Refutation-Driven Performance Engineering: An Empirical Study of a Multi-Week GPU Kernel Campaign

Hanzo AI Research

hanzo-kernel Paper 08 — part of the *One Source, Every Backend* series

Abstract

We report an observational study of a single multi-week campaign to close the inference-throughput gap against `llama.cpp` on three accelerators (AMD gfx1151 RDNA3.5 APU, NVIDIA GB10 Blackwell, Apple M4 Max), conducted inside a one-source GPU kernel DSL. The campaign's primary artifact is not a kernel but a *refutation log*: twenty-five numbered hypotheses, most of them plausible and several of them our own, each killed by a cheap decisive experiment. We formalize the practice behind it. First, the quantified outcomes: a Vulkan-prefill *win ladder* of 212 → 1734 tok/s (8.2× of the gap closed in a single ∼36-hour release cadence, from 11.6× behind the reference to 1.40×), decomposed per lever; a CUDA decode gap closed 1.23× → 1.003× by a single model-eligibility list entry with *zero* kernel work; and an instrument fix that collapsed prefill benchmark noise from ±42% to ±0.8% (52×) and thereby made the second half of the win ladder measurable at all. Second, three algorithms that recur across the log: a multi-fidelity *Gate Ladder* that rejects candidates cheapest-first (free static rejection → bit-exact oracle → sustained per-op timing → in-engine wall → coherence); a *decisive-experiment* selection rule (the cheapest single-variable control that can falsify a live hypothesis); and *refutation-log-as-prior*, negative constraints injected into later work, which provably prevented at least nine documented re-chases. Third, the measurement doctrine as testable claims (same-run ratios only; sustained-vs-sustained; contended-timing refusal; warm-microbench distrust). We close by describing how the discipline was operationalized as a 22-case language-model evaluation whose distractors are the campaign's own refuted theories. We do not overclaim the standing: across the measured fronts the campaign holds two wins (ROCm decode 1.07×; a 470× margin over the fork's upstream on Metal), one parity (CUDA decode), and three open gaps (Vulkan prefill/decode, CUDA prefill) — "beat everywhere" is the objective, not the achieved state. This is an $N{=}1$ study with no control group; we state that boundary explicitly.

1 Thesis: the refutation is the deliverable

Conventional performance write-ups report wins. This campaign's ledger is dominated by *kills*: of the levers seriously proposed and evaluated, the majority were refuted, and refuting them was the value produced — because a refuted lever, recorded with its mechanism, removes an entire branch of future work. The campaign ran against a fixed external baseline (`llama.cpp`, the reference LLM inference engine) on the same silicon and the same model files, so every claim reduces to a measured ratio on shared hardware. The engineering question was never "is our kernel fast?" but "*which* of these competing stories about the bottleneck survives contact with an instrument?"

The stakes are concrete. On the AMD gfx1151 APU (evo), our Vulkan prefill began at 212 tok/s against a same-box `llama.cpp` figure of 2459 — an 11.6× deficit. Twenty-five refutations

later it stood at 1739.6 ± 7.4 vs 2441.9 ± 22.2 same-run, $1.40\times$ behind. The distance between those endpoints is a sequence of decisions, and most of the decisions were *not to do the attractive thing*. This paper is about how those decisions were made and what they cost.

Environment. Three self-hosted boxes, each its own backend: **evo** — AMD Ryzen AI Max+ 395 / Radeon 8060S, gfx1151 (RDNA3.5 APU, unified memory), bare-metal Linux 6.17, ROCm 7.2.4, Vulkan via RADV; **spark** — NVIDIA GB10, compute capability 12.1 (sm_121, Blackwell), aarch64; **dbc** — Apple M4 Max, 40-core GPU, macOS 26. Models: `zen-eco-4b` (a Qwen2-3B derivative) Q4_K_M, and Qwen3-30B-A3B-Instruct Q4_K_M (18.5 GB). Reference: `llama.cpp` (Vulkan/CUDA/Metal/HIP backends) and its `llama-bench`, run on the identical box and GGUF file. All comparisons in this paper are same-box, same-model; cross-box numbers are labelled as such.

