PR #35114 完整报告

sgl-project/sglang

[kernels] Reorganize ops/diffusion by operator domain behind a lazy facade

合并时间：2026-08-18 20:37

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35114

执行摘要

- 一句话：ops/diffusion 按算子域重组并加懒门面统一导出
- 推荐动作：值得精读。该 PR 是 kernels 组织方式的架构级调整，核心看点有三：一是 PEP-562 懒门面的实现与 " 导出表 + 注册表 + 静态扫描守护 " 的组合，为多后端并存的算子库提供了可复制的组织范式；二是将 " 返回 None 表示不支持 " 改为 " 谓词 + raise" 的入口协议，直接消除了扩散模型 denoiser 中静默出错的最坏失败模式；三是测试重组展示了大文件合并时如何用参数化消除真实重复、并识别非本域的测试归属。建议后续新增算子或 backend 时对照 README 选择矩阵与 test_import_surface 的约束进行扩展。

功能与动机

PR body 明确指出 ops/diffusion 是 kernels/ops 中唯一按后端组织而非按算子组织的操作组，导致 "norm + scale/shift" 的六种实现散落在三个目录，回答 "does this fusion exist on ROCm?" 需要 grep 整个分组；同时该分组混入了 kernel、请求级 mount 策略、JIT C++/CUDA 扩展三类不同性质的东西，141 处外部 import 直接穿透叶模块，把布局钉死。懒门面是必须的而非风格选择："the backends have disjoint heavy dependencies ... an eager re-export would make each an import-time requirement on every platform." 入口协议从返回 None 改为 raise + 谓词，是因为 "a forgotten is None check produced a wrong image rather than an exception, which is the worst available failure mode for a denoiser."

实现拆解

按以下 5 步拆解实现：

1. 目录结构重组：将 ops/diffusion 下按后端划分的 triton/、cutedsl/、flydsl/ 目录拆成按算子域划分的子包——norm/、modulate/、rope/、activation/、attention/、layout/——后端改为文件名后缀（_triton、_jit、_cutedsl、_flydsl、_bitexact）。非算子的两类内容单独建目录：sites/ 承载请求级 mount 策略（重写 nn.Module 树），ext/ 承载 Hunyuan3D 光栅化 / 网格处理等无后端维度、无数值契约的扩展。kernel 函数体原样搬迁，包括 layernorm_modulate 与 rmsnorm_scale_shift_bitexact 的 SASS 级 bit-exact 复刻，完全不改动。
2. 懒门面与导入面收敛：__init__.py 变为 PEP-562 懒门面，通过 __getattr__ 按需解析 117 个导出符号与 38 个 KernelSpec 注册（此前只有 4 个）。这样 Triton、CUTLASS/CuTe-DSL、FlyDSL、MLX 等互斥重依赖不会在 import sglang 时全部加载。所

有 141 个外部 import 点改为从门面导入，使内部布局可再次调整而无需改动调用点。

3. 入口协议与平台分发统一：can_fuse_* 谓词全部折叠为 can_use_*；group_norm_silu_4d、group_norm_silu_rows、wan_rmsnorm_silu 三个 kernel 从 " 不支持时返回 None" 改为 " 调用方先查谓词、直接调用则 raise"，避免错误的图像而非异常。scale_shift、norm、rotary、rmsnorm_onepass 四处手写的 import-time 平台判断链收敛到 common.platform.select_impl 一个接缝，并消除了三处对 multimodal_gen.current_platform 的上行 import。新增 KernelBackend.FLYDSL 及其 BACKEND_METHODS 条目，使 ROCm norm kernel 如实注册自己的出身。
4. 测试重组与守护：37 个测试文件合并为 8 个——5 个非 diffusion kernel 测试迁回真实归属（两个量化测试到 ops/quantization/，perf-logger 同步测试到 profiling/），其余按算子域合并为一个套件，外加 test_sites.py（gate 协议）与 test_model_fast_paths.py（模型级接线）。合并时对真实重复做了参数化消除（LN+modulate 有五份近似的模型级拷贝、CuTe-DSL 套件两个类只差 gate 参数），测试主体保留。新增的 test_import_surface.py 用 AST 静态扫描守护导出表、注册表解析、门面惰性及 " 禁止外部代码 import 子模块 " 这两条不变量。
5. 配套文档与 CI 修正：README.md 增加选择矩阵，说明九个看起来可互换实则不同的 norm 实现的数值契约与支持形状 / 布局。提交历史还包含三处配套修正：合并套件后的 CI lane 继承问题拆分（test_layout.py 的 B200 lane 需要 FlashAttention）、B200/HIP 专属测试 lane 分离、平台谓词改为委托平台自身缓存的 is_cuda()/is_hip()，以及 Sana 测试的跨流竞态修复。

