PR #38502 完整报告

vllm-project/vllm

[ROCm] Cap Triton paged attention block size to fix ROCm shared memory OOM

合并时间：2026-05-10 18:03

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/38502

执行摘要

- 一句话：限制 Triton 注意力块大小为 128 修复 ROCm 共享内存溢出
- 推荐动作：值得仔细阅读，尤其是 has_native_kv_cache_layout 的设计和 TRITON_BLOCK_SIZE 硬上限的权衡。展示了如何通过块布局检测与内核选择协同解决硬件兼容性问题，对理解 ROCm 注意力实现和 Hybrid Mamba 支持有参考价值。

功能与动机

Hybrid Mamba models (e.g. Jamba) inflate block_size to 2048 to align attention and Mamba page sizes. When the ROCm custom paged attention kernel rejects this (it only supports 16/32), the Triton fallback kernel_paged_attention_2d used 2048 as its tile size, requesting 262144 bytes of shared memory and thus exceeding the MI325X hardware limit of 65536 bytes. Cap TRITON_BLOCK_SIZE at 128. The kernel already decouples tile size from physical block size via l_block_idx/internal_offsets addressing, so this is safe.

实现拆解

1. 新增连续布局检测函数：在 vllm/v1/attention/ops/chunked_prefill_paged_decode.py 中新增 has_native_kv_cache_layout 函数，通过比较 key_cache.stride(0) 与 key_cache.shape[1:].numel() 判断 KV cache 块是否连续，从而决定使用原生 HIP 还是 Triton 路径。
2. 修改 KV cache 写入路径：在 vllm/v1/attention/backends/rocm_attn.py 的 do_kv_cache_update 方法中，原条件 block_size in (16, 32) 不足以安全使用原生 write_to_paged_cache；新增 has_native_layout 检查后，只有块大小标准且布局连续时才走原生路径，否则回退到 Triton 的 reshape_and_cache_flash。
3. 限制 Triton tile 大小：在 chunked_prefill_paged_decode 中引入 MAX_TRITON_BLOCK_SIZE = 128，对 Triton 的 tile 大小硬上限，避免从物理 block_size（如 2048）推导出过大 tile 导致共享内存溢出。同时对于非连续布局（not has_native_layout）强制禁用原生 kernel（use_custom = False），确保解码路径与缓存更新路径一致。
4. 更新 CI 配置：在 .buildkite/test-amd.yaml 中添加 hybrid_model 测试步骤，安装来自作者分支的 mamba 与 causal-conv1d 依赖，运行混合模型测试以验证修复效果。

关键文件：

- vllm/v1/attention/backends/rocm_attn.py（模块 注意力后端；类别 source；类型 core-logic）：核心 ROCm 注意力后端，修改 do_kv_cache_update 方法，根据块连续性和大小选择正确的 KV 缓存写入路径。
- vllm/v1/attention/ops/chunked_prefill_paged_decode.py（模块 解码算子；类别 infra；类型 infrastructure；符号 has_native_kv_cache_layout）：定义了 has_native_kv_cache_layout 函数和 MAX_TRITON_BLOCK_SIZE 上限，是修复的核心。
- .buildkite/test-amd.yaml（模块 CI 配置；类别 config；类型 configuration）：添加 hybrid_model 测试步骤，确保修复后的功能在 CI 中验证。

关键符号：has_native_kv_cache_layout, do_kv_cache_update, chunked_prefill_paged_decode

关键源码片段

vllm/v1/attention/backends/rocm_attn.py

核心 ROCm 注意力后端，修改 do_kv_cache_update 方法，根据块连续性和大小选择正确的 KV 缓存写入路径。

```
    def do_kv_cache_update(
        self,
        layer: AttentionLayer,
        key: torch.Tensor,
        value: torch.Tensor,
        kv_cache: torch.Tensor,
        slot_mapping: torch.Tensor,
    ):
        if self.attn_type in (AttentionType.ENCODER_ONLY, AttentionType.ENCODER):
            return
        key_cache, value_cache = PagedAttention.split_kv_cache(
            kv_cache, self.num_kv_heads, self.head_size
        )

        # 从 value_cache 获取实际 block_size
        # value_cache 形状：[num_blocks, num_heads, head_size, block_size]
        block_size = value_cache.shape[3]
        # 检查 KV cache 布局是否连续
        has_native_layout = has_native_kv_cache_layout(key_cache, value_cache)

        if block_size in (16, 32) and has_native_layout:
            # 标准 16, 32 且布局连续：使用 vLLM 原生 HIP C++ 逻辑
            PagedAttention.write_to_paged_cache(
                key,
                value,
                key_cache,
                value_cache,
                slot_mapping,
```

```
                self.kv_cache_dtype,
                layer._k_scale,
                layer._v_scale,
            )
        else:
            # 非标准块大小或混合注意力 /Mamba 布局需要 stride-aware 的
            # Triton 写入器。原生 reshape_and_cache kernel 假设连续的
            # 块存储，写入混合缓存块时会出错。
            triton_reshape_and_cache_flash(
                key,
                value,
                key_cache,
                value_cache,
                slot_mapping,
                self.kv_cache_dtype,
                layer._k_scale,
                layer._v_scale,
            )
```

