

r9k: a from-scratch inference engine for Qwen3.8-27B on two Radeon AI PRO R9700

r9k is a C++ and HIP inference engine built by [unassuming.ai](#) for one model on one machine: Qwen3.8-27B, served from its published FP8 checkpoint across two Radeon AI PRO R9700 cards with tensor parallelism. It is the engine behind the Qwen3.8-27B chat and API at [unassuming.ai](#). This report compares it with vLLM, SGLang, llama.cpp and Radiance measured on the same machine with the same harness, lists published results for the same GPU, and documents the engine's design, version history through v9 and the techniques we evaluated.

<div>127.9 tok/s</div> <div>Decode, 8K-token code prompt, one request</div> <div>vLLM 0.31 tuned: 48.5 · 2.6×</div>	<div>264.6 tok/s</div> <div>Total output, eight concurrent 2K requests</div> <div>vLLM 0.31 tuned: 182.1 · 1.45×</div>
<div>480 ms</div> <div>Time to first token, 2K-token prompt</div> <div>vLLM 0.31 tuned: 569 ms</div>	<div>103.4 tok/s</div> <div>Decode, 2K prompt, sampled at the model's default settings</div> <div>r9k with built-in MTP: 83.1</div>

r9k v8 against vLLM on the same machine and harness, 18 October 2026. v9, in production since 26 October, stays within 1% of v8 on this harness and adds faster copy-heavy edits and cached multi-turn prompts ([v9 section](#)).

Summary

Qwen3.8-27B is a hybrid model: 48 Gated DeltaNet layers and 16 full-attention layers, 27 billion parameters, with FP8 E4M3 weights in 128 × 128 scale blocks. On a pair of R9700 cards each forward pass reads about 13.5 GB of weights per GPU, so single-request decoding is bound by memory bandwidth and by how many tokens each pass can produce.

r9k keeps the whole decode step on the GPU as one HIP graph, splits the model over the two cards with its own PCIe all-reduce, verifies several draft tokens per pass with lossless speculative decoding, and batches up to eight requests. Prefill runs through a custom block-scaled FP8 GEMM. The engine is about 8,100 lines of C++ and HIP with 70 GPU kernels, plus an OpenAI-compatible HTTP server.

Against a tuned vLLM 0.31 deployment on the same machine and harness, r9k v8 decodes one request 1.9× faster on a 2K prompt and 2.6× faster on an 8K code prompt, reaches first token 16% sooner on a 2K prompt, and produces 1.45–1.5× the total throughput at four and eight concurrent requests.

v9, serving since 26 October, adds lookup drafting from the conversation's own history with 16-row verification, and a host-memory prefix cache. Edits that reproduce most of their input (renaming a function, proofreading, adding a field to every JSON record) decode 63–73% faster than v8, at 14–15 tokens per step, and follow-up turns of a conversation reach their first token in about 95 ms instead of 760–790 ms. The standard harness stays within 1% of v8, with identical outputs.

Use r9k at unassuming.ai

The engine described here serves Qwen3.8-27B to the public through unassuming.ai: a browser chat and an OpenAI-compatible API. Both need an access key, which you can request from contact@unassuming.ai.

Chat	unassuming.ai/chat
API base URL	https://unassuming.ai/v1 · /chat/completions and /completions, streaming or not; access key as a Bearer token
Model ID	Qwen/Qwen3.8-27B
Limits	65,536 tokens of context (input plus output), up to 32,768 output tokens, eight concurrent requests
Data use	Prompts are not used for training. See the privacy policy and terms .

```
curl https://unassuming.ai/v1/chat/completions \
-H "Authorization: Bearer YOUR_API_KEY" \
-H "Content-Type: application/json" \
-d '{ "model": "Qwen/Qwen3.8-27B",
      "messages": [{ "role": "user", "content": "Hello." }],
      "stream": true }'
```

