PR #34517 完整报告

sgl-project/sglang

[AMD][Spec] Accelerate Qwen3.5 verification with grouped-head shared KV

合并时间：2026-08-15 15:15

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34517

执行摘要

- 一句话：Qwen3.5 验证改用分组共享 KV 内核，高并发吞吐最高 +11%
- 推荐动作：值得精读。三个层面有借鉴价值：1）把『单 TP-local KV head』抽象为 MLA 与 GQA 的统一契机的设计思路，一个内核以 PE_DIM == 0 特例吸收两种布局；2）调度门控集中为纯函数 _should_use_verify_shared_kv 并配 mock 单测，配合『失败即回退』的保守策略，正确性永不因新内核受损；3）对 Triton 编译细节（dot 操作数 16 行下限、next_power_of_2(0) 为 0）的处理方式。建议后续关注 Kimi-K3 短验证宽度路径的性能回归，并推动 kernel benchmark 脚本入库以固化性能基线。

功能与动机

PR body 明确指出性能瓶颈来源：Qwen3.5 使用 GQA，TP2 下每个 rank 有 16 个 query head 共享 1 个 KV head；EAGLE target verification 时，原有 split-KV 路径逐 query head 独立处理并重复扫描同一段prefiKV，在高并发下验证阶段受显存带宽限制，开销随并发放大。此前 PR #33981 已为 Kimi-K3 的 absorbed-MLA 布局引入分组头验证内核，本 PR 目标是把同一思路推广到 Qwen3.5 的普通 GQA 布局，使每个程序只加载一次 KV tile 并在一块 query head 上复用，从而降低带宽压力。

实现拆解

实现分 4 步推进，涉及 3 个源码文件与 1 个新增测试文件：

1. 内核泛化（python/sglang/kernels/ops/attention/verify_mla.py）：把 MLA 专属入口 verify_mla_fwd 重命名为布局中性的 verify_shared_kv_fwd，并用 PE_DIM > 0 守卫包裹 q_pe/k_pe 的加载与点积，使 PE_DIM == 0（Q/K 已整体旋转的普通 GQA）时完全跳过 RoPE 分段；_BLOCK_CONFIG 新增 256: (4, 64, 8) 配置以覆盖 Qwen3.5 的 head_dim=256；_VerifySharedKVContext 中 l_pad 取 max(next_power_of_2(cdiv(16, block_h)), next_power_of_2(l_ext))，保证 Triton tl.dot 操作数至少 16 行；BLOCK_DPE 改为 max(1, next_power_of_2(pe_dim))，规避 Triton 3.6 中 next_power_of_2(0) 返回 0 导致 tl.arange 零上界的问题；入口处新增 k_extend.shape[1] != 1、q_head_dim < v_head_dim、空 kv_indices 的拒绝分支。
2. 调度门控（python/sglang/srt/layers/attention/triton_backend.py）：新增纯函数 _should_use_verify_shared_kv(model_config, topk, use_mla, use_verify_splitkv) 收敛全部条件：先要求 is_gfx95_supported() 且 topk == 1；MLA 路径仅放行 Kimi-K3；GQA 路径要求已开启 SGLANG_ENABLE_SPLITKV_VERIFY、is_qwen3_5() 且

get_num_kv_heads(tp, dcp) == 1。__init__ 中 self.use_verify_mla/self.verify_mla_fwd 重命名为 use_verify_shared_kv/verify_shared_kv_fwd；forward_extend 的 target-verify 路由顺序调整为 grouped-head → split-KV → extend_attention_fwd 兜底，任一路径不可用即自然回退。

3. 架构识别（python/sglang/srt/configs/model_config.py）：新增 is_qwen3_5() 助手，枚举 4 种架构名（条件生成与因果 LM 各两个变体），与既有 is_kimi_k3() 并列，构成同一组架构判定的数据契约。
4. 测试配套（test/registered/attention/test_verify_shared_kv.py，新增 235 行）：以 extend_attention_fwd 为参照做数值 parity，覆盖 TP8/TP4/TP2 本地 query head 形状（4/8/16）、1/2/3 token 短验证宽度、BF16 与 FP8 E4M3（含 K/V descale）KV cache、Kimi-K3 absorbed-MLA 回归形状、多 TP-local KV head 拒绝，以及 mock 化的后端门控 test_backend_dispatch_gate；测试注册到 AMD CI stage-b 套件。提交历史中还有一次与本次无关的多模态 transport 测试提交被显式 revert，体现对 PR 范围的收紧。

