# PR #33596 完整报告

sgl-project/sglang

[Test] Replace GEMM backend e2e matrices with layer-level unit tests

合并时间：2026-08-05 06:50

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33596

## 执行摘要

- 一句话：用层级单元测试替换 GEMM e2e 矩阵，CI 提速并修复 block 检查
- 推荐动作：值得精读，特别是 test_nvfp4_linear_backends.py 中 convert_swizzled_to_linear 的 K-tile crop 修复和 test_fp8_blockwise_linear_backends.py 的按 SM 自适应后端集合设计。这两个文件是量化后端测试的优质模板，展示了如何在不启动服务器的情况下覆盖 kernel 专属的 padding / swizzle / scale 布局逻辑。同时建议关注 fp8.py 的修复是否与后续 skip_block_quant_check 调用方有交互。

## 功能与动机

三个 e2e 测试文件（test_nvfp4_gemm.py、test_fp8_blockwise_gemm.py、test_fp8_gemm_sm120.py）每个后端都要启动一次服务器并跑完整 GSM8K 评测，单文件耗时 146-430 秒，且覆盖存在严重漏洞：TestFP8BlockwiseGemmFlashinferDeepGemm 被 skipIf(sm != 90) 门控在 B200 专属套件中永远不运行，TestMXFP8GemmTriton 带无条件 @unittest.skip，flashinfer_cutlass 的 FP8-blockwise 后端从未被 e2e 覆盖。PR body 指出这些文件本质是 flag 接线冒烟测试（来源于 #14379、#16534、#20717），而非后端正确性测试。新测试的目标是让每个后端的权重预处理（NVFP4 padding/interleave、TRTLLM shuffle、DeepGEMM UE8M0 scale requant、MXFP8 scale packing）和 GEMM dispatch 的数值错误在秒级暴露出来。

## 实现拆解

实现分四步：

1. 新增 NVFP4 层级单元测试（test/registered/unit/layers/quantization/test_nvfp4_linear_backends.py）：构造 ModelOptFp4LinearMethod 并在内存中构建含 NVFP4 checkpoint 格式权重的 linear 层，通过 mock.patch 替换 fp4_utils.FP4_GEMM_RUNNER_BACKEND 来遍历 flashinfer_cutedsl / cutlass / cudnn / trtllm 四个后端。关键工具函数 convert_swizzled_to_linear 从 test_fp4_moe.py 移植并修复了 K-tile padding crop 的 bug——原来只裁剪 M 维 padding，现在同时裁剪 K 维（按 k // block_size 列裁剪 scale）。测试使用 (5, 160, 336) 非对齐 shape 命中 TRTLLM 的 N->128 shuffle pad、scale K/16->4 pad 和 CUTLASS 的 K 32 对齐 pad。
2. 新增 FP8 / MXFP8 / per-tensor 层级单元测试（test/registered/unit/layers/quantization/test_fp8_blockwise_linear_backends.py）：覆盖 Fp8LinearMethod（blockwise 和

MXFP8）以及 ModelOptFp8LinearMethod（per-tensor auto dispatch）。后端集合按 get_device_sm() 动态决定：SM100/103 用 triton/deep_gemm/flashinfer_trtllm/flashinfer_cutlass，SM120 用 triton/cutlass，SM90 用 triton/deep_gemm/flashinfer_deepgemm。三个测试类共享 _check_backend 核心逻辑，用余弦相似度 > 0.99 和宽松的 rtol/atol 做数值断言，以容忍 DeepGEMM UE8M0 单元素 scale 舍入异常。

3. 删除三个 e2e 文件：test/registered/quant/ 下的 test_nvfp4_gemm.py、test_fp8_blockwise_gemm.py、test_fp8_gemm_sm120.py 整体删除。PR body 说明 CLI 参数接线仍由显式传参的模型 e2e（如 test_kimi_k26_nvfp4_dflash.py、test_deepseek_r1_fp8_trtllm_backend.py）覆盖，默认路径精度由常规模型套件兜底。
4. 修复 fp8.py 的 validate_block_quant_shapes：原实现无论是否 skip 都在函数入口调用 get_parallel().tp_size，导致设置 skip_block_quant_check 的单元测试在未初始化分布式组时崩溃。修复把 tp_size 读取移入非 skip 分支，行为对正常路径完全不变。

