

PR #57050 完整报告

vllm-project/vllm

[Bugfix] Fix incorrect Mamba block allocation estimate that prevents request admission

合并时间: 2026-09-16 15:23

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/57050>

执行摘要

- 一句话: 修正 Mamba 准入块估算, 避免长请求永久卡在等待队列
- 推荐动作: 值得精读, 重点不在 diff 规模 (源码仅 4 行), 而在于三个设计决策: (1) 「上限截断」与「条件覆盖」的语义差异, 以及为什么 `min()` 才能同时避免高估导致的准入饥饿和低估计的块池超发; (2) `apply_admission_cap` 从「影响数值」退化为「只影响副作用」, 使准入与分配共用同一估算口径; (3) 新增测试把「估算等于实测」写成断言, 是排查同类 block accounting 问题的通用模板。建议阅读顺序: 先看 PR body 的复现场景与 Stall 描述, 再看 `MambaManager.get_num_blocks_to_allocate` 尾部的构造与 `min()`, 最后看新测试如何先把请求推进 `_allocated_block_reqs` 再制造 checkpoint 场景。后续可跟进: 对 partial hit + checkpoint 组合、`num_prefill_checkpoint_blocks > 1` 与不同 `hash_block_size` 补充参数化覆盖, 并在 review 中确认 #53614 是否为确定回归来源。

功能与动机

PR body 明确给出了线上症状与复现路径: Kimi K3 在启用外部 KV 存储 (Mooncake) 加载前缀缓存时, 长请求会被拒绝准入并「get stuck in the waiting queue forever, causing entire system to stall」。作者定位到问题出在 Mamba 的块分配估算函数 `get_num_blocks_to_allocate`, 并指出这「potentially a regression caused by <https://github.com/vllm-project/vllm/pull/53614>」。复现命令为 8 x B300 上以 `--max-model-len 1048576`、`--enable-prefix-caching`、MooncakeStoreConnector 启动服务, 发送 1M 长度请求后清空本地 GPU KV cache 再重发同一请求。

实现拆解

1. 变更入口: `vllm/v1/core/single_type_kv_cache_manager.py` 的 `MambaManager.get_num_blocks_to_allocate`。该函数同时服务两个调用方——`apply_admission_cap=True` 时是调度器准入闸门的 dry-run 估算 (不写 `self._checkpoints`), `apply_admission_cap=False` 时是真实分配前的估算 (会更新 `self._checkpoints`)。两者共用同一段块数压缩逻辑, 因此估算偏差会直接决定请求能否被准入。
2. 定位失配分支: 旧代码在 `num_new_blocks > 0` 时执行 `if not (checkpoint_block and blocks_allocated): num_new_blocks = 1 + int(has_partial_hit) + checkpoint_block; if not blocks_allocated: num_new_blocks += self.num_speculative_blocks`。当请求已经持有块 (`blocks_allocated=True`) 且本次需要写 checkpoint (`checkpoint_block=1`) 时,

外层条件为假，整段压缩被跳过，`num_new_blocks` 保留按 token 数量推导出的原始值（对 1M 请求可达数千），远超实际需要的常量级物理块，于是准入判定「块不够」并永久拒绝——请求无法推进，也就永远不会减少需求。

3. 改为上限截断：新代码先构造 `physical_block_cap = 1 + int(has_partial_hit) + checkpoint_block`，再在 `not blocks_allocated or checkpoint_block` 时叠加 `self.num_speculative_blocks`（MTP/EAGLE 草稿状态预留），最后 `num_new_blocks = min(num_new_blocks, physical_block_cap)`。语义从「无条件覆盖」变成「上界约束」：既堵住 checkpoint + 已分配这条漏网路径，也避免原始估算更小时被反向抬高。checkpoint 分支现在统一计入 speculative blocks，与 `allocate_new_blocks` 的 `num_required_blocks = cdiv(num_tokens, block_size) + num_speculative_blocks` 口径对齐。
4. 保持 dry-run 副作用隔离：`apply_admission_cap` 仍然只控制是否写 `self._checkpoints`（准入估算不得污染状态），但不再影响数值计算，从而保证「准入估算」与「分配估算」在数值上恒等。
5. 测试配套：`tests/v1/core/test_single_type_kv_cache_manager.py` 新增 `test_mamba_checkpoint_admission_matches_allocation`，用 `MambaSpec(num_prefill_checkpoint_blocks=1, prefill_checkpoint_alignment=64, mamba_cache_mode="align")` 与 `BlockPool(num_gpu_blocks=2048, hash_block_size=128)` 构造场景，先做一轮 `computed_tokens=23040` 的真实分配让请求进入 `_allocated_block_reqs`，再对比 `admission_estimate`（`apply_admission_cap=True`）、`allocation_estimate`（`False`）与 `allocate_new_blocks` 后的实测 `free_before - free_after`，参数化为 `block_size ∈ {896, 1536} × num_speculative_blocks ∈ {0, 1, 4} × prompt_tokens ∈ {25121, 704547}` 共 12 组。PR body 说明该测试在 main 上失败、在本分支通过。
6. 影响边界：改动只在 `mamba_cache_mode == "align"` 且 `num_new_blocks > 0` 时生效，不影响 full attention / sliding window / chunked local attention 等其他 manager，也不改变非 align 模式（走 `super().allocate_new_blocks` 并叠加 speculative 块）的行为。

