



Mapping TP, PP and DP to NVLink and InfiniBand Domains

Plan and build private AI compute with GPU Smith. We connect workload requirements with GPU hardware, networking, deployment and operating decisions so your infrastructure fits the work it must perform.

Published by GPU Smith · 2026-10-29 · tensor parallel pipeline parallel data parallel nvlink infiniband placement · megatron tensor parallel within node · 3d parallelism communication volume · nvlink vs infiniband for tensor parallelism · nvlink vs infiniband · sequence parallelism megatron activation memory

Original article: <https://gpusmith.com/articles/en/mapping-3d-parallelism-to-nvlink-and-infiniband-domains>



In this article

1. The two-level bandwidth hierarchy
2. Communication volume per parallel dimension
3. Frequency and latency sensitivity
4. Placement rules
5. Sequence parallelism and activation memory
6. Interleaved pipeline schedule
7. Case: 8-GPU HGX node
8. Case: NVL72 domain
9. Worked example: 405B-class dense model on 1,024 H100 GPUs
10. Verifying with NCCL tests and topology
11. Common mistakes
12. Sources

The two-level bandwidth hierarchy

Large training jobs run on two networks: a scale-up NVLink/NVSwitch domain and a scale-out InfiniBand or RoCE fabric with one NIC per GPU. Placement means putting each communication pattern on the network that can carry it. The gap is about an order of magnitude per GPU:

System	NVLink per GPU (NVIDIA figure)	NVLink per direction	Scale-out NIC per GPU	Scale-out per direction	Ratio (per direction)
DGX H100 (8-GPU domain)	900 GB/s	450 GB/s	1 x ConnectX-7, up to 400 Gb/s	50 GB/s	9x
GB200 NVL72 (72-GPU domain)	1,800 GB/s	900 GB/s	1 x ConnectX-7, 400G	50 GB/s	18x

The DGX H100 has "4 x 4th generation NVLinks that provide 900 GB/s GPU-to-GPU bandwidth" and eight single-port ConnectX-7 cards at up to 400 Gb/s each ([NVIDIA DGX H100 user guide](#)). The 900 GB/s figure counts both directions: H100 has 18 NVLink links, each "delivering 25 GB/sec effective bandwidth in each direction" ([NVIDIA Hopper architecture in-depth](#)), so $18 \times 25 = 450$ GB/s each way. $400 \text{ Gb/s} / 8 = 50$ GB/s at line rate. For Blackwell, NVIDIA lists 1,800 GB/s NVLink bandwidth per GPU and NVLink GPU domains of 8 or 72 ([NVIDIA NVLink](#), marked preliminary). The same page calls the next generation's per-GPU figure "3 TB/s bidirectional", so 1,800 GB/s is likewise both directions: 900 GB/s each way (18 links x 50 GB/s). A DGX GB200 rack is "18x 1RU compute trays, each with 2 Grace CPUs and 4 Blackwell GPUs", with four ConnectX-7 400G OSFP NICs per tray for the cluster network ([NVIDIA DGX GB rack-scale user guide](#)); the GB300 trays in the same guide use 800G ConnectX-8, which halves the ratio to 9x.

Two consequences follow. A collective that costs 10% of step time on NVLink can dominate it on the NIC. And the fast domain grows from 8 GPUs to 72, which changes which dimensions fit inside it. NVIDIA calls the NVL72 "a 72-GPU NVIDIA NVLink domain that acts as a single, massive GPU" ([NVIDIA GB200 NVL72](#)); for

placement, the operative fact is that any two of those 72 GPUs see the same NVLink bandwidth, so tray boundaries stop mattering.

For the fabric build that consumes these placement decisions, see [fabric sizing](#), [ports and rails](#).

