PR #28073 完整报告

sgl-project/sglang

fix: Fix DSR1 perf regression due to unnecessarily falling back to triton gemm

合并时间：2026-06-15 21:45

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/28073

执行摘要

- 一句话：修复 DSR1 fp8 gemm 回归：修正回退条件
- 推荐动作：建议阅读此 PR，尤其是 fp8_utils.py 中的回退逻辑变化和单元测试的设计。它展示了如何精细化控制 GEMM 后端选择，以及如何通过 benchmark 和测试验证性能回归修复。

功能与动机

PR #22300 修复了 MiniMax-M2.5 的精度准确性问题，但无意中导致 DeepSeek-R1-0528 性能显著下降（输出 token 吞吐率从 652.59 tok/s 降至 510.47 tok/s）。回退条件 `not getattr(weight_scale, "format_ue8m0", False)` 错误地要求权重 scale 具有 format_ue8m0 属性，而 DSR1 使用的 FP8 权重 scale 是普通 float32 类型，导致本应使用 TRTLLM 的层被降级到 triton。本 PR 旨在修正回退逻辑，使其基于输出 dtype（TRTLLM 仅支持 bf16）和 K 维度，消除回归。

实现拆解

1. 核心逻辑修正（python/sglang/srt/layers/quantization/fp8_utils.py）：在 flashinfer_gemm_w8a8_block_fp8_linear_with_fallback 函数中，将回退条件从 input_2d.shape[1] < 256 or not getattr(weight_scale, "format_ue8m0", False) 改为 input_2d.shape[1] < 256 or input_2d.dtype != torch.bfloat16。这样，对于 bf16 输出且 K≥256 的层，无论 scale 是否带有 format_ue8m0 属性，均使用 TRTLLM 内核；fp16 输出则会回退到 triton（因为 TRTLLM 仅正确支持 bf16）。
2. 移除无效代码（python/sglang/srt/model_loader/utils.py）：删除函数 post_load_weights，该函数在 PR #22300 的 rebase 中残留，功能已被其他机制替代。
3. 新增单元测试（test/registered/unit/layers/quantization/test_flashinfer_trtllm_fp8_fallback.py）：添加 TestFlashinferTrtllmFp8Fallback 类，包含 4 个测试用例：
   - test_bf16_uses_trtllm_with_plain_fp32_scales：验证 bf16 + K≥256 + 普通 float32 scale 使用 TRTLLM。
   - test_bf16_uses_trtllm_regardless_of_format_ue8m0：验证即使设置 format_ue8m0 属性，bf16 仍使用 TRTLLM。
   - test_fp16_falls_back_to_triton：验证 fp16 输出回退到 triton。

- test_small_k_falls_back_to_triton：验证 K<256 时回退到 triton。 测试通过 mock 所有 GEMM 实现，可在 CPU CI 上运行。

关键文件：

- test/registered/unit/layers/quantization/test_flashinfer_trtllm_fp8_fallback.py（模块 测试层；类别 test；类型 test-coverage；符号 TestFlashinferTrtllmFp8Fallback, _invoke, test_bf16_uses_trtllm_with_plain_fp32_scales, test_bf16_uses_trtllm_regardless_of_format_ue8m0）：新增单元测试文件，包含 4 个回归测试用例，确保回退条件正确，并通过 mock 在 CPU CI 上运行。
- python/sglang/srt/layers/quantization/fp8_utils.py（模块 量化层；类别 source；类型 core-logic；符号 flashinfer_gemm_w8a8_block_fp8_linear_with_fallback）：核心逻辑变更：修改了回退条件，使其基于 dtype 而非 format_ue8m0 属性。
- python/sglang/srt/model_loader/utils.py（模块 加载器；类别 source；类型 data-contract；符号 post_load_weights）：移除残留的无效函数 post_load_weights，减少维护负担。

