GPU SMITH
COMPUTE ENGINEERING

GPU SMITH / RESEARCH & PRACTICAL GUIDANCE

Diffusion Model GPU Sizing: Accepted Images and Video by Deadline

> 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 · 2026-11-14 · diffusion model gpu sizing · stable diffusion gpu requirements · flux vram requirements · wan 2.1 gpu memory · video generation gpu requirements · private image generation capacity planning · h100 vs l40s image generation · accepted output throughput · ml.energy benchmark · mlperf sdxl

Original article: https://gpusmith.com/articles/en/private-diffusion-gpu-sizing-images-video-deadline



In this article

1. Executive Summary
2. Introduction and Background
3. Key Changes in How Diffusion Capacity Is Sized
4. Implementation Considerations and Process Changes
5. Model and Configuration Reference Points
6. Data Analysis and Evidence
7. Implications and Future Directions
8. Frequently Asked Questions (FAQs)
9. Conclusion

Executive Summary

Sizing a private diffusion platform is a deadline problem measured in **accepted deliverables**, not a VRAM lookup. The unit that matters is an accepted image or video clip, produced by a whole replica (one execution group of one or more GPUs), after rejections and retries, inside a wall-clock window. This report, current as of November 2026, defines that contract and explains why public benchmark cells cannot replace a measured, workload-specific acceptance run. Every figure below is a dated, attributed third-party number shown with its configuration; none is a new measurement.

The published reference points span orders of magnitude. The MLPerf Inference SDXL benchmark fixes **20 denoising steps** and 5,000 validation samples (mlcommons.org), and NVIDIA reported **5 samples/s** from eight L40S GPUs (developer.nvidia.com). A Stability AI and NVIDIA model card lists H100 SDXL throughput of 0.40 to 0.68 images/s at 30 steps and 1024x1024 (huggingface.co). For video, the HunyuanVideo README states a **60 GB** minimum for 720px1280px129f (github.com), Wan2.2 A14B at 1280*720 on one GPU needs at least **80 GB** (github.com), and the Wan2.1 1.3B model needs only **8.19 GB** (github.com). ML.ENERGY reports 26 kJ to 1.16 MJ per generated video (ml.energy).

Three findings drive the method. First, step count, resolution, precision, scheduler and offloading are part of the product: energy rises linearly with denoising steps (arxiv.org), so fewer steps are a different quality operating point until requalified. Second, parallelism inside a request consumes GPUs without guaranteeing speed. xDiT states that more ranks do not guarantee lower latency (raw.githubusercontent.com), and the HunyuanVideo README's 8-GPU latency of 337.58 s versus 1,904.08 s on one GPU is a 5.64x speedup on 8x the hardware. Third, batching is not free: on a B200 at 28 steps, Flux.1-Dev batch 8 took 15.57 s against 2.10 s at batch 1 (pytorch.org).

The arithmetic is short. Accepted replica rate is M/T; replicas needed are ceil(required accepted outputs divided by deadline seconds times that rate); procurement is replicas times GPUs per replica plus reserve that follows the stated fault policy. Expected attempts per accepted output are 1/p under independent retries (stats.libretexts.org). In the worked example, the same 600-clip, 8-hour requirement needs 50 GPUs as single-GPU replicas but 72 as 8-GPU replicas, and the answer flips if a per-clip tail limit applies. The deliverable described here is a workbook template, with the tabs listed later in this report, that publishes each configuration cell as pass, quality fail, unsupported or evidence missing.

Introduction and Background

Enterprise creative and media automation teams that move image or video generation onto owned GPUs face a question that generic GPU lists do not answer: what is the smallest practical pool that delivers a stated number of accepted outputs by a stated deadline? This report covers diffusion inference only. It excludes diffusion language models, video understanding inputs, training and fine-tuning, and it is not a general best-GPU list. It is written from the perspective of GPU Smith, a private AI infrastructure engineering consultancy. All configuration figures come from third-party sources; none are GPU Smith measurements.

Existing guidance already explains why video needs more memory than images. A GPU-cloud vendor's September 2026 guide gives a fleet formula of clips per hour times per-clip runtime times concurrency per GPU, which reads as vendor shorthand with unstated unit conventions (this report's formula below divides by the accepted replica rate instead) (www.onesourcecloud.net) and advises taking runtime and concurrency from a pilot rather than a published table (www.onesourcecloud.net). This report starts from that pilot-first advice and adds the denominator buyers are ultimately held to: outputs that pass review. The same guide states that doubling resolution or frame count roughly doubles the activation-dominated portion while weights stay constant (www.onesourcecloud.net); that is a general heuristic from a single vendor page, and the measured data later in this report show why scaling rules of that kind need validation on the exact runtime path.