2 The refutation dataset

Table 1 is the study's core observation: a census of the campaign's hypotheses, classified by the *kind* of story each told and annotated with the decisive experiment that settled it, its approximate cost to kill, and what killing it saved. We use four hypothesis classes:

- **mechanistic-kernel** — a claim that some change to a compute kernel (fuse these ops, cut this dispatch count, use tensor cores, enlarge the tile) will move the wall clock.
- **configuration-gap** — the bottleneck is not the kernel but a build flag, an eligibility list, a harness setting, or a dispatch-orchestration pattern.
- **measurement-artifact** — the "finding" was an instrument lie: a warm microbenchmark, a warmup-shape mismatch, contended timing, or mixed-phase telemetry sampling.
- **hardware-story** — a narrative about the silicon: thermal throttling, a DVFS utilization spiral, a sustained power limit.

Two structural facts fall out of the census. (1) The attractive hypothesis is nearly always mechanistic-kernel — "fuse it," "use the tensor cores," "cut the dispatches" — and it is the class most often refuted. (2) The largest single wins were *configuration-gap*: a Cargo feature, an enum arm, a warmup shape. Kernels get the glory; configuration eats the throughput. This asymmetry is the study's headline qualitative finding, and §3 quantifies it.

3 Five quantified findings

3.1 The win ladder: 212 → 1734 tok/s, per lever

Table 2 attributes the Vulkan-prefill gain to individual shipped levers. The kernel-level mechanism of each is recorded so the ladder is auditable, not a single aggregate number. Every rung is bit-exact-gated against the CPU oracle and coherence-checked before shipping.

The ladder's shape is itself evidence for the thesis. The first rung ($3.3\times$, Q6_K tiling) was found not by intuition but by a per-op profiler that named `mul_mat_q6k` as 76% of prefill GPU time — a quant type that had no tiled path because it appears in only $\sim$2 tensors per layer (`attn_v`, `ffn_down`) and was therefore invisible to attention aimed at the dominant Q4_K weights. Every later rung was a small, measured, single-variable change read out of the reference implementation's source and A/B'd in-engine. No rung came from a redesign.

3.2 The CUDA gap decomposition: configuration vs kernel

The CUDA front is the cleanest natural experiment separating configuration from kernel work, because its two phases resolved differently. On spark (GB10, `zen-eco-4b` Q4_K_M, same-run):

- **Decode** began at 63.3 ± 0.2 vs `llama` 77.6 ± 0.9 (1.23×). The profiler (`nsys`) showed ∼1,110 kernel launches per token against `llama`'s ∼1 graph launch. The engine already had a decode command-graph path, fail-closed behind a bit-exact gate and a *model-eligibility list*. `Model::Qwen` was simply not in the list (and decode RoPE read a host-side offset that freezes under graph capture — fixed by the device-positions pattern the eligible models already used). Adding the one arm collapsed launches to ∼1 per token and took decode to the CTO-frozen 81.9 ± 0.6 vs `llama` 82.15 ± 1.76 — 1.23× → 1.003× **parity, with zero kernel rewrites**.
- **Prefill** remains open at 5003.4 ± 23.0 vs `llama` 6079 ± 326 (0.82×) and is glue-bound: naive attention `softmax_f32` ∼13–15%, `badd_bf16` 10.5%, dtype casts 6.2%, `ucopy` 4.2%. Here the attractive lever was itself refuted (#25): building with the `flash-attn` feature made prefill 3.6% *slower* (4820.8 ± 166.3), because the paged bench's gather path forces naive attention regardless and Ampere-era flash-v2 kernels do not win on Blackwell. The residual is genuine kernel/dtype work (route `flashinfer` prefill, tighten bf16↔f32 discipline) — in progress.

The decomposition is stark: the *entire* CUDA decode deficit was configuration (a graph-eligibility list entry plus a device-placement fix), closed to statistical parity without touching a compute kernel; the CUDA prefill deficit is what is actually left for kernel engineering. When a fresh front is opened, the prior that "the gap is plumbing" pays before the prior that "the gap is a slow kernel."

3.3 Instrument variance collapse: ±42% → ±0.8%

For much of the campaign the prefill benchmark was unusable for A/B testing: repeated runs of the same configuration reported 728 ± 306 tok/s (±42%), and in the worst observations ±48–62%. The cause was a harness defect, not the GPU: warmup ran a fixed 32-token prefill while the measured batch was 512 tokens, so the first measured forward paid a one-time Vulkan pipeline compile and the warmup touched a different shape than the measurement. The fix (engine v1.7.60) warms the *measured* shapes; prefill went to 946.5 ± 7.6 (±0.8%) — a **52**× variance collapse.

