

CHUNDER: Enabling Million-Token Qwen3 Serving on AMD ROCm

David Lemphers^{*†}

Maincode

dave@maincode.com

Maxwell Twelftree^{*}

Maincode

max@maincode.com

An-chi He

Maincode

anchi@maincode.com

Yue Yang[†]

Maincode

yue@maincode.com

Abstract

Million-token context serving is becoming a critical requirement for industry LLM systems for real deployments. However, practical 1M-context serving for open-source models has remained heavily CUDA-centric, largely because the required long-context runtime features are not fully available on AMD ROCm. With AMD GPUs now reported at around 7% market share (Silicon Analysts Research, 2026), this gap matters. We address it by enabling 1M-token Qwen3 serving on AMD GPUs with ROCm. We present CHUNDER, a runtime stack that enables million-token serving for the widely used open-source Qwen3 family on AMD ROCm. CHUNDER achieves this by intercepting vLLM’s attention routing at runtime and redirecting queries, keys, and values to JIT-compiled Triton Dual Chunk Attention (DCA) kernels, without forking the serving engine. We then harden the surrounding runtime to make full 1M-token execution practical: DCA-aware paged metadata, tiled single-request chunked prefill, MoE token chunking that reduces per-rank transient memory from approximately 91.6 GiB to 10.6 GiB, and FP8 GEMMs with hardened hipBLASLt kernel selection. We evaluate CHUNDER across multiple million-token context tasks and show consistent serving performance and exact retrieval correctness. To our knowledge, this is the first reported end-to-end Qwen3 million-token serving path on AMD ROCm integrated with vLLM.

1 Introduction

Million-token context has moved from a research demonstration to a production requirement. Modern LLM applications increasingly require retrieval over entire knowledge bases, long agent tool-use trajectories, repository-scale code understanding, and book-length reasoning. Open-weight models

now ship explicit recipes for this regime: for example, Qwen3 extends its instruction-tuned Mixture-of-Experts (MoE) checkpoints from a 262,144-token native context to 1,010,000 tokens via Dual-Chunk Attention (DCA) (Yang et al., 2025a,b; An et al., 2024). For operators, the binding constraint is no longer whether a million-token model exists, but whether it can be served at the required context length, latency, reliability, and cost on the hardware available for deployment.

In practice, that choice is constrained by software availability. Efficient long-context serving paths have largely been implemented for NVIDIA/CUDA, so AMD ROCm deployments cannot serve requests beyond the native context length without additional runtime support. The consequence is operational: fleets of AMD MI355X accelerators, with 288 GB of HBM per GPU and attractive price, power, and supply properties, cannot serve these million-token workloads through the existing open-source path. Removing this wall is not a one-line patch, but a cascade of coupled systems problems. DCA must first be reproduced without the missing C++ kernels. Even after attention is restored, a single million-token request stresses the entire runtime: MoE feed-forward layers, rather than attention, dominate transient memory; prefill exceeds the engine’s default worker timeouts; and FP8 matrix multiplications can fall through to unstable library paths on gfx950. Each fix exposes the next bottleneck, and every optimization must preserve exact retrieval correctness: a near-miss answer is a silent product regression, not a numerical rounding error. We present CHUNDER, a runtime stack that serves Qwen3 at its full million-token context on a single AMD MI355X node. Rather than forking vLLM, CHUNDER intercepts the engine’s attention routing at runtime and redirects queries, keys, and values to JIT-compiled Triton (Tillet et al., 2019) DCA kernels. Around this mechanism, we harden the full serving path with

^{*}Equal Contribution

[†]Corresponding authors

DCA-aware paged metadata, tiled single-request chunked prefill, MoE token chunking, FP8 GEMM kernel selection, and reproducible image packaging. Every change is gated by a validation ladder that requires exact retrieval, and every deployable configuration is frozen into an immutable, content-addressed image so that the reported result and the image that produced it are the same artifact. Our contributions are:

- **Runtime DCA on ROCm and vLLM V1.** We implement JIT-compiled Triton DCA kernels and inject them into an unmodified vLLM runtime, enabling long-context serving on AMD GPUs and restoring DCA support for the V1 engine.
- **An end-to-end million-token serving stack.** We identify and resolve the coupled bottlenecks required to make a single 1M-token request fit in HBM and run at practical speed, including DCA-aware metadata, chunked prefill, MoE token chunking, timeout hardening, and FP8 GEMM stabilization.
- **Evidence on production hardware.** On a single $8 \times$ MI355X node, we demonstrate exact 1M-token retrieval for Qwen3-235B and Qwen3-30B, validate against capability baselines and a reference-algorithm oracle, evaluate a ten-task RULER-style suite from 32K to 1M tokens with depth sweeps at 32K and 131K, and report throughput, memory, and per-component cost. We release the runtime patches, tuned configurations, and image build recipes.

We want to emphasize here that our goal is not to show that ROCm is faster than CUDA. The mature CUDA/vLLM stacks remain better optimized, and our measured performance reflects this gap. Rather, this work demonstrates that the 1M-token serving capability itself can be brought to ROCm with Qwen3: we restore the missing long-context runtime path needed to serve Qwen3 at million-token scale on AMD GPUs. Given AMD’s growing deployment footprint, this is an important systems milestone because it turns 1M-context serving from a CUDA-only capability into a portable option for ROCm-based infrastructure.

2 Related Work

Qwen3 and Dual-Chunk Attention. Qwen3 is one of the most widely used open-weight model families to publish an explicit million-token serving recipe. Qwen3-235B-A22B-Instruct-2507 (94 layers, 64 query / 4 key-value heads, 128 experts

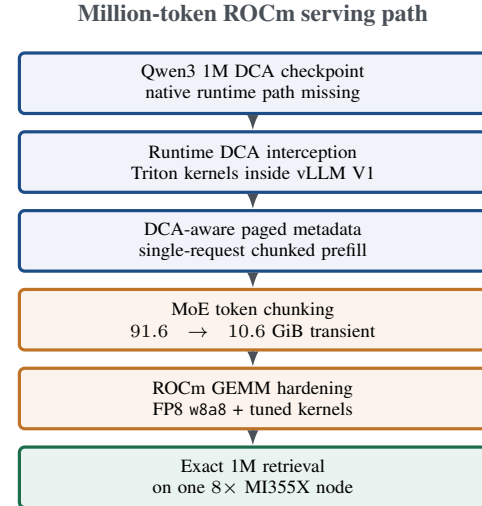


Figure 1: **CHUNDER at a glance.** The dependency graph that turns Qwen3’s published 1M DCA recipe into a ROCm serving path: runtime DCA interception, DCA-aware paged metadata, chunked prefill, MoE token chunking, ROCm GEMM hardening, and exact 1M retrieval on one $8 \times$ MI355X node.

with top-8 routing, 22 B active parameters) ships with a native context of 262,144 tokens, extendable to 1,010,000 via DCA; the smaller Qwen3-30B-A3B-Instruct-2507 (48 layers, 32/4 heads) publishes the same 1M Dual-Chunk Attention(DCA) configuration, together with sparse long-context inference support (Yang et al., 2025a; An et al., 2024; Yang et al., 2025b; Jiang et al., 2024). This makes Qwen3 a natural target for studying production long-context serving: the model-side recipe exists, but the serving system must still implement the DCA semantics, memory layout, prefill strategy, and correctness checks needed to make the advertised context length usable in practice.