The best public anchor for complete-output accounting is the ML.ENERGY benchmark. Its paper reports energy at the granularity of a whole generation response (arxiv.org), and its January 2026 results post defines a diffusion response as one generated image or video (ml.energy). The v3.0 release covers 46 models across 7 tasks and 1,858 configurations on H100 and B200 GPUs (ml.energy). The paper's data describe early-2025 runs on A100 40 GB and H100 80 GB GPUs (arxiv.org), so this report treats the paper and the later post as separate vintages and does not merge their numbers. The response-level framing has been reused as a motivating statistic in a USENIX ATC 2025 paper (www.usenix.org), and NVIDIA's June 2026 efficiency guidance points readers to benchmark-driven comparisons (developer.nvidia.com). Those are citations of the reference, not validation of any sizing result here.

Key Changes in How Diffusion Capacity Is Sized

From generations to accepted deliverables

The first change is the denominator. A GPU that produces 1,000 images per hour has delivered 1,000 accepted images only if every one passes the buyer's rubric. The ML.ENERGY paper makes a related point for latency: a diffusion result is useful only when the full image exists, so latency constraints apply to whole-image generation (arxiv.org). Acceptance extends the logic: a clip that is generated, decoded and encoded but rejected has consumed GPU time and delivered nothing.

Acceptance needs a named owner and rubric. Automated metrics are partial proxies. FID was introduced to capture the similarity of generated images to real ones better than the Inception Score (arxiv.org), yet a later study reports that FID contradicts human raters on text-to-image models (arxiv.org). For video, VBench decomposes quality into 16 dimensions (arxiv.org), and its authors state that existing metrics do not fully align with human perceptions (arxiv.org). The acceptance rate should therefore come from a fixed prompt and seed set scored by the party that will actually accept the media, with automated scores used as screens.

Human raters need a reliability check. Cohen's kappa is documented as a statistic that measures inter-annotator agreement (scikit-learn.org), and a peer-reviewed tutorial suggests no fewer than 30 comparisons (pmc.ncbi.nlm.nih.gov) and points to Krippendorff's alpha for multiple raters (pmc.ncbi.nlm.nih.gov). An audit with fewer than 30 doubly rated items would leave the acceptance rate poorly bounded.

From per-GPU numbers to whole replicas

The second change is the unit of capacity. xDiT distinguishes data parallelism, which runs independent prompts on separate model groups, from within-request strategies (Ulysses, Ring, PipeFusion, tensor and classifier-free-guidance parallelism) whose ranks cooperate on one output. Its launcher computes the GPU count of one group as the product of the parallel degrees, doubled when CFG parallelism is set (raw.githubusercontent.com). Diffusers' data-parallel path replicates the same model on each GPU and gives each GPU different prompts (huggingface.co).

A multi-GPU renderer is therefore one execution group, not several independent per-GPU jobs. Counting it as G units overstates concurrency; counting only its latency gain understates its cost. Layouts are also constrained by the model. xDiT requires the Ulysses degree to divide the query and key-value head counts unless the model supports head padding (raw.githubusercontent.com), and Diffusers notes that Ulysses cannot use more GPUs than the model has attention heads (huggingface.co). Diffusers says tensor parallelism needs a fast interconnect such as NVLink (huggingface.co). The L40S has no NVLink, while the H100 SXM offers 900 GB/s NVLink (per NVIDIA's specification tables (www.nvidia.com)), so the same layout can be a pass on one platform and unsupported on another.

From one stage to the full pipeline

The third change is stage accounting. A deliverable passes through prompt encoding, iterative denoising, VAE decoding, optional upscaling or interpolation, encoding, transfer and storage, and optional checks. NVIDIA states that roughly 90% of end-to-end SDXL processing is spent in the iterative UNet denoising steps (developer.nvidia.com). That dominance is configuration-specific: in the SDXL TensorRT card's four-step Latent Consistency Model timings, the VAE stage on an H100 (126.95 ms) is longer than the UNet (102.8 ms) (huggingface.co). When step count falls, the stage mix changes, and a profile taken at 50 steps does not describe a 4-step product.

Video adds stages with their own resources. FFmpeg documents frame-rate conversion by motion interpolation (ffmpeg.org) as a separate filter from duplication or dropping of frames (ffmpeg.org), so frames rendered by the model, frames after interpolation and the container frame rate are three different quantities. NVIDIA notes that encoding can use dedicated hardware on the GPU (developer.nvidia.com), so whether an encode stage competes with the denoiser depends on the runtime path. Moderation is per model: Diffusers describes the safety checker as a post-generation check on the output image (huggingface.co), so its rejections add retry work after the GPU time is spent.

From defaults to a pinned output contract

The fourth change is that every output-changing option belongs to the contract. Diffusers advises starting from the checkpoint's default scheduler and swapping only for a deliberate speed or quality tradeoff (huggingface.co). Step count is a model-variant property: guidance-distilled FLUX.1 takes about 50 sampling

steps for good-quality generation (huggingface.co), while FLUX.1 [schnell] is documented to produce high-quality images in 1 to 4 steps (huggingface.co). The ML.ENERGY paper observes that beyond some number of steps the gain may be virtually indistinguishable to human users (arxiv.org), while energy keeps rising linearly (arxiv.org). Fewer steps, lower precision or a different scheduler should be treated as a new configuration that must pass requalification.

