PR #23562 完整报告

sgl-project/sglang

[AMD] Enable preshuffle paged MQA and page_size=64 for NSA indexer

合并时间：2026-05-13 17:33

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23562

执行摘要

- 一句话：AMD NSA indexer 启用 preshuffle paged MQA 和 page_size=64，提升高并发性能
- 推荐动作：该 PR 值得精读，特别是 preshuffle 布局与 page_size 配合的设计思路，以及如何通过包装 aiter 和 Triton 两套实现来保持兼容性。建议重点关注 Triton fallback 的风险，并在未来 PR 中补充相应测试。

功能与动机

在 NSA indexer 中，paged MQA 计算在高并发下速度缓慢。例如，8k1kcc64 场景下每层耗时约 88 us（logits 初始化 11us + MQA kernel 77us）。主要瓶颈是过大的 block_table。当 page_size=1、max_seq_len=131k 时，block_table 形状为 (64, 131072)，约 32MB，频繁的间接加载导致 MQA kernel 性能低下。

实现拆解

1. 修改 server_args.py：移除之前 AMD 特有 page_size=1 的设置，统一设为 page_size=64，并删除对应日志。
2. 修改 memory_pool.py：将 HIP 断言约束从 page_size==1 改为 page_size%16==0，以兼容 preshuffle 布局要求。
3. 修改 index_buf_accessor.py：引入 _use_aiter 变量控制是否使用 aiter 库；新增 GetKAndS.aiter 方法，调用 cp_gather_indexer_k_quant_cache 收集 K/S 数据，并指定 preshuffle=True；同时修改 _set_k_and_s_triton 中的断言，使其支持 page_size 为 16 的倍数。
4. 修改 nsa_indexer.py：在 _get_topk_paged 和 _get_topk_ragged 中移除 page_size==1 的分支，统一使用 metadata.get_page_table_64()；将 kv_cache_fp8 的 view 逻辑简化，block_kv 直接取 page_size；将 logits 初始化从 torch.full(-inf) 改为 torch.empty()，消除冗余 kernel；MQA kernel 调用参数 Preshuffle 根据 _use_aiter 动态设置。
5. 在 _store_index_k_cache 中，利用 _use_aiter 守卫并为 HIP 启用 preshuffle 布局存储。

关键文件：

- python/sglang/srt/layers/attention/nsa/index_buf_accessor.py（模块 索引缓存；类别 source；类型 core-logic；符号 aiter）：核心变更文件，新增 aiter 后端路径 GetKAndS.aiter 和 preshuffle 布局支持，同时调整 _set_k_and_s_triton 断言以兼容

page_size 为 16 的倍数。

- python/sglang/srt/layers/attention/nsa/nsa_indexer.py（模块 索引器；类别 source；类型 core-logic）：主要逻辑文件，切换 MQA kernel 到 preshuffle 模式，统一 block_tables 获取方式，并优化 logits 初始化。
- python/sglang/srt/server_args.py（模块 配置；类别 source；类型 configuration）：移除 AMD 专用 page_size=1 设置，统一为 page_size=64，简化配置逻辑。
- python/sglang/srt/mem_cache/memory_pool.py（模块 内存池；类别 source；类型 core-logic）：调整 NSATokenToKVPool 中 page_size 的断言条件，使其支持 page_size%16==0 而非严格等于 1。

关键符号：GetKAndS.aiter, _get_topk_paged, _store_index_k_cache, _set_k_and_s_triton, _get_topk_ragged

关键源码片段

python/sglang/srt/layers/attention/nsa/index_buf_accessor.py

核心变更文件，新增 aiter 后端路径 GetKAndS.aiter 和 preshuffle 布局支持，同时调整 _set_k_and_s_triton 断言以兼容 page_size 为 16 的倍数。

```
# index_buf_accessor.py 的核心变更：新增 aiter 路径

# 在文件开头添加环境变量检查和导入
from sglang.srt.utils import get_bool_env_var, is_hip
_is_hip = is_hip()
_use_aiter = get_bool_env_var("SGLANG_USE_AITER") and _is_hip

if _use_aiter:
    from aiter.ops.cache import cp_gather_indexer_k_quant_cache

# GetKAndS 类新增 aiter 方法，用于 preshuffle 布局的 K/S 收集
class GetKAndS:
    @classmethod
    def execute(cls, *args, **kwargs):
        # 优先使用 aiter 后端（HIP + 环境变量启用时）
        if _use_aiter:
            return cls.aiter(*args, **kwargs)
        return cls.triton(*args, **kwargs) # 否则回退 Triton

    @classmethod
    def aiter(
        cls,
        pool: "NSATokenToKVPool",
        buf: torch.Tensor,
        page_indices: torch.Tensor,
        seq_len_tensor: torch.Tensor,
        seq_len_sum: int,
        max_seq_len: int,
```

```
    ):
        from sglang.srt.layers.quantization.fp8_kernel import fp8_dtype

        page_size = pool.page_size
        index_head_dim = pool.index_head_dim
        quant_block_size = pool.quant_block_size
        scale_elems = index_head_dim // quant_block_size

        # 将缓冲区 reinterpret 为 (num_pages, page_size, index_head_dim + scale_bytes) 布局
        kv_cache = buf.view(-1, page_size, index_head_dim + scale_elems * 4).view(fp8_dtype)
        dst_k = torch.empty((seq_len_sum, index_head_dim), dtype=torch.uint8, device=buf.device)
        dst_scale = torch.empty((seq_len_sum, scale_elems * 4), dtype=torch.uint8, device=buf.
        device)

        # 生成 cu_seq_lens 用于变长序列收集
        cu_seq_lens = torch.zeros(seq_len_tensor.shape[0] + 1, dtype=torch.int32, device=buf.
        device)
        torch.cumsum(seq_len_tensor.to(torch.int32), dim=0, out=cu_seq_lens[1:])

        # 调用 aiter 的 preshuffle 收集 kernel
        cp_gather_indexer_k_quant_cache(
            kv_cache,
            dst_k.view(fp8_dtype),
            dst_scale,
            page_indices.to(torch.int32),
            cu_seq_lens,
            preshuffle=True, # 参数固定为 True，表示使用 preshuffle 布局
        )
        return dst_k, dst_scale
```

