

PR #52960 完整报告

vllm-project/vllm

[Bugfix] Fix batch-invariant fp32 matmul OOR on SM89 for N=1

合并时间: 2026-08-21 11:46

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/52960>

执行摘要

- 一句话: 修复 SM89 上 N=1 fp32 批不变矩阵乘因共享内存溢出报错
- 推荐动作: 该 PR 值得精读, 适合关注 Triton 内核共享内存优化和批不变性实现的工程师。
核心设计决策是: 按输出宽度 N 而非批次 M 调整 tile, 既修复了 OOR 又保持了批不变性;
通过模块级变量与设备能力检查实现自适应配置。实现的细节如 `_fp32_num_stages` 的引入与条件选择逻辑可作为后续扩展参考。

功能与动机

在 SM86/SM89 GPU (如 NVIDIA L20、L4) 上, 当共享内存预算约 100KB (101376B/block) 时, 启用 `VLLM_BATCH_INVARIANT=1` 会在引擎预热阶段报错:
`triton.runtime.errors.OutOfResources: out of resource: shared memory, Required: 131072, Hardware limit: 101376`。原因是 Triton 对 fp32 的 `matmul_persistent` 内核硬编码了 `128x128x32` tile 且 `num_stages=3`, 未像 fp16 那样根据共享内存限制调整 tile。预热路径调用链为 `compile_or_warm_up_model -> dummy_pooler_run -> nn.Linear(..., 1) -> linear_batch_invariant -> matmul_batch_invariant -> matmul_persistent`, 其中 N=1 的特例导致 Triton 增加了约 32KB 的 staging, 最终需要 131072B, 超出硬件限制。PR 指出这不是 #38670 (fp16 N-tile 上限) 或 #49131 (SM80 按 M 调整 tile) 的重复。

实现拆解

该变更为 1 个源码文件的修改, 核心逻辑分三部分:

1. 新增全局配置变量: 在 `batch_invariant.py` 模块底部新增 `_fp32_block_size_n = 128` 和 `_fp32_num_stages = 3` 两个全局变量, 与已有的 `_fp16_block_size_n` 并列, 用于控制 fp32 内核的 tile 与流水线级数。
2. 调整 `enable_batch_invariant_mode()`: 在函数中根据 `get_max_shared_memory_bytes()` 判断设备共享内存。若大于 106496 (如 A100), 保持 fp16 `BLOCK_SIZE_N=256`、fp32 `BLOCK_SIZE_N=128 / num_stages=3`; 否则 (如 L20/L4) 将 fp16 `BLOCK_SIZE_N` 设为 128, 并为 fp32 设置 `BLOCK_SIZE_N=32 / num_stages=2`, 注释说明仅 N=1 时 OOR, 其他 N 保持原 tile。
3. 在 `matmul_batch_invariant` 和 `bmm_batch_invariant` 中按 N 条件选择 tile: 将 fp32 配置由固定 "BLOCK_SIZE_N": 128 和 "num_stages": 3 改为 "BLOCK_SIZE_N": `_fp32_block_size_n if N == 1 else 128` 和 "num_stages": `_fp32_num_stages if N == 1`

else 3, 使 N=1 时使用小 tile, 其余情况保持原样, 从而不改变批不变性 (tile 仅与输出宽度 N 相关, 与批次 M 无关)。配套测试: 修改了 tests/v1/determinism/test_matmul_batch_invariant.py, 新增 test_fp32_n_equals_1 和 test_fp32_n_gt_1_keeps_wide_tile 两个测试函数, 分别验证 fp32 N=1 时与 torch.mm/bmm 一致且保持批不变性、N>1 时仍匹配 torch.mm。

关键文件:

- vllm/model_executor/layers/batch_invariant.py (模块 执行层; 类别 source; 类型 core-logic; 符号 matmul_batch_invariant, bmm_batch_invariant, enable_batch_invariant_mode): 核心修复文件, 调整 fp32 matmul/bmm 内核 tile 配置, 按 N==1 条件选择小 tile, 并新增模块级变量。

