GPU SMITH
COMPUTE ENGINEERING

GPU SMITH / RESEARCH & PRACTICAL GUIDANCE

Prefill Context Parallelism on MI355X: Long-Prompt Gate

> 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 · 2026-11-21 · prefill context parallelism · context parallelism mi355x · amd instinct mi355x · glm-5.2 mxfp4 · sglang rocm · long context prefill · time to first token · llm inference benchmark · vllm rocm

Original article: https://gpusmith.com/articles/en/prefill-context-parallelism-mi355x-long-prompt-gate



In this article

1. Executive Summary
2. Introduction and Background
3. How Prefill Context Parallelism Divides Work and Where It Charges
4. Prerequisites and Support Status as of November 2026
5. Reconstructing AMD's Short, Medium and Long Prompt Test Matrix
6. Telemetry: TTFT, TPOT, Throughput, p95, Errors and Interconnect Load
7. Data Analysis and Evidence
8. Adoption Threshold and Rollback for Mixed Short and Long Traffic
9. Implications and Future Directions
10. Frequently Asked Questions (FAQs)
11. Conclusion

Executive Summary

Prefill context parallelism (CP) on a four-GPU AMD Instinct **MI355X** tensor-parallel group is a prompt-length gate, not a general speedup. AMD's test of GLM-5.2-MXFP4 under SGLang, published on September 21, 2026, found that the sign of the throughput effect flips with input length. At 1,024 input tokens CP lost throughput at every concurrency measured, and AMD summarizes the penalty as **8% to 18%**. At 4,096 tokens the effect sat mostly inside a measurement noise floor of about **8%**, and at 40,960 and 61,440 tokens CP delivered **43% to 54% more total throughput** at concurrency 16 and above (rocm.blogs.amd.com). AMD's per-concurrency table shows the short-prompt loss as 9.4% to 18.3%, a small discrepancy with its own summary that this report keeps visible. All measurements in this report come from that single AMD study, which used one checkpoint on one pinned SGLang branch and has not been independently reproduced, so treat the thresholds below as a provisional map, not a settled rule.

The sign flips because the split is paid for twice. Enabling CP in this SGLang build replicates attention weights on every rank, raising per-rank weights from 101.45 GB to 120.13 GB and removing **199,744 tokens** of KV-cache pool (rocm.blogs.amd.com). Every decode step also becomes more expensive because each rank computes all 64 attention heads instead of 16. The saving is the quartering of the lightning indexer's scoring work, which is real work only once the prompt exceeds the 2,048-key budget. The saving grows with prompt length, while a decode penalty of roughly 1.2 ms per output token recurs on every generated token.

Support status is the second gate. The SGLang pull request carrying the AMD enablement (#32175) was open when AMD wrote its post (rocm.blogs.amd.com), and the GitHub page shows it closed on **November 16, 2026** (github.com). SGLang's **v0.5.20** release notes state that prefill CP on HIP is rejected until the platform is ported (github.com). vLLM's documentation lists ROCm prefill CP only for DeepSeek-V4.1-Flash in eager mode, without speculative decoding or prefix caching (docs.vllm.ai). AMD tested **ROCm 7.2.0** (rocm.blogs.amd.com); **ROCm 10.0.0** shipped on 2026-08-26 (rocm.docs.amd.com), and nothing in this evidence validates CP there.

The practical rule is an on/off worksheet. Adopt CP for a request class only when its same-SLO throughput gain exceeds the larger of AMD's **8.4%** repeat spread and the operator's own 95% repeat spread, when p95 time to first token (TTFT) improves, and when the smaller KV pool still holds the target concurrency. A first-order model built from AMD's concurrency-16 numbers shows that a fleet with 1% of requests at 40,960 tokens loses **11.3%** overall with CP on, while a fleet with 10% long requests gains **6.3%**. Roll back by restarting without the two CP flags, or by routing long prompts to a CP-enabled pool and everything else to a CP-off pool.

Introduction and Background

Operators of long-context private language model deployments on AMD hardware face a recurring question: should prefill context parallelism be enabled on an existing tensor-parallel (TP) group? Red Hat's distributed-inference design guide treats context parallelism as a first-class fifth scaling dimension (developers.redhat.com), and vLLM's documentation says it mainly solves the problem of serving long context requests (docs.vllm.ai). The feature divides the tokens of one prompt across GPUs during the prefill phase, the compute-heavy pass that builds the key-value (KV) cache and produces the first token. vLLM's documentation frames the motivation as controlling time to first token by amortizing prefill computation across query tokens (docs.vllm.ai). The question matters for procurement as well as tuning, because a feature that raises throughput per GPU for one traffic shape and lowers it for another changes how many accelerators a service needs.

