PR #33063 完整报告

sgl-project/sglang

[trtllm_mha] perf: Stop allocating per-layer scratch inside the decode CUDA graph

合并时间：2026-08-05 10:27

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33063

执行摘要

- 一句话：消除 TRTLLM MHA 解码图内每层临时分配
- 推荐动作：值得精读。该 PR 虽然代码量小（+36/-2），但精准命中 CUDA 图捕获中的典型陷阱——把启动开销极小的 fill kernel 也烘焙进图回放。核心设计决策是：把易变对象的生命周期提升到 backend 实例、在图外一次性分配，并用 get_buffer 共享常量张量。建议重点关注 make_persistent_multi_ctas_kv_counter_buffer 的复用方式与 batch 上限推导，这对其他受 CUDA 图约束的后端优化有直接借鉴价值。

功能与动机

PR 动机是 TRTLLM MHA 解码路径中两个分配在每次 forward 都发出 fill kernel，并被 CUDA 图捕获烘焙进每次回放：一是 FlashInfer 在调用方未提供 counter buffer 时每次分配并清零新的 multi-CTA KV counter buffer，尽管其 docstring 说明内核会在 launch 结束时自我重置计数器；二是 fused_fp8_qkv_kv_cache 对无 checkpoint KV scale 的层用 torch.ones() 构建 float32 [1] scale 张量。这些 fill kernel 还位于两个 PDL 链接内核之间，导致 _fused_qk_rmsnorm_rope_gate 的 gdc_launch_dependents() 错误 signal 给 FillFunctor 而非 fused_fp8_qkv_kv_cache 的 PDLWaitPrimary。

实现拆解

实现分四步完成，全部改动集中在 trtllm_mha_backend.py，另有测试 mock 适配。

1. 后端构造时分配持久 multi-CTA counter buffer：在 TRTLLM MHA 后端类构造中调用从 trtllm_mla_backend 导入的 make_persistent_multi_ctas_kv_counter_buffer，以 config.num_attention_heads 为头数，以 (model_runner.max_running_requests + 1) * max(1, self.speculative_num_draft_tokens or 1) 为最大 batch 大小申请一次 buffer。该 buffer 在所有 decode 调用间复用，FlashInfer 内核按文档自我重置计数器，故只需归零一次。
2. 后端持有默认 float32 [1] scale 张量：通过 get_buffer 以 trtllm_mha_default_kv_scale 为键创建 torch.ones(1, dtype=torch.float32, device=self.device)，供所有无 checkpoint KV scale 的层共享，避免每次调用 torch.ones()。
3. 接线到调用点：在 _fused_fp8_qkv_kv_cache 中，当 layer.k_scale / v_scale 为 None 时回退到 self._default_kv_scale；在 forward_decode 和 forward_extend 的所有 trtllm_batch_decode_with_kv_cache 调用（普通 decode、ENCODER_ONLY target

verify、ragged verify、普通 verify）中传入 multi_ctas_kv_counter_buffer=self._multi_ctas_kv_counter_buffer。

4. 测试 mock 适配：在 dense_attention.py 的 MockModelRunner 中补充 self.max_running_requests = pool_batch_size，使新增构造逻辑在注意力单测中可访问该属性。

关键文件：

- python/sglang/srt/layers/attention/trtllm_mha_backend.py（模块 注意力后端；类别 source；类型 dependency-wiring；符号 init, _fused_fp8_qkv_kv_cache, forward_decode, forward_extend）：核心源码，将每层临时分配提升为 backend 构造时持有的持久 buffer，并接入所有 decode 调用点。
- python/sglang/test/kits/attention_unittest/attention_methods/dense_attention.py（模块 注意力测试；类别 test；类型 test-coverage；符号 MockModelRunner.init）：测试 mock 补充 max_running_requests 属性，确保新增后端构造逻辑在注意力单测中可用。

关键符号：init, _fused_fp8_qkv_kv_cache, forward_decode, forward_extend, MockModelRunner.init