The downstream effect is what makes this a headline rather than a footnote. A change worth +5% is invisible under ±42% noise (the ratio of signal to noise is 0.12); it is unambiguous under ±0.8%. Decode was always tight (±1.3%), so decode A/Bs were resolvable throughout — which is exactly why the decode wins landed earlier. Prefill A/Bs became resolvable *only after* the harness fix, and the entire second half of the win ladder (946 → 1734, the packed-LDS, Q6_K-coopmat, and single-buffer rungs, each a 1.2–1.7× op-level change) was A/B-decided on the honest bench. The instrument fix did not speed up a kernel; it made three subsequent kernel wins *legible*.

3.4 The 32-token warmup phantom: one artifact, three dead theories

The most expensive single artifact of the campaign was the same warmup-shape bug viewed from the other side. A profiler labelled every prefill batch `dispatch=1949` — because dispatch count is per-layer-ops × 36 layers, hence *token-count-independent*. The 32-token warmup batches therefore printed an identical dispatch count to the 512-token measured batches while doing 16× less work (byte counters: 334.6 vs 8364 MB). The visible symptom — "the first ∼5 forwards run ∼120 ms, then collapse to ∼523 ms" — looked like accumulating degradation, and spawned three competing hardware/leak stories:

1. a *DVFS utilization death-spiral* (mixed-phase telemetry read 29% busy @ 713 MHz for us vs 77% @ 2431 MHz for `llama` on the same silicon);
2. *accumulating engine state* / a per-forward memory leak;
3. a *sustained power throttle*.

All three were killed by two single-variable controls plus one counter read. Pinning the clock at 2900 MHz left the 512-token forward at 540 ms (throttle and downclock require the clock to move — it did not). Running `warmup=0` made forward #1 573 ms and every subsequent forward ∼545–561 ms flat (no accumulation; stable from the first forward). And the byte counters showed the "fast" forwards were simply smaller. The tell, in hindsight, was that *the fast-window length always equalled the warmup count.* One mislabelled invariant — treating a shape-independent dispatch count as workload identity — manufactured three plausible mechanisms and consumed multi-agent effort until the counters ended it. The lesson entered the log as a rule: *dispatch count is not workload identity; check tokens and bytes before comparing two batches.*

3.5 Optimization velocity: the campaign clock

If the tooling of §4 has a reason to exist, it is velocity: the loop must close gaps faster than unaided hand-tuning, or it is ceremony. The campaign's release history is the direct evidence, and its timestamps are real (Table 3, from the `ml` and `engine` git logs). The Vulkan-prefill ladder of §3.1 — 212 → 1734.4 tok/s, 8.2× of gap closed — shipped across nine tagged releases spanning ∼36 hours, and its four consecutive coopmat rungs (946.5 → 1734.4) landed inside a single window of ∼5.5 hours (11:28–14:58 on 2026-07-18). The CUDA decode close to parity (§3.2) shipped the same evening. Twenty-five numbered refutations gate this cadence, each with a cost-to-kill of minutes to hours (Table 1).

The honest contrast. It is tempting to set this cadence against the maturity of the baseline, and the comparison must be made carefully. `llama.cpp`'s Vulkan matmul lead was built incrementally by many contributors over roughly two years: its Vulkan shader path dates to mid-2024, cooperative-matrix support to December 2024 (PRs #10206/#10713), the DP4A MMQ + Q8_1 quantization shader to March 2025 (PR #12135), and a `mul_mm` shader cleanup to September 2025 (PR #15987), among others — dates read directly from the local `llama.cpp` checkout's git history. We closed 8.2× of the gap to that artifact in ∼36 hours. This is **not** a claim that our method is hundreds of times more productive than theirs: the comparison is method-versus-method *on different starting points.* They pioneered the kernels with no map; we chased, using their shipped source as the map (the BK, tile, and buffering constants were read out of their code, then A/B'd on our silicon). **Chasing a known-good target is strictly easier than pioneering it**, and the velocity figure reflects that asymmetry as much as it reflects the loop.

