

PR #36729 完整报告

sgl-project/sglang

Use a shared byte budget for unified hybrid-SWA memory

合并时间: 2026-09-16 06:27

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/36729>

执行摘要

- 一句话: 统一混合 SWA 内存池改用共享字节预算并重构准入记账
- 推荐动作: 值得精读。重点学习三处设计决策: 一是 `create_prefill_budget` 工厂把准入记账下沉到分配器, 消除调度器对池类型的 `if/else` 分支; 二是 `can_reserve/reclaim_plan` 联合判定配合穷举式派生属性测试, 是防御共享字节记账回归的有效范式; 三是兄弟类重构解决了 Mamba 三池错误继承两池策略的抽象问题。遗留跟进项 (`rationale` 注释、`PrefillDelayer` 水线、动态口径文档化) 应在后续 PR 中闭环。

功能与动机

PR body 明确说明: 统一的全注意力池与滑窗池共享同一字节 arena, 但静态 per-pool 容量门会拒绝那些本可借用对方空闲字节的分配, 容量报告必须跟随同样的动态分配状态。本 PR 是 #36403 合并后的 follow-up 栈第一项, 目标是让 hybrid-SWA unified memory 的准入与回收都基于真实可用的共享字节空间, 而不是两个互不感知的静态分区。

实现拆解

1. 新增 allocator-owned 准入预算抽象: 新建 `python/sglang/srt/mem_cache/prefill_budget.py`, 定义 `PrefillBudget` / `SWAPrefillBudget` / `SharedSWAPrefillBudget` 三级预算, 承载 `check_prefill`、`reserve`、`fit_chunk`、`available_chunk_tokens` 等准入记账; 同时在 `allocator/base.py`、`allocator/swa.py`、`allocator/hisparse.py` 增加 `create_prefill_budget` 工厂方法, 让每种分配器返回与自己容量口径匹配的预算对象, 调度器不再感知具体池类型。
2. 重构 unified hybrid-SWA 分配器: `unified_hybrid_swa.py` 拆出 `UnifiedSWAAllocatorBase` 基类 (共享虚拟 ID/ 生命周期 / 索引翻译), `UnifiedSWATokenToKVPoolAllocator` 与 `UnifiedMambaSWATokenToKVPoolAllocator` 改为兄弟类, 避免 Mamba 三池错误继承两池的联合字节预算判定; 新增 `can_reserve` / `reclaim_plan` / `_fits_page_demand` / `current_full_capacity` / `current_swa_capacity`, `evict_from_tree_cache` (`mem_cache/common.py`) 改为按 `reclaim_plan` 做联合回收。
3. 池尺寸显式字节预算: `model_executor/pool_configurator.py` 新增 `_unified_pool_bytes` 与 `_make_pool_config`, `MemoryPoolConfig` 增加 `unified_memory_pool_bytes` 字段, `kv_cache_configurator.py` 据此直接给 arena 定字节预算; draft worker 显式清空该字段, 避免继承 target 预算; `unified_memory_pool.py` 工厂收敛为单一 `total_bytes` 参数, 移除冗余的 `unified_total_bytes` 双输入与互斥校验。

4. 调度与统计配套: `schedule_policy.py` 的 `PrefillAdder` 改为持有 `memory_budget` 并委托全部预算计算, 删除约 246 行手写 `rem_*_token_offset` 逻辑; `scheduler.py` 的 `enqueue` 裁剪拆出 `_clip_unified_swa_max_new_tokens`, 统一“FULL 需求只取整一次、由 `can_reserve` 做页对齐”的规则; `pool_stats_observer.py` 改用动态容量口径报告; `kv_index_translator.py` 新增 `capture_token_capacity` 修复 `capturer` 索引契约。
5. 测试与清理: 新增 `test_prefill_memory_budget.py`、`test_unified_byte_budget_sizing.py` 等 13+ 个测试文件, 扩展 `test_prefill_adder.py`、`test_pool_configurator.py`、`test_multi_ended_allocator.py`, 覆盖未对齐 final chunk、ignore-EOS 入空池、reclaim_plan 穷举、draft 预算隔离、派生属性边界; 删除死代码 `recover_swa_with_locked_full`, 并将 HiSparse fixture 适配 Python 3.10 的 with 语法。

