GPU SMITH
COMPUTE ENGINEERING

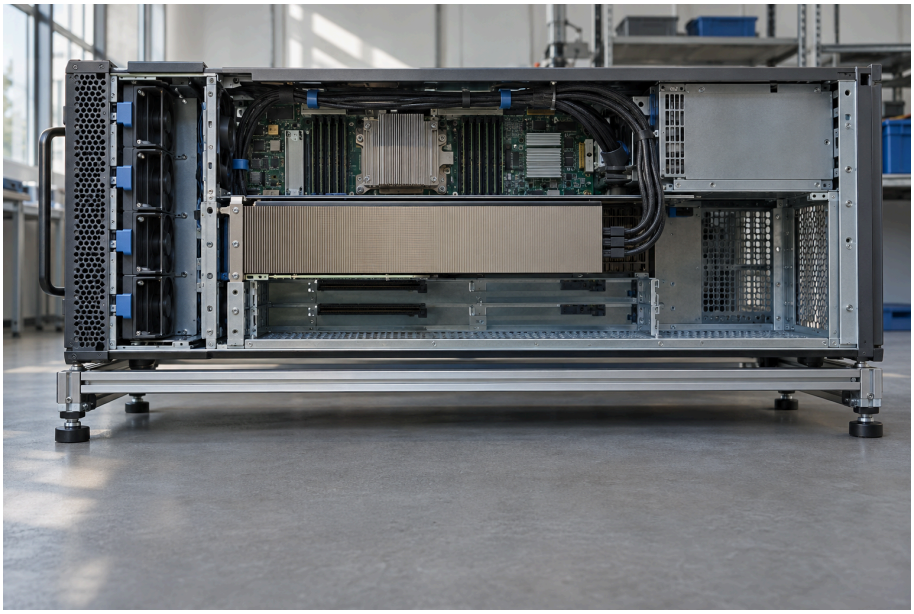
GPU SMITH / RESEARCH & PRACTICAL GUIDANCE

Multilingual LLM GPU Capacity Planning: GPUs per Accepted Answer

> 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 · 2026-11-11 · multilingual llm gpu capacity planning · private llm gpu sizing · llm tokenizer cost by language · non-english token inflation · accepted answers per replica hour · gpu sizing for self-hosted llm · multilingual llm inference cost per request · kv cache capacity planning · energy per accepted answer · vllm capacity planning

Original article: https://gpusmith.com/articles/en/multilingual-private-llm-gpu-capacity-planning



In this article

1. Executive Summary
2. Introduction and Background
3. Key Changes in How Multilingual Capacity Is Planned
4. Per-Language Decision Matrix and Calculation Sheet
5. Implementation Considerations and Process Changes
6. Data Analysis and Evidence
7. Implications and Future Directions
8. Frequently Asked Questions (FAQs)
9. Conclusion

Executive Summary

Multilingual private LLM capacity planning has no universal GPU count. The defensible unit is the **accepted answer**: a correct, useful, language-appropriate response delivered inside the declared latency objective. Capacity per language is then **accepted answers per replica-hour**, measured on the exact model, tokenizer, chat template, runtime and GPU group that will serve production traffic. An English-only pilot cannot supply that number for another language, because token counts, output lengths, stop behavior and quality all move with the language.

The most current primary evidence is *The Language-Energy Divide*, revised on 14 September 2026, which uses Belebele, a reading-comprehension dataset spanning 122 language variants (github.com), and ran its main experiment with Qwen3-8B on a single NVIDIA L40S at batch size 256 (arxiv.org). Its authors report that energy per output token varies by up to 8.3 times across an 8-language subset and total energy for a fixed request set by up to 179 times (arxiv.org). Those are author-reported headline extrema from one model family and two GPU types, and the paper itself says the measured gap is best read as a lower bound for natively authored input (arxiv.org). They are not buying multipliers.

The mechanism is nonetheless planning-relevant. Longer outputs grow the key-value (KV) cache and its memory cost (developer.nvidia.com), and longer output sequences prevent larger batch sizes (ml.energy). Tokenizer length differences are documented, and a tokenizer change can alter counts for the same text, as Change 2 below shows. Generation pathologies compound the effect: for some languages the model enters a repetitive loop of up to about 4,700 output tokens per question (arxiv.org). A language-appropriate denominator must exclude those cases.

The recommended procedure is to freeze the serving stack, build paired diagnostic cases plus separately authored native samples, measure per-language cells and then the real mixed workload, and compute accepted answers per replica-hour as accepted count times 3,600 divided by matched window seconds. Energy per accepted answer keeps failed and retried work in the numerator and is undefined when zero answers pass, and energy per response equals energy per token times output tokens (ml.energy). Small per-language cells need interval estimates, such as the Wilson interval (www.itl.nist.gov).