The falsification path. The velocity claim — that the loop out-paces hand-tuning — is falsifiable, and two controlled experiments are the honest way to settle it, both listed here rather than assumed. (a) The in-flight automated schedule search [4] is machine-search against our own hand-search on the *same* kernels: if the machine cannot match or beat the hand-picked schedules per unit compute, the loop's marginal value over an expert is refuted. (b) A from-zero replication on an untouched front (Metal, currently blocked by an offline box) measures time-to-parity with no prior map on that backend; a slow replication would show the 36-hour figure was mostly borrowed from the reference, not generated by the loop. Until those run, the honest statement is narrow: *on*

this campaign, the measured cadence out-paced this team's prior hand-tuning, and the mechanism (cheap refutation plus a map) is legible in the timestamps.

4 Three algorithms

The practices above are not ad hoc; they instantiate three procedures that recur throughout the log.

4.1 The Gate Ladder

Verification is multi-fidelity: each candidate passes through gates ordered by increasing cost, and is rejected at the cheapest gate that can reject it. Free static analysis (compile + shader statistics) filters occupancy-killers and register spills before any GPU time is spent; the bit-exact oracle enforces correctness as *viability* (an incorrect kernel is not a slow candidate, it is a non-candidate); sustained per-op timing on the honest instrument decides "is it a win"; the in-engine wall A/B and a coherence check confirm the win survives to the deployed regime.

Algorithm 1 GateLadder — multi-fidelity acceptance, cheapest rejection first

Require: candidate schedule C, shape s, device d, incumbent time $t_\star$

1: $(spill, lds, occ) \leftarrow \textsc{Compile+ShaderStats}(C, d)$ ▷ free; no GPU
2: **if** $spill > 0$ **or** $lds > \textsc{OccBudget}(d)$ **then**
3: **return** Reject("static")
4: **end if**
5: $y_C \leftarrow \textsc{Run}(C, s, d)$; $y_{\text{ref}} \leftarrow \textsc{CpuOracle}(s)$
6: **if** $\textsc{ScaleRel}(y_C, y_{\text{ref}}) > \tau$ **then**
7: **return** Reject("not bit-exact") ▷ viability, not penalty
8: **end if**
9: $t_C \leftarrow \textsc{SustainedPerOp}(C, s, d)$ ▷ median of 25, ±0.4%, in-engine
10: **if** $t_C \geq t_\star$ **then**
11: **return** Reject("no op-level win")
12: **end if**
13: $(w_C, w_\star, w_{\text{ref}}) \leftarrow \textsc{SameRunWall}(C, \text{incumbent}, \text{llama})$ ▷ ±0.8%
14: **if** $w_C \not> w_\star$ **then**
15: **return** Reject("no wall win")
16: **end if**
17: **if not** Coherent(C) **then**
18: **return** Reject("incoherent output")
19: **end if**
20: **return** Accept(C) ▷ ship: bit-exact ∧ op-win ∧ wall-win ∧ coherent

The ladder's economic point is the ordering. Static rejection is free and, empirically, disqualified a large fraction of hand-tried variants (register spills, LDS over the occupancy budget) before they ever reached the GPU. Line 9 is deliberately *not* a cache-warm microbenchmark (§5); the honest per-op timer is the first gate that costs real GPU seconds, and it is placed after the two free/cheap gates for that reason.

4.2 Decisive-experiment selection

Given a set of live hypotheses, the campaign repeatedly chose the *cheapest single-variable experiment that could falsify the most of them at once.* The pinned-clock control (§3.4) is the exemplar: one knob, minutes, and it bears on both the throttle and the downclock stories simultaneously. The rule prefers controls that change exactly one variable, because a confounded experiment cannot cleanly kill anything.

Algorithm 2 DecisiveExperiment — maximize hypotheses-killed per unit cost

Require: live hypotheses H, available experiments X

$best \leftarrow \varnothing$; $score_\star \leftarrow \infty$
for $x \in X$ **such that** x varies exactly one factor **do** ▷ single-variable controls only
  $K_x \leftarrow \{\, h \in H : x \text{ can falsify } h \,\}$ ▷ the kill set
  if $|K_x| = 0$ **then**
    continue
  end if
  $score \leftarrow \textsc{Cost}(x) \,/\, |K_x|$ ▷ cost per hypothesis killed
  if $score < score_\star$ **then**
    $score_\star \leftarrow score$; $best \leftarrow x$
  end if
end for
return *best* ▷ e.g. pin-clock kills throttle *and* downclock in one run

4.3 Refutation-log-as-prior

A refutation is only worth its cost if it is not re-purchased. Every kill was written to a durable log with its mechanism, and subsequent task briefs were seeded with the log's negative constraints: forbidden re-chases and masked dead operators. This converts a one-time experiment into a permanent search-space restriction.