关键文件:

- `python/sglang/srt/mem_cache/prefill_budget.py` (模块 准入预算; 类别 source; 类型 core-logic; 符号 `estimate_swa_kv_tokens`, `PrefillBudget`, `SWAPrefillBudget`, `SharedSWAPrefillBudget`): 新增的准入预算抽象核心文件: 定义 `PrefillBudget` / `SWAPrefillBudget` / `SharedSWAPrefillBudget` 三级预算与 `estimate_swa_kv_tokens` 估算器, 承载共享字节预算下的 `check_prefill`、`reserve`、`fit_chunk` 全部逻辑, 是本 PR 架构转变的主入口。
- `python/sglang/srt/mem_cache/allocator/unified_hybrid_swa.py` (模块 共享池; 类别 source; 类型 core-logic; 符号 `UnifiedSWAAllocatorBase`, `UnifiedSWATokenToKVPoolAllocator`, `UnifiedMambaSWATokenToKVPoolAllocator`, `can_reserve`): 分配器重构核心: 拆出 `UnifiedSWAAllocatorBase`, 两池与 Mamba 三池变兄弟类; 新增 `can_reserve` / `reclaim_plan` / `current*_capacity` 联合判定与动态容量报告, 是共享字节预算的落地载体。
- `python/sglang/srt/managers/schedule_policy.py` (模块 调度策略; 类别 source; 类型 core-logic; 符号 `PrefillAdder`, `memory_budget`, `rem_total_tokens`, `cur_rem_tokens`): `PrefillAdder` 从手写 `rem_*_token_offset` 改为持有 `allocator` 创建的 `memory_budget`, 删除约 246 行分支记账逻辑, 是准入路径的主要简化点。
- `python/sglang/srt/model_executor/pool_configurator.py` (模块 池配置; 类别 source; 类型 data-contract; 符号 `MemoryPoolConfig`, `_unified_pool_bytes`, `_make_pool_config`, `_max_unified_full_tokens`): 新增 `unified_memory_pool_bytes` 数据契约: `_unified_pool_bytes` 计算显式共享字节预算, `_make_pool_config` 统一构造 `MemoryPoolConfig`, `draft` 与 `target` 的预算隔离从这里开始。
- `python/sglang/srt/mem_cache/allocator/base.py` (模块 分配器; 类别 source; 类型 core-logic; 符号 `create_prefill_budget`, `max_new_tokens_for_memory`, `evict_to_free_tokens`): 定义 `create_prefill_budget` 工厂契约与 `max_new_tokens_for_memory`、`evict_to_free_tokens` 新语义, 是所有分配器对准入预算抽象的接入点。
- `test/registered/unit/mem_cache/test_prefill_memory_budget.py` (模块 单元测试; 类别 test; 类型 test-coverage; 符号 `TestSharedPrefillMemoryBudget`, `TestSharedPrefillAdmission`, `test_pending_batch_cannot_spend_shared_bytes_twice`, `test_unaligned_final_chunk_makes_progress`): 新增 300 行核心回归: 共享预算的

pending 请求不能重复花共享字节、混合 decode 两侧预留、前缀锁影响准入、未对齐 final chunk 与 ignore-EOS 入空池等，是 P1 修复的直接保障。

