PR #29106 完整报告

sgl-project/sglang

Fix DeepSeek V4 PP HiCache SWA allocation and layer mapping

合并时间：2026-06-27 22:19

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/29106

执行摘要

- 一句话：修复 DeepSeek V4 PP HiCache SWA 分配与层映射
- 推荐动作：面向阅读者的建议：该 PR 值得精读，尤其是关注流水线并行与缓存模块集成的开发者。其设计关键点在于：通过 PP 阶段本地层数分配资源，并通过间接方法封装索引转换，避免了全局层映射带来的状态不一致。讨论中关于近似算法精度的排查思路也值得借鉴。

功能与动机

DeepSeek V4 + HiCache 在流水线并行下会失败，因为 SWA KV pool 按全局模型层数分配，而每个 PP 阶段只拥有局部层切片。在 PP + HiCache 运行时，这会导致后续 PP 阶段的 HiCache 状态构建无效，并在服务初始化时失败，报错 ValueError: deepseek_v4_c4_state state_pools must not contain None。

实现拆解

1. SWA KV 池按 PP 阶段层数分配：在 deepseek_v4_memory_pool.py 的 __init__ 中，计算 stage_layer_num = len(stage_ratios)，替换原来的全局 layer_num，作为 SWA KV 池的 layer_num 参数。
2. HiCache 堆栈构建使用局部层映射：在 hybrid_pool_assembler.py 的 build_deepseek_v4_hicache_stack 中，移除 TODO 注释，添加 SWA 池缓冲区数量与传输层数的校验；遍历层映射时改用 local_layer_id 和 global_layer_id 分别构建 C4 状态本地列表和全局列表，SWA 层映射也基于局部数量构建。
3. 新增局部索引方法：在 deepseek_v4_memory_pool.py 中新增 get_swa_raw_buffer(layer_id) 方法，通过 _swa_local_layer_id 将全局层 ID 转换为本地索引，返回对应的原始缓冲区。
4. 模型前向路径适配：在 deepseek_v4.py 的两个前向准备函数中，将直接访问 swa_kv_pool.kv_buffer[self.layer_id] 替换为调用 get_swa_raw_buffer(self.layer_id)。
5. 测试重构与新增：在基础测试类 test_unified_radix_cache_kl_dsv4.py 中抽取 _server_args 方法，支持通过类属性 pp_size/tp_size 动态设置服务器参数；新增 test_unified_radix_cache_kl_dsv4_pp.py 定义 TestUnifiedDeepSeekV4FlashHiCachePP4TP2 类，继承基础测试并覆盖 pp_size=4, tp_size=2，注册到 CI 的 extra-b 阶段（8-gpu-h200）。

关键文件：

- python/sglang/srt/mem_cache/hybrid_cache/hybrid_pool_assembler.py（模块 缓存层；类别 source；类型 core-logic；符号 build_deepseek_v4_hicache_stack）：HiCache 堆栈构建的核心函数，修正了 SWA 和 C4 状态层映射从全局索引改为局部索引，并添加了池大小校验。
- python/sglang/srt/mem_cache/deepseek_v4_memory_pool.py（模块 缓存层；类别 source；类型 core-logic；符号 get_swa_raw_buffer）：SWA KV 池分配参数改为使用 PP 阶段层数；新增 get_swa_raw_buffer 方法封装局部索引，是修复的关键数据结构变更。
- python/sglang/srt/models/deepseek_v4.py（模块 模型；类别 source；类型 data-contract；符号 _forward_prepare, _forward_prepare_multi_stream_hip）：模型前向路径中直接访问 kv_buffer 的位置替换为 get_swa_raw_buffer，确保 PP 下索引正确。
- test/registered/radix_cache/unified_radix_tree/test_unified_radix_cache_kl_dsv4.py（模块 测试；类别 test；类型 test-coverage；符号 _server_args）：重构基础测试类，提取 _server_args 方法以支持 pp_size/tp_size 参数化，便于被 PP 测试类继承。
- test/registered/radix_cache/unified_radix_tree/test_unified_radix_cache_kl_dsv4_pp.py（模块 测试；类别 test；类型 test-coverage；符号 TestUnifiedDeepSeekV4FlashHiCachePP4TP2）：新增 PP4TP2 集成测试，验证 PP+HiCache 组合的正确性，填补 CI 覆盖空白。

关键符号：build_deepseek_v4_hicache_stack, get_swa_raw_buffer, _forward_prepare, _forward_prepare_multi_stream_hip, _server_args, test_multiturn_logprobs_match

关键源码片段

python/sglang/srt/mem_cache/hybrid_cache/hybrid_pool_assembler.py

HiCache 堆栈构建的核心函数，修正了 SWA 和 C4 状态层映射从全局索引改为局部索引，并添加了池大小校验。

```
def build_deepseek_v4_hicache_stack(
    ...,
    kvcache: Any,
    page_size: int,
    ...
) -> tuple[HostPoolGroup, HybridCacheController]:
    transfer_layer_num = kvcache.end_layer - kvcache.start_layer
    full_layer_mapping = {layer_id: layer_id for layer_id in range(transfer_layer_num)}

    # 验证 SWA KV 池的缓冲区数量必须等于本 PP 阶段的层数
    if len(kvcache.swa_kv_pool.kv_buffer) != transfer_layer_num:
        raise ValueError(
            "DeepSeek V4 SWA KV pool must be PP-stage-local: "
            f"got {len(kvcache.swa_kv_pool.kv_buffer)} buffers for "
            f"{transfer_layer_num} local layers"
        )
    swa_layer_mapping = {layer_id: layer_id for layer_id in range(transfer_layer_num)}

    c4_layer_mapping = {}
```

```
    c128_layer_mapping = {}
    c4_state_local_layers = []
    c4_state_global_layers = []
    for local_layer_id, layer_item in enumerate(
        kvcache.layer_mapping[kvcache.start_layer : kvcache.end_layer]
    ):
        global_layer_id = kvcache.start_layer + local_layer_id
        if layer_item.compress_ratio == 4:
            c4_layer_mapping[local_layer_id] = layer_item.compress_layer_id
            c4_state_local_layers.append(local_layer_id)
            c4_state_global_layers.append(global_layer_id)
        elif layer_item.compress_ratio == 128:
            c128_layer_mapping[local_layer_id] = layer_item.compress_layer_id

    c4_state_mapping = {
        layer_id: local_id for local_id, layer_id in enumerate(c4_state_local_layers)
    }
    # ... 后续构建 host pool 和 entries
```

