GPU SMITH
COMPUTE ENGINEERING

GPU SMITH / RESEARCH & PRACTICAL GUIDANCE

Batch vs Real-Time LLM Inference: GPU Fleet Sizing Guide

> 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 · 2026-10-13 · batch vs real-time llm inference · llm inference gpu fleet sizing · batch inference · real-time inference · continuous batching · vllm · triton · gpu capacity planning · gpu hours

Original article: https://gpusmith.com/articles/en/batch-vs-real-time-llm-inference-gpu-sizing

In this article

1. Executive Summary
2. Introduction and Background
3. Batch LLM Inference
4. Real-Time LLM Inference
5. Feature Comparison
6. Performance and Benchmarks
7. Data Analysis and Evidence
8. Implementation and Admission Control
9. Implications and Future Directions
10. Frequently Asked Questions (FAQs)
11. Conclusion

Executive Summary

Batch vs real-time large language model (LLM) inference is principally a choice of completion contract. A batch backlog must finish before a deadline; an interactive service must satisfy latency and availability objectives while requests arrive. Both can use continuous batching, which reschedules work during generation, so batch scheduling and engine batching are separate decisions. Hugging Face documents that stepwise mechanism; MLPerf Endpoints distinguishes an offline dataset submitted together from a serving performance curve across concurrency. (huggingface.co) (docs.mlcommons.org) (docs.mlcommons.org)

For a private fleet, use **SLO-qualified capacity**, meaning throughput accepted while the interactive service meets its service-level objective (SLO). DistServe formalizes goodput around SLO attainment, and **vLLM** exposes goodput thresholds for time to first token, time per output token and end-to-end latency. Peak tokens per second therefore cannot, by itself, answer how many graphics processing units (GPUs) an enterprise needs. (arxiv.org) (docs.vllm.ai) The recommended engineering sequence is to measure representative workloads, enforce online objectives, and convert the remaining capacity into a deadline budget. The procurement recommendation should therefore optimize accepted work within the latency budget and verify any proposed saving through the fleet replay.

The worksheet below is a **hypothetical example**, not a measured hardware benchmark: **300 million useful tokens**, a **7-hour window**, **30 minutes of total non-processing overhead**, and **1,200 accepted tokens/s/GPU** imply **14 GPUs** after **20% throughput headroom**. Completion takes approximately **6.70 hours**, leaving **18 minutes**; reserving the entire window consumes **98 GPU-hours**. Those results follow the displayed arithmetic and assumed trace, using the SLO-qualified measurement principle above. (arxiv.org) A shared fleet with only the worksheet's assumed residual capacity cannot finish that backlog, even if utilization appears low. Count capacity by time slot, model residency and feasible replica placement.

Prefer spare-capacity reuse when a mixed-load replay demonstrates sufficient deadline capacity and acceptable online tails. Prefer a separate pool when deadlines overlap interactive peaks or isolation simplifies acceptance. Prefer a nightly window when its capacity is dependable and it includes loading, draining and recovery. Slurm backfill protects expected higher-priority start times, but suspended GPU jobs retain their

generic resources; infrastructure priority alone does not establish immediate accelerator reclamation. (slurm.schedmd.com) (slurm.schedmd.com) Public application programming interface (API) discounts are separate commercial terms and do not establish private-cluster savings. The procurement decision should compare incremental expenditure, whole-node commitments and opportunity cost, with the online acceptance test attached.

Introduction and Background

Enterprise inference platforms often serve a mixed workload: user-facing assistants, search and retrieval, plus deferrable summarization, extraction, evaluation and document processing. The practical question is whether a backlog can occupy spare capacity without degrading the live service, whether a separate pool is justified, or whether work should run during a protected calendar window. This report addresses that private-cluster decision, using documentation accessed for the **October 2026** publication anchor. Rolling documentation describes mechanisms; it does not certify the release installed on a particular cluster.

A **large language model (LLM)** request normally involves prompt processing, or **prefill**, followed by generated-token production, or **decode**. Mixing those stages matters. DistServe's research examines their interference and defines goodput through latency attainment, rather than aggregate throughput alone. (arxiv.org) A total-token accounting unit is useful only when its benchmark preserves the workload's input/output mix and length distribution. A prompt token and an output token are not interchangeable units of accelerator work.

