PR #32039 完整报告

sgl-project/sglang

[AMD][Fix] Route MoRI through the Qwen MoE all-to-all path

合并时间：2026-08-25 04:04

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/32039

执行摘要

- 一句话：修复 MoRI 下 Qwen MoE 路由错乱与精度归零
- 推荐动作：值得精读。核心看点：(1) 后端谓词必须与布局契约一一对应 — is_deepep() 只在纯 DeepEP 为真，MoRI 这类 DeepEP 类后端必须显式纳入，否则静默落入错误路径；(2) Duyi-Wang 对 CUDA Graph 与动态 token 数的辩驳展示了先验证假设、再改代码的归因方法，yichiche 用 eager/graph 对照 + 吞吐 + accept length 三组证据收缩 PR 范围，是很好的 review 实践样本；(3) 测试用全局 flags 模拟后端切换并 addCleanup 恢复的模式可复用。建议后续补一个真正跑 MoRI 运行时路径的 PR-CI 测试，避免回归。

功能与动机

PR body 开宗明义：Serving Qwen3.5 MoE (Qwen3.5-397B-A17B-MXFP4) with --moe-a2a-backend mori on MI355X/ROCm (DP=1 / TP=2, EP=2) produced near-zero accuracy — gsm8k 0.006。根因是 Qwen2/Qwen3 的 MoE 门控只识别 get_moe_a2a_backend().is_deepep()，而 MoRI 是 DeepEP 类后端，需要同样的 per-rank EP 专家布局；在 MoRI 下代码块静默落入纯 TP 路径：共享专家保持 TP=2 分片只算一半、can_fuse_shared_expert 仍允许融合进全局 slot（per-rank EP 层从不分配）、且从不进入 _forward_deepep()，路由专家完全没走 MoRI dispatch/combine，被破坏的首个 token 会污染后续所有生成。本 PR 是 #31793 的 correctness follow-up（#31793 仅禁用了全局 slot 的共享专家融合以让 MoRI 通过启动崩溃）。

实现拆解

1. 变更入口与四个门控补全

在 python/sglang/srt/models/qwen2_moe.py 中把 MoRI 纳入与 DeepEP 相同的处理分支，共四处：

- can_fuse_shared_expert()：融合拒绝条件追加 get_moe_a2a_backend().is_mori()，防止共享专家被融合进全局 slot（EP 布局下各 rank 从不分配该 slot，路由会读错行）；
- Qwen2MoeExperts.__init__中 shared_expert 的构造：dict(tp_rank=0, tp_size=1) 的条件追加 is_mori()，让共享专家按 rank 完整复制，每个 token 都拿到完整共享专家输出；
- 同一构造函数的 EP 布局分支：由 if is_deepep() 扩为 is_deepep() or is_mori()，正确设置 ep_size、num_experts（含冗余专家）与 top_k；

- Qwen2MoeForCausalLM.forward 的分发：扩为 is_deepep() or is_mori() 后进入 _forward_deepep()，让路由专家真正走 MoRI dispatch/combine。

2. 谓词范围收敛

commitf2b92d2把最初的is_deepep_class_backend()收敛为显式is_deepep()oris_mori()。原因是 is_deepep_class_backend() 还覆盖 Mooncake、PPLX，这两个后端没有 Qwen MoE 覆盖验证，直接改动会静默改变 CUDA 侧行为；显式谓词保证本 PR 对 main 在非 MoRI 后端的路径字节级一致。

3. 撤销两个错误归因的变更

早期提交（769dab2）包含三个修复，On-hardware 复验后证明其中两个是错误归因：

- *decode* CUDA graph 禁用（server_args.py）：通过 *eager*/ 图形对照（broken gates 下 *eager* 与 graph 均为 0.000；修复后 graph 0.967；*eager* 0.935 与 graph 0.932 在 parallel 200 下高度一致且吞吐 2.2 倍）证明 graph 重放本身不丢路由；且禁用 graph 会把 EAGLE MTP *accept* length 从 3.47-3.59 降到 2.59-2.91，因此移除；
- AITER *eager*-metadata buffer 容量调整（aiter_backend.py）：Target sizes: [111] / Tensor sizes: [112] 崩溃不再复现，剩余高并发 Memory *access* fault 发生在 mamba usage: 1.00 的线性注意力状态池耗尽场景，属另一 bug，单独跟踪。 PR 最终收敛为纯路由修复（commit 7cfc1f2）。

