GPU SMITH
COMPUTE ENGINEERING

GPU SMITH / RESEARCH & PRACTICAL GUIDANCE

KV Cache Memory Calculator: Context, Concurrency and GPUs

> 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 · kv cache memory calculator · kv cache size calculator · kv cache memory formula · vllm · tensorrt-llm · grouped query attention · llm inference · gpu sizing · tensor parallelism

Original article: https://gpusmith.com/articles/en/kv-cache-memory-calculator-gpu-count

In this article

1. Executive Summary
2. Introduction and Background
3. Architecture and the KV Cache Memory Formula
4. Decision Contract and Available HBM
5. Traffic Distributions and Concurrent Cached Tokens
6. Runtime Reconciliation and SLO Validation
7. Cache Optimization, Replicas and GPU Economics
8. Data Analysis and Evidence
9. Implications and Future Directions
10. Conclusion

Executive Summary

A **KV cache memory calculator** should produce a conditional deployment plan: bytes per cached token, simultaneous resident tokens, validated capacity per serving replica, and the resulting graphics processing unit (GPU) count. The key-value (KV) cache retains attention states from previous tokens so generation can reuse them. (huggingface.co) For standard full attention, the theoretical formula is **layers × KV heads × head dimension × 2 × bytes per element**, because each attention layer retains both keys and values. (huggingface.co) Grouped-query attention (GQA) shares KV heads across query heads; multi-query attention (MQA) uses a single KV head. The calculator must read KV-head count rather than substitute query-head count. (arxiv.org)

A labeled hypothetical architecture with **32 layers, 8 KV heads, head dimension 128, and two-byte cache elements** requires **131,072 bytes, or 128 KiB, per cached token**, calculated from that formula. (huggingface.co) Its illustrative workload of **100 simultaneous sequences** split among short, medium and long contexts totals **696,320 cached tokens and 85 GiB**, compared with **400 GiB** if all sequences are assigned the example maximum length. These are reproducible scenario calculations, not model benchmarks. The public Azure trace input check below separately uses actual prompt/output records; its schema lacks completion times, so it cannot establish concurrent cache occupancy. (github.com)

The second stage replaces theoretical capacity with engine evidence. **vLLM** exposes group-aware `kv_cache_size_tokens` and per-data-parallel-engine `kv_cache_max_concurrency`; the latter describes full-length capacity, not a latency service-level objective (SLO). (docs.vllm.ai) (docs.vllm.ai) **TensorRT-LLM** allocates cache subject to token limits and a fraction of free memory, while **SGLang** distinguishes the static weights-and-cache allocation from its token-pool limit. (nvidia.github.io) (docs.sglang.io) Dividing cache memory by tensor-parallel GPU count is conditional: documented layouts can duplicate KV states. (docs.vllm.ai)

The defensible GPU-count calculation is **replicas = ceiling(peak simultaneous cached tokens / validated usable tokens per replica)**, followed by latency validation and multiplication by GPUs per replica. Treat prefix reuse, cache quantization and offload as experiments. (www.lmsys.org) Transformers warns that quantized caches can increase latency for short contexts; central processing unit (CPU) offload also introduces transfers.

(huggingface.co) (nvidia.github.io) Keep **p50, p95 and p99** prompt/output distributions separate from percentiles of simultaneous cache occupancy, and test time to first token (TTFT), inter-token latency (ITL), throughput and queueing together. ITL measures individual token intervals. (docs.cloud.google.com) Physical high-bandwidth memory (HBM) fit establishes feasibility; only the workload test establishes the operating SLO. (mlcommons.org)

Introduction and Background

Private large language model (LLM) deployments often begin with a model-weight estimate and a GPU memory specification. That estimate becomes incomplete when architects choose a context policy and admit overlapping requests. The variable KV term grows with retained attention states rather than registered users. KV caching avoids recomputing earlier keys and values during autoregressive generation. (huggingface.co) The practical sizing question is therefore how much state remains resident while the scheduler is serving the actual workload.

This report addresses architecture-aware preflight, runtime reconciliation and deployment validation as separate stages. **Theoretical** figures describe tensor payload under declared assumptions. **Runtime-reported** figures describe the engine's configured allocation or token capacity. **Measured** figures require a load test with arrival timing, completions and latency observations. No inference service was supplied for this report, so the worked capacity scenarios are calculations. The real public-trace test is restricted to input distributions, with its missing concurrency information stated explicitly.