vllm/v1/attention/ops/chunked_prefill_paged_decode.py

定义了 has_native_kv_cache_layout 函数和 MAX_TRITON_BLOCK_SIZE 上限，是修复的核心。

```
# has_native_kv_cache_layout 检测 KV cache 块布局是否连续
# 原生 reshape_and_cache 写入器假设 packed 块；如果缓存更新需要
# reshape_and_cache_flash（针对 stride-padded 混合布局），则解码
# 也应使用匹配的 Triton 路径。
def has_native_kv_cache_layout(
    key_cache: torch.Tensor,
    value_cache: torch.Tensor,
) -> bool:
    return (
        key_cache.stride(0) == key_cache.shape[1:].numel()
        and value_cache.stride(0) == value_cache.shape[1:].numel()
    )


# ... 在 chunked_prefill_paged_decode 函数内部 ...
# Triton tile 大小的硬上限，解决 Hybrid Mamba 模型因 block_size 为 2048
# 导致共享内存申请超过 65536 字节（MI325X 硬件限制）的问题。
# 内核通过 l_block_idx / internal_offsets 解耦 tile 与物理块大小，
# 因此 128 的上限不影响功能正确性。
MAX_TRITON_BLOCK_SIZE = 128
# 对于 2 的幂的 block_size，使用 min(block_size, 128) 确保不超限；
# 非 2 的幂（如 544）则固定使用 32。
TRITON_BLOCK_SIZE = min(block_size, MAX_TRITON_BLOCK_SIZE) if is_pow2 else 32
```

评论区精华

- 硬编码 128 上限 vs 动态计算：gemini-code-assist 建议动态计算最大块大小保持通用性。Andreas 回应这是 Triton 3.6 的问题，128 对当前模型安全，未来版本会修复。
- 连续性检测方法：hmellor 提出使用 is_contiguous()，Andreas 解释需要块级而非张量级连续性，当前 stride 比较更高效且正确。
- 是否仅 ROCm 特定：hmellor 询问 128 上限是否 ROCm 特有，Andreas 与 tjtanaa 确认该操作仅用于 ROCm。
- kernel 选择对齐：gshtras 指出原生 kernel 选择逻辑未同步，Andreas 随后更新 use_custom 决策以与 cache update 路径保持一致。
- ssd_chunk_scan 修改的撤回：最初对 ssd_chunk_scan.py 的修正被视为必要，后经 micah-wil 提醒确认 Triton PR#9541 已修复编译器崩溃，遂还原该文件。
  - 硬编码 MAX_TRITON_BLOCK_SIZE 与动态计算 (performance): 维持硬编码 128，后续可跟进动态计算。
  - 使用 is_contiguous 替代 stride 比较 (correctness): 保留 stride 比较方式。
  - 128 上限是否仅 ROCm 特有 (question): 确认仅适用于 ROCm。
  - 原生 kernel 选择逻辑与 cache update 路径对齐 (design): 已同步更新 kernel 选择逻辑。
  - ssd_chunk_scan 修改的撤回 (correctness): ssd_chunk_scan.py 已恢复原状。

风险与影响

- 风险：
  - 硬编码性能限制：MAX_TRITON_BLOCK_SIZE=128 可能限制未来更大块大小模型的性能，但目前所有已知适用模型块大小 ≤ 128，无影响。
  - 连续性判断开销：每次 do_kv_cache_update 调用均执行 stride 比较，开销极低可忽略。
  - CI 依赖非官方分支：测试步骤依赖作者 fork 的 mamba 仓库，可能不稳定或未及时上游合并，长期应将依赖切换至官方版本。
  - 路径一致性风险：若未来新增其他块大小或布局，需同步更新 do_kv_cache_update 与 chunked_prefill_paged_decode 中的条件，否则可能出现写与读路径不匹配。
- 影响：
  - 用户影响：ROCm 用户现在可以运行 Hybrid Mamba 模型（如 Jamba），避免之前因共享内存 OOM 而崩溃。对其他模型无负面影响。
  - 系统影响：仅修改 ROCm 注意力后端与非标准块大小的 Triton 路径，非 ROCm 平台不受影响。性能方面标准模型（block_size 16/32）仍使用原生路径，无回归；非标准块模型（如 544）从不可用变为可用，但 Triton 路径可能稍慢。
  - 团队影响：需维护两套写入路径和对应的连续性检查逻辑，增加了代码复杂度。
  - 风险标记：硬编码共享内存限制，ROCm 路径变更，CI 依赖作者分支，写入与解码路径一致性

关联脉络

- 暂无明显关联 PR