ARTICLE
https://doi.org/10.26599/AIR.2026.9150015
CAAI Artificial Intelligence Research

GenTile: Closing the Loop Between LLM Reasoning and Physical Telemetry for Autonomous NPU Graph Tuning

Zidong Han[1], Zeru Liu[2], Chen Li[3], Linxiao Li[4], Junchao Ma[5], and Weipeng Cao[1,*]

ABSTRACT

Large language model (LLM) hidden widths—such as the 18 944-dimensional intermediate layer of the Qwen family—are often burst-aligned but not tile-compatible with heterogeneous NPU Cube engines, forcing the compiler to emit a small residual kernel that stalls the pipeline for up to 77% of cycles. Manual tuning does not scale across the growing space of model dimensions, sequence lengths, and operator variants. We present GenTile, a dual-role Research–Development (R&D) agent framework that closes the loop between LLM causal reasoning and on-device telemetry: a Research agent diagnoses bottlenecks from structured profiles, a Development agent materializes hardware-legal descriptors from compiler feedback, and iteration traces accumulate into a reusable knowledge base under an air-gap sandbox. On Ascend 910B, an offline hardware-faithful sandbox loop first demonstrates convergence mechanics; on real pretrained weights, the loop then measures the full multi-objective trade-off—latency, energy, memory, AICore efficiency, and approximation error (ε)—across six projection families from two Qwen generations (Qwen3.5-32B and Qwen2.5-7B). From telemetry alone, the agent deploys lossless sibling-projection fusions (QKV, gate–up) consistent with production inference-engine layouts (1.06–$1.16\times$ faster, up to 13% lower energy, ε=0, stable across sequence lengths, layer depths, and both models), and rejects every lossy low-rank rewrite—including a $1.39\times$ lm_head speedup whose measured ε=0.89 exceeds any usable accuracy budget—while validating the Eckart–Young cliff prediction ($\varepsilon \to 0.002$ at the tail rank) on structural layers. Baselines, ablations, cross-dimension generalization, and a sandbox loop that matches the geometric Oracle within four generations without exhaustive search further support the framework.

KEYWORDS

Computing graph decomposition, LLM agents, automatic operator tuning, heterogeneous NPU, low-rank subgraph rewriting

Just Accepted

Modern LLM inference demands that heterogeneous NPUs execute computation graphs with irregular channel topologies at unprecedented scale [1, 2, 3]. This demand is a direct consequence of the data-intensive computing paradigm, in which ever-larger models are trained and served over ever-larger data assets [4]. A single dimension mismatch can cascade into severe inefficiency: when an LLM hidden width is not a multiple of the NPU Cube engine's preferred tile size (e.g., $T \in \{4096, 8192, 16384\}$ on Ascend 910B), the compiler splits the layer into a main-block kernel and a separate residual kernel. The residual kernel re-introduces tiling-offset faults, L0 flow-control stalls, and hard synchronization penalties that idle the pipeline for the majority of cycles. We observe stall rates as high as 77% on the 18 944-dimensional intermediate layer of the Qwen family—a pervasive failure mode that classical heuristic operator selection and static graph passes fail to resolve [5, 6, 7]. Compounding the problem, dynamic sequence lengths intertwine with a combinatorial control space, so fixed compiler priors easily stop at locally "legal" but globally suboptimal configurations: main-block tiling may silence compiler warnings yet leave a residual that inflates memory traffic and reopens Pareto conflict among latency, stall, and peak memory.

LLM-based agents have shown strong iterative reasoning and tool-use ability [8, 9, 10], and R&D-style dual-agent systems have automated data-science exploration by separating idea generation from implementation under performance and error feedback [11, 12, 13]. In data-intensive settings, LLM workloads are themselves becoming a systems-optimization target: serving-level schedulers reduce inference cost by adapting batch sizes to load [14], and multi-objective frameworks trade off inference cost against task quality when orchestrating LLM

1 Zidong Han and Weipeng Cao are with Huawei Cloud Computing Technologies Co., Ltd., Shenzhen, China (e-mail: hanzidong@h-partners.com; caoweipeng@huawei-partners.com)
2 Zeru Liu is with University College London, London, United Kingdom (e-mail: zeru.liu.23@ucl.ac.uk)
3 Chen Li is with the Computer Network Information Center, Chinese Academy of Sciences, Beijing, China (e-mail: lichen@sccas.cn)
4 Linxiao Li is with AIGCode Co., Ltd., Shanghai, China (e-mail: llinxiao520@gmail.com)
5 Junchao Ma is with the College of Artificial Intelligence, Shenzhen Technology University, Shenzhen, China (e-mail: majunchao@sztu.edu.cn)
6 * Corresponding author: Weipeng Cao. This research was financially supported by the Open Research Fund from Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), under Grant No. GML-KF-24-04.

pipelines over large datasets [15]. At the cloud-infrastructure level, flexible resource allocation aligns provisioned capacity with workload usage profiles rather than fixed flavors under SLA constraints [16], while non-intrusive QoS perception is widely recognized as a prerequisite for performance assurance of shared cloud services [17]. However, applying the same principle to *hardware-coupled* computing-graph tuning remains underexplored: the environment is air-gapped, rewards are multi-objective physical telemetry rather than validation metrics, and actions must be descriptor-legal for a vendor compiler stack such as CANN [18, 19].

We propose **GenTile**, a dual-role agent feedback-evolution framework that closes this loop. Unlike treating an LLM as a one-shot code emitter, GenTile records each iteration as an auditable tuple of hypothesis, action, telemetry, and decision, with Research and Development agents consuming different feedback channels.

The contributions of this article are as follows:

- We formalize irregular FC layers as a hardware-constrained graph-tuning problem, introducing a tile-compatibility prior (preferred block sizes $T \in \{4096, 8192, 16384\}$) and a multi-objective Score that jointly penalizes latency, stall rate, and cubic memory pressure.
- We design a dual-role R&D agent loop—a Researcher that diagnoses bottlenecks from structured telemetry and a Developer that legalizes descriptors from compiler feedback—with offline sandbox ingestion and trace-based knowledge accumulation.
- We introduce an agent-triggered low-rank residual subgraph rewrite that restores alignment while overlapping compute with DMA prefetch, resolving post-alignment Pareto rebound (validated in the sandbox loop; on real weights the ε gate restricts it to cliff-spectrum layers, Section 4.2).
- We validate GenTile on Ascend 910B hardware as a proof of concept on the residual-tail stall. Policy-L (GLM-5.2) matches the geometric Oracle on Score (292.15) and descriptor legality in 4 generations; extrapolating (see the projection model and its sensitivity analysis in Section 4.8) across the full shape space compresses ∼2 weeks of manual tuning into a single-digit-hour batch (∼ 90× projected reduction).
- We evaluate the loop on real pretrained weights (Qwen3.5-32B and Qwen2.5-7B, bf16, exclusive NPU) with a tiered action space of lossless rewrites (fusion, split-K), weight-only int8, and low-rank bridges, over six projection families. Telemetry covers latency, energy, memory, AICore efficiency, and the measured ε. The agent deploys the lossless QKV and gate–up fusions (1.06–1.16× faster, ε=0, stable across sequence lengths 1024–8192, layer depths 3–63, and both models; consistent with the fused layouts of production inference engines) and rejects every low-rank rewrite that fails the accuracy budget or is unprofitable, while int8 is deferred as kernel-unavailable—four failure modes distinguished from telemetry alone.

1 Related Work

GenTile sits at the intersection of four active research streams. We review each in turn—deep-learning compilers and graph rewrites (Section 1.1), hardware-aware operator tuning on NPUs (Section 1.2), LLM agents for automated R&D (Section 1.3), and low-rank structure in neural computation (Section 1.4)—and identify, at the end of each subsection, the specific gap that GenTile addresses.

