PR #41282 完整报告

vllm-project/vllm

[Bugfix] Fix failure to allocate KV blocks error

合并时间：2026-04-30 09:44

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/41282

执行摘要

- 一句话：修复 KV 块 admission cap 误用于逐 step 分配
- 推荐动作：值得精读，尤其关注 admission gate 与 per-step prediction 的差异设计；新增测试可作为类似回归的参考。

功能与动机

PR body 指出失败由 #40946 引入：_max_admission_blocks_per_request 的 min 限值在 get_num_blocks_to_allocate 中无条件应用，但 allocate_new_blocks 中并无此 cap，导致请求过程中逐步分配时预测值偏低，后续实际分配时拉取更多块导致池子不足。需要将 cap 限制仅用于 admission gate 中的全序列检查。

实现拆解

1) 在 SingleTypeKvCacheManager.get_num_blocks_to_allocate 中添加 apply_admission_cap 参数（默认 False），原有无条件 min 限值改为仅当该参数为 True 时生效。 2) 在 KVCacheCoordinator.get_num_blocks_to_allocate 中同样添加该参数，并将其透传给每个底层 manager 的调用。 3) 在 KVCacheManager.can_fit_full_sequence（admission gate）中调用 coordinator 时传入 apply_admission_cap=True，保留 cap 效果；而其他调用点（如 allocate_slots 内的逐步预测）均不设置该参数，保持与 allocate_new_blocks 一致。 4) 新增测试 test_predictor_matches_allocator_blocks_calculation_with_admission_cap，对 SlidingWindowManager 在不同 token 步长下验证预测块数与实际分配块数相等。

关键文件：

- vllm/v1/core/single_type_kv_cache_manager.py（模块 缓存管理器；类别 source；类型 core-logic；符号 get_num_blocks_to_allocate）：核心变更：在 get_num_blocks_to_allocate 方法中添加 apply_admission_cap 参数，控制是否应用 max_admission_blocks_per_request 限制。
- vllm/v1/core/kv_cache_coordinator.py（模块 协调器；类别 source；类型 core-logic；符号 get_num_blocks_to_allocate）：协调层传递 apply_admission_cap 参数到底层 manager。
- vllm/v1/core/kv_cache_manager.py（模块 准入控制；类别 source；类型 core-logic；符号 can_fit_full_sequence）：准入检查 can_fit_full_sequence 中调用 coordinator 时传入

apply_admission_cap=True，使 cap 仅在该路径生效。

- tests/v1/core/test_single_type_kv_cache_manager.py（模块 测试；类别 test；类型 test-coverage；符号 test_predictor_matches_allocator_blocks_calculation_with_admission_cap）：新增测试验证 get_num_blocks_to_allocate 与 allocate_new_blocks 在 admission cap 存在时计算结果一致。

关键符号：SingleTypeKvCacheManager.get_num_blocks_to_allocate, KVCacheCoordinator.get_num_blocks_to_allocate, KVCacheManager.can_fit_full_sequence, test_predictor_matches_allocator_blocks_calculation_with_admission_cap

关键源码片段

vllm/v1/core/single_type_kv_cache_manager.py

核心变更：在 get_num_blocks_to_allocate 方法中添加 apply_admission_cap 参数，控制是否应用 max_admission_blocks_per_request 限制。

```
def get_num_blocks_to_allocate(
    self,
    request_id: str,
    num_tokens: int,
    new_computed_blocks: Sequence[KVCacheBlock],
    total_computed_tokens: int,
    num_tokens_main_model: int,
    apply_admission_cap: bool = False, # 新增参数：默认为 False，控制是否应用 admission cap
) -> int:
    # ... docstring ...
    num_required_blocks = cdiv(num_tokens, self.block_size)
    # 原来无条件 min，现在依赖 apply_admission_cap
    if apply_admission_cap and self._max_admission_blocks_per_request is not None:
        # 仅 admission gate 调用时传入 True，确保 cap 生效
        num_required_blocks = min(
            num_required_blocks, self._max_admission_blocks_per_request
        )
    num_req_blocks = len(self.req_to_blocks.get(request_id, ()))
    if request_id in self.num_cached_block:
        # 快速路径：运行中的请求不会有新的 prefix cache 命中
        assert len(new_computed_blocks) == 0
        return max(num_required_blocks - num_req_blocks, 0)
    # ... 剩余计算（省略）...
```