Communication volume per parallel dimension

The formulas below come from the Megatron-LM scaling paper ([Narayanan et al., arXiv 2104.04473](#)) and the Llama 3 paper ([arXiv 2407.21783](#)). Notation follows Megatron: microbatch size b , sequence length s , hidden size h , tensor-parallel size t , pipeline-parallel size p , data-parallel size d , microbatches per pipeline per step $m = B/(b \times d)$ for global batch B in sequences, and l_{stage} layers per pipeline stage. Volumes are in elements; multiply by 2 bytes for BF16.

Tensor parallelism (TP). Two all-reduces forward and two backward per layer, each on a tensor of size bsh . The paper gives "a total communication of $8bsh(t-1)/t$ per layer per device for each microbatch" (times l_{stage} per stage). Sequence parallelism (SP) replaces each all-reduce with a reduce-scatter plus an all-gather; because a ring all-reduce is itself a reduce-scatter followed by an all-gather, "the communication bandwidth used for tensor parallelism and tensor together with sequence parallelism are the same" ([Korthikanti et al., arXiv 2205.05198](#)).

Pipeline parallelism (PP). Point-to-point between consecutive stages: bsh per microbatch, forward and backward. With the paper's scatter/gather optimization, each TP rank sends only bsh/t over its own NIC and the receiver re-assembles the tensor with an NVLink all-gather. Interleaving with v chunks per device multiplies PP traffic by v .

Data parallelism (DP). One reduction per step over the $P/(t \times p)$ parameters each GPU holds. With Megatron's distributed optimizer, "each data-parallel rank calls reduce-scatter" after the backward pass and an all-gather rebuilds the parameters after the optimizer step ([Megatron Core distributed optimizer](#)). Ring cost per GPU is $(d-1)/d$ of the buffer for each half.

Context parallelism (CP). Llama 3 splits the sequence and "first all-gather[s] the key (K) and value (V) tensors", arguing that with GQA this is negligible against $O(S^2)$ attention compute.

Expert parallelism (EP). All-to-all token dispatch; its sizing is covered separately in [expert parallelism traffic sizing](#).

Per-GPU, per-step volumes (our algebra; BF16 activations, FP32 gradient reduce-scatter, BF16 parameter all-gather):

Dimension	Collective	Per-GPU bytes per training step
TP (with or without SP)	all-reduce, or reduce-scatter + all-gather	$l_{\text{stage}} \times m \times 8bsh(t-1)/t \times 2$
PP (scatter/gather on)	send/recv	$m \times v \times 2 \times (bsh/t) \times 2$
CP (all-gather K,V; GQA)	all-gather fwd, reduce-scatter bwd	$l_{\text{stage}} \times m \times 2 \times 2bs(h_{\text{kv}}/t)(c-1)/c \times 2$
DP (distributed optimizer)	reduce-scatter + all-gather	$(P/(t \times p)) \times (d-1)/d \times (4 + 2)$

A useful identity drops out of the TP row. Because $l_{\text{stage}} = L/p$ and $m = B/(b \times d)$, the TP volume per GPU is $L \times B \times s \times h \times 8(t-1)/t \times 2 / (p \times d)$. For a fixed GPU count n , $p \times d = n/t$, so **TP traffic per GPU depends only on t** , not on how the remaining GPUs are split between PP and DP. Rebalancing PP against DP moves bubble, memory and DP traffic, never TP traffic.

Frequency and latency sensitivity

Volume is half the story; the other half is how often a collective sits on the critical path.

Dimension	Operations per microbatch per layer	On critical path?	Overlap available
TP	4 all-reduces (8 ops with SP)	Yes, between GEMMs in every layer	Partial (<code>--tp-comm-overlap</code>)
CP	1 all-gather fwd, 1 reduce-scatter bwd (Llama 3 style)	Yes, exposed before attention	Ring variants overlap with compute
PP	2 sends per microbatch per stage boundary ($\times v$)	Partly; bubbles hide some	Async p2p
DP	1 reduce-scatter + 1 all-gather per step, bucketed	Mostly no	<code>--overlap-grad-reduce</code> , <code>--overlap-param-gather</code>

The overlap flags are from the [Megatron Core parallelism guide](#). The Megatron paper: "data parallelism only needs to perform expensive all-reduce communication once per batch", while tensor parallelism all-reduces "for every microbatch". In Llama 3, the innermost dimension "requires the highest network bandwidth and lowest latency", while PP and DP crossed "a multi-hop network, with latency up to tens of microseconds" ([Llama 3](#)).