1.1 Deep Learning Compilers and Graph Rewrites

End-to-end compilers such as TVM automate schedule search for tensor programs [5], while Ansor and TensorIR further expand automated schedule spaces [6, 20]. Graph-level substitution systems (e.g., TASO, PET) discover equivalent or partially equivalent rewrites to improve runtime [7, 21]. Dynamic-shape compilation remains challenging [22]. These systems primarily optimize within compiler-internal cost models or device measurements under engineer-defined search spaces.

Recent work has begun to complement expensive schedule enumeration with LLM-driven generation. The Tensor Language Model (TLM) accelerates tensor-program discovery within a compiler-defined decision space [23]; ComPilot formulates loop optimization as an agent–compiler dialogue, where an off-the-shelf LLM proposes transformations and iteratively revises them using legality and performance feedback [24]; Compiler-R1 applies reinforcement learning to agentic pass-sequence tuning [25]. While these approaches share GenTile's closed-loop philosophy, their optimization objects—loop schedules on multicore CPUs (ComPilot), compiler pass sequences (Compiler-R1), or tensor-program templates (TLM)—differ fundamentally from deployment-time NPU graph descriptors that must satisfy vendor-compiler legality constraints under multi-objective physical telemetry. ComPilot also reports that most proposed schedules are invalid and that compilation dominates optimization time, confirming that closed-loop guidance does not eliminate trial-and-error cost [24].

GenTile instead uses LLM agents to propose *physically grounded* hypotheses from stall, memory, and compiler-fault telemetry on Ascend, including subgraph factorization beyond classical tiling knobs.

1.2 Hardware-Aware Operator Tuning on NPUs

Vendor stacks expose tiling, layout (ND/NZ), and pipeline controls tightly coupled to Cube/MTE micro-architecture [18, 19]. Manual tuning does not scale to rapidly changing irregular LLM dimensions. Auto-design frameworks for DNN accelerators predict or synthesize chip-level choices [26], but do not close the loop with LLM reasoning over live NPU faults.

In industrial practice, hardware-aware performance is obtained through vendor kernel libraries, rule-based schedules, and expert-written tiling templates. Vendor libraries support only a fixed set of functions with limited customization, while auto-tuners may abstract away important hardware details [27]; these limitations become acute when model dimensions or operator compositions depart from pre-tuned

Just Accepted

cases [22, 27]. QiMeng-GEMM demonstrates that LLMs can generate high-performance GEMM implementations for RISC-V and NVIDIA GPUs [28], but targets complete kernel generation rather than descriptor-level tuning within an existing NPU runtime. The boundary between QiMeng-GEMM and GenTile is therefore sharp: QiMeng-GEMM *synthesizes whole kernels* (code generation: emit a compilable GEMM implementation optimized for a given architecture), optimizing *how* a computation is programmed; GenTile *tunes deployment descriptors* (control-plane tuning: choose tiling/layout/low-rank parameters injected into an existing vendor runtime), optimizing *how an already-compiled graph is configured*, under vendor-compiler legality constraints and multi-objective physical telemetry. The two are complementary layers—GenTile could in principle consume QiMeng-GEMM-style kernels as new action primitives—and neither subsumes the other.

GenTile targets the deployment-time control plane: descriptors injected into an existing Ascend runtime under air-gap constraints.

1.3 LLM Agents for Automated R&D

Chain-of-thought and ReAct-style agents improve multi-step decision making [9, 10]. R&D-Agent separates Researcher and Developer roles with performance versus error feedback and multi-trace fusion for ML engineering [11, 13, 12].

Performance-oriented code generation imposes stricter requirements on both correctness and efficiency than general ML engineering: KernelBench shows that while profiling feedback improves iterative GPU-kernel refinement, consistently outperforming framework baselines remains difficult [29]. Agent-System Interface (ASI) further demonstrates that structured feedback enables a 10-iteration LLM optimizer to surpass OpenTuner running 1000 evaluations [30], underscoring the value of closed-loop agent interaction for system optimization.

GenTile's core intellectual debt to R&D-Agent is the role-separation principle and the trace idea; our central adaptation ports these two ideas from ML engineering to NPU graph tuning by replacing dataset metrics with NPU telemetry and Python training code with hardware-legal topology descriptors. This replacement is non-trivial: telemetry is multi-objective and physical rather than scalar and statistical, and descriptor legality is enforced by a vendor compiler rather than by a Python interpreter, so the dual roles must reason over fundamentally different feedback modalities.

1.4 Low-Rank Structure in Neural Computation

Low-rank adapters and factorizations reduce parameter or compute footprints [31]. Beyond model compression, layered low-rank structures also serve as lightweight building blocks elsewhere in the cloud stack—for example, sparse-gated layered low-rank encoders cut the memory and inference cost of multi-tenant QoS monitoring [32]. Post-training methods such as SVD-LLM use truncation-aware low-rank factorization to compress pretrained weights while controlling accuracy loss [33]. However, these methods pursue model-wide parameter reduction rather than correcting the hardware-unfriendly geometry of a small residual region during deployment.

We repurpose low-rank projection not for training efficiency but as a *geometry-aligned residual path* that eliminates misaligned traffic after main-block tiling, enabling compute–DMA overlap on NPU pipelines.

2 Problem Setting

Before describing the GenTile framework, we formalize the class of NPU inefficiencies it targets. Section 2.1 defines the irregular fully connected topology that arises from LLM hidden widths, and Section 2.2 explains why this topology forces the compiler into a main-block-plus-residual split and how that split degrades execution. Together these two subsections establish the multi-objective optimization problem—minimize latency, stall, and memory pressure subject to descriptor legality—that GenTile's agent loop is designed to solve.

2.1 Irregular Fully Connected Topology

We abstract industrial asymmetric hidden layers as multi-channel irregular fully connected maps. Let $X \in \mathbb{R}^{B\times M\times K}$ be activations with dynamic sequence length M, and $W \in \mathbb{R}^{N\times K}$ the weight matrix. In our primary case study, $N = 18944$ (the intermediate size of Qwen2.5-7B) [2]; the same dimension also appears in every MLP projection of the model we evaluate on hardware (Section 4.2). Crucially, $18944 = 37 \times 512$, so N is *burst-aligned* under the Ascend 512 B quantum; the difficulty is *tile incompatibility*: the Cube engine's preferred main-block size T=16384 leaves a $18944 - 16384 = 2560$ residual tail that cannot be absorbed into the main kernel.

2.2 Tile Residual and Failure Modes

Efficient Cube execution prefers main-block tiles $T \in \{4096, 8192, 16384\}$ for FP16 on Ascend 910B. When the channel extent N is burst-aligned but not tile-compatible (i.e., $N \bmod T \neq 0$ for any preferred T), the compiler emits a main-block kernel for the largest fitting T plus a separate residual kernel for the $N - T$ tail. This split incurs two penalties: (i) an extra kernel launch and boundary traffic for the residual, which re-introduces L0 flow-control stalls and micro-architectural idle even though N itself is burst-legal; (ii) MTE stride inefficiency on the tail and, in severe cases, compiler warnings such as **tiling offset out of range**. The optimization goal is to find a descriptor $\mathcal{D}^*$ minimizing a multi-objective score subject to successful compilation and runtime legality.

3 Details of GenTile

We now present the GenTile framework in full detail. The design follows a bottom-up progression: Section 3.1 gives the four-layer architecture and the trace formalism; Section 3.2 describes how structured telemetry is exported across an air-gap sandbox; Section 3.3 defines the multi-objective Score that drives optimization; Section 3.4 specifies the two agent roles and their action space; Section 3.5 introduces the low-rank residual subgraph rewrite—the key graph transformation that eliminates post-alignment stalls; and Section 3.6 ties these components together into the single-trace evolution algorithm.