Reproducibility rules constrain the seed policy. PyTorch states that completely reproducible results are not guaranteed across releases (docs.pytorch.org) and may differ between CPU and GPU executions even with identical seeds (docs.pytorch.org). Diffusers recommends a CPU generator because CPU random number generation is more stable across machines (huggingface.co). A retry policy must decide explicitly whether an attempt replays the same seed or draws a new one, and full determinism has a price, since deterministic operations are often slower (docs.pytorch.org).

Implementation Considerations and Process Changes

Build the output-contract and stage ledger first

Compare configurations only after the contract is written. *Table 1* below lists the fields to fix before any timing is recorded.

Ledger field	What to record	Why it changes capacity
Image dimensions	Native width and height, not post-upscale size	MLPerf SDXL generates 1024x1024 by default (mlcommons.org); a new shape triggers recompilation (huggingface.co)
Video frames and rate	Rendered frames, interpolation factor, container rate	Wan's VAE follows a 1 + T frame format (arxiv.org); LTX-Video requires 8n + 1 frames (github.com)
Scheduler and steps	Scheduler class, step count, guidance, flow shift	ML.ENERGY sweeps steps as a first-class knob (arxiv.org)
Precision and kernels	bf16, fp8, fp4, attention backend	Reduced precision can change output quality (github.com)
Model revision and license	Checkpoint hash, license terms	FLUX.1 [dev] generated outputs may be used commercially, subject to the license's output terms, but the weights fall under the FLUX.1 [dev] Non-Commercial License, so hosting them in a commercial pipeline is governed by separate terms that an enterprise should confirm with the licensor or counsel (huggingface.co); HunyuanVideo's license excludes the EU, UK and South Korea (github.com)

Ledger field	What to record	Why it changes capacity
Seed policy	Generator device, seed per attempt, retry rule	Results can differ between CPU and GPU even with identical seeds (docs.pytorch.org)
Postprocessing	Upscale, interpolate, encode settings	Slower x264 presets give better compression (trac.ffmpeg.org)
Acceptance	Rubric, rater count, decision owner	Metrics are proxies (arxiv.org)
GPU group	GPUs per replica, parallel degrees, interconnect	Group size is the product of the degrees (raw.githubusercontent.com)

Table 1 is a ledger, not a benchmark. Its value is that each row answers a question that is otherwise answered implicitly by whoever ran the test. A result recorded without these fields cannot be reproduced after a driver, runtime or model-revision change, and PyTorch's notes explain why identical seeds will not rescue it (docs.pytorch.org).

For each pipeline stage, also record which resource it uses and whether it overlaps with another stage:

- **Prompt encoding:** text encoder on the rendering GPU group, a separate GPU or CPU.
- **Denoising:** the iterative transformer or UNet passes, usually the dominant term.
- **VAE decoding:** may peak above the denoiser and may be sliced or tiled.
- **Upscaling or interpolation:** separate models or filters with their own memory.
- **Encoding:** CPU encoders or GPU hardware encoders, with their own throughput.
- **Transfer and storage:** network, object storage and retention.
- **Checks:** moderation and quality screens that can trigger retries.

Pin the runtime and enumerate unsupported cells

Pin the model, runtime, container and every output-changing option, then enumerate the cells that are known unsupported before measuring. Diffusers warns that group offloading may fail on models whose forward pass contains weight-dependent device casting of inputs (huggingface.co). Memory stages need separate peaks. VAE tiling divides an image into overlapping tiles (huggingface.co), but decoded tiles may show tone variation (huggingface.co), and xDiT says tile-local normalization and blending can change the output (raw.githubusercontent.com). Tiling is therefore a quality-affecting option to requalify, not a free memory saving.

Run a quality-preserving experiment matrix

Fix a representative prompt and seed set and a review protocol, then vary one factor at a time: steps, dimensions, batch size, precision, offload and hardware. Each cell ends in one of four statuses.

- **Pass:** the cell met the acceptance rate and tail target on the pinned contract, with artifacts hashed.
- **Quality fail:** the cell ran but the accepted fraction or rubric score fell below the contract.
- **Unsupported:** the runtime refused the layout, such as a head count the Ulysses degree does not divide (raw.githubusercontent.com), or a container that fails at startup when the GPU lacks memory for the requested precision (docs.nvidia.com).
- **Evidence missing:** no run exists. The cell stays on the sheet so nobody infers a result from a neighboring cell.

Lower precision deserves its own gate. NVIDIA reports that for FLUX.1 Kontext [dev] the transformer backbone saves approximately 2x memory at FP8 and 3x at FP4 (developer.nvidia.com), and its Torch-TensorRT results attach a speed gain to the precision change (on a B200 at 20 steps, 6.56 s for a batch of two images at the PyTorch baseline (developer.nvidia.com) versus 2.72 s with FP8 (developer.nvidia.com)). Speed that comes from a lossy setting cannot be banked until the accepted fraction is remeasured.