GPU Smith's published scope connects topology and serving-stack sizing to workload models and throughput targets. (gpusmith.com) That adjacent engineering perspective motivates a decision contract rather than a universal GPUs-per-model answer. The contract should make it possible for another architect to reproduce the estimate, explain why the engine disagrees, and decide whether adding replicas changes the limiting resource.

The documentation and public sources were accessed for the **October 2026** publication snapshot. Versioned vLLM material is identified where used; mutable `latest` documentation is not evidence of an installed release. The discussion covers conventional attention, shared-head attention, compressed and windowed caches, parallelism, allocation, traffic and latency. It also treats a capacity increase as a hypothesis about economics: more cached tokens do not necessarily translate into proportionally more completed requests under a fixed response-time requirement.

Architecture and the KV Cache Memory Formula

Count stored heads and layer-specific states

For conventional homogeneous full attention, define `L` as attention layers, `Hkv` as stored KV heads per layer, `D` as head dimension and `s` as bytes per cache element. The derivation follows the separate key/value tensors retained at each layer. (huggingface.co)

```
bytes_per_token = L * Hkv * D * 2 * s
required_KV_bytes = sum(active_sequence_cached_tokens) * bytes_per_token
```

For window-limited conventional layers, a preflight estimate can instead sum `2 * Hkv_l * D_l * s_l * min(tokens, window_l)` across those layers, adding full-attention and fixed-state components separately. Treat that expression as a layout-dependent bound until the engine confirms its allocation.

These equations describe **raw tensor payload**. They exclude allocator metadata, quantization scales, page padding, workspace and retained reusable blocks. Record those separately during reconciliation. For heterogeneous full-attention layers, sum each layer's own key and value dimensions rather than multiply by a uniform layer count.

Multi-head attention (MHA) uses a separate KV head for each query head; GQA uses an intermediate KV-head count and MQA uses one. (arxiv.org) (arxiv.org) For the hypothetical architecture used here, substituting **32**, **8** or **1** stored KV heads yields **512**, **128** or **16 KiB/token**, respectively, with all other inputs held constant. These are arithmetic comparisons of declared inputs, not quality-equivalent model substitutions. (arxiv.org)

Read `num_hidden_layers`, `num_attention_heads`, `num_key_value_heads`, and an explicit `head_dim` from the pinned model configuration. Transformers documents the layer field as decoder-layer count. (huggingface.co) Its Llama configuration defaults head dimension to `hidden_size // num_attention_heads` when absent, but that documented default should not be applied blindly to every architecture. (huggingface.co) Check the implementation if field names differ, if a multimodal model nests its text configuration, or if keys and values have unequal dimensions.

Detect exceptions before calculating

Sliding-window attention (SWA) limits retained history for relevant layers. Transformers documents that sliding-window and chunked-attention layer caches stop growing at their window or chunk size. (huggingface.co) Original Mistral documentation describes a **4,096-state window** per layer and rotating buffers. (mistral.ai) (mistral.ai) A full-context calculation for that layout can therefore be a conservative comparison, but must not be presented as its implemented steady-state cache requirement.

Multi-head latent attention (MLA) also needs a different model. DeepSeek-V2 compresses the cache into a latent vector; its reported reduction is a comparison with a particular predecessor, not a multiplier that applies to every MLA implementation. (arxiv.org) State-space components require separate treatment: Transformers notes that Mamba has neither attention nor KV states. (huggingface.co) A hybrid model may combine multiple growing and fixed-size state pools.

Tensor parallelism (TP) must follow the engine's physical layout. The versioned **vLLM 0.31.0** guide documents KV-head sharding and duplication when TP exceeds available KV heads. (docs.vllm.ai) TensorRT-LLM also documents replication exceptions for shared-head and latent-attention models. (nvidia.github.io) The safe calculation uses observed local heads or latent dimensions on each rank, with the most constrained rank determining capacity. Do not automatically divide every architecture's bytes per token by the number of GPUs.

