PR #24857 完整报告

sgl-project/sglang

Optimize SWA memory preallocation for disaggregated decode

合并时间：2026-05-13 09:09

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/24857

执行摘要

- 一句话：优化分解式解码 SWA KV 缓存预分配策略
- 推荐动作：建议合并。值得关注的设计决策：将 SWA 预分配与 full 预分配解耦、CPU copy 的稀疏 mask 处理。后续可考虑将同一优化扩展到 prefill 节点。

功能与动机

在 PD 分解式服务中，decode 节点需为每个请求预分配 KV 缓存。对混合 SWA 模型，原方案为 full attention 和 SWA 分配相同大小的缓存，导致显存浪费。Issue #24036 指出需要单独计算 decode 端 SWA 预分配大小，仅保留滑动窗口尾部所需空间，从而降低显存占用、提升并发能力。

实现拆解

1. SWA tail 预分配判断：在 decode.py 的 DecodeManager 中新增 _uses_swa_tail_prealloc() 方法，检查 token_to_kv_pool 是否为 SWAKVPool/DeepSeekV4TokenToKVPool、page_size>1 且 allocator 支持 alloc_extend_swa_tail。
2. 尾部长度计算：实现 _swa_tail_len() 和 _swa_retractable_len()，基于 window_size 和 page_size 对齐确定请求在滑动窗口内需保留的 token 数。
3. 双预算追踪：将 _allocatable_tokens() 重构为 _allocatable_token_budgets()，返回 (full_allocatable_tokens, swa_allocatable_tokens) 二元组，独立判断 full pool 和 SWA pool 的可用空间。在 pop_preallocated()、_pre_alloc() 等关键方法中同步使用新预算。
4. CPU copy 筛选：在 swa_memory_pool.py 的 SWATokenToKVPoolAllocator 中新增 _filter_swa_cpu_copy() 方法，配合 swa_mask 仅复制实际映射的 SWA 索引，避免无效的 out-of-window token。修改 get_cpu_copy() 和 load_cpu_copy() 以支持稀疏映射。
5. 测试适配：更新 test_decode_radix_lock_ref.py 测试用例，将 mock 从 _allocatable_tokens 改为 _allocatable_token_budgets，并添加 token_to_kv_pool mock 以通过 _uses_swa_tail_prealloc() 检查。

关键文件：

- python/sglang/srt/disaggregation/decode.py（模块 解码管理；类别 source；类型 core-logic；符号 _uses_swa_tail_prealloc, _swa_tail_len, _swa_retractable_len, _prealloc_kv_lens）：核心变更文件，实现 SWA tail 预分配判断、长度计算、双预算重构

等主要逻辑

- python/sglang/srt/mem_cache/swa_memory_pool.py（模块 SWA 内存池；类别 source；类型 core-logic；符号 _filter_swa_cpu_copy, alloc_extend_swa_tail）：新增 alloc_extend_swa_tail 方法和 CPU copy 筛选逻辑，支持稀疏 SWA 索引映射
- test/registered/unit/mem_cache/test_decode_radix_lock_ref.py（模块 单元测试；类别 test；类型 test-coverage）：测试适配，将 mock 更新为新预算接口，添加 token_to_kv_pool mock

关键符号：_uses_swa_tail_prealloc, _swa_tail_len, _swa_retractable_len, _prealloc_kv_lens, _prealloc_required_tokens, _allocatable_token_budgets, _need_space_for_single_req, _active_req_count, _filter_swa_cpu_copy, alloc_extend_swa_tail

关键源码片段

python/sglang/srt/disaggregation/decode.py

核心变更文件，实现 SWA tail 预分配判断、长度计算、双预算重构等主要逻辑

```
def _uses_swa_tail_prealloc(self) -> bool:
    # 检查当前 allocator 是否支持 SWA tail 预分配:
    # 必须是 SWAKVPool 或 DeepSeekV4TokenToKVPool，且 page size 大于 1，
    # 同时 allocator 暴露了 alloc_extend_swa_tail 方法。
    return (
        isinstance(self.token_to_kv_pool, (SWAKVPool, DeepSeekV4TokenToKVPool))
        and self.token_to_kv_pool_allocator.page_size > 1
        and hasattr(self.token_to_kv_pool_allocator, "alloc_extend_swa_tail")
    )

def _swa_tail_len(self, seq_len: int) -> int:
    # 计算指定序列长度对应的 SWA 尾部长度（按 page 对齐）。
    if not self._uses_swa_tail_prealloc() or seq_len <= 0:
        return max(seq_len, 0)
    window_size = self.scheduler.sliding_window_size
    if window_size is None or window_size <= 0:
        return seq_len
    page_size = self.token_to_kv_pool_allocator.page_size
    window_start = max(0, seq_len - window_size)
    window_start = (window_start // page_size) * page_size
    return seq_len - window_start

def _prealloc_kv_lens(self, req: Req) -> Tuple[int, int]:
    # 返回 (full_len, swa_len) 二元组，表示待预分配的 KV 长度。
    # full_len 是基于请求本身的计算长度，swa_len 是窗口尾部长度。
    allocated_kv_len = len(req.origin_input_ids) + max(len(req.output_ids) - 1, 0)
    if self._uses_swa_tail_prealloc():
        return allocated_kv_len, self._swa_tail_len(allocated_kv_len)
    return allocated_kv_len, allocated_kv_len
```

