PR #47272 完整报告

vllm-project/vllm

[Bugfix][Core] Reserve the KV null block when validating max_model_len

合并时间：2026-08-20 10:49

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/47272

执行摘要

- 一句话：KV 校验预留 null block，修复 max_model_len 边界启动后挂起
- 推荐动作：值得精读。核心价值在于 " 容量校验必须与分配约束（BlockPool null block 预留）保持一致 " 的设计原则，以及把边界 bug 从 override 单一路径推广到所有来源的正确做法——在 available_memory 折算点做单点修复，避免多路径各自打补丁。review 中 njhill 关于把扣减提升到 auto-fit 之前的讨论、malaiwah 的混合 Mamba 复现矩阵与 #52530 的分工，都值得参考。

功能与动机

Issue #35541 报告 vLLM 在 num_gpu_blocks_override 过低时无限挂起；#41069 曾让启动容量校验考虑 override，但按总 block 数比较，而 BlockPool 会永久保留 1 个 null block（get_usage() 已扣减），导致可用块数实际为 num_gpu_blocks - 1。当 KV cache 恰好等于 ceil(max_model_len / block_size) 时，启动校验通过，运行时却差一个 block：请求被反复抢占、num_computed_tokens 重置后无限重试，running == 0 且 waiting >= 1，表现为 0 tok/s。本 PR 把该边界从 override 单一路径推广到所有 block 来源，并让 auto-fit 与校验口径一致。

实现拆解

1. 统一容量折算（核心）：在 vllm/v1/core/kv_cache_utils.py 的 get_kv_cache_configs 中，先沿用 #41069 的逻辑处理 num_gpu_blocks_override 对 available_memory 的折算，随后新增 check_memory 列表：对每个 worker，若存在 KV cache group，则从可用内存中减去 _pool_bytes_per_block 计算的单个 block 字节数（null block 预留）。_auto_fit_max_model_len 与逐 worker 的 _check_enough_kv_cache_memory 都改基于 check_memory 判定；后续 get_kv_cache_config_from_groups 仍用完整 available_memory 做实际分配，与 BlockPool 全量分配、使用量扣减 null block 的行为保持一致。
2. 公共校验函数同步：check_enough_kv_cache_memory 增加同样的扣减逻辑——先通过 get_kv_cache_groups 对 kv_cache_spec 的副本分组（分组可能 in-place 合并 spec），存在 group 时用 available_memory - _pool_bytes_per_block 作为检查基数，避免公共入口与内部路径口径不一致。
3. 测试配套：tests/v1/core/test_kv_cache_utils.py 新增三个测试：参数化 test_kv_cache_reserves_null_block_for_max_model_len（use_override=True/False 分

别覆盖 override 与内存推导路径，32 个 block 配 max_model_len=512 必须抛ValueError）、test_auto_fit_max_model_len_reserves_null_block（64 个 block 内存下auto-fit 收敛到 63 × block_size）、test_check_enough_kv_cache_memory_reserves_null_block（公共检查函数拒绝精确边界）。

4. 既有测试与 CI 修正：CI 暴露三个既有测试恰好配置在边界：tests/v1/e2e/general/test_async_scheduling.py 与 tests/v1/sample/test_logprobs.py 的 num_gpu_blocks_override 由 32 改为 33（补上 null block），tests/v1/e2e/general/test_context_length.py 的 kv_cache_bytes 由 1 MB 改为 2 MB（保证 pool 至少 2 个 block：1 个 null block + 1 个可用）；tests/v1/engine/test_init_error_messaging.py 将 dtype 从字符串 float16 调整为 torch.float16 以适配新类型校验。