Serving support in vLLM and SGLang. Long sequences are usually served by adding devices: Ring/Striped Attention and DeepSpeed-Ulysses shard the sequence (Liu et al., 2023; Brandon et al., 2023; Jacobs et al., 2023), Infinite-LLM distributes attention and KV cache (Lin et al., 2024), and Splitwise/DistServe/Mooncake disaggregate prefill from decode (Patel et al., 2024; Zhong et al., 2024; Qin et al., 2024). DCA is the complement, bounding the attention working set so one million-token request fits a fixed single-node TP allocation. Existing Qwen instructions expose million-token launch paths for mainstream serving engines with DCA, but these paths are primarily aligned with CUDA-oriented

backend support. In vLLM, Qwen 1M serving relies on DUAL_CHUNK_FLASH_ATTN while forcing the legacy V0 engine through `VLLM_USE_V1=0`; in SGLang, the same configuration is launched with `-attention-backend dual_chunk_flash_attn`, `-context-length 1010000`, tensor parallelism, and chunked prefill (Kwon et al., 2023; Zheng et al., 2024; Yang et al., 2025a). These recipes show that Qwen-family million-token serving has an established path in the NVIDIA/CUDA ecosystem, reflecting the broader reality that production long-context inference is still largely optimized, validated, and deployed first on NVIDIA hardware.

ROCm support for million-token serving.

ROCm support in modern serving frameworks has improved, and both vLLM and SGLang provide AMD GPU deployment paths for Qwen-family models (Kwon et al., 2023; Yang et al., 2025a). However, public upstream support stops short of a complete Qwen3 million-token DCA path on ROCm: existing ROCm recipes mainly cover native or shorter long-context configurations, while Qwen’s 1M recipe depends on `dual_chunk_flash_attn`, a backend path documented through CUDA-oriented serving instructions (Yang et al., 2025a; An et al., 2024; Yang et al., 2025b). Thus, the gap is not that ROCm cannot run LLM serving, but that the model-specific runtime path required for Qwen3 1M inference—DCA routing, paged-KV integration, chunked prefill, and stable long-context execution—was missing. CHUNDER closes this gap by implementing and validating DCA-backed million-token Qwen3 serving on AMD ROCm.

3 CHUNDER Design

Our goal is a deployable 1M-context lane for single long requests, integrated into an OpenAI-compatible vLLM endpoint of Qwen3 MoE models on a single $8 \times$ AMD MI355X (gfx950) node. We present CHUNDER in dependency order: each component removes the bottleneck exposed by the previous one. The invariant throughout is exact retrieval correctness, enforced by the validation ladder in Section 3.2. Thus, CHUNDER is not a standalone attention kernel, but an end-to-end runtime stack that makes DCA-backed 1M-token serving fit, execute, and remain reproducible on ROCm.

3.1 End-to-end 1M DCA serving path in CHUNDER

Step 1. Runtime attention interception Rather than fork vLLM’s C++ attention backend or wait on upstream support, CHUNDER intercepts the engine’s attention routing at runtime and redirects $Q/K/V$ to its own `@triton.jit` kernels. A shim attention implementation un-stacks the five RoPE query variants, reuses vLLM’s existing Triton paged-KV metadata for block layout, and writes the merged result back into the engine’s output view. Because the kernels are JIT Triton rather than ahead-of-time C++, they recompile between generations with the model resident, and a standalone bare-kernel harness validates numerics against a reference before any engine-level run. Interception is per-process environment configuration; the only persistent change to vLLM is a small, content-hashed runtime patch (Section 3.2) admitting the DCA keyword arguments and exposing the routing hook.

Step 2. DCA-aware paged metadata and decode

The naive V1 paged-attention metadata assumes one query row per token, which collides with DCA’s five-fold expanded query; The difficulty is that DCA expands each logical token into multiple position-remapped query streams while retaining a single physical KV stream; therefore the metadata must map logical token positions, DCA chunk coordinates, and paged KV blocks consistently across prefill and decode. we therefore build DCA-aware paged metadata so that cached decode is correct against the paged KV cache. This yields two execution modes: **paged DCA** serves cached decode and requests that fit within a single DCA chunk; **tiled paged DCA chunked prefill** (Section 3.1) serves a single long request. Batching is *fail-closed* for correctness: native paged attention is not DCA-correct across chunk boundaries and multi-sequence long DCA is unimplemented, so the backend rejects any batch with more than one sequence longer than c_{len} , and production long-context runs use `max_num_seqs=1`.

Step 3. Single-request chunked prefill

A single 1M-token request cannot be prefilled in one step, so tiled paged DCA chunked prefill schedules it in `max_num_batched_tokens`-sized spans, with split-KV reduction across the million-token key range. Correctness across the chunk boundary requires the scheduler span to include the estimation window

for every touched DCA chunk; when it does not, routing falls back to the dense path. We pre-warm the exact single-request prefill milestones (derived from context length, span size, and the DCA chunk geometry) so runtime slices reuse the warmup’s scratch strides with no inference-time JIT compilation, and raise the engine’s default 300 s worker timeout for the long-context lane. Span and block sizes are swept in Section 4.3; production uses $q=16384$, $block_m=16$ (Table 5).

Sparse prefill acceleration: The dense path above carries our primary 1M claims; this subsection is an *optional* accelerator off that working path. DCA’s critical variants admit a vertical-and-slash sparse approximation (MInference-style) (Jiang et al., 2024) with per-head budgets from the model’s published 1M configuration; the estimator matches the dense estimator exactly on every touched chunk, the Triton consumer matches a dense-KV oracle within bf16 tolerance, and both merge into the same online-softmax accumulator as the dense partials (Appendix B). The sparse path is a material throughput win at equal retrieval correctness (Section 4.3), but at official per-head budgets the index-build and consumer costs remain above the seconds-level target, so it ships as a validated but *experimental* image channel.

Step 4. MoE execution and activation memory

At million-token prefill the MoE feed-forward, not attention, dominates transient memory: an unchunked 1.01M-token MoE step materializes roughly 91.6 GiB of temporary and output activations per rank. We chunk the MoE token dimension (default 32768), reducing the per-rank transient to ≈ 10.6 GiB, the enabling factor for fitting 1M prefill within HBM at $TP=8$. The Qwen3 expert GEMM shape on gfx950 ($E=128$, $N=192$, fp8_w8a8) has no tuned Triton configuration in vLLM, and nearby NVIDIA or different- N AMD configurations are incorrect, so we benchmark the exact shape on MI355X and load the merged measurements as a validated startup artifact, gated on unchanged retrieval and improved prefill time.

Step 5. Precision and ROCm GEMM hardening

Weights are served in dynamic FP8 (w8a8); the online-LSE merge and attention reductions remain in higher precision. The KV cache stays BF16 on the correctness lane: FP8 KV runs without faults but is not yet bit-faithful for cached decode (a near-miss on the decode gate), so it is held behind a decode smoke gate and never used for retrieval

claims. (An emerging FP8-shuffle KV lane with fused AITER kernels does pass exact retrieval at near-million scale) Per-lane GPU memory utilization is set to leave headroom for the million-token KV cache and the chunked-MoE scratch (See Appendix B for details).

