

PR #54845 完整报告

vllm-project/vllm

[ROCm][Perf] Add low-M FP32 router GEMM for gfx950

合并时间: 2026-09-04 01:42

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/54845>

执行摘要

- 一句话: 为 gfx950 添加专用 FP32 路由器 GEMM Triton 内核, 显著降低小 token 批处理延迟。
- 推荐动作: 此 PR 值得精读, 特别是其针对特定硬件和形状的内核设计、与现有分发逻辑的无缝集成方式, 以及全面的性能与正确性验证方法。其设计决策 (如 M 为运行时参数、基于形状的启动配置、自定义算子包装以支持动态编译和 CUDA 图) 体现了对 vLLM 编译和捕获流程的深刻理解, 具有借鉴价值。对于关注 ROCm 性能优化、MoE 路由器实现或 Triton 内核开发的工程师, 本 PR 提供了高质量的参考实现。主要关注点是其硬件特异性和形状硬编码带来的维护性问题, 以及分发逻辑的优先级管理。

功能与动机

PR body 明确指出, 此变更旨在为 gfx950 上的 MoE 路由器提供专用的高性能路径, 解决现有 CUDA 内核 (如 [csrc/libtorch_stable/fp32_router_gemm.cu](#)) 或通用 `F.linear` 在小 token 批量下的性能不足问题。作者通过引用与 #50535 (BF16 AITER GEMM)、#52668 (中等 / 大 M) 等 PR 的差异, 阐明了本 PR 专注于 FP32 权重 / 累加、小 M、以及无需外部依赖 (AITER) 的树内方案, 是对现有 GPU 特定内核的补充。

实现拆解

1. 新增 Triton 内核: 在 `vllm/model_executor/layers/fused_moe/router/rocm_fp32_router_gemm.py` 中实现 `_rocm_fp32_router_gemm_kernel`。该内核采用标准的矩阵乘法分块算法 (BLOCK_M 与 BLOCK_K), 在循环中累加 FP32 部分和, 最终存储结果。M 被设计为运行时参数 (非 `tl.constexpr`), 以避免对每个 `num_tokens` 进行不必要的重编译 (最多 33 次)。内核的启动配置 `_launch_config` 基于 (`hidden_size`, `num_experts`) 和 `num_tokens` 返回 BLOCK_M、BLOCK_K 和 `num_warps`, 为 5 种已调优形状提供了针对性的配置。
2. 扩展 GateLinear 分发逻辑: 修改 `vllm/model_executor/layers/fused_moe/router/gate_linear.py` 中的 `GateLinear` 类。在 `__init__` 中增加对 gfx950 平台的检测, 并基于 `ROCM_FP32_ROUTER_GEMM_SUPPORTED_SHAPES` 设置 `allow_fp32_router_gemm` 标志。在 `forward` 方法的 Tier 2 分发路径 (`fp32_router_gemm_dispatch` 自定义算子) 中, 针对 ROCm 平台, 将调用重定向到新的 `rocm_fp32_router_gemm`。同时, 在 `fp32_router_gemm_dispatch_impl` 函数中, 对 ROCm 路径添加了非连续输入的 `.contiguous()` 处理, 并集成 `can_use_rocm_fp32_router_gemm` 进行运行时检查。

3. 添加全面测试：新增 tests/kernels/test_rocm_fp32_router_gemm.py，包含 384 个测试用例，覆盖所有 M (0-32)、激活类型 (BF16/FP32)、5 种形状、top-k 保留、无效输入拒绝、CUDA 图输入可变性、自定义算子分发以及动态 torch.compile。修改 tests/kernels/test_gate_linear_rocm_dispatch.py，添加了针对 gfx950 上 allow_fp32_router_gemm 标志启用 / 禁用条件的平台无关测试，使用 monkeypatch 模拟 on_gfx950()。
4. 更新 CI 配置：在 .buildkite/test-amd.yaml 的 MI355 Core Operation Kernels 任务中，将新测试文件 test_rocm_fp32_router_gemm.py 及其源码依赖添加到 source_file_dependencies 和 commands 列表中，确保测试在 AMD CI 上自动运行。

