PR #35158 完整报告

sgl-project/sglang

feat(unified-memory): byte-budget sizing, feasibility floor, and a conservation verifier

合并时间：2026-09-01 06:09

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35158

执行摘要

- 一句话：统一内存按字节预算定容，新增守恒校验与 bs=1 可行性检查
- 推荐动作：值得精读。核心设计决策包括：allocator-owned hook + 默认逐位兼容、诊断只读不 raise 的可用性取舍、_reserved_floor_bytes 单一事实来源、方向无关的 chain 排序、flooring 而非 rounding 的 4096 B 对齐。对想理解 SGLang 统一内存池演进方向（N 子池、字节准入）的工程师，这是很好的中间态样例，建议结合 #35154 与 #35177 一起阅读。

功能与动机

PR body 明确列出三个 gap：

1) buffer 由重求和 ratio-derived token counts 决定而非真正 profiling 得到的 byte budget，SWA split 先按 per-token cell size floor 预算、再对每侧 token 数做 page-align，重求和会重建出比实际 profiling 少的容量——每侧最多差约一个 page 的 token，这部分内存从未被分配； 2) under-sized pool 在运行时表现为 retract livelock 而非启动失败，单个最坏情况请求单独运行都装不下时，retract loop 无物可退、服务器挂起； 3) unified pool 的字节记账没有守恒检查，watermark spans、holes、pending compaction、frontier ordering 的漂移会让计数器准入到并不空闲的内存——静默损坏而非崩溃。PR 同时声明：所有新增 hook 的默认值精确复现当前行为，静态池不受影响。

实现拆解

1. 容量判定下沉到分配器：allocator/base.py 在 BaseTokenToKVPoolAllocator 上新增 evict_to_free_tokens、check_decode_capacity、verify_byte_accounting 三个默认 hook，各自精确复现历史行为（token 驱逐、available_size() >= num_tokens、空列表），使 schedule_batch.py::check_decode_mem 直接委托、scheduler.py::on_idle 漏检查经既有 leak-report 通道上报，调度器侧零功能分支。
2. 字节守恒验证器：multi_ended_allocator.py 新增 _byte_accounting_violations（frontier 越界 + lazy 模式 watermark span == live + holes + pending）、_chain_byte_accounting_violations（chain 内 frontier 全序，检测共享 buffer 中 band 重叠）、_end_pair_chain（按 grow direction 排序，参数顺序无关）；mamba 与 SWA 两个 composite 各自实现 verify_byte_accounting。违规只报告不 raise，避免 idle 路径因诊断升级造成可用性事件。

3. 从 profiled budget 直接定容：MemoryPoolConfig 新增 unified_total_bytes 数据契约字段；kv_cache_configurator.py::config_from_budget 在 unified 路径把预算 floor-align 到 4096 B 后写入（factory 会 .view() 整个 uint8 buffer 为 KV dtype，flooring 保证不 overcommit），经 _PoolSizes 传入 mamba/SWA factory；init_unified_swa_pools 直接以预算为 buffer 大小，init_unified_mamba_pools 在预算上叠加 state pool 字节；不设预算或 token-capped 重推导时字段保持 None，走旧 token 数求和逐位兼容，draft worker 显式传 None。
4. bs=1 可行性下限：新 _reserved_floor_bytes 把 slot-0 sink 预留公式收敛为单一事实来源（mamba entry 不乘 page_size），UnifiedKVPool.__init__ 与两个 factory 的 floor 共用；_check_bs1_feasibility_floor 在构造前对 floor_terms 求和，不足则 raise 带分项明细的 RuntimeError。Mamba floor 只收 3 个 state slot + sink，SWA floor 只收 min(model_context_len, sliding_window_size + page_size) 的窗口 + sink——full-attention 侧不收，因为 TpModelWorker.get_worker_info 已把 max_req_len clamp 到 pool，过长请求在准入时被拒而非 livelock。
5. 测试配套：新增 test_unified_byte_budget_sizing.py（预算精确认可、fallback 保真、rounding 场景、floor 单一来源、bs=1 边界与上下文超长回归）和 test_unified_byte_accounting.py（健康生命周期每步守恒、live/hole/watermark 单点漂移检测、chain 重叠与方向无关排序），均注册到 base-a-test-cpu 套件。