This report is written for AMD Instinct cluster operators and procurement engineers. It reconstructs the one public on/off measurement set for this question on MI355X hardware, AMD's September 2026 **GLM-5.2-MXFP4** study, and converts it into a decision worksheet. The study ran four of eight MI355X GPUs at `--tp-size 4`, chosen as the smallest layout that fits the 408 GB of weights with usable KV cache (rocm.blogs.amd.com). Both arms used DeepSeek-style sparse attention and EAGLE speculative decoding. SGLang's cookbook lists GLM-5 serving on AMD MI300X, MI325X and MI355X GPUs (docs.sglang.io).

Three limits shape everything below. First, no original benchmark was run for this report; every measurement is AMD's, and every derived figure (percentage changes, GPU-hours per million tokens, traffic-weighted results, interconnect volumes) is labeled as a calculation from cited inputs. Second, AMD tested one sparse-attention checkpoint on one SGLang branch, so other fleets may see a different crossover. Third, support status moves quickly. Status is stated as of November 21, 2026, and gaps (quality parity, p95 tail latency, interconnect load, other attention layouts) are flagged.

From an independent advisory perspective (GPU Smith does not manufacture accelerators or sell cloud capacity), the useful deliverable is not a verdict on CP but a reproducible gate: a test matrix, a noise floor, an adoption threshold and a rollback path. Abbreviations: TTFT is time to first token, TPOT is time per output token, MoE is mixture of experts, MLA is multi-head latent attention, and DSA is DeepSeek Sparse Attention.

How Prefill Context Parallelism Divides Work and Where It Charges

What CP divides

Context parallelism shards the sequence dimension, which matters because during prefill attention cost grows with the square of the prompt length (developers.redhat.com). NVIDIA's Megatron Core documentation defines CP as a scheme on the sequence-length dimension that partitions the network inputs and all activations, so that all modules except attention work without change (docs.nvidia.com). Attention is the exception because every query needs the keys and values of the whole sequence, so CP requires an additional all-gather across GPUs to collect the full sequence of KV (docs.nvidia.com). The same source notes that each GPU computes on part of the sequence, which scales compute and communication down by the CP degree (docs.nvidia.com).

Inference frameworks implement two families of this idea. vLLM's deployment guide describes "partial query, full key/value", where each GPU gathers key/value tensors from all GPUs and computes attention output for its own query chunk, and "partial query, partial key/value", used when the full key/value tensors no longer fit and ring-attention style send and receive is needed (docs.vllm.ai). vLLM notes that both approaches are under active development (docs.vllm.ai). The ring variant traces to Ring Attention, which distributes long sequences across devices while overlapping communication of key-value blocks with blockwise attention computation (arxiv.org). Meta's inference study applied lossless exact ring variants called pass-KV and pass-Q to Llama 3 405B and reported near-linear prefill latency scaling up to 128 H100 GPUs across 16 nodes (arxiv.org), including a 1M-token prefill of Llama 3 405B in 77 seconds (arxiv.org). DeepSpeed Ulysses, a training-oriented alternative, partitions along the sequence dimension but uses all-to-all communication for attention (arxiv.org).

AMD's setup uses the gather family. With `--enable-prefill-cp --cp-strategy interleave`, prompt tokens are dealt round robin across four ranks (rocm.blogs.amd.com). Reassembly all-gathers keys and values inside every layer and never gathers queries (rocm.blogs.amd.com). The split applies to prefill only, because SGLang enables CP only on extend-step forwards. Decode context parallelism, by contrast, targets KV-cache capacity to increase batch size (docs.vllm.ai), and SGLang's recent notes report GLM-5.2 and GLM-5.3 combining it with multi-token prediction (github.com). It is a different lever from the one tested here.

Why the all-gather cost matters

Two traffic streams cross the interconnect. One is the compact MLA key/value payload of 576 values per token per layer (rocm.blogs.amd.com). The second is larger: before each MoE block, hidden states are all-gathered at the full 6144 width, and after the MoE they are reduce-scattered (rocm.blogs.amd.com). AMD describes the model as 6144 hidden size across 78 layers, and its launch command sets a bfloat16 KV-cache dtype.

A back-of-envelope calculation from those inputs, assuming 2-byte elements and treating every layer as a MoE layer (an upper bound), gives the following gathered volume for one 40,960-token prompt:

- **KV latent:** 576 values × 78 layers × 2 bytes × 40,960 tokens, about **3.7 GB**.
- **Indexer key:** an additional 128 values per token per layer, about **0.8 GB** at the same assumed element size.
- **MoE hidden states:** 6144 values × 78 layers × 2 bytes × 40,960 tokens, about **39 GB** gathered, with a comparable reduce-scatter volume afterward.