4. 测试配套

- 新增 TestA2ABackendGate（test/registered/unit/models/test_shared_experts_fusion_gates.py）：通过 _use_backend 临时改写 get_flags().moe.a2a_backend（addCleanup 恢复），断言 *can*_fuse_shared_expert 对 *deepep*、mori 返回 False，对纯 TP（none）返回 True；
- 修复 CI 收集：早期新增的 AITER 测试文件缺 if name== "main"，会导致 AMD stage-a 在收集阶段整体失败（commit de1cad0 修正，随后该文件随 AITER 变更一起移除）；
- 保持 register_cpu_ci 的 *est*_time 不动（commit 18*eedaa*），让本 PR diff 纯新增。

5. 验证数据（MI355X，TP2/EP2，MoRI normal，*decode* graph 开启）

场景	精度 / 吞吐
修复前	gsm8k 0.006
60 问 / 并行 30	0.967
400 问 / 并行 64	0.925
1000 问 / 并行 200	0.932，1552.6 tok/s
EAGLE MTP 200 问 / 并行 30	0.935，*accept* ~3.5

关键文件：

- python/sglang/srt/models/qwen2_moe.py（模块 MoE 模型；类别 source；类型 core-logic；符号 can_fuse_shared_expert, Qwen2MoeExperts.init, Qwen2MoeForCausalLM.forward）：核心源码文件：在 can_fuse_shared_expert、共享专家 TP 复制、EP 布局与 forward 分发四个门控点追加 is_mori()，是修复 MoRI 路由错乱的关键。
- test/registered/unit/models/test_shared_experts_fusion_gates.py（模块 融合门控；类别 test；类型 test-coverage；符号 TestA2ABackendGate, test_the_a2a_backends_refuse_fusion, test_a_plain_tp_deployment_still_fuses）：新增 TestA2ABackendGate 单元测试，pin 住 can_fuse_shared_expert 对 deepep/mori 拒绝融合、纯 TP 仍融合的契约，是防止回归的关键配套。

关键符号：can_fuse_shared_expert, Qwen2MoeExperts.init, Qwen2MoeForCausalLM.forward, TestA2ABackendGate.test_the_a2a_backends_refuse_fusion, TestA2ABackendGate.test_a_plain_tp_deployment_still_fuses

关键源码片段

python/sglang/srt/models/qwen2_moe.py

核心源码文件：在 can_fuse_shared_expert、共享专家 TP 复制、EP 布局与 forward 分发四个门控点追加 is_mori()，是修复 MoRI 路由错乱的关键。

```
# python/sglang/srt/models/qwen2_moe.py —— 本 PR 整理后的关键实现片段
# 背景：MoRI 是与 DeepEP 同类的 per-rank EP 布局后端，但 Qwen2/Qwen3 门控此前
# 只识别 is_deepep()，导致 MoRI 下 MoE 层落入纯 TP 路径，路由从不经过 dispatch/combine。

def can_fuse_shared_expert(config, quant_config):
    # 融合会把共享专家当作额外 MoE 专家塞进全局 slot；EP 布局下每个 rank 只分配本地
    # slot，全局 slot 永远不会被读写，路由会读到错误行，精度崩坏，因此必须拒绝。
    if (
        get_exec().moe.disable_shared_experts_fusion is True
        or getattr(config, "shared_expert_intermediate_size", 0) <= 0
        or config.shared_expert_intermediate_size != config.moe_intermediate_size
        or get_moe_a2a_backend().is_deepep()
        or get_moe_a2a_backend().is_mori() # 本 PR 新增：MoRI 同样拒绝融合
    ):
        return False
    # 其余量化相关检查保持不变
    ...

class Qwen2MoeExperts(nn.Module):
    def __init__(self, ...):
        ...
        # DeepEP/MoRI 下共享专家按 rank 完整复制（tp_rank=0, tp_size=1），
        # 保证每个 token 都能拿到完整的共享专家贡献，而不是 TP 分片的一半。
        if config.shared_expert_intermediate_size > 0 and not self.enable_shared_expert_fusion:
            self.shared_expert = Qwen2MoeMLP(
                ...,
                **(
```

```
                dict(tp_rank=0, tp_size=1)
                if (
                    get_moe_a2a_backend().is_deepep()
                    or get_moe_a2a_backend().is_mori() # 本 PR 新增
                    or get_moe_a2a_backend().is_flashinfer()
                )
                else {}
            ),
        )
    ...
    # EP 布局：每个 rank 只持有本地专家，需要按 EP 大小重算 num_experts（含冗余）
    if get_moe_a2a_backend().is_deepep() or get_moe_a2a_backend().is_mori():
        self.ep_size = get_parallel().moe_ep_size
        self.num_experts = config.num_experts + get_exec().moe.ep_num_redundant_experts
        self.top_k = config.num_experts_per_tok

def forward(self, hidden_states, forward_batch):
    ...
    # 路由专家必须走 MoRI dispatch/combine，而不是退化为本地 TP 专家计算
    if get_moe_a2a_backend().is_deepep() or get_moe_a2a_backend().is_mori():
        return self._forward_deepep(hidden_states, forward_batch)
```