python/sglang/srt/mem_cache/deepseek_v4_memory_pool.py

SWA KV 池分配参数改为使用 PP 阶段层数；新增 get_swa_raw_buffer 方法封装局部索引，是修复的关键数据结构变更。

```
  def __init__(self, ...):
    # ...
    stage_layer_num = len(stage_ratios) # PP 阶段的局部层数
    c4_layer_num = sum(1 for r in stage_ratios if r == 4)
    c128_layer_num = sum(1 for r in stage_ratios if r == 128)
    # ...
    # 非 unified_kv 路径：SWA KV 池使用 stage_layer_num 而非全局 layer_num
    self.swa_kv_pool = self._make_kv_pool(
        size=swa_size,
        page_size=swa_page_size,
        dtype=dtype,
        layer_num=stage_layer_num, # 修正点
        device=device,
        enable_memory_saver=enable_memory_saver,
        global_page_size=swa_page_size,
    )
    # ...

  def get_swa_raw_buffer(self, layer_id: int) -> torch.Tensor:
    """根据全局层 ID 返回本阶段 SWA 池的原始缓冲区。"""
    return self.swa_kv_pool.kv_buffer[self._swa_local_layer_id(layer_id)]
```

test/registered/radix_cache/unified_radix_tree/test_unified_radix_cache_kl_dsv4.py

重构基础测试类，提取 _server_args 方法以支持 pp_size/tp_size 参数化，便于被 PP 测试类继承。

```
@classmethod
def _server_args(cls):
    args = [
        "--trust-remote-code",
        "--tp-size",
        str(cls.tp_size),
    ]
    if cls.pp_size != 1: # 当 PP 开启时添加 --pp-size 参数
        args += ["--pp-size", str(cls.pp_size)]
    args += [
        # ... 其他固定参数
        "--attention-backend",
        "compressed",
        "--page-size",
        "256",
        # ...
    ]
    return args
```

评论区精华

- CI 测试配置调整：初始测试注册在 base-c 阶段（8-gpu-h20），但 hzh0425 认为应放在 4-gpu-h100 和 extra-b 阶段。后续在 4-gpu-h100 上超时，最终移至 8-gpu-h200 的 extra-b 阶段，估计时间调整为 900 秒。
- 精度稳定性调查：作者 1e4ves 发现 KL 散度不稳定性源于 DSV4 C4 索引器定制的 top-k 实现（v1/v2）是近似算法，而非精确 torch.topk。通过设置环境变量 SGLANG_TOPK_TRANSFORM_512_TORCH=1 回退到 PyTorch 路径后，PP2TP2 预填充缓存命中测试连续 500 次未出现 KL 违约。
- 代码审查：hzh0425 请求 ShangmingCai 复查 deepseek_v4.py 中的变更；ShangmingCai 指出了 lint 问题，在后续提交中修复。
  - CI 测试配置与超时处理 (testing): 测试最终注册在 extra-b 阶段，8-gpu-h200 运行器，est_time=900。
  - KL 精度不稳定性调查 (correctness): 此问题为已知的近似算法限制，不影响 PP+HiCache 修复的正确性；用户可通过环境变量选择精确路径。
  - 代码审查：deepseek_v4.py 变更 (design): 变更通过，无进一步问题。

风险与影响

- 风险：
  - SWA 池分配变更：stage_layer_num 在非 PP 时等于全局层数，行为兼容。但有潜在风险：若其他代码路径直接引用 swa_kv_pool.kv_buffer 而不使用 get_swa_raw_buffer，可能在 PP 下越界。需确保所有 SWA 访问都经过转换。

    - 精度风险：KL 测试通过，但作者指出的 top-k 近似问题可能在其他场景下引发精度退化，建议用户关注环境变量后备机制。
    - CI 覆盖：仅测试 PP4TP2 一种配置，未覆盖其他并行度组合，但基础测试已涵盖 TP4 非 PP 场景。
    - 内存占用：SWA 池分配大小减小，不会出现内存浪费。
- 影响：
    - 用户：启用 PP 和 HiCache 的 DeepSeek V4 场景不再崩溃，可以在流水线并行下正常使用分层缓存。
    - 系统：SWA 池的正确分配修复了 HiCache 状态构建逻辑，PP 环境下的服务初始化顺利通过。
    - 团队：新增的集成测试填补了 PP+HiCache 的 CI 空白，降低后续开发中的回归风险。
    - 风险标记：SWA 池分配变更影响 PP/ 非 PP 路径，近似 top-k 算法可能导致精度偏差，CI 测试覆盖仅限 PP4TP2 一种配置

关联脉络

- PR #28614 [HiCache] remove large host mem constraint: 同为 HiCache 相关修改，涉及内存池约束调整，与本 PR 共同完善 HiCache 功能。