关键文件：

- python/sglang/kernels/ops/diffusion/__init__.py（模块 门面层；类别 source；类型 core-logic；符号 _EXPORTS, _SPECS, getattr）：PR 的核心：从 4 个导出升级为 PEP-562 懒门面，承载 117 个导出符号与 38 个 KernelSpec 注册，是全部 141 个外部 import 的唯一入口，决定了跨平台依赖是否泄漏。
- python/sglang/kernels/ops/diffusion/common/platform.py（模块 平台分发；类别 source；类型 core-logic；符号 select_impl）：新的平台分发接缝 select_impl 收敛了原先四处手写的 import-time if current_platform.is_X() 链，并保留了 multimodal_gen 之外最后的上行引用，是平台相关 kernel 路由的单一决策点。
- test/registered/kernels/ops/diffusion/test_import_surface.py（模块 导入守护；类别 test；类型 test-coverage；符号 _module_defines, test_every_export_resolves_to_a_real_symbol, test_every_symbol_imported_from_the_facade_is_exported, test_every_registered_spec_target_resolves）：新增的布局守护测试，用 AST 静态扫描双向校验导出表与注册表、验证门面惰性、禁止包外代码深导入子模块，是防止本次重组后布局再次退化的关键机制。
- test/registered/kernels/ops/diffusion/test_model_fast_paths.py（模块 模型路径；类别 test；类型 test-coverage；符号 _seed_cuda, _flux_eager, _flux_site_inputs, test_flux_fused_ln_modulate_is_bit_exact）：合并后的模型级接线测试，覆盖 FLUX/GLM-Image/Sana/ERNIE/LTX-2 等模型包装器与 gate 协议的正确配合，强调 bit-exact 路径用 torch.equal、质量门控路径用容差，并断言 gate 最终处于 verified 状态。

- test/registered/kernels/ops/diffusion/test_layout.py（模块 布局测试；类别 test；类型 test-coverage；符号 _cl3d, test_usp_merge_heads_bitwise, test_usp_merge_heads_unsupported_inputs_use_exact_fallback, test_pack_qkv_destination_major_is_bit_exact）：数据搬运类 kernel 的合并套件，覆盖 USP merge-heads、Ulysses QKV pack、varlen pack/scatter、causal Conv3d cat+pad 等；由于这些 kernel 仅搬运数值，全部用 torch.equal 做 bitwise 断言。
- python/sglang/kernels/ops/diffusion/README.md（模块 文档；类别 docs；类型 documentation）：新增的选择矩阵文档，列出九个看似可互换实则不同的 norm 实现各自的数值契约与支持的 shape/layout，是开发者查找 " 某个 fusion 是否可用 " 的权威入口。

关键符号：select_impl, can_use_wan_rmsnorm_silu, can_use_usp_merge_heads, can_use_ln_modulate, can_use_modulate_scale_shift_cuda, can_use_residual_gate_add_cuda, mount_fused_ln_modulate, BitExactFusionGate.accept_or_fallback, _module_defines, test_runtime_code_imports_only_through_the_facade, fused_ln_modulate, fused_ltx2_rms_norm_modulate

