

PR #39292 完整报告

sgl-project/sglang

[Diffusion] Don't route an unreadable checkpoint into the native fallback

合并时间: 2026-09-13 20:30

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/39292>

执行摘要

- 一句话: 阻断不可读 checkpoint 进入 native fallback, 保留原始 I/O 错误
- 推荐动作: 值得精读。虽然改动只有 6 行源码, 但它展示了“异常路由要匹配问题语义”的设计思路: fallback 不是万能兜底, 错误信息保真比二次包装更有价值。配套单测对误拦截的回归防护到位, 适合作为错误诊断类修复的参考案例。

功能与动机

PR body 明确指出: native fallback 回答的问题是“该架构没有定制实现, 所以用库的加载器加载”; 而 I/O 错误回答的是另一个问题——fallback 也读不了同一份文件。把缺失分片路由进 fallback, 会把“文件读不了”转成“缺少实现”的报告, 对于明明有实现的模型产生误导。尤其在缓存快照不完整时 (下载中断、并发下载、blob 被驱逐), 用户看到的报错只有第一行 `FileNotFoundError` 是真实的, 其余指向 `tp_size` 和实现状态, 与实际原因无关。

实现拆解

1. 定位异常路由入口: `python/sglang/multimodal_gen/runtime/loader/component_loaders/component_loader.py` 的 `ComponentLoader.load` 方法内, `_load_customized_with_context` 抛出异常后经过 `except` 链。原本只有 `ComponentAttentionBackendNotAppliedError`、`ComponentCheckpointUnsupportedError`、`ComponentResidencyError` 被原样透传, 其余异常 (含 I/O 错误) 会进入 native fallback 分支。
2. 扩展直传异常集合: 在透传元组中追加 `OSError`。这样 `FileNotFoundError`、`PermissionError` 等所有 I/O 类失败都会直接 `raise`, 跳过 `allow_native_fallback`、`NativeComponentLoaderRequired` 判断等下游逻辑。选择宽泛的 `OSError` 而非仅 `FileNotFoundError`, 是因为权限、磁盘满等错误同样不属于“缺少实现”的语义。
3. 配套测试: 在 `python/sglang/multimodal_gen/test/unit/test_transformer_loader_fallback.py` 新增 `test_unreadable_checkpoint_is_not_a_missing_implementation`, 通过 `mock.patch.object` 让定制加载器抛出 `FileNotFoundError`, 断言 `_load_native_with_context` 未被调用、异常消息保留分片文件名。测试在修复前的代码上会失败, 具备回归防护作用。
4. 配置与部署: 无新增参数、无 schema 变更、无性能影响。

关键文件:

- python/sglang/multimodal_gen/runtime/loader/component_loaders/component_loader.py (模块 组件加载; 类别 source; 类型 core-logic; 符号 ComponentLoader.load) : 核心变更文件: 在 ComponentLoader.load 的透传异常元组中加入 OSError, 使不可读 checkpoint 的原始 I/O 错误不再被 native fallback 吞掉并误报为缺少实现。
- python/sglang/multimodal_gen/test/unit/test_transformer_loader_fallback.py (模块 加载测试; 类别 test; 类型 test-coverage; 符号 test_unreadable_checkpoint_is_not_a_missing_implementation) : 配套测试文件: 新增 test_unreadable_checkpoint_is_not_a_missing_implementation, 验证 FileNotFoundError 原样透传、native fallback 不被调用、错误消息保留分片文件名, 防止误路由回归。

关键符号: ComponentLoader.load, test_unreadable_checkpoint_is_not_a_missing_implementation

关键源码片段

python/sglang/multimodal_gen/runtime/loader/component_loaders/component_loader.py

核心变更文件: 在 ComponentLoader.load 的透传异常元组中加入 OSError, 使不可读 checkpoint 的原始 I/O 错误不再被 native fallback 吞掉并误报为缺少实现。

```
try:
    component = self._load_customized_with_context(
        component_model_path,
        server_args,
        component_name,
        component_attn_backend,
        component_attn_name,
        require_backend_selection,
    )
    source = "sgl-diffusion"
except (
    ComponentAttentionBackendNotAppliedError,
    ComponentCheckpointUnsupportedError,
    ComponentResidencyError,
    # native fallback 回答的是“该架构没有定制实现”; 而 checkpoint 读不了是另一种
    # 失败, fallback 也无法读取同一份文件。因此直接透传 OSError (含 FileNotFoundError) ,
    # 让原始错误保留文件名, 而不是把它误报成“缺少实现”
    OSError,
):
    raise
except Exception as e:
    # 只有真正“没有定制实现 / 需要原生加载器”之类的异常, 才继续走 native fallback
    if require_backend_selection or not allow_native_fallback:
        raise
    # 后续 fallback 决策逻辑保持不变
```