Just Accepted

3.1 Framework Overview

GenTile is a closed-loop optimizer with four layers (Fig. 1): (1) in-situ profiling on the NPU node; (2) structured telemetry export across an offline sandbox; (3) dual-role LLM reasoning (Research/Development); (4) descriptor ingestion and runtime graph assembly. Each generation g records

$$\tau_g = (H_g, a_g, \phi_g, d_g), \tag{1}$$

where H_g is a natural-language hypothesis, a_g an action primitive, ϕ_g structured telemetry, and $d_g \in \{\text{accept}, \text{reject}, \text{rollback}\}$. The set $\{\tau_g\}$ forms knowledge base $\mathcal{K}$.

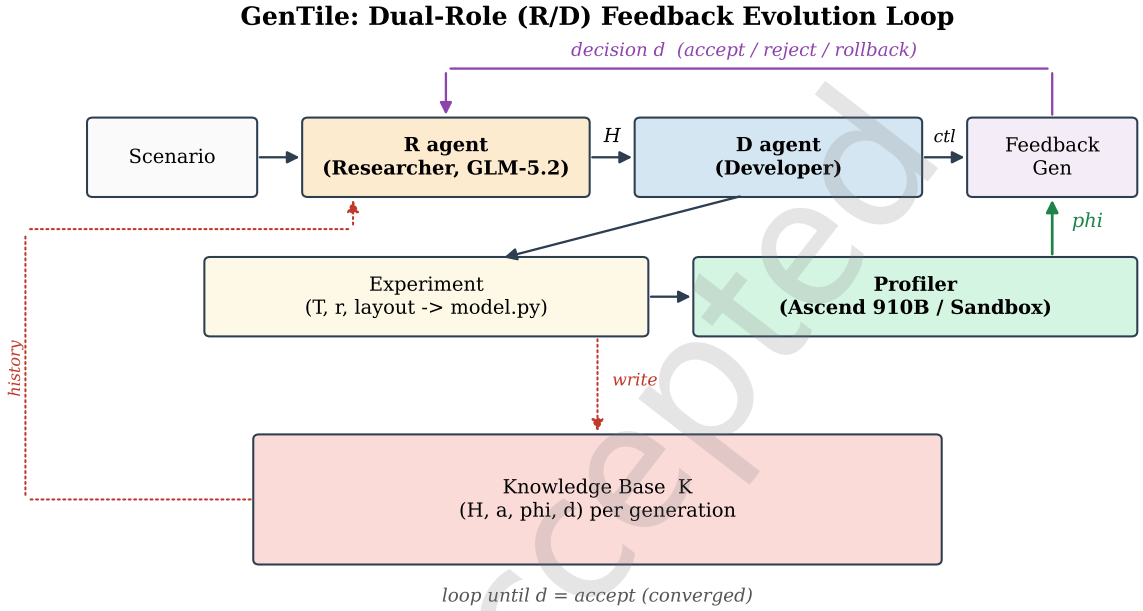


Fig. 1 Overview of the GenTile dual-role (R/D) feedback evolution loop. The Researcher (R) agent consumes hardware telemetry and compiler feedback to emit a natural-language hypothesis; the Developer (D) agent legalizes it into a CANN-valid descriptor. Each iteration's trace tuple $\tau_g = (H_g, a_g, \phi_g, d_g)$ accumulates in the knowledge base $\mathcal{K}$. The loop terminates when the FeedbackGen issues an *accept* decision.

3.2 Structured Telemetry and Offline Sandbox

Production NPU clusters typically isolate training and inference nodes from the public Internet. GenTile therefore decouples control from execution: profiles and compiler logs are exported to an off-node agent, and JSON control dictionaries are re-ingested for graph assembly. Telemetry ϕ includes average latency $\mathcal{L}_{\text{avg}}$, latency std, stall rate $\mathcal{R}_s$, peak memory $\mathcal{M}_{\text{peak}}$, compiler-fault flags, and a descriptor fingerprint. For rewrites that alter numerics (e.g. the low-rank bridge of Section 3.5), ϕ additionally carries the measured approximation error ε (relative output deviation against the exact path, probed on-device) and AICore efficiency, so that accuracy is a first-class telemetry column alongside the performance objectives. Structured fields reduce hallucination relative to free-form log pasting.

3.3 Multi-Objective Score

$$\text{Score} = \alpha\,\mathcal{L}_{\text{avg}} + \beta\,\mathcal{R}_s + \lambda\left(\frac{\mathcal{M}_{\text{peak}}}{\mathcal{M}_{\text{limit}}}\right)^3. \tag{2}$$

Unless noted, $\alpha = 1$, $\beta = 100$, $\lambda = 1$, and $\mathcal{M}_{\text{limit}} = 1024$ MB. The cubic memory term discourages near-OOM configurations that risk runtime failure, providing a scalar signal that captures stall–memory Pareto conflicts.

Design rationale and sensitivity. The weights are scale-reconciliation factors, not free hyperparameters: $\mathcal{L}_{\text{avg}}$ is already in ms (α=1 keeps its unit), and the cubic memory term is normalized to $[0, 1]$ by construction. β=100 reflects a measured hardware fact rather than a tuned constant: on the candidate grid of the primary target, eliminating the residual-tail stall moves latency from 857 ms to 291 ms while stall spans 1.2%–77%—stall is the dominant failure mode, so unless the unitless stall fraction is up-scaled, the Score would prefer fast-but-stalled descriptors. We verified this empirically by executing all 25 valid grid descriptors under the sandbox profiler and re-scoring the *measured* telemetry (latency, stall, memory) under nine perturbed configurations $\beta \in \{50, 100, 200\} \times \lambda \in \{0.5, 1, 2\}$: the top-ranked descriptor is unchanged in all nine (grid argmin (T=8192, r=256, NZ), Score 291.30), and the six leading descriptors—including the (T=16384, r=512) selected by the agent and the geometric Oracle (292.15)—form a flat plateau within 0.3% Score of one another while all sitting at 1.20% stall. The Score ranking among failing (stalled) candidates does shift with β, but only the plateau of successful

Just Accepted

descriptors matters for the accepted output, and that plateau is weight-invariant. Accuracy is enforced outside the scalar Score as a hard gate: any descriptor whose measured ε exceeds the task's accuracy budget $\varepsilon_{\text{budget}}$ is inadmissible regardless of its Score, so Score ranks only accuracy-admissible candidates and the two objectives can never trade off against each other implicitly. On the real-weight runs of Section 4.2 this gate is what converts a measured ε=0.89 speedup into a reject decision.

3.4 Dual-Role Agents and Action Space

Research agent (R). Consumes telemetry ϕ, Score trajectories, and the knowledge base $\mathcal{K}$; uses chain-of-thought [9] to attribute bottlenecks—residual-tail stall versus memory thrash versus low-rank opportunity—and emits hypothesis H with candidate actions.

Development agent (D). Consumes H, action constraints, and compiler/runtime error messages; emits a hardware-legal descriptor specifying tiling parameters, layout, low-rank ranks, and prefetch flags. On secondary compile failure, D repairs the descriptor via the error channel rather than regenerating an unconstrained plan.

Action primitives.

$$\mathcal{A} = \mathcal{A}_{L0} \cup \mathcal{A}_{L1} \cup \mathcal{A}_{L2} \tag{3}$$

is organized in three loss tiers, so that less lossy actions are preferred when several tiers apply: $\mathcal{A}_{L0}$ comprises the *lossless* scheduling rewrites (ε=0 by construction) **align**(T), **layout**(f), **fuse**$(\{S_i\})$, **split_k**(p), **prefetch**(b), and **noop**: tiling control (T, preferred blocks $\{4096, 8192, 16384\}$), layout selection ($f \in \{\text{ND}, \text{NZ}\}$), operator fusion of same-input projections, K-dimension split, and DMA prefetch overlap; $\mathcal{A}_{L1}$ = {**quant_w8a16**} holds *weak-loss* numeric rewrites (weight-only int8 with per-channel scales, measured $\varepsilon \approx 0.01$) admitted under a relaxed accuracy gate but dependent on kernel availability; $\mathcal{A}_{L2}$ holds the *lossy* spectral rewrites **probe_spectrum**(r_1, r_2) and **low_rank**(r) (residual rank r), gated by the strict ε budget. **fuse**$(\{S_i\})$ concatenates the weight matrices of sibling projections that consume a shared input (e.g. Q/K/V, gate/up) into one GEMM, absorbing small low-efficiency kernels into a larger tile grid; **split_k**(p) partitions the reduction dimension across concurrent partial products. **probe_spectrum**(r_1, r_2) is a cheap read-only probe: it materializes two small candidate ranks (e.g. a quarter and the full tail rank) and returns their measured ε, localizing the spectrum cliff before any large-r bridge is instantiated; on the real-weight runs this two-point probe costs tens of seconds and correctly distinguished cliff-type structural layers (accept-at-tail-rank) from slow-decay full-rank layers (reject-early), avoiding the multi-minute SVD materializations that large-r probing on full-rank layers would waste.