Placement rules

- TP inside the NVLink domain.** Megatron's Takeaway #1: "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." Its 161B-parameter experiment found "peak performance when the tensor-parallel size is equal to the number of GPUs in a single node" ([arXiv 2104.04473](#)).
- CP next, as close as possible.** Llama 3 ordered dimensions [TP, CP, PP, DP]. The Megatron Core guide says to "Use CP instead of larger TP for long sequences."
- PP across nodes.** Pipeline parallelism "uses much cheaper point-to-point communication that can be performed across nodes without bottlenecking the entire computation."
- DP outermost.** Takeaway #2: choose a model-parallel size $t \times p$ so the model fits, and "data parallelism can be used to scale up training to more GPUs." Llama 3 made FSDP the outermost dimension "because it can tolerate longer network latency."

Rank order enforces the rules. Megatron Core's `initialize_model_parallel` defaults to `order="tp-cp-ep-dp-pp"`, with TP on adjacent ranks; Megatron-LM's training entry point switches to `tp-cp-ep-pp-dp` when `--use-tp-pp-dp-mapping` is set ([Megatron-LM initialize.py](#), release `core_v0.19.2`, 2026-09-18). The default

makes PP outermost; the alternate makes DP outermost, as in Llama 3. On a non-blocking fabric the difference is small; with an oversubscribed tier, the outermost dimension is the one that crosses it.

Sequence parallelism and activation memory

Sequence parallelism shards the layer-norm and dropout activations that TP alone replicates. The SP paper reports that SP plus selective activation recomputation "reduces activation memory by 5x" and, for a 530B GPT-3-style model on 2,240 A100 GPUs, achieves 54.2% MFU, "29% faster than the 42.1% we achieve using recomputation" ([arXiv 2205.05198](https://arxiv.org/abs/2205.05198)). Megatron Core says to "Always enable when using TP".

The paper's Table 2 gives activation bytes per transformer layer for its GPT-style layer:

- TP only: $sbh(10 + 24/t + 5as/(ht))$
- TP + SP: $sbh(34/t + 5as/(ht))$
- TP + SP + selective recomputation: $sbh(34/t)$

Worked arithmetic for $s = 8,192$, $b = 1$, $h = 16,384$, $t = 8$ (our calculation): $sbh = 134,217,728$. With selective recomputation, TP only gives $13 \times sbh = 1.745$ GB per layer; TP + SP gives $4.25 \times sbh = 0.570$ GB per layer, a 3.06x reduction from SP alone. Under 1F1B the first stage holds p microbatches of L/p layers, roughly L layers' worth (the paper's Equation 5): about $126 \times 0.570 = 71.8$ GB for a 126-layer model, before model state, and more with interleaving. The coefficient assumes a 4h GeLU MLP and multi-head attention, so treat it as an order of magnitude for SwiGLU/GQA models. At $TP = 8$, SP is what lets layouts with fewer pipeline stages fit. Size the memory side with the [training memory calculator](#).

Interleaved pipeline schedule

With 1F1B and periodic flushes, the bubble fraction is $(p-1)/m$. The interleaved schedule assigns v non-contiguous chunks of layers per device and reduces the bubble to $(1/v) \times (p-1)/m$, but "the amount of communication also increases by v " ([arXiv 2104.04473](https://arxiv.org/abs/2104.04473)). The paper reports interleaving "can improve throughput by 10+%", but "without the scatter/gather optimization, the default schedule performs better than the interleaved schedule at larger batch sizes"; scatter/gather alone gave "up to 11% in throughput for communication-intensive schedules."