Measure end to end, and keep warmup honest

Record the following for every cell:

- **End-to-end batch time:** wall clock from request to stored artifact.
- **GPU memory peak:** per stage, not only the whole-pipeline maximum.
- **Host memory and transfer:** offload traffic and pinned-memory needs.
- **Accepted outputs:** attempted, accepted and rejected counts with reasons.
- **Failures and retries:** including out-of-memory events and restarts.
- **Energy:** joules over a window matched to the accepted-output count.

Keep cold-model startup separate from steady operation, but include it in any deadline that actually incurs it. Compilation is the largest hidden term: compiling the Flux transformer cut H100 latency from 6.7 s to 4.5 s (pytorch.org), while regional compilation took 9.6 s cold and 2.4 s with a warm cache (pytorch.org). Cache artifacts are valid only for the same PyTorch and Triton versions (docs.pytorch.org), and xDiT recommends reusing input shapes to lower compilation cost per run (raw.githubusercontent.com). A mixed-shape queue pays that cost repeatedly.

Benchmark runners usually exclude warmup. xDiT performs complete untimed pipeline calls before timing (raw.githubusercontent.com), and the Wan2.2 README reports averages of multiple samples taken after the warm-up phase (github.com). That is good steady-state practice but not the experience of the first request after a restart.

Size from measured accepted rate, then replay

Size only from measured accepted rate, then run a realistic backlog replay with output storage and failure recovery. Validate batch fill and mixed image and video interference in the replay, because isolated throughputs do not add. NVIDIA's NIM documentation notes that Wan2.2 containers select multi-GPU parallelism by GPU count (docs.nvidia.com), so a pool that mixes group sizes changes layouts, and therefore times.

Queueing sets headroom. Little's Law applies to the whole queueing system or to the queue or service facility alone (ocw.mit.edu), and in a single-server queue the time in system grows without bound as utilization approaches one (ocw.mit.edu). Retried work shares the pool with fresh work: the Google SRE book warns that retries can amplify low error rates into higher levels of traffic (sre.google) and that queued requests consume memory and increase latency (sre.google). Reserve capacity follows the fault policy; the SRE book describes provisioning to handle a simultaneous planned and unplanned outage (sre.google), which for GPU groups means whole replicas, not individual cards.

Output volume belongs in the replay. YouTube's published SDR upload recommendations give 8 Mbps for 1080p at standard frame rates (support.google.com) and are framed as recommendations, not requirements (support.google.com). At 8 Mbps, one second of video is about 1 MB, so a 5-second 1080p clip is about 5 MB (arithmetic from the published bitrate, not a measured file size). Rejected generations, intermediates and logs need a retention policy, and storage throughput should be exercised by the same replay as the GPUs.

Model and Configuration Reference Points

Table 2 below collects dated, attributed reference points with their stated conditions. They are not predictions for any other system: each number applies only to the model, precision, resolution, step count, batch size and hardware named in its source, and several sources do not state the batch size.

Model and source	Stated conditions	Memory statement	Timing or throughput statement
SDXL, MLPerf Inference	1024x1024, 20 steps, 5,000 validation samples (mlcommons.org)	Not stated in the benchmark description	8x L40S: 5 samples/s offline (developer.nvidia.com); 8x H200 at 700 W: 13.7 samples/s offline (developer.nvidia.com)
SDXL, TensorRT model card	30 steps, 1024x1024; baseline vs TensorRT (huggingface.co)	Not stated	A100: 0.27 to 0.36 images/s; H100: 0.40 to 0.68 images/s; A10: 0.10 to 0.12 images/s
FLUX.1 [dev], Hugging Face blog	H100 80 GB, 1024x1024, 50 steps, guidance 3.5 (huggingface.co)	BF16 peak 36.166 GB; pipeline weights about 31.447 GB (huggingface.co)	BF16 12 s; 8-bit bitsandbytes 27 s
FLUX.1 [dev], PyTorch blog	H100, 28 steps, bf16 (pytorch.org)	About 33 GB; CPU offload peak 22.7 GB	6.7 s baseline, 4.5 s compiled, 21.5 s with model CPU offload (pytorch.org)
FLUX.1 [dev], PyTorch Blackwell blog	B200, 1024x1024, 28 steps (pytorch.org)	BF16 38.34 GB at batch 1; 53.00 GB at batch 8	2.10 s at batch 1; 15.57 s at batch 8

Model and source	Stated conditions	Memory statement	Timing or throughput statement
Wan2.1 T2V-1.3B, vendor README	480P, RTX 4090 (github.com)	8.19 GB (github.com)	About 4 minutes for a 5-second video
Wan2.2 TI2V-5B and A14B, vendor README	1280*704 (5B) with offload flags (github.com); 1280*720 (A14B), single GPU (github.com)	24 GB minimum (5B); 80 GB minimum (A14B)	5B: under 9 minutes for a 5-second 720P video (github.com)
HunyuanVideo, vendor README	Batch size 1, 129 frames, 50 steps (github.com)	60 GB at 720px1280px129f; 45 GB at 544px960px129f; FP8 weights save about 10 GB (github.com)	1,904.08 s on 1 GPU; 337.58 s on 8 GPUs (GPU type not stated in the README text) (github.com)
LTX-2, PyTorch Blackwell blog	B200, 768x512, 121 frames, 24 fps, 40 steps, VAE tiling (pytorch.org)	BF16 72.77 GB at batch 1; 107.40 GB at batch 8	16.230 s at batch 1; 122.427 s at batch 8 (pytorch.org)