Offline batching describes an external completion window. **Continuous batching** describes an internal serving mechanism that replaces completed work while other sequences remain active. An online server can continuously batch user requests, and an offline job can send records through the same engine. (huggingface.co) Hugging Face documents both a blocking batch interface and a manager suitable for streaming or arriving requests. (huggingface.co) The architecture comparison therefore concerns scheduling contracts, not a binary engine feature.

GPU Smith's stated cluster-sizing scope derives topology, node count and serving stack from workload models and throughput targets. (gpusmith.com) That provides an adjacent engineering perspective for this report: specify the acceptance workload before selecting hardware. It does not supply comparative performance evidence. All illustrative throughput, latency, headroom and cost inputs below are explicitly assumed; no client result or target-system measurement is represented as observed.

Batch LLM Inference

Capabilities

Batch inference turns a finite, auditable workload into a completion obligation. A platform should identify the accepted records, their processing cutoff, and the exact point at which output becomes durable. Ray Data documents checkpoint-based recovery that skips already processed records, providing a useful mechanism for restartable document pipelines. (docs.ray.io) The capacity contract still needs to define whether a record is complete after model generation, validation, persistence or downstream indexing.

The recommended workload manifest includes:

- **Records:** unique identifiers, input location, partitioning and the arrival cutoff.
- **Lengths:** joint input/output token distributions, including long-context tails.
- **Generation:** output caps, stop rules, sampling parameters and task-specific validation.
- **Deadline:** timezone, completion timestamp and treatment of late arrivals.
- **Errors:** retry ceiling, permanent-failure disposition and acceptable incomplete records.
- **Boundary:** tokenization, preprocessing, generation and output persistence included in timing.

These are engineering acceptance inputs, not universal defaults. Keep the original input population alongside attempted and accepted outputs; checkpoint recovery should retain completed-record identity. (docs.ray.io) An apparently completed job can still be operationally incomplete if records were omitted, retries were abandoned or results failed schema validation.

Adoption

Batch processing fits deferrable work when stakeholders accept a completion window rather than immediate responses. The relevant evidence here concerns implemented mechanisms, not market share. Ray's batch documentation describes separating preprocessing and inference so that central processing unit (CPU) work can overlap accelerator execution. (docs.ray.io) Such pipelining can change the system bottleneck; the entire pipeline must be measured if its completion time is the contractual result.

An offline engine test helps characterize the accelerator path. vLLM's benchmark command family distinguishes offline throughput and online serving tests. (docs.vllm.ai) It should be supplemented by the production record pipeline, because a fast generation loop does not establish a fast ingest-to-durable-output job. A capacity estimate must state which boundary its denominator covers.

Strengths and Limitations

The operational strengths are freedom to reorder independent records, use broader concurrency sweeps, and defer work until capacity is available. Those are design opportunities, not promised improvements. Triton says batching **typically** increases throughput, which leaves performance dependent on the model and execution configuration. (docs.nvidia.com)

A batch workload can still exhaust memory or interfere with latency-sensitive work. Ray recommends reducing inference batch size when out-of-memory errors occur. (docs.ray.io) Large documents may also require splitting or a different placement class. The team should benchmark each material length bucket and replay their production proportions. Optimizing solely for the average document can miss the records that determine final completion time.

Real-Time LLM Inference

Capabilities

Real-time inference has an arrival process and a latency contract. **Time to first token (TTFT)** measures response-start delay; **time per output token (TPOT)** measures generation pacing; end-to-end latency includes completion. vLLM accepts separate goodput objectives for these metrics. (docs.vllm.ai) The service should additionally define error, rejection and availability objectives, because excluding rejected requests can make latency look favorable while useful service declines.

Continuous batching belongs inside this architecture as well. It admits newly available work during generation rather than waiting for an entire fixed batch to finish. (huggingface.co) This creates a tuning problem: concurrency that improves aggregate throughput may consume the latency budget. Keep arrival rate, active sequences and per-iteration token budget distinct. (docs.vllm.ai)

Useful online acceptance measures include:

- **Arrival rate:** offered traffic at the client and admitted traffic at the server.
- **Latency tails:** pooled p95 and p99 TTFT, completion latency and generation pacing.
- **Acceptance:** successful responses, rejections, disconnects and timeouts.
- **Memory:** cache occupancy, allocator pressure and sequence-length outliers.
- **Recovery:** behavior during a burst, replica restart or batch-worker drain.