The output is a per-language enable, pilot or defer matrix with evidence owners and a reproducibility manifest. Deferral means "evidence incomplete", never a quiet decision to suppress a low-resource language to improve a fleet efficiency score. As of November 2026 no public benchmark substitutes for these measurements on a given deployed model, and the calculation sheet in this report ships with blank measured inputs.

Introduction and Background

Enterprises that run a private assistant on their own GPUs eventually face the same question: can the existing fleet serve a new language at an acceptable quality and latency, or does enabling it require a configuration change, an output policy change or more hardware? GPU Smith, a private AI infrastructure engineering consultancy, frames that question as an acceptance problem rather than a model-shopping exercise. This report is a reference for operators, multilingual product owners and procurement or performance engineers. It is a method and template rather than a delivered GPU count: it contains no new benchmark runs, and every number below is attributed to the source that produced it.

Three facts make the question harder than a standard sizing exercise. First, token counts depend on the tokenizer, and vendor counting endpoints count under the tokenizer of the model named in the request (docs.claude.com), so the same meaning produces different sequence lengths on different models. Second, tokenizers are built on byte-level or subword schemes; byte-level BPE uses 256 byte values as its base vocabulary (huggingface.co), and UTF-8 preserves the full US-ASCII range (www.rfc-editor.org) while characters from U+0800 to U+FFFF take three octets (www.rfc-editor.org), so Latin-script text starts with a structural advantage in byte terms. Third, the quality of the answer is itself language-dependent, so a throughput figure that ignores quality overstates useful capacity.

The primary study used throughout is the revised *Language-Energy Divide* paper. Its authors present a systematic study of inference energy across languages using the ML.Energy framework (arxiv.org), and the page lists a CC BY-NC-SA 4.0 license (arxiv.org). Several authors overlap with the ML.ENERGY initiative, so the paper's reuse of ML.ENERGY methods is coauthored technical reuse, not independent endorsement, and the arXiv host is a repository, not a second opinion. NVIDIA's June 2026 technical blog likewise acknowledges contributors from the ML.ENERGY initiative at the University of Michigan (developer.nvidia.com), so it is relevant vendor editorial and not independent validation. Independent systems research does cite the ML.ENERGY leaderboard: a USENIX ATC 2025 paper lists it in its bibliography (www.usenix.org). That is evidence of reuse of a response-level reference, not of any language-specific result.

The remainder of the report follows the order of an acceptance program: what changes in planning, how to fill the decision matrix and calculation sheet, how to implement the process, and what the evidence does and does not support.

Key Changes in How Multilingual Capacity Is Planned

Change 1: From tokens per second to accepted answers per replica-hour

Raw throughput answers the wrong question. A server can sustain high token output while a large share of that output is a repetitive loop, a truncated answer or a reply in the wrong language. Systems research already separates throughput from goodput: DistServe defines per-GPU goodput as the request rate that can be served while adhering to the service level objective (SLO) (arxiv.org), tied to an attainment goal such as 90 percent of requests for each GPU provisioned (arxiv.org). Higher per-GPU goodput therefore lowers cost per query for a fixed demand.

The multilingual extension is to add correctness and language appropriateness to the attainment test. An answer is accepted only if all of the following hold:

- **Inside the SLO:** full-answer latency, and TTFT and time per output token where declared, meet the objective at the planned concurrency.
- **Correct and useful:** the answer matches the expected task outcome under the declared rubric, scored by fluent reviewers for native samples.
- **Language-appropriate:** the response is in the expected language and script, with mixed-script and code-switching cases handled by explicit rules.
- **Legitimately stopped:** the stop reason is a natural stop or stop sequence, not a `length` stop or a loop.
- **Counted once:** retries and regenerations stay in the attempted count and in the energy numerator.

The quality rubric is declared before measurement, so that thresholds are not tuned after seeing the results. The vLLM benchmark CLI already supports SLO-based goodput accounting by letting users specify objectives for goodput (docs.vllm.ai), which covers the latency part; the quality part must be added by the evaluation harness.

Reliability practice points the same way. The Google SRE book advises distinguishing the latency of successful requests from that of failed requests (sre.google) and notes that policy can define a slow response as an error (sre.google). A multilingual acceptance test applies the same idea to semantic failure.

Change 2: From a character-per-token rule to the deployed tokenizer

No universal ratio of characters, bytes or words per token holds across languages or across tokenizer generations. Vendor rules of thumb are English-oriented: Google states that a Gemini token is about 4 characters (ai.google.dev), and the tiktoken README says each token averages about 4 bytes in practice (github.com). Published cross-language measurements show why such averages do not transfer. A study of the ChatGPT and GPT-4 tokenizers found Italian needed about 1.6 times the tokens of English, with Arabic at 3 times (arxiv.org), and the same study reports differences of up to 15 times for the same text across languages (arxiv.org). Those are 2023 measurements on specific older tokenizers, so they illustrate the phenomenon and are not current sizing inputs.

