PR #34506 完整报告

sgl-project/sglang

[Diffusion] Make weight-only FP8 dequant cache torch.compile-safe

合并时间：2026-08-12 16:26

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34506

执行摘要

- 一句话：修复 FP8 反量化缓存与 torch.compile 不兼容
- 推荐动作：值得精读。改动虽小，但展示了一个典型的 PyTorch 机制坑位——inference_mode 产物会丢失版本计数器，进而影响 torch.compile 的 guard 建立；其修复方式（局部退出 inference_mode + no_grad）和回归测试写法（先 promotion 再 fullgraph 编译并对比 bit-exact）都值得在后续缓存类功能中复用。

功能与动机

PR body 说明：weight-only FP8 cache 首次使用时将每个 FP8 权重提升为持久化 compute-dtype 参数；服务器 warmup 运行在 torch.inference_mode() 下，导致缓存参数成为无版本计数器的 inference tensor，后续 torch.compile 追踪时抛出 RuntimeError: Inference tensors do not track version counter。

实现拆解

1. 定位根因：在 python/sglang/multimodal_gen/runtime/layers/quantization/weight_only_fp8.py 的 _maybe_promote_fp8_weight 中，首次 forward 会调用 dequantize_rowwise_fp8_weight 生成反量化权重并替换 module.weight。由于 warmup 处于 torch.inference_mode()，新生成的 tensor 也是 inference tensor，没有版本计数器，Dynamo 无法对其建立 guard。
2. 修改方案：将反量化操作放入 with torch.inference_mode(False), torch.no_grad(): 上下文，使生成的参数是普通可追踪 tensor 且不参与梯度图；其余逻辑（缓存开关、低内存 fallback、W8A8 直通路径）保持不变。
3. 回归测试：在 test/unit/test_weight_only_fp8_dequant_cache.py 中新增 test_inference_mode_promotion_supports_torch_compile，先以 inference_mode 触发提升并断言 weight.is_inference() 为 False，再通过 torch.compile(fullgraph=True) 执行并验证 eager/compiled 输出与参考实现 bit-identical；同时测试基类改为 CustomTestCase 以接入仓库统一测试设施。
4. 实机验证：B300 单元套件 6 项通过；Ideogram-4 FP8 compile preset 在 2xB300 上 warmup 完成，真实 20 步请求 denoise 2.202 s、E2E 2.314 s，且 eager 与 compiled 输出 bit-identical。

关键文件：

- python/sglang/multimodal_gen/runtime/layers/quantization/weight_only_fp8.py（模块量化层；类别 source；类型 core-logic；符号 _maybe_promote_fp8_weight）：核心修复点：在 _maybe_promote_fp8_weight 中将反量化物化移入 torch.inference_mode(False) 与 torch.no_grad()，确保缓存参数不是 inference tensor，从而支持后续 torch.compile。
- python/sglang/multimodal_gen/test/unit/test_weight_only_fp8_dequant_cache.py（模块测试套件；类别 test；类型 test-coverage；符号 TestWeightOnlyFP8DequantCache, test_inference_mode_promotion_supports_torch_compile）：新增回归测试覆盖 inference_mode 提升后 torch.compile(fullgraph=True) 路径，并断言缓存参数不是 inference tensor；测试基类改为 CustomTestCase。

关键符号：_maybe_promote_fp8_weight, test_inference_mode_promotion_supports_torch_compile

关键源码片段

python/sglang/multimodal_gen/runtime/layers/quantization/weight_only_fp8.py

核心修复点：在 _maybe_promote_fp8_weight 中将反量化物化移入 torch.inference_mode(False) 与 torch.no_grad()，确保缓存参数不是 inference tensor，从而支持后续 torch.compile。

```
# _maybe_promote_fp8_weight 的核心决策片段：
# 决定是否将 FP8 权重一次性反量化为计算精度并缓存
weight = module.weight
if weight.dtype != FP8_WEIGHT_DTYPE:
    module._fp8_dequant_decided = True
    return

if weight.device.type != "cuda":
    return

if torch.cuda.is_current_stream_capturing():
    # 不能在 CUDA graph capture 内分配内存，留待 eager 调用时再处理
    return

module._fp8_dequant_decided = True
if not envs.SGLANG_DIFFUSION_FP8_WEIGHT_DEQUANT_CACHE:
    return

if module.enable_fused_w8a8:
    # W8A8 GEMM 路径直接消费 FP8 权重，无需反量化
    return

dtype = module.compute_dtype or x_dtype
if dtype not in (torch.float16, torch.bfloat16, torch.float32):
    return

needed_bytes = weight.numel() * dtype.itemsize
```

```
    free_bytes, _ = torch.cuda.mem_get_info(weight.device)
    if free_bytes < needed_bytes + _DEQUANT_CACHE_RESERVE_BYTES:
        # 显存不足时保持 FP8 驻留，每次前向都反量化
        if not _dequant_low_memory_logged:
            logger.warning(
                "Keeping weight-only FP8 linear weights FP8-resident (low "
                "free device memory); they dequantize on every forward."
            )
            _dequant_low_memory_logged = True
        return

    # 服务器 warmup 常在 inference_mode 下运行，若在此模式下物化缓存参数，
    # 生成的 tensor 会丢失版本计数器，导致后续 torch.compile 无法建立 guard。
    # 因此临时退出 inference_mode 并关闭梯度，得到普通可追踪的参数。
    with torch.inference_mode(False), torch.no_grad():
        dequant = dequantize_rowwise_fp8_weight(weight, module.weight_scale, dtype)

    module.weight = nn.Parameter(dequant, requires_grad=False)
```

