PR #25911 完整报告

sgl-project/sglang

Purge usage of pytorch named tensors

合并时间：2026-05-27 05:58

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25911

执行摘要

- 一句话：移除 PyTorch named tensors 依赖，避免向上兼容风险
- 推荐动作：建议阅读核心文件 tensor_naming.py 的变更，理解如何用私有属性模拟命名语义。该模式可推广到其他需要绕过废弃 API 的场景。同时也值得查看 review 讨论中关于原地修改与非连续张量的设计权衡。

功能与动机

PyTorch 将在未来版本中移除 named tensors 特性（参见 PyTorch #173895）。为了避免 SGLang 仓库因此中断，PR 清除了所有该特性的使用。作者强调使用仅局限于调试工具和关联测试，风险极低。

实现拆解

1. 在 tensor_naming.py 中定义了三个辅助函数：get_dim_names（读取）、apply_dim_names（赋予）和 without_dim_names（移除），将维度名称存储为张量的 _dim_names 私有属性。
2. 在涉及维度名称操作的多个 executor 文件（如 unsharder/executor.py、token_aligner/smart/executor.py、reorderer/executor.py、bundle_comparator.py 等）中，将 tensor.names 替换为 get_dim_names，将 tensor.refine_names() 替换为 apply_dim_names，将 tensor.rename(None) 替换为 without_dim_names。
3. 同步更新了关联的测试文件，确保断言中张量比较时调用 without_dim_names 来消除维度名称影响。
4. 根据 review 意见，将 apply_dim_names 从直接修改输入张量改为通过 torch.ops.aten.alias 创建别名后修改，保持非原地语义；同时使用 alias 确保非连续张量的兼容性。

关键文件：

- python/sglang/srt/debug_utils/comparator/dims_spec/tensor_naming.py（模块 张量命名；类别 source；类型 core-logic；符号 get_dim_names, apply_dim_names, without_dim_names）：定义核心辅助函数 get_dim_names、apply_dim_names、without_dim_names，是整个 PR 的变更基础。
- python/sglang/srt/debug_utils/comparator/aligner/unsharder/executor.py（模块 去分片；类别 source；类型 core-logic）：在核心 unshard 逻辑中全面替换 named tensor 操作

为新辅助函数，体现 PR 的主要调用模式。

- python/sglang/srt/debug_utils/comparator/aligner/token_aligner/smart/executor.py（模块 令牌对齐；类别 source；类型 core-logic）：在 token aligner 中替换 strip_dim_names 的使用，展示 BS→T 折叠等场景的适配。
- python/sglang/srt/debug_utils/comparator/bundle_comparator.py（模块 束比较；类别 source；类型 core-logic）：在顶层束比较入口处理对齐结果的维度名称清理，使用 without_dim_names 替换 rename(None)。
- test/registered/debug_utils/comparator/aligner/unsharder/test_executor.py（模块 测试；类别 test；类型 test-coverage）：测试文件同步更新，使用 apply_dim_names 和 without_dim_names 验证 unshard 结果。

关键符号：get_dim_names, apply_dim_names, without_dim_names, resolve_dim_by_name

关键源码片段

python/sglang/srt/debug_utils/comparator/dims_spec/tensor_naming.py

定义核心辅助函数 get_dim_names、apply_dim_names、without_dim_names，是整个 PR 的变更基础。

```
from __future__ import annotations
from typing import Optional
import torch

_DIM_NAMES_ATTR = "_dim_names" # 私有属性名称，用于存储维度名称

def get_dim_names(tensor: torch.Tensor) -> tuple[Optional[str], ...]:
    """获取张量的维度名称。如果未设置名称，则返回全 None 的元组。"""
    names = getattr(tensor, _DIM_NAMES_ATTR, None)
    if names is not None:
        return names
    return (None,) * tensor.ndim

def apply_dim_names(tensor: torch.Tensor, dim_names: list[str]) -> torch.Tensor:
    """为张量设置维度名称。返回一个带有名称的新视图，不修改原张量。"""
    if tensor.ndim != len(dim_names):
        raise ValueError(
            f"dims metadata mismatch: tensor has {tensor.ndim} dims (shape {list(tensor.shape)}) "
            f"but dims string specifies {len(dim_names)} names {dim_names}. "
            f"Please fix the dims string in the dumper.dump() call to match the actual tensor shape.
            "
        )
    view = torch.ops.aten.alias(tensor) # 创建别名，保留 contiguity 和 strides
    view._dim_names = tuple(dim_names) # 在别名上设置私有属性
    return view

def without_dim_names(tensor: torch.Tensor) -> torch.Tensor:
    """返回一个没有维度名称的新视图。"""
```

```
    return torch.ops.aten.alias(tensor) # 别名不会继承 _dim_names
```

python/sglang/srt/debug_utils/comparator/aligner/unsharder/executor.py

在核心 unshard 逻辑中全面替换 named tensor 操作为新辅助函数，体现 PR 的主要调用模式。

```
def _apply_unshard(
    params: UnsharderParams,
    ordered_tensors: list[torch.Tensor],
    *,
    axis: ParallelAxis,
    group_index: int,
) -> tuple[torch.Tensor, list[ReplicatedCheckResult]]:
    if isinstance(params, PickParams):
        checks = _verify_replicated_group(ordered_tensors, axis=axis, group_index=group_index)
        return ordered_tensors[0], checks

    if isinstance(params, ConcatParams):
        dim: int = resolve_dim_by_name(ordered_tensors[0], params.dim_name)
        names = get_dim_names(ordered_tensors[0]) # 获取原始维度名称
        result = torch.cat(ordered_tensors, dim=dim)
        if names[0] is not None:
            result = apply_dim_names(result, list(names)) # 保留名称
        return result, []

    if isinstance(params, ReduceSumParams):
        names = get_dim_names(ordered_tensors[0])
        stripped = [without_dim_names(t) for t in ordered_tensors] # 移除名称后再 stack/sum
        result = torch.stack(stripped).sum(dim=0)
        if names[0] is not None:
            result = apply_dim_names(result, list(names))
        return result, []
    # ...
```

评论区精华

- Qiaolin-Yu 指出新实现 apply_dim_names 直接修改 tensor，而原 refine_names 返回新视图。作者回应改为创建别名以保持原语义。
- Qiaolin-Yu 询问对非连续 tensor 是否有问题。作者回应使用 torch.ops.aten.alias 保留 contiguity 和 strides，并改进了代码。
- apply_dim_names 原地修改 vs 新视图 (design): 作者通过创建别名 (torch.ops.aten.alias) 并在别名上设置属性，保持非原地语义。
- 非连续张量兼容性 (correctness): 作者使用 torch.ops.aten.alias 保留原张量的 contiguity 和 strides，确认无问题。

风险与影响

- 风险：主要风险在于调试工具（debug_utils/comparator）的维度名称传递可能出现遗漏或错误，导致张量对齐或比较失败。但由于所有变更均基于简单的 API 替换，且测试覆盖充分，风险可控。此外，使用私有属性 _dim_names 需要确保在张量别名传播时不丢失，torch.ops.aten.alias 能正确共享数据但不会自动传播属性，因此 without_dim_names 通过不设置 _dim_names 来模拟无名称状态，语义正确。
- 影响：直接影响是清理了废弃 API，保证未来 PyTorch 升级时不会因 named tensors 移除而出错。受影响范围限定在 debug_utils/comparator 模块及其测试，不影响核心推理或训练路径。对于使用调试工具的开发人员，行为保持透明。
- 风险标记：依赖私有属性，调试工具范围，PyTorch 升级兼容

关联脉络

- 暂无明显关联 PR