The hidden-state stream is roughly ten times the KV latent stream, so the compact latent is unlikely to be the dominant interconnect traffic; this is this report's own inference from the stated widths, not a quoted AMD finding. These are calculations, not measurements; the AMD post reports no link counters. AMD lists seven Infinity Fabric links per MI355X at 144 GB/s peak per link (www.amd.com), and its CDNA 4 whitepaper says the MI350 family connects eight GPUs directly in a node (www.amd.com). If one link serves each peer pair, and if the listed 144 GB/s is treated as a per-direction figure (the product page text cited here does not state this), a four-GPU subgroup has three links per GPU, or about **432 GB/s** at listed peak; the link count per peer pair is also an assumption, and other AMD materials may quote a different per-link figure. At that peak, a rank receiving three quarters of 39 GB spends on the order of 70 ms on the all-gather and a similar time on the reduce-scatter, against AMD's derived single-stream prefill saving of **827 ms** at 40,960 tokens. Achieved bandwidth is always below peak, so the real share is higher than this lower bound, and it is the quantity worth measuring. Treat the 70 ms figure as an order-of-magnitude estimate, not a measurement.

What CP charges in this implementation

AMD's explanation of the price is specific. Enabling CP forces attention tensor parallelism to a size of one, so attention weights are replicated on all four ranks rather than sharded. Per-rank weights rise to **120.13 GB** with CP against **101.45 GB** without, an **18.68 GB** penalty paid out of the KV cache (rocm.blogs.amd.com). That is 199,744 tokens of pool capacity. In compute, each rank evaluates all 64 heads instead of 16. CP does not shard the KV cache here; every rank writes the full gathered KV for the whole sequence into its own pool.

AMD stresses that this reduction of attention TP to one is a property of the implementation, not of context parallelism in general. That distinction matters when reading other frameworks. In vLLM, TP and PCP multiply, so TP=2 with PCP=2 requires four GPUs (docs.vllm.ai), and Red Hat's design article says PCP expands the world size and runs in its own communication domain (developers.redhat.com). In AMD's SGLang setup CP instead reuses the four TP ranks. This answers the context parallel versus tensor parallel prefill latency question: they are not interchangeable knobs on the same GPU count in every framework, and here CP consumes attention TP to buy token-level splitting.

What the split actually saves

AMD's analysis is that MLA attention comes out roughly even, because each rank handles a quarter of the query rows but all heads. What the split genuinely divides is the lightning indexer's scoring, and that is the only term in the stack that grows faster than the prompt (rocm.blogs.amd.com). DSA is the sparse-attention mechanism introduced with DeepSeek-V3.2 to reduce computational complexity in long-context scenarios (arxiv.org) (docs.sglang.io). Below the 2,048-key budget, SGLang with CP off skips indexer ranking altogether because top-k would keep every key anyway, a fast path CP cannot use. SGLang's cookbook states that GLM-5 and DeepSeek-V3.2 share the same model structure, so the optimizations carry over. That is the mechanism behind the sign flip, and the reason the crossover may differ for dense-attention models or other top-k budgets.

Prerequisites and Support Status as of November 2026

The prerequisites split into hardware, model and software, and each has a status as of November 21, 2026 and an operator action.

- **Accelerator:** AMD ran four of eight MI355X GPUs (gfx950) in one node. AMD lists the MI355X at 288GB HBM3E and 8TB/s bandwidth (www.amd.com) with 1400W typical board power (www.amd.com). Confirm that the TP4 group sits on one xGMI-connected node and log the topology with `amd-smi`.
- **Model:** the checkpoint is `amd/GLM-5.2-MXFP4`, 282 safetensors shards and 408 GB. The model card says it was quantized from zai-org/GLM-5.2 with AMD Quark (huggingface.co). Neither the card nor the AMD post pins a revision hash, so the exact revision is an unverified gap; record the snapshot hash you test.
- **Serving framework:** AMD used SGLang branch `refactor/amd-glm5x-cp-eagle-20260723`, tied to a pull request titled "AMD ROCm enablement for GLM-5.x" (github.com). That PR is closed (details below), and SGLang's v0.5.20 notes reject HIP prefill CP (github.com). SGLang's documentation for enabling prefill CP names CUDA (docs.sglang.io). Pin the branch commit, and do not assume a stock image has CP.
- **ROCm:** AMD tested ROCm 7.2.0 with driver 6.16.13. ROCm Core SDK 10.0.0 is dated 2026-08-26 (rocm.docs.amd.com), and SGLang v0.5.23 publishes ROCm 10 images for MI35x (github.com). Treat ROCm 10 as untested for CP and re-run the full matrix.
- **vLLM path:** not tested by AMD. vLLM documents ROCm eager prefill CP for DeepSeek-V4.1-Flash only (docs.vllm.ai), requiring PP=DP=DCP=1 (docs.vllm.ai). It is a different model with different constraints and is not a substitute for the GLM-5.2 result.
- **Speculative decoding:** AMD ran EAGLE on both arms, while SGLang's GLM-5 cookbook says EAGLE is not currently supported on AMD for GLM-5 (docs.sglang.io). Expect to carry the branch's EAGLE support and test with your production draft settings.