The fp8 GEMMs on gfx950 fall through to hipBLASLt/rocBLAS, where missing FP8/FNUZ latency-table entries degrade kernel selection with repeated “latency not found” fallbacks. We harden this at two levels – TunableOp records and replays tuned overrides for the served shapes, and we patch hipBLASLt’s hardware tables with deterministic gfx950 entries (Appendix B) – and expose the ROCm AITER fused kernels behind per-component toggles for bisection, keeping AITER MoE *off* for headline claims: its default kernels reject the Qwen3 $N=192$ expert shape and do not beat the tuned Triton MoE lane.

3.2 Validation and reproducibility

Retrieval gate: Correctness is a RULER-style NIAH task (Hsieh et al., 2024): a seven-digit needle (7402913) at a fixed deep position (token 900,000 of a 1M-token context), passing only on exact-string retrieval, with the answer length fixing the number of decode steps. Because HTTP submission re-tokenizes raw text to a slightly different server-side count, the API lane computes the exact server-side token count and warms the scheduler to it (Appendix B).

Reference-algorithm oracle: The removed V0 DCA backend is the only authoritative specification of the algorithm, so we maintain a faithful executable port of it (an SDPA-leaf reference) and a library of ground-truth $Q/K/V$ /output tensors captured from instrumented runs. Every Triton kernel must match this oracle in the bare-kernel harness before any engine-level gate.

Fast validation ladder: Every optimization climbs a five-rung ladder before it is granted a full 1M slot, so that load, initialization, routing, and ABI failures surface in seconds rather than after a 25-minute prefill (Table 8, Appendix B). See Appendix A for full details and link.

4 Experiments and Result Discussion

4.1 Experiments Setup

We evaluate CHUNDER on *capability* (serve at all, where stock engines cannot), *quality* (exact retrieval, matching the V0 reference), and *efficiency*

(throughput, latency, memory, per-component cost). Unless noted, every configuration is pinned to an immutable image.

Hardware. The target platform is a single AMD MI355X node: 8 GPUs (CDNA4, gfx950, 288 GB HBM3E each; rootless podman, ROCm runtime); all results are produced here.

Models. We evaluate the two open-weight Qwen3 MoE checkpoints that publish a 1M DCA configuration (config_1m.json): **Qwen3-235B-A22B-Instruct-2507** (headline; TP=8) and **Qwen3-30B-A3B-Instruct-2507** (fast lane; TP=4), which runs the identical DCA recipe, showing cross-scale generality. Qwen3-8B (TP=1) is a backend/route validation vehicle only (no config_1m.json; never a quality claim). Decoding is deterministic (greedy, temperature=0); the answer length fixes the decode steps per gate.

Serving stack. CHUNDER runs on PyTorch 2.10, Triton 3.6, and ROCm 7.2.4 (Python 3.12), injecting JIT-compiled Triton DCA kernels into vLLM (a pinned 0.22.1rc pre-release source revision, content-hashed per image). The frozen production configuration is summarized in Table 5 (Appendix A.1).

4.2 Experiments and Results

4.2.1 Performance-Oriented Experiments for 1M-Context Serving

Retrieval Quality : We measure exact recall with a needle-in-a-haystack (NIAH) protocol: a unique numeric needle is embedded in a synthetic token haystack of controlled length, and we score exact match of the model’s generation. To show that quality is not the artifact of a single lucky needle position, we run a 5×3 grid—needle depth $\{0, 25, 50, 75, 100\}\%$ crossed with context length $\{131K, 512K, 1M\}$ (Table 1). All fifteen cells retrieve the needle exactly; in particular there is no lost-in-the-middle degradation at 1M, where the needle is recovered at every depth.

Table 1: Needle-in-a-haystack exact recall for Qwen3-235B on a single MI355X node; ✓ = exact retrieval. All 15 cells pass.

Needle depth	131K	512K	1M
0%	✓	✓	✓
25%	✓	✓	✓
50%	✓	✓	✓
75%	✓	✓	✓
100%	✓	✓	✓

RULER Benchmarks: We also run a *ten-task RULER-style suite* (Hsieh et al., 2024) through the live OpenAI endpoint at $\{32K, 131K, 512K, 1M\}$ tokens – the tasks of Table 2: single-needle, multi-key, multi-value, and multi-query retrieval, multi-hop variable tracking, and word aggregation. Tasks are token-budgeted with deterministic seeds and scored by exact string-match recall, submitted sequentially. Table 2 reports the suite. Every completed *serving-correctness* cell is perfect: all three single-needle variants at every length including 1M, both multi-key variants at every length, multi-query retrieval through 512K, and the full depth sweep at 32K and 131K – there the needle is recovered exactly from the very start to the final token of the context. At 512K and 1M, depth coverage is currently the suite’s single-needle cells plus the 90%-depth headline gate (Table 3); The request-latency row doubles as the serving latency-vs-length curve; the 1M samples ($\approx 1,460$ s through the live endpoint) independently reproduce the offline harness wall-clock of Table 3 to within 1% on a different node. The one sub-perfect cell (MV-NIAH at 512K, 0.75) is a model-capability effect, not a serving defect: seeded generation lets us place every missed value at relative depth 0.59–0.89, interleaved with retrieved values and uncorrelated with the DCA chunk boundary, while the single-needle sweep is perfect at those depths (Appendix B). We do not claim performance superiority over mature CUDA stacks. Our claim is capability portability: the missing Qwen3 1M DCA serving path can be implemented, validated, and packaged on ROCm without forking the serving engine

4.2.2 Serving Characterization under Concurrent Workloads

The fail-closed policy applies only to fresh cross-chunk prefill. We therefore evaluate multi-user serving in the validated sub-chunk regime using the closed-loop 1c_8k workload. All evaluated requests retrieve correctly, with success rate 1.0. Because these retrieval requests emit only approximately seven output tokens, we report aggregate total-token throughput rather than output-token goodput.

Both configurations increase aggregate throughput by approximately $3\times$ before saturation. The 235B lane approaches its plateau near $C = 16$, whereas the 30B lane continues scaling to approximately $C = 32$ and reaches about $2.1\times$ the 235B request rate. With 32K inputs, saturation occurs

Task	B200 NVIDIA CUDA								MI355X AMD ROCm							
	Qwen3-30B-A3B / TP4				Qwen3-235B-A22B / TP8				Qwen3-30B-A3B / TP4				Qwen3-235B-A22B / TP8			
	32K	131K	512K	1M	32K	131K	512K	1M	32K	131K	512K	1M	32K	131K	512K	1M
S-NIAH (noise)	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
S-NIAH (prose)	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
S-NIAH (uuid)	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
MK-NIAH (numbers)	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
MK-NIAH (uuids)	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
MV-NIAH	1.00	0.88	0.88	0.62	1.00	1.00	0.75	0.88	1.00	1.00	0.83	0.75	1.00	1.00	0.75	0.88
MQ-NIAH	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
VT (5-hop)	0.60	0.20	0.20	0.20	1.00	0.80	0.70	0.50	0.20	0.47	0.40	0.40	1.00	1.00	0.67	0.80
CWE	0.00	0.00	0.00	0.00	0.90	0.95	0.90	1.00	0.57	0.37	0.30	0.00	1.00	1.00	0.93	0.90
FWE	0.50	0.00	0.00	0.00	1.00	1.00	1.00	1.00	0.22	0.33	0.00	0.00	1.00	1.00	1.00	1.00
Depth sweep	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Mean latency (s)	21.2	33.5	69.8	238.4	26.0	61.1	152.9	373.6	3.0	15.4	211	742	3.5	30	415	1,460

Table 2: RULER-style long-context quality across the B200 vLLM DCA lane and the live endpoint.