Decision Contract and Available HBM

Freeze the inputs that change capacity

A calculator result needs an identity stronger than the model's display name. Preserve the configuration, tokenizer, engine build and launch arguments. An architecture change or a different attention backend can change which cache estimate is valid; vLLM validates explicitly selected backends against the configuration. (docs.vllm.ai)

Table 1 maps the decision contract to a compact spreadsheet. Cells are example locations, not observed deployment values. The field definitions come from model configuration and engine allocation references. (huggingface.co) (docs.vllm.ai)

Input or result	Spreadsheet cell or formula	Verification requirement
Model identity	Text fields above the sheet	Repository revision, config hash, tokenizer revision and attention types.
Attention layers	B2 = L	Count participating attention layers, not every stateful component.
Stored KV heads	B3 = Hkv	Read `num_key_value_heads`; verify rank-local layout.
Head dimension	B4 = D	Prefer explicit configuration and implementation. (huggingface.co)
Cache bytes per element	B5 = s	Record cache dtype separately from weight dtype. (docs.sglang.io)
Raw bytes per token	B6 = 2*B2*B3*B4*B5	Valid for the declared homogeneous full-attention layout.
Available KV bytes	B7 = runtime allocation in bytes	Measure after weight load and execution-buffer reservations.
Runtime capacity	B8 = reported tokens per replica	Preserve group-aware semantics and per-rank limits. (docs.vllm.ai)
Policy headroom	B9 = fraction withheld	Explicit planning assumption beyond runtime reservations.
Usable capacity	B10 = FLOOR(B8*(1-B9),1)	Use only after capacity reconciliation and latency validation.

The spreadsheet separates configuration evidence from planning policy. It also avoids pretending that the headroom entry explains every difference between payload and engine capacity. A hybrid architecture should replace the raw-payload row with a verified group-specific calculation, while preserving the runtime-capacity and traffic rows.

Use this extraction checklist before accepting the result:

- **Configuration:** Save the exact `config.json`, repository revision and hash.
- **Dimensions:** Record layer count, query heads, KV heads and explicit head dimensions.
- **Attention taxonomy:** Identify full, sliding, chunked, latent and state-space components.
- **Context policy:** Record admitted prompt-plus-output limits and any reserved generation budget.
- **Parallel layout:** Record TP, pipeline layout, data parallelism (DP), and rank-local cache shapes.
- **Software:** Save installed vLLM or TensorRT-LLM version, build commit, container digest and backend.

- **Cache policy:** Save dtype, block/page size, prefix reuse, offload and eviction configuration.
- **Evidence labels:** Keep calculated, engine-reported and measured cells visibly distinct.

Model maximum context is another input, not a concurrency assumption. Transformers defines `max_position_embeddings` as the model's maximum sequence length; architecture-specific extensions require their own verification. (huggingface.co) Its `pretraining_tp` field refers to pretraining, so it is not the deployed inference TP setting. (huggingface.co)

Subtract the memory that cannot store cache

Use **available KV bytes = measured allocatable HBM after weights and runtime buffers**. Reserve execution workspaces, communication buffers, graph-related allocations and other actual runtime consumers before deriving capacity. PyTorch's `memory_allocated` reports tensor-occupied bytes, while `memory_reserved` reports bytes managed by its caching allocator. These counters answer different questions. (pytorch.org) (pytorch.org)

Likewise, advertised memory is a starting constraint. NVIDIA lists **H200 at 141 GB HBM3e**. (www.nvidia.com) AMD lists **MI300X at 192 GB HBM3**. (www.amd.com) Neither specification is an allocatable KV budget. Keep decimal vendor units and binary calculator units explicit, and reconcile using actual byte counts rather than treating GB and GiB as interchangeable.

Traffic Distributions and Concurrent Cached Tokens

Model the resident state over time

Concurrency means requests in flight, not employees with accounts. MLCommons describes it as simultaneous queries. (mlcommons.org) For ordinary full attention without shared prefixes, sample `T(t) = sum_i cached_tokens_i(t)` over active sequences. Prompt tokens enter the cache during prefill; generated tokens enlarge it during decode. A conservative admission policy can additionally reserve the remaining output budget, but that reservation is a policy bound rather than already resident state.

Keep **prompt/output distributions** and **occupancy distributions** separate. Collect paired prompt/output lengths so their relationship is preserved. Then combine them with arrivals, queue entry, prefill completion, decode progress, cancellation and completion events. Request-length percentiles alone cannot describe the number of simultaneous cached tokens. A long request that overlaps many others may matter more than several long requests that occur sequentially.