The AMD result therefore rests on a stack that no stock release currently reproduces. The PR status deserves precise wording. AMD wrote that the PR was open at the time; the PR page fetched for this report shows it closed on November 16, 2026, with a repository bot citing a cap on idle open PRs per contributor, not a technical objection (github.com). That is a housekeeping closure, and it also means the PR was not merged. Separately, the v0.5.20 notes say the strategy-based implementation is now the only prefill CP path, and that prefill CP on HIP, NPU and MUSA is rejected until those platforms are ported (github.com). AMD's post describes the branch, at commit `817f2e8b9e`, as a `v0.5.15.post1`-based lineage (rocm.blogs.amd.com). This report infers, rather than quotes from AMD, that this base predates the v0.5.20 change. For an operator, the safest reading of the public record is that prefill CP for GLM-5.x on MI355X exists on a pinned branch, and no released SGLang version on the fetched releases page documents it for HIP.

The ROCm 10 question needs equal care. ROCm Core SDK 10.0.0 lists SGLang 0.5.15 and vLLM 0.27.0 among supported frameworks (rocm.docs.amd.com), and the MI355X is listed under gfx950 (rocm.docs.amd.com). That is a compatibility statement for framework versions, not a validation of a CP branch, and AMD measured on ROCm 7.2.0. Do not carry the numbers across a major ROCm version without re-measuring.

On vLLM, Red Hat's deployment article describes the intended production pairing of TP within a node and PCP across GPUs to split the quadratic attention work (developers.redhat.com), and notes that prefix caching skips prefill entirely for repeated documents (developers.redhat.com). Operators whose long prompts are mostly repeats should model the cache before CP.

Reconstructing AMD's Short, Medium and Long Prompt Test Matrix

AMD's workload design is simple enough to reproduce. All sweeps used SGLang's `bench_serving` with the random dataset and 512 output tokens, and every sweep point pinned prompt length exactly with a range ratio of 1.0. Each configuration ran on a freshly started server. Prefix caching was disabled so the random dataset would not serve shared prefixes for free. A separate single-stream probe used 4,096 input tokens, 512 output tokens and 16 prompts at concurrency 1.

The matrix has three regimes, and *Table 1* below condenses the results:

- **Short:** 1,024 tokens at concurrency 4, 8, 16, 32 and 48.
- **Medium:** 4,096 tokens at the same five concurrencies.
- **Long:** 40,960 and 61,440 tokens at concurrency 1, 4, 8, 16, 20, 24 and 32, seven points each.

In the table, the percentage changes and the count of points outside the 8.4% repeat spread are calculations from AMD's published throughput tables; the TTFT figures are AMD's medians.

Input tokens	Total throughput change, CP on vs off	Points outside ±8.4% spread	Median TTFT, CP off to on
1,024	-9.4% to -18.3% across c4 to c48 (rocm.blogs.amd.com)	5 of 5 (all losses)	417 ms to 452 ms at c16 (slower)
4,096	-7.5% to +8.5% across c4 to c48	1 of 5 (c32, +8.5%)	about 1,680 ms to 1,238 ms at c16 (rocm.blogs.amd.com)
40,960	+4.1% (c1) to +46.9% (c20); +43.1% at c16 (rocm.blogs.amd.com)	6 of 7 (c1 inside)	2.52 s to 1.70 s at c1; 23.38 s to 14.00 s at c16
61,440	+11.4% (c1) to +54.1% (c16); +48.3% at c32 (rocm.blogs.amd.com)	7 of 7	3.95 s to 2.58 s at c1; 35.10 s to 22.07 s at c16

The matrix reads as a clean crossover. At 1,024 tokens every point is a loss, and the mechanism is not subtle: CP has to run the indexer on every rank while the CP-off path skips it. At 4,096 tokens AMD concludes that CP is neither clearly right nor wrong under load through concurrency 16, with only concurrency 32 clearing the noise floor. At the long lengths every one of the fourteen throughput points is a CP win, and the single-stream rows show a TTFT gain at concurrency 1, which batching cannot explain (rocm.blogs.amd.com).

Two details deserve attention. First, the TTFT rows show that medium prompts can benefit on first-token latency even where throughput is inside noise, so a latency-bound product and a throughput-bound product can reach different decisions on the same traffic. Second, AMD's figure caption says the 1,024-token series at concurrency 48 lands "just inside" the noise band at -8.3% (rocm.blogs.amd.com), while the tabulated value is -10.3%. The two are not reconciled in the post, and the more conservative reading used in *Table 1* is the table value. The same applies to the headline range: the prose says 8% to 18%, and the table's range is 9.4% to 18.3%.