关键符号：flashinfer_gemm_w8a8_block_fp8_linear_with_fallback, post_load_weights

关键源码片段

test/registered/unit/layers/quantization/test_flashinfer_trtllm_fp8_fallback.py

新增单元测试文件，包含 4 个回归测试用例，确保回退条件正确，并通过 mock 在 CPU CI 上运行。

```
"""Unit test for the FlashInfer TRTLLM block-FP8 GEMM fallback decision.

Regression guard for two coupled behaviors in
``flashinfer_gemm_w8a8_block_fp8_linear_with_fallback``:

* DeepSeek-R1 perf regression (commit 5da265de): with
  ``--fp8-gemm-backend flashinfer_trtllm`` the dense block-FP8 weight scales are
  plain float32 (they are NOT requantized to UE8M0 -- that only happens on the
  DeepGEMM dispatch path). The TRTLLM groupwise GEMM consumes float32 scales, so
  a bf16 layer must use the TRTLLM kernel, not fall back to triton. Gating the
  fallback on a ``format_ue8m0`` weight-scale attribute wrongly forced every such
  layer onto the slow triton path.
* MiniMax-M2.5 accuracy fix (PR #22300): the TRTLLM GEMM is only numerically
  correct for bf16 output, so fp16 output must fall back to triton.

So the fallback must key on output dtype and K (>= 256), independent of any
``format_ue8m0`` scale attribute. These tests pin that exactly, mocking the
backend selector and the two GEMM implementations so they run on CPU CI.
"""

from sglang.test.ci.ci_register import register_cpu_ci
register_cpu_ci(est_time=5, suite="base-a-test-cpu")
```

```
import unittest
from unittest.mock import MagicMock, patch
import torch
import sglang.srt.layers.quantization.fp8_utils as fp8_utils
from sglang.test.test_utils import CustomTestCase

BLOCK_SIZE = [128, 128]
M = 16
N = 512

class TestFlashinferTrtllmFp8Fallback(CustomTestCase):
    def _invoke(self, dtype, k, *, set_format_ue8m0=False):
        """Call the fallback dispatcher with backend pinned to 'trtllm'.
        Returns (triton_spy, trtllm_spy) so callers can assert which path ran.
        Every GEMM implementation is mocked, so no kernels actually execute."""
        input_2d = torch.zeros((M, k), dtype=dtype)
        weight = torch.zeros((N, k), dtype=torch.float32)
        weight_scale = torch.zeros((N // 128, k // 128), dtype=torch.float32)
        if set_format_ue8m0:
            # Pre-fix, this attribute is what gated the trtllm path. It must now
            # be irrelevant: a bf16 layer uses trtllm whether or not it is set.
            weight_scale.format_ue8m0 = True

        triton_spy = MagicMock(return_value=torch.zeros((M, N), dtype=dtype))
        trtllm_spy = MagicMock(return_value=torch.zeros((M, N), dtype=dtype))
        quant_spy = MagicMock(return_value=(MagicMock(), MagicMock()))

        with patch.object(
            fp8_utils, "_get_flashinfer_groupwise_backend", return_value="trtllm", create=True
        ), patch.object(
            fp8_utils, "gemm_fp8_nt_groupwise", trtllm_spy, create=True
        ), patch.object(
            fp8_utils, "triton_w8a8_block_fp8_linear", triton_spy
        ), patch.object(
            fp8_utils, "sglang_per_token_group_quant_fp8", quant_spy
        ):
            fp8_utils.flashinfer_gemm_w8a8_block_fp8_linear_with_fallback(
                input_2d, weight, BLOCK_SIZE, weight_scale
            )
        return triton_spy, trtllm_spy

    def test_bf16_uses_trtllm_with_plain_fp32_scales(self):
        """DeepSeek-R1 regression guard: bf16 + K>=256 + plain fp32 scales (no
        format_ue8m0) must use the trtllm GEMM, not fall back to triton."""
        triton_spy, trtllm_spy = self._invoke(torch.bfloat16, 512)
        trtllm_spy.assert_called_once()
        triton_spy.assert_not_called()

    def test_bf16_uses_trtllm_regardless_of_format_ue8m0(self):
```

```
        """format_ue8m0 must not affect the decision: bf16 still uses trtllm."""
        triton_spy, trtllm_spy = self._invoke(torch.bfloat16, 512, set_format_ue8m0=True)
        trtllm_spy.assert_called_once()
        triton_spy.assert_not_called()

    def test_fp16_falls_back_to_triton(self):
        """MiniMax-M2.5 accuracy guard: fp16 output must fall back to triton."""
        triton_spy, trtllm_spy = self._invoke(torch.float16, 512)
        triton_spy.assert_called_once()
        trtllm_spy.assert_not_called()

    def test_small_k_falls_back_to_triton(self):
        """K < 256 is unsupported by the trtllm GEMM and must fall back."""
        triton_spy, trtllm_spy = self._invoke(torch.bfloat16, 128)
        triton_spy.assert_called_once()
        trtllm_spy.assert_not_called()

if __name__ == "__main__":
    unittest.main(verbosity=3)
```