python/sglang/multimodal_gen/test/unit/test_weight_only_fp8_dequant_cache.py

新增回归测试覆盖 inference_mode 提升后 torch.compile(fullgraph=True) 路径，并断言缓存参数不是 inference tensor；测试基类改为 CustomTestCase。

```
class TestWeightOnlyFP8DequantCache(CustomTestCase):
    # 回归测试：inference_mode 下的提升必须与后续 torch.compile 兼容
    @unittest.skipUnless(torch.cuda.is_available(), "requires CUDA")
    def test_inference_mode_promotion_supports_torch_compile(self):
        linear = _make_linear(torch.device("cuda"))
        x = torch.randn(8, 64, device="cuda", dtype=torch.bfloat16)
        reference = _reference(linear, x)

        # 模拟服务器 warmup 的 inference_mode 路径触发 FP8 提升
        with torch.inference_mode():
            eager_out = linear(x)
        # 修复后缓存参数不应是 inference tensor，否则 Dynamo 无法建立 guard
        self.assertFalse(linear.weight.is_inference())

        # 再走 torch.compile(fullgraph=True)，确保编译追踪不抛错
        compiled = torch.compile(linear, fullgraph=True)
        with torch.inference_mode():
            compiled_out = compiled(x)
        self.assertTrue(torch.equal(reference, eager_out))
        self.assertTrue(torch.equal(reference, compiled_out))
```

评论区精华

该 PR 无 reviewer 评论，issue 评论仅包含作者触发的 /tag-and-rerun-ci 指令和一次 CI 运行链接，没有出现设计或正确性层面的交锋。

- 暂无高价值评论线程

风险与影响

- 风险：
  - 行为变更：torch.inference_mode(False) 可能让 dequant 运算在普通 eager 模式下记录置位，但外层 torch.no_grad() 保证不追踪梯度，输出仍 bit-identical；已有的 is_current_stream_capturing() 检查可避免在 CUDA graph capture 中触发。
  - 性能影响：仅改变张量创建时的模式标注，不改变反量化算法与缓存策略，低内存 fallback 和 W8A8 路径完全绕过该分支，性能风险极低。
  - 兼容性：改动集中在单个函数，未触碰 dequantize_rowwise_fp8_weight 与 swap_linears_to_weight_only_fp8，对既有 eager 推理用户透明。
  - 测试盲区：新增测试只覆盖 CUDA 上的 bfloat16 输入，未覆盖 fp16/fp32 或 CPU compile 场景，但仓库现有 test_promotion_dtype_follows_input_when_unset 部分覆盖 dtype 分支。
  - 影响：对使用 diffusion 模型 + FP8 weight-only 量化 + torch.compile 的用户，此前 warmup 后编译必现的 RuntimeError 被消除；对 eager 路径无行为变化，输出保持 bit-identical。对团队而言，确立了一个值得复用的模式：持久化缓存参数必须在非 inference_mode 下物化，并配套 fullgraph 回归测试；该 PR 也验证了 Ideogram-4 FP8 compile 端到端可用。
  - 风险标记：缓存参数物化模式变更，torch.compile 兼容性回归风险

关联脉络

- PR #34412 [Diffusion] Improve bit-exact fusion fallback diagnostics: 同属 diffusion 量化路径，涉及 FP8 与融合回退的诊断和回退逻辑，本 PR 的修改与 FP8 权重驻留 / 缓存策略处在同一功能线。
- PR #34314 [Diffusion] Ideogram-4: fuse Qwen3-style RoPE and SwiGLU silu-mul: 同样面向 Ideogram-4 FP8 路径做性能与兼容性优化，本 PR 以 Ideogram-4 FP8 compile preset 作为端到端验证场景。