Algorithm 3 SeedBrief — inject refutations as negative priors

Require: task T, refutation log L

$forbid \leftarrow \varnothing$; $deadOps \leftarrow \varnothing$
for $r \in L$ **do**
  if $\textsc{Class}(r) \in \{\text{mechanistic-kernel}, \text{hardware-story}\}$ **then**
    $forbid \leftarrow forbid \cup \{(\, r.\text{lever},\ r.\text{mechanism}\,)\}$
  end if
  if $r.\text{outcome} = \text{regress}$ **then**
    $deadOps \leftarrow deadOps \cup \{r.\text{operator}\}$
  end if
end for
return $\textsc{Brief}(T,\ \textbf{forbid}{=}forbid,\ \textbf{mask}{=}deadOps)$

The log records at least nine re-chases this prior provably prevented, each carrying an explicit "do NOT re-chase" marker traceable to a prior measurement: coopmat on gfx1151 (killed, then partially un-refuted only when the access pattern — not the tensor cores — was identified as the cause); `BM`≥128 tiles (2.4× slower, occupancy collapse); `BK`=64 on the coopmat path (llama's override read correctly); the RN4 / BM=64 tile family (best occupancy, worst time); f16 accumulators

(LDS-limited, not register-limited); the per-shape tile family (skinny projections are 1.9% of Q4_K time); the Q4_K coalesced-cooperative rewrite (already at 83% of peak); dispatch/instruction-count reduction as a *perf* lever ("10 for 10 — counts never predict wall-clock on this decode path"); and Ampere-era `flash-attn` v2 on Blackwell (3.6% slower). Each avoided re-chase is a multi-hour experiment not re-run.

5 The measurement doctrine, as testable claims

The doctrine is four falsifiable rules, each earned by a specific scar.

Same-run ratios only. The same `llama.cpp` on the same box and model reported prefill figures of 2390.1, 2404.1, 2417.8, 2441.9, 2459, 2465.3, and 2486 tok/s across the campaign's days — a $\pm 2\%$ spread from run conditions alone. A ratio computed from our number today against a `llama` number from last week is therefore uninterpretable; only both sides measured in the same session count. Every ratio in this paper obeys this.

Sustained-vs-sustained. On the gfx1151 APU, a kernel's *burst* per-op time ($\sim$416 μs) and its *sustained* per-op time ($\sim$1634 μs) differ by $\sim 4\times$ — an intrinsic power-limiting swing confirmed under a pinned clock. Comparing a burst measurement of one variant to a sustained measurement of another manufactures a $4\times$ phantom. All A/Bs compare sustained to sustained.

Contended-timing refusal. When a foreign 7 GB job held the GPU at 94% utilization, the candidate-vs-baseline A/B returned "candidate $\approx$ baseline." The correct report is not "refuted" but "blocked": symmetric contention does not make a masked signal fair, it makes it *absent*. Contention-independent facts (compiles, engages, produces correct output) are still reported as fact; the timing is deferred to an idle window.

Warm-microbench distrust. For a memory-bound kernel the cache-warm microbenchmark systematically lies. The dense `dp4a` matvec measured 74 $\rightarrow$ 255 GB/s ($3.4\times$) with weights hot in the 32 MB last-level cache, but only 74 $\rightarrow$ 97 GB/s ($\sim 1.3\times$) in-engine, where decode reads each weight once, cold, from unified memory. In the other direction, a warm microbenchmark rated the f16 coopmat GEMM at 0.07–0.20$\times$ of `dp4a` and it was written off — yet with a coalesced access pattern the same matrix-core path became the shipping prefill default. Benchmarks are proxies for deployment; when they disagree, deployment governs.

6 Operationalizing the discipline as an evaluation

The campaign's reasoning discipline is itself a measurable property, and we operationalized it as a 22-case multiple-choice evaluation (`roofline`) distilled directly from the log. Every case is a real decision point: the evidence shown is what was on the table at that moment; the gold answer is what hardware measurement subsequently proved; and the distractors are the *actual* refuted theories from Table 1 — power throttling, occupancy-maximalism, "port what the reference does," warm-microbench trust, contended-timing acceptance, dispatch-count-as-identity. Six of the 22 are *traps*: the prompt itself asserts a false premise that the provided evidence contradicts, testing whether a model verifies before it complies. Ground truth is hardware-verified and unpublished, so the set is uncontaminated by pretraining.