Chunked prefill, which is enabled by default in vLLM v1 (developers.redhat.com), does not undo the long-prompt gain in AMD's SGLang runs. At the configured chunk size, a 40,960-token prompt arrives as 32,768 plus 8,192 tokens, and the indexer fast path is gated on total sequence length, not chunk length. Nor does CP

win by drafting better: accept length was 5.84 to 5.91 with CP on and 5.76 to 5.91 with CP off in the long-context runs.

Telemetry: TTFT, TPOT, Throughput, p95, Errors and Interconnect Load

What AMD published and what it did not

AMD reports median TTFT, median end-to-end latency, single-stream TPOT and per-GPU total throughput, but no p95 or p99 latency, so tail behavior under production concurrency is an open item. On errors, AMD states that all 28 long-context points completed successfully, and that at 61,440 tokens only about nineteen requests fit in the CP-on pool, after which the server retracts requests instead of failing them. Retraction costs latency, not correctness, in AMD's wording. Interconnect load was not reported.

How to capture each metric on your own deployment

Prometheus histograms are the right instrument for percentiles. SGLang exposes Prometheus metrics when launched with metrics enabled (docs.sglang.io), and its TTFT metric measures the time from request creation in the API server to the first token (docs.sglang.io) (docs.vllm.ai). vLLM exposes a TTFT histogram, a prefill-phase time histogram (docs.vllm.ai), KV-cache usage as a gauge (docs.vllm.ai) and a preemption counter (docs.vllm.ai). Red Hat's guidance names the same family of signals, including queue time, prefill duration, decode duration and KV-cache utilization (developers.redhat.com). One statistical caution applies to the p95 target: pre-computed quantiles cannot be aggregated across instances, so percentiles should be derived from histogram buckets (prometheus.io).

A minimum capture list for each on/off arm:

- **TTFT p50 and p95:** from histogram buckets, per prompt-length class, not pooled.
- **TPOT or inter-token latency:** per class, because the decode penalty is paid on every generated token.
- **Total throughput per GPU:** tokens in plus tokens out per second, the unit AMD uses.
- **End-to-end latency:** AMD's model reproduces measured latency to within 0.005% as TTFT plus 511 times TPOT at 512 output tokens.
- **Errors and retractions:** HTTP failures, timeouts and the SGLang log line "KV cache pool is full. Retract requests.", or the preemption counter on vLLM.
- **KV pool occupancy:** the CP-on pool is 199,744 tokens smaller, so the concurrency ceiling moves.
- **Interconnect counters:** described next.

Interconnect telemetry

The ROCm Systems Profiler samples XGMI link width, link speed, accumulated read data and accumulated write data through AMD SMI (rocm.docs.amd.com), and AMD says these metrics help identify GPU-to-GPU communication patterns and bandwidth utilization (rocm.docs.amd.com). Availability depends on topology and architecture, and the metrics exist only on systems with multiple GPUs connected via XGMI links (rocm.docs.amd.com). The `amd-smi` command line has an `xgmi` subcommand that displays XGMI information, and its topology output reports whether two GPUs are reachable over xGMI regardless of the number of

physical links (rocm.docs.amd.com). The practical procedure is to sample the accumulated counters before and after a fixed batch of long prompts, divide by elapsed time, and compare the CP-on delta against the back-of-envelope volume above. Achieved bandwidth near the link ceiling during prefill signals that the interconnect, not compute, is the limit.

Data Analysis and Evidence

Speedup, penalty and cost per million tokens

AMD's single-stream decomposition shows the mechanism in numbers. On the reference workload of about 1,928 input and 281 output tokens, CP cut median TTFT from 246.99 ms to 204.38 ms but raised TPOT from 3.91 ms to 5.60 ms, and AMD calls the resulting net loss the CP floor, not its verdict. At concurrency 1 for 40,960 input tokens, the derived prefill saving is 827 ms against a decode penalty of 639 ms, a net gain of 188 ms; at 61,440 tokens the saving is 1,361 ms against 726 ms, with the decode penalty near 1.2 ms per output token. The break-even rule that follows is a simple inequality, saving(L) greater than (N_out - 1) times ΔTPOT, where L is the input length and N_out the output length.

Output length moves the break-even sharply. AMD's model puts single-stream break-even near **103,000** input tokens at 2,048 output tokens and near **181,000** at 4,096 (rocm.blogs.amd.com). Those are modeled values, and AMD says it did not measure longer outputs under load, so the results are established for 512-token outputs and untested for longer ones. This matters for GLM-5.2 deployments with reasoning traces: for single-stream long reasoning, AMD's guidance is to expect a faster first token without a throughput win.