test/registered/unit/models/test_shared_experts_fusion_gates.py

新增 TestA2ABackendGate 单元测试，pin 住 can_fuse_shared_expert 对 deepep/mori 拒绝融合、纯 TP 仍融合的契约，是防止回归的关键配套。

```
# test/registered/unit/models/test_shared_experts_fusion_gates.py —— 本 PR 新增用例
class TestA2ABackendGate(_FusionGateCase):
    """can_fuse_shared_expert 必须对每个 DeepEP 类后端拒绝融合。

    MoRI 与 DeepEP 使用相同的 per-rank EP 专家布局，融合的共享专家会占据
    各 rank 层永远不会分配的全局 slot，导致路由读错行、精度崩坏。
    """

    def _config(self):
        return SimpleNamespace(
            model_type="qwen3_5_moe_text",
            shared_expert_intermediate_size=1024,
            moe_intermediate_size=1024,
        )

    def _use_backend(self, name: str):
        from sglang.srt.layers.moe.utils import MoeA2ABackend
        from sglang.srt.runtime_context import get_flags

        # 通过全局 flags 模拟后端切换，并在用例结束后恢复原值，避免污染其他测试
        moe = get_flags().moe
        previous = moe.a2a_backend
        moe.a2a_backend = MoeA2ABackend(name)
```

```
        self.addCleanup(setattr, moe, "a2a_backend", previous)

    def test_the_a2a_backends_refuse_fusion(self):
        from sglang.srt.models.qwen2_moe import can_fuse_shared_expert

        self._seed()
        for backend in ("deepep", "mori"):
            with self.subTest(backend=backend):
                self._use_backend(backend)
                self.assertFalse(can_fuse_shared_expert(self._config(), None))

    def test_a_plain_tp_deployment_still_fuses(self):
        from sglang.srt.models.qwen2_moe import can_fuse_shared_expert

        self._seed()
        self._use_backend("none")
        self.assertTrue(can_fuse_shared_expert(self._config(), None))
```

评论区精华

Review 核心交锋集中在 server_args.py 的 *decode* CUDA graph 禁用上。yichiche 最初主张：MoRI normal 使用动态 token 数与通信缓冲，graph 重放复用 capture 时布局导致路由破坏，GSM8K 由 1.000（*eager*）跌到 0.000（graph），因此要禁图形。Duyi-Wang 反驳：valid token count 与 routing results 虽动态，但 MoRI 通信缓冲是预分配固定容量、地址稳定，CUDA Graph 重放预期复用捕获布局，动态 token 数不构成理由；DeepSeek-V3/V4 在 MoRI normal + *decode* graph 下正常，更像 Qwen3.5/SGLang 的 graph 集成问题。yichiche 复验确认后把根因收敛到 qwen2_moe.py 门控缺失，并从 PR 中移除 graph guard，理由包括 *eager* 0.935 与 graph 0.932 在并行 200 下高度一致、禁用 graph 使 MTP *accept* length 明显下降。另一讨论是后端谓词范围：初始用 is_deepep_class_backend()，review 后收敛为显式 is_deepep() or is_mori()，避免 Mooncake/PPLX 无验证覆盖被静默改动。CI 层面 amd-bot 多轮提示：早期 CI 从未运行、测试文件收集失败、MoRI 运行时路径无 PR-CI 测试覆盖、test_gsm8k 出现过 *accept* 2.51 的中间回归，最终确认无 PR 引起的执行失败。