System under test

GPUs	2 × AMD Radeon AI PRO R9700 (RDNA 4, gfx1201), 32 GB GDDR6 each, 300 W board power limit; about 640 GB/s measured per card
Interconnect	PCIe 5.0 x16 links at the cards; GPU-to-GPU P2P copies measured at 20–27 GB/s per direction
Host	AMD Ryzen 5 9600X (6 cores, 12 threads), 64 GB DDR5
OS and driver	Ubuntu 24.04.5 LTS, Linux 6.8, amdgpu 6.16.6
ROCm	ROCm 7.1.0, HIP 7.1.25424, AMD clang 20; hipBLASLt is used only on the BF16 prefill fallback path
Model	Qwen3.8-27B, published FP8 checkpoint (E4M3, 128 × 128 block scales), BF16 KV cache, fp32 recurrent state
Drafters	The model's built-in MTP head (v1–v7); the DFlash2 block drafter for Qwen3.8-27B, 5 layers, linear weights converted to INT4 (v8 and v9); lookup drafting from the sequence's own history (v9)
Baselines	vLLM 0.31.0 (tuned, FP8, built-in MTP, TP=2); vLLM 0.31.0 with DFlash2; SGLang gfx1201 port (FP8, TP=2); llama.cpp HIP (Q4_1 GGUF, MTP); Radiance 1.3.0 (FP8, built-in MTP, TP=2)

Benchmark method

All rows in the next section were measured with one harness on this machine. Each request streams over HTTP from a local client with fixed token IDs, temperature=0, ignore_eos and 256 output tokens; every scenario gets one warm-up round and three measured rounds, and the medians are reported. The scenarios are a 2,048-token English prompt with one and with four concurrent requests, an 8,192-token code prompt with one request, and a 2,048-token code prompt with eight concurrent requests. Decode speed is output tokens after the first divided by the time after the first token. Total throughput counts all output tokens over the wall time, including prefill. Prefill speed is prompt length divided by time to first token.

The sampled variant of the harness differs only in its sampling parameters (temperature 1.0, top-p 0.95, top-k 20, the model's own defaults).

Comparison on the same machine

Greedy decoding, same harness and hardware for every row. Decode columns are tokens per second for one request; total columns are output tokens per second across all concurrent requests.

One request

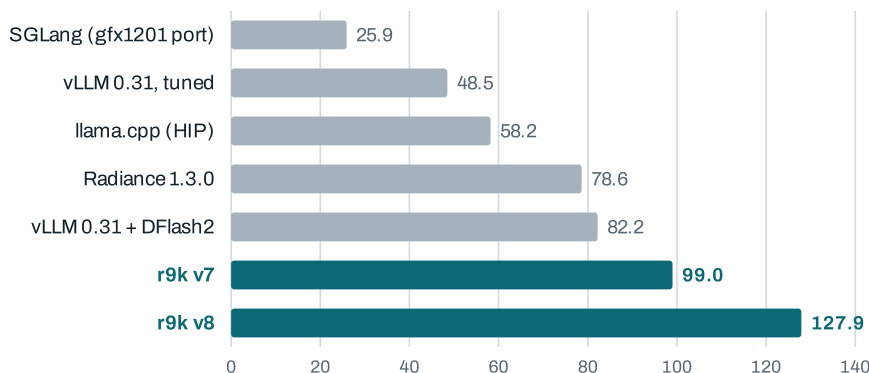
Engine	TTFT	Decode	TTFT	Decode
	2K prompt, ms	2K prompt, tok/s	8K prompt, ms	8K prompt, tok/s
llama.cpp (HIP) Q4_1 GGUF · MTP, 4 drafts	1,864	51.7	7,252	58.2
SGLang (gfx1201 port) FP8 · none	660	26.0	2,726	25.9
vLLM 0.31, tuned FP8 · built-in MTP, 1 draft	569	49.9	2,278	48.5
vLLM 0.31 + DFlash2 FP8 · DFlash2, 7 drafts	582	74.6	2,270	82.2
Radiance 1.3.0 FP8 (FP8 lm_head) · built-in MTP, 3 drafts	531	77.9	2,000	78.6
r9k v7 FP8 · built-in MTP, 3 drafts	730	98.2	2,811	99.0
r9k v8 FP8, W8A8 prefill · DFlash2, adaptive 3–7	480	92.8	2,037	127.9

Concurrent requests and prefill

Engine	4 requests	8 requests	Prefill
	2K prompts, total tok/s	2K prompts, total tok/s	2K prompt, tok/s
llama.cpp (HIP) Q4_1 GGUF · MTP, 4 drafts	59.5	64.8	1,099
SGLang (gfx1201 port) FP8 · none	82.0	131.1	3,105
vLLM 0.31, tuned FP8 · built-in MTP, 1 draft	126.3	182.1	3,599
vLLM 0.31 + DFlash2 FP8 · DFlash2, 7 drafts	136.8	181.9	3,521
Radiance 1.3.0 FP8 (FP8 lm_head) · built-in MTP, 3 drafts	167.3	220.9	3,860
r9k v7 FP8 · built-in MTP, 3 drafts	169.4	213.3	2,807
r9k v8 FP8, W8A8 prefill · DFlash2, adaptive 3–7	191.1	264.6	4,263