Adoption

Interactive workloads require tests that reproduce arrivals rather than only maintaining a fixed number of concurrent clients. vLLM can synthesize Poisson or gamma arrivals; it also warns that a concurrency cap can make actual request rate lower than the requested rate. (docs.vllm.ai) (docs.vllm.ai) A saturated closed-loop test can therefore understate the externally offered load. Record both rates and replay the trace's burst structure where possible. (docs.vllm.ai)

The acceptance population should represent real request classes. Short answers, long-form generation and retrieval-heavy prompts may need different thresholds. Keep the class labels through measurement so that a dominant short-request class cannot hide unacceptable performance for the long-document class.

Strengths and Limitations

Online fleets can expose idle intervals to deferrable work, but **idle utilization is not reserved deadline capacity**. Memory residency, arrivals and latency headroom constrain what a second workload may consume. The original PagedAttention paper identifies cache fragmentation and redundant duplication as batch-size constraints; Hugging Face describes allocating cache blocks as needed. (arxiv.org) (huggingface.co) Neither statement proves that every apparently unused GPU cycle can safely run batch work.

Observability also needs correct aggregation. Prometheus cautions that averaging summary quantiles across replicas is statistically invalid; histograms support quantiles calculated from aggregated distributions. (prometheus.io) (prometheus.io) Measure service-wide tails and retain per-replica diagnostics. Otherwise, a mixed-load experiment can miss the slow or overloaded replica that determines the user experience.

Feature Comparison

The central choice is where the completion obligation sits relative to interactive capacity. *Table 1* compares the operating contracts. It is an architecture matrix, not a product ranking or a claim of universal cost superiority.

Criterion	Shared online fleet	Separate batch pool	Protected nightly window
Acceptance contract	Online SLO plus batch deadline	Independent online and batch acceptance	Deadline plus capacity returned before online ramp
Capacity denominator	Accepted residual throughput during mixed load	Accepted throughput on batch replicas	Accepted throughput during available calendar slots
Resource control	Admission caps and tested interference limits	Explicit GPU allocation and placement	Reservation, draining and model-readiness boundary
Useful scheduling mechanism	Request priority may reorder eligible queued work. (docs.nvidia.com)	Kubernetes labels/selectors identify eligible GPU nodes. (kubernetes.io)	Slurm backfill fits work around expected higher-priority starts. (slurm.schedmd.com)
Cost question	Added energy, operations and capacity displaced	Incremental nodes plus supporting infrastructure	Window commitments, transition costs and displaced work
Evidence to demand	Mixed-load replay with stable online tails	Offline completion test on the proposed pool	Repeated calendar-window completion and release test

The shared option has the strongest dependence on traffic timing. A separate pool simplifies interference accounting but can add whole-node commitments. A protected window can reuse hardware while retaining a clear boundary, provided transition time and the return-to-service test are included. These are decision criteria; the worksheet determines whether the actual quantities support them.

Do not conflate resource scheduling with physical isolation. Ray states that logical resources primarily provide scheduling admission control and do not impose physical-usage limits. (docs.ray.io) (docs.ray.io) Kubernetes GPU requests require limits, and heterogeneous GPU placement can use node labels and selectors. (kubernetes.io) Those controls identify feasible placement but do not themselves demonstrate acceptable co-resident inference latency.

Preemption deserves its own acceptance step. Kubernetes documents a time gap caused by graceful termination during Pod preemption. (kubernetes.io) Slurm can set preemption policy by partition or quality of service, while suspended jobs retain generic resources such as GPUs. (slurm.schedmd.com) (slurm.schedmd.com) Consequently, a design that depends on reclaiming batch capacity should test cancellation, memory release, draining and model readiness. Request-level priority inside a server and terminating a cluster job operate at different boundaries.