Beyond accuracy, two post-hoc metrics fall out of the standard results log, whose per-item token usages are already recorded:

- **overthink index** — mean completion tokens on *correct* answers. A discipline that trusts a decisive measurement answers in hundreds of tokens; an overthinker burns thousands and, on the trap cases, sometimes reasons its way back *into* the refuted theory.
- **trap resistance** — accuracy on the trap subset alone, where the prompt's own framing is adversarial.

This turns "measure, don't theorize" from a slogan into a benchmark: the same skill the campaign rewarded in engineers is scored, per model, on held-out decision points. It is the pilot instance of a distillation pipeline (a companion paper develops the loop in full).

7 Limitations

This is an observational $N{=}1$ study and we do not overclaim from it.

- **No control group.** We did not run a parallel team practising non-refutation-driven engineering on the same problem. We cannot attribute the $11.6\times \rightarrow 1.40\times$ compression to the methodology versus the specific people, tools, or the mere passage of time. The claim is that the recorded decisions were sound given the evidence, not that an alternative method would have failed.
- **Single campaign, one team.** All data come from one multi-week effort by one group against one reference implementation. The hypothesis-class distribution (mechanistic-kernel dominant, configuration-gap highest-yield) may not generalize beyond low-level GPU work against a mature baseline.
- **Hardware specificity.** The mechanisms are tied to particular silicon: gfx1151's unified memory and LDS-limited occupancy, GB10's Blackwell scheduling, the APU's utilization-slaved DVFS. A lever's sign can flip across devices (§on the RN4 tile; the Q4_K coalescing reversal), so the specific verdicts do not transfer even though the *method* may.
- **Survivorship in the log.** The log records hypotheses that were seriously evaluated. Ideas discarded before measurement, or never conceived, leave no trace; the census is of considered hypotheses, not of the full space.
- **The win ladder mixes measurement bases.** Its early rungs predate the instrument fix of §3.3; we mark this in Table 2 rather than silently splicing two noise regimes. The endpoints (212 and 1734) are individually defensible, but the intermediate slope should be read with the basis change in mind.

8 Conclusion

Across a multi-week campaign the reliable path to throughput was not a better idea about kernels but a cheaper way to be wrong about them. The largest wins were configuration gaps that a profiler named in minutes; the most expensive costs were measurement artifacts that dressed as hardware physics; and the most durable asset produced was a log of refutations that stopped the same attractive mistakes from being bought twice. The three procedures — a cost-ordered gate ladder, a decisive-experiment rule, and refutation-as-prior — are simple, and their simplicity is the point: they make the discipline auditable and, as §6 shows, measurable. The companion paper

closes the loop by turning the same gate ladder into the selector of an automated search, and the same decision log into training signal.

Finally, the standing must be stated without inflation (Table 4). The campaign holds two wins (ROCm decode 1.07× over `llama`-ROCm; a 470× margin over the fork's own upstream on Metal), one parity (CUDA decode, 1.003×), and three still-open gaps (Vulkan prefill 0.71×, Vulkan decode 0.55×, CUDA prefill 0.82×), with the Metal-vs-`llama` front blocked by an offline box. "Beat `llama` everywhere" is the standing objective, not the achieved state, and this paper claims only what the same-run measurements support.

References

[1] G. Gerganov and the `ggml/llama.cpp` contributors. *`llama.cpp`: LLM inference in C/C++.* https://github.com/ggml-org/llama.cpp. Reference implementation and `llama-bench` baseline; `ggml-vulkan.cpp` coopmat pipeline referenced by line in §2.

[2] Tracel AI. *CubeCL: multi-platform GPU compute in Rust.* https://github.com/tracel-ai/cubecl. The DSL substrate lowered to SPIR-V / HIP / PTX / MSL.

[3] Hanzo AI Research. *One Oracle, Every Backend: The CPU Bit-Exactness Law and How It Is Enforced.* hanzo-kernel Paper 07.4. The bit-exact gate of Algorithm 1.

[4] Hanzo AI Research. *Evolutionary Schedule Search in a One-Source Kernel DSL.* hanzo-kernel Paper 09 (companion). The automated successor to the manual gate ladder.

