PR #41002 完整报告

vllm-project/vllm

[ROCm][perf] Use workspace manager for sparse indexer allocations

合并时间：2026-06-05 14:46

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/41002

执行摘要

- 一句话：使用 workspace manager 替换 ROCm 稀疏索引器动态分配
- 推荐动作：建议相关开发者仔细阅读，尤其是 workspace manager 的使用模式，以及如何在不影响 torch.compile 的情况下预留内存。对于 ROCm 稀疏索引器的维护者，这是一次重要的对齐。

功能与动机

原始 PR body 提到“重复的动态 torch.empty/torch.full 分配应被 workspace manager 替换”，因为“有界的 scratch 池复用，更少的分配器抖动，更低的碎片或后期热路径分配意外风险”。此外，使用 workspace manager 能使 ROCm 稀疏预填充模式与 CUDA 索引器模式保持一致。

实现拆解

1. 导入与初始化：在文件头部导入 current_workspace_manager。
2. *decode* logits 分配：在 rocm_fp8_paged_mqa_logits 中，将 out_logits 和 out_qk 的创建从 torch.full(...) 改为 current_workspace_manager().get_simultaneous(...) 返回元组后调用 fill_(float("-inf"))。
3. profiling 路径预留：在 rocm_aiter_sparse_attn_indexer 的 profiling 分支（当 attn_metadata 不是 dict 时），调用 workspace_manager.get_simultaneous 预留 prefill 阶段需要的 k_fp8、k_scale 缓冲区以及 *decode* 阶段需要的 logits 缓冲区大小。这些大小基于 total_seq_lens（即 max_prefill_buffer_size）和 head_dim 等固定值，确保 profiling 时已预留足够空间。
4. prefill 路径使用：在 has_prefill 分支中，通过 workspace_manager.get_simultaneous 获取 k_fp8_full 和 k_scale_full 视图，然后在循环中按 chunk 切片使用。
5. torch.compile 兼容：profiling 路径被封装在真实实现（real impl）中，避免在 FakeTensor 模式下调用 workspace manager，保护 PyTorch 的 dispatch 状态。 （测试配套：无专门单元测试，但通过完整的 MTP 和准确率实验验证。）

关键文件：

- vllm/v1/attention/ops/rocm_aiter_mla_sparse.py（模块 稀疏索引器；类别 infra；类型 infrastructure；符号 rocm_fp8_paged_mqa_logits, rocm_aiter_sparse_attn_indexer, current_workspace_manager）：唯一修改文件，核心变更涉及 workspace manager 替换动态分配

关键符号：rocm_fp8_paged_mqa_logits, rocm_aiter_sparse_attn_indexer

关键源码片段

vllm/v1/attention/ops/rocm_aiter_mla_sparse.py

唯一修改文件，核心变更涉及 workspace manager 替换动态分配

```
# 导入 workspace manager 工具函数
from vllm.v1.worker.workspace import current_workspace_manager

def rocm_aiter_sparse_attn_indexer(...):
    # ... 前面逻辑
    if not isinstance(attn_metadata, dict):
        #・仅是 profiling 分支，不是热路径・
        workspace_manager = current_workspace_manager()

        # 预填充 k_fp8 和 k_scale 缓冲区（一次分配，后续复用）
        # total_seq_lens = max_model_len * 40，固定上限
        workspace_manager.get_simultaneous(
            ((total_seq_lens, head_dim), fp8_dtype), # k_fp8 的形状
            ((total_seq_lens, 4), torch.uint8), # k_scale 的形状
        )

        # decode logits 缓冲区
        if _ON_GFX942 or _ON_GFX950:
            # 根据环境变量预留大小
            max_logits_mb = envs.VLLM_SPARSE_INDEXER_MAX_LOGITS_MB
            ... # 调用 workspace_manager.get_simultaneous
        return

    #・运行时 prefill 路径・
    if has_prefill:
        workspace_manager = current_workspace_manager()
        # 获取 workspace 视图（如果已由 profiling 预留，则直接返回对应视图）
        k_fp8_full, k_scale_full = workspace_manager.get_simultaneous(
            ((total_seq_lens, head_dim), fp8_dtype),
            ((total_seq_lens, 4), torch.uint8),
        )
        for chunk in prefill_metadata.chunks:
            # 每个 chunk 只取所需的切片，避免重新分配
            k_fp8 = k_fp8_full[:chunk.total_seq_lens]
            k_scale = k_scale_full[:chunk.total_seq_lens]
            ... # 调用 AITER 内核

def rocm_fp8_paged_mqa_logits(...):
    #・decode logits 分配・
    # 原代码：out_logits = torch.full(..., float("-inf"), ...)
    # 新代码：
    (out_logits,) = current_workspace_manager().get_simultaneous(
```

```
        ((batch_size * next_n, max_model_len), torch.float32),
    )
    out_logits.fill_(float("-inf"))
    # 同理 out_qk
```

