PR #30786 完整报告

sgl-project/sglang

[Kernel] Migrate scattered MoE kernels to sglang.kernels (RFC #29630, Phase 2.5, 2/7)

合并时间：2026-07-14 09:03

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/30786

执行摘要

- 一句话：将 ~42 个 MoE Triton 内核从 srt/layers/moe 迁移至 sglang.kernels
- 推荐动作：值得精读，特别是对内核重构和代码迁移感兴趣的同学。本 PR 展示了大规模代码移动的实践：使用 git mv 保留历史、批量更新导入、处理合并冲突。讨论中的自动审查问题提醒我们需要关注原始代码中的缺陷。

功能与动机

根据 RFC #29630 中定义的内核迁移计划，将分散在 SRT 各处的 Triton 内核统一到 sglang.kernels 命名空间下，便于后续集中维护、测试和性能优化。本 PR 完成 Phase 2.5 中 7 个 PR 的第 2 个，专注于 MoE 相关内核的搬迁。

实现拆解

1. 大规模移动（Wholesale moves）：将 5 个整文件从 sglang/srt/layers/moe/ 移动到 sglang/kernels/ops/moe/，包括 ep_moe/kernels.py → ep_moe_kernels.py（22 个内核）、fused_moe_triton_kernels.py（10个内核）、mxfp8_moe_amd_gfx95.py（2个内核）、rocm_moe_utils.py（2 个内核 + aiter 包装）、router.py（2 个内核 + 启动器）。使用 git mv 保留历史，git 检测为 100% rename。
2. 提取内联内核：从混合模块中将内核与策略分离。 - topk.py：移除 _fill_padded_rows_kernel、_fill_padded_rows 等 Triton 代码，改为从新的 sglang/kernels/ops/moe/fill_padded_rows.py 导入。新文件保留了运行时的获胜副本（使用raise而非assert）。-waterfill.py：移除WaterfillDispatchPlan、_empty_expanded、_count_routed_per_rank_kernel、_waterfill_expand_kernel、materialize_waterfill_dispatch_fused 等内核代码，移到新文件 sglang/kernels/ops/moe/deepep_waterfill_kernels.py，保留策略函数 expand_topk_with_shared_expert。
3. 导入重写（全仓库）：更新所有引用这些内核的模块，包括 deep_gemm.py、fused_moe.py、humming.py、triton.py、lora_moe_runner_marlin.py 以及模型文件 grok.py、longcat_flash.py、qwen2_moe.py 等，共计修改 31 个文件。
4. 注册 KernelSpec 清单：在 sglang/kernels 中将迁移后的公共入口点注册为 TRITON 后端的 KernelSpec 库存，确保导入纯元数据特性。

5. 测试验证：import sglang.kernels 保持元数据纯净；test_kernels_namespace.py + test_fused_op.py 共 33 个测试通过；MoE 正确性 /e2e 测试通过重写的导入继续运行。

关键文件：

- python/sglang/srt/layers/moe/waterfill.py（模块 MoE 策略层；类别 source；类型 core-logic；符号 WaterfillDispatchPlan, _empty_expanded, _count_routed_per_rank_kernel, _waterfill_expand_kernel）：核心变更：从包含大量 Triton 内核转变为仅保留策略函数，所有内核定义被提取到 sglang/kernels/ops/moe/deepep_waterfill_kernels.py。
- python/sglang/srt/layers/moe/topk.py（模块 MoE 路由；类别 source；类型 dependency-wiring；符号 is_power_of_two, _fill_padded_rows_kernel, _can_fuse_padded_region, _fill_padded_rows）：移除了 Triton 填充行内核定义，改为从新的 fill_padded_rows.py 导入；删除了 import triton 和 import triton.language。
- python/sglang/kernels/ops/moe/fill_padded_rows.py（模块 内核库；类别 infra；类型 infrastructure；符号 _fill_padded_rows_kernel, _can_fuse_padded_region, _fill_padded_rows）：新文件：从 topk.py 提取的填充行 Triton 内核，使用显式 raise 替代 assert，确保 python -O 下依然生效。
- python/sglang/kernels/ops/moe/deepep_waterfill_kernels.py（模块 内核库；类别 infra；类型 infrastructure；符号 WaterfillDispatchPlan, _empty_expanded, _count_routed_per_rank_kernel, _waterfill_expand_kernel）：新文件：包含从 waterfill.py 提取的全部 Waterfill Triton 内核（计数、扩展、调度规划），是本次迁移的核心新文件之一。
- python/sglang/srt/layers/moe/moe_runner/deep_gemm.py（模块 MoE 运行器；类别 source；类型 dependency-wiring）：展示导入重写：将旧路径 sglang.srt.layers.moe.ep_moe.kernels 替换为 sglang.kernels.ops.moe.ep_moe_kernels。
- python/sglang/srt/layers/moe/moe_runner/triton_utils/fused_moe.py（模块 MoE 运行器；类别 source；类型 dependency-wiring）：展示导入重写：将相对导入 . fused_moe_triton_kernels 替换为 from sglang.kernels.ops.moe.fused_moe_triton_kernels import ...