Interleaving trades IB bytes for bubble time, which pays when PP traffic is small relative to NIC capacity, as the worked example shows. Llama 3 used it, with bubble $(PP-1)/(V \times M)$. The paper's interleaved schedule requires the number of microbatches to be an integer multiple of the pipeline degree; Llama 3 modified its schedule to remove that constraint.

Case: 8-GPU HGX node

On an 8-GPU H100 node the rules collapse to a short list:

- $t = 8$ inside the node. At fixed GPU count, per-GPU TP volume scales with $t - 1$, so $t = 4$ cuts TP traffic by 57% ($3/7$ of the $t = 8$ volume; 1,623 GB versus 3,788 GB per step in the worked example below), but it doubles parameter memory per GPU or forces more pipeline stages.

- One NIC per GPU, 400 Gb/s. With $t = 8$ and scatter/gather, every PP transfer uses all eight NICs, one per rail. Rail alignment matters: NCCL's `NCCL_CROSS_NIC=0` will "Always use the same NIC for the same ring/tree, to avoid crossing network rails"; the default (2) prefers the same NIC but allows crossing ([NCCL environment variables](#)).
- Head divisibility. Megatron's attention divides query heads and KV groups by the TP size with an exact `divide()` ([Megatron-LM attention.py](#), `core_v0.19.2`); a model with 8 KV heads maps cleanly to $t = 8$.

Llama 3 405B, trained "on up to 16K H100 GPUs" in 8-GPU servers, kept $TP = 8$ throughout. Its Table 4: 8,192 GPUs at $TP 8$, $CP 1$, $PP 16$, $DP 64$ (430 TFLOPs/GPU, 43% BF16 MFU); 16,384 GPUs at $TP 8$, $CP 1$, $PP 16$, $DP 128$ (400, 41%); and 16,384 GPUs at $TP 8$, $CP 16$, $PP 16$, $DP 8$ for 131,072-token sequences (380, 38%) ([arXiv 2407.21783](#)). The long-context stage put $CP = 16$ outside the node because TP filled the NVLink domain.

Case: NVL72 domain

A 72-GPU NVLink domain lets more than one dimension live on NVLink:

- **TP > 8 without crossing the NIC.** Useful for very wide layers or inference, but the TP traffic identity still applies: going from $t = 8$ to $t = 16$ raises per-GPU TP bytes by $15/7 = 2.14\times$ at fixed n (our arithmetic), and GEMMs shrink. Bandwidth no longer forbids it; efficiency still argues against it.
- **TP x PP inside the rack.** For 126 layers, $72 = 8 \times 9$ and $126 = 9 \times 14$, so $TP 8 \times PP 9$ fits a full replica in one rack; only DP crosses IB.
- **TP x EP or TP x CP inside the rack** for MoE or long-context models.

The factor of 9 is the catch. Layer counts, head counts and global batch size must divide cleanly, and most models are built around powers of two; 128 heads cannot be split 72 ways, so "one giant GPU" does not mean $TP = 72$. Check divisibility before committing to rack counts.

Worked example: 405B-class dense model on 1,024 H100 GPUs

Model shape from Llama 3 405B: 126 layers, model dimension 16,384, 128 attention heads, 8 KV heads ([arXiv 2407.21783](#)), so head dimension $16,384/128 = 128$ and $h_{kv} = 8 \times 128 = 1,024$. Assumptions: 128 HGX H100 nodes, $s = 8,192$, $b = 1$, global batch 16M tokens = 2,048 sequences (the Llama 3 batch), BF16 activations, distributed optimizer at $6 + 12/d$ bytes per parameter for BF16 parameters with FP32 gradients ([Megatron Core distributed optimizer](#)). Candidate B is the Megatron Core guide's published layout for LLaMA-3.1 405B on 1,024 GPUs: $TP 8$, $PP 8$, $CP 2$ ([Megatron Core parallelism guide](#)).

