PR #24243 完整报告

sgl-project/sglang

Reserve slot 0 as padding in all req pools

合并时间：2026-05-02 07:41

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/24243

执行摘要

- 一句话：在所有请求池中预留 slot 0 作为填充
- 推荐动作：推荐精读，尤其是关注 HybridReqToTokenPool 中 Mamba 映射大小的对齐问题。本 PR 展示了如何通过一个简单统一的 padding 约定，为未来的复杂功能（DSv4）提前消除数据竞争风险，是典型的前置兼容性变更。

功能与动机

为即将到来的 DeepSeek V4 集成（dsv4-rebase）做准备。V4 压缩器 / 状态池路径读取 req_to_token[req_pool_indices, ...] 并通过 full_to_swa -> state_loc 转换结果。CUDA 图填充批处理使 req_pool_indices[raw_bs:] = 0（零初始化缓冲区），因此只有让 req_pool_idx = 0 永久未被占用，才能防止虚拟写入破坏真实状态。在主干中先采用 padding 行，使契约与 KV 池 padding 保持对称，并让 V4 分支能够删除本地的 -1 变通方案。

实现拆解

1. 修改 ReqToTokenPool（memory_pool.py）：将 req_to_token 张量的第一个维度从 size 增加到 size + 1，free_slots 初始化为 range(1, size + 1)，clear() 方法相应调整。
2. 修改 DecodeReqToTokenPool（decode.py）：将 req_to_token 张量形状从 (size + pre_alloc_size, ...) 改为 (size + pre_alloc_size + 1, ...)，free_slots 从 1 开始，clear() 同步调整。
3. 修改 HybridReqToTokenPool 的 Mamba 映射（memory_pool.py）：req_index_to_mamba_index_mapping 和 req_index_to_mamba_ping_pong_track_buffer_mapping 的大小由 size 改为 size + 1，以匹配父类增加的 padding 行。
4. 修改 HybridMambaDecodeReqToTokenPool.clear()（decode.py）：free_slots 重置为从 1 开始，保持与父类一致。
5. 所有外部接口（如 alloc()、free()、available_size()）对外行为不变，仅内部存储和索引分配逻辑调整。

关键文件：

- python/sglang/srt/mem_cache/memory_pool.py（模块 内存池；类别 source；类型 core-logic；符号 ReqToTokenPool）：核心文件：修改了 ReqToTokenPool 和 HybridReqToTokenPool 的缓冲区分配和空闲槽管理，是所有请求池 padding 的基础。

- python/sglang/srt/disaggregation/decode.py（模块 解耦模块；类别 source；类型 core-logic；符号 DecodeReqToTokenPool, HybridMambaDecodeReqToTokenPool）：次要文件：同步修改了 DecodeReqToTokenPool 及其继承类 HybridMambaDecodeReqToTokenPool 的缓冲区大小和空闲槽管理，保持与 ReqToTokenPool 一致。

关键符号：ReqToTokenPool.init, ReqToTokenPool.clear, DecodeReqToTokenPool.init, DecodeReqToTokenPool.clear, HybridReqToTokenPool._init_mamba_pool, HybridMambaDecodeReqToTokenPool.clear

关键源码片段

python/sglang/srt/mem_cache/memory_pool.py

核心文件：修改了 ReqToTokenPool 和 HybridReqToTokenPool 的缓冲区分配和空闲槽管理，是所有请求池 padding 的基础。

```
# python/sglang/srt/mem_cache/memory_pool.py classReqToTokenPool: """A memory pool that maps a request to its token locations."""  def__init__(  self,  size: int,  max_context_len: int,  device: str,  enable_memory_saver: bool,  ) :  # 省略内存保存适配器初始化 ...  self.size = size  self.max_context_len = max_context_len  self.device = device  with memory_saver_adapter.region(GPU_MEMORY_TYPE_KV_CACHE):  # +1 row for padding slot 0 (mirrors KV pool): cuda-graph padded  # batches default req_pool_indices to 0, so routing dummies through  # unowned slot 0 keeps req_to_token[0, :] zero and downstream writes  # harmless. self.req_to_token = torch.zeros(  (size + 1, max_context_len), dtype=torch.int32, device=device  )  # free_slots 从 1 开始，保留索引 0 作为 padding，永不分配  self.free_slots = list(range(1, size + 1))  # alloc(), free(), available_size() 接口保持不变  defclear(self):  self.free_slots = list(range(1, self.size + 1)) # python/sglang/srt/mem_cache/memory_pool.py # HybridReqToTokenPool._init_mamba_pool 中的关键修改 classHybridReqToTokenPool(ReqToTokenPool):  def_init_mamba_pool(self, ...): # ... 省略其他初始化代码  # 注意：映射大小基于 size + 1（即请求池大小），而非 mamba_size  # 避免越界访问：这些映射通过 req_pool_idx（从请求池分配）索引  self.req_index_to_mamba_index_mapping: torch.Tensor = torch.zeros(  size + 1, dtype=torch.int32, device=self.device  )  if enable_mamba_extra_buffer:  self.req_index_to_mamba_ping_pong_track_buffer_mapping: torch.Tensor = torch.zeros(  size + 1, dtype=torch.int32, device=self.device  )
```