关键文件：

- python/sglang/kernels/ops/attention/verify_mla.py（模块 验证内核；类别 source；类型 core-logic；符号 verify_shared_kv_fwd, verify_mla_fwd, can_handle, _verify_mla_prefix_stage1）：核心内核文件：verify_mla_fwd 重构为布局中性的 verify_shared_kv_fwd，新增 PE_DIM=0 普通 GQA 支持、短宽度 padding、BLOCK_DPE 保底与入口守卫，是本次性能收益的来源，同时影响已上线的 Kimi-K3 路径。
- python/sglang/srt/layers/attention/triton_backend.py（模块 注意力后端；类别 source；类型 core-logic；符号 _should_use_verify_shared_kv, use_verify_shared_kv, verify_shared_kv_fwd）：调度中枢：新增 _should_use_verify_shared_kv 纯函数统一门控条件，重命名 use_verify_mla/verify_mla_fwd 为布局中性命名，并调整 target-verify 的内核路由顺序。
- python/sglang/srt/configs/model_config.py（模块 模型配置；类别 source；类型 data-contract；符号 is_qwen3_5）：新增 is_qwen3_5() 架构识别助手，枚举 4 种 Qwen3.5 架构名，构成调度门控的数据契约。
- test/registered/attention/test_verify_shared_kv.py（模块 验证测试；类别 test；类型 test-coverage；符号 TestVerifySharedKV, _build_inputs, _run_parity, test_backend_dispatch_gate）：新增 235 行正确性测试：以 extend_attention_fwd 为参照做数值 parity，覆盖 TP 形状、短验证宽度、FP8 KV cache、K3 回归与调度门控 mock 测试，并注册 AMD CI stage-b 套件。

关键符号：_should_use_verify_shared_kv, verify_shared_kv_fwd, verify_mla_fwd, _verify_mla_prefix_stage1, can_handle, is_qwen3_5, _run_parity

关键源码片段

python/sglang/kernels/ops/attention/verify_mla.py

核心内核文件：verify_mla_fwd 重构为布局中性的 verify_shared_kv_fwd，新增 PE_DIM=0 普通 GQA 支持、短宽度 padding、BLOCK_DPE 保底与入口守卫，是本次性能收益的来源，同时影响已上线的 Kimi-K3 路径。

```
# sglang/kernels/ops/attention/verify_mla.py —— 分组头共享 KV 验证内核
# prefill 阶段主循环：每个 program 处理 BLOCK_H 个共享同一 KV head 的 query head，
# 一次加载 KV tile 后整块复用，避免 split-KV 路径对每个 query head 重复扫描 prefix KV。
# NOPE_DIM / PE_DIM 语义随布局而变：
# - Kimi-K3 absorbed MLA：NOPE_DIM = latent 宽度，PE_DIM = 附加 RoPE 宽度（64）；
# - Qwen3.5 普通 GQA：Q/K 已整体旋转，PE_DIM == 0，NOPE_DIM 即完整 head_dim（256）。

        # 加载 q_nope；GQA 下这就是完整的 Q。仅当存在 RoPE 分段时才加载 q_pe。
        q_row = tl.reshape(
            (cur_q_start + offs_l)[None, :] * stride_qbs
            + offs_h[:, None] * stride_qh,
            (R,),
        )
        q_nope = tl.load(
            Q + q_row[:, None] + offs_dn[None, :],
            mask=row_mask[:, None] & (offs_dn[None, :] < NOPE_DIM),
            other=0.0,
        ).to(K_Buffer.dtype.element_ty)
        if PE_DIM > 0:
            q_pe = tl.load(
                Q + q_row[:, None] + (NOPE_DIM + offs_dp)[None, :],
                mask=row_mask[:, None] & (offs_dp[None, :] < PE_DIM),
                other=0.0,
            ).to(K_Buffer.dtype.element_ty)

        # 循环内：GQA 时 qk 只有 q_nope 一项；MLA 时再叠加 q_pe 与 k_pe 的点积。
        qk = tl.dot(q_nope, k_nope)
        if PE_DIM > 0:
            k_pe = tl.load(
                K_Buffer + base + (NOPE_DIM + offs_dp)[:, None],
                mask=(offs_dp[:, None] < PE_DIM) & n_mask[None, :],
                other=0.0,
            )
            qk += tl.dot(q_pe, k_pe)
        qk *= sm_scale * k_scale
        qk = tl.where(n_mask[None, :], qk, float("-inf"))

# 构造上下文时的两个 Triton 兼容性处理：
# 1) tl.dot 要求 (BLOCK_H * L_EXT) 至少 16 行，验证宽度过短时把 l_pad 补齐到下限；
# 2) Qwen3.5 的 PE_DIM == 0，而 tl.arange(0, BLOCK_DPE) 需要正数上界，
# Triton 3.6 中 next_power_of_2(0) 返回 0，故用 max(1, ...) 保底，
# 配合 PE_DIM > 0 守卫确保不会产生实际 PE 加载或计算。
        min_l_pad = triton.next_power_of_2(triton.cdiv(16, block_h))
        self.l_pad = max(min_l_pad, triton.next_power_of_2(l_ext))
        # ...
        BLOCK_DPE=max(1, triton.next_power_of_2(self.pe_dim)),
```