Model	Lane	Context (tok)	Wall (s)	Prefill tok/s	Retrieval
30B TP4	offline, q16384	1,000,024	739.5	1,352	exact
30B TP4	API, chunked q16384	998,871	740.5	–	exact
30B TP4	API, split-KV warm	999,412	458.7	–	exact
235B TP8	offline, q16384	1,000,024	1,454.2	688	exact
235B TP8	API, chunked q16384	999,981	1,458.1	–	exact
235B TP8	API, split-KV warm	999,981	901.1	–	exact
30B TP4	API, 130K cross-chunk	≈131K	10.8	–	exact
235B TP8	API, 130K cross-chunk	≈131K	21.5	–	exact

Table 3: Headline retrieval gates on one $8 \times$ MI355X node (dense paged DCA, FP8 w8a8 weights, BF16 KV, tuned MoE). All gates require exact retrieval of the seven-digit needle at token 900,000; offline and API token counts differ because HTTP submission re-tokenizes the raw prompt.

C	235B, TP8		30B, TP4	
	Tok/s	TPOT ₉₅ (ms/tok)	Tok/s	TPOT ₉₅ (ms/tok)
1	6,969	110	13,937	57
2	10,658	162	20,906	82
4	14,552	273	29,924	132
8	17,626	501	38,532	232
16	20,496	854	43,656	399
32	20,906	923	44,681	433

Table 4: Closed-loop concurrent serving on 1c_8k. Throughput includes prompt and generated tokens; every operating point preserves exact retrieval.

earlier: the 235B lane peaks at $C = 4$ with 13,110 total tok/s, and the 30B lane peaks at $C = 8$ with 25,400 total tok/s. Beyond these points, additional concurrency primarily increases queueing latency.

Appendix G reports complete request-throughput results, open-loop Poisson arrivals, p50/p95/p99 TTFT, TPOT, and end-to-end distributions, sustained memory behaviour, and output-length and decode-batching sweeps.

4.3 Efficiency and ablations

We ablate the runtime components that determine whether 1M-token serving is feasible and performant under the exact-retrieval gate. The study varies four groups of mechanisms: MoE token chunking, prefill span size, DCA kernel block size, and ROCm GEMM hardening. For MoE execution, we compare the unchunked path against token chunks of 32768 and 65536. For chunked prefill, we sweep $q \in \{8192, 16384, 32768\}$ and bounded-window variants. For the DCA kernel, we compare $\text{block_m} = 16$ and $\text{block_m} = 64$. Finally, we evaluate GEMM hardening by enabling the hip-BLASLt table patch and TunableOp override used to avoid pathological FP8/FNUZ kernel selection. Every candidate configuration must pass the same serving-correctness ladder; configurations that fail exact retrieval, overflow memory, or regress full 1M prefill are rejected.

Ablation results. MoE token chunking is the feasibility-critical ablation. Disabling it does not merely slow the system down: it makes 1M prefill

impossible, because an unchunked 1.01M-token MoE step materializes approximately 91.6 GiB of transient activations per rank. Chunking the token dimension reduces this to ≈ 10.6 GiB at the production chunk size of 32768 (≈ 15.7 GiB at 65536), which is the enabling factor that allows the 1M request to fit in HBM at $TP = 8$. For prefill span size, the performance frontier is relatively flat: at 1M-context, $q = 8192$ trails the production setting $q = 16384$ by only about 1% on both lanes, while $q = 32768$ overflows mid-prefill scratch at high KV-cache utilization; bounded-window variants do not improve over 16384 (Appendix B). We therefore use $q = 16384$ as the cleanest operating point. For DCA kernel block size, $block_m = 64$ is faster in the bare-kernel microbenchmark, but it regresses the full 1M serving run: on the 30B model, wall-clock time increases from 739.5 s to 758.9 s, a 2.6% penalty. This illustrates why microbenchmark improvements are not adopted unless they survive the full serving ladder. Finally, GEMM hardening removes the repeated FP8/FNUZ “latency not found” fallbacks in hipBLASLt/rocBLAS and makes all correctness gates warning-clean. We do not claim this as a speedup over a baseline. It just eliminates a pathological kernel-selection path.

Sparse prefill At unchanged exact retrieval, the experimental sparse path reduces 1M wall-clock by 14% on 30B (739.5 $\rightarrow$ 635.1 s; 1,575 tok/s) and by 27% on 235B (1,454.2 $\rightarrow$ 1,068.5 s; 936 tok/s) at $q=16384$, with the 130K gate completing in 15.8 s (30B) and 30.6 s (235B). In isolation the sparse consumer beats the dense split-KV path by 3.8–4.8 $\times$; the gap to the end-to-end delta is the estimator and index-build cost at official per-head budgets, which is why sparse prefill ships as a validated but experimental channel rather than the production default.

5 Conclusion

We presented CHUNDER, an end-to-end runtime stack for million-token Qwen3 serving on AMD ROCm. By adding a V1-compatible DCA execution path through runtime attention interception, DCA-aware paged metadata, tiled chunked prefill, MoE token chunking, and ROCm GEMM hardening, CHUNDER makes full 1M-context inference fit, run, and preserve exact retrieval on a single 8 $\times$ MI355X node. Our results show that the million-token barrier is a serving-stack problem rather than an AMD hardware limitation, and demonstrate a practical path for long-context deployment for

Qwen3 or DCA based model with AMD.

Limitations

CHUNDER is designed and evaluated for Qwen3’s published Dual-Chunk Attention long-context recipe. The runtime interception mechanism is reusable, but it depends on vLLM’s internal attention-backend and paged-KV interfaces and must therefore be repinned and revalidated for each supported release. The same DCA kernels serve Qwen3-30B and Qwen3-235B unchanged, while new MoE expert shapes require GEMM retuning. Non-DCA long-context architectures require their own attention kernels. The single-active-sequence restriction applies specifically to fresh cross-chunk prefill. Multi-sequence serving has been validated in the sub-chunk, shared-prefix, and cached-decode regimes, but CHUNDER does not yet support simultaneous fresh million-token prefill requests. The full-context lane is intended for asynchronous analytical workloads rather than interactive chat. Finally, the sustained-load experiment shows no observable resident-VRAM drift over the measured 50-minute interval, but it does not establish the absence of all allocator-fragmentation modes over arbitrarily long production deployments. As in the original evaluation, our claim is capability portability rather than performance superiority over mature CUDA serving stacks.