关键文件：

- `vllm/v1/core/single_type_kv_cache_manager.py`（模块 块分配器；类别 source；类型 core-logic；符号 `MambaManager.get_num_blocks_to_allocate`, `MambaManager.allocate_new_blocks`）：核心修复点：
`MambaManager.get_num_blocks_to_allocate` 在 align 模式下把「条件覆盖式」块估算改为 `physical_block_cap` 上界 + `min()` 截断，直接决定长请求能否通过准入闸门。
- `tests/v1/core/test_single_type_kv_cache_manager.py`（模块 单元测试；类别 test；类型 test-coverage；符号 `test_mamba_checkpoint_admission_matches_allocation`, `estimate`）：新增回归测试 `test_mamba_checkpoint_admission_matches_allocation`，用 12 组参数断言准入估算、分配估算与实测分配块数三者相等，把「估算与分配一致」固化为可执行的不变量。

关键符号：`MambaManager.get_num_blocks_to_allocate`,
`MambaManager.allocate_new_blocks`, `MambaManager._needs_internal_checkpoint`,
`MambaManager._has_partial_local_hit`, `test_mamba_checkpoint_admission_matches_allo`

cation, estimate

关键源码片段

vllm/v1/core/single_type_kv_cache_manager.py

核心修复点: MambaManager.get_num_blocks_to_allocate 在 align 模式下把「条件覆盖式」块估算改为 physical_block_cap 上界 + min() 截断, 直接决定长请求能否通过准入闸门。

```
# vllm/v1/core/single_type_kv_cache_manager.py
# MambaManager.get_num_blocks_to_allocate —— align 模式分支 (PR#57050 修复后的形态)
```

```
has_partial_hit = (
    self._has_partial_local_hit(new_computed_blocks, num_local_computed_tokens)
    or request_id in self._partial_hit_reqs
)
```

```
if has_partial_hit:
    # 前缀只被部分命中时, 额外保留 1 个块承载未命中的 state
    num_new_blocks = max(num_new_blocks, 0) + 1
```

```
checkpoint_position = get_mamba_prefill_checkpoint_position(
    num_tokens,
    self.block_pool.hash_block_size,
    self.drop_eagle_checkpoint_block,
)
```

```
if not self._needs_internal_checkpoint(
    request_id,
    total_computed_tokens,
    num_tokens,
    checkpoint_position,
):
```

```
    checkpoint_position = 0
```

```
checkpoint_block = int(checkpoint_position > 0)
```

```
if not apply_admission_cap:
```

```
    # 只有真正的分配路径才写 _checkpoints; 准入侧的 dry-run 不产生副作用,
```

```
    # 但仍复用同一套数值逻辑, 保证「准入估算 == 分配估算」
```

```
    if checkpoint_position > 0:
```

```
        checkpoint_idx = cdiv(num_tokens, self.block_size) - 2
```

```
        self._checkpoints[request_id] = (checkpoint_position, checkpoint_idx)
```

```
    else:
```

```
        self._checkpoints.pop(request_id, None)
```

```
if num_new_blocks > 0:
```

```
    blocks_allocated = request_id in self._allocated_block_reqs
```

```
    # 物理块上限: 当前状态块 + partial hit 块 + 本次 checkpoint 块
```

```
    physical_block_cap = 1 + int(has_partial_hit) + checkpoint_block
```

```
    # 首次分配, 或本次要新建 checkpoint 时, 还要为 MTP/EAGLE 的草稿状态预留块;
```

```
    # 旧实现正是在 blocks_allocated and checkpoint_block 时跳过了这段调整,
```

```
    # 使得 num_new_blocks 保持按 token 数线性增长的原始值 (1M 请求可达数千块),
```

```

# 结果被准入闸门永久拒绝、请求堵死 in waiting 队列
if not blocks_allocated or checkpoint_block:
    physical_block_cap += self.num_speculative_blocks
# 关键修复：用 min() 做上界约束而不是条件覆盖，
# 既不会高估（避免准入饥饿），也不会把更小的原始估算反向抬高
num_new_blocks = min(num_new_blocks, physical_block_cap)

num_evictable_computed_blocks = self._get_num_evictable_blocks(new_computed_blocks)
return num_new_blocks + num_evictable_computed_blocks

# 参考口径：allocate_new_blocks 在 align 模式下的实际需求（注释说明其本身是 over-estimate）
# num_required_blocks = cdiv(num_tokens, self.block_size) + self.num_speculative_blocks