配套的 CI 注册：FP8 文件注册在 4-gpu-b200（est 120s）、1-gpu-small / 1-gpu-large（各 60s）；NVFP4 文件注册在 4-gpu-b200（est 120s），新增测试均在 base-b 阶段运行，秒级完成。

关键文件：

- test/registered/unit/layers/quantization/test_fp8_blockwise_linear_backends.py（模块 量化测试；类别 test；类型 test-coverage；符号 _fp8_block_backends, _mxfp8_backends, _quantize_fp8_blockwise, _quantize_mxfp8）：新增的 FP8/MXFP8/per-tensor 层级单元测试主文件，覆盖三种量化格式、按 SM 自适应后端集合，包含核心数值校验逻辑 _check_backend，并首次在 CI 中覆盖 flashinfer_deepgemm 和 flashinfer_cutlass FP8-blockwise。
- test/registered/unit/layers/quantization/test_nvfp4_linear_backends.py（模块 量化测试；类别 test；类型 test-coverage；符号 convert_swizzled_to_linear, break_fp4_bytes, dequantize_nvfp4_to_dtype, _make_quantized_layer）：新增的 NVFP4 层级单元测试，覆盖四个 SM100 后端，并包含 convert_swizzled_to_linear 修复（K-tile padding crop bug）及非对齐 shape 的 padding 路径测试。
- python/sglang/srt/layers/quantization/fp8.py（模块 量化实现；类别 source；类型 core-logic；符号 validate_block_quant_shapes）：核心源码修改：修复 validate_block_quant_shapes 在 skip_block_quant_check 时仍读取 tp_size、需要分布式初始化的问题，使单元测试可独立运行。
- test/registered/quant/test_fp8_blockwise_gemm.py（模块 量化测试；类别 test；类型 deletion；符号 FP8BlockwiseGemmBase, MXFP8GemmBase, TestFP8BlockwiseGemmTriton, TestFP8BlockwiseGemmDeepGemm）：被删除的 e2e 测试文件之一，原注册在 extra-b 阶段（约 430 秒），包含多个从未实际运行的测试类（flashinfer_deepgemm 被错误 gate，MXFP8 Triton 被无条件 skip）。
- test/registered/quant/test_fp8_gemm_sm120.py（模块 量化测试；类别 test；类型 deletion；符号 FP8GemmSM120Base, TestFP8PerTensorGemmSM120Auto, TestFP8BlockwiseGemmSM120Auto）：被删除的 SM120 e2e 测试文件，原注册在 extra-a 阶段（约 146 秒），per-tensor 与 blockwise 各一个用例。

- test/registered/quant/test_nvfp4_gemm.py（模块 量化测试；类别 test；类型 deletion；符号 FP4GemmBase, TestFP4GemmFlashinferCutlass, TestFP4GemmFlashinferCudnn, TestFP4GemmFlashinferTrtllm）：被删除的 NVFP4 e2e 测试文件，原注册在 base-c 阶段（约 350 秒），每个后端启动一个服务器跑 GSM8K。

关键符号：convert_swizzled_to_linear, break_fp4_bytes, dequantize_nvfp4_to_dtype, _check_backend, _quantize_fp8_blockwise, _quantize_mxfp8, validate_block_quant_shapes

## 关键源码片段

### test/registered/unit/layers/quantization/test_fp8_blockwise_linear_backends.py

新增的 FP8/MXFP8/per-tensor 层级单元测试主文件，覆盖三种量化格式、按 SM 自适应后端集合，包含核心数值校验逻辑 _check_backend，并首次在 CI 中覆盖 flashinfer_deepgemm 和 flashinfer_cutlass FP8-blockwise。

```
# 核心校验逻辑：对每个后端、每个 shape 执行量化层全流程并断言数值
class _LinearBackendCheck(CustomTestCase):
    def _check_backend(self, backend: str, allowed, shapes, build_layer):
        # 后端不在当前 SM 的支持集合里就直接跳过，保证同一文件可跨 SM 运行
        if backend not in allowed:
            self.skipTest(f"{backend} not in SM{get_device_sm()} backend set")
        torch.manual_seed(7)
        for m, n, k in shapes:
            with self.subTest(backend=backend, shape=(m, n, k)):
                # 用 mock 替换全局 GEMM runner 选择，模拟 --fp8-gemm-backend 指定
                with mock.patch.object(
                    fp8_utils,
                    "FP8_GEMM_RUNNER_BACKEND",
                    Fp8GemmRunnerBackend(backend),
                ):
                    method, layer, w_dequant = build_layer(n, k)
                    method.process_weights_after_loading(layer)

                    x = torch.randn((m, k), device="cuda", dtype=torch.bfloat16) / 10
                    out = method.apply(layer, x)

                    # 参考 = 反量化权重矩阵乘，不经过任何 kernel 优化
                    ref = x.float() @ w_dequant.T
                    self.assertEqual(out.shape, (m, n))
                    cos = torch.nn.functional.cosine_similarity(
                        out.float().flatten(), ref.flatten(), dim=0
                    ).item()
                    self.assertGreater(cos, 0.99)
                    # atol 容忍 DeepGEMM UE8M0 单元素 scale 舍入异常；
                    # 真正的 kernel/layout 错误会差出数量级，宽松阈值仍能抓住
                    torch.testing.assert_close(out.float(), ref, rtol=5e-2, atol=1e-1)
```