Intermediate numbers: $bsh \times 2$ bytes = 268.4 MB; TP volume per layer per microbatch at $t = 8$ is 7×268.4 MB = 1.879 GB; with $p \times d = 128$, per-GPU TP traffic is $126 \times 2,048 \times 8,192 \times 16,384 \times (7/8) \times 16 / 128 = 3,788$ GB per step for every $t = 8$ layout.

Layout (TP, PP, CP, DP)	m	Params/G PU	Model state GB/GPU	TP GB (NVLink)	PP GB (IB)	CP GB	DP GB (IB)	Bubble (v=1)
A: 8, 16, 1, 8	256	3.16B	23.7	3,788	17.2	0	16.6	5.9%
B: 8, 8, 2, 8	256	6.33B	47.5	3,788	8.6	16.9 (IB)	33.2	2.7%
C: 16, 8, 1, 8	256	3.16B	23.7	8,117 (crosses IB)	8.6	0	16.6	2.7%
D: 8, 32, 1, 4	512	1.58B	14.2	3,788	34.4	0	7.1	6.1%
E: 4, 16, 1, 16	128	6.33B	42.7	1,623	17.2	0	35.6	11.7%

Volumes are per GPU per step (our arithmetic); model state excludes activations. For A, 126 layers over 16 stages is 7.875; Llama 3 removed one layer each from the first and last stages, leaving 8 layers on the middle stages.

For scale, using $6 \times P \times$ tokens, one step is $6 \times 405e9 \times 16.78e6 = 4.08e19$ FLOP. At Llama 3's reported 400 TFLOPs per GPU on 1,024 GPUs that is about 99.5 s per step. Then, at line rate:

- A's TP traffic on NVLink: $3,788 \text{ GB} / 450 \text{ GB/s} = 8.4 \text{ s}$, about 8% of the step, partly overlappable.
- The same TP traffic forced onto the NIC: $3,788 / 50 = 75.8 \text{ s}$, about 76% of the step.
- C's TP16 group spans two nodes, so its ring is bounded by the per-GPU NIC: $8,117 / 50 = 162 \text{ s}$, longer than the entire compute time. On an NVL72 at 900 GB/s per direction, the same group would take about 9.0 s.
- A's PP traffic over IB: $17.2 / 50 = 0.34 \text{ s}$ (0.69 s with $v = 2$, which halves the bubble to 2.9%).
- A's DP traffic: $16.6 / 50 = 0.33 \text{ s}$, largely overlapped with backward.

C is eliminated by bandwidth. E cuts TP traffic by 57% but pays an 11.7% bubble and 1.8x the model state of A. D minimizes memory with the deepest pipeline. A (with $v = 2$) and B are the serious candidates: A carries more activation memory on its first stage, while B halves per-GPU activations through CP but doubles model state and puts a per-layer CP exchange on IB. Measured memory headroom and whether the CP gather is hidden decide between them; both appear in published recipes. Apart from C's TP traffic, every IB-crossing volume here is under 1% of what a 50 GB/s NIC moves in a 100 s step (5,000 GB): once TP stays on NVLink, the fabric question is latency and placement, not raw bandwidth.

On a GB200 NVL72 system the same model maps as TP 8 x PP 9 per rack. With 14 racks (1,008 GPUs, a global batch of 2,016 sequences for divisibility), each GPU holds $405e9 / 72 = 5.63\text{B}$ parameters, or $5.63\text{B} \times (6 + 12/14) = 38.6 \text{ GB}$ of model state, against 186 GB of HBM3E per GPU (372 GB per two-GPU superchip). PP traffic moves onto NVLink, and the only IB traffic is DP: $5.63e9 \times 13/14 \times 6 = 31.3 \text{ GB}$ per step, about 0.63 s at 400G.

