

PR #55235 完整报告

vllm-project/vllm

[ROCm][Perf] Tune MiniMax-M3 decode top-k for short contexts

合并时间: 2026-09-14 10:36

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/55235>

执行摘要

- 一句话: gfx950 短上下文 top-k 改用单 CTA, selector 提速约 28%
- 推荐动作: 值得精读, 尤其是 `_decode_topk_launch_policy` 与两个对应测试用例: 它们展示了 ROCm 内核性能调优的严谨姿势 (把编译期参数纳入 launch policy、128/129 显式边界、CPU 全序 reference 验证 NaN/Inf/ 有符号零、图回放验证 counter 重置)。对非 ROCm 开发者参考价值有限, 但「只有输出精确一致才允许跳过模型评估」的判断标准和「launch geometry 与算术解耦」的设计思路值得借鉴。合并本身无阻塞, 维护者已批准。

功能与动机

gfx950 上 MiniMax-M3 decode 阶段的 BF16 top-k selector 在短上下文 (生产 8k1k graph 容量仅 74 个稀疏块) 下仍使用 B512/C16 的大网格, 过度启动大量无用 CTA。PR body 给出的测量显示: query length 4、top-k 16 时, fused selector + sparse table 从 4.138-4.440 us 降至 2.947-3.213 us, 边界扫描 (容量 65/74/128, query length 1/4/8, index heads 1/2/4) 确认 selector 时间下降 32.78-35.32%。PR 同时声明这是已合并 #54682 的窄范围 follow-up, 并经过 duplicate-work 检查, 与 #53448、#52664、#52059 不冲突。

实现拆解

1. 变更入口与 policy 扩展: 修改 `vllm/models/minimax_m3/amd/ops/index_topk.py` 中的 `_decode_topk_launch_policy`, 新增常量 `DECODE_TOPK_SHORT_BLOCKS_PER_CHUNK = 128`, 并把返回值从 3 元组扩展为 6 元组 (`num_topk_chunks`, `topk_block_size`, `selector_num_warps`, `selector_num_stages`, `single_tile_guaranteed`, `adaptive_final_merge`), 使 Triton kernel 的编译期特化参数 (`BLOCK_SIZE_K`、`num_warps`、`num_stages`) 由 launch policy 统一输出, 消除调用点的隐式推导。
2. 核心分支逻辑: 在 `is_gfx950 and topk == 16 and 0 < max_block <= 16 * 512` 的快速路径内再分两档: `max_block <= 128` 时返回 `(1, 128, 2, 1, True, True)` (单 CTA、128 块、2 warps、1 stage); 否则保持 16-chunk 快速路径 `(16, 512, 4, 2, True, True)`。范围外 (非 gfx950、topk 非 16 或超长上下文) 维持原有 target-grid 推导, 统一返回 `(1 << (target.bit_length() - 1), 512, 8, 2, False, False)`, 保证 128/129 边界有显式分支锁定。
3. 调用点改造: `minimax_m3_index_decode` 解包新的 6 元组, 将 `topk_block_size`、`selector_num_warps`、`selector_num_stages` 分别传给 `_decode_topk_fused_kernel` 的 `BLOCK_SIZE_K`、`num_warps`、`num_stages`, 替代原先由 `adaptive_final_merge` 推断 `num_warps` 的逻辑。

4. 测试配套: tests/kernels/attention/test_minimax_m3.py 中

test_amd_decode_topk_launch_policy 的期望值扩展为 6 元组, 并新增 (74, ...), (128, ...), (129, ...) 三组用例分别覆盖生产容量、边界内最后一档和越界切换;

test_amd_decode_fused_topk_total_order_and_replay 增加 block_size_k、num_warps、num_stages 参数化并新增 (1, True, True, 74, 128, 2, 1, 16, 1) 用例, block_counts 按 BLOCK_SIZE_K 动态生成 15/16/17/63/64/65/127/128/129 等边界长度, 分数注入点改为 block_size_k // 2 与 block_size_k, 并用 repeat 对齐短序列长度列表。

5. 验证与部署: 测试全量 121 passed、13 skipped; pre-commit 全部通过; 由于输出精确一致, 未重新跑模型评估。CI 共触发 4 轮 Buildkite (#87142、#87276、#87371、#88455), 最终由维护者批准合并。

关键文件:

- vllm/models/minimax_m3/amd/ops/index_topk.py (模块 选择器内核; 类别 source; 类型 core-logic; 符号 _decode_topk_launch_policy, minimax_m3_index_decode, DECODE_TOPK_SHORT_BLOCKS_PER_CHUNK) : 核心实现文件: 扩展 _decode_topk_launch_policy 返回值, 新增短上下文单 CTA 分支, 并让调用点 minimax_m3_index_decode 使用 policy 输出的 BLOCK_SIZE_K、num_warps、num_stages 编译特化参数。
- tests/kernels/attention/test_minimax_m3.py (模块 内核测试; 类别 test; 类型 test-coverage; 符号 test_amd_decode_topk_launch_policy, test_amd_decode_fused_topk_total_order_and_replay) : 测试配套: 将 launch policy 断言扩展为 6 元组, 新增 74/128/129 边界用例; fused kernel 测试新增短上下文特化参数组合, 并把边界序列改为按 BLOCK_SIZE_K 动态生成, 覆盖图回放与全序一致性。