References

- Chenxin An, Fei Huang, Jun Zhang, Shansan Gong, Xipeng Qiu, Chang Zhou, and Lingpeng Kong. 2024. Training-free long-context scaling of large language models. *arXiv preprint arXiv:2402.17463*. ICML 2024.
- William Brandon, Aniruddha Nrusingha, Kevin Qian, Zachary Ankner, Tian Jin, Zhiye Song, and Jonathan Ragan-Kelley. 2023. Striped attention: Faster ring attention for causal transformers. *arXiv preprint arXiv:2311.09431*.
- Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. 2024. RULER: What’s the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*. COLM 2024.
- Sam Ade Jacobs, Masahiro Tanaka, Chengming Zhang, Minjia Zhang, Shuaiwen Leon Song, Samyam Rajbhandari, and Yuxiong He. 2023. DeepSpeed ulysses: System optimizations for enabling training of extreme long sequence transformer models. *arXiv preprint arXiv:2309.14509*.

- Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H. Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2024. MInference 1.0: Accelerating pre-filling for long-context LLMs via dynamic sparse attention. *arXiv preprint arXiv:2407.02490*. NeurIPS 2024.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*, pages 611–626.
- Bin Lin, Chen Zhang, Tao Peng, Hanyu Zhao, Wencong Xiao, Minmin Sun, Anmin Liu, Zhipeng Zhang, Lanbo Li, Xiafei Qiu, Shen Li, Zhigang Ji, Tao Xie, Yong Li, and Wei Lin. 2024. Infinite-LLM: Efficient LLM service for long context with DistAttention and distributed KVCache. *arXiv preprint arXiv:2401.02669*.
- Hao Liu, Matei Zaharia, and Pieter Abbeel. 2023. Ring attention with blockwise transformers for near-infinite context. *arXiv preprint arXiv:2310.01889*.
- Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics (TACL)*, 12:157–173.
- Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. 2024. Splitwise: Efficient generative LLM inference using phase splitting. In *Proceedings of the 51st Annual International Symposium on Computer Architecture (ISCA)*.
- Ruoyu Qin, Zheming Li, Weiran He, Mingxing Zhang, Yongwei Wu, Weimin Zheng, and Xinran Xu. 2024. Mooncake: A KVCache-centric disaggregated architecture for LLM serving. *arXiv preprint arXiv:2407.00079*.
- Silicon Analysts Research. 2026. Amd vs nvidia: The ai gpu war in numbers. <https://siliconanalysts.com/analysis/amd-vs-nvidia-ai-gpu-market-share-2026>. Accessed: 2026-06-16.
- Philippe Tillet, H. T. Kung, and David Cox. 2019. Triton: An intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages (MAPL)*, pages 10–19.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.
- An Yang, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoyan Huang, Jiandong Jiang, Jianhong Tu, Jianwei Zhang, Jingren Zhou, and 1 others. 2025b. Qwen2.5-1M technical report. *arXiv preprint arXiv:2501.15383*.
- Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. 2024. SGLang: Efficient execution of structured language model programs. *arXiv preprint arXiv:2312.07104*. NeurIPS 2024.
- Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. 2024. DistServe: Disaggregating prefill and decoding for goodput-optimized large language model serving. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 193–210.

A Reproducibility

The production recipe (Table 5, Appendix A.1) is frozen into immutable, content-addressed container images: each tag pins the ROCm, vLLM-base, and CHUNDER revisions and the SHA-256 of both runtime patches, and a validation manifest is emitted only after that exact image passes the 8K, 130K, and 1M API gates. Tuned configurations, override tables, and patch artifacts are versioned alongside, so any reported number replays from a single image reference plus the model checkpoint.

See <https://github.com/Maincode-Research/Qwen3-1M-ROCM/tree/main> for the whole implementation.

A.1 Production Configuration and Baseline Matrix

Table 5 gives the frozen production configuration pinned per immutable container image; Table 9 the baseline tiers; and Table 7 the workload-to-claim mapping.

B Extended Engineering and Ablation Detail

MoE tuning. The $E=128$, $N=192$ fp8_w8a8 expert shape is benchmarked on MI355X as a batch sweep sharded across the eight GPUs; the merged measurements form the tuned configuration loaded at startup.

GEMM hardening mechanics. At the PyTorch level, TunableOp records the served GEMM shapes on a short production-shaped gate, tunes them offline, and replays a fixed override table; at the library level, we patch hipBLASLt’s Tensile/origami hardware tables to add deterministic gfx950 (and gfx942) FP8/FNUZ entries, rebuild, and preload the library ahead of the system one. The ROCm AITER fused GEMM/RMSNorm/MoE kernels are exposed behind per-component toggles for bisection.

Prefill span-size sweep. At 1M on the 30B lane, $q=8192$ completes in 747.7 s (1,337 tok/s) versus 739.5 s (1,352 tok/s) for the production $q=16384$; on 235B the comparison is 1,469.3 s versus 1,454.2 s. $q=32768$ overflows mid-prefill scratch at high KV utilization; constrained variants with bounded internal q -windows of {8192, 16384, 24576} complete in 748.3/742.2/742.8 s, none beating 16384.

Sparse estimator and consumer detail. An estimator computes the active vertical/slash index set over the scheduled prefill span and a Triton consumer attends only the selected blocks. The estimator exactly matches the dense estimator ($vdiff=sdiff=0$) on every touched chunk; the consumer matches a dense-KV oracle within a bf16 tolerance of 1.95×10^{-3} ; and a GPU top- k compact-index builder feeds the consumer, with overflow falling back to the proven scan-sparse route.

Sparse kernel-level measurements. On representative spans the sparse consumer beats the dense split-KV path by 3.8–4.8 $\times$, and a precomputed-worklist variant reaches 0.28 ms versus 7.43 ms dense on a 32K span.

Server-side retokenization. HTTP submission re-tokenizes raw text to a slightly different server-side count; e.g. a raw 1.1M-character prompt re-tokenizes to 999,981 server tokens on the 235B lane, which is the count the API lane warms the scheduler to.

Memory headroom. GPU memory utilization is set per lane (0.86–0.977) to leave headroom for the million-token KV cache and the chunked-MoE scratch.

Preliminary: FP8-shuffle KV with fused AITER kernels. An emerging lane combines an FP8 KV cache in AITER’s shuffle layout with fused QK-norm/dual-chunk-RoPE/cache-write kernels and an online per-token FP8 AITER MoE path. It passes exact retrieval at 131,810 tokens in 5.9 s (30B) and at 918,156 tokens in 94.7 s (30B) and 195.2 s (235B), roughly 4.6 $\times$ faster than the fastest dense API lane at comparable scale. These numbers are source-tree validated but not yet replayed from a promoted immutable image, so under the artifact discipline of Section 3.2 we report them only here and exclude them from all claims; they indicate the production frontier of this stack is well below the dense numbers in Table 3.