Vendors have changed tokenizers since. Meta states that Llama 3 uses a 128K-token vocabulary that encodes language much more efficiently (ai.meta.com) and reports up to 15 percent fewer tokens than Llama 2 (ai.meta.com). Those are vendor claims for that model family. Anthropic documents that a newer tokenizer

produces counts roughly 30 percent higher than on earlier models (docs.claude.com), and that token counting uses the tokenizer of the model named in the request (docs.claude.com). The practical rule is to count tokens with the exact deployed model's tokenizer revision and chat template, and to repeat the count on every model or tokenizer upgrade.

Change 3: From average response length to output distributions and pathologies

The mean hides the tail that consumes memory. The Language-Energy Divide paper describes three pathologies. In token explosion, the model enters a repetitive loop and emits up to about 4,700 output tokens per question (arxiv.org). In over-generation, some languages produce more tokens than English at similar per-token cost. In language mismatch, the model responds in English or another high-resource language (arxiv.org). The paper reports no per-pathology rates, so operators must measure them on their own model.

Degenerate repetition is a known decoding behavior. The vLLM documentation acknowledges that models sometimes produce repetitive token patterns and stop only at the maximum length (docs.vllm.ai), and it exposes an n-gram repetition detector that ends generation early (docs.vllm.ai) and a repetition penalty (docs.vllm.ai). The vLLM sampling parameters also include a hard cap on tokens per output sequence (docs.vllm.ai), which is the setting that converts a loop into a `length` stop.

Record the stop reason for every response. The OpenAI API reference defines `length` as the maximum token count being reached (platform.openai.com) and `stop` as a natural stop point or provided stop sequence (platform.openai.com); the vLLM OpenAI-compatible server follows the same convention, although the vLLM documentation does not spell the values out. A response that ends with `length` is a truncation candidate, not an accepted answer.

Change 4: From one batch setting to memory-aware concurrency

KV cache is the mechanism that links token inflation to capacity. NVIDIA gives the per-token KV size as a product that starts with a factor of 2 for keys and values (developer.nvidia.com) and notes that the requirement grows linearly with batch size and sequence length (developer.nvidia.com), and that it limits the throughput that can be served (developer.nvidia.com). Lowering the maximum number of sequences reduces the number of concurrent requests in a batch and therefore KV demand (docs.vllm.ai), at the cost of throughput.

Serving engines expose the relevant levers. vLLM documents a maximum number of sequences per iteration (docs.vllm.ai), a maximum number of batched tokens per iteration (docs.vllm.ai), a context length that covers prompt and output (docs.vllm.ai), and a GPU memory fraction pre-allocated for the cache (docs.vllm.ai). Chunked prefill is enabled by default where possible (docs.vllm.ai), which affects how long prompts interleave with decoding. When KV space runs out, vLLM can preempt requests (docs.vllm.ai) and recompute them later (docs.vllm.ai), so long-sequence languages can appear as latency spikes rather than errors. Prefix caching lets a new query reuse the KV cache of a shared prefix (docs.vllm.ai); warm-cache tests and uncached work must be reported separately, as shared system prompts flatter every language equally while per-user text does not.

The study authors report that the per-token energy disparity persists at every batch size tested (arxiv.org) and attribute it primarily to sequence-length effects. The ML.ENERGY v3 analysis makes the same general point for task types, stating that longer output sequences stress memory capacity and prevent larger batch sizes

(ml.energy).

Change 5: From English-pilot quality to paired plus native evaluation

Paired diagnostics isolate tokenizer and generation effects, but they do not represent native use. Belebele is built from 488 distinct passages with one or two questions each (github.com), and the Language-Energy study uses it in full. The Global-MMLU authors report that translation often introduces artefacts that can distort the meaning or clarity of questions (arxiv.org), so translated tests are a diagnostic, not a substitute for native content. The study under review states its own limit as well: its prompts are standardized translations, and the measured gap is a lower bound for natively authored input.

A sound program therefore uses both layers. Paired cases give auditable lengths and expected outputs. Separately authored native tasks, reviewed by fluent reviewers, give the quality denominator. Where public benchmarks are translated, record how the translation was produced and reviewed.

Per-Language Decision Matrix and Calculation Sheet

The decision matrix

Table 1 below is the distinctive reference asset: one row per language and task, with measured fields that decide whether to enable, pilot or defer. It is a template with declared thresholds to be filled by the operator; it contains no delivered benchmark values.