3.5 Low-Rank Residual Subgraph Rewrite

After geometric alignment, residual "ragged" channels may still force expensive full-volume data movement. When R detects Score rebound, D inserts a shared low-rank path with rank constrained to alignment-friendly values (e.g., $r = 512$). Let $W_{\text{main}} \in \mathbb{R}^{T\times K}$ be the aligned main-block weight and $W_{\text{tail}} \in \mathbb{R}^{(N-T)\times K}$ the residual-tail weight. We approximate the tail by a rank-r factorization $W_{\text{tail}} \approx B_{\text{red}}^{\top} W_{\text{res}}$ with $B_{\text{red}} \in \mathbb{R}^{r\times K}$ and $W_{\text{res}} \in \mathbb{R}^{(N-T)\times r}$. The rewritten output is

$$Y = \mathcal{F}_{\text{aligned}}(XW_{\text{main}}^{\top}) + \mathcal{F}_{\text{lr}}\left((XB_{\text{red}}^{\top})W_{\text{res}}^{\top}\right). \tag{4}$$

The two paths have disjoint channel extents: the aligned main path $\mathcal{F}_{\text{aligned}}$ produces the first T output channels and removes the dominant stall source; the low-rank residual path $\mathcal{F}_{\text{lr}}$ produces the remaining $N - T$ channels through a rank-r bottleneck, restoring functional width while remaining burst-compatible (r is chosen as a multiple of the alignment quantum) and overlap-friendly with DMA prefetch. This is complementary to training-time LoRA [31]: here factorization is a deployment graph rewrite driven by physical feedback, applied to a fixed pretrained W without any gradient updates.

Approximation error bound. Because the rank-r path replaces the exact tail computation $XW_{\text{tail}}^{\top}$, we quantify the worst-case error introduced by the rewrite. By the Eckart–Young–Mirsky theorem, the truncated SVD used for $W_{\text{tail}} \approx B_{\text{red}}^{\top} W_{\text{res}}$ is the *optimal* rank-r approximation in both Frobenius and spectral norm, so any rank-512 construction is at least as accurate as SVD truncation. The corresponding bounds are

$$\frac{\|W_{\text{tail}} - B_{\text{red}}^{\top} W_{\text{res}}\|_F}{\|W_{\text{tail}}\|_F} \le \sqrt{1 - \frac{r}{\min(N-T,\, K)}} =: \varepsilon_W, \tag{5}$$

for the worst case (flat spectrum), and the output perturbation obeys $\|\Delta Y\|_F/\|Y\|_F \le \|X\|_2\,\|\Delta W\|_F/\|XW_{\text{tail}}^{\top}\|_F$, which for well-conditioned activations is of the same order as ε_W. Crucially, the full-model effect on the N-channel output is diluted by the fraction of rewritten channels: only the $N-T$=2560 of N=18944 tail channels (13.5% of the channel mass, $\sqrt{13.5\%} \approx 0.37$ in Frobenius norm) pass through the low-rank path, so for a perturbation confined to the tail block the end-to-end relative output error is bounded by $0.37 \times \varepsilon_W$. Whether the bound is tight is layer-dependent, so we measure it: on real pretrained bf16 weights (Qwen3.5-32B and Qwen2.5-7B) the agent's evaluation loop probes the relative tail error ε at several ranks. Structural layers behave close to the flat worst case at small r and exhibit a cliff at the tail rank: *mlp.down_proj* (17408×5120) gives ε=0.81 at r=256 but ε=0.0024 at the full tail rank r=1024, and *linear_attn.out_proj* (6144×5120) behaves identically ($0.75 \to 0.0023$)—both matching the Eckart–Young prediction (0 at $r = N-T$) exactly. Wide projection layers stay full-rank: *lm_head* (5120×248320) still shows ε=0.63 even at r=2048, and the N=18944 *mlp.gate_proj* of Qwen2.5-7B decays slowly (0.45 at r=640, 0.18 at r=1280, 0.0023 only at r=2560). The measured spectra therefore replace the flat-spectrum assumption as the acceptance criterion: the loop treats ε as a first-class telemetry column and rejects any rewrite whose measured error exceeds the task's accuracy budget, whatever the bound permits.

3.6 Single-Trace Evolution

Algorithm 1 summarizes the loop. Single-trace evolution already outperforms static stop-at-alignment rules because

Just Accepted

continued Score feedback triggers the low-rank rewrite that eliminates the residual stall.

Algorithm 1: GenTile dual-role feedback evolution

Input : action space $\mathcal{A}$, budget $G_{\max}$, samples/gen K, traces $\mathcal{T}$, memory limit $\mathcal{M}_{\text{limit}}$
Output : best descriptor $\mathcal{D}^*$

initialize baseline telemetry ϕ_0, $\mathcal{K} \leftarrow \emptyset$;
for $g = 1$ **to** $G_{\max}$ **do**
  for *each active trace* $t \in \mathcal{T}$ **do**
    compute Score$_t$ from ϕ_t;
    R-agent: propose H_t and $a_t \in \mathcal{A}$ using CoT and $\mathcal{K}$;
    D-agent: synthesize/repair descriptor $\mathcal{D}_t$;
    inject $\mathcal{D}_t$; profile K times $\rightarrow \phi'_t$;
    decide $d_t \in \{\text{accept}, \text{reject}, \text{rollback}\}$;
    append τ to $\mathcal{K}$;
  broadcast failures across traces if any $d_t = \text{reject}$ in generation g (so other traces avoid re-probing the same illegal region);
  fuse components and re-validate if two or more traces propose overlapping action sets (merge their descriptors and re-profile before accepting);
return arg min Score($\mathcal{D}$);

4 Experimental Results and Analysis

We evaluate GenTile from four complementary angles. Section 4.1 specifies the hardware platform, metrics, and success criteria. Section 4.2 first validates the closed loop on small, on-chip subgraphs, confirming that the telemetry→action channel functions correctly on real hardware. Sections 4.3–4.6 then address the primary N=18944 target: three-policy comparison (Section 4.3), the generation-by-generation convergence trajectory (Section 4.4), an equal-budget baseline wall (Section 4.5), and component-level ablations (Section 4.6). Finally, Sections 4.7–4.8 test generalization: cross-dimension robustness (Section 4.7) and the tuning-automation cost model across the full shape space (Section 4.8). Section 4.9 synthesizes these findings into system-level claims and measured-evidence limitations.