- test/registered/unit/managers/test_prefill_adder.py (模块 单元测试; 类别 test; 类型 test-coverage; 符号 create_shared_adder, create_shared_req, test_shared_admission_reserves_all_pending_requests, test_shared_admission_rechecks_after_prefix_lock) : 扩展 172 行: 覆盖共享准入预留全部 pending 请求、前缀锁后重新判定、decode 消耗 chunk 预算时 continuation 延迟, 以及 memory_budget 偏移断言。
- python/sglang/srt/mem_cache/kv_cache_configurator.py (模块 缓存配置; 类别 source ; 类型 core-logic; 符号 hybrid_swa_token_capacity, _init_unified_swa_pools) : 承载 hybrid_swa_token_capacity 与 draft 预算清除逻辑: effective_max_total_num_tokens 的容量策略从 model_runner 迁移至此, draft worker 不再继承 target 的 unified_memory_pool_bytes。

关键符号: estimate_swa_kv_tokens, PrefillBudget, SWAPrefillBudget, SharedSWAPrefillBudget, create_prefill_budget, can_reserve, reclaim_plan, current_full_capacity, current_swa_capacity, _clip_unified_swa_max_new_tokens, hybrid_swa_token_capacity, _unified_pool_bytes, _make_pool_config, evict_to_free_tokens, max_new_tokens_for_memory

关键源码片段

python/sglang/srt/mem_cache/allocator/unified_hybrid_swa.py

分配器重构核心: 拆出 UnifiedSWAAllocatorBase, 两池与 Mamba 三池变兄弟类; 新增 can_reserve / reclaim_plan / current_*_capacity 联合判定与动态容量报告, 是共享字节预算的落地载体。

```
class UnifiedSWAAllocatorBase(SWATokenToKVPoolAllocator):
    """共享 FULL/SWA 虚拟 ID、分配生命周期与索引翻译的公共基类

    两个具体子类 (两池 / Mamba 三池) 各自定义容量策略, 避免
    Mamba+SWA 池错误继承两池的联合字节预算判定。
    """

    # 动态容量口径: 当前可用 + 已分配, 随对侧实际字节占用浮动
    # 这取代了旧的静态 per-layer 上限作为统计分母
    @property
    def current_full_capacity(self) -> int:
        return self.full_available_size() + self.full_attn_allocator.allocated_count()

    @property
    def current_swa_capacity(self) -> int:
        return self.swa_available_size() + self.swa_attn_allocator.allocated_count()

    # 每侧容量视图: 静态守恒上限约束借贷方, schedulable 视图约束
    # 向共享间隙增长的一侧, 两者取小避免过承诺
    def full_available_size(self) -> int:
```

```

return min(
    self._conserve_full_available_size(),
    self.schedulable_full_available_size(),
)

```

python/sglang/srt/managers/schedule_policy.py

PrefillAdder 从手写 `rem_*_token_offset` 改为持有 allocator 创建的 `memory_budget`，删除约 246 行分支记账逻辑，是准入路径的主要简化点。

```

# PrefillAdder 不再自己维护 rem_total_token_offset / rem_swa_token_offset 两套
# 手写偏移，而是把准入记账整体委托给 allocator 创建的预算对象
self.memory_budget = token_to_kv_pool_allocator.create_prefill_budget(
    tree_cache, num_mixed_decode_tokens=num_mixed_decode_tokens
)

```

```

if running_batch is not None:
    # 运行中请求的估算 token 占用并入预算的 total 偏移
    self.memory_budget.total_offset += sum(
        self._get_running_request_total_token_offset(r)
        for r in running_batch.reqs
    )

```

```

@property
def rem_total_tokens(self):
    return self.memory_budget.remaining_total

```

```

@property
def cur_rem_tokens(self):
    return self.memory_budget.remaining_current

```

```

def _check_prefill_budget(self, req, *, extend_input_len, total_tokens, swa_host_hit_length):
    # 准入统一走预算对象的 check_prefill: fixed / shared / ring 三种
    # 分配器各自实现容量口径，调度器不再感知具体池类型
    return self.memory_budget.check_prefill(
        extend_input_len=extend_input_len,
        total_tokens=total_tokens,
        max_new_tokens=self._swa_new_tokens(req),
        input_tokens=len(req.full_untruncated_fill_ids),
        swa_host_hit_length=swa_host_hit_length,
        chunk_limit=self.rem_chunk_tokens,
    )