Track at least these parallel views:

- **Prompt p50/p95/p99:** Include retrieved documents, tool messages and carried conversation history.
- **Output p50/p95/p99:** Separate requested output limits from realized generation.
- **Length pairs:** Preserve which output belongs to which prompt rather than combine independent marginals.
- **Arrival bursts:** Retain timing and tenant clustering instead of smoothing all demand into a daily average.
- **Active occupancy:** Sample resident tokens and unique physical blocks at the engine.

- **Queue occupancy:** Track waiting requests separately from allocated cache.
- **Routing:** Preserve replica assignment, prefix affinity and imbalance.
- **Tail exposure:** Record the windows and sample counts behind occupancy and latency percentiles.
- **Prefix residency:** Distinguish active-request state from reusable blocks retained after completion.

The **95th percentile** is the 0.95 quantile, but a percentile remains specific to its measured variable. (prometheus.io) Taking the p95 prompt length times the p95 active count does not yield a justified p95 occupancy estimate without their joint timing. Use the observed time series or a replay that retains the dependence between requests. Also report exceedance frequency and burst duration: a brief capacity breach and a sustained queue can require different operating policies.

Prometheus warns that averaging per-instance quantiles is statistically nonsensical. (prometheus.io) Aggregate appropriate histogram observations before calculating fleet percentiles, with buckets chosen around the operating threshold; quantile error depends on bucket width. (prometheus.io) (prometheus.io) For memory planning, keep per-replica occupancy visible even when a fleet summary looks acceptable. Aggregate free memory cannot repair an overloaded individual replica without a valid routing or migration mechanism.

Replay admission, residency and block rounding

For an unshared conventional cache with block size `P`, the payload-bound estimate becomes `bytes_per_token * sum(P * ceil(tokens_i/P))`. TensorRT-LLM defines a block as the smallest KV allocation unit. (nvidia.github.io) In a block-rounding calculation (Hypothetical Example), a chosen **16-token** block policy means a **4,097-token** sequence rounds to **4,112 tokens**, leaving **15 token slots** unused. This is a calculation under the selected block size, not a documented universal engine default. (nvidia.github.io)

Prefix sharing changes physical occupancy: count shared blocks once while recording private suffixes separately. The original RadixAttention explanation describes common-prefix cache sharing. (www.lmsys.org) Retained reusable blocks can occupy memory after requests finish, so active logical token totals and total physical pool occupancy need separate gauges. Eviction and admission determine whether reusable state is displaced when demand increases; original RadixAttention combines reuse with a least-recently-used policy. (www.lmsys.org)

A production trace replay should yield peak and percentile physical occupancy per replica, queued demand, and completion latency under the same routing and admission policy intended for deployment. When those inputs are absent, a histogram is useful sensitivity analysis, but it is not validated concurrent capacity.

Runtime Reconciliation and SLO Validation

Replace payload capacity with engine capacity

For vLLM, retain the startup report and exposed cache configuration. `kv_cache_size_tokens` is **group-aware and per DP engine**; simple `num_gpu_blocks * block_size` can misstate hybrid capacity because a request occupies multiple cache groups. (docs.vllm.ai) `kv_cache_max_concurrency` describes capacity at

`max_model_len`, not how many concurrent requests meet a latency target. (docs.vllm.ai) Explicit `kv_cache_memory_bytes` is a per-GPU cache allocation and overrides the memory-utilization-based inference path. (docs.vllm.ai)

TensorRT-LLM uses `free_gpu_memory_fraction` and `max_tokens` as cache constraints; the fetched reference documents a **0.9 default fraction**, applied to free memory rather than the entire device specification. (nvidia.github.io) Compatible layer configurations occupy cache pools, so heterogeneous layer groups require pool-aware reconciliation. (nvidia.github.io) SGLang similarly documents `mem-fraction-static` for weights plus its KV pool and `max-total-tokens` as the pool token limit. (docs.sglang.io)

Table 2 provides the acceptance record. It compares three kinds of evidence without inserting invented deployment results.