关键文件：

- vllm/v1/core/kv_cache_utils.py（模块 缓存校验；类别 source；类型 core-logic；符号 check_enough_kv_cache_memory, get_kv_cache_configs）：核心修复文件：get_kv_cache_configs 新增 check_memory 折算，auto-fit 与逐 worker 容量校验统一按可用 block 规划；check_enough_kv_cache_memory 公共入口同步扣减 null block。
- tests/v1/core/test_kv_cache_utils.py（模块 缓存校验；类别 test；类型 test-coverage；符号 test_kv_cache_reserves_null_block_for_max_model_len, test_auto_fit_max_model_len_reserves_null_block, test_check_enough_kv_cache_memory_reserves_null_block）：新增三个边界测试，分别覆盖 override 与内存推导双路径的拒绝行为、auto-fit 收敛到可用 block 数、公共检查函数同步扣减。
- tests/v1/e2e/general/test_async_scheduling.py（模块 调度器；类别 test；类型 test-coverage）：CI 失败根因之一：num_gpu_blocks_override 从 32 改为 33，补上 null block 占用的一个块，同时保持强制抢占语义。
- tests/v1/e2e/general/test_context_length.py（模块 长度校验；类别 test；类型 test-coverage）：CI 失败根因之一：kv_cache_bytes 从 1 MB 上调到 2 MB，保证 pool 至少 2 个 block（1 个 null block + 1 个可用），auto-fit 测试才能正常工作。
- tests/v1/sample/test_logprobs.py（模块 采样输出；类别 test；类型 test-coverage）：CI 失败根因之一：num_gpu_blocks_override 从 32 改为 33，避免强制抢占测试在新校验下启动失败。
- tests/v1/engine/test_init_error_messaging.py（模块 初始化报错；类别 test；类型 test-coverage）：check_enough_kv_cache_memory 公共入口同步预留逻辑后，测试的 dtype 从字符串 float16 调整为 torch.float16，验证错误消息测试仍通过。

关键符号：check_enough_kv_cache_memory, get_kv_cache_configs, test_kv_cache_reserves_null_block_for_max_model_len, test_auto_fit_max_model_len_reserves_null_block, test_check_enough_kv_cache_memory_reserves_null_block

关键源码片段

vllm/v1/core/kv_cache_utils.py

核心修复文件：get_kv_cache_configs 新增 check_memory 折算，auto-fit 与逐 worker 容量校验统一按可用 block 规划；check_enough_kv_cache_memory 公共入口同步扣减 null block。

```
def check_enough_kv_cache_memory(
    vllm_config: VllmConfig,
    kv_cache_spec: dict[str, KVCacheSpec],
    available_memory: int,
):
    """检查 available_memory 是否足以让 KV cache 容纳至少一个
    max_model_len 长度的请求，不够时抛 ValueError。

    注意 BlockPool 会永久保留 1 个 null block，因此实际可用块数为
    总数减 1；这里先把可校验内存扣掉一个 block 的字节，再交给
    _check_enough_kv_cache_memory 判定，避免边界配置
    启动通过、运行时却差一块而挂起。
    """
    # 空 spec 无需检查
    if kv_cache_spec:
        # 先分组取一份拷贝：分组过程可能 in-place 合并 spec
        groups = get_kv_cache_groups(vllm_config, dict(kv_cache_spec))
        check_memory = (
            available_memory - _pool_bytes_per_block(vllm_config, groups)
            if groups
            else available_memory
        )
        _check_enough_kv_cache_memory(
            check_memory,
            lambda: max_memory_usage_bytes(vllm_config, kv_cache_spec.values()),
            vllm_config.model_config.max_model_len,
            lambda am: estimate_max_model_len(vllm_config, kv_cache_spec, am),
        )

    # num_gpu_blocks_override 会把实际分配块数钳制到 override 值，
    # 这里先把 available_memory 折算成 override 对应的字节数，
    # 让 auto-fit、容量校验与 per-worker 配置构建看到同一容量。
    override = vllm_config.cache_config.num_gpu_blocks_override
    if override is not None:
        adjusted_memory = []
        for groups, avail_mem in zip(projected_groups_per_worker, available_memory):
            if not groups:
                adjusted_memory.append(avail_mem)
                continue
            bytes_per_block = _pool_bytes_per_block(vllm_config, groups)
            adjusted_memory.append(override * bytes_per_block)
        available_memory = adjusted_memory

    # 扣掉 null block：auto-fit 与 admission check 都按可用块规划，
```

```
    # 而真正的分配仍使用完整内存，与 BlockPool 行为保持一致。
    check_memory = [
        avail_mem - _pool_bytes_per_block(vllm_config, groups) if groups else avail_mem
        for groups, avail_mem in zip(projected_groups_per_worker, available_memory)
    ]

    if vllm_config.model_config.original_max_model_len == -1:
        _auto_fit_max_model_len(vllm_config, projected_groups_per_worker, check_memory)

    # 逐 worker 校验容量是否足够；不足时抛 ValueError，
    # 报错信息包含 estimated maximum model length 提示。
    for groups, avail_mem in zip(projected_groups_per_worker, check_memory):
        if not groups:
            continue
        _check_enough_kv_cache_memory(
            avail_mem,
            partial(_max_memory_usage_bytes_from_groups, vllm_config, groups),
            vllm_config.model_config.max_model_len,
            partial(_estimate_max_model_len_from_groups, vllm_config, groups),
        )
```