关键符号: enable_batch_invariant_mode, matmul_batch_invariant, bmm_batch_invariant

关键源码片段

vllm/model_executor/layers/batch_invariant.py

核心修复文件, 调整 fp32 matmul/bmm 内核 tile 配置, 按 N==1 条件选择小 tile, 并新增模块级变量。

```
# vllm/model_executor/layers/batch_invariant.py
# 新增模块级 fp32 tile 配置变量, 默认与旧行为一致 (128 tile / 3 stages) 。
_fp32_block_size_n = 128
_fp32_num_stages = 3

def enable_batch_invariant_mode():
    global _batch_invariant_MODE, _batch_invariant_LIB
    # 声明全局变量, 使本次初始化对后续调用生效
    global _fp16_block_size_n, _fp32_block_size_n, _fp32_num_stages
    if _batch_invariant_MODE:
        return

    # ... 原有逻辑省略 ...

    # 根据共享内存容量选择 tile 配置, 避免 Triton OutOfResources
    if get_max_shared_memory_bytes() > 106496:
        # 大共享内存设备 (如 A100) 维持原 tile
        _fp16_block_size_n = 256
        _fp32_block_size_n = 128
        _fp32_num_stages = 3
    else:
        # 小共享内存设备 (如 L20/L4), fp16 已有 N 上限;
        # SM89 上 fp32 N=1 时 128/stages=3 编译需 131072B (超限),
        # 因此仅当 N==1 时退化为 32/stages=2, 其余 N 保持原 tile。
        _fp16_block_size_n = 128
        _fp32_block_size_n = 32
        _fp32_num_stages = 2

def matmul_batch_invariant(a, b, *, bias=None, out=None) -> torch.Tensor:
```

```
# ... 前置逻辑省略 ...
# fp32 配置：仅 N==1 时使用小 tile，保持批不变性（tile 由 N 决定，与 M 无关）
configs[torch.float32] = {
    "BLOCK_SIZE_M": 128,
    "BLOCK_SIZE_N": _fp32_block_size_n if N == 1 else 128,
    "BLOCK_SIZE_K": 32,
    "GROUP_SIZE_M": 8,
    "num_stages": _fp32_num_stages if N == 1 else 3,
    "num_warps": 8,
}
# bmm_batch_invariant 中采用相同的条件选择逻辑（配置复制）
```

评论区精华

Review 中主要包含以下讨论：

- 测试必要性讨论：维护者 yewentao256 在测试文件评论中建议 I don't think we need a specific unit tests for this small update，即认为该小改动无需专门单元测试，但作者保留并添加了测试（状态未明确，最终 PR 已合入）。
- N=1 使用场景提问：yewentao256 问 What is the case that N will equal to 1, in production setting?, 作者 vhagor 回复 Pooling mode + classify head (num_labels=1)，即池化模式与分类头（标签数为 1）的场景。
- 无用导入建议：yewentao256 指出测试文件有 `idleimport`（无用导入），建议清理（Nit）。
- N=1 的实际场景 (question)：作者回应：Pooling mode + classify head (num_labels=1)，即池化模式与单标签分类头。
- 单元测试必要性 (testing)：作者仍保留了新增的测试用例，并已合入，表明测试被接受但非强需求。
- 无用导入清理 (style)：未明确是否修改，但属低优先级风格问题。

风险与影响

- 风险：风险集中在 `vllm/model_executor/layers/batch_invariant.py` 的 `matmul_batch_invariant` 与 `bmm_batch_invariant` 函数：
 - 回归风险：仅对 `N==1` 且共享内存受限的设备改变 tile，其他形状与设备保持原有配置，影响面较小。但需确认 `N == 1` 的判断是否覆盖所有入口（包括 `bmm` 与 `matmul` 的调用路径），以及 `_fp32_block_size_n/_fp32_num_stages` 在未调用 `enable_batch_invariant_mode()` 时默认值为 `128/3`，不会意外改变行为。
 - 性能风险：`N=1` 场景使用更小的 tile，可能在少数模型（如分类头）上降低 fp32 `matmul` 吞吐，但影响范围有限。
 - 兼容性风险：`get_max_shared_memory_bytes() > 106496` 的阈值需适配不同 GPU，但已有 fp16 逻辑作为参照，风险较低。
 - 影响：对用户：解决了在 L20/L4 等 SM89 设备上启用批不变模式时引擎预热崩溃的问题，特别是使用池化 + 单标签分类头（`N=1`）的模型（如 BAAI/bge-reranker-base）。对系统：改动集中在内核配置，不影响核心调度或推理路径。对团队：提供了清晰的 tile 选

择策略，为其他 dtype 的类似问题提供了参考。影响程度中等，主要针对特定硬件和形状的组合。

- 风险标记：核心路径变更，仅在特定硬件 / 形状生效

关联脉络

- PR #38670 fp16 N-tile cap 相关 PR（推测）：PR 正文明确提及 fp16 已按共享内存限制 BLOCK_SIZE_N (#38670)，本 PR 为 fp32 提供类似处理，属于同一设计思路。
- PR #49131 SM80 decode perf via M-dependent tiles（推测）：PR 正文说明该 PR 不是 #49131 的重复，两者都涉及 tile 调整但依据不同（M vs N），在批不变内核优化上关联。