

PR #38192 完整报告

sgl-project/sglang

[AMD][DSV4] Reland unified-KV pool sizing and SWA ring accounting, fully gated

合并时间: 2026-09-08 04:13

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/38192>

执行摘要

- 一句话: 修复 DSV4 统一 KV 池内存计量, 将 SWA 和 C4 状态从 token 级预算改为 per-request ring 计数, 释放 83.5% 容量。
- 推荐动作: 该 PR 是对一个已知问题的高质量 reland, 核心逻辑清晰、门控完整、测试充分。建议精读以下部分: (1) pool_configurator.py 中 `_get_bytes_per_full_token` 的 unified 分支和三个固定偏置方法, 理解 token 级预算 vs 字节预算的转换; (2) swa.py 中 per-request ring 的 `available_size/swa_available_size` 语义变化, 这是调度器正确性的关键; (3) schedule_policy.py 中 ring 模式下的计费逻辑变更。对于关注 DSV4 部署的工程师, 该 PR 的容量提升是直接可感知的性能改善。

功能与动机

在 unified-KV 内核路径中, SWA 与 C4 压缩状态按 per-request ring (ring_size 个 slot 固定预分配给每个请求槽位) 而非 token-scaled 池分配。但 `DSV4PoolConfigurator` 的 `_get_bytes_per_full_token` 仍按 `swa_ratio * kv_bytes` 和 `swa_ratio * c4_state_ratio * c4_state_bytes` 将二者计入每 token 字节预算, 导致 full-token 池预留了远超实际使用的内存, 可用 token 容量大幅缩水。PR#30315 修复了该问题但在 PR#38163 中被回滚, 原因是存在三处改动未被统一 -KV 开关门控, 非统一 (fp8) 路径也被影响。本 PR 的动机是 reland 该修复并将所有共享路径变更完全 gate 在 `_unified_kv` 条件分支后, 确保非统一路径不受影响。

实现拆解

1. 池容量计算重构 (pool_configurator.py): 在 `DSV4PoolConfigurator.__init__` 中通过 `is_unified_kv_triton()` 解析 `self._unified` 开关, 新增 `_swa_ring_size`、`_spec_infl` 属性; `_get_bytes_per_full_token()` 在 unified 下将 SWA 和 C4 state 相关项置为 0.0 (因为它们不再是 per-token 缩放), 同时 `kv_bytes` 改用 `attn_head_dim * 2` (bf16 latent); `_compute_dsv4_sizes()` 在 unified 下将 `c4_state_pool_size` 置为 0 (延迟到并发已知时才按 ring 精确计算); 新增 `_unified_c4_state_pool_size()`、`_fixed_c4_state_bytes()`、`_fixed_swa_bytes()` 三个方法, 分别在 `calculate_pool_sizes` 中作为固定偏置从字节预算中扣除, 并在 `finalize_with_max_running_requests` 中将精确的 `c4_state_pool_size` 写入 config。
2. SWA 分配器适配 (swa.py): `SWATokenToKVPoolAllocator` 新增 `req_to_token_pool` 参数, 通过 `kvcache.swa_req_ring_size` 判断是否启用 per-request ring 模式; 启用后 `available_size()` 返回 `full_attn_allocator.available_size()` (SWA ring 已预分配、不再约

束 token 增长) , swa_available_size() 返回 free_req_slots * swa_ring_cost_tokens, alloc_extend/alloc_decode/new_pages_available 统一走 full-only 早期返回路径; 新增顶层 is_swa_req_ring() 辅助函数供外部查询。

