PR #21985 完整报告

sgl-project/sglang

perf: eliminate attention DtoD copy by passing pre-allocated output to FA

合并时间：2026-04-25 03:05

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/21985

执行摘要

- 一句话：消除注意力层 DtoD 拷贝，每层节省约 14μs
- 推荐动作：值得精读。本 PR 展示了如何通过 PyTorch 的 out 参数和 op schema 别名标注消除不必要的张量拷贝，是性能优化的经典案例。团队内的推理引擎开发人员应关注其中的设计权衡（如用 forward_batch 属性而非 kwargs 传递输出），以应用到其他相似场景。

功能与动机

在 unified_attention_with_output 中，注意力后端内部新分配输出张量，然后通过 .copy_() 将其拷贝到预分配的输出缓冲区，导致每个注意力层产生约 14μs 的 Memcpy DtoD。对于 28 层模型，每次前向传递累积约 392μs。vLLM 通过直接传递 out= 参数避免了这一开销。本 PR 旨在消除这一冗余拷贝。

实现拆解

1. radix_attention.py：在调用 attn_backend.forward 前将 output[:real_num_tokens] 赋值给 forward_batch._attn_output；调用后仅当返回张量 data_ptr 与 output 不同时才执行 copy_()，避免了强制拷贝。
2. flashattention_backend.py：在 forward_extend 和 forward_decode 中，从 forward_batch._attn_output 读取并 reshape（.view）为 FA 期望的 shape (-1, tp_q_head_num, v_head_dim)，作为 out 参数传递给 flash_attn_with_kvcache 和 flash_attn_varlen_func。
3. sgl-kernel/python/sgl_kernel/flash_attn.py：flash_attn_with_kvcache 和 flash_attn_varlen_func 新增 out=None 参数，并将其传递给 torch.ops.sgl_kernel.fwd 的 out 位置。
4. jit_kernel 包装层（flash_attention.py 和 flash_attention_v3.py）：透传 out 参数到下层调用。
5. sgl-kernel/csrc/flash_extension.cc：修改 op schema，将 out 类型从 Tensor? 改为 Tensor(a!)?，返回类型从 Tensor 改为 Tensor(a!)，让 PyTorch dispatch 知道返回别名 out，从而避免防御性拷贝。

关键文件：

- python/sglang/srt/layers/radix_attention.py（模块 注意力层；类别 source；类型 core-logic；符号 unified_attention_with_output）：核心入口：在该文件中将预分配的输

出绑定到 forward_batch._attn_output，并实现条件拷贝逻辑，是消除冗余拷贝的关键决策点。

- python/sglang/srt/layers/attention/flashattention_backend.py（模块 注意力层；类别 source；类型 core-logic；符号 forward_extend, forward_decode）：FA 后端实现：在该文件中提取 _attn_output 并作为 out 参数传递给所有 FA 调用路径，覆盖 forward_extend 和 forward_decode。
- sgl-kernel/python/sgl_kernel/flash_attn.py（模块 内核封装；类别 source；类型 core-logic；符号 flash_attn_with_kvcache, flash_attn_varlen_func）：sgl-kernel 封装层：新增 out 参数并透传到底层 C++ 算子，是使 out 生效的必要环节。
- python/sglang/jit_kernel/flash_attention.py（模块 JIT 内核；类别 source；类型 core-logic；符号 flash_attn_with_kvcache, flash_attn_varlen_func）：JIT 内核路由层：转发 out 参数到具体实现（ver=3/4），确保 jit_kernel 用户也能受益。
- python/sglang/jit_kernel/flash_attention_v3.py（模块 JIT 内核；类别 source；类型 core-logic；符号 flash_attn_with_kvcache, flash_attn_varlen_func）：JIT 内核 FA3 实现：接收 out 参数并传递给底层 sgl-kernel 调用。
- sgl-kernel/csrc/flash_extension.cc（模块 C++ 扩展；类别 source；类型 core-logic）：C++ 算子注册：修改 op schema 以标注输出别名，这是避免 PyTorch 防御性拷贝的关键。

关键符号：unified_attention_with_output, forward_extend, forward_decode, flash_attn_with_kvcache, flash_attn_varlen_func

关键源码片段

python/sglang/srt/layers/radix_attention.py

核心入口：在该文件中将预分配的输出绑定到 forward_batch._attn_output，并实现条件拷贝逻辑，是消除冗余拷贝的关键决策点。

```
# python/sglang/srt/layers/radix_attention.py
# unified_attention_with_output 函数中的关键变更部分
# 在调用注意力后端之前，将预分配的输出切片传递给后端
forward_batch._attn_output = output[:real_num_tokens] # 切片匹配 FA 的 query 长度

# ... 调用后端 forward ...

# 后端可能直接写入 output（data_ptr 相同），或者返回新张量
if ret.data_ptr() != output.data_ptr():
    # 仅当后端没有直接写入 output 时才拷贝（保障正确性）
    output[:real_num_tokens].view(ret.shape).copy_(ret)
```

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

FA 后端实现：在该文件中提取 _attn_output 并作为 out 参数传递给所有 FA 调用路径，覆盖 forward_extend 和 forward_decode。

```
# python/sglang/srt/layers/attention/flashattention_backend.py
# forward_extend 开头从 forward_batch 中提取预分配输出
_fa_out = (
```

```
    forward_batch._attn_output.view(-1, layer.tp_q_head_num, layer.v_head_dim)
    if getattr(forward_batch, "_attn_output", None) is not None
    else None
)
# ... 后续原有逻辑 ...
# 在调用 flash_attn_with_kvcache 或 flash_attn_varlen_func 时传递 out
result = flash_attn_with_kvcache(
    q=...,
    k_cache=key_cache,
    v_cache=value_cache,
    # ... 其他参数 ...
    out=_fa_out, # 直接写入预分配缓冲区
)
```

评论区精华

审核人 Qiaolin-Yu 已批准，无其他评论。主要技术决策包括：1）不使用 kwargs 传递 output（会破坏不支持 **kwargs 的后端如 FlashInfer），改为在 forward_batch 上存储 _attn_output；2）_attn_output 需要按 real_num_tokens 切片以匹配 FA 的 shape 校验；3）Op schema 的别名标注是消除拷贝的关键。全程无争议。

- 暂无高价值评论线程

风险与影响

- 风险：回归风险较低：条件拷贝路径 (if ret.data_ptr() != output.data_ptr()) 保持兼容性，其他后端不受影响。潜在风险包括：1）_attn_output 为 None 时不会传递 out，行为与改造前一致；2）CUDA graph 捕获场景需要正确切片；3）非 FA 后端（如 FlashInfer、Triton）不会使用 out 参数，因此无影响。性能上，通过 29MB 的 DtoD 拷贝每层消除约 14μs，但仅在 FA3 后端生效。
- 影响：对使用 FA3 后端的用户：前向性能提升 ~1.5%（低尾部延迟场景更明显）；对使用其他后端的用户：无影响；对开发者：需要确保新添加的 out 参数在所有 FA 调用路径中正确传递。不改变 API 或配置，用户透明。
- 风险标记：核心路径变更，兼容性（非 FA 后端），CUDA graph 切片

关联脉络

- PR #21734 相关性能优化的基础 PR: 该 PR 与本 PR 共同实现了性能提升（+21.2% 的基础上再提升 1.5%），实测数据来自这两个 PR 的组合。
- PR #21971 新增 fa_skip_kv_cache 路径的 PR: 本 PR 在 fa_skip_kv_cache 路径中也需要传递 out 参数，最终 commit 确保两者兼容。