PR #32709 完整报告

sgl-project/sglang

[Refactor] Remove dead allocator `backup_state` / `restore_state`

合并时间：2026-07-29 10:49

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/32709

执行摘要

- 一句话：移除分配器中未使用的 backup_state/restore_state 方法
- 推荐动作：值得略读，尤其适合学习如何系统性清理死代码。PR body 提供了清晰的背景说明和验证步骤，可作为类似重构的参考。

功能与动机

Spec V1 的 draft preprocess 通过 alloc_*(backup_state=True) 分配草稿 KV slot，然后调用 restore_state 回滚分配器状态。这是唯一的调用路径，且已在 #25464 和 #27959 中移除。分配器端的实现却遗留了下来，成为了死代码。移除它们可以减少维护成本并避免未来误用。

实现拆解

1. 删除基类方法(mem_cache/allocator/base.py)：移除 BaseTokenToKVPoolAllocator.backup_state() 和 restore_state()，它们只保存和恢复 free_pages / release_pages。
2. 删除 SWA 分配器重写(mem_cache/allocator/swa.py)：移除 SWATokenToKVPoolAllocator 和 PureSWATokenToKVPoolAllocator 中对上述方法的转发重写。
3. 删除多端分配器重写(mem_cache/multi_ended_allocator.py)：移除 MultiEndedAllocator、UnifiedMambaTokenToKVPoolAllocator 和 UnifiedSWATokenToKVPoolAllocator 中的方法实现及其不可达的 relocation 警告日志。
4. 删除 NPU DSV4 分配器重写(hardware_backend/npu/dsv4/dsv4_allocator.py)：移除 DSV4NPUTokenToKVPoolAllocator 中备份 / 恢复多个子分配器的 5 元组版本的 backup_state / restore_state。
5. 清理 allocation 辅助函数(mem_cache/allocation.py)：从 alloc_token_slots 和 alloc_paged_token_slots_extend 中删除 backup_state 参数以及相关的状态保存 / 返回分支，并将返回值简化为单一 out_cache_loc。验证：通过全局 grep 确认 python/ 和 test/ 下无任何 backup_state 或 restore_state 引用残留；签名变更安全，因为调用者均使用关键字传参。

关键文件：

- python/sglang/srt/mem_cache/allocator/base.py（模块 基础分配器；类别 source；类型 core-logic；符号 backup_state, restore_state）：基类分配器，定义了 backup_state 和

restore_state 的原始实现；改动虽仅 6 行，但影响所有继承类。

- python/sglang/srt/mem_cache/allocator/swa.py（模块 SWA 分配器；类别 source；类型 core-logic；符号 backup_state, restore_state）：SWA 分配器包含两个类 (SWATokenToKVPoolAllocator 和 PureSWATokenToKVPoolAllocator) 的重写方法，移除后简化了接口。
- python/sglang/srt/mem_cache/multi_ended_allocator.py（模块 多端分配器；类别 source；类型 core-logic；符号 backup_state, restore_state）：改动量最大的文件（-53 行），涉及三种分配器 (MultiEndedAllocator, UnifiedMambaTokenToKVPoolAllocator, UnifiedSWATokenToKVPoolAllocator)，其中包含 relocation 警告日志也被移除。
- python/sglang/srt/hardware_backend/npu/dsv4/dsv4_allocator.py（模块 NPU DSV4；类别 source；类型 core-logic；符号 backup_state, restore_state）：NPU DSV4 分配器的 backup_state / restore_state 重写涉及 5 个子分配器的多级回滚，清理后减少了特定硬件的维护负担。
- python/sglang/srt/mem_cache/allocation.py（模块 分配辅助；类别 source；类型 core-logic；符号 alloc_token_slots, alloc_paged_token_slots_extend）：入口函数 alloc_token_slots 和 alloc_paged_token_slots_extend 移除了 backup_state 参数和相关分支，简化了调用接口。

关键符号：backup_state, restore_state, alloc_token_slots, alloc_paged_token_slots_extend

关键源码片段

python/sglang/srt/mem_cache/multi_ended_allocator.py

改动量最大的文件（-53 行），涉及三种分配器 (MultiEndedAllocator, UnifiedMambaTokenToKVPoolAllocator, UnifiedSWATokenToKVPoolAllocator)，其中包含 relocation 警告日志也被移除。

```
    def clear(self) -> None:
        """Reset to initial state. Pages in `[0, min_page_index)` are reserved."""
        if self.grow_direction == "up":
            self.watermark_physical = self.min_page_index
        else:
            self.watermark_physical = self.num_pages - 1
        self.virtual_to_physical.fill_(-1)
        self.virtual_to_physical[0] = 0
        self.virtual_to_physical[-1] = -1
        self.physical_to_virtual.fill_(-1)
        self.physical_to_virtual[0] = 0
        self.physical_to_virtual[-1] = -1
        if self.is_id_owner:
            self.free_virtual_ids = torch.arange(
                self.min_page_index, self.num_pages,
                dtype=torch.int64, device=self.device,
            )
        else:
```

```
            self.free_virtual_ids = None
        self.is_not_in_free_group = True
        self.free_group: List[torch.Tensor] = []
        self._inverse_history.clear()
        self._free_phys_pages = torch.empty(0, dtype=torch.int64, device=self.device)
        self._pending_reuse.clear()
        self._pending_reuse_pages_cpu.clear()
        self.live_page_count = 0
        self._inflight_forward = None
        self._latest_forward_done_event = None

    # PR #32709: 删除了原本位于此处的 backup_state() 和 restore_state()
    # 它们曾是 Spec V1 的回滚机制，已无调用者。

    def clear_inverse_history(self) -> None:
        self._inverse_history.clear()
```

python/sglang/srt/hardware_backend/npu/dsv4/dsv4_allocator.py

NPU DSV4 分配器的 backup_state / restore_state 重写涉及 5 个子分配器的多级回滚，清理后减少了特定硬件的维护负担。

```
    # PR #32709: 移除了 backup_state() 和 restore_state() 方法
    # 原本的逻辑是：
    # - backup_state() 返回 (super().backup_state(), c4.backup_state(), c128.backup_state(),
    # c4_state.backup_state(), c128_state.backup_state()) 的 5 元组
    # - restore_state() 分别回滚这 5 个子分配器
    # 现已全部删除，clear() 方法保持不变：

    def clear(self):
        super().clear()
        for attr in (
            "c4_attn_allocator",
            "c128_attn_allocator",
            "c4_state_attn_allocator",
            "c128_state_attn_allocator",
        ):
            allocator = getattr(self, attr, None)
            if allocator is not None:
                allocator.clear()
```

评论区精华

该 PR 没有产生 review 讨论。变更本身简单明确，由作者独立完成并合入。

- 暂无高价值评论线程

风险与影响

- 风险：风险极低：所有被移除的方法和参数均已无任何调用者（通过 grep 验证）。alloc_paged_token_slots_extend 的签名变更（删除第 8 个位置参数）不会影响现有调用，因为所有调用点都通过关键字传参。无行为变化，无性能影响，无 API 破坏。
- 影响：
  - 用户：无影响，此变更仅涉及内部分配器实现。
  - 系统：减少了 121 行死代码，降低了未来维护时的认知负荷。
  - 团队：消除了混淆，避免新开发者误用已废弃的 API。
  - 影响程度：低。
  - 风险标记：死代码清理，无行为变化，低风险

关联脉络

- PR #25464 Remove Spec V1 eagle_worker: 移除了 backup_state/restore_state 的唯一调用者
- PR #27959 Remove DFLASH V1 path: 移除了另一条相关调用路径