关键源码片段

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

核心源码，将每层临时分配提升为 backend 构造时持有的持久 buffer，并接入所有 decode 调用点。

```
# 后端构造时（CUDA 图捕获之外）准备好所有持久 scratch，避免每层每次 forward 的 fill kernel 被烘焙进图回放。
# 1) multi-CTA KV counter buffer：FlashInfer 内核每次 launch 后自重置计数器，所以只需归零一次即可复用。
self._multi_ctas_kv_counter_buffer = make_persistent_multi_ctas_kv_counter_buffer(
    torch.device(self.device),
    num_q_heads=config.num_attention_heads,
    # 按最坏 batch（含 speculative draft tokens）申请；FlashInfer 拒绝过小的 buffer。
    max_batch_size=(model_runner.max_running_requests + 1)
    * max(1, self.speculative_num_draft_tokens or 1),
)

# 2) 默认 FP8 KV scale：无 checkpoint KV scale 的层共用一个 float32 [1] 的 1.0 张量。
self._default_kv_scale = get_buffer(
    "trtllm_mha_default_kv_scale",
    lambda: torch.ones(1, dtype=torch.float32, device=self.device),
)

def _fused_fp8_qkv_kv_cache(self, q, k, v, layer, forward_batch):
    # 获取本层 cache 位置与 K/V buffer。
    cache_loc = self._get_layer_cache_loc(layer, forward_batch)
    k_cache, v_cache = self.token_to_kv_pool.get_kv_buffer(layer.layer_id)
    return fused_fp8_qkv_kv_cache(
```

```
        q=q, k=k, v=v,
        k_cache=k_cache, v_cache=v_cache,
        cache_loc=cache_loc,
        # 层没有检查点 scale 时回退到 backend 持有的常量 1.0，避免每次构建新张量。
        k_scale=layer.k_scale if layer.k_scale is not None else self._default_kv_scale,
        v_scale=layer.v_scale if layer.v_scale is not None else self._default_kv_scale,
    )
```

评论区精华

唯一的核心 review 讨论来自 b8zhong 对持久 counter buffer 尺寸是否覆盖 prefill 的疑问。他在第 239 行询问："will this create a large enough buffer for prefill, or is prefill not affected by this change"。mattteochen 回复：普通 prefill 不受 counter buffer 影响，因为该 buffer 仅传给 decode 与 speculative verify/draft-extend 调用，且大小按最坏情况 bs × num_draft_tokens 计算；默认 KV scale 虽适用于 fused-FP8 prefill，但它是跨所有 token 共享的标量 [1] 张量，与 batch 或 prefill 长度无关。讨论结论清晰，双方无分歧，随后 b8zhong 与 Fridge003 均 APPROVED。

- 持久 counter buffer 对 prefill 的影响 (question): 作者澄清了影响范围，确认 prefill 无风险，讨论闭环。

风险与影响

- 风险：主要风险有三点：

1. 依赖 FlashInfer 内核自重置契约：复用 counter buffer 的前提是 trtllm_batch_decode_with_kv_cache 每次 launch 后自我清零计数器。若后续 FlashInfer 版本破坏该契约或存在未覆盖路径（如非预期的新调用点忘记传 buffer），会导致计数器越用越脏。当前实现未添加显式断言或归零逻辑。
2. buffer 大小计算可能低估：max_batch_size 用 max_running_requests + 1 乘以 draft token 数，但若未来出现更大的验证 batch（如多步验证或 MTP 扩展）可能超出。FlashInfer 拒绝过小 buffer，过大会报错，但不会静默截断；若超限会直接失败而非数据错误。
3. 测试覆盖不足：没有新增专门针对持久 buffer 复用和默认 scale 共享的单元测试，仅适配了现有 mock。回归检测依赖既有注意力测试和 GSM8K 等端到端指标，GSM8K 分数波动（0.822 对 0.813）在 run-to-run 约 2pp 的噪声内，但未做统计显著性验证。 - 影响：影响范围限定在 trtllm_mha 后端（Blackwell SM100 解码路径）。对使用 Qwen3.5 等 FP8 大模型在 B200 上运行的场景，每个 decode step 减少约 90us（约 1.64% 延迟），并移除每层 3 个 fill kernel 启动，降低 GPU 占用和调度开销。对团队而言，该模式（构造时分配持久 buffer、图外准备常量）可推广到其他 CUDA 图后端，作为避免图内动态分配的参考实现。由于改动只在后端内部，对 prefill 和其他后端无影响。 - 风险标记：核心路径变更，依赖第三方内核契约，缺少针对性测试覆盖

关联脉络

- PR #33306 Avoid TRTLLM prefill output copy: 同样针对 trtllm_mha_backend.py 消除不必要的显存拷贝与启动开销，与本 PR 同属 TRTLLM MHA 后端的 CUDA 图性能优化线。