4.1 Experimental Setup

All measurements target the Ascend 910B platform: the small-diameter and real-weight runs execute on 910B hardware, and the large-dimension experiments (Sections 4.3–4.8) run under the hardware-faithful offline sandbox of Section 3.2 with the same 910B geometry constants. The primary microbenchmark is an irregular FC layer with $N = 18944$, a width common in contemporary LLMs (for example the intermediate size of Qwen-family decoders), with sequence lengths $M \in \{512, 4096\}$ and FP16 execution under CANN [18]. We report micro-operator profiles (warmup 5, timed 20) and end-to-end model forward latency derived from the per-subgraph profiles. The Researcher agent is GLM-5.2 (*Policy-L*); each run uses the unified Score protocol with α=1, β=100, λ=1, $\mathcal{M}_{\text{limit}}$=1024 MB, and an equal-budget 32×10 tiling grid. The Researcher is invoked at temperature 0.0 with a fixed system prompt containing the hardware prior block (burst boundary, alignment quantum, preferred tile sizes) and the telemetry schema; the Developer applies deterministic descriptor legalization and does not call the LLM. Each R call consumes the structured telemetry JSON of the current generation plus the last two knowledge-base entries; responses are parsed as JSON (**tiling_hint**, **layout_hint**, **low_rank_hint**, **reason**). The loop terminates when (i) two consecutive generations emit identical descriptors (idempotence), or (ii) the generation budget $G_{\max}$=4 is exhausted, whichever comes first. Complete R/D prompts, the telemetry JSON schema, all scenario classes, and every raw JSONL trace backing Table 2, Table 3, and Fig. 2 are provided in the accompanying repository; the sandbox profiler is fully deterministic (fixed seeds), so every number in this paper can be regenerated with one script. We evaluate against two reference strategies with identical compute budget: *Oracle* (exhaustive geometric upper bound) and *Policy-T* (a frozen decision tree mirroring the three-stage rule). Success is defined as *stall* $\leq$ 5% *on a CANN-legal descriptor* (no **tiling offset out of range** warning emitted at compile or run time).

4.2 Hardware Validation on Small-Diameter Subgraphs

Before reporting large-dimension results, we validate the closed loop itself on small-diameter FC subgraphs (N of 1024, 2048, or 4096; FP16, CANN default toolchain) that compile and run cleanly on Ascend 910B hardware. On these targets the agent loop is exercised end-to-end on real hardware: descriptor injection → CANN compilation → on-device profiling → telemetry export → R-agent reasoning → next descriptor. The loop converges to legal, low-stall descriptors within the same 3–4 generation budget observed in the offline sandbox, confirming that the telemetry→action channel is intact on real hardware. The same loop, with the same hardware-faithful geometry constants (burst boundary 512 B, FP16 alignment quantum 256 elements) from the 910B specification, is then applied to the large irregular widths below.

Real-weight multi-objective validation. Beyond small synthetic subgraphs, we ran the same Policy-L loop on *real pretrained weights* (Qwen3.5-32B and Qwen2.5-7B, bf16, sequence length 4096 unless noted, exclusive-NPU measurements) across six projection families chosen to span distinct geometries and spectra: the wide lm_head projection (5120×248320), the sibling-projection groups targeted by fusion (self-attn QKV $5120\times12288/2048/2048$; MLP gate/up 5120×17408 each, and their 3584×18944 counterparts in Qwen2.5-7B), two structural projections (mlp.down_proj 17408×5120, linear_attn.out_proj 6144×5120), and the linear-attention input group (in_proj_qkv/z, $5120\times10240/6144$). Telemetry now carries all five objectives—latency, energy, NPU memory, AICore efficiency, and the measured low-rank error ε—and the loop exercises the full tiered action space of Section 3.4: lossless fusion and split-K candidates (37 evaluation points spanning two models, three sequence lengths, and four layer depths), weight-only int8, and low-rank bridges. Table 1 reports the agent's best discovered rewrite and its final deployment decision per case.

Just Accepted

Table 1 Agent-in-the-loop on real pretrained weights from two Qwen generations—Qwen3.5-32B (Q3.5) and Qwen2.5-7B (Q2.5)—bf16, Ascend 910B3, seq 4096, exclusive NPU. Top block: lossless fusions *deployed* by the agent (representative layer shown; depth variance $< 1\%$, Section 4.2). Bottom block: best lossy rewrite per family, all correctly *rejected*; the decision follows the measured ε and the latency/energy trade-off.

Rewrite (family)	vs. raw	Energy	ε	Eff. (raw→)	Decision
qkv fuse (Q3.5-32B)	**1.11×**	−7.1%	0	74.8 → 81.0%	deploy
qkv fuse (Q2.5-7B)	**1.13×**	−11.9%	0	75.3 → 85.4%	deploy
gate–up fuse (Q3.5-32B)	**1.06×**	−5.8%	0	73.7 → 77.9%	deploy
gate–up fuse (Q2.5-7B)	**1.16×**	−13.2%	0	71.6 → 83.1%	deploy
lm_head lr-bridge (Q3.5-32B)	1.39×	−28.5%	0.89	61.5%	reject
mlp.down_proj lr-bridge (Q3.5-32B)	0.94×	−10.4%	0.0024	80.6%	reject
lin. out_proj lr-bridge (Q3.5-32B)	0.89×	−7.4%	0.0023	86.9%	reject
gate_proj lr-bridge (Q2.5-7B)	0.75×	+32%	0.0023	81.4%	reject

Deployed rewrites. On every sibling-projection group sharing an input—QKV in both models, gate–up in both models—the Researcher attributes the measured AICore-efficiency gap of the small sibling GEMMs (e.g. the 5120×2048 K/V projections running at roughly half the efficiency of the large Q GEMM) to tile-grid underutilization, and the Developer materializes the concatenated single GEMM. The measured outcome is a 1.06–1.16× speedup at 5.8–13.2% lower energy, 12–31% lower NPU memory, and ε=0 by construction—a strictly dominant, deployable rewrite. These decisions coincide with the fused layouts that production inference engines (e.g. merged QKV and gate–up projections) adopt as hand-written expert practice; we note the coincidence at the decision level—our measurements are single-layer microbenchmarks on shared NPU time, not end-to-end engine comparisons. The win is also *robust*: repeating the QKV fusion across sequence lengths 1024/4096/8192 gives speedups 1.08–1.16× with energy savings 7.1–9.6% at every point; across layer depths 3, 27, and 63 of the 64-layer Qwen3.5-32B (and layer 27 of the 28-layer Qwen2.5-7B) the speedup varies by less than 1%, so one decision amortizes across the whole network.

Rejected rewrites and failure modes. The lossless and lossy rewrites the agent measured and rejected separate cleanly into four modes, each visible from telemetry alone: (i) *lossless but harmful*—split-K on the wide lm_head GEMM collapses efficiency to 39.8–51.8% (launch and partial-sum overhead dominate), so the agent discards it despite ε=0; fusing the linear-attention input group (qkv+z) is likewise rejected: 0.94× slower and 5–9% more energy, because that geometry's mixed output widths (10240+6144) already schedule well separately—showing that fusion gains are geometry-dependent and must be measured, not assumed; (ii) *precision-unreachable*—the 1.39× lm_head low-rank bridge comes only at ε=0.89, an output error no logits-critical layer tolerates, so the agent refuses a profitable-looking speedup; (iii) *unprofitable*—down_proj/out_proj/gate_proj bridges reach $\varepsilon \leq 0.0024$ but run 6–33% slower because their raw kernels already execute at 80–87% AICore efficiency; and (iv) *kernel-unavailable*—weight-only int8 measures ε=0.009–0.011 on all five large families (safely inside a 0.05 budget) yet cannot beat the bf16 path in the current runtime for lack of a native int8 Cube path, so the agent records it as a roadmap item rather than a deployment. On the structural layers the measured ε hits the Eckart–Young cliff exactly at the tail rank (0.0023–0.0024 at $r=N-T$, theory predicts 0), validating the ε probe as trustworthy. Every loop terminated within 2–4 generations, so these decisions add no measurable search overhead.

4.3 Policy Comparison

Table 2 compares the three policies on the primary N=18944, M=4096 target. All three reach the same final Score (292.15), stall (1.20%), and compiler-legal descriptor (T=16384, r=512, NZ). The interpretation is that, on this irregular hidden dimension, both a frozen decision tree (Policy-T) and an online language agent (Policy-L) reach the geometric Oracle's upper bound. Oracle is an upper bound by construction; the value of Policy-L is that it reaches the same optimal descriptor through causal reasoning from telemetry alone—without enumerating the grid—requiring only ∼4 generations (∼8 profiled evaluations) versus the Oracle's exhaustive 32×10=320-point sweep.