Check	Preflight evidence	Runtime or measured acceptance evidence
Layout	Config and rank-local bytes/token	Verify actual heads, latent states and cache pools. (nvidia.github.io)
Allocation	Weights, runtime buffers and remaining HBM	Record tensor and allocator counters separately. (pytorch.org)
Token capacity	Payload-derived upper bound	Save vLLM group-aware token capacity or equivalent engine pool totals. (docs.vllm.ai)
Concurrent demand	Traffic lengths and replay assumptions	Measure per-replica resident blocks, queued requests and admission decisions.
Prefix reuse	Exact reusable-prefix assumptions	Calculate interval token hits/queries and record cold runs. (docs.vllm.ai)
Latency	Project-defined response contract	Measure TTFT, individual ITL, end-to-end latency and completion throughput. (docs.ray.io) (docs.cloud.google.com)
Scaling	GPU count per replica and redundancy policy	Test startup delay and burst queueing, including a replica unavailable. (docs.ray.io)

Interpret disagreement rather than hiding it inside a generic overhead factor. A payload estimate larger than the engine report may reflect rank imbalance, duplicate cache, group layout, padding or reservations. An unexpectedly larger report may reflect compressed states, window-limited storage or changed dtype. Trace the difference to configuration and measured allocations before using it to approve procurement.

Test the same response contract at increasing load

TTFT is request-arrival-to-first-streamed-token latency, including queueing and prefill effects. (docs.ray.io) **Time per output token (TPOT)** is an average during decode, whereas ITL represents individual token intervals and can expose stalls hidden by the average. (docs.ray.io) (docs.cloud.google.com) Capture end-to-end completion latency and failures alongside both metrics. (docs.cloud.google.com)

A compact load-test protocol is:

- **Baseline:** Warm the engine and retain a separate cold-cache run.

- **Arrival process:** Replay realistic timing and an explicitly defined burst stress test.
- **Length classes:** Include short, medium, long and near-policy-limit requests.
- **Concurrency sweep:** Increase offered load until queueing, capacity or the response contract limits operation.
- **Measurements:** Record per-replica capacity, resident blocks, preemptions, prefix counters and completion counts.
- **Acceptance:** Require the chosen latency percentiles, error budget and throughput simultaneously.
- **Repeatability:** Preserve engine build, launch arguments, trace version and measurement window.

TensorRT-LLM's Prometheus example includes request, queue, cache and latency metrics. (nvidia.github.io) Benchmark latency contracts illustrate why fit alone is insufficient: MLCommons publishes **99th-percentile TTFT ≤3 seconds and TPOT ≤80 milliseconds** for one GPT-OSS server scenario, versus **≤2 seconds and ≤15 milliseconds** for its interactive scenario. Those are benchmark-specific constraints, not enterprise recommendations or results for this report's examples. (mlcommons.org) (mlcommons.org)

Cache Optimization, Replicas and GPU Economics

Change one cache policy at a time

Cache precision and weight precision are separate decisions. vLLM documents per-tensor and per-attention-head floating-point 8-bit (FP8) KV strategies, with backend and calibration conditions. (docs.vllm.ai) Its Flash Attention 3 path also quantizes queries when using FP8 KV, making accuracy and kernel behavior part of the experiment. (docs.vllm.ai) TensorRT-LLM exposes FP8 KV configuration and separately documents offline ModelOpt preparation for NVIDIA floating-point 4-bit (NVFP4) KV cache. (nvidia.github.io) (nvidia.github.io)

For a precision-change calculation (Hypothetical Example), moving from two-byte to one-byte cache payload means the standard tensor term halves. That arithmetic excludes scale metadata, alignment and runtime buffers; it does not establish a doubled usable capacity or throughput. (huggingface.co) Transformers specifically warns that quantized-cache latency can worsen at short context lengths. (huggingface.co) Validate task accuracy, TTFT and token-interval tails against the unchanged workload before treating the memory saving as economically usable.

Prefix reuse also needs the correct denominator. vLLM's token-based hit accounting counts queried tokens present in cache, not requests that fully hit. (docs.vllm.ai) TensorRT-LLM documents cross-request block reuse, but traffic and routing determine the realized benefit. (nvidia.github.io) Test warm, cold and eviction-heavy conditions. Keep any tenant isolation and cache-key requirements from the deployed engine in the acceptance contract rather than assuming all identical-looking prompts are shareable.

