PR #28074 完整报告

sgl-project/sglang

[AMD] Fix no-op dtype cast in _topk_ids_logical_to_physical_dynamic on HIP

合并时间：2026-06-21 02:07

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/28074

执行摘要

- 一句话：修复 HIP 动态调度 dtype 转换无操作 bug
- 推荐动作：该 PR 值得所有 AMD HIP 用户关注。设计亮点：作者通过对比静态路径的正确实现，发现动态路径中因变量覆盖顺序导致的常见 bug 模式（no-op cast）。新增的测试设计清晰，覆盖边界情况，可作为类似 bugfix 测试的参考。

功能与动机

修复动态专家调度路径下 dtype 保留失效导致的专家路由损坏。PR body 指出："Without this fix, int64 expert indices flow into MORI's C++ dispatch kernel, which reads via raw data_ptr() assuming TOPK_IDX_DTYPE = torch.int32. The int64 data is misinterpreted: every other int32 element reads as 0 ... completely corrupting expert routing."

实现拆解

1. 保存原始 dtype：在 _topk_ids_logical_to_physical_dynamic 函数中，在 topk_ids 被 partial_logical_to_all_physical_map 覆盖之前，增加 original_dtype = topk_ids.dtype 语句，保留输入 tensor 的数据类型。
2. 修正 cast 目标：将条件判断 if _is_hip: 内的 cast 语句从 topk_ids.to(topk_ids.dtype) 改为 topk_ids.to(original_dtype)，确保结果与输入 dtype 一致。这是唯一的逻辑变更。
3. 新增测试覆盖：创建 test_dispatch_dtype_preservation.py，包含 TestStaticDispatchDtype 和 TestDynamicDispatchDtype 两个测试类，共 9 个 CPU 单元测试，验证 dtype 保留、值正确性和形状保持。通过 register_cpu_ci 注册到 CI 套件 base-a-test-cpu。
4. CI 配置：测试文件通过 register_cpu_ci 自动注册，无需手动修改 CI 流程。

关键文件：

- python/sglang/srt/eplb/expert_location_dispatch.py（模块 专家调度；类别 source；类型 core-logic；符号 _topk_ids_logical_to_physical_dynamic）：核心修复文件，修改了动态调度函数的 dtype 保留逻辑，仅 2 行变更。
- test/registered/unit/eplb/test_dispatch_dtype_preservation.py（模块 单元测试；类别 test；类型 test-coverage；符号 _make_identity_info, _make_permuted_info, TestStaticDispatchDtype, TestDynamicDispatchDtype）：新增的测试文件，提供全面的 dtype 保留测试，确保修复正确且无回归。

关键符号：_topk_ids_logical_to_physical_dynamic, _topk_ids_logical_to_physical_static

关键源码片段

python/sglang/srt/eplb/expert_location_dispatch.py

核心修复文件，修改了动态调度函数的 dtype 保留逻辑，仅 2 行变更。

```
def _topk_ids_logical_to_physical_dynamic(
    topk_ids: torch.Tensor, info: Optional[ExpertLocationDispatchInfo]
) -> torch.Tensor:
    topk_ids_original_shape = topk_ids.shape
    # 保存原始 dtype，因为下一行会覆盖 topk_ids 为 int64 映射表结果
    original_dtype = topk_ids.dtype
    device = topk_ids.device
    topk_ids = topk_ids.flatten()

    chosen_dispatch_index = (
        torch.randint(0, 65536, topk_ids.shape, dtype=torch.int32, device=device)
        % info.partial_logical_to_all_physical_map_num_valid[topk_ids]
    )
    # 此时 topk_ids 变为 int64（映射表 dtype）
    topk_ids = info.partial_logical_to_all_physical_map[
        topk_ids, chosen_dispatch_index
    ]
    if _is_hip:
        # 转换为原始 dtype，而不是 topk_ids.dtype（已为 int64）
        topk_ids = topk_ids.to(original_dtype)

    topk_ids = topk_ids.view(topk_ids_original_shape)
    return topk_ids
```

test/registered/unit/eplb/test_dispatch_dtype_preservation.py

新增的测试文件，提供全面的 dtype 保留测试，确保修复正确且无回归。

```
class TestDynamicDispatchDtype(CustomTestCase):
    """Tests for _topk_ids_logical_to_physical_dynamic dtype preservation."""

    def test_preserves_int32_dtype_on_hip(self):
        """int32 输入在 HIP 下必须产生 int32 输出。"""
        info = _make_permuted_info() # int64 映射表
        info.ep_dispatch_algorithm = "dynamic"
        topk_ids = torch.tensor([5, 103, 206], dtype=torch.int32)
        with patch("sglang.srt.eplb.expert_location_dispatch._is_hip", True):
            result = _topk_ids_logical_to_physical_dynamic(topk_ids, info)
        self.assertEqual(result.dtype, torch.int32)
        # 验证值正确：映射表取出后 int32 保留
        expected = info.partial_logical_to_all_physical_map[
            topk_ids.long(), 0
        ].to(torch.int32)
```

```
        self.assertTrue(torch.equal(result, expected))
```

评论区精华

本 PR 无人工 review 评论，只有作者与 CI bot 的交互。HaiShaw 在 CI 失败后发起 /tag-and-rerun-ci，amd-bot 回复的 CI 总结确认："No executed failure is caused by this PR — all reds are pre-existing infra/network/flake on unrelated backends". 最终 HaiShaw 批准合并。

- CI 状态确认 (test): PR 无 CI 回归问题，可以合并。

风险与影响

- 风险：变更极其聚焦（2 行源码 + 测试），核心风险是 original_dtype 是否始终正确。若 topk_ids 输入为 int64 且映射表也为 int64，转换到 int64 是多余的但无害。测试覆盖了 int64 输入和 int32 输入情况。仅影响 HIP 路径（_is_hip=True），CUDA 或其他后端不走此分支。无性能影响。回归风险较低。
- 影响：修复影响使用 AMD GPU、MORI 专家并行（EP）和动态（或 fake）EPLB 算法的用户。此前动态 EPLB 导致专家路由完全错误（路由到错误专家），影响推理质量。修复后动态 EPLB 的正确性与静态路径一致，高并发解码均衡度从 ~0.49 恢复到 ~0.87。对静态路径用户无影响。
- 风险标记：AMD 专用路径，专家路由逻辑

关联脉络

- 暂无明显关联 PR