PR #26588 完整报告

sgl-project/sglang

Optimize Gemma4 H200 MoE and extend attention

合并时间：2026-06-06 14:14

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26588

执行摘要

- 一句话：优化 Gemma4 H200 MoE 与 extend attention 性能
- 推荐动作：推荐精读。尤其注意 kernel dedup 设计方法和 BF16 精度分析。对于 Gemma4 部署有直接收益；对编写数值稳定的 Triton kernel 有参考价值。

功能与动机

Gemma4 模型拥有 E=128, N=704 的独特 MoE 结构，现有 MoE Triton 配置未针对该 shape 优化，且 Hopper extend attention block size 默认值在 Lq 129-256 区间并非最优。MTP 推测解码时 QKV RMSNorm 的计算成为小批量瓶颈。PR 旨在通过 Triton 调优和 kernel 优化缩小这些差距。

实现拆解

1. MoE Triton 配置调优：在 triton_3_6_0 配置目录下新增两个 JSON 文件，分别为 normal 和 down 投影提供 H200 专用的 block size / warp / stage 调优参数（E=128, N=704）。这些配置覆盖从 1 到 1024 的各类 M 值，通过调整 BLOCK_SIZE_M/N/K、GROUP_SIZE_M、num_warps 和 num_stages 使 Triton 编译器输出更优的 GPU kernel。
2. Extend-attention block size 分段调整：修改 extend_attention.py 中的 _get_block_sizes_for_extend_attention 函数，在 Hopper 架构（CUDA capability >= 9）下将 Lq ≤ 256 这一档拆分为 ≤128 和 128–256 两档，分别使用 (128,64) 和 (64,64) 的 (BLOCK_M, BLOCK_N) 组合，以匹配 Gemma4 典型请求长度，减少 prefill 阶段耗时。
3. 小批量 QKV RMSNorm by-head kernel：在 gemma4_fused_ops.py 中新增辅助函数 _gemma_qkv_rmsnorm_store，封装单 head 的 load-rms-scale-store 逻辑；然后改造 _gemma_qkv_rmsnorm_kernel，通过编译时常量 BY_HEAD 支持两种 launch 模式：当 BY_HEAD=True 时 grid 为 (M, total_heads)，每个 program 处理一个 token 的一个 head，适合小批量；BY_HEAD=False 时 grid 为 (M,)，串行循环 heads，适合大批量。外层 Python 函数 gemma_qkv_rmsnorm 根据 M 或传入参数自动选择 launch 路径。该 kernel 原位归一化 Q、K、V，使用 strided 视图避免 contiguous 拷贝。
4. （回滚）MoE routing 融合：原始提交包含两个融合优化：a) 将 Gemma4Router.forward 中的 rmsnorm + scale 合并为单步 fused kernel；b) 新增 _gemma4_topk_softmax_scale_kernel 将 topk、softmax、per-expert 缩放合并为一个 pass。由于 BF16 精度回归导致 MTP GSM8K 得分从 0.445 降至 0.360，两个 commit 被先后回滚。回滚后精度恢复，其余优化保留。

5. 配套验证：无新增测试文件，但通过 CI 中的 test_gemma4_mtp_26b_a4b_extra 测试和自定义 benchmark 验证了性能提升和准确性等价。

关键文件：

- python/sglang/srt/layers/gemma4_fused_ops.py（模块 融合算子；类别 source；类型 core-logic；符号 _gemma_qkv_rmsnorm_store, _gemma_qkv_rmsnorm_kernel, gemma_qkv_rmsnorm）：核心逻辑修改，新增辅助函数 _gemma_qkv_rmsnorm_store，改造 _gemma_qkv_rmsnorm_kernel 支持 BY_HEAD 双模式，实现 QKV RMSNorm 小批量加速与 kernel 去重。
- python/sglang/srt/layers/moe/moe_runner/triton_utils/configs/triton_3_6_0/E=128,N=704,device_name=NVIDIA_H200.json（模块 MoE 配置；类别 config；类型 configuration）：新增 H200 专用的 MoE normal projection Triton 配置，针对 E=128,N=704 的 shape 进行手动调优，是性能提升的关键之一。
- python/sglang/srt/layers/moe/moe_runner/triton_utils/configs/triton_3_6_0/E=128,N=704,device_name=NVIDIA_H200_down.json（模块 MoE 配置；类别 config；类型 configuration）：新增 H200 专用的 MoE down projection Triton 配置，与 normal 配置并行，确保 down 投影同样获得最佳性能。
- python/sglang/srt/layers/attention/triton_ops/extend_attention.py（模块 扩展注意力；类别 infra；类型 infrastructure；符号 _get_block_sizes_for_extend_attention）：调整 Hopper extend attention block size 阈值，将 Lq 256 上限拆分为 ≤128 和 128-256 两档，改善 Gemma4 prefill 性能。