vllm/v1/core/kv_cache_coordinator.py

协调层传递 apply_admission_cap 参数到底层 manager。

```
def get_num_blocks_to_allocate(
    self,
    request_id: str,
    num_tokens: int,
```

```
    new_computed_blocks: tuple[Sequence[KVCacheBlock], ...],
    num_encoder_tokens: int,
    total_computed_tokens: int,
    num_tokens_main_model: int,
    apply_admission_cap: bool = False, # 新增参数：控制是否应用 cap
) -> int:
    # ... docstring ...
    num_blocks_to_allocate = 0
    for i, manager in enumerate(self.single_type_managers):
        if isinstance(manager, CrossAttentionManager):
            num_blocks_to_allocate += manager.get_num_blocks_to_allocate(
                request_id, num_encoder_tokens, [], 0, num_encoder_tokens,
                apply_admission_cap=apply_admission_cap, # 透传
            )
        else:
            num_blocks_to_allocate += manager.get_num_blocks_to_allocate(
                request_id,
                num_tokens,
                new_computed_blocks[i],
                total_computed_tokens,
                num_tokens_main_model,
                apply_admission_cap=apply_admission_cap, # 透传
            )
    return num_blocks_to_allocate
```

tests/v1/core/test_single_type_kv_cache_manager.py

新增测试验证 get_num_blocks_to_allocate 与 allocate_new_blocks 在 admission cap 存在时计算结果一致。

```
def test_predictor_matches_allocator_blocks_calculation_with_admission_cap():
    """验证在 admission cap 启用时，get_num_blocks_to_allocate 的预测值
    与 allocate_new_blocks 实际分配数一致，避免 pool 抛出 ValueError。
    """
    block_size = 2
    sliding_window = 8 # 4-block live window
    cap = sliding_window // block_size # admission cap = 4

    spec = SlidingWindowSpec(
        block_size=block_size,
        num_kv_heads=1,
        head_size=1,
        dtype=torch.float32,
        sliding_window=sliding_window,
    )
    block_pool = BlockPool(
        num_gpu_blocks=100, enable_caching=True, hash_block_size=block_size
    )
    # 创建带 max_admission_blocks_per_request 的 Manager
    manager = SlidingWindowManager(
```

```
        spec,
        block_pool=block_pool,
        enable_caching=False,
        kv_cache_group_id=0,
        max_admission_blocks_per_request=cap,
    )

    request_id = "req"
    total_computed = 0
    # 模拟多个 forward step，验证每次预测与实际一致
    for num_tokens in (4, 8, 12, 16):
        predicted = manager.get_num_blocks_to_allocate(
            request_id=request_id,
            num_tokens=num_tokens,
            new_computed_blocks=[],
            total_computed_tokens=total_computed,
            num_tokens_main_model=num_tokens,
            # 不传 apply_admission_cap，默认 False，与 per-step 行为一致
        )
        new_blocks = manager.allocate_new_blocks(
            request_id, num_tokens=num_tokens, num_tokens_main_model=num_tokens
        )
        assert predicted == len(new_blocks), (
            f"num_tokens={num_tokens}: predictor returned {predicted} "
            f"but allocator pulled {len(new_blocks)}"
        )
        total_computed = num_tokens
```

评论区精华

PR 获得两位 reviewer (njhill, Dao007forever) approve。Dao007forever 评论变量命名不够好但未提出具体修改，作者表示欢迎建议。此外 gemini-code-assist 自动化 review 摘要了变更内容。

- 变量命名建议 (style): 未达成一致，已 resolve

风险与影响

- 风险：风险较低：变更局限于 KV cache 管理内部，引入了新参数但默认行为不变（apply_admission_cap=False），仅 admission gate 主动开启；测试覆盖了回归场景。但需要注意所有调用 get_num_blocks_to_allocate 的地方是否都正确地未传递 apply_admission_cap=True（除 admission gate 外）。当前代码库中除 can_fit_full_sequence 外还有多处调用，需检查防止遗漏。
- 影响：影响 DeepSeek V4 及所有使用 recycling-aware attention spec（SWA、chunked-local）的用户，修复了服务 OOM 崩溃。对不使用 admission cap 的模型无影响。
- 风险标记：admission cap 依赖，多调用点需审查

关联脉络

- PR #40946 引入 admission cap 的 PR: 该 PR 引入了 _max_admission_blocks_per_request 的无条件应用，此 PR 修复其引入的 bug。