PR #49932 完整报告

vllm-project/vllm

[Linear] [Kernel] add block-wise scaled_mm

合并时间：2026-08-06 14:40

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/49932

执行摘要

- 一句话：新增基于 torch._scaled_mm 的 FP8 block-scale 内核后端
- 推荐动作：值得精读：它示范了 vLLM 内核抽象中的三件套（is_supported / can_implement / apply_*）如何承接平台能力探测与数据布局契约，尤其是基于 PyTorch 上游实现约束做剪裁并可回退的设计。对要扩展 FP8 block-scale 后端或接入 cuBLASLt 新特性的同学有参考价值。建议同时关注其随 PyTorch 版本的兼容性回归测试。

功能与动机

PR body 明确指出目标是提供一个 native、dependency-free 的 torch._scaled_mm block-wise FP8 后端，不需要 CUTLASS/DeepGEMM/Triton 那样的额外库或 per-shape tuning。作者在 H20 上的基准显示：在 M≥1024 的 compute-bound 区间，torch 后端与次优后端相当或更快（最高 1.09x）；小 batch 下虽明显落后于 DeepGEMM（约 0.43x-0.68x），但新内核排在 FP8 block kernel 优先级列表末尾，默认不会抢占现有高性能后端。

实现拆解

1. 新增内核类：在 vllm/model_executor/kernels/linear/scaled_mm/pytorch.py 中定义 BlockWiseTorchFP8ScaledMMLinearKernel，继承 Fp8BlockScaledMMLinearKernel。构造函数读取 activation_quant_key.scale 描述符并构造 QuantFP8 实例，其中 column_major_scales=current_platform.is_cuda_alike()，保证 CUDA 上激活 scale 以列主序布局交给 torch._scaled_mm。
2. 平台与 shape 约束：is_supported 先限定 CUDA/ROCm/XPU，再对 CUDA 仅放行 SM90（compute capability 90），原因是 PyTorch 的 _check_deepseek_support() 只为 SM90 实现了 DeepSeek 风格 (1x128, 128x128) block scaling；can_implement 复查激活 group shape 为 (1, 128)、权重 block shape 为 (128, 128)，不匹配时返回明确原因。
3. GEMM 执行：apply_block_scaled_mm 先按 cuBLASLt 要求把 M 填充到 4 的倍数（A 与列主序 scale 同步填充，填充行不初始化），调用 torch._scaled_mm(A, B.t(), scale_a=As, scale_b=Bs.t(), out_dtype=...)，兼容 PyTorch < 2.5 的 tuple 返回后切回有效 M 行。
4. 注册与优先级：在 vllm/model_executor/kernels/linear/__init__.py 中导出该类，加入 "torch" backend 集合，并追加到 _POSSIBLE_FP8_BLOCK_KERNELS 的 CUDA 列表末尾（FlashInfer、DeepGEMM、CUTLASS 之后），默认不会抢占现有高性能后端。

5. 测试配套：tests/kernels/quantization/test_block_fp8.py 新增 test_w8a8_block_fp8_torch_scaled_mm_matmul，SM90 门控，选 M=83 非 4 倍数覆盖 padding 路径，对照 native_w8a8_block_matmul 验证 rel_diff < 0.001；未引入新配置或文档改动。

关键文件：

- vllm/model_executor/kernels/linear/scaled_mm/pytorch.py（模块 量化内核；类别 source；类型 core-logic；符号 BlockWiseTorchFP8ScaledMMLinearKernel, init, is_supported, can_implement）：核心实现文件：新增 BlockWiseTorchFP8ScaledMMLinearKernel，完成 torch._scaled_mm block 路径的平台校验、shape 契约与 M 填充逻辑。
- tests/kernels/quantization/test_block_fp8.py（模块 内核测试；类别 test；类型 test-coverage；符号 test_w8a8_block_fp8_torch_scaled_mm_matmul）：新增 SM90 门控端到端测试，用 M=83 覆盖 M 填充路径，是 kernel 正确性的主要保障。
- vllm/model_executor/kernels/linear/__init__.py（模块 内核注册；类别 source；类型 data-contract；符号 BlockWiseTorchFP8ScaledMMLinearKernel）：内核注册表集成点：把新 kernel 加入 torch backend 集合与 CUDA block kernel 优先级列表，决定其默认启用顺序。

关键符号：BlockWiseTorchFP8ScaledMMLinearKernel.init, BlockWiseTorchFP8ScaledMMLinearKernel.is_supported, BlockWiseTorchFP8ScaledMMLinearKernel.can_implement, BlockWiseTorchFP8ScaledMMLinearKernel.apply_block_scaled_mm, test_w8a8_block_fp8_torch_scaled_mm_matmul