Converting throughput into capacity terms uses the identity GPU-hours per million total tokens equals 1,000,000 divided by (tokens per second per GPU times 3,600). *Table 2* applies it to AMD's concurrency-16 throughput numbers and prices the difference at a **Hypothetical Example** rate of **$3.00 per GPU-hour**. The rate is illustrative, not a market quote; substitute your own contract price.

Input tokens (c16)	Throughput off to on, tok/s per GPU	GPU-hours per M tokens, off to on	Delta GPU-hours per M tokens	Delta cost at $3.00 per GPU-hour (Hypothetical Example)
1,024	1,099.4 to 937.5	0.2527 to 0.2963	+0.0436	+$0.131
4,096	2,386.0 to 2,498.7	0.1164 to 0.1112	-0.0053 (inside noise)	-$0.016
40,960	3,825.1 to 5,473.2	0.0726 to 0.0508	-0.0219	-$0.066
61,440	3,658.7 to 5,639.8	0.0759 to 0.0493	-0.0267	-$0.080

Table 2 makes the asymmetry concrete. A short-prompt class pays 0.0436 extra GPU-hours per million tokens with CP on, while a 40,960-token class saves 0.0219. In GPU-hours, which does not depend on the assumed price, the per-token gain on long prompts is smaller than the per-token loss on short prompts. Whether a mixed fleet is net negative depends on its mix of requests and tokens, which *Table 3* addresses, so a fleet dominated by short prompts can lose even if the long class looks excellent in isolation. The 4,096-token row sits inside the 8.4% spread and should be treated as zero.

Independent context for MI355X LLM inference benchmarks

Independent operator reports on this hardware give context but not a CP comparison. One MI355X operator reported that a 20,000-token input workload with 60% cache hits was primarily prefill bound, and that switching from TP8 to TP4 with data-parallel replication of two improved throughput markedly on that workload (www.wafer.ai). That is a vendor-run result on a different configuration, but it shows that replica-level data parallelism is a competing lever for prefill-heavy traffic: it adds throughput without CP's attention-replication penalty, though it gives no single-request TTFT reduction.

Traffic-weighted fleet result

A fleet serves several request classes, so the on/off decision should be evaluated on the mix. The first-order model is: fleet throughput equals total tokens divided by the sum over classes of (class tokens divided by class throughput). Equivalently, when shares are measured as fractions of GPU-time, fleet throughput is the sum of share times same-SLO throughput. The model assumes GPU-time additivity and ignores interference between classes, so it is a screening tool, not a forecast. *Table 3* applies it to AMD's concurrency-16 figures with request-count shares and 512 output tokens per request (a **Hypothetical Example**).

Mix by request count (1,024 / 4,096 / 40,960 tokens)	Fleet tok/s per GPU, off	Fleet tok/s per GPU, on	CP on vs off
95% / 4% / 1%	1,360	1,207	-11.3%
90% / 5% / 5%	1,944	1,895	-2.5%
80% / 10% / 10%	2,437	2,591	+6.3%
50% / 20% / 30%	3,258	4,083	+25.3%

The inputs behind *Table 3* are the concurrency-16 throughputs in *Table 2* and the token counts per request (1,536, 4,608 and 41,472). With only 1,024-token and 40,960-token classes, the crossover sits at about **7%** of requests being long. The lesson is that a long-prompt gain of 43% to 54% does not automatically carry to the fleet: at a 1% long share the fleet loses **11.3%**, even though that 1% of requests carries about a fifth of all tokens. At a 5% long share the result sits inside the 8.4% spread, so only the extremes of this table are decisions; the middle needs a repeat run on real traffic.

Adoption Threshold and Rollback for Mixed Short and Long Traffic

Defining a threshold above repeat noise

AMD's own noise estimate is a spread of about 8.4% between repeat measurements of one 4,096-token configuration, which it treats as a noise floor of roughly 8% for a sweep point (rocm.blogs.amd.com). Another pair of repeat runs differed by about 1%, so the floor is point-dependent and should be measured, not borrowed. A defensible procedure:

1. **Run at least five repeats** of the CP-off and CP-on arms at each prompt-length class and concurrency, each on a freshly started server, as AMD did.

2. **Compute the 95% repeat spread** as the t-based half-width of the mean (for five repeats, the Student t multiplier for four degrees of freedom is **2.776**) divided by the mean, per point.
3. **Set the threshold** T as the larger of 8.4% and your own 95% spread at that point.
4. **Adopt CP for a class** only if the same-SLO throughput change exceeds +T, p95 TTFT improves by more than T, and TPOT or end-to-end latency stays within the product's budget at the real output length.
5. **Reject CP for a class** if throughput falls by more than T. If the change is inside T, treat it as no effect and keep CP off for simplicity.
6. **Check the KV ceiling.** AMD measured a useful ceiling of about **24** concurrent requests at 40K and about **16** at 60K, and the CP-on pool is smaller by 199,744 tokens, which is about **3.2** requests of 61,952 tokens each.