同时，_set_k_and_s_triton 中的断言调整为支持任意 page_size（16 的倍数）： # 原来的断言： # if _is_hip: # assert buf_numel_per_page == 1 * (128 + 4) # else: # assert buf_numel_per_page == 64 * (128 + 4) # 修改为： assert buf_numel_per_page == page_size * (128 + 4) if _is_hip: assert page_size % 16 == 0, f"HIP preshuffle requires page_size to be a multiple of 16, got{page_size}" else: assert page_size == 64

python/sglang/srt/layers/attention/nsa/nsa_indexer.py

主要逻辑文件，切换 MQA kernel 到 preshuffle 模式，统一 block_tables 获取方式，并优化 logits 初始化。

```
 # nsa_indexer.py 中 _get_topk_paged 方法的关键变更

    def _get_topk_paged(self, forward_batch, layer_id, q_fp8, weights, metadata):
        page_size = forward_batch.token_to_kv_pool.page_size
        # 原来 HIP 断言 page_size==1，CUDA 断言 64
        # 现在统一：HIP 检查 page_size%16==0，CUDA 仍为 64
        if _is_hip:
            assert page_size % 16 == 0, \
```

```
            f"HIP preshuffle requires page_size to be a multiple of 16, got {page_size}"
        else:
            assert page_size == 64, "only support page size 64"

        # 始终使用 get_page_table_64()，不再区分 page_size==1 的特殊路径
        block_tables = metadata.get_page_table_64()
        max_seq_len = block_tables.shape[1] * page_size
        # ...

        block_kv = page_size # 原来 HIP 写死 block_kv=1，CUDA 写死 64，现在由 page_size 决定
        num_heads_kv = 1
        head_dim_with_sf = 132
        # view 逻辑统一，不再分 HIP/CUDA 分支
        kv_cache_fp8 = kv_cache_fp8.view(
            kv_cache_fp8.shape[0], block_kv, num_heads_kv, head_dim_with_sf
        )

        if _is_hip:
            from aiter.ops.triton.pa_mqa_logits import deepgemm_fp8_paged_mqa_logits
            batch_size, next_n, heads, _ = q_fp8.shape
            # 使用 torch.empty 替代 torch.full(-inf)，消除冗余 kernel
            logits = torch.empty(
                (batch_size * next_n, max_seq_len),
                device=q_fp8.device,
                dtype=torch.float32,
            )
            deepgemm_fp8_paged_mqa_logits(
                q_fp8, kv_cache_fp8, weights, logits,
                seqlens_32, block_tables, max_seq_len,
                Preshuffle=_use_aiter, # 根据环境变量动态设置
                KVBlockSize=block_kv,
            )
```

同时，_store_index_k_cache 方法中也根据 _use_aiter 启用了 preshuffle 布局存储。

评论区精华

- gemini-code-assist 指出当 SGLANG_USE_AITER 未设置时，fallback 到 Triton kernel 会因布局不匹配而导致计算错误，需考虑强制使用 aiter 或更新 Triton kernel。
- Jacob0226 建议将 _get_topk_paged 中 Preshuffle 参数从 True 改为 _use_aiter，以与存储路径的守卫一致；作者 1am9trash 确认已修复。
- Triton fallback 布局不匹配 (design): PR 合并时该问题未被完全解决，但实际中 HIP 环境通常默认启用 aiter，且后续提交 'Only use preshuffle with _use_aiter' 已确保 Preshuffle 参数仅在 _use_aiter=True 时设为 True，因此当 _use_aiter=False 时 Preshuffle=False，布局匹配无问题（但存储路径仍可能不匹配，需注意）。
- Preshuffle 参数应使用 _use_aiter 变量 (correctness): 作者 1am9trash 回复 'Addressed. Thanks.'，并在后续提交中修改为 Preshuffle=_use_aiter。问题已解决。

风险与影响

- 风险：
  1. 依赖外部 aiter 库版本：PR 依赖 aiter PR#2879 的 cp_gather_indexer_k_quant_cache 和 preshuffle 支持，若未正确合并或版本不匹配，会导致运行时错误。
  2. Triton fallback 布局兼容性：当 SGLANG_USE_AITER 处于关闭状态时，Triton 路径未更新 preshuffle 布局，会导致 MQA kernel 计算错误。虽然 _use_aiter 在 HIP 下通常为 True，但用户若显式关闭则存在风险。
  3. 仅影响 AMD HIP 后端：CUDA 路径不受影响，但 AMD 场景下的内存使用和调度行为因 page_size 增大而改变，需关注极端序列长度下的分页效率。
  4. 测试覆盖不足：本次改动未包含新增测试，依赖现有 AMD CI 覆盖，可能遗漏边界条件。 - 影响：影响范围限定于 AMD GPU 上使用 DeepSeek DSA 模型且启用 NSA indexer 的场景。性能提升显著，高并发下吞吐量提升约 16%，单层延迟从 88us 降至 19us。对 CUDA 后端无影响，对未使用 NSA indexer 的模型无影响。用户需确保 aiter 库版本满足依赖要求，且推荐保持 SGLANG_USE_AITER 开启状态。 - 风险标记：依赖 aiter 版本，Triton fallback 布局兼容性，仅限 AMD HIP

关联脉络

- 暂无明显关联 PR