv 2204.02311, April 2022) proposed a measure of efficiency it argued was "implementation-independent", unlike counting the arithmetic the hardware actually performs. Model FLOPs utilisation, or MFU, "is the ratio of the observed throughput (tokens-per-second) relative to the theoretical maximum throughput of a system operating at peak FLOPs", where the theoretical maximum counts only the arithmetic the model strictly requires. PaLM 540B, trained "pipeline-free" on 6,144 TPU v4

chips across two pods, reached 46.2%. Google recomputed the figure for earlier large models from their reported throughput (for Gopher, a training speed obtained by personal communication with its authors), and later papers adopted the measure: ByteDance's MegaScale (arXiv 2402.15627, February 2024) reports that it "achieves 55.2% Model FLOPs Utilization (MFU) when training a 175B LLM model on 12,288 GPUs", in strong-scaling experiments whose table runs as high as 65.3% on 256 GPUs.

Figure 7. Model FLOPs utilisation reported for large training runs, 2020–2024



Sources: the first three and PaLM from Google's PaLM paper, arXiv 2204.02311, Table 3 and section 4.1, where Google computed MFU for the other models from their reported throughput (Gopher's from personal communication with its authors); Llama 3 from Meta, arXiv 2407.21783, section 3.3.2, which reports 38–43% across configurations (41% on 16,384 GPUs is plotted). MegaScale from ByteDance, arXiv 2402.15627 (February 2024), abstract and Table 2 (strong-scaling experiments; 59.1% on 3,072 GPUs and up to 65.3% on 256). PaLM's hardware FLOPs utilisation, which also counts recomputation, was 57.8%. Every figure comes from the organisation that ran or recomputed the run.

▼ Table view

Figure 7. Model FLOPs utilisation reported for large training runs, 2020–2024

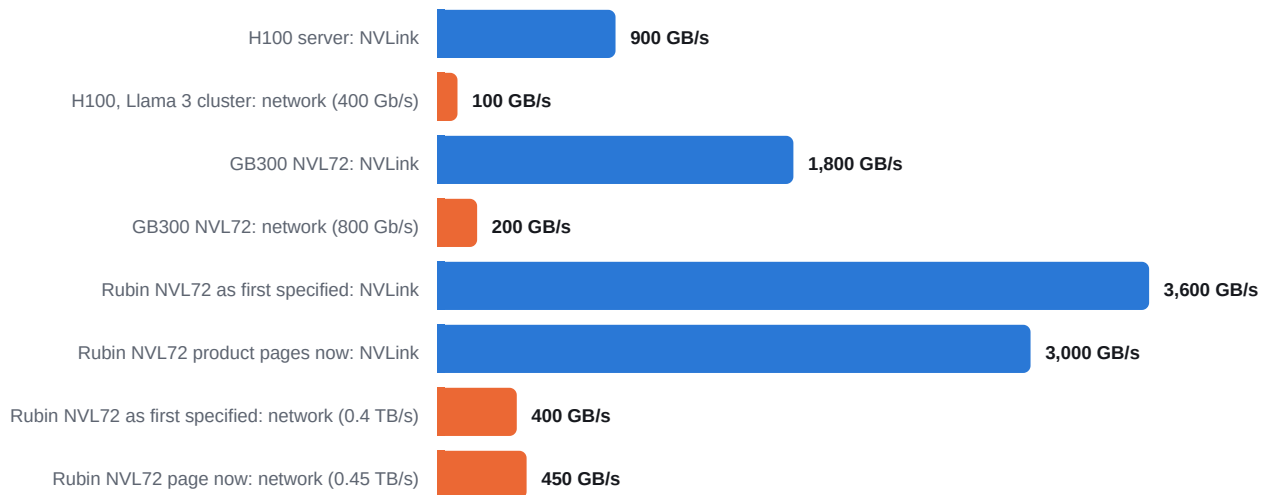
Model and hardware	MFU
GPT-3 175B, V100 (2020)	21.3%
Megatron-Turing NLG 530B, 2,240 A100s (2022)	30.2%
Gopher 280B, 4,096 TPU v3 (2021)	32.5%
Llama 3 405B, 16,384 H100s (2024)	41%
PaLM 540B, 6,144 TPU v4 (2022)	46.2%
MegaScale 175B, 12,288 GPUs, scaling experiment (2024)	55.2%