python/sglang/srt/mem_cache/swa_memory_pool.py

新增 alloc_extend_swa_tail 方法和 CPU copy 筛选逻辑，支持稀疏 SWA 索引映射

```
  def _filter_swa_cpu_copy(self, swa_kv_cpu, row_mask: torch.Tensor):
      # 根据 row_mask 从 CPU 拷贝的 SWA KV 数据中筛选出有效 chunk。
      if swa_kv_cpu is None:
          return None
      if row_mask is None or bool(torch.all(row_mask).item()):
          return swa_kv_cpu
      chunk_size = getattr(
          self.swa_kv_pool, "cpu_offloading_chunk_size", len(row_mask)
      )
      filtered = []
      for layer_chunks in swa_kv_cpu:
          if len(layer_chunks) == 0:
              filtered.append([])
              continue
          # 拼接该层的所有 chunk，按 row_mask 筛选，再切回 chunk 列表。
          k_cpu = torch.cat([chunk[0] for chunk in layer_chunks], dim=0)
          v_cpu = torch.cat([chunk[1] for chunk in layer_chunks], dim=0)
          k_cpu = k_cpu[row_mask]
          v_cpu = v_cpu[row_mask]
          filtered_layer = []
          for i in range(0, len(k_cpu), chunk_size):
              filtered_layer.append(
                  [k_cpu[i : i + chunk_size], v_cpu[i : i + chunk_size]]
              )
          filtered.append(filtered_layer)
      return filtered

  def get_cpu_copy(self, indices, mamba_indices=None):
      # 改写：返回额外 swa_mask 字段，标记哪些 full 索引在 SWA 池中有映射。
      full_kv_cpu = self.full_kv_pool.get_cpu_copy(indices)
      swa_mask = None
      if self.full_to_swa_index_mapping is not None:
          swa_indices = self.full_to_swa_index_mapping[indices]
          # Slot 0 是保留 dummy，tail-only 分配会将窗口外 full KV 索引设为 0，
          # 只复制有映射（>0）的索引。
          swa_mask = swa_indices > 0
          if torch.any(swa_mask):
              swa_kv_cpu = self.swa_kv_pool.get_cpu_copy(swa_indices[swa_mask])
              swa_mask = swa_mask.cpu()
          else:
              swa_kv_cpu = None
      else:
          swa_kv_cpu = None
      return {"full": full_kv_cpu, "swa": swa_kv_cpu, "swa_mask": swa_mask}
```

评论区精华

无公开 review 评论。从 commit 历史可见多轮反馈已被逐一处理：修复预算计算中缺少 evictable_size() 调用、重命名 allocatable_tokens 为 full_allocatable_tokens、缓存 _uses_swa_tail_prealloc() 结果、添加注释说明 prefix_len > 0 时 SWA 回退逻辑等。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 核心路径变更：修改了 pop_preallocated() 等热路径，可能引入调度逻辑错误，导致请求无法正确预分配。
  2. 双池预算逻辑复杂度：独立的 full/SWA 预算追踪增加了计算开销，高并发下需关注性能。
  3. SWA 映射稀疏性敏感性：swa_mask 要求 get_cpu_copy 与 load_cpu_copy 严格对称，mask 处理有误可能导致 KV 数据丢失或错位。
  4. 兼容性：非 SWA 模型或旧版 allocator 的 fallback 路径测试覆盖不足。 - 影响：直接影响 PD 分解式部署中 SWA 模型（如 DeepSeek-V4）的 decode 节点显存占用和吞吐，benchmark 显示 1%-4% 收益。对非 SWA 模型无影响（走 fallback）。代码向后兼容，无需用户手动配置。团队应在实际负载中监控内存压力变化。 - 风险标记：核心路径变更，双池预算逻辑复杂度，SWA 映射稀疏性敏感性

关联脉络

- PR #24036 Fix disagg decode SWA prealloc sizing: 当前 PR 将此优化从 deepseek_v4_dev 分支迁移到 main，并进一步重构了预算计算