PR #35883 完整报告

sgl-project/sglang

Fix stale GLM MoE routing after runtime weight updates

合并时间：2026-08-31 05:13

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35883

执行摘要

- 一句话：修复 GLM MoE 路由过期：门控权重改存 FP32
- 推荐动作：值得精读。核心价值在于“消除重复状态”的设计决策：与其为 BF16 参数维护 FP32 影子缓存并不断补刷新钩子（weight_loader + post_direct_write），不如让单一 FP32 参数成为唯一事实来源，类型转换统一交给加载器。这一取舍对任何需要支持运行时权重热更新的模型实现都有借鉴意义；同时值得关注 WeightChecker 过滤契约的收紧会给 RL 链路带来的行为变化。

功能与动机

GLM MoE 路由要求 FP32 投影，原实现用 BF16 参数加 FP32 影子缓存来避免每次 forward 重复 cast，但代码中留有 FIXME 注释明确警告：运行时权重更新后 _weight_fp32 必须失效而未实现。PR body 指出：Keeping a BF16 gate parameter plus an FP32 shadow duplicated state and allowed runtime weight updates to leave routing on stale weights。RL 训练通过 update_weights_from_tensor 同步权重，正是触发该 bug 的主要路径，会使路由长期使用最初加载的旧权重。

实现拆解

按实现演进与最终落地方案拆解如下：

1. 方案迭代收敛（提交历史）：8 个 commit 呈现三次方案演进。初版给 gate.weight 挂 weight_loader，在运行时更新时原地刷新 _weight_fp32 缓存（CUDA graph 捕获 buffer 地址，不能重新分配）；第二版针对 Codex 评审指出的 direct 加载绕过问题，在 _model_load_weights_direct 中增加 opt-in post_direct_write 钩子；第三版（commit 7535960）放弃双状态：门控权重直接以 FP32 存储，删除缓存、钩子及相关 weight-checker 改动，从根上消除一致性维护负担。
2. 模型层改动：python/sglang/srt/models/glm4_moe.py 的 Glm4MoeGate 与 python/sglang/srt/models/glm4_moe_lite.py 的 Glm4MoeLiteGate 中，self.weight = nn.Parameter(torch.empty((config.n_routed_experts, config.hidden_size), dtype=torch.float32)) 显式声明 FP32；forward 直接执行 F.linear(hidden_states.to(torch.float32), self.weight, None)，删除 _weight_fp32 非持久 buffer 与首次 forward 惰性缓存分支。类型转换职责移交给加载器：BF16 checkpoint 与运行时 BF16 权重在 default_weight_loader copy 进参数时统一 cast 为 FP32。

3. WeightChecker 契约变更：python/sglang/srt/utils/weight_checker.py 的 _NON_PERSISTENT_BUFFER_PATTERNS 删除 _weight_fp32 模式。后果是名称含该子串的 buffer（如 gate_proj_weight_fp32_cache，因子串匹配而受影响）不再被跳过，snapshot / reset / compare / checksum 全流程恢复正常检测；_reset_tensors 会原地随机化这类 buffer，测试语义从“跳过”改为“毒化”。
4. 测试配套：新增 test/registered/unit/models/test_glm_moe_gate_fp32.py（注册到 CPU CI，register_cpu_ci(est_time=4)），验证 BF16 默认 dtype 下构造 gate 权重仍为 FP32、_model_load_weights_direct 原地更新后 data_ptr 不变、路由输出与 FP32 线性参考一致；更新 test/registered/unit/utils/test_weight_checker.py 与 test/registered/rl/test_weight_checker_e2e.py 以匹配新过滤契约，并删除两份针对私有 helper 的冗余断言。