关键符号：_gemma_qkv_rmsnorm_store, _gemma_qkv_rmsnorm_kernel, gemma_qkv_rmsnorm, _get_block_sizes_for_extend_attention

关键源码片段

python/sglang/srt/layers/gemma4_fused_ops.py

核心逻辑修改，新增辅助函数 _gemma_qkv_rmsnorm_store，改造 _gemma_qkv_rmsnorm_kernel 支持 BY_HEAD 双模式，实现 QKV RMSNorm 小批量加速与 kernel 去重。

```
# 辅助函数：单个 head 的 RMSNorm 归一化（带可选权重）
@triton.jit
def _gemma_qkv_rmsnorm_store(
    X_ptr, W_ptr, stride_m, m, h, cols, mask,
    HEAD_DIM: tl.constexpr, eps, HAS_WEIGHT: tl.constexpr,
):
    # 计算每个 head 的偏移量
    off = m * stride_m + h * HEAD_DIM + cols
    x = tl.load(X_ptr + off, mask=mask, other=0.0).to(tl.float32)
    # RMSNorm: x / sqrt(mean(x^2) + eps)
    rrms = tl.rsqrt(tl.sum(x * x, axis=0) / HEAD_DIM + eps)
    out = x * rrms
    # 若有权重，则乘以权重（权重长度为 head_dim，广播使用）
    if HAS_WEIGHT:
```

```
        w = tl.load(W_ptr + cols, mask=mask, other=0.0).to(tl.float32)
        out = out * w
    tl.store(X_ptr + off, out.to(X_ptr.dtype.element_ty), mask=mask)


@triton.jit
def _gemma_qkv_rmsnorm_kernel(
    Q_ptr, K_ptr, V_ptr, Q_w_ptr, K_w_ptr,
    stride_q_m, stride_k_m, stride_v_m,
    NUM_Q_HEADS: tl.constexpr, NUM_KV_HEADS: tl.constexpr,
    HEAD_DIM: tl.constexpr, eps, HAS_KV: tl.constexpr,
    BY_HEAD: tl.constexpr, BLOCK: tl.constexpr,
):
    # 通过 BY_HEAD 实现两种 launch 形状:
    # - True: grid (M, total_heads)，每个 program 处理一个 token 的一个 head
    # - False: grid (M,)，串行循环所有 head
    m = tl.program_id(0)
    cols = tl.arange(0, BLOCK)
    mask = cols < HEAD_DIM

    if BY_HEAD:
        # 每个 program 处理单个 head
        h_all = tl.program_id(1)
        if h_all < NUM_Q_HEADS:
            _gemma_qkv_rmsnorm_store(
                Q_ptr, Q_w_ptr, stride_q_m, m, h_all, cols, mask, HEAD_DIM, eps, True
            )
        elif HAS_KV and h_all < NUM_Q_HEADS + NUM_KV_HEADS:
            h = h_all - NUM_Q_HEADS
            _gemma_qkv_rmsnorm_store(
                K_ptr, K_w_ptr, stride_k_m, m, h, cols, mask, HEAD_DIM, eps, True
            )
        elif HAS_KV:
            h = h_all - NUM_Q_HEADS - NUM_KV_HEADS
            # V head 不使用权重（weight=1）
            _gemma_qkv_rmsnorm_store(
                V_ptr, Q_w_ptr, stride_v_m, m, h, cols, mask, HEAD_DIM, eps, False
            )
    else:
        # 串行循环：适用于大批量，避免过多 program
        for h in tl.static_range(NUM_Q_HEADS):
            _gemma_qkv_rmsnorm_store(
                Q_ptr, Q_w_ptr, stride_q_m, m, h, cols, mask, HEAD_DIM, eps, True
            )
        if HAS_KV:
            for h in tl.static_range(NUM_KV_HEADS):
                _gemma_qkv_rmsnorm_store(
                    K_ptr, K_w_ptr, stride_k_m, m, h, cols, mask, HEAD_DIM, eps, True
                )
```

```
    for h in tl.static_range(NUM_KV_HEADS):
        _gemma_qkv_rmsnorm_store(
            V_ptr, Q_w_ptr, stride_v_m, m, h, cols, mask, HEAD_DIM, eps, False
        )
```