The table shows how little transfers between rows. A FLUX.1 [dev] BF16 pipeline of about 31 GB of weights peaks at 36 GB on an H100 at batch 1 (huggingface.co), and the Diffusers documentation says Flux needs about 50 GB of RAM or VRAM to load all components unoptimized (huggingface.co). Different peak definitions (weights, peak activation, whole-pipeline load) yield different "requirements" for the same model. The two H100 Flux entries agree on per-step cost: 6.7 s over 28 steps and 12 s over 50 steps are both about 0.24 s per step, which supports a linear step term for the denoiser on that setup but says nothing about other hardware.

Images: what the numbers support

For images per hour per GPU, the published figures are system-level or configuration-specific. The MLPerf results are for eight-GPU systems, so dividing by eight gives about 0.63 samples/s per L40S (2,250 per hour) and about 1.71 samples/s per H200 (6,165 per hour); these are derived values that assume even scaling inside the system, not published per-GPU results. MLPerf SDXL also binds quality: a valid submission must achieve FID and CLIP scores within a stated range (mlcommons.org), and MLCommons added a human compliance check (mlcommons.org), which is the accepted-output principle at benchmark scale. The model card figures translate to 1,440 to 2,448 images per hour per H100 at 30 steps (0.40 to 0.68 images/s times 3,600), but the card states neither batch size nor acceptance criteria.

For H100 versus L40S, the published sources cited here do not report both on the same model and settings. MLPerf v4.0's nearest pair is an 8x H200 system against an 8x L40S system, 13.7 versus 5 samples/s (about 2.7x) on SDXL at 20 steps; the H200 shares the Hopper architecture with the H100 but carries more memory. The H100 SXM lists 80 GB and 3.35 TB/s of memory bandwidth against 48 GB of GDDR6 and 864 GB/s for the L40S (www.nvidia.com). NVIDIA's NIM support matrix lists FLUX.1-dev engines for both an H100 SXM and an L40S (docs.nvidia.com). The ratio is a hypothesis to test per workload, not a constant.

For Flux class models, a single 1024x1024 image usually needs one GPU: FLUX.1 [dev] is a 12 billion parameter rectified flow transformer (huggingface.co) and fits in about 36 GB at BF16 on an H100 (Table 2). FLUX.2 [dev] is a 32 billion parameter model (huggingface.co) paired with a 24B parameter vision-language text encoder (bfl.ai), so its memory and layout need fresh qualification rather than extrapolation from FLUX.1.

Video: why memory and time do not scale from one number

Wan2.1's 1.3B model is recommended at 480P although it can produce 720P (github.com); Wan2.2's A14B models need at least 80 GB for a single-GPU 720P run, and the 5B model runs on 24 GB with offload flags (Table 2). The Wan paper states that frame counts follow a 1 + T format (arxiv.org), that typical sequences are 81 frames for 1280 x 720 video (arxiv.org), and that the VAE limits frames per processing chunk to at most 4 to avoid memory overflow (arxiv.org). Part of the decode memory is therefore bounded by design, one reason a linear memory model from frame count is unreliable.

Measured batch data confirm that scaling is not proportional. For LTX-2 on a B200, batch 8 took 122.427 s against 16.230 s at batch 1, or 15.3 s per video, only about 6% better per clip, while peak memory rose from 72.77 GB to 107.40 GB (about 48%) (pytorch.org). For Flux.1-Dev on the same GPU, batch 8 gave 1.95 s per image against 2.10 s, about 7% better, with 38% more peak memory. Batching bought little throughput at a large memory cost, consistent with the PyTorch team's description of diffusion models as heavily compute-bound (pytorch.org) and ML.ENERGY's observation that diffusion models cannot run as many concurrent generations as language models (arxiv.org).

Video also has contract traps. LTX-Video works best on resolutions under 720 x 1280 and below 257 frames (github.com), and Wan2.1's 14B text-to-video task samples 50 steps while image-to-video uses 40 (github.com), so a mixed queue has different per-job costs even at one resolution. A rack-scale anchor exists but does not translate to a small pool: NVIDIA's September 2026 blog on MLPerf Inference v6.1 reports 0.65 720p videos per second at 5.7 seconds per video on a GB300 NVL72 system (blogs.nvidia.com), with the benchmark using Wan-2.2-T2V-A14B and the VBench dataset (www.nvidia.com). The source does not state whether this result is for the server or offline scenario, so no per-GPU rate is inferred.

Data Analysis and Evidence

The evidence falls into three layers: benchmark methodology (what a number counts), configuration-specific timings (what it was measured on) and the arithmetic that converts a measured rate into hardware. The first two are third-party; the third is this report's contribution and is labeled as such.