Applied to AMD's data without re-measurement, this gives a provisional map: CP off below roughly 4K tokens when throughput is the goal, CP on from roughly 40K tokens up, and a measurement-required band in between. AMD's deployment checklist agrees on the 40K figure and keeps the interleave strategy because the zigzag strategy did not start on its build. SGLang's DeepSeek-V3.2 documentation agrees that zigzag is temporarily unavailable for DeepSeek V3.2 and the GLM-5 family (docs.sglang.io), and it documents CP enablement for CUDA (docs.sglang.io), not for HIP.

Factors the evidence cannot settle

Several items remain unverified and should be treated as test requirements rather than assumptions:

- **Other attention layouts and longer decodes:** the mechanism depends on a sparse indexer with a 2,048-key budget, and AMD measured 512 output tokens only; dense-attention or different top-k models, and longer outputs, may cross over elsewhere. No p95 exists in the public data either.
- **Output quality:** AMD reports GSM8K 10-shot in the same 0.93 to 0.95 band on both arms and concludes CP does not change accuracy (rocm.blogs.amd.com). One benchmark is a screen; run your own task evaluations at your longest prompts.
- **Production caching:** benchmarks disabled prefix caching, and AMD advises re-enabling radix cache for production. Cache hits shorten effective prefill and move the crossover. Red Hat identifies the prefix-cache hit rate as the primary factor in cost per million tokens for prefill-dominated shapes (developers.redhat.com).

Rollback for mixed short and long traffic

The CP switch is a launch-time flag pair, and AMD's arms were separate server launches. A rollback is therefore a restart, and a mixed-traffic deployment should plan for it before enabling the feature. Restart cost is modest on this hardware: SGLang's notes report GLM-5.2 at TP4 on four MI355X GPUs loading in 40.4 s instead of 505.7 s (github.com). Three patterns are available:

- **Full rollback:** drop `--enable-prefill-cp --cp-strategy interleave` and restart. AMD's fallback for short prompts is to run CP off and raise the concurrency cap so the server uses the KV pool the default leaves idle.

- **Split pools by prompt length:** keep one CP-on group for requests above the threshold and CP-off groups for the rest, with a router that classifies by token count. SGLang's documentation notes that attention data parallelism can coexist with CP, with tp_size divided by attn_dp_size CP ranks in each group (docs.sglang.io). This turns the traffic-weighted model into a capacity-planning input.
- **Prefill and decode disaggregation:** AMD's Infera recipes show a disaggregated GLM-5.2 option with an RDMA KV handoff between two nodes and no TCP fallback (rocm.docs.amd.com). Red Hat advises disaggregating only when the measured prefill and decode imbalance justifies the network transfer hop (developers.redhat.com).

Write the rollback triggers down in advance: a p95 TTFT regression above T for any class, a retraction or preemption rate above baseline, sustained xGMI bandwidth near its ceiling during prefill, or a quality-evaluation drop beyond the benchmark's standard error.

Implications and Future Directions

Three developments could change the answer within months. The first is software support. SGLang's release stream moves quickly, with **v0.5.23** the latest release on the fetched page and ROCm 10 images published for MI35x (github.com), and the v0.5.20 note about unported platforms (github.com) reads as current scope, not a permanent limit. A ported prefill CP in a released image would remove the pinned-branch risk, but would invalidate prior numbers until re-measured. The second is memory. SGLang's v0.5.16 notes describe a DSA cache layer split under prefill CP that shards KV and indexer cache layers across CP ranks, reducing per-rank KV memory by about **74%** in a GLM-5.2-FP8 example (github.com). AMD says the equivalent option is off by default and supported only under prefill and decode disaggregation, and that it plans to measure it. If it applies on MI355X, it would rewrite the 199,744-token penalty that currently sets the price of the split. The third is the vLLM path, where ROCm prefill CP exists but is limited to a different model and has feature exclusions (docs.vllm.ai).

For procurement, the finding is a sizing input, not a purchasing argument. Fleets dominated by prompts above 40K tokens may need fewer GPUs per unit of throughput with CP on, while fleets dominated by chat-length prompts may need more than a CP-assuming plan counts. Because the crossover depends on the model's sparse-attention budget, a model change or a ROCm major version change is a trigger to re-run the matrix. An independent advisor's value here is in making that matrix repeatable and its threshold auditable.

The open verification list is short: merged status and released support for the AMD enablement, the exact model revision, other attention layouts and longer decodes, p95 under production concurrency, all-gather counters and ROCm 10 results. None can be settled from the public record at this date.