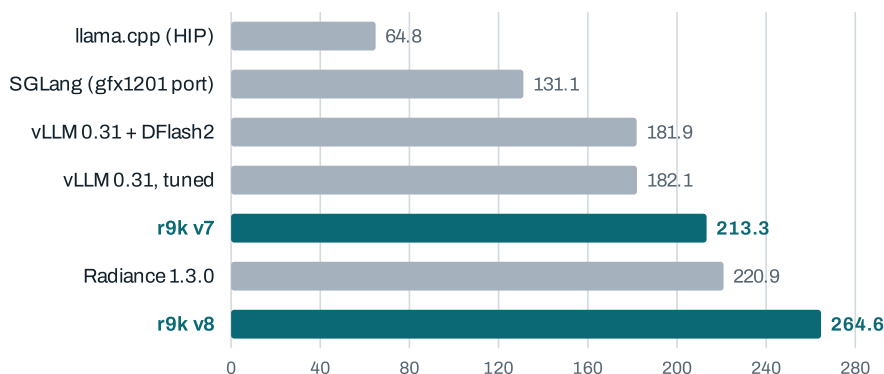
Decode speed, one request

8K-token code prompt, tok/s



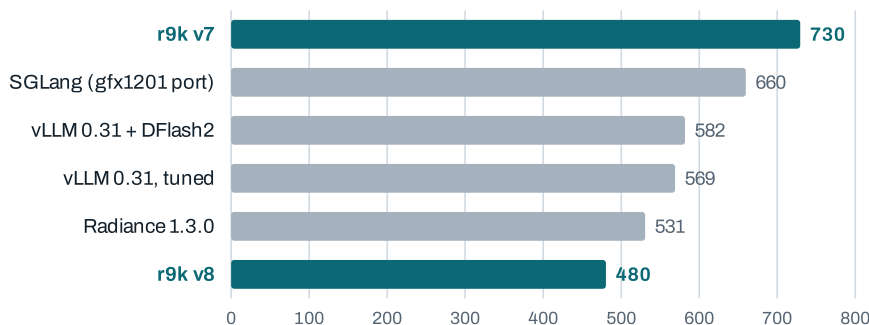
Total throughput, eight concurrent requests

2K-token code prompts, tok/s



Time to first token

2K-token prompt, ms, lower is better



vLLM was tuned during an earlier optimization pass on this machine. The DFlash2 row for vLLM includes a fix that keeps the drafter's convolution projections in BF16; without it, acceptance falls to nearly zero. The Radiance row uses its built-in MTP head, which is the configuration comparable to r9k v7. r9k v7 and v8 differ in drafter and prefill path; both run the same FP8 target weights.

Published results on Radeon AI PRO R9700

Figures reported by others for the same GPU, with their own models, prompts and harnesses. They describe the state of the ecosystem but are not like-for-like with the table above: prompt lengths, output lengths, model versions, quantization and GPU counts differ, as each row states. The last card shows r9k results from our records with the closest workload we have measured: six tasks (a 2K English prompt and five short chat prompts) and 256 output tokens, run in-process with the engine's command-line driver and averaged over two runs.

C=1 figures are output tokens per second for one request; C=4 and C=8 are aggregate output throughput; TTFT is for one request. A dash means the source did not report the value. Puget Systems tested Qwen3.6-27B, the predecessor model with the same architecture class.

Eliovp: Paiton on Radeon AI PRO R9700				September 2026
Paiton (vLLM plugin), Qwen3.8-27B INT4 W4A16 (AMD Quark), 1× R9700, greedy, one request				
Configuration	C=1	C=4	C=8	TTFT
Interactive, 256 in / 256 out	39.8	–	–	290 ms
Coding, 1,024 in / 512 out	37.9	–	–	737 ms
Long context, 4,096 in / 256 out	25.5	–	–	2,984 ms

Puget Systems: R9700 dual-GPU inference				August 2026
Qwen3.6-27B FP8, 2× R9700 with TP=2, 500 input / 200 output tokens, GenAI-Perf				
Configuration	C=1	C=4	C=8	TTFT
vLLM 0.20.2, stock	15.9	69.3	156.2	337 ms
vLLM, tuned	32.0	110.6	198.8	136 ms
vLLM, tuned, with MTP	62.8	186.3	320.2	184 ms