tests/v1/core/test_kv_cache_utils.py

新增三个边界测试，分别覆盖 override 与内存推导双路径的拒绝行为、auto-fit 收敛到可用 block 数、公共检查函数同步扣减。

```
@pytest.mark.parametrize('use_override', [True, False])
def test_kv_cache_reserves_null_block_for_max_model_len(use_override):
    """KV cache 恰好只够一个 max_model_len 请求时，启动必须拒绝：
    BlockPool 预留 1 个 null block 后，可用块数少 1，放行会导致
    请求永远无法调度、引擎 0 tok/s 挂起。
    参数化同时覆盖 num_gpu_blocks_override 与内存推导两条路径。
    """
    block_size = 16
    max_model_len = 512 # 需要 512 / 16 = 32 个 block
    vllm_config = VllmConfig(model_config=ModelConfig(max_model_len=max_model_len))
    spec = new_kv_cache_spec(block_size=block_size)

    # 32 个 block 扣掉 null block 后只剩 31 个可用：差一个就拒绝
    if use_override:
        # 内存充足，仅靠 override 把块数钳到 32
        vllm_config.cache_config.num_gpu_blocks_override = 32
        available_memory = [spec.page_size_bytes * 1024]
    else:
        available_memory = [spec.page_size_bytes * 32]

    with pytest.raises(ValueError, match='max seq len'):
        get_kv_cache_configs(vllm_config, [{'layer1': spec}], available_memory)

def test_auto_fit_max_model_len_reserves_null_block():
```

```
    """auto-fit（max_model_len=-1）必须按可用块数规划：
    内存只够 64 个 block 时，1 个是 null block，只能收敛到
    63 * block_size；若按全池 64 规划，满长请求会差一块而挂起。
    """
    block_size = 16
    model_config = ModelConfig(max_model_len=1024)
    model_config.original_max_model_len = -1
    vllm_config = VllmConfig(model_config=model_config)
    spec = new_kv_cache_spec(block_size=block_size)

    # 恰好是 1024 / 16 = 64 个 block 的内存
    available_memory = [spec.page_size_bytes * 64]

    get_kv_cache_configs(vllm_config, [{'layer1': spec}], available_memory)

    assert vllm_config.model_config.max_model_len == 63 * block_size
```

评论区精华

Auto-fit 路径的边界漏洞（njhill 提出，已解决）：njhill 在 review 中指出 _auto_fit_max_model_len 对完整 block 数做二分搜索，max_model_len=-1 时可能选中恰好需要 null block 的长度；建议把扣减提升到 auto-fit 分支之前，去掉 check_memory 分裂。92hyungjun 采纳并新增第三个测试，CPU 端到端验证 auto-fit 从 800 修正为 784（= 49 可用 block × 16）。

混合 Mamba 模型与 #52530 分工（malaiwah 提出）：malaiwah 在 issue 评论区报告，混合 Mamba（GDN linear attention + full attention、--mamba-cache-mode align、MTP 3 个 speculative blocks）上可复现边界，且上方还有一行本 PR 捕不到（mamba-align admission timing、lookahead slots）。他声明已开 #52530 处理 request 侧，双方确认本 PR 只负责 capacity 侧，保持范围小。