```

tests/v1/core/test_single_type_kv_cache_manager.py

新增回归测试 test_mamba_checkpoint_admission_matches_allocation，用 12 组参数断言准入估算、分配估算与实测分配块数三者相等，把「估算与分配一致」固化为可执行的不变量。

```

# tests/v1/core/test_single_type_kv_cache_manager.py
@pytest.mark.parametrize("block_size", [896, 1536])
@pytest.mark.parametrize("num_speculative_blocks", [0, 1, 4])
@pytest.mark.parametrize("prompt_tokens", [25121, 704547])
def test_mamba_checkpoint_admission_matches_allocation(
    block_size, num_speculative_blocks, prompt_tokens
):
    """Checkpoint admission must match the subsequent physical allocation."""
    spec = MambaSpec(
        block_size=block_size,
        shapes=((1, 1),),
        dtypes=(torch.float32,),
        mamba_cache_mode="align",
        num_speculative_blocks=num_speculative_blocks,
        num_prefill_checkpoint_blocks=1,
        prefill_checkpoint_alignment=64,
    )
    pool = BlockPool(num_gpu_blocks=2048, enable_caching=True, hash_block_size=128)
    manager = MambaManager(
        spec,
        block_pool=pool,
        enable_caching=True,
        kv_cache_group_id=0,
        scheduler_block_size=block_size,
    )
    request_id = "prefill"
    computed_tokens = 23040

    def estimate(num_tokens, total_computed_tokens, apply_admission_cap):
        # 同一个函数同时扮演准入估算与分配估算两个角色，
        # apply_admission_cap=True 表示 dry-run（不写 _checkpoints）

```

```

return manager.get_num_blocks_to_allocate(
    request_id=request_id,
    num_tokens=num_tokens,
    new_computed_blocks=[],
    total_computed_tokens=total_computed_tokens,
    num_local_computed_tokens=total_computed_tokens,
    num_tokens_main_model=num_tokens,
    apply_admission_cap=apply_admission_cap,
)

# 先走一轮真实分配，让请求进入 _allocated_block_reqs —— 复现 blocks_allocated=True
estimate(computed_tokens, 0, False)
manager.allocate_new_blocks(request_id, computed_tokens, computed_tokens)
assert request_id in manager._allocated_block_reqs

# 此时请求已持有块且需要新建 checkpoint，正是旧实现跳过块数压缩的分支
admission_estimate = estimate(prompt_tokens, computed_tokens, True)
allocation_estimate = estimate(prompt_tokens, computed_tokens, False)
assert request_id in manager._checkpoints

free_before = pool.get_num_free_blocks()
manager.allocate_new_blocks(request_id, prompt_tokens, prompt_tokens)
allocated = free_before - pool.get_num_free_blocks()

# 不变量：准入估算 == 分配估算 == 真实分配的块数
assert admission_estimate == allocation_estimate == allocated

```

评论区精华

本次 PR 的 review 讨论非常轻量，没有行内 review 评论（review_comments_count = 0），也没有针对 `min()` 语义变更的技术辩论：

- ZeldaHuang（KV cache manager 方向 maintainer）APPROVED，只留下一句 "Thanks for the catch!"，等于确认了作者对失配分支的定位。
- ywang96 APPROVED（无正文），随后由作者合并。
- ZJY0516 在 issue 评论中触发 CI: `"/ci run"`，随后 github-actions[bot] 回报 "🔗 Triggered Buildkite CI #89268 for commit 5f7c64c608a7"。
- claude[bot] 的自动审查因 PR 来自 fork 被禁用: "This pull request is from a fork — automated review is disabled. A repository maintainer can comment @claude review to run a one-time review."，实际上没有人再触发。

因此，虽然修复本身涉及准入路径的正确性，但缺少对「估算函数是否应作为准入与分配唯一真源」「其他 manager 是否存在同类覆盖式估算」等问题的公开讨论，结论主要依赖作者自述的复现与单测。

- Mamba align 模式准入估算：条件覆盖 vs 上界截断 (correctness): ZeldaHuang 与 ywang96 先后 APPROVED 后合并，PR body 说明新增测试在 main 上失败、在本分支通过。