Radiance: Qwen3.8-27B FP8 model card				2026
Radiance ≥1.2.3, FP8 target with FP8 lm_head, DFlash2 with 7 drafts, 2× R9700 with TP=2, eight mixed short prompts, 768 output tokens, greedy, FP8 KV cache				
Configuration	C=1	C=4	C=8	TTFT
All eight prompts	141	314	–	–
Prose / code prompts	93 / 184	–	–	–

AMD: Qwen3.8-27B Day-0 blog				August 2026
llama.cpp (Vulkan), Qwen3.8-27B, MTP with 2 drafts, 1× R9700				
Configuration	C=1	C=4	C=8	TTFT
Output generation, up to	51.8	–	–	–

r9k v8: closest workload in our records				October 2026
Qwen3.8-27B FP8, DFlash2 with adaptive verification, 2× R9700 with TP=2, six tasks (a 2K English prompt and five short chat prompts), 256 output tokens, command-line driver, mean of two runs				
Configuration	C=1	C=4	C=8	TTFT
Greedy	125.2	–	337	–
Temperature 1.0, top-p 0.95, top-k 20	117.3	–	324	–

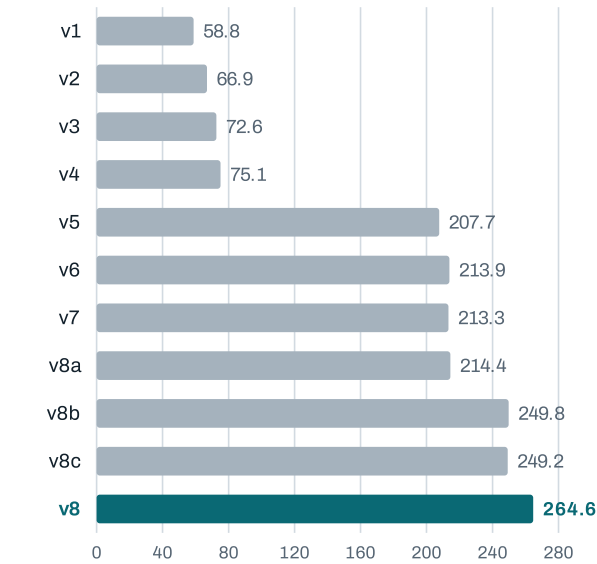
Version history

Each version was benchmarked with the same harness when it was completed. v1 through v4 decoded one request at a time, so their concurrent totals reflect queued requests. Continuous batching arrived in v5.

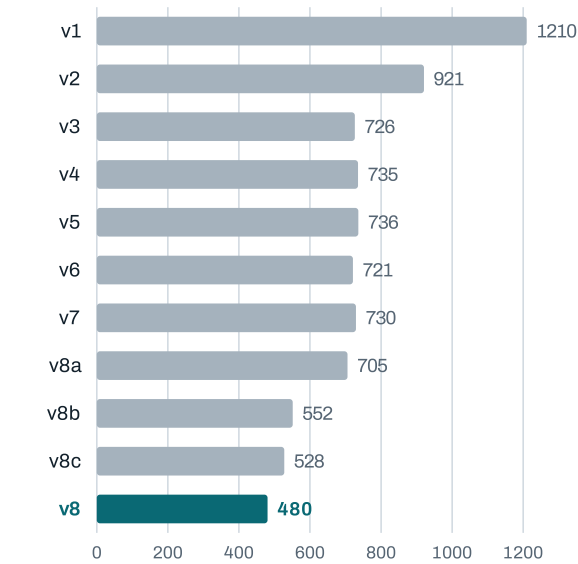
Version	Change	TTFT 2K, ms	Decode 8K, tok/s	4 req. total	8 req. total
v1	First complete engine: FP8 GEMV, built-in MTP (3 drafts), TP=2 over PCIe P2P, one HIP graph per step	1,210	78.9	57.4	58.8
v2	INT4 draft head, attention rewrite, GDN and all-reduce improvements	921	90.4	66.3	66.9
v3	Per-shape GEMV K-split, layer-major dual-stream prefill	726	94.1	72.0	72.6
v4	fp16-activation GEMV (v_dot2), INT4 MTP layer weights	735	98.7	74.4	75.1
v5	Continuous batching, 8 sequences, paged KV cache	736	98.4	164.7	207.7
v6	WMMA skinny GEMM, GDN replay fused into verify, WMMA flash-decoding	721	98.9	169.2	213.9
v7	Sampling: temperature, top-p, top-k, min-p, penalties, seeds	730	99.0	169.4	213.3
v8a	DFlash2 block drafter: exact rescoring, sampled drafts, INT4 drafter	705	123.9	162.7	214.4
v8b	FP8 W8A8 prefill GEMM, multi-prompt batched prefill	552	128.9	183.8	249.8
v8c	Adaptive verification width (3 / 5 / 7 drafts per step)	528	127.9	185.1	249.2
v8	Copy-engine (SDMA) prefill all-reduce, per-sequence prefill ordering	480	127.9	191.1	264.6
v9	Lookup drafting with 16-row verification, host-memory prefix cache (measured through the production gateway, see note)	445	129.2	197.1	275.7