关键文件：

- python/sglang/srt/mem_cache/multi_ended_allocator.py（模块 字节分配器；类别 source；类型 core-logic；符号 _byte_accounting_violations, _chain_byte_accounting_violations, _end_pair_chain, verify_byte_accounting）：字节守恒验证器的核心实现：per-pool 守恒、chain frontier 全序、方向无关配对，并挂到 mamba 与 SWA 两个 composite 上。
- python/sglang/srt/mem_cache/unified_memory_pool.py（模块 统一内存池；类别 source；类型 core-logic；符号 _reserved_floor_bytes, _check_bs1_feasibility_floor）：profiled budget 直接定容、bs=1 可行性 floor 与 _reserved_floor_bytes 单一事实来源都在此落地，是行为变更的主战场。
- test/registered/unit/mem_cache/test_unified_byte_budget_sizing.py（模块 预算测试；类别 test；类型 test-coverage；符号 TestBudgetSizing, test_swa_factory_honors_the_budget_exactly, test_fallback_is_the_token_count_resum, test_budget_beats_resum_on_rounding）：预算精确认可、fallback 保真、rounding 场景、floor 单一来源与 bs=1 边界回归测试，是行为契约的锁定点。
- test/registered/unit/mem_cache/test_unified_byte_accounting.py（模块 守恒测试；类别 test；类型 test-coverage；符号 TestHealthyLifecycleReportsClean, TestDriftReportsLoudly, TestChainFrontierOrder, test_overlapping_frontiers_report）：验证器健康生命周期、单点漂移检测与 chain 重叠场景测试，证明诊断非空转。
- python/sglang/srt/mem_cache/allocator/base.py（模块 分配器基类；类别 source；类型 core-logic；符号 evict_to_free_tokens, check_decode_capacity, verify_byte_accounting）：allocator-owned 容量 hooks 的基类默认实现，调度器零功能

分支的关键。

- python/sglang/srt/mem_cache/kv_cache_configurator.py（模块 缓存配置器；类别 source；类型 core-logic）：统一内存预算从配置到 factory 的传递链路，含 draft worker 与 token-capped 路径的降级处理。
- python/sglang/srt/managers/scheduler.py（模块 调度器；类别 source；类型 core-logic）：idle leak battery 接入字节守恒诊断，经既有 leak-report 路径上报。
- python/sglang/srt/model_executor/pool_configurator.py（模块 池配置器；类别 source；类型 data-contract）：MemoryPoolConfig 新增 unified_total_bytes 数据契约字段。
- python/sglang/srt/managers/schedule_batch.py（模块 批调度；类别 source；类型 core-logic）：check_decode_mem 语义改为委托 allocator，是容量判定下沉的消费端。

关键符号：evict_to_free_tokens, check_decode_capacity, verify_byte_accounting, _byte_accounting_violations, _chain_byte_accounting_violations, _end_pair_chain, _reserved_floor_bytes, _check_bs1_feasibility_floor