Performance and Benchmarks

A benchmark contract that can be reproduced

A transferable benchmark needs a complete execution identity. Pin the engine release and container digest, model revision or hash, tokenizer revision, precision, quantization recipe, device type, device count and parallelism. Record the operating system, driver, firmware, interconnect and storage path. PyTorch explicitly does not guarantee reproducibility across releases, commits or platforms, supporting the need to retain that environment record. (docs.pytorch.org)

The benchmark package should contain:

- **Trace:** length histograms, joint length buckets, request classes and arrival timestamps.
- **Model:** exact weights hash, tokenizer, context limit and precision configuration.
- **Engine:** release, image digest, launch command and resolved configuration.
- **Timing:** model load, warmup, tokenization, preprocessing and persistence boundaries.
- **Results:** raw per-request timing, errors, retries and aggregate accepted throughput.
- **Hardware:** topology, physical GPU allocation, memory and measured node power.
- **Repeatability:** independent repetitions and the spread of completion results.

This report does not name a supposedly universal latest release. Its fetched documentation includes rolling aliases and versioned pages; use the documentation matching the selected deployment. Publish the exact command actually executed alongside the resulting artifacts, rather than treating an unexecuted command template as evidence.

PyTorch's Timer performs warmups and synchronizes asynchronous accelerator work when necessary; its guidance recommends replicates to quantify noise. (docs.pytorch.org) (docs.pytorch.org) For end-to-end inference, apply the same principle to the whole timed pipeline. Report cold start and warm steady state separately, then include whichever transitions the completion contract actually requires.

Engine and scheduler sweeps

Triton's dynamic batcher is documented for stateless models; its sequence batcher routes a request sequence to the same model instance. (docs.nvidia.com) (docs.nvidia.com) An autoregressive LLM's state does not mean generic sequence batching automatically implements its decode scheduler. Verify the selected backend's scheduling interface and supported configuration. Use the documented combination that the target engine actually executes.

Model Analyzer characterizes memory requirements under batching and instance configurations and can sweep Triton configuration values. (docs.nvidia.com) (docs.nvidia.com) Its throughput metric counts **inferences per second**, not tokens per second, and it separately exposes queue-time measurements. (docs.nvidia.com) (docs.nvidia.com) Convert request throughput into token throughput only with the actual trace and an explicit token accounting definition.

For vLLM, distinguish the maximum sequences from the token budget per iteration. (docs.vllm.ai) (docs.vllm.ai) Chunked prefill divides large prompts into smaller chunks mixed with decode work. (docs.vllm.ai) The sweep should therefore vary arrival rate, concurrency, sequence limits, token budget, cache settings and replica count. Change a small number of factors at a time, and retain rejected configurations as evidence of the feasible boundary.

Online interference and the Pareto frontier

Start with an online-only baseline, then add batch traffic while keeping the same offered interactive trace. Track p95/p99 latency, accepted requests and errors. Model Analyzer exposes p99 latency, and vLLM can retain per-request errors and timing detail. (docs.nvidia.com) (docs.vllm.ai) Stop increasing batch traffic when any declared online acceptance condition fails, not when average utilization reaches a chosen target.

Table 2 is a **Hypothetical Example** of a queue-delay and batch-size sweep. Every entry is invented for teaching the selection rule, not attributed to Triton, vLLM or a GPU. Assume an online p99 TTFT limit of **500 ms**. The configurable delay mechanism is documented by Triton; MLPerf Endpoints describes interpreting performance across a Pareto curve. (docs.nvidia.com) (docs.mlcommons.org)

Configured queue delay	Batch cap	Accepted batch tokens/s per GPU	Online p95 TTFT	Online p99 TTFT	Decision
0 microseconds	8	700	180 ms	320 ms	Eligible
0 microseconds	16	950	220 ms	380 ms	Eligible
2,000 microseconds	32	1,200	280 ms	460 ms	Eligible frontier candidate
5,000 microseconds	64	1,350	390 ms	650 ms	Exclude: online threshold exceeded

The fastest row is unsuitable under the assumed contract. The eligible frontier candidate is only a candidate: generation pacing, completion latency, rejection rate and recovery still need to pass. Batch-cap meanings also differ between engines; the illustrative column is not a portable configuration parameter. Keep this mixed-load table separate from offline results.

Data Analysis and Evidence

What primary measurements establish

Published benchmark contracts illustrate why a headline throughput number needs context. MLCommons' **September 2025 MLPerf Inference 5.1 Llama3.1-8B** example reports average input/output lengths of **778/73 tokens**. (mlcommons.org) Its documented Server thresholds are TTFT at most **2 seconds** and TPOT at most **100 ms**, while Interactive uses **0.5 seconds** and **30 ms**. (mlcommons.org) (mlcommons.org) These are historical benchmark settings, not recommended enterprise objectives or an assertion about the latest submission rules.