Evidence field	Enable	Pilot (limited traffic, monitored)	Defer (evidence incomplete)
Quality on native task sample	Meets the declared rubric floor, with the lower bound of a Wilson interval above the floor (www.itl.nist.gov)	Point estimate meets floor but interval is wide, or native review covers a narrow set of tasks	Below floor, or no fluent-reviewer sample exists
Wrong-language and mixed-script rate	Within declared ceiling on language-identification checks, with short-text limits noted (github.com)	Elevated on short inputs or code-switched text	Dominant failure mode, or identification tool unvalidated for the script
Truncation and loop rate	Zero or within ceiling, with the upper bound stated (zero observed in n trials gives roughly 3/n at 95 percent, per the rule of three (arxiv.org))	Loops confined to identifiable input types with a mitigation in test	Loops or `length` stops frequent enough that accepted rate is unstable
p95 and p99 latency vs SLO	Both inside SLO at the planned concurrency	p95 inside SLO, p99 outside, with a defined queue policy	p95 outside SLO at any tested concurrency
Accepted answers per replica-hour	Measured on matched windows, mixed-workload verified	Isolated-cell rate only	Zero accepted answers, reported as undefined
Energy per accepted answer (J)	Reported at a declared boundary, failed work in numerator	Reported for the isolated cell only	Undefined (no accepted answers)
Evidence owner and follow-up	Named owner, rerun trigger on model, tokenizer or runtime change	Named owner, date for next sample	Named owner, specific missing evidence listed

The matrix keeps "defer" distinct from "suppress". A deferred language is a language whose evidence is incomplete, with an owner and a list of what is missing. It is not a recommendation to withhold service from low-resource languages to raise a fleet efficiency score, and the study authors themselves recommend that multilingual models report energy for each individual language (arxiv.org).

The calculation sheet

Table 2 below lists the proposed calculation sheet. All measured inputs are blank by design.

Quantity	Formula	Measured inputs required	Notes
Accepted answers per replica-hour	accepted_count x 3600 / matched_window_seconds	Accepted count, matched window seconds, GPUs per replica	Accepted means correct, language-appropriate, inside SLO and legitimately stopped
Attainment	accepted_count / attempted_count	Attempted and accepted counts per language	Attainment is the proportion of requests that meet the SLOs
Required replicas	ceil(demand_accepted_per_hour x (1 + reserve) / (observed_rate x (1 - interference_derate)))	Demand forecast, declared reserve, declared interference derate	Verify on the mixed workload; isolated rates need not add linearly under batching
Total GPUs	required_replicas x GPUs_per_replica	Tensor or pipeline parallel degree	Count GPUs per replica explicitly, including tensor or pipeline parallel groups
Concurrency check	L = arrival_rate x mean_latency	Arrival rate, mean full-answer latency	Little's law relates mean units in the system to arrival rate and time in system (econpapers.repec.org) under stationarity conditions (econpapers.repec.org)
KV demand per sequence	bytes_per_token x (prompt_tokens + output_tokens)	Layers, KV heads, head dimension, precision	Compare to the memory left after weights, at the planned maximum sequences
Energy per accepted answer	measured_energy_J for the matched window / accepted_count	Energy at a declared boundary; failed and retried work retained	Report undefined if accepted_count is zero

The calculation sheet obeys three rules. Demand must be expressed in accepted answers, not requests. Reserve and interference are declared assumptions, not measured facts. Isolated language rates are inputs to a bound, and the mixed workload is the test. Operators should also remember that the SRE book recommends a utilization target because many systems degrade before reaching 100 percent (sre.google), and that latency increases are often a leading indicator of saturation (sre.google).

A worked calculation (Hypothetical Example)

The numbers below are invented to show arithmetic only. A language cell on a 2-GPU replica attempts 600 requests in a 1,800-second matched window, and 420 are accepted. Accepted answers per replica-hour is 420 x 3,600 / 1,800 = 840. If demand is 5,000 accepted answers per hour, with a declared 25 percent reserve and no interference derate, the bound is 5,000 x 1.25 / 840 = 7.44, so 8 replicas, or 16 GPUs. If the measured energy for the window is 1.2 megajoules, energy per accepted answer is 1,200,000 / 420, about 2,857 joules. If instead zero answers had passed, the rate would be zero and the energy per accepted answer would be reported as undefined.

A second hypothetical example sizes memory. Using the Hugging Face per-token KV formula, 2 x 2 x layers x KV heads x head dimension bytes for float16 (huggingface.co), a model with 32 layers, 32 KV heads and head dimension 128 needs 524,288 bytes per token. A 600-token sequence then holds about 0.29 GiB, while a 5,700-token sequence (a 1,000-token prompt plus a 4,700-token loop) holds about 2.8 GiB, roughly 9.5 times as much. Many current models use grouped-query attention, which reduces the KV heads and therefore the per-token size, so the deployed model's real configuration must replace these figures.

