

PR #49750 完整报告

vllm-project/vllm

[Perf] RMSNorm uncontiguous support, 1.2~3.1x kernel performance improvement

合并时间: 2026-07-30 04:45

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/49750>

执行摘要

- 一句话: RMSNorm 内核支持非连续张量, 性能提升 1.2-3.1 倍
- 推荐动作: 值得精读。改动虽小但收益显著, 且 review 中关于 aliasing 的讨论清晰展示了内核设计中“所有权 / semantics”的权衡。可作为性能优化 PR 的典范。

功能与动机

这是 Issue #46654 (GLM5.2 性能优化) 的一部分。作者 @tlrmchlsmth 发现大量模型在 fused_add_rms_norm 前做了不必要的 .contiguous(), 这些操作本可通过内核直接支持非连续输入来消除。

实现拆解

步骤 1: 放宽 C++ 内核输入要求 在 `csrc/libtorch_stable/layernorm_kernels.cu` 中, 将 `STD_TORCH_CHECK(residual.is_contiguous())` 改为 `STD_TORCH_CHECK(residual.stride(-1) == 1)`, 允许最后一维连续但前面维度不连续的输入。同时调整内部偏移计算以适配非连续 stride。

步骤 2: 移除模型层冗余 .contiguous() 在 `vllm/model_executor/models/` 下的 6 个模型文件中, 删除了调用 fused_add_rms_norm 或 RMSNorm 前的 .contiguous() 调用, 包括:

- hunyuan_v1.py: HunYuanAttention 和 HunYuanCrossAttention 的 qk-norm
- deepseek_v2.py: DeepseekV2DecoderLayer 中 all-gather 后的 residual 处理
- step3p5.py: Step3P5Attention 的 qk-norm
- apertus.py: ApertusAttention 的 qk-norm
- lfm2.py 和 lfm2_moe.py: q 和 k 的 view 后去 contiguous

步骤 3: 更新单元测试 在 `tests/kernels/core/test_layernorm.py` 中调整测试用例, 确保新条件 `stride(-1)==1` 被正确验证, 并添加对非连续输入的支持 (通过 view 而非 contiguous 来构造)。所有模型代码改动仅涉及删除 .contiguous(), 不改变语义, 但需确认调用处的 tensor 确实满足 stride 要求。配置、文档、部署无变更。

关键文件:

- `csrc/libtorch_stable/layernorm_kernels.cu` (模块 层归一化内核; 类别 other; 类型 core-logic; 符号 fused_add_rms_norm): 核心性能改动: 放宽 fused_add_rms_norm 的输入连续性要求, 移除基于 is_contiguous() 的严苛检查并支持 stride(-1)==1 输入。

- `vllm/model_executor/models/hunyuanyuan_v1.py` (模块 模型代码; 类别 source; 类型 data-contract; 符号 `HunYuanAttention.forward`, `HunYuanCrossAttention.forward`) : 代表性模型改动, 展示 `HunYuanAttention` 和 `HunYuanCrossAttention` 中移除冗余 `.contiguous()` 的典型模式。
- `vllm/model_executor/models/deepseek_v2.py` (模块 模型代码; 类别 source; 类型 data-contract; 符号 `DeepseekV2DecoderLayer.forward`) : 另一个重要模型, 改动中移除了 `all-gather` 后的 `residual.contiguous()`, 该调用紧邻 `fused_add_rms_norm`, 移除后性能提升直接。
- `vllm/model_executor/models/step3p5.py` (模块 模型代码; 类别 source; 类型 data-contract; 符号 `Step3P5Attention.forward`) : `Step3P5` 模型, `Attention` 中的 `qk-norm` 移除了 `.contiguous()`。
- `tests/kernels/core/test_layernorm.py` (模块 单元测试; 类别 test; 类型 test-coverage) : 测试配套, 验证内核新行为和模型改动正确性。

关键符号: `fused_add_rms_norm`, `HunYuanAttention.forward`, `HunYuanCrossAttention.forward`, `DeepseekV2DecoderLayer.forward`, `Step3P5Attention.forward`, `ApertusAttention.forward`, `test_input_perturbations_have_expected_behavior`

关键源码片段

`vllm/model_executor/models/hunyuanyuan_v1.py`

代表性模型改动, 展示 `HunYuanAttention` 和 `HunYuanCrossAttention` 中移除冗余 `.contiguous()` 的典型模式。

```
def forward(self, positions, hidden_states, kv_states=None):
    qkv, _ = self.qkv_proj(hidden_states)
    q, k, v = qkv.split([self.q_size, self.kv_size, self.kv_size], dim=-1)
    q, k = self.rotary_emb(positions, q, k)
    ori_k = k
    if self.use_qk_norm:
        # 不再需要 .contiguous(), 因为 fused_add_rms_norm 内核现在支持
        # stride(-1)==1 的输入。view 产生的 tensor 自动满足最后一维连续,
        # 因此可直接传入。
        q = self.query_layernorm(
            q.view(-1, self.num_heads, self.head_dim),
        )
        k = self.key_layernorm(
            k.view(-1, self.num_kv_heads, self.head_dim),
        )
    attn_output = self.attn(q, k, v)
    attn_output = attn_output.view(q.shape[0], -1)
    output, _ = self.o_proj(attn_output)
    return output, (ori_k, v)
```

评论区精华

mgoin 在 `csrc/libtorch_stable/layernorm_kernels.cu:319` 提出疑问：“既然调用端去掉了 `.contiguous()`，是否应在内核内部做条件拷贝确保安全？如 `if (residual.stride(-1) != 1) residual = residual.contiguous()`”。yewentao256 回应：“条件拷贝对于非修改的 `rms_norm` 是安全的，但 `fused_add_rms_norm` 会原地更新 `residual`。如果让 `residual` 重新绑定到临时拷贝，则调用方的原始 `tensor` 不会被修改，违背了操作符的别名语义。所以不能添加条件拷贝，必须保持对外要求 `stride(-1)==1`。”mgoin 认可后予以批准。

- Conditional contiguous in `fused_add_rms_norm` kernel (design): 决定不添加条件拷贝，保持对外要求 `stride(-1)==1`。

风险与影响

- 风险：回归风险：内核断言从 `is_contiguous()` 收紧到 `stride(-1)==1`，若任何路径传递了 `stride(-1)!=1` 的残差张量，将触发断言失败。但通过审核所有调用点，确认它们移除 `contiguous()` 后仍满足 `stride(-1)==1`（均为 `split` 或 `view` 结果）。风险较低。兼容性：`fused_add_rms_norm` 的 C 接口未变，仅加强文档。向上兼容。性能反向：非极少数情况下无负面性能影响；对于调用方传入的已是 `contiguous` 的残差，性能不变；对于非 `contiguous` 输入，避免了额外拷贝，提升显著。无安全风险。
- 影响：用户：使用 DeepSeek、Hunyuan、Apertus、LFM、MiniCPM 等模型的用户获得免费加速，最大 3.15x。无需配置变更。系统：内核改动可能影响所有使用 `fused_add_rms_norm` 的路径，但经过静态审查和测试覆盖。团队无需额外维护成本。团队：此优化可推广到其他可能保留 `.contiguous()` 的模型 PR 中。作为性能优化的范例，有助于培养“移除冗余、利用内核能力”的意识。
- 风险标记：内核边界条件变化，多处模型调用耦合

关联脉络

- 暂无明显关联 PR