DistServe's motivating experiment uses a **13-billion-parameter model**, **512-token inputs**, **64-token outputs**, and an **80 GB A100**; its colocated example reaches approximately **1.6 requests/s** under a **90% SLO-attainment** criterion. (arxiv.org) The paper also reports that additional batching need not improve GPU efficiency once prefill compute is saturated in its tested configuration. (arxiv.org) Neither result supplies a fleet-sizing denominator for an unrelated model, precision or request mix.

Hardware capacity is a feasibility check before throughput sizing. NVIDIA specifies **141 GB** of high-bandwidth memory (HBM), using HBM3e, for H200. (www.nvidia.com) AMD's MI300X acceptance guide specifies **192 GB HBM3 per accelerator**. (instinct.docs.amd.com) Weight memory alone is insufficient: budget runtime allocations, cache growth and the tested concurrent sequence population. Verify the model's placement group on the proposed system before applying per-GPU arithmetic.

Completion-window worksheet (Hypothetical Example)

Use accepted useful tokens as the work unit, preserving the benchmark's input/output distribution. Let **B** be backlog tokens and **T** be usable processing seconds. Let **q** be measured accepted tokens/s/GPU for that same distribution, including retry and failure overhead inside its elapsed-time boundary. Let **h** be conservative throughput headroom. Let **O** be total non-processing overhead in seconds, covering setup, model loading, warmup and drain outside the measured processing interval. Subtract this same overhead when calculating usable processing time and add it when calculating total completion time. DistServe's goodput definition motivates acceptance-qualified capacity; the following arithmetic is this report's explicit planning model. (arxiv.org)

```
T = deadline_window_seconds - O
required_effective_throughput = B / T
planning_throughput_per_GPU = q * (1 - h)
required_GPUs = ceiling(required_effective_throughput / planning_throughput_per_GPU)
completion_seconds = O + B / (GPUs * planning_throughput_per_GPU)
completion_margin_seconds = deadline_window_seconds - completion_seconds
reserved_GPU_hours = GPUs * reserved_window_hours
incremental_batch_cost = extra_GPU_hours * fully_loaded_GPU_hour_cost
```

The equations assume approximately linear scaling among independently measured replica groups. They are a first sizing estimate, not a substitute for a full-fleet replay. Round to complete placement groups, and account for imbalance among long-document partitions. If measured goodput already includes retries, adding a retry multiplier again double counts them. If using raw attempt throughput instead, explicitly expand work by measured retry attempts and exclude failed outputs from useful completion.

Table 3 applies the worksheet to a **Hypothetical Example**. All workload, throughput, headroom and cost entries are planning assumptions; calculated entries are directly reproducible from them. The separate pool is assumed to use identical two-GPU replicas. The shared scenario measures residual capacity during online acceptance, rather than subtracting utilization from a theoretical peak. (docs.vllm.ai)

Worksheet field	Separate batch pool: assumed or calculated	Shared fleet: assumed or calculated
Useful backlog	300,000,000 tokens, assumed	Same assumed trace
Deadline window / total non-processing overhead	7 hours / 0.5 hours for setup, loading, warmup and drain, assumed	Same assumed window and total overhead
Usable processing time	23,400 seconds, calculated	23,400 seconds, calculated
Accepted useful rate before headroom	1,200 tokens/s/GPU, assumed	400 residual tokens/s/GPU, assumed
Throughput headroom	20%, assumed	20%, assumed
Planning rate	960 tokens/s/GPU, calculated	320 tokens/s/GPU, calculated
Required aggregate throughput	12,820.51 tokens/s, calculated	Same backlog requirement
Required GPUs before placement constraints	14, calculated	41, calculated
Feasible placement / available GPUs	14, seven assumed two-GPU replicas	Only 8 GPUs available, assumed
Forecast completion including total non-processing overhead	6.70 hours, calculated	33.05 hours on 8 GPUs, calculated
Completion margin	0.30 hours, approximately 18 minutes	Negative 26.05 hours
Entire-window GPU reservation	98 GPU-hours	56 GPU-hours available in window
Illustrative fully loaded cost	$3/GPU-hour, assumed	Existing fleet cost recorded separately
Incremental reservation cost	$294 if all 98 GPU-hours are incremental	Feasibility fails before a savings comparison