关键符号：expand_topk_with_shared_expert, _fill_padded_rows, _fill_padded_rows_kernel, _count_routed_per_rank_kernel, _waterfill_expand_kernel, materialize_waterfill_dispatch_fused, tma_align_input_scale, silu_and_mul_masked_fwd, moe_ep_deepgemm_preprocess, post_reorder_deepgemm, ep_scatter, ep_gather, silu_and_mul_masked_post_quant_fwd, act_and_mul_triton, invoke_fused_moe_kernel, moe_sum_reduce_triton, support_tensor_descriptor

关键源码片段

python/sglang/srt/layers/moe/waterfill.py

核心变更：从包含大量 Triton 内核转变为仅保留策略函数，所有内核定义被提取到 sglang/kernels/ops/moe/deepep_waterfill_kernels.py。

```
# waterfill.py（迁移后）— 内核定义已移到 sglang.kernels.ops.moe.deepep_waterfill_kernels
```

```
from typing import Optional, Tuple

import torch
from torch import Tensor

from sglang.kernels.ops.moe.deepep_waterfill_kernels import (
    LOCAL_SHARED_MARKER,
    WaterfillDispatchPlan,
    _count_routed_per_rank_kernel,
    _empty_expanded,
    materialize_waterfill_dispatch_fused,
)
from sglang.srt.environ import envs
from sglang.srt.layers.moe.topk import StandardTopKOutput

@torch.compile(dynamic=True)
def expand_topk_with_shared_expert(
    topk_ids: Tensor,
    topk_weights: Tensor,
    num_routed_experts: int,
    world_size: int,
    source_rank: int,
    shared_weight: float,
) -> Tuple[Tensor, Tensor]:
    """Expand topk [N, 8] → [N, 9] 并重映射 ID；共享专家始终本地处理。"""
    num_tokens = topk_ids.shape[0]
    topk = topk_ids.shape[1]
    device = topk_ids.device
    old_epr = num_routed_experts // world_size
    new_epr = old_epr + 1
    has_valid = (topk_ids >= 0).any(dim=1)
    valid_mask = topk_ids >= 0
    old_ranks = torch.where(valid_mask, topk_ids // old_epr, torch.zeros_like(topk_ids))
    expanded_topk_ids = torch.empty(num_tokens, topk + 1, dtype=topk_ids.dtype, device=device)
    expanded_topk_ids[:, :topk] = torch.where(valid_mask, topk_ids + old_ranks, topk_ids)
    shared_id = source_rank * new_epr + old_epr
    expanded_topk_ids[:, topk] = torch.where(has_valid, shared_id, LOCAL_SHARED_MARKER)
    # ... ( 后续使用已迁移的内核进行 dispatch)
    return expanded_topk_ids, torch.empty(num_tokens, topk + 1, dtype=topk_weights.dtype,
    device=device)
```

python/sglang/srt/layers/moe/topk.py

移除了 Triton 填充行内核定义，改为从新的 fill_padded_rows.py 导入；删除了 import triton 和 import triton.language。

```
# topk.py ( 迁移后 ) — 移除了 Triton 内核定义，改为从 sglang.kernels 导入

# 删除的导入 :
# import triton
```

```
# import triton.language as tl

# 新增导入：
from sglang.kernels.ops.moe.fill_padded_rows import (
    _can_fuse_padded_region,
    _fill_padded_rows,
)

# 其余代码保持不变，包括路由算法、topk 选择等
# 策略函数 _mask_topk_ids_padded_region 现在调用 _fill_padded_rows 而非内联内核
```