Frequently Asked Questions (FAQs)

When should an operator use context parallelism for LLM inference?

Use it when long prompts dominate prefill cost and the first token is slow, since attention cost grows with the square of prompt length (developers.redhat.com). In AMD's MI355X test the sign of the throughput effect depended on prompt length, with losses at 1,024 tokens and gains of 43% to 54% at 40,960 and 61,440

tokens from concurrency 16 (see *Table 1*). Outside that regime, measure first.

Does context parallelism reduce time to first token on MI355X?

For long prompts, yes in AMD's data: median TTFT at concurrency 1 fell from 2.52 s to 1.70 s at 40,960 tokens and from 3.95 s to 2.58 s at 61,440 tokens. At 1,024 tokens it rose from 417 ms to 452 ms at concurrency 16.

Is ring attention what AMD used?

No. AMD's configuration gathers keys and values inside each layer and never gathers queries. Ring Attention and Meta's pass-KV and pass-Q variants are related methods that overlap communication with computation (arxiv.org).

Does it work in vLLM on ROCm?

vLLM documents eager prefill CP on ROCm for DeepSeek-V4.1-Flash with TP=2 and PCP=2, with exclusions including speculative decoding and prefix caching (docs.vllm.ai). It is not a documented GLM-5.2 path in the pages reviewed.

Can the result be trusted on ROCm 10?

Not yet. AMD tested ROCm 7.2.0, and ROCm 10.0.0 was released on 2026-08-26 (rocm.docs.amd.com). Re-run the matrix.

What is the single most important check before enabling it?

Compare same-SLO throughput and p95 TTFT per prompt-length class against a threshold set above your own repeat spread, and compute the traffic-weighted result. A 1% share of long requests was not enough to offset the short-prompt penalty in the model above.

Conclusion

Prefill context parallelism on a four-GPU MI355X tensor-parallel group is best understood as a conditional optimization. AMD's measurements show a clear loss at 1,024 tokens, an indistinguishable result at 4,096 tokens, and a large gain at 40,960 tokens and above, produced by a mechanism that divides indexer work which grows with prompt length while charging a per-token decode penalty and a smaller KV pool. The enabling software was a pinned branch whose upstream pull request is now closed without a merge, and the released SGLang notes reject the feature on HIP, so the first adoption step is confirming that your exact stack runs it at all.

The decision therefore reduces to a worksheet: measure on and off per prompt-length class with enough repeats to know your noise, set the threshold above it, compute the traffic-weighted outcome, check KV headroom and output length, and plan the rollback before the change. The result for one checkpoint on ROCm 7.2.0 is a hypothesis for any other fleet. Operators who run the matrix on their own model, ROCm version and traffic will know which side of the crossover they occupy.

Source links

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

1. https://rocm.blogs.amd.com/artificial-intelligence/glm52-mxfp4-mi355/README.html
2. https://github.com/sgl-project/sglang/pull/32175
3. https://github.com/sgl-project/sglang/releases
4. https://docs.vllm.ai/en/latest/serving/context_parallel_deployment
5. https://rocm.docs.amd.com/en/docs-10.0.0/about/release-notes.html
6. https://developers.redhat.com/articles/2026/06/22/designing-distributed-ai-inference-core-concepts-and-scaling-dimensions
7. https://docs.sglang.io/cookbook/autoregressive/GLM/GLM-5
8. https://docs.nvidia.com/megatron-core/developer-guide/0.20.0/user-guide/features/context_parallel.html
9. https://arxiv.org/abs/2310.01889
10. https://arxiv.org/abs/2411.01783
11. https://arxiv.org/abs/2309.14509
12. https://www.amd.com/en/products/accelerators/instinct/mi350/mi355x.html
13. https://www.amd.com/content/dam/amd/en/documents/instinct-tech-docs/white-papers/amd-cdna-4-architecture-whitepaper.pdf
14. https://arxiv.org/abs/2512.02556
15. https://huggingface.co/amd/GLM-5.2-MXFP4
16. https://docs.sglang.io/basic_usage/deepseek_v32.html
17. https://developers.redhat.com/articles/2026/06/26/deploying-distributed-ai-inference-blueprints-troubleshooting
18. https://docs.sglang.io/references/production_metrics.html
19. https://docs.vllm.ai/en/latest/usage/metrics
20. https://prometheus.io/docs/practices/histograms
21. https://rocm.docs.amd.com/projects/rocprofiler-systems/en/latest/how-to/xgmi-pcie-sdma-sampling.html
22. https://rocm.docs.amd.com/projects/amdsmi/en/latest/how-to/amdsmi-cli-tool.html
23. https://www.wafer.ai/blog/glm52-amd
24. https://rocm.docs.amd.com/projects/infera/en/latest/recipes/glm5.2.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.