- Decode CUDA graph 禁用是否必要（server_args.py） (correctness): yichiche 复验确认 Duyi-Wang 正确，根因是 qwen2_moe.py 门控缺失，graph 禁用被从 PR 移除；并用 *eager* 0.935 与 graph 0.932 一致、吞吐 2.2 倍、MTP *accept* length 3.47-3.59 等数据支撑。
- 后端谓词范围：is_deepep_class_backend() 与 is_deepep()/is_mori() (design): 采用显式谓词，保持 CUDA 后端路径与 main 字节级一致。
- 新增测试文件导致 AMD *stage-a* 收集失败 (testing): *commit* de1cad0 修复为 CustomTestCase 风格；后续 AITER 变更整体移除，该文件不再存在。
- test_gsm8k *spec-decode* *accept* length 回归 (testing): 最终 CI 摘要确认无 PR 引起的执行失败；运行时路径仍未被 PR-CI 直接测试。

风险与影响

- 风险：
  - 回归风险：qwen2_moe.py 为 Qwen2/Qwen3/Qwen3.5 家族共用模型文件，改动虽只在 is_mori() 为真时生效，但 MoRI 后端在 AMD 上还服务其他模型（如 DeepSeek），本次验证仅覆盖 Qwen3.5-397B，其他模型组合需回归确认；
  - 性能风险：can_fuse_shared_expert 对 MoRI 返回 False 后，共享专家走独立 MLP（不进融合 MoE kernel），可能带来额外 kernel 与显存开销；PR 给出端到端 1552 tok/s，但未与融合路径做同配置对比；
  - 测试覆盖风险：新增单测只 pin 静态门控；MoRI 运行时路径的 8-GPU test_moriep_small.py::TestMTP::test_gsm8k 在中间版本出现 accept 2.51 回归（阈值 2.8），最终版本虽说明与 graph 变更相关并已移除，但 amd-bot 提示该运行时路径未被 PR-CI 真正执行；
  - CI 过程风险：多轮 Rebase Required，合并基线需有最新 main 的完整 CI 信号；
  - 全局状态风险：测试通过 get_flags().moe.a2a_backend 修改全局配置，依赖 addCleanup 恢复，并行执行测试时需注意隔离。
- 影响：
  - 用户影响：AMD MI355X/ROCm 上以 MoRI 后端部署 Qwen3.5-397B-A17B-MXFP4 的用户，精度从不可用（0.006）恢复到 0.93+，不再需要牺牲 decode CUDA graph（吞吐 1552 vs 691 tok/s），spec-decode 接受长度保持约 3.5；
  - 系统契约影响：is_mori() 正式成为 Qwen MoE 布局契约的一部分，后续接入新的 DeepEP 类后端时必须同步评估四个门控；MoRI 的官方支持面扩大；
  - 团队流程影响：PR 展示了先收敛根因、再打补丁的流程价值，amd-bot 的多轮状态检查暴露 AMD 后端 CI 覆盖缺口（MoRI 运行时路径无 PR-CI 测试）。影响程度中等：代码面小、行为变化面受限，但修复的是核心路径的严重正确性 bug。
  - 风险标记：核心模型文件门控变更，AMD/MoRI 特定路径，MoRI 运行时 CI 覆盖不足，共享专家融合禁用或带来性能开销，测试操纵全局状态

关联脉络

- PR #31793 （PR body 引用的前序 init 修复，标题未在本材料中提供）：PR body 明确说明本 PR 是其 correctness follow-up：#31793 仅禁用了全局 slot 的共享专家融合以让 MoRI 通过启动崩溃，本 PR 补全四个门控解决路由错误。
- PR #35719 [AMD] Fix Qwen3.5 MTP dropping fused shared-expert weights: 同一条 Qwen3.5 MoE 融合共享专家 + AMD + spec-decode 修复线，且本 PR 验证数据覆盖 EAGLE MTP 路径；两者都围绕 shared expert 布局正确性。