PR #35867 完整报告

sgl-project/sglang

[diffusion] refactor: hand out pinned host memory per layer

合并时间：2026-08-22 09:37

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35867

执行摘要

- 一句话：offload 按层发放固定内存，大组件也能部分驻留
- 推荐动作：值得精读。这是 diffusion offload 内存管理的一次精细重构，四个缺陷的发现与修复过程很有代表性：重复计费、毛额 / 净额、过度承诺、静默 fallback。重点看 _plan_layer_hosting 的优先级设计（streamed 优先于 resident、尾部归还）与 anonymous_new_bytes 的净成本思维，以及测试如何用 torch.from_file 构造真实 checkpoint 映射场景来模拟 50 GiB 级问题。

功能与动机

PR 描述指出旧的 pin 预算按组件一次性申请：'The pin budget was asked once per component, all or nothing, so a DiT larger than the whole spendable budget pinned zero bytes'。在 MiniMax-H3 上，DiT 有 61.73 GiB 权重、其中 50.53 GiB 是 checkpoint 视图，而预算只剩 5.32 GiB，旧逻辑导致整个 DiT 一层也无法 pin。同时作者在过程中发现并修复了四个内存规划缺陷（重复计费、毛额计费、超额 pins 不归还、步数排名静默退化为裸字节），并用测量数据表明 pin 的传输重叠价值（Wan2.1 上 1.03 s vs 1.90 s 每步）。

实现拆解

1. 构造器注入预算与组件名：LayerwiseOffloadManager.__init__ 新增 pin_budget: HostPinBudget | None 与 pin_component_name 参数。没有显式预算时创建私有 HostPinBudget()，消除了旧行为中「无预算 → 每层都可负担 → copies-fit 检查永不触发」的漏洞。
2. 按层统计字节：新增 _layer_byte_totals，对每个 layer 统计总字节数以及其中属于 checkpoint 视图（_mapped_regions.holds 判定的 mapped 字节数），为后续净成本核算提供输入。
3. 按 streamed-resident 顺序规划 hosting：新增 _plan_layer_hosting 替代原先布尔化的 _keep_weights_on_their_mapping。按 _streamed_order 优先、resident 层按索引排序在后的确定性顺序逐层决策，预算内标记 pinned 并记录 pin_order，否则标记 pageable。
4. 净成本 fit 检查与 pins 归还：内部函数 anonymous_new_bytes 只统计从 mapping 复制进匿名内存的净新增字节（替换匿名原始 fused-qkv 的 store buffer 视为 wash，不计费）。若整个计划的净增量不适合，先把非 pinned 层降级为 mapped，再从 pin_order 尾部归还仍放不下的 pins，最后通过 _pin_budget.request 登记占用并输出各层去向的日志。

5. 步数解析修复与测试配套：步数排名不再读 server_args.pipeline_class_name 这个通常未设置的 override，而是按 build_pipeline 的方式从 model index 解析采样默认值（测试用 registry_mod.get_model_info 返回 sampling_param_cls 验证 50 步 DiT 排在 encoder 前）。测试文件扩展 _mapped_manager 支持 num_blocks、available_bytes、pin_budget_bytes 参数，新增 _FileBackedModel 多块模型与半映射半匿名的 _MixedBlock/_MixedModel，并补充 4 个针对性回归测试。

关键文件：

- python/sglang/multimodal_gen/runtime/managers/memory_managers/layerwise_offload.py（模块 层间卸载；类别 source；类型 core-logic；符号 _keep_weights_on_their_mapping, _layer_byte_totals, _plan_layer_hosting, anonymous_new_bytes）：核心逻辑所在：pin 预算从整组件一次性申请改为按层发放，新增 _layer_byte_totals 与 _plan_layer_hosting，修复重复计费、毛额计费等 4 个内存规划缺陷，是影响 diffusion offload 行为的关键改动。
- python/sglang/multimodal_gen/test/unit/test_layerwise_offload.py（模块 单元测试；类别 test；类型 test-coverage；符号 _mapped_manager, _MixedBlock, _MixedModel, test_pin_budget_ranks_by_steps_resolved_from_model_index）：为每个缺陷提供回滚即失败的回归测试，并引入映射 / 匿名混合模型基础设施（_MixedBlock/_MixedModel），是本 PR 正确性的保障。

关键符号：_plan_layer_hosting, _layer_byte_totals, anonymous_new_bytes, _keep_weights_on_their_mapping, _mapped_manager, test_pin_budget_ranks_by_steps_resolved_from_model_index, test_only_the_layers_the_budget_covers_are_pinned, test_pins_are_given_back_when_they_do_not_fit_the_host, test_replacing_an_anonymous_original_is_not_charged_as_new