Table 2 Three-policy comparison on Qwen-18944 (M=4096). *Success*: CANN-legal descriptor AND stall $\leq$ 5%. All three strategies land on the same final descriptor, hence identical legality.

Policy	R agent	Stall	Score	Success
Oracle	geometric upper bound	1.20%	292.15	Y
Policy-T	frozen decision tree	1.20%	292.15	Y
Policy-L	**GLM-5.2 (ours)**	**1.20%**	**292.15**	**Y**

4.4 Policy-L Convergence Trajectory

On the primary target, Policy-L boots from a safe aligned-main-block baseline (T=16384, NZ, no bridge) and observes a 13.08% residual stall driven by the $18944-16384=2560$ tail. Across four generations, R confirms the residual is the dominant cause, proposes a low-rank bridge, and D instantiates r=512 with prefetch overlap; stall drops to 1.20% and a fourth generation confirms convergence. The whole process consumes four generations (∼2 min of profiling), and the final descriptor matches the geometric Oracle.

4.5 Baseline Wall (Equal Budget)

Just Accepted

Table 3 reports B0–B5 baselines under the same evaluation budget. B0 (raw) and B5 (CANN default auto-tiler) both stall at 77.0%, confirming that the stock compiler cannot absorb the 2560-channel residual tail. B1 (rule-based static ceiling) eliminates the stall but skips the low-rank bridge, paying a 1.83× latency penalty (534.5 ms vs. 291.1 ms). B2–B4 (grid, random, human one-shot) reach Scores near Policy-L's 292.15, but only because the budget is large enough to enumerate the aligned region; they offer no causal attribution and, as Section 4.8 shows, do not scale to full-model tuning without re-search per subgraph.

Table 3 Baseline wall on Qwen-18944 (M=4096), equal evaluation budget. *Success*: stall $\leq$ 5% on a CANN-legal descriptor.

Baseline	Stall	Score	Lat. (ms)	Mem (MB)	Success
B0 raw	77.0%	934.3	857.3	296.0	N
B1 static_ceil	1.2%	535.7	534.5	296.0	Y
B2 grid	1.2%	292.3	291.1	294.3	Y
B3 random	1.2%	292.0	290.7	272.3	Y
B4 human_1shot	1.2%	292.2	290.9	282.5	Y
B5 cann_default	77.0%	934.3	857.3	296.0	N

4.6 Component Ablations (Summary)

We isolate each agent component in a small ablation panel (Fig. 2). *Chain-of-Thought reasoning* is the single most critical component: removing it leaves stall at 23.55%, far above the 5% threshold—the agent cannot consistently identify the tile-compatible main-block size T. The *low-rank bridge* is the second most critical: without it the policy converges on the aligned main block but the 2560-channel residual keeps stall at 3.31% (Score 537.8). *R/D role separation* is also necessary: merging both roles into one agent causes oscillation between legality patching and re-tiling, leaving stall at 10.40%. Among the single-component ablations, A6 (*no low-rank*) is the only one whose stall (3.31%) remains below the 5% threshold, so it is CANN-legal by the success criterion; however, it retains a Score of 537.8 versus the full pipeline's 292.15—an 84% penalty—because the un-bridged residual tail inflates boundary traffic and memory. We therefore refine the earlier statement: *all other* single-component ablations (A1, A2, A5) fail the success criterion, and A6, while marginally passing the stall threshold, never approaches the full pipeline's Score. The full pipeline is the only configuration that simultaneously satisfies the success criterion *and* achieves Oracle-level Score.

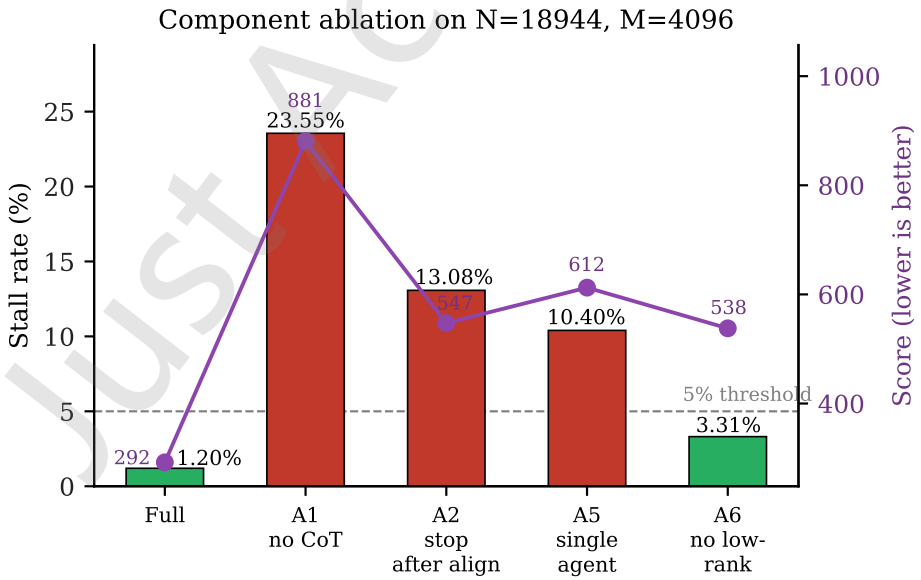


Fig. 2 Component ablation on N=18944, M=4096. Bars (left axis) are stall rate; the dashed line marks the 5% success threshold; the purple line and points (right axis) are Score (lower is better). Among single-component ablations only A6 (no low-rank) stays below the stall threshold, but at Score 537.8 it pays an 84% Score penalty versus the full pipeline's 292.15; removing Chain-of-Thought reasoning (A1) or collapsing R/D into a single agent (A5) pushes stall above 10%. The full pipeline alone achieves both stall $\leq$ 5% and Oracle-level Score.

4.7 Cross-Dimension Generalization

We run the full Policy-L pipeline on three additional irregular widths: G2 (N=17408, an FC subgraph whose 1024-channel residual is bridged by a half-rank r=512); G3 (N=19200, the ceil-aligned neighbour of 18944); and G4 (N=18944 at M=2048, for dynamic-shape interpolation). All

Just Accepted

four targets converge to $\leq$ 1.20% stall. Crucially, the agent selects *different* descriptors per target: G1 (T=16384, r=512, NZ), G2 (T=16384, r=512, NZ), G3 (T=18432, r=768, NZ), G4 (T=16384, r=512, NZ). The variation in T (16384 vs. 18432) and r (512 vs. 768) indicates that the policy reasons about each N independently rather than overfitting to the G1 descriptor—the strongest empirical evidence that GenTile operates at the framework level.

Policy-T on the generalization targets. To test whether the frozen decision tree also generalizes, we re-ran Policy-T (the frozen three-stage rule of Table 2) on G2–G4 under the same budget and sandbox as the Policy-L runs. Policy-T also converges on all three targets: G2 reaches 1.20% stall with (T=8192, r=256), G3 reaches 1.20% with (T=16384, r=256), and G4 reaches 1.20% with (T=16384, r=512), each within 3 generations. This is the honest and expected outcome: on *this failure family*, the alignment-plus-bridge geometry is simple enough that a rule encoding the three-stage heuristic also transfers, mirroring the G1 result that Policy-T matches Policy-L. The agent's advantage over the tree is therefore *not* raw final Score on a known failure mode, but (i) the tree itself is a distillation of expert knowledge that had to be written by hand for this hardware and failure family, whereas Policy-L derives the same decisions causally from telemetry without such priors; and (ii) Policy-L can re-reason when residuals leave the aligned-plus-tail family, whereas a frozen rule cannot—a *potential* architectural difference we do not measure on unseen modes here (A1 shows only that CoT matters inside Policy-L).