MFU is not a measure of waste. It leaves out work the chips really perform — PaLM's own hardware utilisation, which includes recomputing activations to save memory, was 57.8% — and its denominator is a datasheet peak, not something any chip is expected to sustain. Nor do the published figures say what the missing half consists of: memory stalls inside each chip, time in communication, the pipeline bubble, recomputation. What MFU does provide is comparability. Narayanan's 52% of peak and PaLM's 46.2% cannot be ranked against each other, because the first counts recomputation and the second does not.

4. How the hardware grew around the idea

The architecture of the machines has followed the two-road logic. In 2019 the Megatron-LM experiments ran on NVIDIA DGX-2H servers of 16 GPUs, joined inside each server at 300 gigabytes a second and to other servers at 100 gigabytes a second per server. By the H100 generation, a server's eight GPUs each had 900 GB/s of NVLink, and Meta's Llama 3 clusters gave each GPU a 400-gigabit network link — "Both RoCE and Infiniband clusters leverage 400 Gbps interconnects between GPUs". With the GB200 NVL72 NVIDIA moved the fast domain from the server to the rack, which its technical blog describes as "elevating the rack to the primary unit of integration".

Figure 8. Two roads per GPU: fast links inside the domain against the network connection beyond it (GB/s, both directions)



Sources: NVIDIA H100, GB300 NVL72, NVLink and Vera Rubin NVL72 pages, read 25 September 2026; NVLink page captures of 14 August to 14 September 2026 (Internet Archive); NVIDIA Technical Blog, 'Inside the NVIDIA Vera Rubin Platform: Six New Chips, One AI Supercomputer' (5 January 2026, modified 21 August 2026); Meta, arXiv 2407.21783, section 3.3.1. NVIDIA labels its NVLink 6 figures bidirectional, and the H100 and GB300 NVLink figures are read on the same convention. Network ratings in gigabits (400 Gb/s for Meta's H100 cluster, 800 Gb/s on the GB300 page) are converted assuming the usual per-direction convention and doubled for comparison; those pages do not state a direction. On that basis the ratio is about 9:1 for H100, GB300 and Rubin as first specified (NVIDIA's pre-edit Rubin page listed 0.4 TB/s of scale-out per GPU, and ConnectX-9 at 1.6 Tb/s). Between the captures of 1 and 11 September the page's NVLink figure fell to 3 TB/s and its scale-out figure rose to 0.45 TB/s, footnoted bi-directional: about 6.7:1 on NVIDIA's revised figures. DeepSeek's H800 cluster (arXiv 2412.19437, section 3.2.2) had 160 GB/s of NVLink against 50 GB/s of InfiniBand, 3.2:1 by DeepSeek's figures. Vendor peaks, except DeepSeek's, which it reports for its own cluster.

▼ Table view

Figure 8. Two roads per GPU: fast links inside the domain against the network connection beyond it (GB/s, both directions)

Platform and road	Bandwidth
H100 server: NVLink	900 GB/s
H100, Llama 3 cluster: network (400 Gb/s)	100 GB/s
GB300 NVL72: NVLink	1,800 GB/s
GB300 NVL72: network (800 Gb/s)	200 GB/s
Rubin NVL72 as first specified: NVLink	3,600 GB/s
Rubin NVL72 product pages now: NVLink	3,000 GB/s
Rubin NVL72 as first specified: network (0.4 TB/s)	400 GB/s
Rubin NVL72 page now: network (0.45 TB/s)	450 GB/s

Two things stand out. The fast domain has grown — from 16 GPUs in a 2019 DGX-2H server and eight in an H100 server to 72 in a rack — while the ratio between the two roads per GPU has stayed close to nine to one across three generations, on vendor figures and Meta's cluster, falling to about 6.7 to one on NVIDIA's revised Rubin page (on Megatron-LM's 2019 figures the gap had been wider still, some 24 to 48 to one).