v1-v8 were measured directly against the engine. v9 was measured on 26 October through the production gateway, under the memory limits that let two smaller models share the GPUs; v8 measured the same way on the same day gave 441 ms, 129.7, 198.4 and 277.9, so the differences from the v8 row come from the newer setup, not from v9. The charts stop at v8 for that reason.

Total throughput, eight concurrent requests
tok/s by version, v1-v8



Time to first token, 2K-token prompt
ms by version, v1-v8, lower is better



Sampled decoding

Chat deployments sample from the model rather than decode greedily. Speculative decoding usually loses acceptance under sampling, so we measure this path separately. In v8 the drafter samples its proposals from its own distribution and the engine accepts each with probability $\min(1, p/q)$, resampling from the residual on rejection; the output distribution is unchanged.

Benchmark scenarios

Scenario	v7 built-in MTP	v8 DFlash2	Change
2K-token prompt, TTFT (ms)	723.8	481.9	1.50×
2K-token prompt, decode, 1 request (tok/s)	83.1	103.4	1.24×
8K-token code prompt, TTFT (ms)	2,823.4	2,028.3	1.39×
8K-token code prompt, decode, 1 request (tok/s)	90.5	125.8	1.39×
4 concurrent 2K requests, total (tok/s)	158.6	192.4	1.21×
8 concurrent 2K requests, total (tok/s)	206.7	265.5	1.28×

Single-request decode by task

Task	v7 MTP	v8 DFlash2	Change
English (2K prompt)	84.4	101.1	1.20×
Code	102.7	154.6	1.51×
Chinese	84.8	87.8	1.04×
Prose	69.2	74.7	1.08×
JSON	121.6	158.6	1.30×
Math	95.9	126.9	1.32×
Mean	93.1	117.3	1.26×

tok/s, one request, 256 output tokens, mean of two runs. The English prompt is 2,048 tokens; the other prompts are short chat requests.

v9: lookup drafting and a prefix cache

By v8, edits that reproduce most of their input, such as renaming a function and returning the file, were already accepting 7.5–8 tokens per step: the ceiling of an eight-row verification block. v9 doubles the block to 16 rows and fills the extra rows from the sequence's own history. It also keeps snapshots of processed prompts in host memory, so the next turn of a conversation computes only its new tokens. v9 has served production traffic since 26 October 2026.

<div>445 tok/s</div> <div>Rename a function in a 150-line Python file and return the file, one request</div> <div>v8: 257 · 1.73×</div>	<div>15.4</div> <div>Tokens per verification step on the same task</div> <div>v8: 8.0, at its eight-row limit</div>
<div>94 ms</div> <div>Time to first token, second turn of a 3.4K-token conversation</div> <div>v8: 756 ms</div>	<div>66 / 66</div> <div>Standard-harness outputs identical to v8</div> <div>Speed within 1% of v8</div>

16-row verification

The verification block grows from 8 to 16 rows, and for one or two concurrent requests the scheduler gains a decode graph that verifies 15 drafts. Attention treats the 16 rows as two eight-row WMMA halves, the second reading the first half's keys and values from the KV cache; the other kernels were already generic in the row count.

Lookup drafting inside the graph

Each sequence keeps its token history on the GPU. After DFlash2 proposes its seven drafts, a kernel in the same graph looks for the longest earlier match of the context plus those drafts (at least three tokens) and continues it as drafts 8–15. When the context alone matches at least 32 tokens and the continuation starts with DFlash2's top choice, the continuation replaces all the drafts: the model is copying.

When to verify wide

Wide steps cost extra rows in every layer, so the scheduler uses them only after a match of 12 or more tokens. Each sequence tracks how often its wide steps were accepted beyond the eighth draft; below 0.4 it stops asking for them and retries later.