关键符号: _decode_topk_launch_policy, minimax_m3_index_decode,

test_amd_decode_topk_launch_policy, test_amd_decode_fused_topk_total_order_and_replay

关键源码片段

vllm/models/minimax_m3/amd/ops/index_topk.py

核心实现文件: 扩展 _decode_topk_launch_policy 返回值, 新增短上下文单 CTA 分支, 并让调用点 minimax_m3_index_decode 使用 policy 输出的 BLOCK_SIZE_K、num_warps、num_stages 编译特化参数。

```
# 短上下文特化阈值: gfx950 生产 8k1k graph 容量仅 74 个稀疏块,
```

```
# 远小于默认 chunk 的 512 块, 继续用大 CTA 会造成明显浪费。
```

```
DECODE_TOPK_SHORT_BLOCKS_PER_CHUNK = 128
```

```
DECODE_TOPK_BLOCKS_PER_CHUNK = 512
```

```
MAX_DECODE_TOPK_FAST_CHUNKS = 16
```

```
def _decode_topk_launch_policy(  
    max_block: int,  
    total_q: int,  
    num_idx_heads: int,
```

```

topk: int,
*,
is_gfx950: bool,
) -> tuple[int, int, int, int, bool, bool]:
    """选择 selector 的 CTA 网格与编译期 launch 配置。

    返回值依次为: num_topk_chunks、block_size_k、num_warps、num_stages、
    single_tile_guaranteed、adaptive_final_merge, 直接决定 Triton kernel
    的编译特化, 因此 128/129 边界必须用显式分支锁定。
    """
    if (
        is_gfx950
        and topk == 16
        and 0 < max_block <= MAX_DECODE_TOPK_FAST_CHUNKS * DECODE_TOPK_BLOCKS_
        PER_CHUNK
    ):
        # 短上下文特化: 单 CTA、128 块、2 warps、1 stage,
        # 覆盖 74 块这种生产容量, 避免启动 16 个 512 块的大 CTA。
        if max_block <= DECODE_TOPK_SHORT_BLOCKS_PER_CHUNK:
            return 1, DECODE_TOPK_SHORT_BLOCKS_PER_CHUNK, 2, 1, True, True
        # 中等容量仍走 16-chunk 快速路径, 但 warps 收敛为 4。
        return (
            MAX_DECODE_TOPK_FAST_CHUNKS,
            DECODE_TOPK_BLOCKS_PER_CHUNK,
            4,
            2,
            True,
            True,
        )
    # 长上下文 / 非 gfx950: 按目标网格反推 chunk 数, 保持原有语义。
    target = max(
        1,
        1 << (DECODE_TOPK_TARGET_GRID // max(1, total_q * num_idx_heads)).bit_length() - 1,
    )
    return (
        1 << (target.bit_length() - 1),
        DECODE_TOPK_BLOCKS_PER_CHUNK,
        8,
        2,
        False,
        False,
    )

# 调用点不再自行推导编译参数, 统一由 launch policy 决定。
(
    num_topk_chunks,
    topk_block_size, # 短上下文分支为 128, 其余为 512
    selector_num_warps, # 短上下文分支为 2
    selector_num_stages, # 短上下文分支为 1

```

```

    single_tile_guaranteed,
    adaptive_final_merge,
) = _decode_topk_launch_policy(
    max_block=max_block,
    total_q=total_q,
    num_idx_heads=num_idx_heads,
    topk=topk,
    is_gfx950=is_gfx950,
)

_decode_topk_fused_kernel[(batch, num_idx_heads, num_topk_chunks)](
    ...,
    BLOCK_SIZE_K=topk_block_size,
    BLOCK_SIZE_T=block_size_t,
    EMIT_SPARSE_TABLE=emit_sparse_table,
    SINGLE_TILE_GUARANTEED=single_tile_guaranteed,
    ADAPTIVE_FINAL_MERGE=adaptive_final_merge,
    num_warps=selector_num_warps,
    num_stages=selector_num_stages,
)