What the leaderboard layer does and does not show

ML.ENERGY's v2 paper measures time and energy per configuration with the Zeus library (arxiv.org), and because diffusion models are batched as a whole, per-request energy is the whole-batch energy divided by the batch size (arxiv.org). The January 2026 post used default denoising steps, resolution and frames for all models (ml.energy), reports text-to-image models from 0.6B to 12B parameters with 20 to 50 denoising steps (ml.energy), and reports 26 kJ to 1.16 MJ for a single video (ml.energy). For B200 versus H100 at matched latency, B200 won 86% (18 of 21) of text-to-image comparisons with a median 15% energy reduction (ml.energy), using whole-generation latency constraints of 10, 30 and 60 s for text-to-image and 100, 500 and 1,000 s for text-to-video (ml.energy), a useful template for stating deadlines per output rather than per token.

The limits matter as much. The leaderboard is interactive, so this report cites no current slider values or computed winners, and the paper reports energy mostly in figures. Neither source reports rejection or retry rates. They support the framing (complete outputs, step count as a knob, batch-level measurement), not a fleet

size.

The planning calculation

For M accepted generations measured in T seconds on a G-GPU replica:

- **Accepted replica rate:** M/T.
- **GPU-seconds per accepted output:** G times T divided by M, with idle and cold-start scope declared.
- **Replica lower bound:** ceil(required accepted outputs divided by (deadline seconds times the observed accepted replica rate)).
- **Procurement count:** replicas times G plus an explicitly justified reserve.
- **Energy per accepted output:** measured matched-window joules divided by accepted outputs; kWh is joules divided by 3,600,000.

If a rejection model is used for planning, expected attempts per accepted output are 1/p, because the attempt number of the first success in Bernoulli trials is geometric (stats.libretexts.org). That assumes independent attempts and a stationary acceptance rate, which fail when a bad prompt batch or a changed model revision shifts p; replace the model with measured accepted throughput as soon as it exists. For heterogeneous work, sum empirical replica-seconds per cell and confirm with a replay.

Image example (Hypothetical Example)

Assume a 1,024x1,024 contract with a fixed revision, scheduler and step count. A single-GPU replica is measured for 3,600 s: 2,400 attempted images, 2,040 accepted (85%). The accepted rate is 2,040/3,600 = 0.567 per second. The requirement is 20,000 accepted images in 8 hours (28,800 s). One replica delivers 16,320 accepted images in that window, so replicas = ceil(20,000/16,320) = 2, running at about 61% of capacity. With an N+1 reserve, procurement is 3 GPUs. If the matched window used 1,020,000 J, energy per accepted image is 500 J and the job totals 10,000,000 J, about 2.78 kWh. Every number here is invented for illustration and none is a benchmark.

Video example (Hypothetical Example)

This example combines the latencies published in the HunyuanVideo README (1280x720, 129 frames, 50 steps, 1 to 8 GPUs via xDiT) (github.com) with an assumed 80% acceptance rate, a 600-clip requirement and an 8-hour (28,800 s) deadline. The README does not state the GPU type, its table caption reads "on 8xGPU" while its only hardware statement is that the model "is tested on a single 80G GPU", so the test setup is ambiguous and the 1-GPU row inherits that uncertainty; Table 3 is purely illustrative. The 3, 5 and 6 GPU layouts it also lists are omitted. *Table 3* below shows the result.

Layout	Latency per attempt	Speed up	GPU-seconds per attempt	Accepted per replica in 8 h	Replicas for 600	GPUs (no reserve)
1 GPU per replica	1,904.08 s	1.00x	1,904	12.10	50	50

Layout	Latency per attempt	Speed up	GPU-seconds per attempt	Accepted per replica in 8 h	Replicas for 600	GPUs (no reserve)
2 GPUs per replica	934.09 s	2.04x	1,868	24.66	25	50
4 GPUs per replica	514.08 s	3.70x	2,056	44.82	14	56
8 GPUs per replica	337.58 s	5.64x	2,701	68.25	9	72

Table 3 shows that if the objective is only 600 accepted clips in 8 hours, the 1-GPU and 2-GPU layouts tie at 50 GPUs and the 8-GPU layout needs 72, about 44% more hardware, because the 8-GPU group spends about 42% more GPU-seconds per clip. The result reverses under a per-clip latency limit: a 1-GPU attempt takes about 32 minutes and an 8-GPU attempt about 5.6 minutes, so a 15-minute tail target rules out the single-GPU layout. Reserve granularity differs too: one spare replica adds 1 GPU to the 1-GPU plan (51) but 8 GPUs to the 8-GPU plan (80). The 1-GPU row also requires that the model fit on one GPU, which the README's 60 GB minimum for this shape makes plausible on an 80 GB card. Because acceptance and speedup interact with the real runtime path, this table is a template, not a recommendation.

Cross-checking published speedups

Published multi-GPU gains support caution. In Diffusers' context-parallel benchmark on 4 H100 GPUs at 50 steps per image, Ulysses took 6,670.789 ms per image while Ring took 13,076.492 ms (huggingface.co), so the strategy choice on the same four GPUs changed time by almost 2x. xDiT says each additional device adds communication and synchronization, so the workload should be measured (raw.githubusercontent.com), and a hybrid layout such as two CFG branches each with four Ulysses ranks uses eight devices for one request (raw.githubusercontent.com). Its determinism check compares timed outputs against the first run with exact equality (raw.githubusercontent.com), a useful regression test when changing layouts.