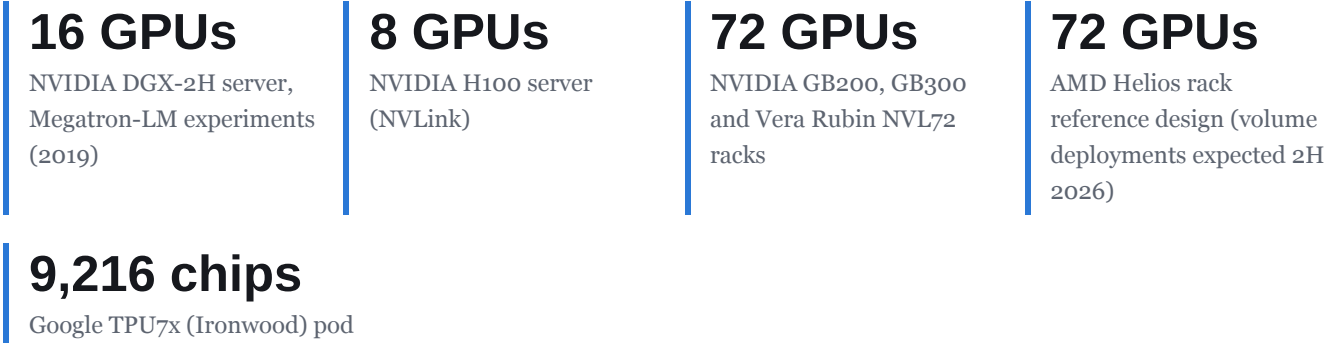
And the longer view is less favourable to every road off the chip. Amir Gholami and colleagues ("AI and Memory Wall", arXiv 2403.14123, March 2024) estimate that over twenty years interconnect bandwidth grew about 1.4 times every two years, far more slowly than peak compute — rates normalised to a 1990s baseline, and an interconnect series built from PCIe and NVLink generations rather than from data-centre networks.

The newest of NVIDIA's figures has just been revised. NVIDIA's NVLink product page, in Internet Archive captures from mid-August to 14 September 2026, said that sixth-generation NVLink "enables 3.6 TB/s of bandwidth per GPU for the NVIDIA Rubin platform —2x more bandwidth than the previous generation and over 14x the bandwidth of PCIe Gen6". On 25 September the same sentence reads "3 TB/s", "1.7x" and "12x", and the rack total has gone from 260 to 216 terabytes a second. As the Vera Rubin NVL72 page's early-September revision had already shown, between Internet Archive captures of 1 and 11 September, NVIDIA's Rubin figures were moving: 3 TB/s and 216 TB/s for NVLink, and a per-GPU scale-out figure raised from 0.4 to 0.45 TB/s. The NVLink page's own NVLink figures now match. NVIDIA's technical blog on the Rubin platform, last modified on 21 August, still says that "NVIDIA NVLink 6 delivers 3.6 TB/s of bidirectional GPU-to-GPU bandwidth per GPU, doubling scale-up bandwidth over the prior generation". The NVLink page therefore changed between 14 and 25 September, and neither product page gives a reason.

5. Other postures

Google has built its fast domain on a different scale and in a different shape. Its documentation for TPU7x, its Ironwood chip (last updated 18 September 2026), describes "a 9,216-chip footprint per Pod", joined by Google's own inter-chip interconnect at 1,200 GB/s per chip, bidirectional, against 100 gigabits a second of data-centre networking per chip. The comparison is loose — a TPU pod is a torus of direct links, not a switched rack — but the difference in scale is two orders of magnitude.

Figure 9. How many accelerators share the fast road



Sources: Shoeybi et al., arXiv 1909.08053, section 5; NVIDIA NVLink page specification table ('NVLink GPU Domains 8 | 72') and GB200/GB300/Vera Rubin NVL72 pages; AMD Instinct MI400 page; Google Cloud TPU7x documentation (last updated 18 September 2026), all read 25 September 2026. The domains are not equivalent: NVIDIA's is switched; AMD's page describes a 72-GPU scale-up domain without giving its topology; Google's is a three-dimensional torus of direct links.

