PR #35499 完整报告

sgl-project/sglang

[AMD] Improve K3 dspark draft attn kernel perf

合并时间：2026-08-21 12:49

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35499

执行摘要

- 一句话：K3 DSpark draft 注意力改走 verify_shared_kv，ROCm 长上下文提速
- 推荐动作：值得精读，适合关注 AMD ROCm 平台、投机解码与 Triton kernel 优化的工程师。核心学习点有两个：一是用编译期常量（HAS_KV_HEADS、IS_CAUSAL）把新场景融入通用内核，同时保证既有 Kimi-K3 MLA 路径编译结果零变化；二是用调优表按 head_dim 选择 kernel 配置。建议后续补一个针对 DSpark draft（h_kv > 1、非因果）的 kernel 级单元测试，并对更长上下文做回归验证。

功能与动机

PR body 中的追踪数据显示：Kimi-K3 DSpark 每步运行 5 个 draft attention 层，共耗时 2.368 ms，其中 5 个 attention kernel 占 1.244 ms（超过 50%），原因是 draft 注意力回退到通用 extend_attention_fwd，而该通用内核不适合共享 KV 形态。且 --attention-backend 只选择目标验证内核，aiter 不在 draft worker 的受支持后端列表中，ROCm 上 draft 始终回退到 triton，因此该热点影响所有 Kimi-K3 DSpark 部署。

实现拆解

1. 新增架构识别 helper：在 python/sglang/srt/configs/model_config.py 中新增 is_dspark_draft(config)，通过 HF architecture 名称 DSparkDraftModel 识别 K3 的 draft 模型，作为后续路由 gate 的数据契约基础。
2. 修改路由 gate：在 python/sglang/srt/layers/attention/triton_backend.py 的 _should_use_verify_shared_kv 中，于 MLA 分支与 qwen3_5 分支之间插入 DSpark draft 分支——只要 use_verify_splitkv 开启即走 verify_shared_kv，避免落到低效的 extend_attention_fwd。该位置说明 draft attention 是 qwen3 风格 GQA 而非 MLA，且使用双向（非因果）模式。
3. 扩展共享 KV 内核：在 python/sglang/kernels/ops/attention/verify_mla.py 中，stage-1（_verify_mla_prefix_stage1）与 stage-2（_verify_mla_combine_stage2）新增 KV head stride 参数与 KV_GROUP_NUM、HAS_KV_HEADS 编译期常量：多 KV head 时按 head // kv_group_num * stride_h 计算偏移，单 KV head（MLA/MQA）时偏移恒为 0，保留原有内层访存算术；stage-2 新增 IS_CAUSAL 编译期常量以支持 draft attention 的非因果模式（stage-1 前缀循环本来就不加因果 mask，无需改动）。同时调优表新增 head_dim=64 -> (BLOCK_H=4, BLOCK_N=256, num_warps=4) 条目，适配 K3 GQA draft attention。

4. 配套验证：本 PR 未新增单元测试文件；通过 GSM8k 端到端准确率（0.955）与 8k/16k/32k 输入下的性能表验证，最后一次提交（"Correct the api"）修正了 kernel 调用点的 API 签名。

关键文件：

- python/sglang/kernels/ops/attention/verify_mla.py（模块 注意力内核；类别 source；类型 core-logic；符号 _get_vmla, _verify_mla_prefix_stage1, _verify_mla_combine_stage2）：核心 kernel 变更：扩展 stage-1/stage-2 支持多 KV head 与非因果模式，新增 head_dim=64 调优条目，是性能提升的直接来源。
- python/sglang/srt/layers/attention/triton_backend.py（模块 注意力后端；类别 source；类型 dependency-wiring；符号 _should_use_verify_shared_kv）：路由 gate _should_use_verify_shared_kv 增加 DSpark draft 分支，让 draft attention 从通用 extend kernel 切换到 verify_shared_kv，是本次性能优化的控制点。
- python/sglang/srt/configs/model_config.py（模块 模型配置；类别 source；类型 data-contract；符号 is_dspark_draft）：新增 is_dspark_draft() helper，作为路由 gate 的架构识别依据，是数据契约层面的配套变更。

关键符号：_should_use_verify_shared_kv, is_dspark_draft, _verify_mla_prefix_stage1, _verify_mla_combine_stage2, _get_vmla