python/sglang/kernels/ops/moe/fill_padded_rows.py

新文件：从 topk.py 提取的填充行 Triton 内核，使用显式 raise 替代 assert，确保 python -O 下依然生效。

```
"""Fused padded-row fill for MoE top-k outputs.

Migrated from sglang.srt.layers.moe.topk (RFC #29630, Phase 2.5), where two
near-identical copies had accumulated; this keeps the later, runtime-winning
copy (explicit raises instead of asserts).
"""

import torch
import triton
import triton.language as tl


@triton.jit
def _fill_padded_rows_kernel(
    out_ptr,
    num_token_non_padded_ptr,
    n_cols,
    fill_value,
    stride_row,
    BLOCK_COLS: tl.constexpr,
):
    """单行填充内核，每个程序处理一行。"""
    row = tl.program_id(0)
    n_valid = tl.load(num_token_non_padded_ptr)
    if row >= n_valid:
        cols = tl.arange(0, BLOCK_COLS)
        mask = cols < n_cols
        ptrs = out_ptr + row * stride_row + cols
        fill = tl.full((BLOCK_COLS,), fill_value, dtype=out_ptr.dtype.element_ty)
        tl.store(ptrs, fill, mask=mask)


def _can_fuse_padded_region(x: torch.Tensor) -> bool:
    """只有行主序 2D 张量且列连续时才能使用融合内核。"""
```

```
    return x.dim() == 2 and x.stride(1) == 1


def _fill_padded_rows(
    x: torch.Tensor,
    num_token_non_padded: torch.Tensor,
    fill_value,
) -> None:
    """将大于等于 num_token_non_padded 的行填充为 fill_value。

    使用单个 Triton 内核启动替代 eager 的 arange + index_put_ 序列，
    减少启动延迟，且可被 CUDA/HIP 图捕获。
    """
    # 使用显式 raise 替换 assert，确保在 python -O 优化模式下仍然生效
    if not isinstance(num_token_non_padded, torch.Tensor):
        raise TypeError("num_token_non_padded must be a torch.Tensor")
    if num_token_non_padded.numel() != 1:
        raise ValueError(
            "num_token_non_padded must be a single-element tensor, got shape "
            f"{tuple(num_token_non_padded.shape)}"
        )
    if num_token_non_padded.dtype.is_floating_point:
        raise TypeError(
            "num_token_non_padded must be an integer tensor, got "
            f"{num_token_non_padded.dtype}"
        )
    if num_token_non_padded.device != x.device:
        raise ValueError("num_token_non_padded and x must be on the same device")
    n_rows, n_cols = x.shape
    _fill_padded_rows_kernel[(n_rows,)](
        x,
        num_token_non_padded,
        n_cols,
        fill_value,
        x.stride(0),
        BLOCK_COLS=triton.next_power_of_2(n_cols),
    )
```

评论区精华

自动代码审查工具 gemini-code-assist[bot] 在 review 中指出了几个潜在问题：

- 模零除风险（deepep_waterfill_kernels.py#L200）：Triton 的 tl.where 会评估两个分支，total_w 为 0 时导致模零除，建议用 tl.maximum(total_w, 1) 作为除数。
- 位掩码溢出与除零（deepep_waterfill_kernels.py#L287）：32 位位掩码在 world_size > 32 时溢出，且 old_experts_per_rank 可能为 0。
- 缺少布局检查（fill_padded_rows.py#L57）：未验证张量是否为支持的行主序布局。

- 空张量处理（fill_padded_rows.py#L71）：n_rows 为 0 时启动 Triton 内核可能浪费资源。 所有问题均未在本次 PR 中修复（它们也存在于原始代码中），但值得后续关注。
- 模零除风险 (deepep_waterfill_kernels.py) (correctness): 未在本 PR 中修复；该问题在原始代码中已存在。
- 位掩码溢出与除零 (deepep_waterfill_kernels.py) (correctness): 未修复，同样属于原始代码缺陷。
- 缺少布局验证 (fill_padded_rows.py) (correctness): 未修复，原始代码也未检查。
- 空张量处理 (fill_padded_rows.py) (performance): 未修复。

风险与影响

- 风险：本 PR 为纯代码移动和导入重写，内核逻辑无任何变更，运行时行为保持不变。主要风险来自导入重写可能遗漏某些调用点，但通过 11 个提交中多次合并 main 并修复导入冲突，已确保覆盖。自动审查指出几个在原始代码中已存在的边缘问题（模零除、位掩码溢出），迁移后同样存在，未引入新风险。
- 影响：影响范围：影响所有使用 MoE 内核的模块（DeepEP、LoRA、模型文件等），但均为导入路径变更，接口保持不变。 影响程度：无功能影响，无性能变化（字节一致迁移）。开发团队现在从统一的内核目录 sglang/kernels/ops/moe 导入，有利于后续集中优化和测试。
- 风险标记：自动审查提示模零除风险未修复，自动审查提示位掩码溢出风险未修复，自动审查提示缺少布局验证，自动审查提示空张量处理可优化

关联脉络

- PR #30784 [Kernel] Migrate fp8 kernel to sglang.kernels (Phase 2.5, 1/7): 同一迁移系列的前置 PR，本 PR 的改动建立在 #30784 打开的命名空间之上，且需在 #30784 之后合并。
- PR #31089 [Kernel] Hotfix: update sgl-kernel imports of relocated fp8_kernel (RFC #29630 #30784): 与本 PR 类似，修复了内核迁移过程中产生的导入路径问题，反映了该系列常见的迁移后维护工作。