The dedicated calculation fits the assumed window, but its small remaining margin deserves repeated completion tests and an explicit recovery allowance. Headroom should reflect observed variability, not a habitual percentage. If shared capacity were expanded using the assumed two-GPU placement group, the arithmetic requirement would round from **41 to 42 GPUs**, although their residual throughput would need to be measured again. The worksheet illustrates whole-GPU fleet sizing; it does not establish a performance claim for any accelerator.

A useful-token total is a calibrated accounting unit. For a substantially different input/output ratio, fit separate prefill/decode demand or use request-length buckets with measured service demand. A single total-token denominator cannot safely price both long prompts with short extraction outputs and short prompts with long generation outputs.

Calendar heat map and GPU-hours

A deadline needs capacity in the right hours. The following **Hypothetical Example** is a weekly planning heat map in **Coordinated Universal Time (UTC)**. Each cell is assumed accepted residual capacity expressed as continuously available whole GPUs for that interval, before the worksheet headroom. It is a visualization of supplied planning inputs, not measured availability.

```
UTC interval  Mon Tue Wed Thu Fri Sat Sun
00:00-04:00    8   8   8   8   8  12  12
04:00-08:00    4   4   4   4   4  12  12
08:00-12:00    0   0   0   0   0   8   8
12:00-16:00    0   0   0   0   0   8   8
16:00-20:00    2   2   2   2   2   8   8
20:00-24:00    6   6   6   6   6  12  12
Legend: 0 unavailable; 2-6 limited; 8-12 larger window
```

For a Monday cutoff at **00:00** and deadline at **07:00**, that assumed calendar provides **44 raw GPU-hours**: **8 × 4 + 4 × 3**. It does not provide the **98 reserved GPU-hours** in the separate-pool example. Availability later in the week cannot rescue that deadline. With varying rates, calculate accepted work as the sum of each slot's usable seconds multiplied by measured aggregate residual throughput and its headroom factor; subtract transitions inside the relevant slots.

Slurm notes that accurate job time limits matter to backfill and that reservations use whole nodes. (slurm.schedmd.com) (slurm.schedmd.com) Reflect those allocation boundaries in the calendar. Preserve timezone and daylight-saving rules for operational schedules, and publish the underlying slot data alongside the heat map.

Cost and power boundary

Batch vs real-time LLM inference cost depends on what expenditure changes when batch work runs. Spare capacity may avoid incremental capital purchases while still adding energy, wear, staffing and opportunity cost. Dedicated capacity may add whole nodes, networking, storage and facility commitments. Treat the assumed **$3/GPU-hour** above as a declared scenario input, not a market quote. Depreciation and fixed operations belong in fully loaded allocation; marginal energy belongs in incremental cash cost. Report both views when they lead to different decisions.

NVIDIA labels H200 power as maximum thermal design power, which is not measured whole-server energy consumption. (www.nvidia.com) Meter the node or rack, include relevant supporting equipment, and apply the site's tariff and cooling boundary. EIA's July 2026 data, released **September 24, 2026**, lists U.S. industrial and commercial averages of **9.77 and 14.53 cents/kWh**; those are national sector averages, not a contracted datacenter tariff. (www.eia.gov) (www.eia.gov) EIA labels the values preliminary. (www.eia.gov) Use them only as dated context, not as a substitute for a site-specific power price.

Implementation and Admission Control

A scheduler policy should translate the measured boundary into enforceable controls. Triton supports queue timeout policies that reject or defer requests; those decisions need to be reconciled with the batch completion contract. (docs.nvidia.com) Priority can improve ordering, but starvation, retries and late records must remain visible in the durable job ledger.