▼ Table view

Figure 9. How many accelerators share the fast road

Measure	Value
NVIDIA DGX-2H server, Megatron-LM experiments (2019)	16 GPUs
NVIDIA H100 server (NVLink)	8 GPUs
NVIDIA GB200, GB300 and Vera Rubin NVL72 racks	72 GPUs
AMD Helios rack reference design (volume deployments expected 2H 2026)	72 GPUs
Google TPU7x (Ironwood) pod	9,216 chips

The opposite posture is to live with a slow road and engineer around it. DeepSeek trained its V3 model on "2048 NVIDIA H800 GPUs" (arXiv 2412.19437, December 2024), eight to a node, and reported that on its cluster "NVLink offers a bandwidth of 160 GB/s, roughly 3.2 times that of IB (50 GB/s)". Its pipeline schedule, DualPipe, was designed to hide communication behind computation; the report says the method "has fewer pipeline bubbles and hides most of the communication during training through computation-communication overlap". The same report is the source of a widely repeated cost figure, \$5.576m, which it defines narrowly: 2.788m H800 GPU-hours priced at an assumed "\$2 per GPU hour", covering "only the official training of DeepSeek-V3, excluding the costs associated with prior research and ablation experiments".

MLCommons has since made the model a yardstick for large clusters: it introduced a DeepSeek-V3 benchmark in MLPerf Training v6.0 (16 June 2026), calling it "the largest benchmark in our suite with 671 billion parameters", and NVIDIA's own statement for the round says it "scaled DeepSeek-V3 training across 8,192 GPUs using GB200 NVL72 systems". MLCommons describes its suite as "open-source and peer-reviewed"; the results themselves are submitted by vendors and cloud providers.

6. What to watch

- **Which NVLink figure NVIDIA stands behind for Rubin.** Its NVLink and Vera Rubin NVL72 product pages say 3 TB/s per GPU, and the NVLink page "1.7x"; its technical blog, modified on 21 August 2026, says 3.6 TB/s and "doubling". Whether the blog is revised, and what the first independent collective-communication measurements on Vera Rubin NVL72 racks report, are both checkable.
- **The next MLPerf Training round.** MLCommons said on 24 September 2026 that a new benchmark begins "with the v6.1 submission round in October 2026"; its post gives no results date. v6.0 appeared on 16 June 2026. The questions are whether Vera Rubin NVL72 or AMD's MI455X appear among available systems, and whether any submitter's DeepSeek-V3 times keep improving beyond 8,192 GPUs.
- **AMD's Helios rack.** AMD's MI400 page says Helios combines "72 AMD Instinct MI455X GPUs", that it is "a reference design, not a product for sale" for partners to build, and that "volume deployments" are "expected in 2H 2026". By 31 December 2026 it should either appear in a public benchmark or rating, or not.

7. The idea to keep

The concept that makes cluster announcements legible is **utilisation**: the share of a cluster's peak arithmetic that becomes model, which Google's PaLM paper formalised in 2022 as model FLOPs utilisation. It is low for structural reasons. Every way of dividing a model across chips — copying it, cutting its layers, or pipelining them — ends each step in an exchange, and an exchange waits for its slowest participant over roads of two speeds: fast links inside a server or rack, a network perhaps a ninth as fast beyond it. Published MFU figures for large runs range from 21.3% (GPT-3, on Google's recomputation) to 46.2% (PaLM), with Llama 3 at 38% to 43% and ByteDance's scaling experiments higher. Only the organisations that ran or recomputed them can check those numbers.

A reader faced with a headline about tens of thousands of accelerators can ask three things. How large is the fast domain? How fast is the road between domains? And what share of peak did the run actually use?

The fullest single account of how the three kinds of split fit together is Deepak Narayanan and colleagues, "Efficient Large-Scale Language Model Training on GPU Clusters Using Megatron-LM", arXiv 2104.04473 (SC21, 2021).