关键文件：

- python/sglang/srt/models/glm4_moe.py（模块 MoE 路由；类别 source；类型 data-contract；符号 Glm4MoeGate, Glm4MoeGate.init, Glm4MoeGate.forward）：核心修复文件：Glm4MoeGate 的权重从 BF16 参数 + FP32 影子缓存改为 FP32 单一参数，forward 直接使用 canonical 权重做 FP32 路由投影，删除 FIXME 标记的缓存失效隐患。
- python/sglang/srt/models/glm4_moe_lite.py（模块 MoE 路由；类别 source；类型 data-contract；符号 Glm4MoeLiteGate, Glm4MoeLiteGate.init, Glm4MoeLiteGate.forward）：与 glm4_moe.py 同构的修复：Glm4MoeLiteGate 同样从 BF16 + FP32 影子缓存改为 FP32 单一参数，保持 GLM Lite 系列行为一致。
- test/registered/unit/models/test_glm_moe_gate_fp32.py（模块 回归测试；类别 test；类型 test-coverage；符号 TestGlmMoeGateFp32Weight, test_bf16_load_updates_fp32_weight_in_place）：新增 CPU 回归测试，覆盖两种 gate 在 BF16 加载与运行时更新后的 FP32 存储、原地更新与投影一致性，是本次修复的核心验证。
- python/sglang/srt/utils/weight_checker.py（模块 权重检查；类别 source；类型 core-logic；符号 _is_non_persistent_buffer_name, _NON_PERSISTENT_BUFFER_PATTERNS）：WeightChecker 的过滤契约变更点：从 _NON_PERSISTENT_BUFFER_PATTERNS 删除 _weight_fp32，使含该子串的 buffer 恢复被 snapshot / reset / checksum 正常检测。
- test/registered/unit/utils/test_weight_checker.py（模块 权重检查；类别 test；类型 test-coverage；符号 test_skips_weight_fp32_substring, test_skips_weight_fp32, test_poisons_weight_fp32_cache, test_matches_weight_fp32_substring）：配套更新 WeightChecker 单元测试：删除针对 _weight_fp32 的 skip 断言，新增毒化语义与 checksum 包含断言，验证新过滤契约。
- test/registered/rl/test_weight_checker_e2e.py（模块 权重检查；类别 test；类型 test-coverage）：e2e 测试同步删除 _weight_fp32 的 skip 断言，确保端到端 checksum 行为与新的过滤契约一致。

关键符号：Glm4MoeGate.init, Glm4MoeGate.forward, Glm4MoeLiteGate.init, Glm4MoeLiteGate.forward, _is_non_persistent_buffer_name, test_bf16_load_updates_fp32_weight_in_place

关键源码片段

python/sglang/srt/models/glm4_moe.py

核心修复文件：Glm4MoeGate 的权重从 BF16 参数 + FP32 影子缓存改为 FP32 单一参数，forward 直接使用 canonical 权重做 FP32 路由投影，删除 FIXME 标记的缓存失效隐患。

```
class Glm4MoeGate(nn.Module):
    """GLM MoE 路由门控：权重直接以 FP32 存储，避免影子缓存失效问题。"""

    def __init__(self, config, prefix: str = ""):
        super().__init__()
        # 权重显式声明为 FP32：BF16 checkpoint 和运行时更新均由加载器
        # 在 copy 进参数时完成类型转换，forward 不再需要维护额外缓存。
        self.weight = nn.Parameter(
            torch.empty(
                (config.n_routed_experts, config.hidden_size),
                dtype=torch.float32,
            )
        )
        self.e_score_correction_bias = nn.Parameter(
            torch.empty((config.n_routed_experts), dtype=torch.float32)
        )

    def forward(self, hidden_states):
        # 路由投影要求 FP32 输入与权重。hidden_states 每次 cast，
        # weight 本身已是 FP32，直接参与线性投影，无过期缓存风险。
        logits = F.linear(hidden_states.to(torch.float32), self.weight, None)
        return logits
```

test/registered/unit/models/test_glm_moe_gate_fp32.py

新增 CPU 回归测试，覆盖两种 gate 在 BF16 加载与运行时更新后的 FP32 存储、原地更新与投影一致性，是本次修复的核心验证。

```
class TestGlmMoeGateFp32Weight(CustomTestCase):
    def test_bf16_load_updates_fp32_weight_in_place(self):
        """BF16 加载 / 运行时更新必须原地覆盖 FP32 门控权重。"""
        hidden_states = torch.arange(8, dtype=torch.bfloat16).reshape(2, 4)
        initial = torch.arange(12, dtype=torch.bfloat16).reshape(3, 4)
        updated = initial + 1

        for gate_cls in (Glm4MoeGate, Glm4MoeLiteGate):
            with self.subTest(gate=gate_cls.__name__):
                with set_default_torch_dtype(torch.bfloat16):
                    gate = gate_cls(_CONFIG)

                # 即使在 BF16 默认 dtype 下构造，门控权重也必须保持 FP32。
                self.assertEqual(gate.weight.dtype, torch.float32)
                # 旧的 _weight_fp32 影子缓存已被移除，避免重复状态。
                self.assertFalse(hasattr(gate, "_weight_fp32"))
```

```
weight_ptr = gate.weight.data_ptr()

# BF16 checkpoint 加载：default_weight_loader 在 copy 时 cast 成 FP32。
default_weight_loader(gate.weight, initial)
torch.testing.assert_close(gate.weight, initial.float())

# 运行时权重更新（RL 权重同步）：原地写入，指针不变。
_model_load_weights_direct(gate, [("weight", updated)])
self.assertEqual(gate.weight.data_ptr(), weight_ptr)
torch.testing.assert_close(gate.weight, updated.float())

# 路由输出与 FP32 线性参考一致，证明没有 stale 状态。
torch.testing.assert_close(
    gate(hidden_states),
    F.linear(hidden_states.float(), updated.float()),
)
```