Hypothesis	Class	Decisive experiment	Cost	What it saved
Cut router GPU→CPU→GPU id syncs (144/tok) speeds decode	mechanistic-kernel	Removed the syncs; `VK_PROFILE` decode A/B	minutes	Retired "sync count" as a decode lever (8.97→9.14, flat)
Fuse the MoE router op-chain (softmax+argsort)	mechanistic-kernel	Fused to one kernel; dispatches 55272 → 43656 (−21%); measure t/s	hours	Retired "cheap-dispatch count" (decode ∼9→8.56, flat)
Fused GQA flash attention is the decode lever	mechanistic-kernel	Built `sdpa_blk` (roofline 284 GB/s); fused ON/OFF A/B	days	"Attention was never the lever" (2.2 vs 2.3 t/s, identical)
Prefill is submission-bound; a command graph is the fix	measurement-artifact	`fence_wait`=per-op GPU sum ⇒ GPU is computing	minutes	Graph saves ∼0.6% of prefill, not the 11.6×; redirected to the GEMM
Q6_K prefill needs a tiled GEMM (not a column matvec)	mechanistic-kernel	Per-op profile: `mul_mat_q6k` = 76% of prefill	minutes	The first real lever: 212 → 704 t/s
32KB accumulator spill caps occupancy	mechanistic-kernel	shaderstats before/after (scratch 32768 → 0)	free	occupancy 2 → 6; op 1722 → 1634 μs
`llama` 64×64 tile (RN4) is better (max occupancy)	mechanistic-kernel	Sustained per-op: RN4 1915 vs RN8 1634 μs	hours	Occupancy-maximalism killed; kept RN8
Sustained power throttle / DVFS spiral (29%@713 vs 77%@2431 MHz)	hardware-story	Pin clock 2900 MHz; forward still 540 ms	minutes	Throttle + downclock narratives both dead
Per-forward state accumulates ("120ms→523ms")	measurement-artifact	`warmup=0`: forward #1 = 573 ms, all flat	minutes	Killed the "memory leak" hunt
The "120ms fast window" is real GPU capability	measurement-artifact	Byte counters: 334.6 vs 8364 MB at equal dispatch count	minutes	32-tok warmup phantom; killed 3 theories at once
Brief: port `llama`'s BK=64 into our coopmat	configuration-gap	Read source: `:3968` overrides BK=32 for coopmat	minutes	Refuted the brief; BK=64 later tried and lost
f16 accumulators raise occupancy past 8	mechanistic-kernel	shaderstats: LDS-limited (20480B), VGPR 168 → 180	minutes	"`llama` does it" is not a mechanism
Double-buffer the K-tile to hide latency	mechanistic-kernel	Weight stream = 10.8% of peak (no latency crisis)	hours	Single-buffer: occ 6 → 8; 1383 → 1734 t/s
Coopmat (tensor cores) beats `dp4a` on gfx1151	mechanistic-kernel	Cache-cold in-engine: coopmat 4× worse e2e	hours	Killed until the access pattern, not the cores, was fixed
Bigger BM tile cuts cold-weight re-reads	mechanistic-kernel	`VK_Q4K_BM` A/B: BM128/256 2.4× slower	minutes	Occupancy collapse; BM64 is the sweet spot
Coalesced-cooperative layout will speed Q4_K matvec too	mechanistic-kernel	In-engine A/B: 115 vs 169 GB/s (1.46× slower)	hours	Lever value = distance-from-wall; Q4_K already at 83%
CUDA prefill needs the `flash-attn` feature	configuration-gap	Idle-window A/B: flash-ON 3.6% slower	hours	Redirected to `flashinfer` + glue, not v2 flash
CUDA decode is 1.23× because of a slow kernel	configuration-gap	`Model::Qwen` absent from graph-eligibility list	minutes	One arm ⇒ 1.23× → 1.003×, no kernel touched
Metal is 21× behind: the tiled GEMM was never written	configuration-gap	Read backend: `kernel_mul_mm_*` exists for every quant	minutes	Refused to port an existing kernel; re-measure first
Quantize-fusion (drop 96 dispatches/tok) is a perf win	mechanistic-kernel	Decode A/B: +2.2%, error bars touching	hours	Kept as cleanliness, not perf; "10 for 10" on counts

Table 1: The refutation census (representative subset of the log). "Cost" is order-of-magnitude engineer time to the decisive result. The modal class is *mechanistic-kernel* (the attractive story is almost always "write a better kernel"); the modal *outcome* is refutation. Several rows kill our own prior conclusions — the log is adversarial toward its author.