Copy-heavy edits

Task	v8	v9	Change
	tok/s, tokens per step	tok/s, tokens per step	
Rename a function, output the whole file	257.1	445.2	1.73×
	7.99 / step	15.35 / step	
Fix spelling, output the whole text	254.0	414.7	1.63×
	7.91 / step	14.16 / step	
Add a field to every JSON record	241.1	402.4	1.67×
	7.52 / step	13.83 / step	
Replace punctuation in Chinese text	247.7	273.9	1.11×
	7.73 / step	8.80 / step	
Control: explain the code function by function	172.8	188.0	1.09×
	5.35 / step	6.12 / step	

One request, thinking off, greedy, up to 1,536 output tokens; prompts of 0.9K–2.5K tokens. Sampled at temperature 0.7, top-p 0.8, top-k 20, the first three tasks run at the same speed. The control task writes new text, so lookup rarely finds a match there.

Still lossless when sampling

DFlash2 drafts sampled from q are accepted with $\min(1, p/q)$ as in v8. Lookup drafts are deterministic given their prefix and are accepted with probability $p(x)$. Whether lookup replaces the drafts depends only on the match length and DFlash2's top choice, never on the sampled drafts, so the drafts that remain still follow q.

Prefix cache

A snapshot holds both GPUs' DeltaNet state and convolution history, the drafter's context ring and the KV cache of the first P positions, about 330 MB for a 2K-token prefix, in an 8 GB host pool pinned once at start-up. A new prompt resumes from its longest cached prefix and prefills only the rest. Snapshots are taken one token before the end of the prompt, because the next turn's chat template re-tokenizes the end of the generation prompt.

Snapshots that leave the computation alone

The prefill's DeltaNet scan writes the state at the snapshot position straight into mapped host memory, so the prompt is never split. The copy engine moves the KV pages on a side stream that overlaps the following decode steps. Requests that miss the cache produce the same output as v8.

Multi-turn conversations

Scenario	v8 TTFT, ms	v9 TTFT, ms
Second turn, 3.4K-token prompt	756	94
Third turn, 3.6K-token prompt	786	95
Regenerate the last answer	783	54
Four conversations one after another, turns 2 and 3 (2.2–6K tokens, mean)	946	97
Four conversations at once, turns 2 and 3 (mean)	1,012	166
Cache miss: first turn, 3.1K-token prompt	709	713

Each conversation has a reference document in its system prompt and three questions, with up to 220 output tokens per turn. A cache miss costs the same as in v8.

Standard harness

Scenario	v8	v9	Change
2K English prompt, decode, 1 request tok/s	94.1	93.8	−0.3%
8K code prompt, decode, 1 request tok/s	129.7	129.2	−0.4%
4 concurrent 2K requests, total tok/s	198.4	197.1	−0.7%
8 concurrent 2K requests, total tok/s	277.9	275.7	−0.8%
2K English prompt, TTFT ms	441	445	+0.9%
8K code prompt, TTFT ms	1,922	1,916	−0.3%

The harness from the comparison above (greedy, 256 output tokens, medians of three rounds), run through the production gateway on 26 October for both versions. All 66 outputs, 42 from these scenarios and 24 from eight natural-task prompts, are identical to v8 with the same tokens per step; the natural tasks change by −0.1% to −1.0%. The remaining cost is the lookup kernel, about 0.1 ms per step, and the occasional wide step that is not accepted.

Engine design

Execution model

A decode step covers verification of the draft tokens, sampling and acceptance, the KV and recurrent-state commit, and drafting the next tokens. All of it is recorded once per batch size and verification width as a HIP graph and replayed with no host round trip; per-step state lives in a device control block. The host only reads finished tokens from pinned ring buffers and grants the same number of graph launches to both GPUs.

Tensor parallelism over PCIe

Attention heads, DeltaNet heads, MLP columns and the vocabulary are split across the two cards. A custom all-reduce pushes partial sums over P2P with epoch flags and fuses the residual add and RMSNorm, about 12 μs for a single-request step. Batched steps use a few pushing workgroups instead of one per row, because each system-scope fence costs about 0.8 μs and they serialize. Prefill moves its partial sums with the copy engines, so compute units stay with the other pipeline's GEMMs.