关键源码片段

python/sglang/srt/mem_cache/multi_ended_allocator.py

字节守恒验证器的核心实现：per-pool 守恒、chain frontier 全序、方向无关配对，并挂到 mamba 与 SWA 两个 composite 上。

```
def _byte_accounting_violations(self) -> List[str]:
    """Per-sub-pool 字节守恒：frontier 必须落在共享 buffer 内；
    lazy 模式下 watermark span 必须等于 live + holes + pending。
    这是 idle 期诊断，纯 host 算术，不触碰 GPU 数据。"""
    out: List[str] = []
    total = self.unified_buffer.total_bytes
    lo_b, hi_b = self._byte_low_frontier(), self._byte_high_frontier()
    # grow-up 侧 high 是最后分配页之后的字节，grow-down 侧 low 是
    # 最低存活页之下的字节；任何一侧越界都说明 band 跑出了 buffer。
    if not (0 <= lo_b <= hi_b <= total):
        out.append(
            f"[{self.sub_pool_name}] frontier out of bounds: "
            f"low={lo_b}, high={hi_b}, total={total}"
        )
    if self.lazy_compaction:
        # lazy 端：watermark 区间内含 live + holes + pending（eager
        # 无洞与待压实，span 天然等于 live）。任一计数器漂移都会现形。
        holes = int(self._free_phys_pages.numel())
        pending = len(self._pending_reuse_pages_cpu)
        wm_span = self._allocated_pages()
        if wm_span != self.live_page_count + holes + pending:
            out.append(
                f"[{self.sub_pool_name}] span {wm_span} != live "
                f"{self.live_page_count} + holes {holes} + pending {pending}"
            )
    return out
```

```
def _chain_byte_accounting_violations(
    chain: List[MultiEndedAllocator],
) -> List[str]:
    """按低地址→高地址排列的一串 band allocator 的守恒检查：每个成员各自
    守恒之外，还要求后一个成员的 low frontier 越过前一个成员的 high
    frontier，否则两个 band 在同一 byte buffer 里重叠。今天的 chain 是
    2-pool end pair；N-pool track 会在中间插入 float middles。"""
    out: List[str] = []
    for a in chain:
        out.extend(a._byte_accounting_violations())
    frontier = 0
    for a in chain:
        lo_b, hi_b = a._byte_low_frontier(), a._byte_high_frontier()
        if lo_b < frontier:
            out.append(
                f"[chain] {a.sub_pool_name} low frontier {lo_b} overlaps the "
                f"previous pool's high frontier {frontier}"
            )
        frontier = max(frontier, hi_b)
    return out


def _end_pair_chain(
    a: MultiEndedAllocator, b: MultiEndedAllocator
) -> List[MultiEndedAllocator]:
    """把 end pair 按 grow direction 排成低→高顺序：工厂与单测夹具对 pair
    的朝向不同，chain 检查必须对参数顺序不敏感（grow-up 在前、grow-down
    在后）。"""
    return sorted((a, b), key=lambda x: x.grow_direction != "up")
```

python/sglang/srt/mem_cache/unified_memory_pool.py

profiled budget 直接定容、bs=1 可行性 floor 与 _reserved_floor_bytes 单一事实来源都在此落地，是行为变更的主战场。

```
def _reserved_floor_bytes(sub_pool_specs: List[SubPoolSpec], page_size: int) -> int:
    """buffer 底部为 slot-0 哑写预留的 padding sink 字节数。

    slot-0 哑写会落在这里；每个 sub-pool 的第一个可分配 slot 选在它之后。
    page-aware 子池的 slot-0 写会触达整个 page-0 envelope（最多
    page_size * entry_bytes），而 mamba 子池是 page_size=1，其 entry 只收一次——
    若按 page 乘 mamba entry，会在 pool 里虚留 page_size * ~100 MB 从不会被
    触碰的缓冲（GPU eval_434/436 Falcon-H1 启动回归即源于此）。

    `UnifiedKVPool` 预留的正是这个值，factory 的 bs=1 floor 也收这个值——
    单一事实来源，避免公式再次漂移。"""
    return max(
        [max(s.entry_bytes() for s in sub_pool_specs)]
        + [
            page_size * s.entry_bytes()
```

```
            for s in sub_pool_specs
            if not isinstance(s, MambaSubPoolSpec) # mamba 是 page_size=1
        ]
    )


def _check_bs1_feasibility_floor(
    *, total_bytes: int, floor_terms: List[Tuple[str, int]], factory: str
) -> None:
    """bs=1 可行性下限——retract 循环的终端保证。

    调度器会把请求 retract 到只剩最后一个；若最坏情况请求单独运行都放不进
    buffer，配置过小就会变成运行时 retract livelock（挂起而非报错）。因此
    在任何 pool 构建之前，启动期就要带明细地 fail loud。total_bytes >= floor
    时直接通过（>= 是契约，恰好相等不算失败）。"""
    floor = sum(b for _, b in floor_terms)
    if total_bytes >= floor:
        return
    detail = " + ".join(f"{name}={b}" for name, b in floor_terms)
    raise RuntimeError(
        f"[unified-memory-pool] {factory}: byte budget {total_bytes} cannot fit "
        f"ONE worst-case request (bs=1 floor {floor} = {detail}). A pool this "
        f"size retract-livelocks at runtime. Raise --mem-fraction-static, lower "
        f"the model context length, or reduce reserved memory."
    )
```