关键源码片段

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

核心 kernel 变更：扩展 stage-1/stage-2 支持多 KV head 与非因果模式，新增 head_dim=64 调优条目，是性能提升的直接来源。

```
# file: python/sglang/kernels/ops/attention/verify_mla.py
# 共享 KV 注意力内核 stage-1（extend / prefill 阶段）的关键变更：
# 让 score 阶段支持多 KV head 的 GQA 形态，并用 HAS_KV_HEADS 编译期开关
# 保证 MLA 单 latent head 路径的地址算术与改动前完全一致。

@triton.jit
def _verify_mla_prefix_stage1(
    # … 省略既有指针与 stride 参数 …
    stride_buf_kbs, # K 在 KV 序列维度上的 stride
    stride_buf_vbs, # V 在 KV 序列维度上的 stride
    stride_buf_kh, # 新增：K 在 KV head 维度上的 stride
    stride_buf_vh, # 新增：V 在 KV head 维度上的 stride
    # … 省略运行时张量（kv_indices、K_Buffer、V_Buffer 等）…
    KV_GROUP_NUM: tl.constexpr, # query heads / kv heads 的组数
    HAS_KV_HEADS: tl.constexpr, # 0 = MLA 单 latent head；1 = 普通 GQA 多 KV head
    # … 省略 head_dim 等常量 …
):
    # 每个程序块处理 BLOCK_H 个 query head；调用方保证 KV_GROUP_NUM % BLOCK_H == 0，
    # 因此块内所有 query head 落在同一个 KV head 上，K/V 仍按二维 tile 加载。
    head_start = head_block * BLOCK_H
    offs_h = head_start + tl.arange(0, BLOCK_H)
```

```
    # 多 KV head 时，由 query head 编号反推 KV head，并换算成缓存偏移；
    # MLA 路径偏移恒为 0，完全不增加内层循环的地址计算开销。
    if HAS_KV_HEADS:
        kv_head_off_k = (head_start // KV_GROUP_NUM) * stride_buf_kh
        kv_head_off_v = (head_start // KV_GROUP_NUM) * stride_buf_vh
    else:
        kv_head_off_k = 0
        kv_head_off_v = 0

    offs_l = tl.arange(0, L_EXT)
    offs_dn = tl.arange(0, BLOCK_DNOPE)
    offs_dp = tl.arange(0, BLOCK_DPE)

    # 每个 query 位置对应的 KV 序列号，来自 kv_indices 表
    kv_loc = tl.load(
        kv_indices + cur_batch_kv_start_idx + offs_n,
        mask=n_mask,
        other=0,
    )
    # 缓存基址 = 序列号 * 序列 stride + KV head 偏移
    base = kv_loc[None, :] * stride_buf_kbs + kv_head_off_k
    k_nope = tl.load(
        K_Buffer + base + offs_dn[:, None],
        mask=(offs_dn[:, None] < NOPE_DIM) & n_mask[None, :],
        other=0.0,
    )
    # V 同理加上 KV head 偏移；MLA 与普通共享 KV 都用同一个 V_Buffer 入口
    v = tl.load(
        V_Buffer + kv_loc[:, None] * stride_buf_vbs + kv_head_off_v + offs_dv[None, :],
        mask=n_mask[:, None] & (offs_dv[None, :] < V_HEAD_DIM),
        other=0.0,
    )
    # ... 后续为 QK^T / softmax 与概率写回，逻辑不变 ...

# file: python/sglang/kernels/ops/attention/verify_mla.py
# stage-2（combine / 归并阶段）：每个程序只负责一个 query head。
# 新增 IS_CAUSAL 支持非因果 draft attention，新增 KV head 偏移支持 GQA 多 head。

@triton.jit
def _verify_mla_combine_stage2(
    # ... 省略既有参数 ...
    stride_keh, # 新增：K 在 KV head 维度上的 stride（combine 阶段）
    stride_veh, # 新增：V 在 KV head 维度上的 stride（combine 阶段）
    KV_GROUP_NUM: tl.constexpr,
    HAS_KV_HEADS: tl.constexpr, # 1 = 普通 GQA；0 = MLA 单 latent head
    IS_CAUSAL: tl.constexpr, # 0 = 非因果（DSpark draft）；1 = 因果
):
    cur_batch = tl.program_id(0)
    cur_head = tl.program_id(1)
```

```
    # 多 KV head 时按组反推 KV head 偏移；MLA 单头路径保持 0 偏移
    if HAS_KV_HEADS:
        kv_head_off_ke = (cur_head // KV_GROUP_NUM) * stride_keh
        kv_head_off_ve = (cur_head // KV_GROUP_NUM) * stride_veh
    else:
        kv_head_off_ke = 0
        kv_head_off_ve = 0

    # ... 加载 partial softmax 结果并做跨 split-kv 归并 ...
    # 原实现按 causal mask 生成遮罩；当 IS_CAUSAL 为 0（DSpark draft 双向注意力）时，
    # 取消该遮罩，使每个 token 都能看到完整 KV 序列。
```