```

tests/kernels/attention/test_minimax_m3.py

测试配套：将 launch policy 断言扩展为 6 元组，新增 74/128/129 边界用例；fused kernel 测试新增短上下文特化参数组合，并把边界序列改为按 BLOCK_SIZE_K 动态生成，覆盖图回放与全序一致性。

```

@pytest.mark.parametrize(
    ("max_block", "total_q", "num_idx_heads", "topk", "is_gfx950", "expected"),
    [
        # 空容量回退到通用网格。
        (0, 24, 1, 16, True, (2, 512, 8, 2, False, False)),
        # 短上下文特化：单 CTA、128 块、2 warps、1 stage。
        (1, 1, 1, 16, True, (1, 128, 2, 1, True, True)),
        # 生产 8k1k graph 的实际容量 74 块，必须命中特化分支。
        (74, 4, 1, 16, True, (1, 128, 2, 1, True, True)),
        # 边界内最后一档 128 块。
        (128, 4, 1, 16, True, (1, 128, 2, 1, True, True)),
        # 129 块越过边界，立即切回 16-chunk 快路径。
        (129, 4, 1, 16, True, (16, 512, 4, 2, True, True)),
        # 非 gfx950 或 topk != 16 保持原有回退策略。
        (513, 16, 4, 16, False, (1, 512, 8, 2, False, False)),
    ],
)

def test_amd_decode_topk_launch_policy(
    max_block: int,
    total_q: int,
    num_idx_heads: int,
    topk: int,
    is_gfx950: bool,

```

```

expected: tuple[int, int, int, int, bool, bool],
):
    """短上下文特化有严格的 dispatch 边界。"""
    from vllm.models.minimax_m3.amd.ops.index_topk import (
        _decode_topk_launch_policy,
    )

    assert (
        _decode_topk_launch_policy(
            max_block,
            total_q,
            num_idx_heads,
            topk,
            is_gfx950=is_gfx950,
        )
        == expected
    )

```

评论区精华

该 PR 的技术讨论集中在 PR body 而非 review 评论中：审阅侧只有 zhou9402 的 "LGTM" 和维护者 AndreasKaratzas 的 APPROVED（空 body）。自动审查方面，coderabbitai[bot] 的全量 review 明确表示 "No actionable comments were generated"，claude[bot] 因 PR 来自 fork 而禁用自动审查。PR body 中最重要的设计论述是：只改 launch geometry、保持 BF16 precision 契约、不切换到 AITER FP8 路径，并用独立 CPU 全序参考（含 NaN、±Inf、±0.0、并列分数）证明输出精确一致，这也是跳过模型评估的依据。

- 自动审查：fork 限制与 CodeRabbit 无意见 (other): 无自动审查阻塞，依赖维护者手动审查。
- 人类审阅签收 (other): 维护者批准合并，无实质技术争议。
- 精度契约与 FP8 路径边界 (design): 作为设计约束被测试固化，无反对意见。

风险与影响

- 风险：风险面较窄：变更只命中 gfx950 + topk==16 + max_block<=128 的 decode 分支，非 ROCm 平台或无此条件的路径完全不受影响。但该分支直接决定 decode 主路径的 kernel 编译特化，若 128/129 边界判断错误会启动错误 CTA，因此测试专门锁定了 74、128、129 三个关键点。另一个潜在风险是编译特化参数现已散落在 policy 返回值中，未来新增 specialization 时必须同步更新调用点和测试参数化，否则容易遗漏。性能上，129-512 块区间仍走 B512/C16 (warps 4、stages 2) 的快速路径，未提供中间容量过渡配置，可能仍有优化空间。CI 上测试标记为 skipif not is_rocm，回归验证依赖 AMD CI（本 PR 跑了 4 轮 Buildkite）。数值风险很低，因为仅改变 launch geometry 且测试覆盖全序排序、有符号零、counter 重置与图回放。
- 影响：对用户：gfx950 ROCm 上 MiniMax-M3 短上下文 (≤128 稀疏块，约 8k context) decode 的 selector 阶段提速约 33-35%，selector + sparse table 总耗时下降约 26-29%，属于直接可见的 decode 延迟改善；其他架构、容量和 top-k 取值行为完全不变。对系统：无 API、配置、调度器或精度契约变化，输出与之前逐位一致，不需要模型重新评估。对

团队：确立了一个可复用的内核调优模式——launch policy 直接返回编译期特化参数（BLOCK_SIZE_K、num_warps、num_stages），并用边界用例和 CPU 全序参考把 dispatch 行为锁死；代价是测试变得更复杂，后续维护需要同步更新。

- 风险标记：ROCm 专用路径，128 块 dispatch 边界，launch-policy 编译特化参数化，测试仅 ROCm 生效

关联脉络

- PR #56170 [ROCm][Performance] Avoid blocking MiniMax M3 scalar upload: 同仓库同模型（MiniMax M3）的 ROCm 性能优化，说明该模型在 ROCm 平台上的性能调优正在持续进行。
- PR #54682 MiniMax-M3 decode top-k selector（前身 PR，标题按 PR body 描述）：PR body 明确声明本 PR 是已合并 #54682 的窄范围 launch-policy follow-up。
- PR #53448 MiniMax-M3 prefill + optional FP8 index cache（PR body 提及）：同一 fused decode selector 功能线的并行改动，专注 prefill，不做本 PR 的 decode launch-policy 调整。
- PR #52664 外部 AITER indexer 路径集成（PR body 提及）：与 FP8/AITER 索引路径相关，本 PR 保持 BF16 路径不切换，二者互不影响。
- PR #52059 MiniMax-M3 prefill scoring（PR body 提及）：同一 selector 的 prefill 方向改动，与本 PR 的 decode 方向互补。