Implications and Future Directions

For procurement teams, the number to request from a vendor or produce in a pilot is accepted outputs per whole replica per deadline window, with the contract attached, not raw frames or generic throughput. Single-GPU replicas favor throughput deadlines when the model fits, multi-GPU groups buy tail latency at a GPU-seconds premium, and reserve granularity grows with group size. Offloading can turn a fit into a slowdown: PyTorch's H100 measurements show model CPU offload taking 21.5 s against 6.7 s for the same Flux pipeline while peak memory fell from about 33 GB to 22.7 GB (pytorch.org).

For platform engineers, stage ownership should be explicit. Whether encoding, interpolation and moderation run on the rendering group, on separate GPUs or on CPUs determines whether they overlap with denoising or add to it, and overlapping stages cannot simply be summed. Warmup and compile policy deserve a service-level decision, since a resolution change triggers recompilation (huggingface.co) and cache artifacts are version-bound (docs.pytorch.org).

Three uncertainties remain. Benchmark coverage is moving: MLPerf Inference now lists a Wan 2.2 text-to-video workload (www.nvidia.com), but results must be read for their stated scenario. Model families change quickly, and FLUX.2 [dev] at 32 billion parameters (huggingface.co) shows a successor can change memory needs several-fold, so each sizing sheet should carry a revision stamp. And the quality measures in common use remain proxies (arxiv.org), so the acceptance owner's rubric is the durable part of the workbook. Keep the workbook under version control so a runtime, driver or hardware change triggers a re-run of the affected cells rather than a re-argument. A minimal workbook has these tabs:

- **Output contract:** the ledger in Table 1, with model revision and rubric.
- **Configuration matrix:** model, shape, scheduler, steps, precision, GPU group and runtime per cell.
- **Stage timings:** end-to-end and per-stage times, memory peaks and energy window.
- **Attempt log:** attempted, accepted, rejected and retried counts with decisions and reviewers.
- **Sensitivity:** deadline, reserve and acceptance-rate variations, as in Table 3.
- **Provenance:** artifact hashes, timestamps and re-run instructions.

Frequently Asked Questions (FAQs)

How many GPUs do I need for Stable Diffusion at volume?

Divide the accepted outputs required by the deadline in seconds times the measured accepted replica rate, round up, and multiply by GPUs per replica. For a single-GPU SDXL or Flux replica the replica count equals the GPU count. The published SDXL range of 0.40 to 0.68 images/s on an H100 at 30 steps (huggingface.co) is a starting hypothesis only, because it carries no acceptance rate.

What are the self-hosted image generation GPU requirements, and the FLUX VRAM figures?

They depend on the model, precision and options. FLUX.1 [dev] at BF16 measured a 36.166 GB peak on an H100 (huggingface.co). Quantization and offload lower memory but change speed and sometimes output, so each is a separate workbook cell.

What GPU memory does Wan 2.1 video generation need?

The vendor README states 8.19 GB for the 1.3B model (github.com). Larger Wan models need far more, and HunyuanVideo states 60 GB for 720px1280px129f, so treat any single figure as valid only for its stated shape.

How do I size GPUs for private image generation and plan on-premise capacity?

Write the output contract, measure accepted outputs per replica on the pinned pipeline, add a measured retry term, set reserve by fault policy, then replay a realistic backlog with storage and failure recovery.

H100 or L40S for image generation throughput?

No same-settings public pair is cited here. MLPerf v4.0 compares eight L40S GPUs (5 samples/s) with eight H200 GPUs (13.7 samples/s) on SDXL (Table 2). Test both on your contract.

Conclusion

Diffusion GPU sizing is a measured-throughput exercise over a defined output contract. The unit is the accepted deliverable per whole replica per deadline window, after rejection and retries, and the replica is the full execution group that renders it. The public record supplies methodology and reference points, such as complete-output accounting in ML.ENERGY, fixed-step MLPerf SDXL rules and vendor memory statements, but each is tied to a model, shape, precision and hardware that may not match a production mix.

The practical path is short:

- **Define:** write the output contract and acceptance owner.
- **Pin:** fix the runtime and every output-changing option.
- **Qualify:** run a quality-preserving matrix with a rater-checked acceptance rate.
- **Measure:** record end-to-end accepted rate per replica.
- **Compute:** derive the replica lower bound and the reserve the fault policy requires.
- **Replay:** run a backlog with storage and failures.
- **Publish:** mark each cell pass, quality fail, unsupported or evidence missing.

A sheet built that way survives hardware and runtime changes because every number carries its conditions.