评论区精华

内核去重建议：审查者 yuan-luo 指出 _gemma_qkv_rmsnorm_by_head_kernel（2D grid）与既有 _gemma_qkv_rmsnorm_kernel（1D grid）实质相同，维护两份会同步困难。BBuf 接受建议并在后续 commit 中通过 BY_HEAD constexpr 合并为一个 kernel，验证了精度和性能等价。

Extend attention block size 自动调优：yuan-luo 提出硬编码阈值（128/256）应改为 autotune key。BBuf 同意并承诺作为后续改进，暂不阻塞当前性能提升。

H100/H20 配置缺失：yuan-luo 建议增加 H100 和 H20 的 MoE 配置。BBuf 表示在没有相应硬件验证的情况下不应复制 H200 配置，留作后续。

精度回归根本原因：PR body 详细分析了两个融合 kernel 导致精度下降的原因——BF16 累加顺序和 softmax reduction tree 差异使路由 logits 出现系统性偏移，累积后扩大了 MTP draft/target 分布差异。这一分析是重要的工程教训。

- 合并 QKV RMSNorm kernel (BY_HEAD) (design): BBuf 接受建议，在后续 commit 中通过 BY_HEAD constexpr 合并为一个 kernel，验证了精度和性能等价。
- Extend attention block size 应 autotune (performance): 作为后续任务，当前 PR 暂保留硬编码。
- 增加 H100/H20 MoE 配置 (other): 暂不添加，留作后续。
- MoE 路由融合精度回归分析 (correctness): 两个融合 kernel 被回滚，精度恢复。未来需要 bit-exact 实现才能启用。
- 关联 PR 合并努力 (other): 未在本次 PR 中合并，仅作为后续参考。

风险与影响

- 风险：
  1. 精度回归风险：路由融合优化在 BF16 下因浮点顺序不等价导致精度下降（已回滚）。若未来有人恢复或修改 fused kernel 而未严格 bit-wise 校验，可能再次引入。
  2. 核心路径变更：extend attention block size 的修改会影响所有 Hopper 架构上的 prefill 性能。尽管对 Gemma4 正向，但未在非 Gemma4 模型上验证，可能存在细微回归。
  3. H200 特有配置：新增 MoE 配置仅针对 H200 调优，在其他 Hopper 卡（如 H100，H20）上可能不是最优，但不会导致错误。
  4. 缺少直接单元测试：QKV RMSNorm 和 MoE 配置变更没有对应的 Python 单元测试，仅依赖 CI 中的端到端 MTP 测试覆盖。
  5. 配置可维护性：自动调优（Triton Autotune）尚未集成，手动维护 JSON 配置可能跟不上模型变化。 - 影响：用户：Gemma4 用户将直接受益于更低的 TTFT（-23.3%）和

TPOT（-18.2%），以及约 18% 的端到端延迟改善。其他模型用户受影响极小（extendattention block 调整可能略有影响，但整体中性）。系统：修改了 attention 和 MoE 两个关键计算路径，但改动范围小，无新增依赖。团队：维护者需要理解两个 kernel 路径的共存；融合路由的教训值得团队其他优化项目参考。

- 风险标记：精度回归风险，核心路径变更，缺少测试覆盖，H200 特有配置

关联脉络

- PR #26502 fused router optimization for Gemma4: issue 评论中 pyc96 建议与当前 PR 的 fused router 优化合并，但最终未合并。
- PR #25461 fused norm optimization for Gemma4: issue 评论中 pyc96 建议与当前 PR 的 norm 优化合并。