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

allocator-owned 容量 hooks 的基类默认实现，调度器零功能分支的关键。

```
# -- scheduler-facing capacity hooks --
# 调度器无条件调用这些 hook（调度器侧零功能分支）；默认实现精确复现历史
# token 行为，unified composite 用字节口径逻辑覆盖它们。

def evict_to_free_tokens(self, tree_cache, num_tokens: int) -> None:
    """让前缀缓存驱逐未锁条目，直到此 allocator 能满足 num_tokens
    （或没有可驱逐项）。默认走共享的 token 数驱逐；joint-byte composite
    覆盖（驱逐一个多生命周期树节点会在多个 side 同时释放字节）。"""
    from sglang.srt.mem_cache.common import evict_from_tree_cache

    evict_from_tree_cache(tree_cache, num_tokens)

def check_decode_capacity(self, *, num_tokens: int, tree_cache) -> bool:
    """下一个 decode 步的 num_tokens 分配是否放得下（先驱逐可回收缓存）。
    retract 循环收敛到同一个检查，所以 allocator 侧的不足会优雅 retract，
    而不是触发 fail-loud alloc 错误。默认复现历史
    `ScheduleBatch.check_decode_mem` 的正文；unified composite 用字节门
    + 自己的逐拍预留覆盖。"""
    self.evict_to_free_tokens(tree_cache, num_tokens)
    return self.available_size() >= num_tokens
```

```
def verify_byte_accounting(self) -> list:
    """idle 期守恒诊断：重算本 allocator 的字节/slot 记账，返回可读的
    违规串（空 == 健康）。默认：静态池没有字节模型。"""
    return []
```

评论区精华

本 PR 的 comments 与 review_comments 均为 0，评审交锋体现在合并者 ch-wan 随后的 3 个直达提交中：

- 移除 bs=1 floor 的 full-context 项（fix commit）：初版 floor 收取 model_context_len * full_entry_bytes，理由是“单个请求装不下就是 retract livelock”。ch-wan 指出 TpModelWorker.get_worker_info 已把 max_req_len clamp 到池容量，长于池的请求在准入时就被拒绝，不是 livelock；保留该项会让单 GPU 上的长上下文模型健康配置在 boot 时被拒。
- 修正 call site 注释（docs commit）：删掉 full-context term 后，调用点注释仍在论证被删 term 成立的理由。提交说明的原话是“You change the line, you own its comment”。
- 压缩冗余注释：byte-budget 与 bs=1 floor 的四段注释被压缩到事实所需长度，4096 B 对齐的 provenance（.view() 为 KV dtype）得到保留。

此外，测试文件记录了一次已被修复的回归：_reserved_floor_bytes 手抄公式曾对 mamba entry 乘以 page_size，产生约 33 GiB 幻影需求，导致 Falcon-H1 在 page_size=256 时启动失败（GPU eval_434/436）。PR 以单一来源函数 + 对拍测试锁定该行为。

