PR #52836 完整报告

vllm-project/vllm

Revert DSv4 eager workspace reuse

合并时间：2026-08-19 08:02

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/52836

执行摘要

- 一句话：回滚 DSv4 *eager* workspace 复用，恢复 allocator 安全分配
- 推荐动作：值得精读。该 PR 是理解 GPU 多流编程与 CUDAGraph 捕获交互的典型案例：
  - 核心教训是绕过 *caching allocator* 自定义 workspace 池时必须自行管理 stream 依赖，否则跨流覆写会成为隐蔽的正确性炸弹。
  - 展示了如何在性能优化引入安全风险时果断回滚，并依赖后续架构改动（#51430、#52401）保留大部分性能收益。
  - 回滚前应核对模型级验证是否完成，作者已诚实标注为 draft 状态且未跑模型评估，阅读时应关注合并后的补测结果。

功能与动机

PR body 明确指出：模型级 scratch pool 可以跨层和 CUDA 流重用存储，但缺少 *caching allocator* 的 stream/event 生命周期跟踪，允许 producer 覆写共享 workspace 而之前的 consumer 可能仍在使用它。回滚是为了恢复 pre-#49236 的分配生命周期行为，直到实现安全的跨流同步。作者还补充说明 #51430 已把 Q projection/KV 插入、indexer 准备、compressor 准备移出窄 *eager* break 并进入 *captured* region，因此回滚在 Model Runner V2 上不会放弃 #49236 报告的 3.9% TTFT 改进；而 #52401 为 Model Runner V1 恢复宽 *eager* region 的原因是该窄区域会破坏 V1 输出，回滚后该路径 TTFT 影响尚未测量。

实现拆解

1. 删除 scratch pool 类：移除 vllm/models/deepseek_v4/*eager*_scratch.py（-137 行），包括 DeepseekV4EagerScratchPool 的 __init__、_packed_size、_views、q_out、compressor_scratch、indexer_q_outputs、global_topk_outputs 等方法。该类通过单个 uint8 存储按 256 字节对齐切分 FP4 索引器、global topk、compressor 三类临时缓冲，并缓存不同 num_tokens 的 slice 视图。
2. 清理注意力层数据契约：在 vllm/models/deepseek_v4/attention.py 中移除 *eager*_scratch_pool 参数、self.*eager*_scratch_pool 属性、_global_topk_output_buffers 方法，以及 _fused_qnorm_rope_kv_insert 中走 fused_deepseek_v4_qnorm_rope_kv_rope_quant_insert_out 的分支，恢复直接调用返回新张量的 fused_deepseek_v4_qnorm_rope_kv_rope_quant_insert；DeepseekV4Indexer 的 wq_b_and_q_quant 中 fused_indexer_q_rope_quant 的

output_buffers 参数也被移除。

3. 清理模型与压缩机装配：在 vllm/models/deepseek_v4/nvidia/model.py 中删除 eager_scratch_pool 的创建（含 ubatching 判断、padded heads 计算）及向 DeepseekV4DecoderLayer 的传递；在 vllm/models/deepseek_v4/compressor.py 中删除 eager_scratch_pool 参数和 compressor_scratch 的使用，恢复 cutedsl 路径自身分配 scratch。
4. 移除 out-param 算子：在 csrc/libtorch_stable/ops.h 和 torch_bindings.cpp 中删除 fused_deepseek_v4_qnorm_rope_kv_rope_quant_insert_out 的声明、schema 定义和 CUDA 实现绑定；同步修改 fused_deepseek_v4_qnorm_rope_kv_insert_kernel.cu、fused_indexer_q.py、sparse_attn_compress_cutedsl.py、cache_utils.py 中相关 output_buffers 支持。
5. 测试配套：删除 tests/kernels/test_compressor_kv_cache.py 中的 test_compute_global_topk_reuses_output_buffers，删除 tests/kernels/test_fused_indexer_q_rope_quant.py 中的 output-buffer 复用断言，调整 tests/kernels/test_fused_deepseek_v4_qnorm_rope_kv_insert.py 以适配恢复后的算子签名；作者自述运行 240 passed、11 skipped。