python/sglang/srt/utils/weight_checker.py

WeightChecker 的过滤契约变更点：从 _NON_PERSISTENT_BUFFER_PATTERNS 删除 _weight_fp32，使含该子串的 buffer 恢复被 snapshot / reset / checksum 正常检测。

```
# 非持久 buffer 的过滤模式：这些名字的 buffer 不参与权重篡改检测。
# 之前包含 "_weight_fp32"，因为 GLM gate 用它做 FP32 影子缓存；
# 门控权重改存 FP32 后此模式已删除，避免误跳过真实权重。
_NON_PERSISTENT_BUFFER_PATTERNS = (
    "cos_sin_cache",
    "inv_freq",
    "freqs_cis",
    "expert_mask_gpu",
)


def _is_non_persistent_buffer_name(name: str) -> bool:
    # 子串匹配，模式出现在名称任意位置即视为非持久缓存。
    return any(pat in name for pat in _NON_PERSISTENT_BUFFER_PATTERNS)
```

评论区精华

核心讨论围绕“如何让 FP32 缓存与运行时权重更新保持一致”展开：

- Codex 机器人（P2）指出 direct 加载路径绕过缓存刷新：update_weights_from_tensor(load_format="direct") 走 _model_load_weights_direct 直接调用 default_weight_loader，不经过 param.weight_loader，因此初版方案下更新 mlp.gate.weight 后 _weight_fp32 仍保留旧路由权重。
- 作者回应并修复：direct 路径保持 raw-copy 语义是对的——若改走 param.weight_loader，对 fused qkv_proj 等 mapping loader 是错误的，因为后者期望 checkpoint 格式张量与 shard id；因此改为在 _model_load_weights_direct 增加 opt-in post_direct_write 钩子。

- guapisolo 对实现提出风格意见：对 getattr(param, "post_direct_write", None) 直接回复“no getattr”。该代码在后来的方向性重构中被整体删除，未形成争议。
- 最终方案获得 rwang5203、JustinTong0323、Fridge003 的 approve，其中 JustinTong0323 提到 rebase 是为了绕过 main 上既有的 CI 失败。
- direct 加载路径绕过 weight_loader 导致 FP32 缓存过期 (correctness): 作者在 cd36066 为 direct 路径增加 opt-in post_direct_write 钩子；最终方案改为 gate 权重直接存 FP32，彻底移除缓存与钩子，问题从根上消失。
- direct 路径为何不能复用 mapping loader 语义 (design): 设计被 reviewer 接受；该钩子在后来的 FP32 单一参数方案中被整体删除，但讨论澄清了 direct 路径与 mapping loader 的边界。
- post_direct_write 获取方式的风格意见 (style): 该代码在后续方向性重构中被整体删除，未产生进一步争议。

风险与影响

- 风险：
  1. 显存与精度：门控权重从 BF16 改为 FP32，参数显存翻倍；但门控矩阵很小（n_routed_experts × hidden_size 量级，GLM-4 约每层 4 MB），且同时删除了等大的 FP32 缓存，净占用基本持平。BF16 转 FP32 无损，路由精度只会更稳。
  2. WeightChecker 行为变化：删除 _weight_fp32 过滤模式后，任何名称含该子串的 buffer 都会进入 checksum 与 reset 流程。当前仅 GLM gate 系列使用该命名，但若第三方模型或后续代码沿用此命名，行为会静默改变，需要回归验证。
  3. 数值一致性：新实现与原实现的 FP32 投影数学等价，但由于权重不再经过“BF16 存参 + 首次 forward cast”路径，输出不再与旧版本 bit-exact 一致，涉及精度对齐的测试需留意。
  4. 验证缺口：PR 自述未运行模型级精度基准与速度测试（Accuracy Tests 与 Speed Tests 均标注 Not run），仅靠 CPU 单元回归保障；建议合入后在 GPU 上对 GLM-4 系列做一次路由输出与端到端精度抽检。
- 影响：
  1. 用户侧：GLM-4 / GLM-4-Lite 部署且使用运行时权重更新（RL 在线权重同步、专家再平衡）的用户是直接受益者，路由不再使用过期权重；普通静态加载推理不受影响。
  2. 系统侧：WeightChecker 的 snapshot / reset / compare / checksum 对 weight_fp32 命名 buffer 的过滤契约改变，影响依赖该工具做权重篡改检测的 RL 训练运维流程，但这是向“更完整检测”方向的收紧。
  3. 团队侧：消除了代码中明确标注的 FIXME 技术债，门控模块状态从“双状态 + 手动同步”简化为“单一 FP32 事实来源”，后续维护成本显著降低。 - 风险标记：模型权重契约变更，WeightChecker 过滤行为收紧，无 GPU 精度基准验证

关联脉络

- 暂无明显关联 PR