```

评论区精华

评审由 ch-wan 与 merrymercy 多轮深度跟进，核心交锋如下：

- ch-wan 的第一轮总结："核心记账是自洽的……残余风险与其说是 ' 此方案不可行 '，不如说是 ' 新数学几乎没有被测试 '。" 作者最初保持测试中立，后在 c41b06843c 补齐派生属性与空池字节边界覆盖。

- merrymercy 提问: "do we have a e2e unified hybrid swa memory pool test with an engine launch?" 作者确认 test_gemma4_unified_swa_virtual_ids.py 会启动真实 gemma-4-E2B-it 引擎并跑 200 条 GSM8K, 但也明确 "I have not rerun this engine-launch test for the newly pushed fixes", 随后通过 /rerun-test 补跑一次通过。
- merrymercy 指出: "class UnifiedMambaSWATokenToKVPoolAllocator(UnifiedSWATokenToKVPoolAllocator) is wrong abstraction", 作者在 f20860697232 中把两个分配器改为 UnifiedSWAAllocatorBase 下的兄弟类。
- ch-wan 报告 P2: enqueue 裁剪先对 prompt 取整、can_reserve 再取整一次, 导致 page_size=4 时 FULL 需求按 13 而非 10 记账, 可行生成被裁剪到 0 token; 作者修复为单一取整并加 test_unified_budget_rounds_prompt_and_decode_together。
- merrymercy 报告两个 P1: 未对齐 final chunk 与 ignore-EOS 入空池在共享预算下被永久拒绝 (fit_chunk 多计一页 FULL), 作者在 1f3b496c23 修复并新增 TestSharedPrefillAdmission 四个真实 Req/PrefillAdder 边界回归。
- 未闭环项: ch-wan 要求把 min(extend+decode, window) 与 chunk 收缩的 rationale 注释移入 estimate_swa_kv_tokens, 作者承认仍待补; 统计分母变为动态容量后, PrefillDelayer 水线是否需要重调也未处理, 作者标注为后续跟进。
- 双重取整导致可行生成被裁剪为 0 token (correctness): 修复为 _clip_unified_swa_max_new_tokens 单一取整、can_reserve 统一舍入, 并加 test_unified_budget_rounds_prompt_and_decode_together 回归。
- 未对齐 final chunk 与 ignore-EOS 入空池被永久拒绝 (correctness): 1f3b496c23 修复: 共享预算下统一传未取整长度, 新增 TestSharedPrefillAdmission 四个真实 Req/PrefillAdder 边界回归, page_size 覆盖 4/64。
- estimate_swa_kv_tokens 的 allocation_limit=0 语义重载 (design): 先加 include_allocated_tail 标志, 最终在 ab2ea29 删除该标志与 _swa_reserved_tokens, 统一为 allocation_limit 单一参数。
- reclaim_plan 对 SWA>FULL 需求直接返回 None 导致队头阻塞 (correctness): 4d1fa0ff 移除 SWA>FULL 的硬拒绝, 按供需处理; test_restore_swa_without_allocating_more_full 与穷举式 reclaim 测试覆盖。
- draft worker 继承 target 的共享字节预算 (correctness): ab2ea29 在 draft config 显式清除 unified_memory_pool_bytes, 并加 test_draft_does_not_inherit_target_shared_byte_budget 回归。
- UnifiedMamba 池继承两池分配器的抽象错误 (design): f208606 拆出 UnifiedSWAAllocatorBase 作为公共基类, 两池与三池变兄弟类, 原 supports_asymmetric_reservation 标志机制被移除。
- 派生属性测试缺口: _max_unified_full_tokens、can_reserve、reclaim_plan (testing): c41b068 补 derived-property 覆盖 (最大 page-aligned full 容量、空池 packed-byte 边界); test_multi_ended_allocator 增加 reclaim_plan 穷举对照。
- 动态容量口径对 token usage 指标与 capturer 的影响 (question): 动态口径保留, capturer 索引修复; PrefillDelayer 政策未改, 水线重调作为运维跟进项。
- estimate_swa_kv_tokens 丢失 rationale 注释 (documentation): 作者同意并确认仍未恢复, 作为文档跟进项开放。

