PR #50302 完整报告

vllm-project/vllm

[Bugfix] Universally align block table width to 128 tokens

合并时间：2026-07-31 23:52

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/50302

执行摘要

- 一句话：统一 block table 宽度计算，修复 DSA 稀疏索引器崩溃
- 推荐动作：值得精读。核心价值在于用一个纯函数收敛 block table 宽度的计算契约，并通过 requires_block_table_width 类属性让 metadata builder 按需声明依赖，是典型的 ' 单一事实来源 ' 设计。建议阅读 vllm/v1/worker/block_table.py 的 get_block_table_width 与 vllm/v1/worker/utils.py 的 create_metadata_builders，并结合后续 PR #50823 理解 DCP 分片对宽度计算的影响。

功能与动机

Issue #46074 报告 GLM-5.2（DSA sparse MLA）+ fp8_ds_mla 在 max_model_len >= 325K 并发解码时崩溃，错误为 expanded_block_table_buffer 尺寸差一（5469 vs 5470），且随 max_model_len 放大，说明索引器 buffer 宽度计算与 block table 实际宽度不一致。PR body 明确目标：'Share one block-table width calculation between MRV1, MRV2, and the DSA indexer. It applies 128-token alignment and kernel-block splitting, preventing indexer buffer mismatches under alignment and DCP.'

实现拆解

1. 新增统一宽度计算函数：在 vllm/v1/worker/block_table.py 中新增 get_block_table_width(max_num_blocks, block_size, kernel_block_size=None, *, token_alignment=128)，内部先按 token_alignment 与 block_size 的最大公约数换算 block 对齐量，再执行 cdiv 向上取整，最后返回 max_num_blocks * block_size // kernel_block_size，同时支持虚拟 block 分割。
2. 改造 MultiGroupBlockTable：MultiGroupBlockTable.__init__ 中原有的 cdiv(n, 128 // bs) * (128 // bs) 对齐逻辑替换为调用 get_block_table_width，并且对 SlotMappingMode.NONE（Mamba 状态表）传入 token_alignment=None 跳过对齐，保留原有行为。
3. MRV2 接入：vllm/v1/worker/gpu/model_runner.py 的 initialize_kv_cache 删除本地 TRTLLM 对齐代码，改为对每个 KV cache group 调用 get_block_table_width；Mamba spec 分支同样跳过对齐。
4. Builder 宽度注入：vllm/v1/attention/backend.py 在 AttentionMetadataBuilder 基类新增类属性 requires_block_table_width = False；vllm/v1/worker/utils.py 的 AttentionGroup.create_metadata_builders 检测该标志，为需要的 builder 计算并传入

block_table_width。

5. DSA 索引器改造：vllm/v1/attention/backends/mla/indexer.py 中 DeepseekV32IndexerMetadataBuilder 声明 requires_block_table_width = True，构造函数新增 block_table_width 参数，移除原 cdiv(max_model_len, block_size * get_kv_cache_shard_count()) 的本地估算，expanded_block_table_buffer 直接使用注入宽度。
6. 测试配套：在 tests/v1/worker/test_gpu_model_runner.py 新增 3 条针对新函数的单元测试；tests/v1/attention/test_mla_backends.py 复用 helper 计算 padding；tests/v1/attention/test_indexer_deepseek_v4_slot_mapping.py 构造 builder 时显式传入 block_table_width。