评论区精华

讨论 1：profiling 路径使用 total_seq_lens 是否安全

- gemini-code-assist[bot] 指出 profiling 时使用当前 batch 的 total_seq_lens 可能小于运行时的值，导致 workspace 不足。
- tuukkjs 解释 total_seq_lens 实际上是 max_prefill_buffer_size（max_model_len * 40），是一个固定上限，与 CUDA 路径一致，不会出现不足。
- 结论：设计正确，已解决。

讨论 2：envs 调用是否影响热路径性能

- tjtanaa 认为 envs.VLLM_SPARSE_INDEXER_MAX_LOGITS_MB 不应在热路径调用（CPU 开销大）。
- tuukkjs 澄清该调用位于 profiling 分支（not isinstance(attn_metadata, dict)），不是热路径。
- tjtanaa 随后确认仅 profiling 时调用，不再有异议。

讨论 3：预填充缓冲区分配大小是否过大

- gemini-code-assist[bot] 建议只分配最大 chunk 大小，而非整个 total_seq_lens。
- tuukkjs 回应 total_seq_lens 已经是最大 chunk 大小（max_prefill_buffer_size），且 profiling 后 size 固定，无浪费。
- 结论：当前设计合理。
- Profiling 路径使用 total_seq_lens 是否安全 (design): total_seq_lens 实际是固定上限（max_model_len * 40），不是 batch 依赖的值，且与 CUDA 路径一致，不会不足。
- envs.VLLM_SPARSE_INDEXER_MAX_LOGITS_MB 是否在热路径 (performance): 仅在 profiling 路径，不影响运行时性能。
- 预填充 k_fp8 缓冲区分配大小是否应使用最大 chunk 而非 total_seq_lens (design): total_seq_lens 已经是 max_prefill_buffer_size（最大 chunk 大小），且 profiling 后 size 固定，无浪费。

风险与影响

- 风险：
  1. workspace 预留不足风险：profiling 时使用 total_seq_lens = max_model_len * 40 作为上限，该值在运行时恒定，因此预留总量固定，不会不足。风险低。
  2. envs 调用性能开销：envs.VLLM_SPARSE_INDEXER_MAX_LOGITS_MB 仅在 profiling 路径调用，不影响热路径。风险消除。
  3. 仅影响 ROCm 特定路径：改动仅限于 vllm/v1/attention/ops/rocm_aiter_mla_sparse.py，不干扰其他平台或通用逻辑。

4. torch.compile 兼容性：workspace 操作被正确隔离在 real impl 中，fake impl 不调用，不会破坏编译。

- 影响：
  - 对用户：无行为变化，仅需确保使用 ROCM_AITER_MLA_SPARSE 后端。预期热路径分配减少，可能提升吞吐量并减少 P99 延迟抖动。
  - 对系统：减少 PyTorch 分配器调用，降低内存碎片和保留内存与分配内存的差值（尤其在 MTP 多步时）。
  - 对团队：代码架构更有规范性，缩小与 CUDA 路径的差距，便于后续维护。
  - 风险标记：workspace 预留依赖固定上限，envs 调用潜在开销（已确认安全）

关联脉络

- PR #38704 stale: [ROCm] Use workspace manager for sparse indexer allocations: 本 PR 继续了 stale PR #38704 的工作，由相同作者之一贡献。