关键文件：

- vllm/model_executor/layers/fused_moe/router/rocm_fp32_router_gemm.py (模块 路由器内核；类别 source；类型 core-logic；符号 _rocm_fp32_router_gemm_kernel, _launch_config, can_use_rocm_fp32_router_gemm, _validate_inputs)：核心变更：包含为 gfx950 优化的 Triton GEMM 内核、启动配置和输入验证逻辑。
- vllm/model_executor/layers/fused_moe/router/gate_linear.py (模块 路由器分发；类别 source；类型 data-contract；符号 GateLinear.init, GateLinear.forward, fp32_router_gemm_dispatch_impl)：关键分发逻辑：GateLinear 类初始化新增 gfx950 检测，forward 方法和 dispatch_impl 函数中集成新内核的调用路径。
- tests/kernels/test_rocm_fp32_router_gemm.py (模块 内核测试；类别 test；类型 test-coverage；符号 test_rocm_fp32_router_gemm_matches_reference, test_rocm_fp32_router_gemm_preserves_topk, test_rocm_fp32_router_gemm_rejects_invalid_inputs, test_rocm_fp32_router_gemm_cuda_graph_obeys_input_mutation)：全面测试：384 个用例覆盖内核正确性、边界条件、动态编译和 CUDA 图兼容性，是验证变更质量的关键。
- tests/kernels/test_gate_linear_rocm_dispatch.py (模块 分发测试；类别 test；类型 test-coverage；符号 test_rocm_gfx950_enables_fp32_router_gemm, test_rocm_fp32_router_gemm_is_replicated_across_ep_sizes, test_rocm_fp32_router_gemm_rejects_unsupported_configs)：分发逻辑测试：使用 monkeypatch 模拟平台条件，验证 GateLinear 的 allow_fp32_router_gemm 标志在 gfx950 和不同形状下的正确设置。
- .buildkite/test-amd.yaml (模块 CI 配置；类别 config；类型 configuration)：CI 集成：将新测试和源文件依赖添加到 AMD CI 管道，确保变更在 AMD 硬件上持续验证。

关键符号：_rocm_fp32_router_gemm_kernel, _launch_config, can_use_rocm_fp32_router_gemm, _validate_inputs, rocm_fp32_router_gemm, fp32_router_gemm_dispatch_impl

关键源码片段

vllm/model_executor/layers/fused_moe/router/rocm_fp32_router_gemm.py

核心变更：包含为 gfx950 优化的 Triton GEMM 内核、启动配置和输入验证逻辑。

```

# 内核核心逻辑：在循环中累加 FP32 部分和，处理 M 可能不是 BLOCK_M 整数倍的情况
@triton.jit
def _rocm_fp32_router_gemm_kernel(
    hidden_states_ptr,
    router_weight_ptr,
    output_ptr,
    M, # 运行时参数，避免为每个 num_tokens 重编译
    K: tl.constexpr, # 隐藏大小
    N: tl.constexpr, # 专家数量
    BLOCK_M: tl.constexpr, # 由 _launch_config 根据形状和 token 数决定
    BLOCK_K: tl.constexpr,
):
    pid = tl.program_id(0)
    pid_m = pid // N
    pid_n = pid - pid_m * N
    offsets_m = pid_m * BLOCK_M + tl.arange(0, BLOCK_M)
    offsets_k = tl.arange(0, BLOCK_K)
    partials = tl.zeros((BLOCK_M, BLOCK_K), dtype=tl.float32) # FP32 累加器

    for k_start in range(0, K, BLOCK_K):
        offsets_k_block = k_start + offsets_k
        # 加载隐藏状态，当 offsets_m 超出 M 范围时用 0.0 填充
        hidden_states = tl.load(
            hidden_states_ptr + offsets_m[:, None] * K + offsets_k_block[None, :],
            mask=offsets_m[:, None] < M,
            other=0.0,
        ).to(tl.float32) # 提升到 FP32 进行累加
        router_weight = tl.load(router_weight_ptr + pid_n * K + offsets_k_block)
        partials += hidden_states * router_weight[None, :]

    accumulator = tl.sum(partial, axis=1)
    # 写回结果，同样使用掩码处理边界
    tl.store(
        output_ptr + offsets_m * N + pid_n,
        accumulator,
        mask=offsets_m < M,
    )

```

评论区精华

核心讨论围绕实现细节和正确性展开。reviewer zhou9402（来自 Issue 评论）提出了关键设计建议：

1. 将 M 从 tl.constexpr 改为运行时参数以避免不必要的编译。
2. 使用 on_gfx950() 替代 is_device_capability((9, 5)) 以匹配代码库惯例。
3. 建议对非连续输入使用 .contiguous() 而非回退到 F.linear。作者 Fangzhou-Ai 在 00f351e4fe 提交中全部采纳并实施。reviewer jeejeelee 质疑了内核中对 FP32 路由器权重进行类型转换的必要性，作者确认并移除了冗余转换。另一个讨论线程涉及支持的 5 种形状

的具体含义，benenzhu询问了背景，作者详细解释了每种形状对应的模型（如 (6144,128) 是 MiniMax-M3, (3072, 256) 是 MiniMax-M2/M2.5），并说明这是通过审计所有树内 FP32 GateLinear 消费者得出的。reviewer AndreasKaratzas和 tjtananaa对 buildkite/test-amd.yaml 中 --ignore 命令的用法产生疑问，作者澄清了 --ignore 仅针对 test_minimax_reduce_rms.py，而新测试是作为独立的 pytest 目标添加的，并未被忽略。所有讨论均在合并前得到解决或澄清。