关键文件：

- vllm/v1/worker/block_table.py（模块 块表；类别 source；类型 core-logic；符号 get_block_table_width, MultiGroupBlockTable.init）：新增统一的 block table 宽度计算函数 get_block_table_width，并重构 MultiGroupBlockTable.__init__，是本 PR 的单一事实来源。
- vllm/v1/attention/backends/mla/indexer.py（模块 索引器；类别 source；类型 core-logic；符号 DeepseekV32IndexerMetadataBuilder.init）：DSA 稀疏索引器是 #46074 崩溃的直接发生地，改为接收外部注入的 block_table_width，消除 buffer 宽度估算差异。
- vllm/v1/worker/gpu/model_runner.py（模块 模型运行；类别 source；类型 data-contract；符号 GPUModelRunner.initialize_kv_cache）：MRV2 的 initialize_kv_cache 删除本地对齐逻辑，统一走 get_block_table_width，是宽度统一的接入点之一。
- vllm/v1/worker/utils.py（模块 注意力组；类别 source；类型 dependency-wiring；符号 AttentionGroup.create_metadata_builders）：AttentionGroup.create_metadata_builders 通过 requires_block_table_width 为 builder 注入 block_table_width，是实现宽度传递的桥梁。
- vllm/v1/attention/backend.py（模块 后端基类；类别 source；类型 core-logic；符号 AttentionMetadataBuilder.requires_block_table_width）：在 AttentionMetadataBuilder 基类新增 requires_block_table_width 类变量，定义新的 builder 契约。
- tests/v1/worker/test_gpu_model_runner.py（模块 模型运行；类别 test；类型 test-coverage；符号 test_get_block_table_width_aligns_to_128_tokens, test_get_block_table_width_splits_virtual_blocks, test_mamba_state_table_width_is_not_aligned）：新增 3 条针对 get_block_table_width 的单元测试，覆盖 128 token 对齐、虚拟 block 分割和 Mamba 不对齐路径。
- tests/v1/attention/test_mla_backends.py（模块 MLA 后端；类别 test；类型 test-coverage）：将测试内的手动 padding 逻辑替换为 get_block_table_width，保持测试与实现一致。
- tests/v1/attention/test_indexer_deepseek_v4_slot_mapping.py（模块 索引器；类别 test；类型 test-coverage）：适配 DeepseekV32IndexerMetadataBuilder 新构造函数，显式传入 block_table_width。

关键符号：get_block_table_width, DeepseekV32IndexerMetadataBuilder.init, AttentionGroup.create_metadata_builders, GPUModelRunner.initialize_kv_cache, MultiGroupBlockTable.init

关键源码片段

vllm/v1/worker/block_table.py

新增统一的 block table 宽度计算函数 get_block_table_width，并重构 MultiGroupBlockTable.__init__，是本 PR 的单一事实来源。

```
# vllm/v1/worker/block_table.py
def get_block_table_width(
    max_num_blocks: int,
    block_size: int,
    kernel_block_size: int | None = None,
    *,
    token_alignment: int | None = 128,
) -> int:
    """返回可选对齐与 virtual block 分割后的宽度。"""
    # kernel block 未指定时退化为 kv cache block size
    if kernel_block_size is None:
        kernel_block_size = block_size
    # kernel block 必须整除 kv cache block，否则无法拆分为多个 kernel block
    if block_size % kernel_block_size != 0:
        raise ValueError(
            f"kernel_block_size {kernel_block_size} 必须整除 "
            f"block_size {block_size}"
        )
    if token_alignment is not None:
        if token_alignment <= 0:
            raise ValueError("token_alignment 必须为正数")
        # 按 token 数对齐：先换算为 block 数对齐量，再向上取整到倍数。
        # 例：block_size = 64，目标对齐 128 token，则每 2 个 block 对齐一次。
        block_alignment = token_alignment // math.gcd(token_alignment, block_size)
        max_num_blocks = cdiv(max_num_blocks, block_alignment) * block_alignment
    # 最终宽度是 block 数 × 每 block 拆出的 kernel block 数
    return max_num_blocks * block_size // kernel_block_size
```