CI 既有测试踩边界（92hyungjun 说明）：6 个失败 job 的根因是三个既有测试把 pool 恰好在边界配置（32 块、1 MB），请求远低于 max_model_len 所以从未触发挂起；修复是给 pool 补上 null block 占用的 1 个 block。

- Auto-fit 路径未预留 null block 的边界漏洞 (design): 预留扣减提升到 auto-fit 之前，新增 auto-fit 边界测试；相关既有测试 test_auto_fit_max_model_len_with_hybrid 相应多要一个 block。
- 混合 Mamba 模型上边界上一行的残余风险与 #52530 分工 (correctness): 分工明确：本 PR 负责 capacity 侧（启动容量校验），#52530 负责 request 侧；92hyungjun 表示保持本 PR 范围小，auto-fit 已覆盖无需再扩。
- CI 既有测试恰好在边界上 (testing): 给每个 pool 补上 null block 占用的 1 个 block: test_async_scheduling.py 与 test_logprobs.py 的 override 32→33, test_context_length.py 的 kv_cache_bytes 1→2 MB。
- check_enough_kv_cache_memory 同步预留逻辑 (design): 合并者直接修改并 approve，测试同步调整 dtype 为 torch.float16。

风险与影响

- 风险：
  - 启动行为变更：精确边界配置（如 num_gpu_blocks_override 恰好等于 ceil(max_model_len/block_size)）从 " 能启动但挂起 " 变为 " 启动即抛 ValueError"，这是有意修正，但依赖旧行为的用户会看到新报错。
  - auto-fit 推导值变化：max_model_len='auto' 时结果可能比之前小一个 block（如 800 → 784），对依赖旧推导值的用户可见，这是保证可调度性的必要代价。
  - 与 BlockPool 内部实现耦合：kv_cache_utils.py 硬编码 " 预留 1 个 block" 的假设，若 BlockPool 预留策略变化需同步；测试中的 32→33、1→2 MB 也依赖该假设。
  - 默认 profiled 路径测试缺口：PR body 承认自动 profiling 命中边界 "uncommon"，新增测试只覆盖 override 与 memory bytes 两条路径，profiled 默认路径没有直接测试。
  - 公共函数依赖分组正确性：check_enough_kv_cache_memory 依赖 get_kv_cache_groups 正确推断 group，混合模型（如 Mamba hybrid）下的分组行为需留意。
- 影响：
  - 用户侧：低内存或精确 override 配置不再无限挂起，启动报错信息包含 "estimated maximum model length"（如 784），可操作性显著提升。
  - 系统侧：三条 KV block 来源路径（override、kv_cache_memory_bytes、自动 profiling）统一按可用 block 核算，消除校验与运行时分配不一致；auto-fit 与 admission check 共用同一容量口径。
  - 团队侧：为 #35541 画上句号，并与 #52530 形成 capacity/request 两侧的清晰分工，后续 mamba-align admission timing 与 lookahead slots 问题单独跟进。
  - 风险标记：核心路径容量校验变更，启动行为变更（边界改报错），auto-fit 推导值可能缩小 1 个 block，边界测试硬编码依赖 null block 预留，默认 profiled 路径缺少直接测试覆盖

关联脉络

- PR #41069 [Bugfix] Account for num_gpu_blocks_override in startup KV cache check: PR body 明确引用其为前序已合并修复：让启动校验考虑 num_gpu_blocks_override，但按总 block 数比较且只覆盖 override 路径，边界仍挂起；本 PR 在此基础上把 null block 预留推广到全部来源。标题系按描述整理，材料未给出确切标题。
- PR #35542 [Bugfix] Add override-only startup KV cache check (unmerged attempt): PR body 提及的早期未合并尝试，只加 override 专属检查；本 PR 改为在共享容量上扣减 null block，覆盖范围更全且不引入重复检查。标题系按描述整理，材料未给出确切标题。
- PR #52530 [Bugfix] Request-side fix for the remaining boundary band (opened by malaiwah): malaiwah 在 issue 评论区说明与本 PR 分工：本 PR 负责 capacity 侧，52530 负责 request 侧（mamba-align admission timing、lookahead slots），两者互补。标题未在材料中给出。