风险与影响

- 风险：技术风险集中在以下几处：

1. 取整语义回归：本 PR 历史上有两个 P1/P2 缺陷都源于“FULL 需求双重取整”。修复后统一为 `can_reserve` 单一舍入口径，但该契约依赖调用方传入未取整长度；后续新增调用者（如 HiCache/PD 路径在 #37496 才可达）若沿用旧的 `paged_input` 传参方式，会重新触发“enqueue 接受但 PrefillAdder 永远拒绝”的队头阻塞。
2. 动态容量口径对指标与缓冲的影响：`pool_stats_observer` 的分母从静态 per-layer 上限变为 `available + allocated`，同一负载 token usage 从 0.38 变为 0.21；`PrefillDelayer` 的 `waterwall` 与 `capturer` 尺寸都可能需重调，`capturer` 索引契约已由 `kv_index_translator.capture_token_capacity` 修复，但 GPU 环境下的长稳表现未充分验证。
3. 启动期行为收紧：`server_args._handle_unified_memory_pool` 曾把 `--enable-unified-memory` 与 PP/mixed-chunk 组合一律断言禁止，后放宽为仅 PD 场景限制 PP；切换过程中若用户配置恰好踩线，会在启动阶段直接失败。
4. CI 未全绿即合入：PR body 中三个 CI 看板均为小红叉；合入主要依赖本地 331 passed/722 subtests 与单次 gemma-4 E2E，未覆盖 multi-node PD、HiCache 组合，属于覆盖盲区。
5. 冻结文件规则越界：`model_runner.py` 的 `effective_max_total_num_tokens` 一度直接内联 unified 容量策略，后移至 `kv_cache_configurator.hybrid_swa_token_capacity()`；新契约暴露后，后续调用方仍需遵守 frozen-core 边界。- 影响：影响面限定在开启 `--enable-unified-memory` 的 hybrid-SWA 部署：调度准入、预取回收、统计口径、`capturer` 容量、draft worker 池尺寸均随共享字节预算改变；UMP-off 的静态分区路径保持上游行为。对运维而言，token usage 类指标含义变化，`--prefill-delayer-token-usage-low-watermark` 等参数可能需要重调。对团队而言，本 PR 确立了 allocator-owned budget 的契约、派生属性测试风格与兄弟类继承模式，为后续 UMP 堆栈（#36730/#36731/#37496）打下基础。- 风险标记：双重取整曾致 P1/P2 缺陷，动态容量口径需重调水线，CI 未全绿即合入，冻结文件规则越界后修复，文档跟进项未闭环

关联脉络

- PR #36403 Unified memory pool groundwork（前序，已合并）：PR body 明确说明本 PR 构建在其之上（"Builds on #36403, which is now merged"），是 unified memory 字节预算线的直接前序。
- PR #36730 Follow-up stack PR（讨论中提及）：review 评论确认本 PR 的修复已通过 #36730 传播，属于同一 UMP 共享预算演进栈。
- PR #36731 Follow-up stack PR（讨论中提及）：review 评论确认本 PR 的修复已通过 #36731 传播，属于同一 UMP 共享预算演进栈。
- PR #37496 HiCache allow-list 相关 PR（讨论中提及）：ch-wan 指出 #37496 的 HiCache allow-list 使 `reclaim_plan` 的 SWA>FULL 缺陷变得可达，说明该 PR 与本 PR 的联合回收逻辑直接交互。