- 回归来源 #53614 是否确定 (question): 维持「疑似」表述, 未在本次 PR 中定论; 修复本身不依赖该结论。
- 回归测试与 CI 验证 (testing): 测试与 CI 均通过, 随 PR 合入。
- fork PR 无法运行自动化审查 (other): 该 PR 的静态审查仅依赖两位人工 approve, 没有额外自动检查记录。

风险与影响

• 风险:

1. 准入路径的正确性风险已被收敛, 但语义变化需要留意: `num_new_blocks` 从「条件覆盖为常量」变为「`min(raw, cap)`」。当 `raw` 估算小于 `cap` 时不再被抬高, 理论上存在低估物理需求的可能, 从而让准入放行、分配阶段过量占用块池。不过从 `allocate_new_blocks` 的注释 ("this is an over-estimate of how many blocks we need") 与新增测试的等值断言看, `align` 模式的真实分配确实被同一 `cap` 约束, 风险主要体现在未覆盖的参数组合上。
1. 测试参数空间有限: 新增测试固定 `hash_block_size=128`、`num_prefill_checkpoint_blocks=1`、`prefill_checkpoint_alignment=64`、`computed_tokens=23040`, 并假设「无 partial hit」(`new_computed_blocks=[]`)。`has_partial_hit` 分支 (会额外 +1 块)、`num_prefill_checkpoint_blocks > 1`、`drop_eagle_checkpoint_block=True`、以及不同 `hash_block_size` 的组合没有回归覆盖。
2. 测试耦合私有状态: 断言直接读取 `manager._allocated_block_reqs` 与 `manager._checkpoints`, 一旦 `MambaManager` 内部记账结构重命名或迁走, 测试会失败而非静默误解, 但也会增加后续重构成本。
3. 回归来源未定论: PR body 指向 #53614 可能是回归引入点, 但 review 中没有 bisect 或版本验证结论, 若根因还包含其他路径 (例如外部分页 KV 的加载顺序), 当前 4 行修复可能只覆盖了主要分支。
4. 缺少 fork PR 的自动审查: `claude[bot]` 未运行, 静态检查维度仅靠人工 approve, 没有额外的边界条件审查记录。
5. 线上表现形态变化: 修复后长请求不再被拒绝, 改为正常的块分配与可能的抢占 (preemption) / recompute 路径, 因此仍需关注极端长上下文 + 外部 KV 加载时的块池瞬时压力。- 影响: 对用户: 直接修复 Mamba 混合架构模型 (Kimi K3 等) 在开启 prefix caching 且从外部 KV 存储 (Mooncake) 加载缓存时, 长请求 (百万 token 级) 触发整机 stall 的严重问题; 此前该场景下请求不可恢复, 现在可正常准入并继续推理。

对系统: `get_num_blocks_to_allocate` 是调度器准入闸门的预算来源, 修复后「准入估算 == 分配估算 == 实际分配块数」成为显式不变量, 降低了准入与分配脱节引发的排队饥饿风险; 同时把 risky 的「覆盖式常量估算」替换为更易推理的「上界约束」。

对团队: 改动只有 4 行源码, 但影响的是 v1 KV cache manager 的核心记账逻辑, 属于「小 diff、大后果」的典型修复, 值得作为其他单类型 manager (如 `SlidingWindowManager` 的 `max_admission_blocks_per_request`) 一致性测试的范式参考。

- 风险标记: 核心准入路径变更, 估算与分配一致性依赖单一函数, 测试参数空间有限 (固定 `hash_block_size` / 单 checkpoint), 回归来源未定论, fork PR 缺少自动审查

关联脉络

- PR #53614 PR#53614: Mamba align/checkpoint 相关改动（标题未在上下文中提供）：
PR body 直接指出本次 bug 「potentially a regression caused by」该 PR，是理解失配分支引入时的关键背景。
- PR #51787 [Bugfix][KV Offload] Track cache recency once per request: 同属 v1 KV cache manager/kv_offload 侧的记账正确性修复，并且都以 manager 级单测固化「内存核算与淘汰 / 分配顺序」的不变量，脉络相近。
- PR #55055 [BugFix][Model Runner V2][Spec Decode] Fix decode instance's multi-layer MTP kv caches during P/D: 同样涉及 speculative/MTP 草稿状态与 KV cache 记账的交互，本 PR 的 physical_block_cap 也要为 num_speculative_blocks 预留块，两者在「spec 块如何计入缓存预算」上互相印证。
- PR #54014 [Bugfix][KV Offload] Check cgroup memory before SHM allocation: 同属「分配前预检 / 容量核算」主题，反映近期 vLLM 在 KV cache 与内存分配路径上系统性补充容量安全网的趋势。