关键源码片段

tests/kernels/quantization/test_block_fp8.py

新增 SM90 门控端到端测试，用 M=83 覆盖 M 填充路径，是 kernel 正确性的主要保障。

```
@pytest.mark.skipif(
    not (current_platform.is_cuda() and current_platform.has_device_capability(90)),
    reason="torch._scaled_mm DeepSeek-style block scaling only supports SM90.",
)
def test_w8a8_block_fp8_torch_scaled_mm_matmul():
    # M=83 故意选非 4 倍数，覆盖 apply_block_scaled_mm 的 M 填充路径
    from vllm.model_executor.kernels.linear.scaled_mm.pytorch import (
        BlockWiseTorchFP8ScaledMMLinearKernel,
    )

    M, N, K = 83, 576, 7168
    block_size = [128, 128]
    out_dtype = torch.bfloat16
    torch.manual_seed(0)
    factor_for_scale = 1e-2
    fp8_info = torch.finfo(torch.float8_e4m3fn)
```

```
fp8_max, fp8_min = fp8_info.max, fp8_info.min

# 随机生成输入，并用 native_w8a8_block_matmul 作为数值参考
A_fp32 = (torch.rand(M, K, dtype=torch.float32) - 0.5) * 2 * fp8_max
B_fp32 = (torch.rand(N, K, dtype=torch.float32) - 0.5) * 2 * fp8_max
B_fp8 = B_fp32.clamp(min=fp8_min, max=fp8_max).to(torch.float8_e4m3fn)

n_tiles = (N + block_size[0] - 1) // block_size[0]
k_tiles = (K + block_size[1] - 1) // block_size[1]
Bs = torch.rand(n_tiles, k_tiles, dtype=torch.float32) * factor_for_scale

# 参考实现用行主序激活 scale，kernel 路径用列主序
A_fp8, As = per_token_group_quant_fp8(
    A_fp32, block_size[1], column_major_scales=False
)
ref_out = native_w8a8_block_matmul(A_fp8, B_fp8, As, Bs, block_size, out_dtype)

A_fp8_cuda, As_cuda = per_token_group_quant_fp8(
    A_fp32, block_size[1], column_major_scales=True
)

# 用 __new__ + SimpleNamespace 构造最小 stub，只测 GEMM 路径本身
stub = BlockWiseTorchFP8ScaledMMLinearKernel.__new__(
    BlockWiseTorchFP8ScaledMMLinearKernel
)
stub.config = types.SimpleNamespace(out_dtype=out_dtype)
out = stub.apply_block_scaled_mm(
    A_fp8_cuda.cuda(), B_fp8.cuda(), As_cuda.cuda(), Bs.cuda()
)

ref_out = ref_out.cuda()
rel_diff = torch.mean(
    torch.abs(out.to(torch.float32) - ref_out.to(torch.float32))
) / torch.mean(torch.abs(ref_out.to(torch.float32)))
# 与参考实现相对误差小于 0.001
assert rel_diff < 0.001
```

评论区精华

核心讨论只有一条：jikunshang 在 pytorch.py 第 285 行追问 "emmm, it's supported on SM90 only? SM100/SM120 are not supported?"，作者引用 PyTorch 的 _check_deepseek_support() 确认限制是上游行为，非本 PR 主动收窄。随后 jikunshang 请求 mgoin 复核并触发 CI，最终给出 APPROVED。另因 PR 来自 fork，Claude bot 的自动 review 被禁用。

- SM90-only 限制与上游 PyTorch 支持范围 (question): 确认限制来自 PyTorch 上游实现，非本 PR 主动收窄；代码中通过 is_supported 返回明确报错信息并带注释指引。

风险与影响

- 风险：
  - 平台偏差：is_supported 声明 ROCm/XPU 也支持，但测试仅覆盖 CUDA SM90，且 torch._scaled_mm 的 block 路径在非 CUDA 平台行为未验证，存在运行时才暴露的风险。
  - 上游耦合：依赖 PyTorch 内部 _check_deepseek_support() 的行为，后续 PyTorch 版本可能调整支持范围或 scale 布局（如 1x128/128x128），需要随版本回归。
  - 优先级顺序：新内核排在 CUDA block kernel 列表最后，但若其 is_supported 在非 Hopper 返回 False，选择逻辑应能正确跳过；需确认 can_implement 在选择时不会因先于性能排序被提前命中。
  - 数值精度：FP8 block-scale 量化结果以 rel_diff < 0.001 校验，但单测只覆盖一组形状和 seed，未覆盖 N、K 边界及 out_dtype=float16 等情况。
  - M padding 额外开销：padding 会引入一次拷贝，M 不是 4 的倍数时开销占比随 M 减小而增大，小 batch 场景本已落后其他后端。
  - 影响：对用户：为 DeepSeek 风格 FP8 block 量化模型新增一条无需依赖 DeepGEMM/CUTLASS 等库的原生路径，Hopper 上在长上下文 / 大 batch 场景可能带来至多约 9% 的 kernel 收益（H20 数据 M≥1024 时 1.01x-1.09x），同时小 batch 保持回退到其他后端。对系统：不改变默认 backend 选择，风险隔离在新增分支。对团队：为后续在 torch 后端上扩展更多 block scale 布局（如 1x64、128x64）打下模式基础。影响程度为中低，主要面向 Hopper 用户。
  - 风险标记：仅 SM90 生效，依赖 PyTorch 上游行为，ROCm/XPU 缺乏测试，小 batch 性能回退

关联脉络

- PR #50840 [XPU] Route AWQ linear through choose_mp_linear_kernel: 同为 linear kernel 分发 / 注册体系改动，与本 PR 在 kernels/linear 后端选择逻辑上属于同一演进线。
- PR #50942 [MoE] Align TRTLLM MXFP4 autotune buckets: 同为量化 GEMM 性能优化，通过统一 autotune 配置提升 prefill 约 10%，与本次 FP8 block-scale 后端性能调优方向呼应。
- PR #51093 [Bugfix][Humming] Preserve ModelOpt FP8 weight dimensions: 同为 FP8 量化内核正确性维护，关注权重 /scale 布局元数据，与本 PR 的 scale 列主序契约处理同源。