3. C4 状态 per-request 生命周期 (deepseek_v4_memory_pool.py) : 在 DeepSeekV4TokenToKVPool.__init__ 中将 _unified_kv 解析提前到 c4_state_pool_size 计算之前; unified 路径下 c4_state_pool_size = num_req_slots * c4_ring_size; 新增 clear_c4_req_states(req_pool_indices) 方法, 按 req_pool_idx * ring_size + arange(ring_size) 定位并清零指定请求槽的 C4 状态; kv_cache_configurator.py 中为目标 worker 注册 req_to_token_pool.register_on_alloc_rows(token_to_kv_pool.clear_c4_req_states), 确保新分配的请求槽自动清零。
4. 请求池钩子机制 (memory_pool.py, decode.py) : ReqToTokenPool 和 DecodeReqToTokenPool 新增 _on_alloc_rows 回调和 register_on_alloc_rows() 方法, 在 alloc_rows/alloc 中仅对新分配的槽位触发回调, 复用已有槽位的 chunked continuation 不触发, 避免误清中间状态。
5. 调度策略适配 (schedule_policy.py) : rem_swa_tokens 属性在 ring 模式下直接返回 allocator.swa_available_size() (已不含 evictable) , _swa_budget_for_req 返回固定的 swa_ring_cost_tokens; _update_prefill_budget 中 chunked continuation 不再重复计费 ring slot; add_one_req 中 ring 模式用 > 替代 >= 做容量判断。
6. 测试覆盖: 新增 test_dsv4_c4_state_lifecycle.py (4 个 mock 测试验证 c4 state pool 尺寸、clear 选择性、alloc hook 触发条件) ; 扩展 test_pool_configurator.py (新增 unified/non-unified 分支对比、token-cap 单调性回归测试、固定偏置字节预算测试) ; 扩展 test_c4_v2.py (验证 unified request-ring plan 地址计算) 。

关键文件:

- python/sglang/srt/model_executor/pool_configurator.py (模块 内存池配置器; 类别 source; 类型 data-contract; 符号 _get_c128_state_fixed_bytes_for_token_capacity, _unified_c4_state_pool_size, _fixed_c4_state_bytes, _resolve_max_running_requests_per_worker) : 核心变更: 重构 DSV4 池容量计算, 将 SWA/C4 state 从 token 级预算改为 per-request ring 固定偏置, 新增三个固定字节计算方法, unified 分支完整门控
- python/sglang/srt/mem_cache/allocator/swa.py (模块 SWA 分配器; 类别 source; 类型 core-logic; 符号 swa_req_ring, swa_ring_cost_tokens, is_swa_req_ring) : SWA 分配器核心适配: 启用 per-request ring 模式, 重写 available_size/swa_available_size 语义, alloc_extend/alloc_decode 走 full-only 路径
- test/registered/unit/mem_cache/test_dsv4_c4_state_lifecycle.py (模块 C4 状态生命周期; 类别 test; 类型 test-coverage; 符号 TestUnifiedC4StateLifecycle, test_pool_size_is_exact_request_ring_product, test_clear_resets_only_selected_request_rings, test_clear_is_noop_off_the_unified_path) : 新增核心测试: 验证 C4 state per-request ring 尺寸计算、选择性清零、alloc hook 仅对新槽位触发

关键符号: _get_bytes_per_full_token, _unified_c4_state_pool_size, _fixed_c4_state_bytes, _fixed_swa_bytes, _resolve_max_running_requests_per_worker,

clear_c4_req_states, get_swa_ring_size, register_on_alloc_rows,
is_swa_req_ring, _swa_budget_for_req

关键源码片段

[python/sclang/srt/mem_cache/allocator/swa.py](#)

SWA 分配器核心适配：启用 per-request ring 模式，重写

available_size/swa_available_size 语义，alloc_extend/alloc_decode 走 full-only 路径

```
class SWATokenToKVPoolAllocator(BaseTokenToKVPoolAllocator):
    """Allocator for SWA hybrid KV cache."""

    # Per-request SWA ring (BaseSWAKVPool.swa_req_ring_size). Class default so
    # subclasses that bypass this __init__ read False.
    _swa_req_ring = False

    def __init__(self, size, size_swa, page_size, dtype, device, kvcache,
                 need_sort, req_to_token_pool=None):
        # ... existing init ...
        ring_size = kvcache.swa_req_ring_size
        self._swa_req_ring = ring_size is not None
        self._req_to_token_pool = req_to_token_pool
        if self._swa_req_ring:
            assert req_to_token_pool is not None, (
                "per-request SWA ring: capacity is counted in req slots"
            )
            self._swa_ring_cost = (
                (ring_size + self.page_size - 1) // self.page_size
            ) * self.page_size
            # Total SWA capacity is every req slot's ring; all slots are free here.
            self._size_swa = req_to_token_pool.available_size() * self._swa_ring_cost
            logger.info(
                "SWA per-request ring accounting enabled: "
                f"ring_size={ring_size}, ring_cost_tokens={self._swa_ring_cost}, "
                f"size_swa={self._size_swa} (paged size_swa={size_swa} bypassed)"
            )
        else:
            self._swa_ring_cost = 0

    def available_size(self):
        if self._swa_req_ring:
            # SWA ring is pre-allocated per slot and reused by decode,
            # so it never constrains token growth; full attention is the real limiter.
            return self.full_attn_allocator.available_size()
        return min(
            self.full_attn_allocator.available_size(),
            self.swa_attn_allocator.available_size(),
        )
```

```
def swa_available_size(self):
    if self._swa_req_ring:
        # Ring-based availability: free request slots * per-slot ring cost.
        return self._req_to_token_pool.available_size() * self._swa_ring_cost
    return self.swa_attn_allocator.available_size()
```

```
def is_swa_req_ring(allocator) -> bool:
    return isinstance(allocator, SWATokenToKVPoolAllocator) and allocator.swa_req_ring
```