Multi-value 512K depth analysis. The single sub-perfect cell, multi-value retrieval at 512K (0.75: the model returns three of four values for one key on average), is attributable to model capability rather than the serving stack. Because task generation is deterministically seeded, we can reconstruct the exact depth of every missed value: misses sit at relative depths 0.59–0.89, are interleaved with *retrieved* values at neighboring

Component	Production setting
GPUs	8× MI355X (gfx950), TP=8
Stack	ROCm 7.2.4, PyTorch 2.10, Triton 3.6
Attention	dense paged DCA, chunked prefill
Prefill query chunk	max_num_batched_tokens 16384
Batch policy	max_num_seqs 1 (fail-closed)
Block size	block_m=16
Weights / KV	FP8 w8a8 / BF16 KV
MoE	chunk 32768, tuned $E=128$, $N=192$
GEMM	hipBLASLt FP8 patch + TunableOp
GPU memory utilization	0.86–0.977 per lane
Context	1,000,000 tok, needle @ 900,000

Table 5: Frozen production configuration for the 235B 1M lane, pinned per immutable container image. The 30B fast lane is identical with TP=4.

Tier	Baseline	Role
1	stock vLLM + 1M cfg (ROCm)	capability
1	stock vLLM + 1M cfg (B200)	capability
1	stock ROCm vLLM @ 262K	non-DCA ceiling
2	V0 port + captured tensors	oracle
3	CHUNDER minus one component	ablations

Table 6: Baseline tiers. Tier-1 rows establish that no stock engine serves the 1M configuration on either vendor; Tier 2 anchors correctness; Tier 3 attributes efficiency.

Workload	Lane	Claim
NIAH 8K	offline + API	load / routing
NIAH 130K	offline + API	cross-chunk DCA
NIAH 1M	offline + API	headline retrieval
Chunk-size sweep	offline 1M	efficiency frontier
Sparse vs. dense	offline 1M	sparse speedup
MoE chunk on/off	offline 1M	memory feasibility

Table 7: Benchmarks and the claim each supports. All quality gates require exact needle retrieval.

depths, and show no correlation with the 262,144-token DCA chunk boundary; meanwhile the single-needle depth sweep retrieves perfectly at those same depths and lengths. Free recall of several values bound to one key is precisely the regime where long-context models are known to degrade first (Hsieh et al., 2024; Liu et al., 2024); retrieval of any *individually addressed* item remains exact.

C Baselines and comparison protocol

Our evaluation separates three questions that are easy to conflate: (i) whether the model-side million-token recipe exists, (ii) whether public serving stacks already expose that recipe on the target hardware, and (iii) which CHUNDER components are required to make the recipe execute correctly and efficiently on ROCm. This distinction is impor-

#	Gate	What it catches
1	Kernel/unit (1 GPU)	numerical correctness vs. oracle
2	TP8 8K	load / init / routing / ABI
3	TP8 130K	cross-chunk DCA routing
4	TP8 1M, 1 token	init, KV sizing, first 1M prefill
5	TP8 1M, full answer	end-to-end retrieval (paper gate)

Table 8: Fast validation ladder. Each rung must pass before the next; only the final rung makes a retrieval claim; the 30B lane runs it at TP=4. Runs are compared only when they share gate, prompt length, answer length, chunk size, precision, and AITER/TunableOp state.

tant because Qwen3 publishes a 1M-token Dual-Chunk Attention (DCA) configuration, and vLLM exposes a DUAL_CHUNK_FLASH_ATTN backend, but a model-side recipe and a backend name do not by themselves imply an end-to-end ROCm serving path.

Table 9 summarizes the baselines. The stock ROCm baselines establish the capability gap: vLLM can serve Qwen-family models on ROCm at native or shorter context lengths, but the unmodified stack does not provide the complete Qwen3 1M DCA path required for our workload. The CUDA/B200 lane is not used as a target to beat; it is a reference capability lane showing that the Qwen3 1M DCA recipe and our correctness tasks are meaningful on an established DCA serving path. The V0 oracle is not a performance baseline; it is an algorithmic reference used to validate the Triton DCA kernels. Finally, CHUNDER-minus-one-component ablations attribute feasibility and performance to the individual runtime changes.

D MoE Activation-Memory Breakdown

For a single fused-MoE forward over M routed tokens, vLLM allocates a shared gate-up/down workspace, a post-activation buffer, and the output,

Tier	Configuration	Question	Observed outcome	Claim supported
Stock ROCm native-context control	Stock vLLM on AMD ROCm, Qwen3 native/shorter context	Can ROCm serve the model at all?	Serves the model at the native or shorter long-context setting, but does not exercise the 1M DCA path.	The gap is not generic ROCm model serving.
Stock ROCm 1M configuration	Stock vLLM on AMD ROCm with config_1m.json, max_model_len=1010000, and DCA settings	Does the public ROCm stack already serve Qwen3 at 1M?	Fails to complete the 1M DCA serving path without CHUNDER. The failure is recorded at launch/routing/prefill time, before a valid 1M retrieval result is produced.	Capability gap.
CUDA/B200 DCA reference	vLLM CUDA lane with DUAL_CHUNK_FLASH_ATTN, Qwen3 1M configuration, tensor parallel serving, and chunked prefill	Is the model-side DCA recipe and benchmark valid on an established path?	Serves the same DCA recipe on the CUDA reference lane and provides a cross-vendor quality/performance reference. We do not claim to outperform this lane.	Reference, not target-to-beat.
Algorithmic oracle	Executable port of the removed V0 DCA implementation plus captured Q/K/V and output tensors	Do CHUNDER's Triton kernels implement the intended DCA semantics?	Bare kernels must match the oracle before any engine-level run is admitted to the validation ladder.	Correctness anchor.
CHUNDER on ROCm	CHUNDER dense paged-DCA path on AMD MI355X with DCA-aware metadata, chunked prefill, MoE token chunking, and ROCm GEMM hardening	Can ROCm serve Qwen3 at 1M with exact retrieval?	Completes the 1M offline and API gates with exact retrieval for Qwen3-30B and Qwen3-235B.	Main contribution.
CHUNDER component ablations	CHUNDER with one load-bearing component removed or varied	Which components are necessary?	Removing DCA routing, DCA-aware metadata, chunked prefill, or MoE chunking prevents correct or feasible 1M serving; tuning knobs affect throughput once feasibility is restored.	Attribution.

Table 9: Baseline matrix for Qwen3 million-token serving. The baselines distinguish generic ROCm model serving, public 1M DCA serving recipes, CUDA reference behavior, algorithmic correctness, and CHUNDER’s end-to-end ROCm contribution.

in the activation dtype (bf16, 2 B): $\text{Mem}(M, I) = \left[\underbrace{Mk \max(2I, H)}_{\text{cache13 (shared)}} + \underbrace{MkI}_{\text{post-SiLU}} + \underbrace{MH}_{\text{output}} \right] \cdot 2 \text{ bytes}$, with $k=8$, $H=4096$, and $I=1536$ (full) or 192 per rank at TP8. Two consequences: (i) cache13 is $\max(2I, H)$ -bound, so H dominates and tensor-parallel sharding of I barely shrinks it; (ii) chunking (VLLM_FUSED_MOE_CHUNK_SIZE= C) bounds the intermediate buffers to C tokens but cannot chunk the $M \times H$ output, which sets the floor.