- bs=1 floor 是否应计入 full-context token (correctness): 移除该 term；floor 只收 mamba state slots / SWA window + slot-0 sink，并同步修正 call site 注释。
- sink 预留公式的单一事实来源 (correctness): 单一来源 + 对拍测试（test_floor_sink_equals_what_the_pool_reserves）锁定。
- 守恒诊断报告 vs 升级异常 (design): 只报告不 raise 是生产姿势，严格模式留给验证场景。
- 解码容量 gate 的归属 (design): allocator-owned hooks，默认逐位复现历史行为；调度器零功能分支。

风险与影响

- 风险：
  1. 启动期新检查的误判风险：_check_bs1_feasibility_floor 计算错误会拒绝健康配置。历史上已有手抄公式把 mamba entry 按 page_size 乘的教训（约 33 GiB 幻影需求），本 PR 通过 _reserved_floor_bytes 单一来源与对拍测试缓解，但未来新增 sub-pool spec 类型时仍需同步维护该函数。
1. 内存足迹变化：直接以预算定容后，SWA 与 Mamba 模型的 unified buffer 各增加约 12.42 MB / 11.35 MB，Qwen3.5-9B 增加 0.36 MB，Kimi-Linear 增加 4.5 KB。这是利用率提升，但相同配置下显存占用略增，与 #35154 对比测量需留意基线漂移。
2. 容量判定语义迁移：check_decode_mem 从 schedule_batch.py 内联逻辑改为委托 allocator。静态池默认行为逐位复现，但 unified composite 未来的字节口径覆盖版本一旦算错，会表现为准入过宽（静默损坏）或过紧（吞吐下降）。当前 PR 未提供 unified 的

decode byte gate 覆盖（留给下一代字节准入账本），该风险是“已预留、未启用”。

3. 诊断非阻断：verify_byte_accounting 只报告不 raise，计数漂移不会被主动 fail——这是可用性取舍，实际依赖 leak-report 通道被运维监控；严格升级由 env 控制且默认关。
4. 兼容性：未设预算、draft worker、token-capped 路径全部回退旧行为；4096 B flooring 最多损失 4095 B 预算，不会 overcommit，静态池完全不受影响。 - 影响：
   - 用户 / 运维：配置过小的 unified pool 从“运行期服务器挂起”变为“启动时带分项明细报错”，可诊断性显著提升；统一内存池实际可用容量比旧 re-sum 多出每侧约一页 token。
   - 系统：unified-memory 功能线（#35154 之后的第二相位）补上容量闭环：预算 → 定容 → 可行性 → 运行时守恒校验。
   - 团队：确立了 allocator-owned capacity hook 模式，后续字节准入账本可以直接在 allocator 层落地；调度器不再感知池类型差异。
   - 范围：静态内存池路径完全不受影响；只影响开启 --enable-unified-memory 的 hybrid Mamba / SWA 模型。
   - 风险标记：核心分配路径变更，启动期新增 fail-loud 检查，内存足迹增加约 12 MB/ 侧，诊断默认只报告不中断，跨调度 / 配置 / 内存池模块改动

关联脉络

- PR #35154 unified-memory 早期相位（本 PR 的 stacked base）：PR body 明确要求先读 #35154；本 PR 的 diff 是在其 tip 之上的 4 个 commits，容量闭环的前置工作。
- PR #35177 feat(unified-memory): three sub-pools for mamba + hybrid-SWA models: unified 池泛化至 N 子池；本 PR 的 chain verifier 注释明确预留了 N-pool track 插入 float middles 的扩展点。
- PR #37167 [mem_cache] Make release, row-reuse asserts, and presence checks read the KV record: 同属 mem_cache 一致性工作线：那边统一 KV record 读取，这边统一字节守恒校验，互为补充。
- PR #37170 [unified-memory] Drop the vacated 'dense' qualifier and the restating comments: 同一功能线的术语与注释清理，说明 unified-memory 正在收尾稳定期。