Decode kernels

FP8 weights are re-tiled offline so one wave-wide 16-byte load reads 512 contiguous bytes; activations stay in fp16 and products accumulate in fp32 through `v_dot2_f32_f16`. The main GEMVs run at 580–610 GB/s, about 92% of measured bandwidth. From 6 to 64 rows a WMMA skinny GEMM takes over for FP8, BF16 and INT4 weights. Attention decode is a WMMA flash-decoding kernel over a paged KV cache.

Speculative decoding

v1–v7 use the model's own MTP head with three drafts and an INT4 copy of the output head for drafting. v8 adds the DFlash2 block drafter: five layers that fill a block of eight positions in one pass from the target's hidden states, then a candidate selector picks a path through the top 16 tokens per position. r9k rescores the drafter's candidates exactly against the target's BF16 output head, samples drafts at temperatures above zero, and picks 3, 5 or 7 drafts per step from the batch's measured acceptance.

Batching and memory

Up to eight sequences decode together with continuous batching. Each sequence has its own slot for the recurrent DeltaNet state, a paged KV cache in 256-token pages and an output ring. Accepted tokens are folded into the DeltaNet state inside the next verification kernel instead of a separate pass. The scheduler admits several waiting prompts in one prefill pass.

Prefill

Prompts run layer by layer in chunks of up to 2,048 rows on two alternating streams. Linear layers use a custom block-scaled FP8 GEMM: activations are quantized dynamically per token and 128-element group, the scheme vLLM applies to this checkpoint, and the original 128×128 weight blocks are used unchanged. It reaches about 190 TFLOPS on the model's shapes and removes the per-prompt weight dequantization of the earlier BF16 path. Attention prefill is a WMMA flash-attention kernel.

Serving features

- OpenAI-compatible `/v1/chat/completions` and `/v1/completions`, streaming and non-streaming
- Thinking output split into `reasoning_content` and `content`, with reasoning effort levels and an option to hide reasoning
- Usage accounting including reasoning tokens, and server-side timings (time to first token, decode speed, tokens per step) on every response
- Sampling controls: temperature, top-p, top-k, min-p, presence, frequency and repetition penalties, per-request seeds
- Up to 65,536-token context and eight concurrent requests, with immediate 429 responses beyond that
- Multiple API keys reloaded without restart, SSE keep-alive before the first token
- OpenRouter provider feed (schema 2.4) and a browser chat client
- Prompt caching across conversation turns, reported as `usage.prompt_tokens_details.cached_tokens`
- Served publicly at <https://unassuming.ai/v1> through a reverse tunnel; one environment file selects drafter, prefill path and cache budget

Techniques evaluated

Measured effects on this machine. Each change was benchmarked in isolation before being kept.

Technique	Measured effect	Status
Whole decode step in one HIP graph	About 520 kernels per step launched as one graph; idle gaps between kernels total about 1.9 ms per step	ADOPTED
Re-tiled FP8 GEMV with fp16 activations	580–610 GB/s on the main matrices; v3 → v4 (with INT4 MTP weights) single-request decode 91.7 → 98.4 tok/s	ADOPTED
INT4 draft head and MTP layer	Drafting cost cut by more than half with no change to outputs	ADOPTED