Offload provides a slower storage tier. vLLM documents pinned CPU memory for completed blocks and a CPU primary tier through which secondary transfers stage. (docs.vllm.ai) (docs.vllm.ai) Its host-byte reservation is shared across workers, so do not multiply that setting blindly by worker count. (docs.vllm.ai) AMD notes that pinned allocations reduce host random-access memory (RAM) available to other processes and that device access over the host interconnect is slower than on-device access. (rocmdocs.amd.com) (rocmdocs.amd.com)

Reserve system RAM, inspect non-uniform memory access (NUMA) placement and test transfer saturation. Transformers' layer-offloading strategy illustrates that offload implementations can move different units of state. (huggingface.co)

Convert capacity to replicas, then apply latency and availability

For equivalent replicas and a declared occupancy target, calculate `R_memory = ceil(T_target / C_usable)`. Then test whether that replica count satisfies latency and routing constraints. If it does not, increase replicas or change the architecture/policy and repeat. **GPU count = replicas × GPUs per replica** only after the per-replica layout is fixed. For independently deployed complete serving replicas, multiplying replicas also multiplies their full-model weight allocations. In vLLM mixture-of-experts (MoE) deployments combining attention DP with expert or tensor parallelism across ranks, the DP ranks are not fully independent, and expert weights can be sharded across ranks. (docs.vllm.ai) (docs.vllm.ai) Treat these coupled ranks as a shared deployment when planning memory, latency and availability; do not automatically equate their DP count with a count of independent full-model replicas. TensorRT-LLM attention-DP routing likewise assigns different requests to different ranks under its documented strategy. (nvidia.github.io)

Treat the ceiling calculation as a lower bound when demand cannot be perfectly balanced. A request cannot generally be divided across ordinary independent replicas merely because aggregate token space is available. Include hot-prefix affinity, tenant isolation, heterogeneous request sizes and maintenance capacity in routing tests.

Autoscaling policy is part of the capacity calculation. KServe distinguishes a soft concurrency target from a hard `containerConcurrency` limit that buffers excess requests. (kserve.github.io) (kserve.github.io) Ray identifies replica startup time as a determinant of burst response. (docs.ray.io) A theoretical future replica does not absorb requests while it is still loading. (docs.ray.io)

For economics, use reader-supplied prices: `cost_per_hour = R * cost_per_replica_hour`, then divide by measured successful requests or useful output tokens under the SLO. For owned hardware, include amortization, power, facilities, networking and support in that replica cost. These equations are accounting definitions, not claims about current rental rates. Offload and quantization are worth testing when their validated reduction in required replicas exceeds added infrastructure cost without degrading the response contract.

Data Analysis and Evidence

Capacity sensitivity (Hypothetical Example)

Table 3 applies the declared **128 KiB/token** full-attention architecture to a hypothetical simultaneous workload. All values are calculated from the layerwise tensor formula, not observed GPU results. (huggingface.co)

Active workload class	Sequences	Cached tokens each	Total cached tokens	Calculated KV payload
Short	60	2,048	122,880	15 GiB
Medium	30	8,192	245,760	30 GiB

Active workload class	Sequences	Cached tokens each	Total cached tokens	Calculated KV payload
Long	10	32,768	327,680	40 GiB
Total	100	Mixed	696,320	**85 GiB**

The sum is the relevant input because the classes are explicitly simultaneous. Giving all these requests the illustrative maximum length produces **3,276,800 tokens and 400 GiB**, while the mixed snapshot averages **6,963.2 tokens** per request. These arithmetic results demonstrate the difference between a declared worst-case admission bound and an observed or specified occupancy mix. They do not prove that future bursts retain the same mix. (huggingface.co)

With an assumed runtime cache allocation of **12 GiB per equivalent single-GPU replica**, the payload calculation gives **98,304 tokens** before additional planning headroom. With headroom of **0%, 10%, 20% and 30%**, floored usable capacities are **98,304; 88,473; 78,643; and 68,812 tokens**, yielding **8, 8, 9 and 11 replicas** for the example demand. These are sensitivity outputs for hypothetical inputs; validated runtime tokens must replace the assumed allocation before deployment. (huggingface.co)

The following calculator reproduces the scenario and incorporates per-sequence page rounding. The page size is an input; SGLang defines it as tokens per page. (docs.sglang.io)

```
from math import ceil, floor
L, Hkv, D, element_bytes = 32, 8, 128, 2  # hypothetical full attention
bytes_per_token = 2 * L * Hkv * D * element_bytes
classes = [(60, 2048), (30, 8192), (10, 32768)]
page_tokens = 16  # chosen example, not an engine default
demand = sum(n * page_tokens * ceil(t / page_tokens) for n, t in classes)
assumed_kv_bytes = 12 * (2 ** 30)
preflight_tokens = assumed_kv_bytes // bytes_per_token
capacity_tokens = preflight_tokens  # replace with reconciled engine capacity
for headroom in [0.0, 0.1, 0.2, 0.3]:
    usable = floor(capacity_tokens * (1 - headroom))
    replicas = ceil(demand / usable)
    print(headroom, usable, replicas)
```