Verifying with NCCL tests and topology

1. **Topology as NCCL sees it.** `NCCL_TOPO_DUMP_FILE` dumps "the topology as seen by NCCL"; `NCCL_DEBUG=INFO` with `NCCL_DEBUG_SUBSYS` shows ring and tree construction ([NCCL environment variables](#)). Confirm each GPU's NIC and rail.
2. **Benchmark the group sizes you will use.** Run `all_reduce_perf` at `t = 8` inside a node, `all_gather_perf` and `reduce_scatter_perf` across the DP group size, and `sendrecv_perf` between PP neighbours. Compare busbw, not algbw: busbw for all-reduce applies a $2(n-1)/n$ factor and "should reflect the speed of the hardware bottleneck", so it can be compared with link peak "independently of the number of ranks used" ([nccl-tests PERFORMANCE.md](#)).
3. **Check rank-to-node mapping.** Log each rank's TP, PP and DP groups with hostnames; every TP group should resolve to one host or NVL72 rack.
4. **Profile one step.** A TP all-reduce far slower than measured intra-node busbw implies is a placement bug, not a tuning problem.

Algorithm and protocol choices for these collectives are covered in [NCCL algorithm selection](#).

Common mistakes

- **TP across nodes.** From `t` larger than GPUs per node, or a launcher that does not assign contiguous ranks per host: a 9x slower link (H100, 400G NICs, per direction) carrying the largest volume.
- **Rail mismatch.** Flows leaving on NIC `i` and arriving on NIC `j` cross the spine needlessly; check `NCCL_CROSS_NIC` and cabling against the rail plan.
- **Wrong dimension on the oversubscribed tier.** Llama 3's RoCE cluster had full bisection inside 3,072-GPU pods and a 1:7 oversubscription ratio at the aggregation layer, with parallelism and scheduling "optimized to be aware of network topology, aiming to minimize network communication across pods" ([arXiv 2407.21783](#)). Only the outermost dimension should cross such a tier, and the rank order decides which one that is.
- **Treating NVL72 as TP = 72.** Use the domain for TP x PP, TP x EP or TP x CP instead.
- **Skipping SP.** At `t = 8`, TP-only activations are about $3 \times \text{TP} + \text{SP}$ for the paper's layer shape.

GPU Smith can review a proposed parallel layout against a specific node, rack and fabric design.

Sources

1. Narayanan et al., Megatron-LM at scale, arXiv 2104.04473 (2021). <https://arxiv.org/abs/2104.04473>
2. Korthikanti et al., Reducing Activation Recomputation, arXiv 2205.05198 (2022). <https://arxiv.org/abs/2205.05198>
3. Meta, The Llama 3 Herd of Models, arXiv 2407.21783 (2024). <https://arxiv.org/abs/2407.21783>
4. NVIDIA, Megatron Core parallelism guide (accessed 2026-10-28). <https://docs.nvidia.com/megatron-core/developer-guide/latest/user-guide/parallelism-guide.html>
5. NVIDIA, Megatron Core distributed optimizer (accessed 2026-10-28). https://docs.nvidia.com/megatron-core/developer-guide/latest/user-guide/features/dist_optimizer.html

