PR #23596 完整报告

sgl-project/sglang

[AMD] Fix memory access fault when `--page-size > 1` with speculative decoding on AMD GPUs

合并时间：2026-04-24 14:56

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23596

执行摘要

- 一句话：修复 AMD GPU 上使用默认页大小 + 推测解码时的内存访问错误
- 推荐动作：PR 值得快速合并，是专为 AMD GPU 上的 Triton 编译器兼容性问题而设计的安全修复。建议后续为 get_last_loc_triton_safe 添加单元测试，以避免类似编译器回归。

功能与动机

在 AMD GPU 上使用 --page-size 16 和推测解码（EAGLE3）运行服务器时，出现 HSA 内存访问错误（memory access fault）。该问题仅在 HIP/Triton 后端下触发，根源是 Triton HIP 编译器对混合宽度存储的错误编译。

实现拆解

1. 导入与全局检测（python/sglang/srt/mem_cache/common.py）：新增 from sglang.srt.utils import is_hip 导入，并定义模块级常量 _is_hip = is_hip()，用于在模块作用域内提前判断当前运行时是否为 HIP 平台。
2. 新增安全内核（common.py）：定义 _get_last_loc_safe_kernel Triton JIT 内核，其输出缓冲区 result_i32 固定为 int32 类型。内核内部根据 PREFIX_DTYPE_IS_I64 条件选择是否进行类型提升，最终存储结果时保持 int32，避免混合宽度存储。
3. 新增安全函数封装（common.py）：定义 get_last_loc_triton_safe，在 Triton 外分配 int32 类型的输出张量，调用安全内核，最后用 .to() 提升到与输入一致的 dtype，从而在 Triton 之外完成类型转换。
4. 修改调度逻辑（get_last_loc）：将原先的 attention_backend 条件提取为 uses_triton_dispatch 变量，并在其之上增加 _is_hip and uses_triton_dispatch 条件分支，将 HIP 平台下本应走 get_last_loc_triton 的路径全部导向 get_last_loc_triton_safe。非 HIP 平台保持不变。

关键文件：

- python/sglang/srt/mem_cache/common.py（模块 缓存层；类别 source；类型 core-logic；符号 _get_last_loc_safe_kernel, get_last_loc_triton_safe）：唯一修改的文件，包含所有核心变更：新增安全内核 _get_last_loc_safe_kernel、封装函数 get_last_loc_triton_safe，以及修改调度函数 get_last_loc 以在 HIP 路径下使用安全变体。

关键符号：_get_last_loc_safe_kernel, get_last_loc_triton_safe, get_last_loc

关键源码片段

python/sglang/srt/mem_cache/common.py

唯一修改的文件，包含所有核心变更：新增安全内核 _get_last_loc_safe_kernel、封装函数 get_last_loc_triton_safe，以及修改调度函数 get_last_loc 以在 HIP 路径下使用安全变体。

```
# python/sglang/srt/mem_cache/common.py

@triton.jit
def _get_last_loc_safe_kernel(
    req_to_token,
    req_pool_indices_tensor,
    prefix_lens_tensor,
    result_i32, # 固定为 int32 输出缓冲区，避免 Triton HIP 后端对混合宽度存储的错误编译
    num_tokens,
    req_to_token_stride,
    BLOCK_SIZE: tl.constexpr,
    PREFIX_DTYPE_IS_I64: tl.constexpr, # 编译期常量：输入 dtype 是否为 int64
):
    pid = tl.program_id(0)
    offset = tl.arange(0, BLOCK_SIZE) + pid * BLOCK_SIZE
    mask = offset < num_tokens

    if PREFIX_DTYPE_IS_I64:
        # 当输入已经是 int64 时，直接进行乘法避免额外转换
        prefix_lens = tl.load(prefix_lens_tensor + offset, mask=mask, other=0)
        req_pool_indices = tl.load(req_pool_indices_tensor + offset, mask=mask, other=0)
        token_index = req_pool_indices * req_to_token_stride + (prefix_lens - 1)
    else:
        # 当输入是 int32 时，将索引操作数显式提升为 int64，提升精度
        prefix_lens = tl.load(prefix_lens_tensor + offset, mask=mask, other=0)
        req_pool_indices = tl.load(req_pool_indices_tensor + offset, mask=mask, other=0)
        token_index = req_pool_indices.to(tl.int64) * req_to_token_stride + (
            prefix_lens.to(tl.int64) - 1
        )

    token_mask = mask & (prefix_lens > 0)
    tokens = tl.load(req_to_token + token_index, mask=token_mask, other=-1)
    # 结果存储进 int32 缓冲区，后续由调用者在 Triton 外交互完成类型提升
    tl.store(result_i32 + offset, tokens, mask=mask)


def get_last_loc_triton_safe(
    req_to_token: torch.Tensor,
    req_pool_indices_tensor: torch.Tensor,
    prefix_lens_tensor: torch.Tensor,
) -> torch.Tensor:
    """int32安全的last_loc Triton实现：使用int32输出缓冲区，由PyTorch完成最终类型提升。"""
    num_tokens = prefix_lens_tensor.shape[0]
```

```
    BLOCK_SIZE = 256
    # 分配 int32 缓冲区，避免 Triton 内发生混合宽度存储
    result_i32 = torch.empty(
        num_tokens, dtype=torch.int32, device=prefix_lens_tensor.device
    )
    grid = (triton.cdiv(num_tokens, BLOCK_SIZE),)
    _get_last_loc_safe_kernel[grid](
        req_to_token,
        req_pool_indices_tensor,
        prefix_lens_tensor,
        result_i32,
        num_tokens,
        req_to_token.stride(0),
        BLOCK_SIZE=BLOCK_SIZE,
        PREFIX_DTYPE_IS_I64=(prefix_lens_tensor.dtype == torch.int64),
    )
    # 在 Triton 外部将结果提升回目标 dtype，避免编译器 bug
    return result_i32.to(prefix_lens_tensor.dtype)
```

评论区精华

本 PR 无公开 review 评论。commit message 中提到该 PR 是从 #23146 拆分出的，应 HaiShaw 要求独立合并以加速修复。PR 由 HaiShaw 批准。

- 暂无高价值评论线程

风险与影响

- 风险：
  - 回归风险：修改集中在单一文件 common.py 中的 get_last_loc 调度函数，仅影响 HIP 平台下 attention_backend 为 triton/aiter 且使用推测解码的场景。非 HIP（CUDA）平台逻辑完全不变，回归风险低。
  - 性能影响：新引入的 get_last_loc_triton_safe 增加了一次 torch.empty 分配和一次 .to() 类型转换，开销极小，在 AMD GPU 上可忽略不计。
  - 覆盖范围：仅修复了 get_last_loc 路径下的问题，其他使用 Triton 内核的地方若存在类似混合宽度存储问题，仍需单独修复。
  - 测试覆盖：无配套测试文件，依赖集成测试（PR 提供了 GSM8K 精度测试）和手动复现。
  - 影响：用户影响：修复了 AMD GPU 用户在使用推测解码（特别是 EAGLE）且页面大小大于 1 时的崩溃问题，使该配置可正常工作。 系统影响：无，仅改变 HIP 分支下的内核选择路径，不涉及数据结构或协议变更。 团队影响：代码意图清晰，注释充分，易于维护。
- 风险标记：平台特定编译器 bug，核心路径变更，缺少测试覆盖

关联脉络

- PR #23146 [WIP] Original PR containing this fix: 本 PR 从中拆分出来，应审查者要求独立合并以加速 AMD 修复进度。