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

路由 gate _should_use_verify_shared_kv 增加 DSpark draft 分支，让 draft attention 从通用 extend kernel 切换到 verify_shared_kv，是本次性能优化的控制点。

```
# file: python/sglang/srt/layers/attention/triton_backend.py
# 作用：决定共享 KV 注意力内核（verify_shared_kv）是否可用。
# 本 PR 新增 DSpark draft 分支，让 K3 的 draft attention 不再回退通用 extend kernel。

def _should_use_verify_shared_kv(model_config, topk, use_mla, use_verify_splitkv):
    # 仅在 AMD gfx95 平台且 topk == 1（单 token 验证）时启用
    if not is_gfx95_supported() or topk != 1:
        return False

    # MLA 路径：Kimi-K3 目标验证走原有的 MLA 专用分支
    if use_mla:
        return is_kimi_k3(model_config.hf_config)

    # 新增分支：K3 DSpark draft 模型是 qwen3 风格的 GQA 注意力，使用双向（非因果）模式；
    # 只要 split-kv 开启就进入 verify_shared_kv，避免落到性能差的 extend_attention_fwd
    if is_dspark_draft(model_config.hf_config):
        return use_verify_splitkv

    # 其余模型维持原有条件：Qwen3.5 系列且单 KV head 才走共享 KV 内核
    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
    )
```

评论区精华

该 PR 没有产生正式的 review 评论线程。HaiShaw 直接发布 /tag-and-rerun-ci 触发 CI 重跑，并以 APPROVED 状态批准合并。技术验证主要依赖 PR body 自带的 GSM8k 精度与 TTT/ITL 性能表；未标注基线准确率是报告中的一个小信息缺口，但未在评论区被质疑。

- CI 重跑与最终审批 (other): CI 通过并完成合并；没有展开公共技术讨论。

风险与影响

- 风险：
  1. 核心 kernel 行为扩展：verify_mla.py 是 Kimi-K3 目标验证的既有内核，本次同时改了 stage-1/stage-2 的地址计算；若 HAS_KV_HEADS 分支的 KV head 偏移算错，会影响所有多 KV head 的 GQA 注意力正确性。
  2. 路由 gate 影响默认行为：_should_use_verify_shared_kv 对所有 DSparkDraftModel 在 AMD gfx95 平台上生效，若未来出现非 qwen3 风格或头数配置不同的 DSpark draft 模型，可能误入新内核路径。
  3. 缺少针对性单元测试：本次没有新增测试文件，只有 GSM8k 端到端验证；对 h_kv > 1、非因果等边界场景缺少 kernel 级回归覆盖。
  4. 调优条目全局生效：head_dim=64 的新调优配置会被其他 head_dim 为 64 的模型复用，通常只影响性能不影响正确性，但仍需关注。
  5. 收益与输入长度强相关：短上下文（8k 以下）收益较小，不应假设所有负载都能获得同等加速。 - 影响：影响范围集中在 AMD ROCm（gfx95）+ Kimi-K3 DSpark 投机解码场景：draft attention 是每步都执行的公共路径，本次优化直接降低端到端延迟，长上下文（32k）下 ITL 提升最高 12%、TTT 提升最高 6%，短上下文也有 1-5% 的收益。系统层面不改变模型输出语义、不新增 API 或配置项，只影响 --attention-backend triton 与 aiter 后端的 draft 内核选择。对团队而言，该改动为 AMD 平台投机解码性能优化提供了范例，但也意味着后续内核维护需要同时兼顾 MLA 单头与 GQA 多头两种地址模式。 - 风险标记：核心 kernel 行为扩展，无新增单元测试，AMD gfx95 平台限定，路由 gate 影响默认行为

关联脉络

- PR #35197 fix(kernel) Fix Helion small-token prefill bug: 同属 python/sglang/kernels/ops/attention 目录的 AMD 平台 attention kernel 修改，kernel 边界场景（短 prefill）回归思路可互相参考。
- PR #32832 [AMD] [sgl-kernel] Bypass caches for peer traffic in ROCm custom all-reduce: 同属 ROCm/AMD 平台性能优化系列，体现 SGLang 在 AMD 平台持续投入 kernel 级性能调优。