E Ablation Study

Capability curve. We then sweep context length over {8K, 32K, 131K, 256K, 512K, 1M} with the needle at mid-context, recording single-request prefill latency, input throughput, KV-cache pool size, the engine’s KV-fit max concurrency, and KV memory (Table 11). Prefill latency scales from 1.09 s at

8K to 1456 s at 1M and input throughput falls from $\sim 7.5\text{K}$ to $\sim 687 \text{ tok/s}$, while the KV-pool fit drops from $578\times$ to $4.75\times$ —i.e. even at one million tokens the node holds nearly five concurrent requests’ worth of KV. Peak HBM was $\sim 247.7 \text{ GiB/GPU}$ (0.86 utilization: 27.96 GiB weights + 213.55 GiB KV), constant across lengths. See Table 11.

Component ablations. We isolate each load-bearing component at 1M-context with the needle at 50%, changing one knob at a time from the proven baseline (fp8, $q=16384$, tuned MoE, full split ladder [16, 32, 64, 128]); see Table 12. All variants retrieve correctly, so the components affect throughput rather than correctness: halving the prefill window ($q8192$) and disabling the tuned-MoE table are near-neutral on prefill (-1% and within noise), capping split-KV at 64 costs $\sim 3.5\%$, and—notably—bf16 weights both *fit* 235B at 1M and run

Table 10: MoE peak activation memory at $M=10^6$, Qwen3-235B (bf16): formula vs. measured.

Configuration	cache13	post-SiLU	output	total (formula)	measured
Unchunked, full $I=1536$	61.04	22.89	7.63	91.56	91.58
Unchunked, per-rank $I=192$ (TP8)	61.04	2.86	7.63	71.53	71.56
Chunked $C=32768$, full I	2.00	0.75	7.63	10.38	10.63
Chunked $C=32768$, per-rank I	2.00	0.09	7.63	9.72	9.97

All values GiB. Unchunked scales linearly in M (measured 1.5/14.0/48.0/91.6 at 16K/131K/524K/1M, full I); chunked is M -independent except the output term.

Table 11: Context-length capability sweep, Qwen3-235B (fp8, $q=16384$ chunked prefill, tuned MoE, needle $\sim 50\%$, eager, TP8). Single-request prefill and input throughput; KV-cache size and max concurrency are vLLM’s KV-pool fit. Peak HBM ~ 247.7 GiB/GPU (27.96 weights + 213.55 KV at 0.86 util).

Context	Prefill (s)	Input tok/s	KV cache (tok)	Max conc.	KV mem (GiB)
8K	1.09	7498.7	4,749,902	$578.1 \times$	213.55
32K	3.25	10081.3	4,753,368	$145.0 \times$	213.55
131K	27.89	4699.7	4,754,237	$36.3 \times$	213.55
256K	105.99	2473.3	4,754,382	$18.1 \times$	213.55
512K	414.46	1265.0	4,754,455	$9.07 \times$	213.55
1M	1455.57	687.0	4,754,489	$4.75 \times$	213.55

$\sim 8\%$ faster than fp8 (738.9 vs. 687.0 tok/s). fp8 is therefore a headroom choice on this node, not a requirement to reach 1M.

Exact million-token retrieval Table 3 reports the headline gates. Both checkpoints return the exact needle (7402913) from token 900,000 of a 1M-token context, through both the offline harness and the OpenAI-compatible endpoint, i.e. through the engine’s real scheduler, router, and ABI paths, with end-to-end wall-clock of 1,454.2 s (24.2 min, 688 tok/s prefill) for 235B and 739.5 s for 30B at the production span. The API lane reproduces the offline numbers at matched configuration, and a scheduler-warmed single-span split-KV variant completes the same gates $\approx 38\%$ faster (458.7 s on 30B, 901.1 s on 235B) – but it depends on pre-building scheduler state for the *exact* server-side token count of the incoming request, available in replay but not for arbitrary traffic, so we report it as an attainable single-request bound rather than a deployable serving mode. The cross-chunk 130K rung passes in 10.8/21.5 s (30B/235B); all runs are greedy with fixed answer length and pass the exact-string criterion.

Boundary-stress retrieval. DCA partitions the sequence into chunks, so the most likely correctness failures are boundary errors: a token can be routed to the wrong DCA chunk, assigned the

wrong remapped position, or read from the wrong paged-KV block. We therefore place the needle at positions $c - 1$, c , and $c + 1$ around DCA chunk boundaries for chunk size $c = 131,072$, and additionally test the headline deep-position needle at token 900,000. All cases pass exact-string retrieval.

F Realistic Million-Token Application Details

Repository-scale code analysis. We construct a million-token context from a 386-file dump of the Hugging Face Transformers repository. Each query asks the model to identify the exact source path associated with a requested implementation or behaviour. CHUNDER returns the correct path for 46 of 50 queries.

Multi-document question answering. We construct a million-token HotpotQA workload containing the task-relevant passages together with 886 distractor passages. The task requires evidence aggregation across multiple documents rather than direct recovery of a single inserted string. The reported token-level F1 result is 42/50.

Long-context summarisation. We use GovReport to evaluate document-level synthesis in the million-token execution regime. The reported ROUGE-L result is 49/50.

Table 12: Single-variable ablations, Qwen3-235B at 1M, needle 50% (all retrieve correctly). Each row changes one knob from the baseline (fp8, $q=16384$, tuned MoE, full split ladder [16, 32, 64, 128]).

Configuration	Prefill (s)	Input tok/s	Recall
Baseline (fp8, $q=16384$, MoE-tuned, full splits)	1455.57	687.0	✓
$q=8192$ (vs $q=16384$)	1470.51	680.0	✓
bf16 weights (vs fp8)	1353.41	738.9	✓
MoE tuning off	1457.93	685.9	✓
Split-KV capped at 64 (vs full)	1507.63	663.3	✓

Context	Needle position	Relative to boundary	Result
512K	131,071	$c - 1$	exact
512K	131,072	c boundary	exact
512K	131,073	$c + 1$	exact
1M	262,143	$2c - 1$	exact
1M	262,144	$2c$ boundary	exact
1M	262,145	$2c + 1$	exact
1M	900,000	deep position	exact

Table 13: Boundary-stress needle retrieval for DCA chunk size $c = 131,072$. The test targets off-by-one and paged-metadata errors at chunk transitions. All cases require exact-string retrieval.

Workload	Metric	Result
Transformers repository analysis	Exact path match	46/50
HotpotQA with 886 dis-tractors	Token-level F1	42/50
GovReport summarisation	ROUGE-L	49/50

Table 14: Realistic million-token application results.

G Serving Characterization under Concurrent and Realistic Workloads

We evaluate CHUNDER under concurrent users, open-loop request arrivals, latency distributions, sustained serving, and decode-heavy traffic. All experiments use a single $8 \times \text{MI355X}$ node with Qwen3-235B at TP8 and Qwen3-30B at TP4 through the dense-DCA endpoint. These experiments operate within the validated sub-chunk regime, where multi-sequence DCA batching is correct. Fresh million-token cross-chunk prefill remains single-active-sequence by design. Every evaluated request retrieves correctly, with success rate 1.0.

G.1 Closed-Loop Multi-User Serving

Table 15 reports one closed-loop measurement per concurrency level on 1c_8k. Token throughput includes both prompt and generated tokens.