- 内核 M 参数设计 (design): 采纳建议，将 M 改为运行时参数，避免不必要的 JIT 编译。
- gfx950 设备检测方式 (style): 采纳建议，统一使用 on_gfx950()。
- 非连续输入处理 (correctness): 在 fp32_router_gemm_dispatch_impl 中为 ROCm 路径添加了 x.contiguous() 处理，并通过测试 monkeypatch F.linear 来证明非连续输入确实走了优化内核。
- 支持的 5 种形状含义 (question): 作者逐一解释：(3072, 256) MiniMax-M2/M2.5, (4096, 8) Nemotron-H, (4096, 192) HY-V3/Hunyuan V3, (6144, 128) MiniMax-M3, (6144, 256) GLM-5.2。这些是审计所有树内 FP32 GateLinear 消费者后确定的。
- CI 测试是否被忽略 (testing): 作者澄清 --ignore 仅针对 test_minimax_reduce_rms.py，新测试作为独立 pytest 目标添加，并通过 CI 运行结果验证。

风险与影响

- 风险：技术风险主要集中在硬件特异性和正确性验证上。1. 硬件特异性：内核专为 gfx950 调优 (on_gfx950() 检查)，且形状硬编码在 ROCM_FP32_ROUTER_GEMM_SUPPORTED_SHAPES 中。风险在于未来新形状或硬件变体出现时，可能需要手动更新支持列表。2. 正确性验证：内核的正确性依赖于与 F.linear 的逐元素对比 (ATOL=5e-4)。虽然测试非常全面 (384 用例)，但精度误差 (最大绝对误差 1.23e-4) 在可接受范围内，需确保在端到端模型评估 (GSM8K, GPQA Diamond, MMMU-Pro) 中不会累积。3. 性能回归：分发路径的改变 (GateLinear.forward 中新增条件分支) 理论上可能对非 gfx950 或超出形状范围的调用引入微小开销，但 PR 中的性能测试显示 MiniMax-M3 在 gfx950 上获得显著提升，且未报告其他平台性能下降。4. 与现有 PR 的互操作：PR body 指出与 #50535 (BF16 AITER)、#52668 (大 M) 互补，但 fp32_router_gemm_dispatch_impl 中的分发逻辑 (先尝试 gfx950 内核，再回退到 F.linear 或 CUDA 内核) 需要确保不会因优先级问题导致次优选择。
- 影响：该 PR 主要影响使用 gfx950 硬件运行特定 MoE 模型 (MiniMax-M2/M3, Nemotron-H, HY-V3, GLM-5.2) 的用户。1. 性能影响：直接影响是路由器计算延迟降低约 50%，间接提升端到端推理吞吐量 (MiniMax-M3 在并发 1-32 下提升 3.15%-8.32%) 和每令牌时间 (TPOT/ITL 降低 3.11%-6.77%)。模型评估结果 (GSM8K, GPQA Diamond, MMMU-Pro) 验证了精度无损。2. 系统影响：在 vLLM 核心的 MoE 路由器分发逻辑中添加了新的硬件特定分支，增加了代码路径的复杂性，但通过清晰的测试和封装 (独立文件、自定义算子) 进行了控制。CI 配置的更新确保了变更在 AMD CI 上的持续验证。3. 团队影响：为 vLLM 的 ROCm 性能优化提供了新的模式 (Triton 内核 + 形状特定调优)，可为其他 gfx950 优化 (如 PR#54916) 提供参考。清晰的 PR 描述和 review 讨论为后续维护者理解设计决策提供了良好基础。
- 风险标记：硬件特异性强，形状硬编码需维护，分发路径新增条件分支

关联脉络

- PR #50535 [ROCm][Perf] Add BF16 AITER GEMM for gfx950 MoE routers: PR body 中明确对比：本 PR 是树内 Triton 实现，无需 AITER 依赖，且保持 FP32 权重和累加；而 #50535 使用外部调优的 BF16 AITER GEMM。两者互补，针对不同精度路径。
- PR #52668 [ROCm][Perf] Add stacked hipBLASLt GEMM for medium/large MoE router batches: PR body 中明确对比：#52668 针对中等 / 大 M ($M > 32$) 并使用权重分解；本 PR 专注于小 M ($M \leq 32$) 且无权重分解。两者是不同 M 范围的优化。
- PR #54682 [ROCm][Perf] Optimize MiniMax-M3 decode indexer and top-k: 同属 gfx950 和 MiniMax-M3 性能优化线。#54682 优化稀疏注意力 top-k，本 PR 优化路由器 GEMM，共同提升该模型整体性能。
- PR #54916 [WIP][ROCm][Perf] Optimize low-M router GEMM for larger concurrencies: PR 评论中 benenzhu 提到此 PR 是后续优化（支持更大并发）的基础。本 PR 提供了基础内核和形状支持，#54916 在此基础上进行扩展优化。