6. Megatron-LM initialize.py, release core_v0.19.2 (2026-09-18; accessed 2026-10-28).
https://github.com/NVIDIA/Megatron-LM/blob/core_v0.19.2/megatron/training/initialize.py
7. Megatron-LM attention.py, release core_v0.19.2 (accessed 2026-10-28).
https://github.com/NVIDIA/Megatron-LM/blob/core_v0.19.2/megatron/core/transformer/attention.py
8. NVIDIA DGX H100/H200 user guide. <https://docs.nvidia.com/dgx/dgxm100-user-guide/introduction-to-dgxm100.html>
9. NVIDIA, Hopper Architecture In-Depth. <https://developer.nvidia.com/blog/nvidia-hopper-architecture-in-depth/>
10. NVIDIA NVLink (preliminary specs, accessed 2026-10-28). <https://www.nvidia.com/en-us/data-center/nvlink/>
11. NVIDIA GB200 NVL72 (accessed 2026-10-28). <https://www.nvidia.com/en-us/data-center/gb200-nvl72/>
12. NVIDIA DGX GB rack-scale user guide, Hardware. <https://docs.nvidia.com/dgx/dgxm200-user-guide/hardware.html>
13. nccl-tests PERFORMANCE.md. <https://github.com/NVIDIA/nccl-tests/blob/master/doc/PERFORMANCE.md>
14. NCCL environment variables. <https://docs.nvidia.com/deeplearning/nccl/user-guide/docs/env.html>

Source links

Links cited in this article, in order of first appearance. Full addresses are provided for readers using extracted text.

1. <https://docs.nvidia.com/dgx/dgxm100-user-guide/introduction-to-dgxm100.html>
2. <https://developer.nvidia.com/blog/nvidia-hopper-architecture-in-depth>
3. <https://www.nvidia.com/en-us/data-center/nvlink>
4. <https://docs.nvidia.com/dgx/dgxm200-user-guide/hardware.html>
5. <https://www.nvidia.com/en-us/data-center/gb200-nvl72>
6. <https://arxiv.org/abs/2104.04473>
7. <https://arxiv.org/abs/2407.21783>
8. <https://arxiv.org/abs/2205.05198>
9. https://docs.nvidia.com/megatron-core/developer-guide/latest/user-guide/features/dist_optimizer.html
10. <https://docs.nvidia.com/megatron-core/developer-guide/latest/user-guide/parallelism-guide.html>
11. https://github.com/NVIDIA/Megatron-LM/blob/core_v0.19.2/megatron/training/initialize.py
12. <https://docs.nvidia.com/deeplearning/nccl/user-guide/docs/env.html>
13. https://github.com/NVIDIA/Megatron-LM/blob/core_v0.19.2/megatron/core/transformer/attention.py
14. <https://github.com/NVIDIA/nccl-tests/blob/master/doc/PERFORMANCE.md>

THE PUBLISHER

About GPU Smith

GPU Smith provides independent engineering and research for private AI infrastructure. We help technical buyers and operators reason about GPU workload sizing, hardware selection, procurement, deployment and inference operations, from the compute system to the networking, power and cooling requirements around it.

Start with the workload

Useful infrastructure decisions begin with the models, data, throughput, latency and operating constraints a team actually has. GPU Smith helps connect those requirements with system design and deployment choices. The goal is a privately controlled AI environment whose capacity and operational demands are understood before a purchasing decision.

Hardware and supplier research

Our public reference library covers GPUs, complete systems and networking components. The server-vendor directory and manufacturing research help teams investigate suppliers, compare options and follow sources behind technical claims. We distinguish manufacturer specifications from measured benchmarks and advertised capabilities from tested configurations. Reference pages are research resources, not a live inventory listing or a binding equipment quote.

Deployment and operations

GPU Smith's engineering scope includes infrastructure integration, networking, power, cooling and the practical operation of inference workloads. Our published articles explain the tradeoffs behind deployment, procurement and ongoing operations so teams can ask better questions and document decisions.

Work with GPU Smith

Explore the [hardware library](#), [GPU references](#), [networking references](#) and [vendor research](#). [Contact GPU Smith](#) with your workloads and deployment constraints to discuss a project and confirm current scope, pricing and availability.

Inclusion of a manufacturer or product in our research does not imply a partnership, certification or endorsement.

Website: <https://gpusmith.com>

This article is educational. Verify consequential medical, legal, financial, regulatory and technical decisions with appropriate professionals and the cited primary sources. Company services are described separately from the article's research findings.