评论区精华

Review 讨论集中在两个方面：(1) Disaggregation 路径安全性 — ShangmingCai 确认 decode.py 中的改动对 PD 分离路径安全，并建议后续 PR 减少 `getattr` 使用以遵循仓库防御性编码规范，hnyls2002 已在本次 PR 中将 `getattr(kvcache, 'swa_req_ring_size')` 替换为类型化属性访问；(2) CI 验证 — CI gate 测试当前在 head 上失败（未诊断），但 Extra 和 NV CI 通过，HaiShaw 确认 NV CI 干净。Issue 中 yuttian1 补充了 benchmark 细节：c=512 使用了 7 baseline + 6 PR + 3 baseline-control 运行，baseline-control 方差大于 PR delta，因此结论为无可测量回归；内存差额 11.63 GiB 已完全解释为 SWA/C4 state 从 token 级池转为 per-request ring 后释放的容量，无 PR 特异性残留。

- Disaggregation 路径安全性 (design): 已确认安全，`getattr` 优化留待后续 PR
- CI gate 失败未诊断 (correctness): 待诊断，Extra 和 NV CI 通过，HaiShaw 确认 NV CI 干净
- 内存差额解释 (correctness): 已解释，分配账本匹配 PR 实际占用

风险与影响

- 风险:
 1. 门控不完全风险（已缓解）：PR#30315 被回滚正是因为 shared-path 变更未 gate。本次 reland 逐文件审计，所有 unified-KV 分支均使用 `self._unified is True` 或 `is_swa_req_ring()` 判断，并通过 identity-compare 扫描消除 truthy 陷阱。
 2. 调度器 SWA 容量估算偏差：ring 模式下 `swa_available_size()` 返回 `free_req_slots * ring_cost`，而调度器的 `swa_growth_potential` 计算假设 token 级线性增长，ring 模式下 growth 实际为 0（预分配），`swa_evictable` 被显式置 0 避免双重计数。若未来 ring 大小动态调整需同步修改此处。
 3. PD decode 未诊断的 CI 失败：PR body 提到 pr-gate 在当前 head 上失败，原因未诊断，可能导致下游 suite 未运行。
 4. token-cap 单调性约束：config_from_budget 新增 assert 确保 token constraints 不提升容量，若未来引入 `max()` 以外的约束需同步更新。
 5. XPU 无 use_req_ring 路径：compress plan builder 对 XPU 添加了 assert，若 XPU 后续支持 unified-KV 需实现对应 plan 生成逻辑。- 影响：该 PR 直接影响 DeepSeek-V4 统一 KV 内核路径的内存池容量计算，将单 DP worker full-token 容量从 6,848,000 提升至 12,569,600 (+83.55%)，显著提高吞吐上限。对非统一（fp8）路

径无影响。对用户的影响：使用 unified-KV 的 AMD MI355X 部署可获得近翻倍的 token 容量，直接提升批处理吞吐和长序列支持能力。对团队的影响：统一了 SWA/C4 状态的内存生命周期模型（per-request ring vs per-token pool），为后续 speculativ decode 和 PD disaggregation 的 ring 复用奠定基础；测试覆盖新增了 4 个 mock 测试和 3 个回归测试，强化了池容量计算的正确性保证。 - 风险标记：CI gate 失败未诊断，调度器 SWA 容量估算与 ring 模式语义差异，XPU unified-KV 路径 assert 兜底，token-cap 单调性约束需同步维护

关联脉络

- PR #30315 [AMD][DSV4] Fix unified-KV pool sizing and SWA ring accounting: 原始修复 PR，本次 reland 的源头；因未充分 gate 被回滚
- PR #38163 Revert "[AMD][DSV4] Fix unified-KV pool sizing and SWA ring accounting": 回滚 PR#30315 的原因；本次 PR 的动机
- PR #38192 当前 PR: relend PR#30315 并完整 gate 所有 unified-KV 分支