Implementation Considerations and Process Changes

Step 1: Define languages, tasks, mix and SLO

State which languages, tasks, volume mix and quality and latency requirements justify rollout. Keep language identification explicit. fastText offers models that recognize 176 languages (fasttext.cc), while GlotLID's documentation advises against very short sentences (github.com), recommends pairing with a script identifier (github.com), notes that useful confidence thresholds differ by language (github.com) and points to separate handling for code-switching (github.com). Mixed-script and code-switched inputs belong in the test set as their own cells.

Step 2: Freeze the serving configuration

Record the model, tokenizer revision, chat template, precision, GPU group, runtime version, sampling parameters, output policy and cache behavior. This matters because template handling changes token counts: the Hugging Face documentation warns that adding extra special tokens to templated text is often incorrect or duplicated and can hurt performance (huggingface.co), and the tokenizer applies no truncation by default (huggingface.co); special tokens are handled carefully and not split (huggingface.co).

Also check whether context and output caps count hidden reasoning tokens. OpenAI documents that reasoning tokens are not visible but still occupy context space (platform.openai.com) and that generation can reach the cap before any visible output is produced (platform.openai.com). OpenAI recommends reserving at least 25,000 tokens for reasoning and outputs when starting out with reasoning models (platform.openai.com), and Qwen documents a maximum assistant message length of 38,912 tokens for thinking modes (qwen.readthedocs.io), so a cap tuned on short English chat can be far too small for reasoning output. Anthropic documents that its thinking budget is a target rather than a strict cap (docs.claude.com) and that `max_tokens` remains the hard ceiling on total output (docs.claude.com).

Record what the English pilot actually tested, including prompt length, output cap and whether a shared prefix was warm. The reproducibility manifest should list:

- **Model and revision:** weights, quantization and precision.
- **Tokenizer revision and chat template:** the exact files, since template handling changes token counts.
- **GPU group:** GPU model, memory, GPUs per replica and parallelism degree.
- **Runtime and version:** for example, the ML.ENERGY v3 benchmark recorded vLLM 0.11.1 (ml.energy).
- **Sampling and output policy:** temperature, maximum tokens, stop sequences, repetition controls.
- **Cache behavior:** whether prefix reuse was on and whether the run was warm or cold.
- **Measurement boundary:** what the energy counter covers and the window definition.
- **Data lineage:** the paired set, the native set, reviewer identities by role and the rubric version.

Step 3: Build paired and native test sets

Construct meaning-equivalent paired cases with auditable lengths and expected outputs, plus separately authored native samples reviewed by fluent reviewers. Some tasks are culturally specific and cannot be normalized through translation. Label character-per-token ratios diagnostic, not a fairness or utility score. Tokenizer fertility, the average number of tokens per word, is a diagnostic and not a utility score, and an association between tokenization and accuracy in one dataset is not a rule to attribute poor reasoning to a language.

Step 4: Measure language cells, then the real mix

Run controlled language cells on the same serving configuration, then the actual mixed workload. Report the following for each cell:

- **Attempted and accepted counts:** with the matched window in seconds.
- **Batch occupancy and KV pressure:** including preemption events where the runtime exposes them.
- **Latency percentiles:** p50, p95 and p99 for full answers, plus TTFT and time per output token.
- **Stop reasons:** natural stop, `length`, loop-detector stop and error, as shares of attempts.
- **Energy:** at a declared boundary, with failed and retried work retained.

Also state the arrival process. Use open-loop arrivals where possible; the MLPerf Inference rules send queries on a Poisson distribution (github.com). Be explicit about metric definitions, because vLLM warns that terminology is not standardized across benchmarking tools (docs.vllm.ai) and NVIDIA's documentation notes tools differ on whether TTFT is included in the inter-token average (docs.nvidia.com).

Time to first token (TTFT) includes queuing, prefill and network latency (developer.nvidia.com), and longer prompts raise it (developer.nvidia.com); under excess concurrency requests queue (developer.nvidia.com). Time per output token is calculated once per request, excluding the first token (docs.vllm.ai). Mean latency can hide the tail: the SRE book's example shows 1 percent of requests taking 5 seconds behind a low mean (sre.google). Check that concurrency is bounded, since the vLLM benchmark leaves concurrency unlimited when no cap is supplied (docs.vllm.ai).

For energy, state the boundary. NVML's total energy call returns millijoules for the GPU since the driver was last reloaded (docs.nvidia.com) and returns an error on devices without support (docs.nvidia.com); power readings on Ampere or newer are averaged over a 1-second interval (docs.nvidia.com). DCGM exposes the same counter as a field (docs.nvidia.com). The Zeus paper measures GPU power through NVML (www.usenix.org), and the Zeus library supports NVIDIA devices via NVML (ml.energy), supports multiple concurrent measurement windows (ml.energy), and documents typical call overhead below 10 ms (ml.energy).