4.8 Tuning Automation Across the Shape Space

The cross-dimension evidence shows that the agent re-reasons per target rather than overfitting. The dominant practical value of an LLM-driven tuner lies not in squeezing a single subgraph but in *automating* the full shape space that a production deployment must cover.

Two structural facts make this space large even though every decoder layer shares a template. First, the topology-distinct irregular FC types per model (Q/K/V/O projections, MLP up/down, attention head slices) number only ∼4–6—not layer-count×per-layer. Second, the optimal descriptor depends on *shape*, not on layer index or weight values, so each (subgraph type, sequence length, model variant) triple is an independent tuning instance. A production cluster serving 5 sequence lengths across a 7–400 B model family yields ∼125 such instances.

At ∼4 generations and ∼30 s per generation, the Policy-L pipeline tunes one instance in ∼2 minutes, versus ∼3 hours for a human NPU engineer (profiling, tiling trials, compile-error debugging). Scaling to ∼125 instances: manual tuning requires ∼375 hours (∼16 working days), while Policy-L completes serially in ∼4.2 hours—roughly a 90× wall-clock reduction. On a multi-node cluster the agent pipeline is embarrassingly parallel across instances, pushing the wall-clock down by another order of magnitude. Compared with *manual* closed-loop tuning rather than with Policy-T, removing the human from the inner cycle converts multi-week effort into an overnight batch. An LLM is not required for that automation, yet Policy-L still matches the Oracle on this failure family at far lower expert cost.

Extrapolation transparency and sensitivity. We stress that the 90× figure is a *projection*, not a measured end-to-end wall-clock: what is *measured* is the per-instance agent budget (∼2 min, i.e. 4 generations of ∼30 s sandbox profiling) and the on-hardware convergence of the loop (Sections 4.2–4.4); what is *assumed* are the per-instance human effort (∼3 h), the instance count (∼125), and serial execution. Table 4 makes these three parameters explicit and shows the compression ratio under conservative and aggressive settings; even the most conservative cell (1 h manual, 50 instances) retains a 30× reduction.

Table 4 Sensitivity of the projected tuning-time compression ratio to the three extrapolation parameters. The headline 90× corresponds to (3 h, 125 instances, serial). Even the most conservative setting retains 30×.

	n=50	n=125	n=250
manual = 1 h/inst	30×	30×	30×
manual = 3 h/inst	90×	90×	90×
manual = 6 h/inst	180×	180×	180×

The ratio is independent of the instance count n because both arms scale linearly in n under the serial model; the binding parameter is the per-instance manual effort. A 10–20% variation in the 2 min agent budget (e.g. 5 generations instead of 4) shifts the ratio by the same small factor and does not change the order of magnitude.

4.9 Discussion

Why dual roles matter. Performance feedback and compiler-error feedback are different modalities. Binding both into one prompt encourages unstable hopping between speculative rewrites and brittle patching—our ablation (Fig. 2, A5) confirms this: a merged agent leaves stall at 10.40%. Role separation mirrors successful automated R&D designs [11] while matching NPU reality: legality is enforced by CANN, optimality by telemetry.

Hardware-agnostic versus hardware-coupled components, and porting cost. The framework splits cleanly into two parts. *Hardware-agnostic* (port unchanged): the iterate structure telemetry→hypothesis→descriptor, the dual-role R/D separation and its two feedback channels (performance vs. error), the CoT causal-reasoning mechanism, the trace/knowledge-base formalism $\tau_g = (H_g, a_g, \phi_g, d_g)$, the Score protocol, and the sandbox ingestion loop. *Hardware-coupled* (re-instantiate per target): the hardware constants (burst boundary, alignment quantum, preferred tile-size set $\{4096, 8192, 16384\}$), the descriptor-legality checker (CANN-specific warning strings and tiling grammar), the action-space hardware priors (which T and r are alignment-friendly), and the profiler backend. Porting to NVIDIA GPUs (CUDA/Triton kernel tuning) requires replacing the constants with warp/block quantization and shared-memory tile constraints, and swapping the legality checker for **nvcc**/Triton diagnostics—roughly a scenario-class rewrite. AMD MI (ROCm/HIP) follows the same pattern; the expected effort is dominated by the profiler and legality integration, which we estimate at the scenario-class level (days, not weeks), not the core-loop level. Physical cross-hardware porting remains future work. Whether the current action space

Just Accepted

suffices for non-FC structures (irregular tiling of attention/KV-cache, sparse MoE-routing memory access, Conv channel alignment) is an open question: the FC template covers shape-driven residuals, whereas attention couples shape with data-dependent masks. We expect the framework structure (roles, traces, Score) to remain unchanged while the action space gains new primitives, mirroring how **align**(T) was added for the FC case.

From case study to system claim. Table 2–Table 3 and Fig. 2 provide a three-policy comparison, component ablation, baseline wall, and cross-scale projection. The critical claim is a *two-level* convergence: (i) on the per-subgraph level, Policy-L matches both the frozen decision tree and the geometric Oracle on Score and descriptor legality; (ii) on the system level, the agent pipeline compresses multi-week manual tuning into a single-digit-hour batch (Section 4.8). The agent's value lies not in beating an Oracle that is by construction an upper bound, but in reaching Oracle-quality descriptors through causal reasoning from telemetry alone, *and* in doing so automatically across an entire model deployment.

A companion open-source implementation of the dual-role loop (Hypothesis / Experiment / Feedback / Trace dataclasses; Ascend profilers; multiple LLM backends) is released alongside the paper to make every ablation directly reproducible [11]. The real-weight deployments sharpen the system claim: the loop's value is not a single tuned descriptor but a *decision policy* over a mixed-loss action space—it converts telemetry into the four-way verdict (deploy / precision-refuse / profit-refuse / kernel-refuse) that a production tuning pipeline needs at scale, and the agreement of its fusion decisions with established production practice suggests the policy generalizes beyond the failure family it was designed for.

Approximation semantics. Low-rank residual rewrite changes the algebraic pathway of the ragged channels. Section 3.5 bounds the worst-case truncation error analytically (Eckart–Young optimality, $0.37 \times \varepsilon_W$ end-to-end dilution); on real pretrained weights the measured ε replaces the bound as the acceptance criterion (Section 4.2), with the hard gate of Section 3.3 rejecting any rewrite above the task's accuracy budget. In deployment, we treat it as a controlled graph transformation whose acceptance is gated by Score and task-level checks; accuracy auditing should accompany production rollout.

Scope and future work. We structure the boundaries of this work in three tiers. *(i) Validated in this paper:* FC-type irregular maps on the residual-stall failure family, on Ascend 910B (small-diameter on-hardware validation of the loop; real-weight multi-objective evaluation with deployment decisions across six projection families, Section 4.2; large-dimension runs under the hardware-faithful offline sandbox). *(ii) Requires new experimental validation:* full multi-layer, full-model wall-clock measurements on production hardware; physical deployment of the low-rank rewrite on a served model with task-level accuracy auditing; extension to attention/KV-cache irregular tiling and sparse MoE-routing memory access, where the failure geometry departs from the aligned-plus-tail template and new action primitives are expected. *(iii) Open framework-level questions:* whether the CoT causal attribution scales to multi-fault interleaved telemetry (several failure modes active simultaneously); how the knowledge base $\mathcal{K}$ composes across hardware generations; and whether role separation remains optimal as the descriptor grammar grows. The tuning-automation numbers assume serial execution; on a multi-node cluster the wall-clock can be reduced by another order of magnitude.

5 Conclusions

We presented GenTile, a dual-role agent framework that closes the loop between LLM causal reasoning and NPU physical telemetry for autonomous tuning of irregular computation graphs. A Research agent (GLM-5.2, Policy-L) diagnoses bottlenecks from structured profiles; a Development agent materializes hardware-legal descriptors from compiler feedback; traces accumulate across iterations under an air-gap sandbox.