关键文件：

- vllm/models/deepseek_v4/eager_scratch.py（模块 临时缓冲；类别 source；类型 deletion；符号 DeepseekV4EagerScratchPool, init, _packed_size, _views）：被整体删除的文件，包含回滚的核心对象 DeepseekV4EagerScratchPool。该类是 #49236 引入的模型级共享 workspace 池，通过单个 uint8 存储按对齐切片复用 FP4 indexer、global topk、compressor 的临时缓冲，正是跨流生命周期风险的来源。
- vllm/models/deepseek_v4/attention.py（模块 注意力层；类别 source；类型 data-contract；符号 _global_topk_output_buffers, DeepseekV4Attention.init, DeepseekV4Attention._fused_qnorm_rope_kv_insert, DeepseekV4Indexer.init）：注意力输入准备路径的核心文件。移除 eager_scratch_pool 参数、_global_topk_output_buffers 方法以及 out-param 算子分支，恢复 kernel 内部 allocator 分配输出张量，是本次回滚的直接落地处。
- vllm/models/deepseek_v4/nvidia/model.py（模块 模型构建；类别 source；类型 data-contract；符号 DeepseekV4DecoderLayer.init, DeepseekV4Model.init）：模型装配入口。删除 DeepseekV4EagerScratchPool 的创建逻辑（含 padded heads 计算与 ubatching 排除判断），并停止向各 DecoderLayer 传递 pool，是移除整个复用链路的源头。
- vllm/models/deepseek_v4/compressor.py（模块 压缩器；类别 source；类型 data-contract；符号 DeepseekCompressor.init, DeepseekCompressor.forward）：压缩机路径。移除 eager_scratch_pool 参数及 compressor_scratch 的传入，cutedsl 路径恢复自身分配 scratch，与注意力层回滚保持一致。
- csrc/libtorch_stable/ops.h（模块 CUDA 内核；类别 source；类型 core-logic；符号 fused_deepseek_v4_qnorm_rope_kv_rope_quant_insert_out）：C++ op 接口层。删除 out-param 算子 fused_deepseek_v4_qnorm_rope_kv_rope_quant_insert_out 的声明，与其在 torch_bindings.cpp 的 schema 注册和 CUDA 实现绑定一起被移除。

- vllm/models/deepseek_v4/nvidia/flashinfer_sparse.py（模块 推理后端；类别 source；类型 data-contract；符号 _forward_decode, _forward_prefill）：稀疏注意力后端。移除 decode/prefill 路径中 compute_global_topk_indices_and_lens 的 output_buffers 参数，恢复函数内部自行分配结果缓冲。
- tests/kernels/test_compressor_kv_cache.py（模块 测试用例；类别 test；类型 test-coverage；符号 test_compute_global_topk_reuses_output_buffers）：测试配套。删除 test_compute_global_topk_reuses_output_buffers，该测试断言复用缓冲的数据指针相等，与回滚后的 allocator-back 行为矛盾。
- tests/kernels/test_fused_indexer_q_rope_quant.py（模块 测试用例；类别 test；类型 test-coverage）：测试配套。移除 output_buffers 复用场景的构造与指针断言，避免测试依赖已删除的复用语义。

关键符号：DeepseekV4EagerScratchPool.init, DeepseekV4EagerScratchPool.global_topk_outputs, DeepseekV4Attention.init, DeepseekV4Attention._fused_qnorm_rope_kv_insert, DeepseekV4Attention._global_topk_output_buffers, DeepseekV4Indexer.wq_b_and_q_quant, DeepseekCompressor.forward, DeepseekV4Model.init

关键源码片段

vllm/models/deepseek_v4/eager_scratch.py

被整体删除的文件，包含回滚的核心对象 DeepseekV4EagerScratchPool。该类是 #49236 引入的模型级共享 workspace 池，通过单个 uint8 存储按对齐切片复用 FP4 indexer、global topk、compressor 的临时缓冲，正是跨流生命周期风险的来源。

```
# 被删除的 DeepseekV4EagerScratchPool 是 #49236 引入的模型级共享 workspace 池。
# 它把多类临时缓冲打包进单个 uint8 storage，按 256 字节对齐切块，并按 num_tokens
缓存切片视图。
# 危险点：这个 pool 不经过 caching allocator，无法感知 CUDA stream/event 依赖；
# 不同 layer 或不同 stream 的 producer/consumer 可能读写同一块内存，产生覆写竞态。
class DeepseekV4EagerScratchPool:
    _ALIGNMENT = 256

    def __init__(self, max_num_tokens, padded_q_heads, q_head_dim,
                 index_q_heads, index_q_head_dim, index_topk, device):
        self._q = torch.empty(
            (max_num_tokens, padded_q_heads, q_head_dim),
            dtype=torch.bfloat16, device=device)
        # FP4 indexer 输出、global topk、compressor scratch 共享一块 aux storage，
        # 用 max 而非 sum 计算大小，因为三者不同时使用（见下方注释）。
        aux_bytes = max(
            self._packed_size(specs)
            for specs in (fp4_specs, global_specs, compressor_specs)
        )
        storage = torch.empty(aux_bytes, dtype=torch.uint8, device=device)
        # ... 后续通过 storage 切出各模板视图，再按 num_tokens 缓存切片 ...
```

```
    def global_topk_outputs(self, topk_indices):
        # 返回缓存的 (indices, lens) 切片，调用方写完后下一个 layer 会覆盖同一块内存
        output = self._global_outputs.get(num_tokens)
        if output is None:
            indices, lens = self._global_template
            output = (indices[:num_tokens], lens[:num_tokens])
            self._global_outputs[num_tokens] = output
        return output
```