## test/registered/unit/layers/quantization/test_nvfp4_linear_backends.py

新增的 NVFP4 层级单元测试，覆盖四个 SM100 后端，并包含 convert_swizzled_to_linear 修复（K-tile padding crop bug）及非对齐 shape 的 padding 路径测试。

```
# 将 flashinfer swizzled scale 布局转回 linear 布局，供参考反量化使用
def convert_swizzled_to_linear(a_sf_swizzled: torch.Tensor, m, k, block_size):
    m_tiles = (m + 128 - 1) // 128
    f = block_size * 4
    k_tiles = (k + f - 1) // f
    tmp = torch.reshape(a_sf_swizzled, (1, m_tiles, k_tiles, 32, 4, 4))
    tmp = torch.permute(tmp, (0, 1, 4, 3, 2, 5))
    out = tmp.reshape(m_tiles * 128, k_tiles * f // block_size)
    # 关键修复：除了裁剪 M 维的 128 对齐 padding，还必须裁剪 K 维 tile padding。
    # 原来从 test_fp4_moe.py 移植的实现只裁了 M 维，在非对齐 K 时会残留错误列
    return out[0:m, 0 : k // block_size]


# NVFP4 反量化参考：把 uint8 双 packed fp4 拆成高低 nibble 并映射到 float
def dequantize_nvfp4_to_dtype(
    tensor_fp4, tensor_sf, global_scale, dtype, device, block_size=16
):
    assert tensor_fp4.dtype == torch.uint8
    m, packed_k = tensor_fp4.shape
    k = packed_k * 2
    tensor_f32 = break_fp4_bytes(tensor_fp4, torch.float32)
    tensor_f32 = tensor_f32.reshape(m, k // block_size, block_size)
    tensor_sf = tensor_sf.view(torch.float8_e4m3fn)
    tensor_sf = convert_swizzled_to_linear(tensor_sf, m, k, block_size)
    tensor_sf_dtype = tensor_sf.to(torch.float32) / global_scale
    out = (tensor_f32 * tensor_sf_dtype.unsqueeze(-1)).reshape(m, k)
    return out.to(dtype=dtype)
```

## python/sglang/srt/layers/quantization/fp8.py

核心源码修改：修复 validate_block_quant_shapes 在 skip_block_quant_check 时仍读取 tp_size、需要分布式初始化的问题，使单元测试可独立运行。

```
@staticmethod
def validate_block_quant_shapes(
    quant_config,
    input_size: int,
    input_size_per_partition: int,
    output_size: int,
    output_size_per_partition: int,
    output_partition_sizes: List[int],
    skip_block_quant_check: bool = False,
):
    block_n, block_k = (
        quant_config.weight_block_size[0],
        quant_config.weight_block_size[1],
```

```
    )

    if skip_block_quant_check:
        # 跳过校验时不再触碰分布式全局状态，让单进程单元测试也能走通
        print_warning_once(
            "Skipping block quantization checks for weight partition."
        )
    else:
        # 仅在真正需要校验时才读取 TP 世界大小，避免未初始化分布式组时崩溃
        tp_size = get_parallel().tp_size
        # row parallel 约束：输入分区需整除 block_k
        if tp_size > 1 and input_size // input_size_per_partition == tp_size:
            if input_size_per_partition % block_k != 0:
                raise ValueError(
                    f"Weight input_size_per_partition = "
                    f"{input_size_per_partition} is not divisible by "
                    f"weight quantization block_k = {block_k}."
                )
        # column parallel / merged weights 约束：每个输出分区需整除 block_n
        if (
            tp_size > 1 and output_size // output_size_per_partition == tp_size
        ) or len(output_partition_sizes) > 1:
            for output_partition_size in output_partition_sizes:
                if output_partition_size % block_n != 0:
                    raise ValueError(
                        f"Weight output_partition_size = "
                        f"{output_partition_size} is not divisible by "
                        f"weight quantization block_n = {block_n}."
                    )
```