The implementation checklist is:

- **Separate identities:** tag interactive and batch traffic through admission, scheduling and telemetry.
- **Protect arrivals:** cap batch admission against the tested online operating region.
- **Bound work:** limit context length, output length and admitted outstanding records.
- **Preserve deadlines:** attach cutoff and completion timestamps to each batch partition.
- **Set priorities:** distinguish server queue order from cluster job preemption.
- **Specify timeouts:** define queue, generation and end-to-end cancellation behavior.
- **Control retries:** use bounded attempts, idempotent writes and a permanent-failure queue.
- **Checkpoint progress:** retain accepted-record identifiers and durable output locations.
- **Drain predictably:** stop admission before reclaiming GPU capacity for online use.
- **Observe jointly:** alert on online tails, acceptance rates and batch completion margin.
- **Publish evidence:** retain commands, configurations, trace statistics and aggregate results.
- **Requalify changes:** repeat acceptance after model, engine, driver or firmware changes.

The batch admission decision should respond to measured latency and deadline margin together. If online conditions tighten, pause admitting new batch records while retaining completed work. If the backlog's forecast exceeds the remaining window, trigger a declared escalation such as another compatible pool or a revised business deadline. Avoid automatically resubmitting unlimited work into the same saturated queue.

Infrastructure requests should match tested placement. In a heterogeneous cluster, node selectors can identify GPU types; in a replicated serving configuration, data parallelism runs separate batches across GPU sets. (kubernetes.io) (docs.vllm.ai) Keep tensor-parallel replica groups intact when rounding capacity; do not confuse them with independently replicated GPU sets. (docs.vllm.ai) A model that requires a linked GPU group cannot consume arbitrarily scattered free devices just because their total count appears sufficient.

Validation should include interruptions, not only steady-state averages. Check durable recovery, batch cancellation, cache release and online return to readiness. AMD's acceptance guidance calls for stepping through validation guides in sequence, reinforcing the distinction between a specification and an accepted installation. (instinct.docs.amd.com) A capacity plan should likewise have a documented acceptance sequence with workload-specific completion and latency gates.

Implications and Future Directions

The central engineering implication is that procurement and scheduling should share an acceptance contract. Purchase decisions need measured replica throughput, memory feasibility and calendar availability; scheduler decisions need the same inputs plus online objectives. Neither a token-price comparison nor a utilization dashboard can independently settle the capacity requirement.

GPU Smith describes its independent position as having no cloud of its own. (gpusmith.com) In this report, that adjacent advisory role belongs in the methodology discussion, not as a competing inference architecture in the comparison table. An objective assessment can recommend reuse, reservation or additional nodes according to measured completion and online acceptance, without presuming that one topology is always preferable.

Future optimization should target the limiting stage identified in the trace. Larger batches are useful only inside the demonstrated latency and memory envelope. Prefill/decode separation, cache policy and alternative precision require new quality and performance acceptance, because changing execution identity changes the evidence. Collect memory and compute telemetry on the target system to identify candidate configurations, then test their completion and online acceptance.

Maintain a lightweight requalification process: preserve the baseline trace, rerun the chosen operating point and its neighboring sweep points, and compare accepted throughput, tail latency, errors and completion margin. Treat improved single-request latency or peak throughput as a hypothesis about fleet savings until the mixed-workload replay demonstrates a smaller feasible reservation. The durable asset is the benchmark contract and its retained evidence, not an unqualified tokens-per-second headline.

Frequently Asked Questions (FAQs)

How many GPUs are needed for LLM inference?

First establish a memory-feasible replica group. Then divide required effective throughput by measured accepted throughput per GPU, apply declared headroom, and round to complete replica groups. (docs.vllm.ai) For shared capacity, use the residual rate measured while online objectives hold. (arxiv.org) The worksheet shows each step; it cannot supply an actual count without a target trace and measured configuration.

Can batch work run in spare real-time capacity?

Yes, when a mixed-load replay demonstrates enough accepted work before the deadline and the online service still meets its objectives. A low utilization reading alone does not establish this. Calendar slots, loading, interruption and drain times belong in the completion calculation; scheduler time limits should reflect that obligation. (slurm.schedmd.com)

Is continuous batching the same as offline batching?

No. Continuous batching is an engine scheduling mechanism; offline batching is a completion contract for a backlog. A real-time service and a batch pipeline may both use continuous batching. (huggingface.co) Their acceptance tests and capacity tables should remain separate.

How should GPU-hours and batch savings be calculated?

Calculate both processing occupancy and reserved GPU-hours. Price extra reserved capacity at the applicable fully loaded rate, then separately show incremental cash expenditure and opportunity cost. Public API batch discounts cannot replace that calculation because API commercial terms do not establish the cost of operating a private fleet.