GPU-only measurement is a boundary, not the whole facility: the Zeus paper cites GPUs at around 70 percent of total training energy in the benchmarks it reviews (www.usenix.org), and ML.ENERGY found that CPU-side vision preprocessing can be a bottleneck that reduces batch size (ml.energy). Declare the boundary and keep it fixed across languages; CPU-side tokenization and language identification fall outside a GPU-only counter.

Step 5: Test output caps and mitigations

Test fixed token caps against task completion by language. If a cap change is justified, recheck memory, concurrency and abuse protection; a larger allowance alone does not fix comprehension or looping. The study authors suggest language detection to apply per-language maximum output lengths (arxiv.org). Mitigations such as repetition detection (docs.vllm.ai) and penalties (docs.vllm.ai) need their own quality regression. Translation-first processing is a separate quality and privacy experiment: the study notes that a translate-then-process pipeline can reduce total energy even after two translation steps (arxiv.org), but it adds a translation model, data-handling exposure and a fresh quality risk, so it is not a default shortcut.

Step 6: Allocate headroom and repeat on upgrades

Allocate fleet headroom from measured accepted throughput under the actual language mix, and publish uncertainty when cells are small. NIST notes that normal-approximation intervals can be inaccurate when failures or the sample size are very small (www.itl.nist.gov), so use Wilson or exact intervals. Repeat the acceptance tests whenever the model, tokenizer or runtime changes. The tokenizer change discussed under Change 2 is one documented case of a counting change between model versions.

Unresolved-data ledger

Keep a ledger of what is still unknown and who owns it:

- **Exact deployed languages and model support:** owner is the model platform team.
- **Task meaning and review quality:** owner is the multilingual product owner and the fluent reviewers.
- **Context and output caps versus hidden reasoning tokens:** owner is the serving engineer.
- **Special tokens and chat-template revision:** owner is the serving engineer.
- **Runtime stop behavior and `length` rates:** owner is the performance engineer.
- **Actual arrival mix and shared-prefix reuse:** owner is the product analytics owner.
- **Hardware normalization and energy measurement boundary:** owner is the performance engineer.
- **Statistical uncertainty per language cell:** owner is the evaluation lead.

Data Analysis and Evidence

Table 3 below summarizes the published evidence used in this report, what each source supports, and what it does not. It is organized by source type so that coauthored and independent materials stay separate.

Source	What was measured	Supports	Does not support
Language-Energy Divide v2 (arXiv, 14 Sep 2026)	122 Belebele languages, Qwen3-8B on one L40S at batch 256; cross-model tests on L40S and RTX 6000 Pro Blackwell GPUs	Language-dependent output tokens, per-token energy and pathologies; disparity persists at all tested batch sizes and across Qwen3 sizes	Cross-hardware generalization (the authors state absolute values are hardware-specific (arxiv.org)), closed models, native-language use; coauthored with ML.ENERGY contributors
Belebele (Meta repository and paper)	900 questions per language, 488 passages (github.com), 122 variants	Paired reading-comprehension diagnostic with common structure across languages	Enterprise task quality; the dataset is translated from English
Tokenization premium study (2023)	Same text across languages on ChatGPT and GPT-4 tokenizers	Existence and size of length gaps on those tokenizers; even Portuguese, the closest to English, still needed about 50 percent more tokens (arxiv.org)	Current tokenizer behavior
ML.ENERGY v3 blog (December 2025)	1,858 configurations on H100 and B200 (ml.energy) with vLLM 0.11.1 (ml.energy)	Energy per response equals energy per token times output tokens (ml.energy); longer outputs limit batch size	Language-specific penalties; authors overlap with the language study
CLONE (USENIX ATC 2025)	Independent systems paper citing the ML.ENERGY leaderboard	Independent reuse of a response-level reference (www.usenix.org)	Any language result or current GPU sizing figure
MLPerf Inference rules (MLCommons)	Latency constraints, accuracy and length checks	Declared TTFT and time-per-output-token limits, for example 2000 ms and 200 ms in a conversational category (github.com) and 450 ms and 40 ms in an interactive one (github.com), plus a generation-length check (github.com)	Multilingual benchmarks; the cited limits are for a specific Llama 2 70B benchmark

The study's headline extrema deserve careful handling, and the study's native-language caveat applies to every row above (arxiv.org). The 8.3 times per-token figure is described in the paper as the ratio between the most and least expensive of an 8-language subset (arxiv.org). The four low-resource languages in that subset emit far more output tokens per response than English. The authors also state that accuracy and energy co-occur, but emphasize this is a co-occurrence, not a causal relationship (arxiv.org). Because these extrema are author-reported and have not been independently reproduced, they inform what to measure and not what to multiply.