# 评论区精华

该 PR 没有 review 评论，主要交互来自 issue 的 /rerun-test 命令。第一次 rerun 在 4-gpu-b200 上两个测试均通过，第二次 rerun 中 4-gpu-b200 出现一次失败（😕），而 1-gpu-5090 和 1-gpu-h100 均通过。由于没有关联的失败日志或后续说明，这次失败大概率是 B200 runner 的环境抖动或偶发数值问题，但也提示新单元测试在 4-gpu 环境可能存在轻微 flakiness。PR body 中特别强调的两个历史覆盖漏洞值得关注：flashinfer_deepgemm 因 skip 条件与 runner 不匹配从未运行、MXFP8 Triton 被无条件跳过——这些正是 *e2e* 矩阵结构性失效的典型例子。

- CI rerun 结果：4-gpu-b200 偶发失败 (test): 无法确定失败原因，疑似 runner 环境抖动或数值敏感性；从最终 PR 合并状态看，重跑后通过。
- flashinfer_deepgemm 首次获得真实 CI 覆盖 (design): 设计上通过按 SM 自适应后端集合解决该覆盖盲区。

# 风险与影响

- 风险：

1. 端到端覆盖缺失：删除三个 GSM8K *e2e* 后，量化后端在真实模型上的端到端精度不再被这些测试守护。PR body 已声明由模型 *e2e* 覆盖 CLI 接线、常规套件覆盖默认路径，但 per-backend 显式指定的精度回归（如 DeepGEMM 在特定模型上的精度问题）可能漏检。
2. 参考实现偏差：新测试使用自写的反量化参考（如 convert_swizzled_to_linear、dequantize_nvfp4_to_dtype），若参考本身有误（例如 layout 理解错误），测试会得出错误结论。测试用宽松容差（rtol=5e-2）和 cosine > 0.99 缓解，但无法覆盖所有 kernel 细节。
3. fp8.py 核心路径变更：validate_block_quant_shapes 是权重量化校验的公共路径，改动虽小（移动 tp_size 读取位置），但若未来有人依赖 skip 分支中的副作用会受影响；当前行为与之前非 skip 时完全一致，风险低。
4. CI 偶发失败：第二次 rerun 中 4-gpu-b200 出现过失败，需关注是否由新测试引入的资源竞争或 kernel 数值敏感导致。 - 影响：对 CI 基础设施影响显著：base-c、extra-a、extra-b 三个阶段合计减少约 926 秒的单次 PR 运行时间，且将原本从未执行的 flashinfer_deepgemm 后端纳入真实覆盖，补上了 flashinfer_cutlass FP8-blockwise 的盲区。对量化后端开发者的影响是调试手段的转变——从启动服务器跑 GSM8K 变为秒级数值断言，能更快定位权重预处理或 kernel dispatch 的错误。对 SGLang 仓库的测试方法论也有示范作用（与 #33611 的 MoE 层测试形成系列），后续新增 GEMM 后端可以沿用该模板补充覆盖。 - 风险标记：删除端到端精度覆盖，CI 新测试偶发失败，量化校验路径源码变更

# 关联脉络

- PR #33611 [Test] Replace NVFP4 MoE runner backend *e2e* matrix with a layer-level unit test: 同一作者、同一模式：将 *e2e* 矩阵替换为层级单元测试，本 PR 的 convert_swizzled_to_linear 即从该 PR 涉及的 test_fp4_moe.py 适配而来。
- PR #33605 [CI] Make B200 base-b suites single-GPU as prep for 1-gpu B200 runners: 同期 CI 优化，调整 B200 测试套件分布，与本 PR 的 CI 时间节省目标一致。
- PR #33586 [CI] Trim redundant B200 test registrations: 同样在精简 B200 相关测试注册，与本 PR 的 *e2e* 矩阵移除同属 CI 减负系列。