关键源码片段

test/registered/kernels/ops/diffusion/test_import_surface.py

新增的布局守护测试，用 AST 静态扫描双向校验导出表与注册表、验证门面惰性、禁止包外代码深导入子模块，是防止本次重组后布局再次退化的关键机制。

```
# test/registered/kernels/ops/diffusion/test_import_surface.py
# 这类守护测试把“只允许通过门面导入”的约定固化成可执行的断言，
# 防止将来有人为了图省事直接 import 子模块导致布局再次被锁死。


def _module_defines(module_path: str) -> set[str]:
    """读取子模块源码，找出其在顶层绑定的名字，但不真正 import 它。

    原因：import 会拉入 Triton / CuTe-DSL / FlyDSL 这些彼此互斥的重依赖，
    CPU CI lane 上没有它们；所以这里用 ast 静态解析替代 import。
    """
    path = _PACKAGE_DIR / (module_path.replace(".", "/") + ".py")
    if not path.exists():
        path = _PACKAGE_DIR / module_path.replace(".", "/") / "__init__.py"
    assert path.exists(), f"{PACKAGE}.{module_path} does not exist"

    names: set[str] = set()
    tree = ast.parse(path.read_text(encoding="utf-8"))
    for node in tree.body:
        if isinstance(node, (ast.FunctionDef, ast.AsyncFunctionDef, ast.ClassDef)):
            names.add(node.name)
        elif isinstance(node, ast.Assign):
            names.update(t.id for t in node.targets if isinstance(t, ast.Name))
        elif isinstance(node, (ast.Import, ast.ImportFrom)):
```

```
            names.update((a.asname or a.name).split(".")[0] for a in node.names)
        elif isinstance(node, (ast.If, ast.Try)):
            # 平台条件式 rebind（例如 x = select_impl(...)）依然绑定公开名字
            for inner in ast.walk(node):
                if isinstance(inner, (ast.FunctionDef, ast.ClassDef)):
                    names.add(inner.name)
                elif isinstance(inner, ast.Assign):
                    names.update(t.id for t in inner.targets if isinstance(t, ast.Name))
    return names


def test_runtime_code_imports_only_through_the_facade(root):
    """扫描整个仓库，禁止运行时代码 import diffusion 的子模块。

    只有门面（sglang.kernels.ops.diffusion）是唯一对外合法入口，
    这样内部布局将来可以再移动，而不会破坏任何调用点。
    """
    root_dir = _REPO_ROOT / root
    if not root_dir.exists(): # 源码检出才检查
        pytest.skip(f"{root} not present in this install")

    offenders = []
    for path in root_dir.rglob("*.py"):
        rel = path.relative_to(_REPO_ROOT).as_posix()
        if rel.startswith("python/sglang/kernels/ops/diffusion/"):
            continue # 包内部子模块互相 import 是重组的本意
        if rel in _DEEP_IMPORT_ALLOWLIST:
            continue
        try:
            tree = ast.parse(path.read_text(encoding="utf-8"))
        except (SyntaxError, UnicodeDecodeError):
            continue
        for node in ast.walk(tree):
            if (
                isinstance(node, ast.ImportFrom)
                and node.module
                and node.module.startswith(f"{PACKAGE}.")
            ):
                offenders.append(f"{rel}:{node.lineno} imports {node.module}")
            elif isinstance(node, ast.Import):
                offenders.extend(
                    f"{rel}:{node.lineno} imports {a.name}"
                    for a in node.names
                    if a.name.startswith(f"{PACKAGE}.")
                )
    assert not offenders, (
        "import from sglang.kernels.ops.diffusion instead of a submodule"
    )
```