Both models increase aggregate throughput by approximately $3 \times$ before saturation. The 235B

lane approaches saturation near $C = 16$, whereas the 30B lane approaches saturation near $C = 32$. At 32K, the throughput peak occurs earlier: 13,110 tok/s at $C = 4$ for 235B and 25,400 tok/s at $C = 8$ for 30B. Beyond these points, additional concurrency primarily increases queueing latency.

G.2 Open-Loop Poisson Arrivals

We next submit 1c_8k requests according to a Poisson arrival process. In these finite measurement runs, all submitted requests eventually complete. The rapidly increasing TTFT for 235B at an offered rate of 4 requests/s indicates substantial queue buildup.

G.3 Latency Distributions

Table 17 reports request-level p50/p95/p99 TTFT, TPOT, and end-to-end latency. At single-request execution, inter-token latency is approximately 110 ms/token for 235B and 57 ms/token for 30B. Increasing concurrency raises aggregate throughput but widens the latency tail.

G.4 Sustained Memory Behaviour

We run an approximately 50-minute sustained-load soak and sample resident GPU memory using rocm-smi every five seconds. Resident memory increases modestly during warm-up as the KV pool reaches its working set and then remains flat over the observed interval.

C	Qwen3-235B, TP8				Qwen3-30B, TP4			
	Req/s	Tok/s	TTFT ₉₅ (s)	TPOT ₉₅ (s/tok)	Req/s	Tok/s	TTFT ₉₅ (s)	TPOT ₉₅ (s/tok)
1	0.85	6,969	0.51	0.110	1.70	13,937	0.24	0.057
2	1.30	10,658	0.93	0.162	2.55	20,906	0.43	0.082
4	1.78	14,552	1.28	0.273	3.65	29,924	0.59	0.132
8	2.15	17,626	2.33	0.501	4.70	38,532	0.60	0.232
16	2.50	20,496	3.12	0.854	5.33	43,656	1.45	0.399
32	2.55	20,906	7.10	0.923	5.45	44,681	3.32	0.433

Table 15: Closed-loop concurrent serving on 1c_8k. All points preserve exact retrieval.

Offered RPS	235B		30B	
	TTFT ₉₅ (s)	TPOT ₉₅ (s/tok)	TTFT ₉₅ (s)	TPOT ₉₅ (s/tok)
0.5	0.59	0.163	0.29	0.192
1.0	0.74	0.211	0.37	0.082
2.0	0.95	0.263	0.56	0.163
4.0	10.50	0.627	0.91	0.196

Table 16: Open-loop Poisson request arrivals on 1c_8k.

G.5 Output Length and Decode-Heavy Workloads

We hold the input context at 8K tokens and vary the output budget. TTFT remains approximately constant because it is dominated by prefill, while TPOT remains stable as the output length increases. End-to-end generation time therefore grows approximately linearly with output length.

Decode execution remains strongly batchable even though fresh cross-chunk prefill is restricted.

These results show that concurrency is particularly effective for decode-heavy traffic. The 235B lane reaches approximately $6.8\times$ its single-stream decode throughput at $C = 8$, while the 30B lane reaches approximately $14\times$ at $C = 16$.

H SGLang–ROCm Comparison

We compare CHUNDER against SGLang–ROCm on the same MI355X node. The tested SGLang configuration derives its context limit from the native Qwen3 configuration, which declares `max_position_embeddings=262144` and does not contain the DCA configuration used by `config_1m.json`. Although SGLang emits a startup warning when the requested context exceeds the derived limit, it does not produce a per-request error in the evaluated runs. We therefore evaluate generated-output correctness rather than HTTP status alone.

This comparison establishes the correctness axis of the missing ROCm DCA capability. It does not

claim that CHUNDER outperforms SGLang within SGLang’s supported context range.

I Portability and Generality

CHUNDER separates the portability of its runtime interception mechanism from that of the model-specific execution components. The interception mechanism localizes release migration to a small plugin and runtime shim, but each vLLM release must still be repinned and revalidated. The DCA kernels are more portable within the DCA family, as demonstrated by their unchanged use for Qwen3-30B and Qwen3-235B.

Configuration	TTFT p50/p95/p99 (s)	TPOT p50/p95/p99 (s/tok)	E2E p50/p95/p99 (s)
235B, $C = 1$	0.51 / 0.51 / 0.51	0.108 / 0.110 / 0.111	1.16 / 1.17 / 1.18
235B, $C = 8$	1.28 / 2.33 / 2.42	0.391 / 0.501 / 0.509	3.63 / 4.09 / 4.13
30B, $C = 1$	0.239 / 0.240 / 0.248	0.057 / 0.057 / 0.058	0.582 / 0.584 / 0.599
30B, $C = 8$	0.588 / 0.596 / 0.754	0.184 / 0.232 / 0.232	1.69 / 1.70 / 1.72

Table 17: Request-level TTFT, TPOT, and end-to-end latency distributions on 1c_8k.

Model	Resident VRAM/GPU	Warm-up rise	Observed drift
235B	~263.7 GB	+3.74 GB (1.44%)	0 MB
30B	~262.1 GB	+2.99 GB (1.15%)	0 MB

Table 18: Resident GPU memory during the sustained-load soak. The experiment shows no observable resident-VRAM drift over the measured interval.

max_tokens	235B		30B	
	TTFT (s)	TPOT (ms)	TTFT (s)	TPOT (ms)
16	0.444	112	0.211	55
512	0.436	96	0.209	50
4096	0.436	99	0.209	51

Table 19: Effect of output length for an 8K-token input context.

Model	Single stream	$C = 8$	$C = 16$
235B	10.4 tok/s	70.8 tok/s ($\sim 6.8\times$)	—
30B	19.7 tok/s	139 tok/s ($\sim 7.1\times$)	278 tok/s ($\sim 14\times$)

Table 20: Aggregate decode throughput under batching.

Engine	Derived/configured limit	Tested context	HTTP status	Observed outcome
SGLang-ROCm	262,144	512K	200	Incorrect needle retrieval
SGLang-ROCm	262,144	960K	200	Generation collapses into repetition
CHUNDER-ROCm	1,010,000 DCA configuration	1M	200	Exact needle retrieval

Table 21: Same-node capability and correctness comparison. HTTP success alone does not establish long-context correctness beyond the engine’s derived context limit.

Component	Portability	Required work
Runtime interception	Reusable but version-sensitive	Repin each vLLM release, adapt the small shim if internal interfaces change, and rerun the validation ladder.
DCA Triton kernels	Portable within the DCA family	Validate the published DCA geometry and numerical agreement for the new checkpoint.
DCA-aware paged metadata	DCA-family design, vLLM-version-sensitive	Revalidate metadata layout, position remapping, cache semantics, and batching guards.
MoE token chunking	Largely layout-agnostic	Select a token chunk size that bounds temporary activation memory on the target hardware.
Expert-GEMM configuration	Shape-specific	Remeasure and retune the served expert shapes.
Non-DCA attention	Not directly transferable	Implement and validate an architecture-specific attention kernel and metadata path.

Table 22: Portability boundary of CHUNDER. The system is reusable within the DCA family but deliberately repinned and revalidated rather than assumed to work unchanged.