PR #42153 完整报告

vllm-project/vllm

[Perf] Use 2D-grid to eliminate divmod in W8W8 group quant

合并时间：2026-05-12 22:01

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/42153

执行摘要

本 PR 针对 csrc/libtorch_stable/quantization/w8a8/fp8/per_token_group_quant.cu 中的分组量化内核进行性能优化：将 1D 网格启动的 divmod 索引计算替换为 2D 网格 + 模板常量索引，使编译器将坐标计算降级为位运算。微基准测试显示大 shape 下吞吐量提升 5-10%，且不影响模型精度。

功能与动机

当前内核使用 global_group_id % padded_groups_per_row 和 global_group_id / padded_groups_per_row 计算分组索引，运行时除法带来额外开销。利用 2D 网格可将这些计算替换为乘法 / 加法，编译器进一步优化为位操作。PR body 中的性能表显示多数 shape 获得约 1.03x-1.10x 加速。

实现拆解

1. 新增辅助函数：在同一文件中添加 GetGroupsPerBlockX，根据 padded_groups_per_row 选择 X 向图块大小（16/8/4），确保 kx * ry == 16（Y 向图块自动为 16 / kx）。
2. 模板参数编译时常量化：在内核 per_token_group_quant_8bit_packed_register_kernel 的模板中增加 kGroupsPerBlockX、kRowsPerBlock，移除运行时参数 groups_per_block。
3. 索引计算改为 2D：内核体内将一维 local_group_id 拆解为 sf_k_local = local_group_id % kGroupsPerBlockX 和 row_local = local_group_id / kGroupsPerBlockX，全局索引改为 blockIdx.x * kGroupsPerBlockX + sf_k_local 和 blockIdx.y * kRowsPerBlock + row_local，消除对 padded_groups_per_row 的除法和取模。
4. 启动逻辑适配：per_token_group_quant_8bit_packed 根据 GetGroupsPerBlockX 返回值分支实例化模板，设置 2D grid 尺寸 (padded_groups_per_row / kGroupsPerBlockX, tma_aligned_mn / kRowsPerBlock)，线程数固定为 16 * THREADS_PER_GROUP = 128。

csrc/libtorch_stable/quantization/w8a8/fp8/per_token_group_quant.cu

所有变更都在这个文件中：新增辅助函数 GetGroupsPerBlockX，修改内核模板增加编译时常量，替换索引计算逻辑，调整启动配置。

```
// 根据 padded_groups_per_row 选择一个 <= 16 的最大因子，
// 使得 kx * ry 恒为 16（每组可容纳 2 图块，每个图块 8 线程 × 2 组）。
// 因为 padded_groups_per_row 总是 4 的倍数，结果只会在 16/8/4 中选择。
inline int GetGroupsPerBlockX(int64_t padded_groups_per_row) {
```

```
  if (padded_groups_per_row % 16 == 0) {
    return 16; // 每行可分配 16 组 X 向图块
  }
  if (padded_groups_per_row % 8 == 0) {
    return 8; // 8 组 X 向图块，Y 向自动取 2（16 / 8 == 2）
  }
  return 4; // 4 组 X 向图块，Y 向自动取 4（16 / 4 == 4）
}

template <typename T, typename DST_DTYPE, int GROUP_SIZE,
          int kGroupsPerBlockX, int kRowsPerBlock>
__global__ void per_token_group_quant_8bit_packed_register_kernel(
    const T* __restrict__ input, void* __restrict__ output_q,
    unsigned int* __restrict__ output_s_packed,
    const int padded_groups_per_row, const int groups_per_row,
    const int mn, const int output_q_mn_extent,
    const int tma_aligned_mn, const int64_t num_scale_elems,
    const float eps, const float min_8bit, const float max_8bit) {
  constexpr int THREADS_PER_GROUP = 8;
  constexpr int VEC_SIZE = 32 / sizeof(T); // 对于 bf16/fp16 = 16

  const int local_group_id = threadIdx.x / THREADS_PER_GROUP;
  const int lane_id = threadIdx.x % THREADS_PER_GROUP;

  // 将线程块内的一维 local_group_id 拆解为 X 和 Y 两个维度
  const int sf_k_local = local_group_id % kGroupsPerBlockX;
  const int row_local = local_group_id / kGroupsPerBlockX;
  // 全局索引 = 块起始 + 块内偏移，无需运行时除法和取模
  const int sf_k_idx = blockIdx.x * kGroupsPerBlockX + sf_k_local;
  const int mn_idx = blockIdx.y * kRowsPerBlock + row_local;

  if (mn_idx >= tma_aligned_mn) {
    return;
  }
  // ... 后续量化计算不变
}
```

评论区精华

- gemini-code-assist评论指出 LAUNCH_REG_KERNEL 宏假设 kx 只能是 16/8/4，建议增加静态断言确保 kx * ry == 16。作者未回复，但 PR 已被合并，说明当前场景下假设成立。该讨论体现了对内核配置鲁棒性的关注。

风险与影响

- 风险：GetGroupsPerBlockX 的后备逻辑假设 padded_groups_per_row 为 4 的倍数，若未来不满足则该配置会静默使用 kx=4, ry=4，可能引入性能回退（但不会崩溃）。此外，LAUNCH_REG_KERNEL 宏缺少对所有可能 kx 值的显式分支检查。

- 影响：仅影响 per_token_group_quant_8bit_packed 内核路径，用户无 API 变更。推理吞吐量在特定 shape 下提升 1-10%，大 batch 场景收益更明显。

关联脉络

本 PR 是 vLLM W8A8 量化内核的持续性能优化。近期相关量化内核改进包括 #41664（MXFP4 线性层支持）和 #41382（FlashInfer NVLink 修复），这些 PR 共同提升了 vLLM 在 FP8 量化推理路径上的效率。本 PR 专注于消除除法指令，是更细粒度的性能调优。