vllm/models/deepseek_v4/attention.py

注意力输入准备路径的核心文件。移除 *eager_scratch_pool* 参数、*_global_topk_output_buffers* 方法以及 *out-param* 算子分支，恢复 *kernel* 内部 *allocator* 分配输出张量，是本次回滚的直接落地处。

```
# Revert #49236 后恢复的 fp8_ds_mla paged 路径：
# Q 侧 norm+RoPE、KV 侧 quant+insert 仍由 fused kernel 完成；
# 但 q 的 padded 输出不再来自模型级 scratch pool，
# 而是由 kernel 内部通过 caching allocator 分配并返回新张量。
if cache_dtype == torch.uint8:
    # fp8_ds_mla UE8M0 paged path。水平融合：
    # Q 侧：per-head RMSNorm（无 weight）+ GPT-J RoPE，零填充 padding head 槽；
    # KV 侧：GPT-J RoPE + UE8M0 FP8 quant + paged cache insert。
    # kernel 分配并返回 padded q 张量，生命周期由 caching allocator 跟踪。
    swa_kv_cache_2d = swa_kv_cache.view(swa_kv_cache.shape[0], -1)
    return torch.ops._C.fused_deepseek_v4_qnorm_rope_kv_rope_quant_insert(
        q, kv, swa_kv_cache_2d, swa_metadata.slot_mapping,
        positions, cos_sin_cache, self.padded_heads,
        self.eps, swa_metadata.block_size)

# 注意：回滚前这里存在 eager_scratch_pool is not None 的分支，
# 会先取 pool 的 q_out 视图再调用 ..._insert_out 写进去。
# 该做法绕过了 caching allocator 的 stream/event 生命周期跟踪，
# 可能导致前一个 consumer 尚未读完时，下个 producer 已覆写同一块存储。
```

评论区精华

该 PR 没有实质性的 *code review* 讨论，只有两个机器人评论。核心决策均由作者在 PR body 中说明：

- mergify[bot] 指出 *pre-commit* 检查失败并要求安装后运行，属于流程性提示。
- 作者执行 /ci run 触发 Buildkite CI #84486。
- 作者在 body 中阐述了最关键的权衡：模型级 scratch pool 会绕过 *caching allocator* 的 *stream/event* 生命周期跟踪，存在跨流覆写风险；回滚后 MRV2 因 #51430 的捕获区改造不受性能影响，MRV1 因 #52401 的宽 *eager region* 存在未测量的 TTFT 回退。
- pre-commit 检查失败 (style)：作者后续执行 /ci run 触发 CI，检查问题已处理。
- 跨流并发安全与性能权衡 (design)：采用回滚以恢复安全生命周期；MRV2 上因 #51430 捕获区改造保留性能收益，MRV1 需后续测量。

- CI 触发 (other): CI 已运行。

风险与影响

- 风险：
  1. 性能回退风险：MRV1 路径上，torch.empty 分配发生在 *eager* 区域，TTFT 可能回退，PR 作者明确表示未在当前树测量。
  2. 缺少模型级验证：作者自述 DeepSeek-V4 checkpoint 不在本地，未运行 TP4 serving 与 GSM8K 评估，正确性只由 kernel 级单测覆盖。
  3. C++ 接口变更：移除 fused_deepseek_v4_qnorm_rope_kv_rope_quant_insert_out 会破坏依赖该 out-param 算子的外部代码或第三方后端，需要重新编译 CUDA 扩展。
  4. 内存开销变化：从模型级共享 single storage 恢复到按层分配临时张量，显存峰值可能上升，但生命周期由 caching allocator 精确管理，消除跨层 / 跨流悬垂引用风险。
  5. 低风险点：回滚本身是代码删除，不引入新逻辑，主要风险集中在性能与验证覆盖。 - 影响：对用户：DeepSeek V4 推理在真实并发 / 多流场景下的正确性提升，避免因 workspace 被覆写导致的输出损坏或稳定性问题。对性能：MRV2 路径 TTFT 保持（captured region 内分配无额外开销），MRV1 路径可能回退但未量化。对系统：显存分配模式从静态共享池变为每层临时分配，依赖 caching allocator 的 stream/event 跟踪，内存管理更安全；同时删除约 354 行代码，降低维护面。对团队：该变更需要重新编译 CUDA ops，并建议在合并后尽快补充 TP4/GSM8K 验证以确认 MRV1 性能影响。 - 风险标记：核心路径变更，性能回退风险，缺少模型评估，C++ 接口变更，跨流并发修复

关联脉络

- PR #49236 [DSv4 Perf] Optimize workspace reuse for eager break: 本 PR 回滚的原始 PR，引入了 DeepseekV4EagerScratchPool 与 out-param 算子。
- PR #51430 [V1] Move DSv4 input preparation into captured region: 该 PR 将 Q projection/KV 插入、indexer 准备、compressor 准备移入 captured region，使回滚后 MRV2 路径的分配开销变为 captured，性能不损失。
- PR #52401 [MRV1] Restore wide eager region for DSV4 attention: 该 PR 为 Model Runner V1 恢复宽 *eager* region，回滚后该路径分配仍在 *eager* 侧，TTFT 影响未测量。