python/sglang/srt/layers/attention/triton_backend.py

调度中枢：新增 _should_use_verify_shared_kv 纯函数统一门控条件，重命名 use_verify_mla/verify_mla_fwd 为布局中性命名，并调整 target-verify 的内核路由顺序。

```
# sglang/srt/layers/attention/triton_backend.py
# 分组头共享 KV 验证的调度门控：决定 target-verify 阶段是否使用 verify_shared_kv_fwd
# 而非逐 query head 扫描 prefix KV 的 split-KV 路径。

def _should_use_verify_shared_kv(model_config, topk, use_mla, use_verify_splitkv):
    # 该内核是 AMD gfx95 专属优化；非 HIP 构建或 gfx942 等平台返回 False，
    # 从而保留原有 split-KV / extend_attention_fwd 行为，无需额外 is_hip() 判断。
    # topk > 1 时验证树非因果链，内核只保证 topk == 1 的纯因果等价。
    if not is_gfx95_supported() or topk != 1:
        return False
    if use_mla:
        # Kimi-K3 的 absorbed MLA 布局：唯一 TP-local KV head 即 MLA latent。
        return is_kimi_k3(model_config.hf_config)
    # Qwen3.5 普通 GQA：要求每 rank 恰好 1 个 TP-local KV head，
    # 且 split-KV 快速路径开关已打开，否则回退到逐头路径。
    return (
        use_verify_splitkv
        and is_qwen3_5(model_config.hf_config)
        and model_config.get_num_kv_heads(
            get_parallel().attn_tp_size, get_parallel().attn_dcp_size
        )
        == 1
    )

# __init__ 中把分散的硬件 / 模型判断收敛到统一门控，便于测试直接 mock 验证。
        self.use_verify_shared_kv = _should_use_verify_shared_kv(
            model_runner.model_config,
            self.topk,
            self.use_mla,
            self.use_verify_splitkv,
        )

# forward_extend 中 target-verify 的路由顺序：grouped-head 优先，
# 其次 per-head split-KV，最后 extend_attention_fwd 兜底；
# 各内核 can_handle 失败时返回 False，调用方自然回退，正确性不受威胁。
        if self.use_verify_shared_kv:
            verify_fwd = self.verify_shared_kv_fwd
        elif self.use_verify_splitkv:
            verify_fwd = self.verify_splitkv_fwd
        else:
            verify_fwd = None
```

python/sglang/srt/configs/model_config.py

新增 is_qwen3_5() 架构识别助手，枚举 4 种 Qwen3.5 架构名，构成调度门控的数据契约。

```
# sglang/srt/configs/model_config.py
```

```
# 架构识别契约：与 is_kimi_k3 并列，供验证内核调度门控复用。

def is_kimi_k3(config) -> bool:
    return _hf_arch(config) == "KimiK3ForConditionalGeneration"


def is_qwen3_5(config) -> bool:
    # 覆盖因果 LM 与条件生成（多模态）两类 Qwen3.5 架构；
    # 新增变体若未枚举，只会错过加速而不会走错路径（fail-safe）。
    return _hf_arch(config) in (
        "Qwen3_5ForConditionalGeneration",
        "Qwen3_5MoeForConditionalGeneration",
        "Qwen3_5ForCausalLM",
        "Qwen3_5MoeForCausalLM",
    )
```

评论区精华

Review 阶段共有 3 个有效讨论线程：

- 命名泛化建议（design）：审核者 1am9trash 给出 LGTM，并指出内核已不再 MLA-only，建议把 self.use_verify_mla / self.verify_mla_fwd 重命名为 use_verify_shared_kv / verify_shared_kv_fwd 以表达真实意图；chuyeh 随后修复，1am9trash 确认。
- 是否需要is_hip守卫（design）：yichiche质疑PE_DIM分支改动是否只对MI355X有益、是否需 if_hip 保护；1am9trash 指出该 verify 内核仅 MI355X 使用且已被 is_gfx95_supported() 门控；chuyeh 补充说明 dispatch 通过 _should_use_verify_shared_kv() 限定 gfx950，非 HIP 构建与 gfx942 上 is_gfx95_supported() 返回 False，无需额外 is_hip() 判断。
- BLOCK_DPE 取 max 的原因（question）：sogalin 询问 BLOCK_DPE=max(1, next_power_of_2(pe_dim)) 是否为防溢出；chuyeh 解释 Qwen3.5 的 PE_DIM=0，Triton 3.6 中 next_power_of_2(0) 返回 0，而 tl.arange(0, BLOCK_DPE) 需要正数上界，max(1, ...) 只提供最小合法编译期 extent，配合 PE_DIM > 0 守卫确保不会产生实际 PE 加载或计算。
- 命名泛化：verify_mla 相关符号重命名为 verify_shared_kv (design): 接受建议，全部重命名为布局中性命名，并同步更新 _BLOCK_CONFIG 注释与路由注释。
- PE_DIM > 0 分支是否需要 is_hip() 守卫 (design): 维持现有门控，不引入 is_hip()。
- BLOCK_DPE = max(1, next_power_of_2(pe_dim)) 的用途 (question): 属于 Triton 编译约束适配而非防溢出，方案获认可。