test/registered/kernels/ops/diffusion/test_layout.py

数据搬运类 kernel 的合并套件，覆盖 USP merge-heads、Ulysses QKV pack、varlen pack/scatter、causal Conv3d cat+pad 等；由于这些 kernel 仅搬运数值，全部用 torch.equal 做 bitwise 断言。

```
# test/registered/kernels/ops/diffusion/test_layout.py
# 数据搬运 kernel 只移动数值，天然 bitwise 等价于它替换的 aten 链，
# 因此全文件用 torch.equal 而非 assert_close，容差会掩盖真实 bug。


def _build_mask(bs, s_txt, s_img, valid_txt_lens):
    """构造 varlen 测试用的 [B, S] 布尔掩码：文本段只保留前 vt 行，图像段全保留。"""
    mask = torch.zeros(bs, s_txt + s_img, dtype=torch.bool, device=DEVICE)
    for b, vt in enumerate(valid_txt_lens):
        mask[b, :vt] = True
        mask[b, s_txt:] = True
    return mask


@pytest.mark.parametrize("dtype", VARLEN_DTYPES)
@pytest.mark.parametrize("shape", VARLEN_SHAPES, ids=lambda s: s[0])
def test_varlen_pack_matches_index_select(dtype, shape):
    """fused_pack_qkv 必须与 index_select 严格一致，允许的形状矩阵覆盖生产 shape。"""
    _, bs, s_txt, s_img, num_heads, head_dim, valid_txt_lens = shape
    torch.manual_seed(0)
    s = s_txt + s_img
    indices, _ = _build_meta(_build_mask(bs, s_txt, s_img, valid_txt_lens))

    q, k, v = (
        torch.randn(bs, s, num_heads, head_dim, dtype=dtype, device=DEVICE)
        for _ in range(3)
    )
    fused = fused_pack_qkv(q, k, v, indices)
    for got, src in zip(fused, (q, k, v), strict=True):
        want = src.reshape(bs * s, num_heads, head_dim).index_select(0, indices)
        assert torch.equal(got, want)


def test_varlen_pack_handles_non_contiguous_input():
    """Q/K/V 可能以 (B, H, S, D) permute 形式到达，helper 必须自行 contig。"""
    torch.manual_seed(2)
    bs, s_txt, s_img, num_heads, head_dim = 2, 64, 128, 4, 64
    indices, _ = _build_meta(_build_mask(bs, s_txt, s_img, [32, 48]))

    pre = torch.randn(
        bs, num_heads, s_txt + s_img, head_dim, dtype=torch.bfloat16, device=DEVICE
    )
    q, k, v = (torch.randn_like(pre).permute(0, 2, 1, 3) for _ in range(3))
```

```
    assert not q.is_contiguous()

    fused = fused_pack_qkv(q, k, v, indices)
    for got, src in zip(fused, (q, k, v), strict=True):
        want = src.contiguous().flatten(0, 1).index_select(0, indices)
        assert torch.equal(got, want)
```

评论区精华

该 PR 的 review 评论数为 0，但 7 个 commit 的提交信息揭示了演进中的关键问题与决策：

> 关于测试套件合并后的 CI lane 继承："Merging N test files into one operator-domain suite also merges their CI lane registrations... test_layout.py picked up the B200 lane from the causal-Conv3d section, and the varlen<->USPAttention comparison folded in from test_varlen_uspattn_equivalence.py needs FlashAttention, which that image does ..."
>
> 关于 B200/HIP 专属测试 lane："test_rope.py carried the B200 lane because the LTX-2 split-RoPE kernel is validated there... the fused_inplace_qknorm_rope cases in that file are held to the split baseline... three bit-exact assertions that hold on H200 do not hold on B200."
>
> 关于平台谓词性能："is_cuda()/is_hip() in common/platform.py derived their answer through platform_key()... walks a chain of getattr(current_platform, f"is_{name}")() calls. Those predicates sit in per-call kernel guards... Delegate straight to that method instead."
>
> 关于 Sana 测试竞态："test_sana_fused_ln_modulate_is_bit_exact fills x/scale/shift on the default stream, then enters a fresh torch.cuda.Stream() and reads them there without making the side stream wait... if a mismatch disables _SANA_LN_MOD permanently, every later parametrization also stops registering its signature -- so one lost race fails three of the four cases."