Conclusion

Batch and real-time inference should be compared through their obligations: completing a backlog by a deadline and serving arriving requests within latency and availability objectives. Engine batching is compatible with either obligation. (huggingface.co) The architecture decision is where to place work, how to admit it and which measured capacity can be committed to it.

The recommended sequence is to define the trace and completion boundary, verify memory-feasible placement, benchmark accepted throughput, and test online interference. Translate that evidence into usable processing seconds and complete replica groups. Retain the rejected sweep points so the acceptable operating region is visible, matching the practice of evaluating a performance curve rather than only its peak. (docs.mlcommons.org) For shared capacity, sum residual work over the actual calendar rather than extrapolating from average utilization.

A separate pool is justified when its incremental reservation is needed to meet the deadline or maintain a clear acceptance boundary. Resource-admission accounting should remain distinct from physical usage. (docs.ray.io) A nightly window is useful when its availability, transitions and return-to-service behavior are dependable. Spare-capacity reuse is credible when the mixed-load test establishes enough accepted work while the online contract holds. None has a universal cost advantage.

Finally, present assumptions separately from observations. The worksheet and heat map in this report explain the arithmetic; they are not procurement benchmarks. A reviewable fleet-sizing recommendation should include the exact execution identity, representative trace, raw aggregate results, completion margin, online tail measurements and cost boundary. Changes in release or platform require renewed evidence. (docs.pytorch.org) That evidence makes the decision reproducible and gives the platform team a concrete basis for retesting after its workload or software changes.

Source links

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

1. https://huggingface.co/docs/transformers/continuous_batching
2. https://docs.mlcommons.org/endpoints-submission-docs/understand/what-is-mlperf-endpoints/
3. https://arxiv.org/html/2401.09670v3
4. https://docs.vllm.ai/en/stable/cli/bench/serve/
5. https://slurm.schedmd.com/sched_config.html
6. https://slurm.schedmd.com/gres.html
7. https://gpusmith.com/
8. https://docs.ray.io/en/latest/data/batch_inference.html
9. https://docs.vllm.ai/en/stable/cli/bench/
10. https://docs.nvidia.com/deeplearning/triton-inference-server/user-guide/docs/user_guide/batcher.html
11. https://docs.vllm.ai/en/stable/api/vllm/config/scheduler/
12. https://arxiv.org/abs/2309.06180

13. https://huggingface.co/docs/text-generation-inference/main/en/conceptual/paged_attention
14. https://prometheus.io/docs/practices/histograms/
15. https://prometheus.io/docs/concepts/metric_types/
16. https://kubernetes.io/docs/tasks/manage-gpus/scheduling-gpus/
17. https://docs.ray.io/en/latest/ray-core/scheduling/resources.html
18. https://kubernetes.io/docs/concepts/scheduling-eviction/pod-priority-preemption/
19. https://slurm.schedmd.com/preempt.html
20. https://docs.pytorch.org/docs/2.14/notes/randomness.html
21. https://docs.pytorch.org/docs/2.14/benchmark_utils.html
22. https://docs.nvidia.com/deeplearning/triton-inference-server/user-guide/docs/user_guide/model_analyzer.html
23. https://docs.nvidia.com/deeplearning/triton-inference-server/user-guide/docs/model_analyzer/docs/config.html
24. https://docs.nvidia.com/deeplearning/triton-inference-server/user-guide/docs/model_analyzer/docs/metrics.html
25. https://docs.vllm.ai/en/stable/configuration/optimization/
26. https://gpusmith.com/articles/en/vllm-kv-cache-capacity-planner
27. https://mlcommons.org/2025/09/small-llm-inference-5-1/
28. https://www.nvidia.com/en-us/data-center/h200/
29. https://instinct.docs.amd.com/projects/system-acceptance/en/latest/gpus/mi300x.html
30. https://gpusmith.com/articles/en/gpu-cluster-energy-calculator
31. https://www.eia.gov/electricity/monthly/epm_table_grapher.php?t=epmt_es1a
32. https://gpusmith.com/articles/en/disaggregated-prefill-decode-break-even
33. https://gpusmith.com/articles/en/llm-inference-hardware-sizing-guide

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.