风险与影响

- 风险：主要风险集中在以下几点：
- Kimi-K3 现有路径被共享代码影响：verify_mla.py 的 stage1 与 padding 逻辑同时服务于已上线的 K3 路径；l_pad 下限调整为至少 4（BLOCK_H=4 时 16/4=4），仅当验证宽度 l_ext < 4 时 K3 的 pad 尺寸与网格形状才会变化，数值因 mask 等价，但短宽度下的性能

特征需回归确认，且新增测试未覆盖 K3 短宽度。

- Triton 版本行为耦合：正确性依赖 Triton 3.6 的 next_power_of_2(0) == 0 语义与 tl.dot 行数下限；升级 Triton 后 padding 行为可能改变，需要回归验证。
- 平台判定依赖：未显式加 is_hip() 守卫，完全依赖 is_gfx95_supported() 在非 HIP 构建返回 False；若未来 gfx95 出现在 CUDA 侧或 ROCm 版本改变该判断，存在误路由的可能（概率低）。
- 架构枚举的 fail-safe 方向：is_qwen3_5() 枚举 4 个架构名，未来新增 Qwen3.5 变体不会自动获得加速但也不会出错；反之若某模型被误判为 Qwen3.5，在 gfx95 上会被优先尝试共享 KV 内核，但因 can_handle 拒绝而回退，正确性仍安全。
- CI 噪音：PR Test (Extra) 套件曾有失败并多次 /rerun-failed-ci，最后以绿色合并；PR body 也提示需排查 AMD runner 不一致问题。
- 性能可复现性：kernel benchmark 脚本刻意未入库，端到端数据来自临时 gate 与固定 seed 的单次运行，后续回归难以直接复现这些数字。
- 影响：影响范围被刻意收窄：
- 用户 / 模型影响：仅 AMD MI355X（gfx950）+ Triton attention + Qwen3.5（4 种架构）+ EAGLE topk==1 且 TP 本地恰 1 个 KV head 的场景自动启用；高并发（C64/C128）端到端吞吐 +9.6%~+11%，低并发基本持平（C4 +2.2%），prefix 越长内核收益越大（16K prefix 时 1.49x）。CUDA、gfx942、其他模型、topk>1 全部走原路径，无用户配置变更。
- 系统影响：target-verify 阶段的显存带宽占用显著下降，缓解高并发下的带宽瓶颈；验证内核路由从两层（split-KV / extend）变为三层（shared-KV / split-KV / extend）。
- 团队影响：注意力内核的命名契约从 MLA 语义升级为布局中性的 shared-KV 语义，后续新增 GQA/MHA 模型只需扩展 is_qwen3_5 类判断与 _BLOCK_CONFIG 即可复用，为 AMD 侧验证内核的持续泛化留下清晰扩展点；同时 _should_use_verify_shared_kv 纯函数化使门控逻辑可单测，降低后续维护误伤风险。
- 风险标记：核心验证内核路径变更，强硬件门控（gfx95/ROCm），依赖 Triton 版本行为，Kimi-K3 现有路径共享代码，CI 附加套件曾有失败

关联脉络

- PR #33981 （标题未在本次材料中提供）Kimi-K3 absorbed-MLA 分组头验证内核：本 PR 直接扩展该 PR 引入的 verify_mla_fwd 内核至普通 GQA 布局，PR body 明确引用其为前作；两组改动共享同一内核与测试范式。
- PR #34238 [AMD] Broadcast the EAGLE greedy verify decision across TP ranks on ROCm: 同为 AMD ROCm 上 EAGLE verify 路径的修复与优化，属于 AMD 推测解码验证链路的持续建设。
- PR #32593 [Kernel] Enable Helion backend for Kimi Delta-Attention: 同为 gfx95 上注意力内核性能优化（jit-kernel/attention/performance），且同样通过 server_args 与后端分发控制内核选择，与本 PR 的硬件门控思路一脉相承。