这些都属于实现过程中发现并解决的问题，全部已解决。

- 测试套件合并后的 CI lane 继承 (testing): 拆分 lane 注册，按依赖隔离平台专属测试，避免测试被静默跳过或误跑。
- B200 与 HIP 专属测试 lane 分离 (testing): 将需要相反 runner 策略的两组用例拆分到不同文档与 lane。
- 平台谓词缓存与性能 (performance): 平台谓词改为直接委托 current_platform 的缓存方法，避免每次 kernel guard 重新解析平台。
- Sana 测试跨流竞态 (correctness): 修复流同步，避免 GPU 争用条件下偶发失败。

风险与影响

- 风险：主要风险集中在导入面与测试面：

- 跨模块大范围重构：167 个文件、141 处 import 改写，任何遗漏的深导入或导出表缺失都会在特定平台、特定模型运行时才暴露。虽然 test_import_surface.py 用 AST 静态扫描覆盖了导出表与合法 import 路径，但 allowlist 中仍有 3 个文件直接深导入（test 文件），属于特意放行。
- 懒门面依赖解析：__getattr__ 在首次访问时才解析符号，若导出表指向的模块路径错误，错误会延迟到模型调用那一刻才抛出，且只会发生在有对应后端的机器上，CPU/Apple 环境无法预演。
- 平台分发统一：common.platform.select_impl 替代了原有的 import-time 平台 if 链，新接缝需要保证懒加载语义与原有行为完全一致；提交 bb749c5 专门修正了谓词缓存问题，说明此处容易踩性能与正确性坑。
- 测试 lane 继承：合并套件后各部分的 CI lane 会互相继承，已出现过两次 lane 错配（B200 与 HIP 场景），若未来继续合入新的平台相关测试，需警惕同类问题。
- 验证范围有限：精度验证仅在 1x H200 上完成，FlyDSL（ROCm gfx950）与 LTX-2 split-RoPE（B200）的 13 个跳过用例没有全平台覆盖；PR body 还披露了一个与本次改动无关的 pre-existing 失败 test_diffusion_modelopt_fp8_scaled_mm.py，需在后续单独跟进。
- 影响：对用户与运行时透明：kernel 数学逻辑零变更，外部 API 面（通过门面的导入）保持兼容。对开发者的影响是结构性的——新增 diffusion kernel 时必须遵循按算子域放置、后端写进文件名、经由门面导出、按 can_use_*+raise 协议暴露、测试归入对应域名套件这几条约定，否则会被 test_import_surface.py 拦截。影响范围覆盖全部 32 个 consumer 模块（每个 DiT、VAE、layer 与 stage），所有 diffusion 相关 CI lane（CUDA/AMD/B200/nightly）均会运行重组合并后的测试套件。跨平台依赖被隔离到懒门面之后，未来接入新 backend（如 MLX）不再污染其他平台的 import 路径。
- 风险标记：跨模块大范围重构，import 面 141 处改动，懒门面依赖静态检查，仅 H200 单机验证，存在无关的 pre-existing 失败

关联脉络

- PR #31453 [Diffusion][Refactor] Refactor and extract complex RoPE implementation to layers/rotary_embedding for MOVA DiT: 同属 diffusion 内核区域的共享提取重构，与本次 rope/ 算子域重组方向一致，后续 RoPE kernel 的迁移可复用同一组织范式。
- PR #34933 [diffusion] Per-section LoRA adapters on fused linear layers: 依赖 fused linear 层与 diffusion 内核，PR 中 32 个 consumer 模块的 import 改写会直接影响 LoRA 路径的接线。
- PR #34197 [diffusion] RL rollout support for the Cosmos3 pipeline: 同为 diffusion 管道功能演进，涉及 pipeline 运行时与权重更新，使用的 kernel 入口可能因本 PR 的 can_fuse_-> can_use_ 协议变化而需要同步调整。