This calculator intentionally covers homogeneous full attention and an unshared payload. It does not implement MLA, hybrid cache groups, prefix deduplication or engine allocation. Its useful output is a reproducible preflight comparison and a clearly defined place to insert reconciled capacity.

Real-trace input check and its limits

The report's arithmetic test used the **first ten rows** of Azure's public coding inference CSV, whose first sampled timestamp is **2023-11-16 18:17:03.9799600**. (github.com) The prompt/output pairs were **(4808,10), (3180,8), (110,27), (7433,14), (34,12), (374,14), (6985,9), (34,23), (1145,7), (201,24)**, copied from the fetched dataset. (github.com)

Local calculations produced **24,304 prompt tokens**, **148 output tokens**, and **24,452 combined tokens**. With nearest-rank quantiles, prompt **p50/p95/p99 = 374/7433/7433**, and output **p50/p95/p99 = 12/27/27**. (github.com) This tiny contiguous sample is an input-format and arithmetic check; its upper percentiles equal the sample maximum and should not be used as production tail estimates. The source documents invocation and token lengths but not completion timing, so neither actual overlap nor per-replica cache occupancy can be reconstructed. (github.com)

Larger original research reinforces why workload type matters: Splitwise reports median coding prompts of **1,500 tokens** and conversation outputs of **129 tokens**. These describe its studied traces, not this ten-row sample or a universal private-LLM workload. (www.microsoft.com) (www.microsoft.com) LMCache's FAQ also gives a configuration-specific capacity example of **275,760 cache tokens and 6.73 full-length 40,960-token prompts**. Treat it as a documented example, not a benchmark prediction for a different model or engine. (docs.lmcache.ai)

Implications and Future Directions

A useful procurement record joins the architecture spreadsheet to startup evidence and the load-test acceptance record. It should explain what changes when an architect raises the context limit, enables a different cache dtype or shifts request routing. GPU Smith's published method describes acceptance testing against written criteria; the corresponding criterion here is validated capacity under the selected traffic and response contract. (gpusmith.com) That perspective belongs alongside the sizing decision, without treating an engineering advisor as a cache implementation.

Near-term optimization should focus on the measured limiting term. If cache allocation limits admission while latency remains acceptable, test quantization or prefix-aware routing. If latency limits operation before the pool fills, extra nominal cache capacity may leave replica count unchanged. Requests per second is especially sensitive to context mix, as Google Cloud's inference guidance notes. (docs.cloud.google.com) Compare configurations using the same replay and successful completion definition.

Architecture and parallelism will continue to complicate a single bytes-per-token answer. DeepSeek-V2's authors reported a **93.3% cache reduction** in their specific model comparison, illustrating the need to identify the stored representation rather than extrapolate conventional KV-head arithmetic. (arxiv.org) The versioned vLLM guide documents decode context parallelism as a way to reduce duplication, with communication tradeoffs that require separate performance testing. (docs.vllm.ai)

Refresh the result whenever these recorded conditions change:

- **Model revision:** Recheck configuration, attention groups and maximum-context behavior.
- **Engine revision:** Recheck supported dtype, allocator semantics and capacity fields.
- **Parallelism:** Recheck local cache shapes and the most constrained rank.
- **Traffic:** Recompute paired lengths, occupancy tails, burst duration and routing skew.
- **Reuse or offload:** Repeat cold, warm, eviction and transfer-heavy runs.
- **SLO:** Re-establish the highest sustainable offered load under the new contract.

Retain the percentile definition as carefully as the numeric result. MLCommons' Endpoints page specifies a **90th-percentile TTFT** measure, while other benchmark scenarios use different contracts. (mlcommons.org) A p95 or p99 enterprise objective must be explicitly chosen and measured; a statistic from another protocol cannot silently replace it.

Conclusion

A defensible KV-cache memory estimate begins with the stored attention representation and ends with a workload-specific deployment test. Count layers, KV heads, head dimensions and cache element size for a conventional full-attention preflight. Check windowed, latent and hybrid architectures before applying that formula, and verify the physical cache layout on each tensor-parallel rank.