关键源码片段

python/sglang/multimodal_gen/test/unit/test_layerwise_offload.py

为每个缺陷提供回滚即失败的回归测试，并引入映射 / 匿名混合模型基础设施（_MixedBlock/_MixedModel），是本 PR 正确性的保障。

```
def test_pins_are_given_back_when_they_do_not_fit_the_host(tmp_path, monkeypatch):
    if not pathlib.Path("/proc/self/maps").exists():
        # 映射判定依赖 /proc/self/maps，非 Linux 环境跳过
        pytest.skip("needs /proc to tell a mapping from anonymous memory")
    # 四个完全 mapped 的层：预算覆盖两个 pin，但每个 pin 都要把整层
    # 从 mapping 复制进匿名内存；host 只放得下 1.5 层加 reserve，
    # 计划必须归还第二个 pin，而不是分配超出物理内存。
    available = 4 * 1024**3 + int(1.5 * _BLOCK_BYTES)
    manager = _mapped_manager(
        tmp_path,
        monkeypatch,
        available_bytes=available,
        pin_budget_bytes=2 * 1024**3 + 2 * _BLOCK_BYTES,
```

```
        num_blocks=4,
    )
    on_mapping = {i for i in range(4) if manager._mapped_cpu_weights.get(i)}
    # 两个 pin 增加两层匿名内存，但物理内存只有 1.5 层余量，
    # 所以最不值的 pin（layer 1）被归还，只剩 layer 0 保持 pinned。
    assert on_mapping == {1, 2, 3}
    pinned = {i for i in range(4) if manager._consolidated_cpu_weights.get(i)}
    assert pinned == {0}
```

评论区精华

本 PR 没有 review 评论或讨论线程，设计论证集中在 PR body 与提交信息中，值得提炼的要点包括：

- body 明确指出旧逻辑的核心问题：预算按组件一次性申请，'so a DiT larger than the whole spendable budget pinned zero bytes'，同时按层发放后预算会流向价值最高的地方——streamed 层（每步传输一次）优于 resident 层（每 stage 传输一次），stepped DiT 优于 once-per-request encoder。
- 四个缺陷均配有 ' 回滚即失败 ' 的测试：'Four defects found while getting there, each fixed with a test that fails when the fix is reverted'。
- 提交 5179234 记录了重复计费的教训：'Asking about any one tier alone counts the same free bytes twice, and the error only ever says "fits"'。
- merge note 提示本 PR 与 #35882 触碰同一文件，后合入者需 rebase，'the resolution is mechanical — both sides add independent methods'。
- 暂无高价值评论线程

风险与影响

- 风险：
  1. 匿名净成本依赖释放假设：anonymous_new_bytes 认为 fused-qkv 的匿名原始张量在参数 rebind 后即释放。若加载链路中还有其他引用持有该原始存储，实际匿名增量会被低估，导致主机内存超卖；代码以 warning 日志兜底，但没有硬性保护。
  2. 无预算时的行为变更：构造器现在默认创建私有 HostPinBudget()，意味着所有未显式传预算的 offload 组件都会受主机可用内存约束，不再全量 pin。这会影响同一路径上的 Wan、SANA、VAE 等组件，需要回归观察。
  3. 内部 API 移除：_keep_weights_on_their_mapping 被删除，任何外部直接调用都会 AttributeError；虽属内部方法，但合并说明中应声明。
  4. 平台依赖：mapped 判定依赖 /proc/self/maps，测试在非 Linux 环境 skip，映射识别逻辑在其他平台的正确性未被覆盖。
  5. CI 状态：PR 的 Extra PR Test 显示为失败（:x:），合入前需确认是否为已知抖动或与本 PR 相关。 - 影响：影响范围集中在 multimodal_gen 子系统的 layerwise offload 管理器，所有使用 LayerwiseOffloadManager 的 diffusion 组件（DiT、VAE、text encoder）都会走新的按层规划逻辑。对 32 GiB 这类小主机内存场景是显著改进：H3 从「整个 DiT 无法 pin」变为「可 pin 13/50 层」，峰值匿名内存控制在计划 envelope

内。对 SRT 主推理路径无影响。团队侧的收益是内存预算语义更可预测，测试基础设施（多块文件映射模型、混合匿名 / 映射模型）可供后续内存管理 PR 复用。 - 风险标记：主机内存计费口径重构，匿名净成本依赖释放假设，无预算时行为变更，/proc 映射判定平台依赖，Extra CI 未通过

关联脉络

- PR #36034 [Diffusion] UX: clean up startup and offload logs: 改动同一文件 layerwise_offload.py 与 host_memory_budget.py，同属 diffusion offload 与内存预算演进线，后续可对照验证日志与预算语义。
- PR #36051 [diffusion] CI: guard the anonymous-host budget alongside peak VRAM: 同一匿名主机内存预算主题，涉及 host_memory_budget 与流式加载器映射回退，与本 PR 的匿名净成本检查相互印证。
- PR #36000 [diffusion] Keep Cosmos3 Nano resident on high-memory GPUs: 同属 diffusion 层间驻留与 offload 策略，resident layers 决策与本 PR 的 pin 优先级设计直接相关。