Source links

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

1. https://mlcommons.org/2024/08/sdxl-mlperf-text-to-image-generation-benchmark
2. https://developer.nvidia.com/blog/nvidia-h200-tensor-core-gpus-and-nvidia-tensorrt-llm-set-mlperf-llm-inference-records
3. https://huggingface.co/stabilityai/stable-diffusion-xl-1.0-tensorrt
4. https://github.com/Tencent-Hunyuan/HunyuanVideo
5. https://github.com/Wan-Video/Wan2.2
6. https://github.com/Wan-Video/Wan2.1
7. https://ml.energy/blog/measurement/energy/diagnosing-inference-energy-consumption-with-the-mlenergy-leaderboard-v30
8. https://arxiv.org/html/2505.06371v2
9. https://raw.githubusercontent.com/xdit-project/xDiT/main/docs/methods/usp.md
10. https://pytorch.org/blog/faster-diffusion-on-blackwell-mxfp8-and-nvfp4-with-diffusers-and-torchao
11. https://stats.libretexts.org/Bookshelves/Probability_Theory/Probability_Mathematical_Statistics_and_Stochastic_Processes_%28Siegrist%29/11%3A_Bernoulli_Trials/11.03%3A_The_Geometric_Distribution
12. https://www.onesourcecloud.net/cms/video-diffusion-gpu-requirements.html
13. https://www.usenix.org/system/files/atc25-tian.pdf
14. https://developer.nvidia.com/blog/maximize-ai-factory-energy-efficiency-through-full-stack-inference-and-training-optimizations
15. https://arxiv.org/abs/1706.08500
16. https://arxiv.org/abs/2401.09603
17. https://arxiv.org/abs/2311.17982

18. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.cohen_kappa_score.html
19. https://pmc.ncbi.nlm.nih.gov/articles/PMC3900052
20. https://raw.githubusercontent.com/xdit-project/xDiT/main/docs/runner/launching.md
21. https://huggingface.co/docs/diffusers/en/optimization/data_parallelism
22. https://huggingface.co/docs/diffusers/en/optimization/context_parallelism
23. https://huggingface.co/docs/diffusers/en/training/distributed_inference
24. https://www.nvidia.com/en-us/data-center/h100
25. https://ffmpeg.org/ffmpeg-filters.html
26. https://developer.nvidia.com/video-codec-sdk
27. https://huggingface.co/docs/diffusers/en/conceptual/ethical_guidelines
28. https://huggingface.co/docs/diffusers/en/using-diffusers/schedulers
29. https://huggingface.co/docs/diffusers/en/api/pipelines/flux
30. https://huggingface.co/black-forest-labs/FLUX.1-schnell
31. https://docs.pytorch.org/docs/stable/notes/randomness.html
32. https://huggingface.co/docs/diffusers/en/using-diffusers/reusing_seeds
33. https://huggingface.co/docs/diffusers/en/optimization/fp16
34. https://arxiv.org/pdf/2503.20314
35. https://github.com/Lightricks/LTX-Video
36. https://github.com/xdit-project/xDiT
37. https://huggingface.co/black-forest-labs/FLUX.1-dev
38. https://github.com/Tencent-Hunyuan/HunyuanVideo/blob/main/LICENSE.txt
39. https://trac.ffmpeg.org/wiki/Encode/H.264
40. https://huggingface.co/docs/diffusers/en/optimization/memory
41. https://raw.githubusercontent.com/xdit-project/xDiT/main/docs/methods/parallel_vae.md
42. https://docs.nvidia.com/nim/visual-genai/1.7.1/support-matrix.html
43. https://developer.nvidia.com/blog/optimizing-flux-1-kontext-for-image-editing-with-low-precision-quantization
44. https://developer.nvidia.com/blog/double-pytorch-inference-speed-for-diffusion-models-using-torch-tensorrt
45. https://pytorch.org/blog/torch-compile-and-diffusers-a-hands-on-guide-to-peak-performance
46. https://docs.pytorch.org/tutorials/recipes/torch_compile_caching_tutorial.html
47. https://raw.githubusercontent.com/xdit-project/xDiT/main/docs/runner/benchmarking.md
48. https://ocw.mit.edu/courses/esd-86-models-data-and-inference-for-socio-technical-systems-spring-2007/6f188063c7051ea94c5db10683665f21_lec9.pdf
49. https://ocw.mit.edu/courses/2-854-introduction-to-manufacturing-systems-fall-2016/927056a1af54772a587fd84ad4951e71_MIT2_854F16_Mm1Queue.pdf
50. https://sre.google/sre-book/service-best-practices
51. https://sre.google/sre-book/addressing-cascading-failures
52. https://support.google.com/youtube/answer/1722171

53. https://huggingface.co/blog/diffusers-quantization
54. https://huggingface.co/black-forest-labs/FLUX.2-dev
55. https://bfl.ai/blog/flux-2
56. https://pytorch.org/blog/presenting-flux-fast-making-flux-go-brrr-on-h100s
57. https://blogs.nvidia.com/blog/vera-rubin-nvl72-mlperf-inference
58. https://www.nvidia.com/en-us/data-center/resources/mlperf-benchmarks
59. https://raw.githubusercontent.com/xdit-project/xDiT/main/docs/methods/hybrid.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.