The response-level framing is consistent across the ML.ENERGY materials. A response is a complete answer to a prompt with all output tokens included (ml.energy), the leaderboard reports an energy-per-response column (ml.energy) and filters by a median inter-token latency deadline (ml.energy). The measurement

window in that work is a steady state, and the language study adopts a steady-state window for per-token values. For capacity planning, a steady-state per-token energy is not a substitute for the end-to-end matched window described in *Table 2*, because short and long responses meet different ramp-up and queue conditions.

MLCommons derived its original latency limits from typical human reading speed (mlcommons.org) and requires accuracy above 99.9 percent of reference in its high-accuracy category (mlcommons.org); the multilingual analogue is a declared quality floor per language. Three uncertainty sources matter for any measured cell. Hardware counters carry their own error margin, which the study cites as 5 percent. Statistical error for small cells follows the binomial interval guidance above. Metric-definition differences between benchmarking tools (docs.vllm.ai) can shift reported latencies without any change in the system. Finally, the mixture can matter: energy per token drops at first and then plateaus as GPU utilization nears saturation (ml.energy), and larger batches trade higher latency for a 3 to 5 times reduction in energy per token (ml.energy), so the batch operating point should be reported next to every energy figure.

Implications and Future Directions

For procurement. Hardware choices change the sequence-memory trade-off. NVIDIA lists the H100 SXM memory bandwidth at 3.35TB/s (www.nvidia.com), the L40S at 48GB of GDDR6 with ECC (www.nvidia.com) and the RTX PRO 6000 Blackwell Server Edition at 96 GB (www.nvidia.com). Memory headroom for KV cache matters as much as peak compute for long-output languages, but the language paper's own cross-hardware comparison is limited to two GPU types, so no hardware ranking is implied here. Procurement documents should specify acceptance tests, not a language multiplier.

For performance engineering. NVIDIA notes that overall energy is governed by computation, hardware efficiency and GPU utilization (developer.nvidia.com), which aligns with the per-language metrics proposed above: more computation per accepted answer, lower utilization through smaller batches, or both. The same NVIDIA blog points readers to the ML.ENERGY leaderboard for benchmark-driven comparison (developer.nvidia.com); treat it as vendor editorial referencing a coauthored resource. Reasoning-style models magnify output length effects: ML.ENERGY reports problem solving averaged 25 times more energy per response than text conversation (ml.energy), so enabling hidden reasoning in a new language needs its own cell.

For product owners. Treat the open evaluation gap honestly: public benchmarks cover few native tasks. A language with sparse evidence should be piloted with monitoring, not hidden.

Future directions. Tokenizer vocabularies continue to change between model generations, and the tokenizer change described under Change 2 shows that a tokenizer update alone can move token budgets. Expect the acceptance program to be re-run, not completed once. Developments to watch include per-language energy reporting and improvements to language identification for short and code-switched text.

Frequently Asked Questions (FAQs)

How many GPUs does a private LLM need for several languages? The count is required replicas times GPUs per replica, where required replicas come from demand in accepted answers per hour divided by the measured accepted rate, with declared reserve and interference. There is no language-independent figure.

GPUs per replica depends on the model and its parallelism configuration.

How does tokenizer cost vary by language? It varies by tokenizer and text. Older commercial tokenizers showed differences up to 15 times for the same text across languages, and a newer vendor tokenizer can change counts as well (see Change 2). Newer tokenizers change these ratios, so measure with the deployed one.

What is non-English token inflation? It is the tendency for the same task to require more prompt or output tokens in some languages. A newer tokenizer can shift counts (see Change 2), and generation pathologies add output tokens when a response loops until the cap (docs.vllm.ai). Inflation therefore has two parts, the tokenizer and the generated answer length, and they should be measured separately.

How should GPU sizing for a self-hosted LLM treat tokens per second? Use tokens per second as a diagnostic, not the capacity number. System throughput rises with concurrency until it reaches a saturation point (developer.nvidia.com), and KV memory limits how much concurrency is possible. Convert to accepted answers per replica-hour at the declared latency objective.

What is multilingual LLM inference cost per request? It is the GPU time or energy for the full response, including prompt processing and all output tokens. ML.ENERGY states that energy per response is energy per token multiplied by output tokens (ml.energy). Cost per request becomes cost per accepted answer when failed and retried work stays in the numerator.

What does a private LLM infrastructure sizing guide need beyond throughput? It needs the quality rubric, the stop-reason distribution, the latency percentiles at planned concurrency, the cache state, and the unresolved-data ledger above. The SRE book's advice to track the 99th percentile over a small window as an early saturation signal applies directly (sre.google).