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

配套测试文件：新增 test_unreadable_checkpoint_is_not_a_missing_implementation，验证 FileNotFoundError 原样透传、native fallback 不被调用、错误消息保留分片文件名，防止误路由回归。

```
def test_unreadable_checkpoint_is_not_a_missing_implementation(self):
    # native fallback 只回答“没有定制实现”；读不了的 checkpoint 是另一种失败，
    # 把它路由进 fallback 会把“缺少分片”误报成“缺少实现”
    loader = TransformerLoader()
    missing_shard = FileNotFoundError(
        2, "No such file or directory", "/cache/transformer/shard-00002.safetensors"
    )
    customized_load = mock.patch.object(
        loader, "_load_customized_with_context", side_effect=missing_shard
    )
    # native 加载器本来也会失败（同一文件读不了），这里返回 object() 以验证
    # 它在修复后的代码中根本不会被调用
    native_load = mock.patch.object(
        loader, "_load_native_with_context", return_value=object()
    )
    available_memory = mock.patch(
        "sglang.multimodal_gen.runtime.loader.component_loaders."
        "component_loader.current_platform.get_available_gpu_memory",
        return_value=0.0,
    )

    with customized_load, native_load as native, available_memory:
        with self.assertRaises(FileNotFoundError) as caught:
            loader.load(
                "/model/transformer_2",
                self._server_args(),
                "transformer_2",
                "diffusers",
            )

    # 原始错误信息里保留文件名，且 native fallback 未被调用
    self.assertIn("shard-00002.safetensors", str(caught.exception))
    native.assert_not_called()
```

评论区精华

本 PR 没有 review 评论；设计判断集中在 PR body 与 commit message 中。作者明确了两点取舍：一是 fallback 只能回答“没有定制实现”，无法回答“文件读不了”，因此不可读的 checkpoint 不应进入 fallback；二是选用 `OSError` 这类宽泛异常而非仅 `FileNotFoundError`，覆盖权限、磁盘满等同类 I/O 失败，并保持 re-raise 原样透传以保留文件名。

- 暂无高价值评论线程

风险与影响

- 风险：变更核心是异常路由范围扩大：OSError 覆盖面较宽，除文件缺失外，权限错误、磁盘满、设备错误等也会直接透传。若存在某种定制加载失败但 fallback 能读取不同路径的场景，则会失去一次“救回”机会；但 checkpoint 通常指向同一批文件，fallback 大概率同样失败，实际风险很低。测试仅覆盖 FileNotFoundError 一个子类，PermissionError 等行为一致但缺少显式用例，属轻微覆盖缺口。正常加载路径完全不受影响，无性能与安全影响。
- 影响：对用户和运维的影响主要是排障体验：缓存快照不完整、并发下载、blob 驱逐等场景下，错误首屏直接显示真实文件名，不再被“缺少实现”或 tp_size 误导。对运行系统无影响，正常加载路径零变化。对团队而言，该改动为 loader 的 fallback 语义建立了明确的异常分类边界：只有“缺少定制实现”类问题才进入 fallback，I/O 类问题一律原样上抛，后续新增组件加载器可复用该模式。
- 风险标记：异常路由行为变更，宽泛 OSError 覆盖，测试仅覆盖 FileNotFoundError 子类

关联脉络

- PR #38549 [Diffusion] Return Qwen-Image-Layered outputs and preserve CFG2 rounding: 同属 multimodal_gen runtime 可靠性改进线，均涉及组件加载 / 执行链路的防御性修复。
- PR #39291 [Diffusion] Make the SP sequence gather pass contiguous shards: 同为 diffusion runtime 的稳定性修复，且都用单元测试覆盖分片 (shard) 相关处理。
- PR #39255 [CI] Make diffusion GT generation usable without the publish token, and cover the 5090 lane: 本 PR 的 CI 状态依赖 diffusion CI 工作流，两者同属 diffusion 质量保障基础设施。