Technique	Measured effect	Status
Continuous batching, paged KV	Eight concurrent requests 75.1 → 207.7 tok/s (v4 → v5)	ADOPTED
DFlash2 block drafter with exact candidate rescoring and sampled drafts	8K code decode 99.0 → 123.9 tok/s; six-task sampled mean 93 → 117 tok/s	ADOPTED
INT4 drafter weights	Same tokens per step as FP8 drafter weights; 2% faster single-request decode	ADOPTED
Adaptive verification width	Six-task mix at eight concurrent requests 300 → 335 tok/s (sampled)	ADOPTED
Batched all-reduce with few pushing workgroups	64-row all-reduce 72 → 48 μs; results bit-identical	ADOPTED
64-row GEMM with block scales folded into weight fragments	down and output projections 102 → 73 μs per call at 64 rows (four k-splits without register spills)	ADOPTED
Block-scaled FP8 W8A8 prefill GEMM	About 190 TFLOPS vs about 154 for BF16 hipBLASLt plus dequantization; 2K prompt prefill 709 → 556 ms	ADOPTED
Several prompts per prefill pass	Bit-identical outputs; eight short prompts at C=8: 157 → 193 tok/s	ADOPTED
Copy-engine prefill all-reduce	Bit-identical outputs; copies run at full speed beside a GEMM with no GEMM slowdown; 2K prompt prefill 553 → 481 ms	ADOPTED
Lookup drafting with 16-row verification (v9)	Copy-heavy edits 241–257 → 402–445 tok/s at 14–15 tokens per step; standard harness within 1%, identical outputs	ADOPTED
Host-memory prefix cache (v9)	Follow-up turns 756–786 → 94–95 ms to first token; regenerating an answer 783 → 54 ms	ADOPTED
hipBLASLt FP8 GEMM for prefill	340–370 TFLOPS with per-tensor scales, but 128-element and 128 × 128 block scale modes are not yet supported on this target in ROCm 7.1	REPLACED BY OWN KERNEL
All-reduce pushed from the GEMV epilogue	Slower than a separate all-reduce on this PCIe topology	NOT ADOPTED
Shared-activation GEMM staged in LDS	LDS use per workgroup limits occupancy; at 48–64 rows it ran 1.7–2.0× slower than the per-wave kernel, equal at 32	NOT ADOPTED
Re-quantizing the drafter from its BF16 release	FP8 with 16 × 128 blocks and INT4 both match the existing FP8 drafter's acceptance	NOT NEEDED
Causal attention inside the draft block	Same acceptance as the trained bidirectional attention	NOT ADOPTED
Replacing the drafts on short lookup matches	A 3-token threshold lowered 8K code tokens per step 4.11 → 3.92; at 12 tokens a summary turn quoting its source fell 3.71 → 3.53	THRESHOLD SET TO 32
Wide verification without acceptance tracking	+0.7 ms per step on the natural English task, where the extra drafts are rarely accepted	NOT ADOPTED
Snapshot by prefilling the last prompt token separately	+31 ms to first token, and outputs no longer matched v8	REPLACED BY IN-SCAN SNAPSHOT
Copying the snapshot synchronously at admission	Eight concurrent requests 2.6% slower	REPLACED BY SIDE-STREAM COPY

Technique	Measured effect	Status
DSpark drafter	The DFlash2 authors report higher acceptance than DSpark on Qwen 27B; DSpark's confidence-based depth is covered by adaptive verification	NOT ADOPTED

Correctness

- A float32 CPU reference of the full model checks every kernel change. With BF16-activation prefill, greedy output matches the reference token for token on the three reference prompts (40 / 40, 64 / 64 and 64 / 64 tokens), and the DFlash2 drafter's first block matches its own float32 reference exactly.
- Speculative decoding is lossless: greedy output with MTP and with DFlash2 is identical, and so is output with batched and unbatched prefill and with either all-reduce path.
- Sampling was checked statistically: first-token distributions against exact distributions from full logits (χ^2 p = 0.15–0.89 across temperature, top-p, min-p and repetition settings), and speculative against non-speculative sampling over 2,000 samples each (two-sample tests on tokens 2–6 and their joint prefixes, p = 0.17–0.97).
- Both GPUs draw identical counter-based random numbers, so every sampling and acceptance decision agrees across ranks without extra communication.
- v9 requests that miss the prefix cache match v8 word for word: 42 of 42 standard-harness outputs, 24 of 24 natural-task outputs and the cold turns of the multi-turn test. The lookup kernel matches a host reference implementation on 2,400 cases.
- On a cache hit only the rest of the prompt is prefilled, as a separate group, and GEMM rounding depends on the row count; one turn of the multi-turn test diverged from the cold computation at a point where two continuations were almost equally likely. Wide verification evaluates the output head with fp16 activations, so greedy output can differ at near-ties when it is used, as it already could in v8 with two or more concurrent requests.

Next steps

- Snapshots at message boundaries, so conversations that share a system prompt also share its cache entry.
- Wide verification at three and four concurrent requests.
- Long contexts: an FP8 KV cache and more attention splits. At 61K tokens of context a decode step takes 6.8 ms (about 18%) longer than at 2K.
- Chunked Gated DeltaNet prefill. The prefill recurrence is serial per token today, about 1.5–2.3 ms per layer for a 2K prompt; a chunked delta-rule formulation turns most of it into matrix products.
- Mixed prefill and decode batches, so running requests keep decoding while new prompts are processed.
- Vendor block-scaled FP8 GEMMs for prefill once hipBLASLt supports 128-element scale modes on gfx12.
- More models and more ROCm targets on the same engine structure.

r9k is developed by unassuming.ai. Technical report, October 2026, updated 28 October 2026 for v9. Same-machine figures come from the benchmark records of each run; published figures come from the linked sources. The Qwen3.8-27B chat and API run on r9k at unassuming.ai.