What is cost per accepted answer? It is total cost over a matched window divided by accepted answers. If zero answers are accepted, report it as undefined, not as a number.

Should a low-resource language be disabled if it lowers fleet efficiency? No. The matrix defers on missing evidence, and the response to a poor result is to investigate quality, caps, tokenizer and runtime behavior and to report per-language results.

Conclusion

Multilingual GPU capacity is a measurement problem with a precise denominator: accepted answers delivered within the declared SLO. The best current primary study reports large language-dependent differences in output tokens, energy and generation behavior on specific open models and two GPU types, and its authors caution that absolute values are hardware-specific and that the gap is a lower bound for native input. That is a reason to run the acceptance program, not a multiplier to apply.

The program is short to state. Freeze the serving configuration, construct paired and native samples, record token distributions, stop reasons and wrong-language outputs, run controlled language cells and then the real mix, and compute accepted answers per replica-hour and energy per accepted answer from matched windows. Bound required replicas with declared reserve and interference, verify on the mixed workload, and publish intervals when cells are small. Repeat on every model, tokenizer or runtime change.

The matrix then yields one of three outcomes per language: enable, pilot with monitoring, or defer with named missing evidence. Deferral is a statement about evidence and ownership. It is never a decision to withhold service from a language in order to improve an efficiency score.

Source links

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

1. https://github.com/facebookresearch/belebele
2. https://arxiv.org/html/2606.21869v2
3. https://developer.nvidia.com/blog/llm-benchmarking-fundamental-concepts
4. https://ml.energy/blog/measurement/energy/diagnosing-inference-energy-consumption-with-the-mlenergy-leaderboard-v30
5. https://www.itl.nist.gov/div898/handbook/prc/section2/prc241.htm
6. https://docs.claude.com/en/docs/build-with-claude/token-counting
7. https://huggingface.co/docs/transformers/en/tokenizer_summary
8. https://www.rfc-editor.org/rfc/rfc3629
9. https://developer.nvidia.com/blog/maximize-ai-factory-energy-efficiency-through-full-stack-inference-and-training-optimizations
10. https://www.usenix.org/system/files/atc25-tian.pdf
11. https://arxiv.org/pdf/2401.09670
12. https://docs.vllm.ai/en/latest/cli/bench/serve
13. https://sre.google/sre-book/monitoring-distributed-systems
14. https://ai.google.dev/gemini-api/docs/tokens?lang=python
15. https://github.com/openai/tiktoken
16. https://arxiv.org/pdf/2305.15425
17. https://arxiv.org/abs/2305.15425
18. https://ai.meta.com/blog/meta-llama-3
19. https://docs.vllm.ai/en/latest/api/vllm/sampling_params.html
20. https://platform.openai.com/docs/api-reference/chat/object
21. https://developer.nvidia.com/blog/mastering-llm-techniques-inference-optimization
22. https://docs.vllm.ai/en/latest/configuration/optimization.html
23. https://docs.vllm.ai/en/latest/configuration/engine_args.html
24. https://docs.vllm.ai/en/latest/features/automatic_prefix_caching.html
25. https://arxiv.org/abs/2412.03304
26. https://github.com/cisnlp/GlotLID
27. https://arxiv.org/pdf/2009.00827v1
28. https://econpapers.repec.org/RePEc:inm:oropre:v:9:y:1961:i:3:p:383-387
29. https://huggingface.co/blog/kv-cache-quantization
30. https://fasttext.cc/docs/en/language-identification.html

31. https://huggingface.co/docs/transformers/en/chat_templating
32. https://huggingface.co/docs/transformers/en/main_classes/tokenizer
33. https://platform.openai.com/docs/guides/reasoning
34. https://qwen.readthedocs.io/en/latest/getting_started/concepts.html
35. https://docs.claude.com/en/docs/build-with-claude/extended-thinking
36. https://github.com/mlcommons/inference_policies/blob/master/inference_rules.adoc
37. https://docs.vllm.ai/en/latest/benchmarking/cli
38. https://docs.nvidia.com/nim/benchmarking/llm/latest/metrics.html
39. https://docs.nvidia.com/deploy/nvml-api/api/group__nvmlDeviceQueries.html
40. https://docs.nvidia.com/datacenter/dcgm/latest/dcgm-api/dcgm-api-field-ids.html
41. https://www.usenix.org/system/files/nsdi23-you.pdf
42. https://ml.energy/zeus/measure
43. https://ml.energy/leaderboard
44. https://mlcommons.org/2024/03/mlperf-llama2-70b
45. https://www.nvidia.com/en-us/data-center/h100
46. https://www.nvidia.com/en-us/data-center/l40s
47. https://www.nvidia.com/en-us/data-center/rtx-pro-6000-blackwell-server-edition

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.