Separate the weights and execution allocations from the cache budget. Reconcile the resulting payload estimate with engine-reported capacity, retaining group-aware semantics and the configured software identity. A capacity field answers a storage question. It does not establish how quickly requests are admitted, how smoothly tokens stream, or how the queue behaves during a burst.

Use paired prompt/output lengths and arrival timing to construct simultaneous occupancy. The hypothetical histogram demonstrates how a declared workload mix changes the estimate; the real public-trace sample validates input arithmetic but cannot establish concurrency because completion information is absent. The next deployment step is an instrumented replay with resident-state, completion and latency telemetry.

Finally, translate validated usable capacity into replica count, apply routing and availability requirements, and multiply by GPUs per replica. Compare quantization, prefix reuse and offload by whether they reduce the required infrastructure while meeting the same response contract. Preserve the spreadsheet, runtime report and load-test record together. That evidence makes the sizing decision reproducible and identifies when a model, engine or workload change requires it to be recalculated.

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/en/cache_explanation
2. https://arxiv.org/abs/2305.13245
3. https://github.com/Azure/AzurePublicDataset/blob/master/AzureLLMInferenceDataset2023.md
4. https://docs.vllm.ai/en/latest/api/vllm/config/cache/
5. https://nvidia.github.io/TensorRT-LLM/features/kvcache.html
6. https://docs.sglang.io/docs/advanced_features/server_arguments
7. https://docs.vllm.ai/en/v0.31.0/serving/context_parallel_deployment/
8. https://www.lmsys.org/blog/2024-01-17-sglang/
9. https://huggingface.co/docs/transformers/en/kv_cache
10. https://docs.cloud.google.com/kubernetes-engine/docs/concepts/machine-learning/inference
11. https://mlcommons.org/2026/03/mlperf-inference-gpt-oss/

12. https://gpusmith.com/articles/en/llm-inference-hardware-sizing-guide
13. https://gpusmith.com/
14. https://huggingface.co/docs/transformers/en/model_doc/llama
15. https://mistral.ai/news/announcing-mistral-7b/
16. https://arxiv.org/abs/2405.04434
17. https://nvidia.github.io/TensorRT-LLM/features/parallel-strategy.html
18. https://docs.vllm.ai/en/latest/design/attention_backends/
19. https://pytorch.org/docs/stable/generated/torch.cuda.memory.memory_allocated.html
20. https://pytorch.org/docs/stable/generated/torch.cuda.memory.memory_reserved.html
21. https://www.nvidia.com/en-us/data-center/h200/
22. https://www.amd.com/en/newsroom/press-releases/2023-12-6-amd-delivers-leadership-portfolio-of-data-center-a.html
23. https://mlcommons.org/benchmarks/endpoints/
24. https://prometheus.io/docs/practices/histograms/
25. https://nvidia.github.io/TensorRT-LLM/advanced/kv-cache-management.html
26. https://gpusmith.com/articles/en/vllm-kv-cache-capacity-planner
27. https://docs.vllm.ai/en/latest/design/metrics/
28. https://docs.ray.io/en/latest/serve/llm/benchmarks.html
29. https://nvidia.github.io/TensorRT-LLM/examples/prometheus_metrics.html
30. https://docs.vllm.ai/en/latest/features/quantization/quantized_kvcache/
31. https://nvidia.github.io/TensorRT-LLM/features/quantization.html
32. https://docs.vllm.ai/en/latest/features/kv_offloading_usage/
33. https://rocmdocs.amd.com/projects/HIP/en/latest/how-to/hip_runtime_api/memory_management/host_memory.html
34. https://docs.vllm.ai/en/latest/serving/data_parallel_deployment/
35. https://docs.vllm.ai/en/latest/serving/expert_parallel_deployment/
36. https://kserve.github.io/website/docs/model-serving/predictive-inference/autoscaling/kpa-autoscaler
37. https://github.com/Azure/AzurePublicDataset/raw/refs/heads/master/data/AzureLLMInferenceTrace_code.csv
38. https://www.microsoft.com/en-us/research/uploads/prodnew/2023/12/Splitwise_ISCA24.pdf
39. https://docs.lmcache.ai/getting_started/faq.html

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.