python/sglang/srt/layers/quantization/fp8_utils.py

核心逻辑变更：修改了回退条件，使其基于 dtype 而非 format_ue8m0 属性。

```
# Before (base):
def flashinfer_gemm_w8a8_block_fp8_linear_with_fallback(
    input: torch.Tensor, weight: torch.Tensor, block_size: List[int],
    weight_scale: torch.Tensor, input_scale: Optional[torch.Tensor] = None,
    bias: Optional[torch.Tensor] = None) -> torch.Tensor:
    # ...
    if backend == "trtllm" and (
        input_2d.shape[1] < 256 or not getattr(weight_scale, "format_ue8m0", False)
    ):
        return triton_w8a8_block_fp8_linear(...)

# After (head):
def flashinfer_gemm_w8a8_block_fp8_linear_with_fallback(
    input: torch.Tensor, weight: torch.Tensor, block_size: List[int],
    weight_scale: torch.Tensor, input_scale: Optional[torch.Tensor] = None,
    bias: Optional[torch.Tensor] = None) -> torch.Tensor:
    # ...
    # Fall back to triton for non-supported formats.
    # TODO: Check if flashinfer supports other output dtypes besides bf16.
    if backend == "trtllm" and (
        input_2d.shape[1] < 256 or input_2d.dtype != torch.bfloat16
    ):
        return triton_w8a8_block_fp8_linear(...)
```

评论区精华

本 PR 无审查评论；维护者 b8zhong 直接批准。PR body 中包含详细的性能 benchmark，展示了回归前后的对比以及修复后的恢复情况。

- 暂无高价值评论线程

风险与影响

- 风险：风险较低。关键变更仅在 flashinfer_trtllm 后端路径中修改了回退条件，并添加了单元测试覆盖。潜在风险：若有其他模型依赖旧的 format_ue8m0 属性进行正确回退，但根据注释仅 DeepGEMM 路径使用该属性，本变更不影响该路径。FP16 输出回退到 triton 可能引入精度差异，但这是 PR #22300 中已确认的正确行为。
- 影响：正面影响：恢复 DeepSeek-R1-0528 在使用 flashinfer_trtllm 时的性能（约 28% 吞吐提升）。影响范围限于使用 --fp8-gemm-backend flashinfer_trtllm 的用户，且模型权重 scale 为普通 float32 的情况（如 DSR1）。无破坏性变更。
- 风险标记：潜在遗漏：若未来其他模型依赖 format_ue8m0 进行回退，需重新评估；当前变更仅影响 trtllm 后端路径

关联脉络

- PR #22300 Fix MiniMax-M2.5 fp8 accuracy: 该 PR 引入了本 PR 修复的回归问题：错误地使用 format_ue8m0 属性作为回退条件。