python/sglang/srt/disaggregation/decode.py

次要文件：同步修改了 DecodeReqToTokenPool 及其继承类 HybridMambaDecodeReqToTokenPool 的缓冲区大小和空闲槽管理，保持与 ReqToTokenPool 一致。

```
# python/sglang/srt/disaggregation/decode.py
```

```
class DecodeReqToTokenPool:
    """
    The difference of DecodeReqToTokenPool and ReqToTokenPool is that
    DecodeReqToTokenPool subscribes memory for pre-allocated requests.
    """

    def __init__(
        self,
        size: int,
        max_context_len: int,
        device: str,
        enable_memory_saver: bool,
        pre_alloc_size: int,
    ):
        # ... 省略保存适配器初始化代码
        self.size = size
        self.pre_alloc_size = pre_alloc_size
        with memory_saver_adapter.region(tag=GPU_MEMORY_TYPE_KV_CACHE):
            # +1 row 0 padding; mirrors ReqToTokenPool / KV pool padding slot 0.
            self.req_to_token = torch.zeros(
                (size + pre_alloc_size + 1, max_context_len),
                dtype=torch.int32,
                device=device,
            )
        # free_slots 从 1 开始，保留索引 0 作为 padding
        self.free_slots = list(range(1, size + pre_alloc_size + 1))

    def clear(self):
        self.free_slots = list(range(1, self.size + self.pre_alloc_size + 1))


class HybridMambaDecodeReqToTokenPool(HybridReqToTokenPool):

    def clear(self):
        # 调用父类 clear 后会重置 free_slots，需要同步调整 padding 起始
        self.free_slots = list(range(1, self.size + self.pre_alloc_size + 1))
        self.mamba_pool.clear()
```

评论区精华

Review 中主要讨论了一个关键正确性问题：在 HybridReqToTokenPool._init_mamba_pool() 中，req_index_to_mamba_index_mapping 和 req_index_to_mamba_ping_pong_track_buffer_mapping 的大小基于 mamba_size（即 Mamba 状态池的大小），但这些映射是通过 req_pool_idx（来自请求池的索引）来索引的。如果 mamba_size 小于请求池的大小，可能导致越界访问。建议将这些映射的大小改为基于请求池的大小，而不是 Mamba 池的大小。作者采纳了建议，在后续提交中将映射大小改为基于 size + 1（即请求池大小加 padding），修复了潜在越界问题。

- HybridReqToTokenPool 中 Mamba 映射的大小应基于请求池而非 Mamba 池 (correctness): 作者采纳建议，在后续提交中将映射大小改为 size + 1（请求池大小加 padding），修复了潜在的越界问题。

风险与影响

- 风险：低风险。本变更仅修改内部缓冲区分配和空闲槽管理，对外接口（alloc、free、available_size）行为不变。主要风险在于：
  - 如果外部代码直接访问 req_to_token 张量的第 0 行并假设其为有效数据，可能被 padding 行影响，但检查所有使用点后未发现此类情况。
  - HybridReqToTokenPool 中 Mamba 映射的大小调整如果未同步，可能导致越界，但已通过修复确保同步。
  - 无测试文件变更，虽然风险较低，但缺少回归测试覆盖。
  - 影响：影响范围：所有使用请求池的路径（常规推理、Mamba 模型、解码池、DP 注意力等）。影响程度：低。对外语义不变，仅内部多占用一行整数张量（每个池约 4 字节 * max_context_len 的内存开销，可忽略）。但该变更为未来 DeepSeek V4 集成提供了关键前提条件，避免在 CUDA 图填充场景下的数据损坏。团队影响：开发者需要了解 slot 0 为保留 padding，不应复用。
- 风险标记：缺少测试覆盖，核心路径变更

关联脉络

- PR #23635 PR#23635（已修复类似问题）：PR body 提及该问题也在 PR#23635 中修复，由 @foraxe 贡献。