vllm/v1/attention/backends/mla/indexer.py

DSA 稀疏索引器是 #46074 崩溃的直接发生地，改为接收外部注入的 block_table_width，消除 buffer 宽度估算差异。

```
# vllm/v1/attention/backends/mla/indexer.py
class DeepseekV32IndexerMetadataBuilder(AttentionMetadataBuilder):
    # 声明本 builder 需要外部传入 block table 宽度，
    # worker 侧据此计算并注入 block_table_width 参数
    requires_block_table_width = True

    def __init__(self, *args, block_table_width: int, **kwargs) -> None:
```

```
        super().__init__(*args, **kwargs)
        # ... 省略其他初始化 ...
        # 原来这里用 cdiv(max_model_len, block_size * shard_count) 本地估算，
        # 在 128 token 对齐与 kernel block 分割下会少算一列，导致并发 decode 时
        # expanded_block_table_buffer 与 block table 宽度不匹配而崩溃（#46074）。
        # 现在直接使用 worker 侧统一计算出的宽度，保证与 block table 一致。
        self.expanded_block_table_buffer = torch.zeros(
            (scheduler_config.max_num_batched_tokens, block_table_width),
            dtype=torch.int32,
            device=self.device,
        )
```

评论区精华

Review 中 LucasWilkinson 在 get_block_table_width 的 API 设计上提出 nit：建议让 kernel_block_size 参数可选，以清理调用点重复传 block_size 的写法。作者 MatthewBonanni 在 commit 2650652 中采纳，默认 kernel_block_size = block_size。

合并后有后续问题报告：Leoyzen 在 #46074 评论区指出，在 DCP2 + fp8_ds_mla + MTP 场景下，UniformTypeKVCacheSpecs 没有覆写 max_num_blocks_per_req，导致 runner 与 builder 之间相差 dcp_world_size 倍，expanded_block_table_buffer（8192）与 block table（16384）仍不匹配。drakosha 定位到根因并指向 issue #50825 / PR #50823。

- kernel_block_size 参数可否可选化 (design): MatthewBonanni 在 commit 2650652 中采纳，kernel_block_size 默认取 block_size，简化了 MRV2 与 builder 侧的调用。
- DCP 场景下 block table 宽度仍不匹配 (correctness): 本 PR 未覆盖 DCP 分片路径，需由后续 PR #50823 单独修复。

风险与影响

- 风险：
  1. 行为变更影响面广：get_block_table_width 被 MRV1/MRV2 所有 KV cache group 使用，原先只在 block_size <= 128 时对齐，现在所有 block_size 都会按 128 token 对齐并做 kernel block 分割，可能改变部分后端的 block table padding 量和显存占用，需要回归验证 TRTLLM、FlashMLA 等后端。
  2. DCP 场景仍有缺口：Leoyzen 报告证明在 decode_context_parallel_size > 1 时，UniformTypeKVCacheSpecs.max_num_blocks_per_req 未按 DCP world size 分片，本 PR 未覆盖该路径，需依赖后续 #50823 修复。
  3. Builder 协议扩展风险：新引入 requires_block_table_width 标志，若未来 builder 需要宽度但未声明该属性，构造时会缺少 block_table_width 参数而直接失败；属于显式协议，风险较低但需注意新后端接入。 - 影响：影响所有 v1 引擎路径的 block table 分配与 DSA 稀疏索引器，特别是 GLM-5.2/DeepSeek 系列使用 fp8_ds_mla 的高并发长上下文场景，修复了 #46074 的崩溃；Mamba hybrid 模型行为保持不变。对内部而言，统一宽度计算消除了 MRV1、MRV2、索引器三处逻辑漂移，降低后续维护成本；但 DCP 场景仍需跟进修复。 - 风险标记：核心路径变更，存在 DCP 后续回归，后端宽度契约变更

关联脉络

- PR #48404（被取代的先前修复尝试）: PR body 声明本 PR 取代 #48404，统一了 block table 宽度计算的实现方式。
- PR #50050（被取代的先前修复尝试）: PR body 声明本 PR 取代 #50050，用统一的 get_block_table_width 覆盖其修复目标。
- PR #43970（被取代的先前修复尝试）: PR body 声明本 PR 取代 #43970 的 block table 对齐部分，将 128 token 对齐逻辑收敛到单一函数。
- PR #50823（后续 DCP 宽度修复）: 合并后暴露的 DCP 场景 UniformTypeKVCacheSpecs.max_num_blocks_per_req 未按 dcp_world_size 分片问题，由该 PR 跟进修复。