As a proof of concept, we validated GenTile on the residual-tail stall—a pervasive failure mode where burst-aligned but tile-incompatible LLM widths idle the NPU pipeline. Policy-L matches the geometric Oracle on Score and descriptor legality in 4 generations, cutting stall from 77.00% to 1.20%. Extrapolating the measured per-instance budget across the full shape space compresses ∼2 weeks of manual tuning into a single-digit-hour batch ($\sim 90\times$ projected reduction). On real pretrained weights from two Qwen generations, the same loop measured the full multi-objective trade-off—latency, energy, memory, AICore efficiency, and ε—across six projection families and a tiered lossless/weak-loss/lossy action space. It deployed the telemetry-guided QKV and gate–up fusions (1.06–$1.16\times$ faster, up to 13% less energy, ε=0), consistent with the fused layouts of production inference engines, validated the Eckart–Young cliff prediction on structural layers, and rejected every rewrite that failed accuracy, profitability, or kernel availability—four separable failure modes, each resolved from telemetry alone. This evidence-backed decision-making over mixed-loss rewrites is the capability we consider most transferable to production NPU tuning. The scope validated in this paper covers FC-type irregular maps, the residual-stall failure mode, and real-weight multi-objective evaluation with agent deployment decisions (Section 4.2); attention/KV-cache, MoE routing, and full-model wall-clock measurements remain future work (Section 4.9). Future work will extend the action space to attention/KV and MoE routing graphs, validate full-model wall-clock on production hardware, and evaluate additional failure modes.

Dates

Received: 29 July 2026; Revised: 18 September 2026;
Accepted: 28 September 2026

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Just Accepted

[2] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

[3] Aixin Liu, Bei Feng, Bing Xue, et al. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[4] Guojie Li. Big data and computing models. *Big Data Research*, 10(1):9–16, 2024.

[5] Tianqi Chen, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Yan, Haichen Shen, Meghan Cowan, Leyuan Wang, Yuwei Hu, Luis Ceze, et al. TVM: An automated end-to-end optimizing compiler for deep learning. In *USENIX OSDI*, pages 578–594, 2018.

[6] Lianmin Zheng, Chengfan Jia, Minmin Sun, Zhao Wu, Cody Hao Yu, Ameer Haj-Ali, Yida Wang, Jun Yang, Danyang Zhuo, Koushik Sen, et al. Ansor: Generating high-performance tensor programs for deep learning. In *USENIX OSDI*, pages 863–879, 2020.

[7] Zhihao Jia, Oded Padon, James Thomas, Todd Warszawski, Matei Zaharia, and Alex Aiken. TASO: Optimizing deep learning computation with automatic generation of graph substitutions. In *SOSP*, pages 47–62, 2019.

[8] Josh Achiam, Steven Adler, Sandhini Agarwal, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[9] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V. Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837, 2022.

[10] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *ICLR*, 2023.

[11] Xu Yang, Xiao Yang, Shikai Fang, et al. R&D-Agent: An LLM-agent framework towards autonomous data science. *arXiv preprint arXiv:2505.14738*, 2025.

[12] Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, et al. MLE-bench: Evaluating machine learning agents on machine learning engineering. In *ICLR*, 2025.

[13] Zhengyao Jiang, Dominik Schmidt, Dhruv Srikanth, et al. AIDE: AI-driven exploration in the space of code. *arXiv preprint arXiv:2502.13138*, 2025.

[14] Xiaocong Jia, Bruce Gu, Jinjun Chen, Longxiang Gao, Weiguang Pang, Guangtong Lv, Youyang Qu, and Lei Cui. Dynamic batch processing with FlexiDecode scheduler for efficient LLM inference in IIoT. *Big Data Mining and Analytics*, 8(6):1307–1323, 2025.

[15] Tianze Hu, Jiacheng Wang, Wenqi Pu, Jiajun Li, Ruixin Gu, Xin Bi, Haijun Yin, and Yu-Ping Wang. A multi-objective optimization framework for data cleaning using large language models. *Big Data Mining and Analytics*, 9(3):672–686, 2026.

[16] Weipeng Cao, Jiongjiong Gu, Zhong Ming, Zhiyuan Cai, Yuzhao Wang, Changping Ji, Zhijiao Xiao, Yuhong Feng, Ye Liu, and Liang-Jie Zhang. Flexible computing: A new framework for improving resource allocation and scheduling in elastic computing. *IEEE Transactions on Services Computing*, 18(1):198–211, 2025.

[17] Weipeng Cao, Xi Tao, Yinghui Pan, Ye Liu, and Zhong Ming. Qos perception for cloud databases: Necessity, trends, and challenges. In *2024 IEEE/ACM 32nd International Symposium on Quality of Service (IWQoS)*, pages 1–2, 2024.

[18] Huawei Technologies Co., Ltd. *CANN Architecture and Operator Development Guide*, 2024.

[19] Huawei Technologies Co., Ltd. *Ascend AI Processor Architecture and Programming*, 2021.

[20] Siyuan Feng et al. TensorIR: An abstraction for automatic tensorized program optimization. In *ASPLOS*, pages 804–817, 2023.

[21] Haojie Wang et al. PET: Optimizing tensor programs with partially equivalent transformations and automated corrections. In *USENIX OSDI*, pages 37–54, 2021.

[22] Ruihang Lai et al. Relax: Composable abstractions for end-to-end dynamic machine learning. In *ASPLOS*, pages 998–1013, 2025.

[23] Yi Zhai, Sijia Yang, Keyu Pan, Renwei Zhang, Shuo Liu, Chao Liu, Zichun Ye, Jianmin Ji, Jie Zhao, Yu Zhang, and Yanyong Zhang. Enabling tensor language model to assist in generating high-performance tensor programs for deep learning. In *USENIX OSDI*, pages 289–305, 2024.

[24] Massinissa Merouani, Islem Kara Bernou, and Riyadh Baghdadi. Agentic auto-scheduling: An experimental study of LLM-guided loop optimization. In *PACT*, pages 186–200, 2025.

[25] Haolin Pan, Hongyu Lin, Haoran Luo, et al. Compiler-R1: Towards agentic compiler auto-tuning with reinforcement learning. In *NeurIPS*, 2025.

[26] Pengfei Xu, Xiaofan Zhang, Cong Hao, Yang Zhao, Yongan Zhang, Yue Wang, Chaojian Li, Zetong Guan, Deming Chen, and Yingyan Lin. AutoDNNchip: An automated DNN chip predictor and builder for both FPGAs and ASICs. In *ACM FPGA*, pages 40–50, 2020.

[27] Jiarong Xing, Leyuan Wang, Shang Zhang, Jack Chen, Ang Chen, and Yibo Zhu. Bolt: Bridging the gap between auto-tuners and hardware-native performance. In *Proceedings of Machine Learning and Systems*, volume 4, 2022.

[28] Qirui Zhou, Yuanbo Wen, Ruizhi Chen, et al. QiMeng-GEMM: Automatically generating high-performance matrix multiplication code by exploiting large language models. *Proc. AAAI Conf. Artif. Intell.*, 39(21):22982–22990, 2025.

[29] Anne Ouyang, Simon Guo, Simran Arora, Alex L. Zhang, William Hu, Christopher Ré, and Azalia Mirhoseini. KernelBench: Can LLMs write efficient GPU kernels? In *ICML*, pages 47356–47415, 2025.

[30] Anjiang Wei, Allen Nie, Thiago S. F. X. Teixeira, et al. Improving parallel program performance with LLM optimizers via agent-system interfaces. In *ICML*, pages 66155–66177, 2025.

[31] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

[32] Pengwei Liu, Minxian Xu, Jiongjiong Gu, Chuanfei Xu, Weipeng Cao, and Zhong Ming. SG-LR QoS: Sparse-gating and layered low-rank based lightweight QoS monitoring for multi-tenant clouds. In *2026 IEEE International Conference on Web Services (ICWS)*, pages 1–7, 2026.

[33] Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. SVD-LLM: Truncation-aware singular value decomposition for large language model compression. In *ICLR*, 2025.

Just Accepted