Ship	Lever	pp512 (t/s)	Op-level result
— start	column Q6_K matvec, no tiled path	212	11.6× behind `llama`
ml 0.11.64	Q6_K 2D-tiled `dp4a` GEMM	704	Q6_K op 7.5–29.8×; it was 76% of prefill
eng v1.7.60	warmup-shape bench fix + Q4_K coalesced coopmat	946.5 ± 7.6	noise ±42% → ±0.8%; prefill A/B now legible
ml 0.11.69	packed f16vec2 LDS (wide `ds_load`)	1264.8 ± 4.1	Q4_K GEMM 356 → 233 ms (1.53×); VGPR 204 → 168
ml 0.11.70	Q6_K `dp4a`→coopmat	1383.1 ± 5.0	Q6_K op 104 → 62 ms (1.67×)
ml 0.11.71	single-buffer, BK=32	1734.4 ± 4.4	Q4_K 232 → 181 ms (occ 6 → 8); Q6_K 62 → 38.7 ms

Table 2: Vulkan prefill win ladder on evo (gfx1151, `zen-eco-4b` Q4_K_M), CTO-frozen figures. 212 → 1734.4 is 8.2× shipped in one campaign; the same-run gap to `llama` went 11.6× to 1.40× behind. Current standing (ml 0.11.72 / engine v1.7.65): 1739.6 ± 7.4 vs `llama` 2441.9 ± 22.2 same run (0.71× of the reference). The 704 rung predates the warmup-shape harness fix (§3.3); 946.5 onward are on the ±0.8% honest bench. Every rung is bit-exact-gated and coherence-checked.

Release	Landed (PDT)	Change	Measured effect
ml 0.11.64	07-17	Q6_K tiled GEMM	pp512 212 → 704
ml 0.11.67 / eng 1.7.58	07-17 21:23	coalesced coopmat Q4_K	Q4_K GEMM 473 → 369 ms
eng v1.7.60	07-18 09:45	warmup-shape bench fix	noise ±42% → ±0.8%
ml 0.11.69 / eng 1.7.61	07-18 11:28	packed f16vec2 LDS	pp512 946.5 → 1264.8
ml 0.11.70 / eng 1.7.62	07-18 12:30	Q6_K coopmat	pp512 1264.8 → 1383.1
ml 0.11.71 / eng 1.7.63	07-18 14:58	single-buffer coopmat	pp512 1383.1 → 1734.4
ml 0.11.72 / eng 1.7.64	07-18 17:07	coalesced Q6_K decode matvec	decode 43.0 → 50.6
eng v1.7.65	07-18 17:18	qwen2 CUDA/ROCm decode graph	CUDA decode → 1.003× parity

Table 3: Release cadence with real git commit timestamps (`ml` and `engine` logs, PDT). Nine releases in ∼36 hours; the four coopmat prefill rungs within a single ∼5.5-hour window. This is the "why the tooling exists" evidence: the loop's throughput is in the timestamps.

Front (backend, phase)	Ours	Baseline, same box	Ratio	Standing
ROCm gfx1151, decode	70.9	`llama`-ROCm 66.5	1.07×	win
Metal M4 Max, decode	94.4	upstream `mistral.rs` 0.20	470×	win
CUDA GB10, decode	81.9 ± 0.6	`llama` 82.15 ± 1.76	1.003×	parity
CUDA GB10, prefill	5003 ± 23	`llama` 6079 ± 326	0.82×	open
Vulkan gfx1151, prefill	1739.6 ± 7.4	`llama` 2441.9 ± 22.2	0.71×	open
Vulkan gfx1151, decode	50.6 ± 0.1	`llama` 92.6 ± 0.2	0.55×	open
Metal M4 Max, vs `llama`	*blocked* — box offline; last (stale) reading decode 0.80×, prefill 0.047×			

Table 4: Standing scoreboard, CTO-frozen same-run figures (`zen-eco-4b` Q4_K_M except the Metal rows, Qwen3-30B-A3B). The 470× is measured against the fork's *upstream* `mistral.rs` on Metal, not against `llama`. Two wins, one parity, three open; Metal-vs-`llama` blocked.