PR #34106 完整报告

sgl-project/sglang

[jit_kernel] Fix missing JIT kernel namespaces

合并时间：2026-08-08 22:15

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34106

执行摘要

这是一次针对 #33400 "Move JIT kernels into namespace sglang" 全局迁移的修补型变更，共改动 2 个 .cuh 头文件、净增加 6 行代码：

- 为 #32541 新增的单 token MoE 对齐内核 align_single_token.cuh 补上 namespace sglang 包裹（原为匿名命名空间）；
- 为 diffusion 的 modulate_scale_shift.cuh 在最内层伪命名空间之外补齐最外层 namespace sglang。

两处都只改命名空间包装、不改任何计算，配套 bit-exact 测试在 H200 上 12 项全部通过。属于低风险、低影响、但必要的合规性修复，延续了 JIT 内核命名空间统一约定。

功能与动机

PR body 明确说明这是 #33400 的后续修复：

> Fix the JIT namespace break from #33400 for the single-token MoE kernel added in #32541. Fix the same helper-namespace migration break in the diffusion adaLN modulation kernel. Restore first-use JIT compilation without changing either kernel's computation.

33400 将 python/sglang/kernels/jit/ 下所有 JIT C++ 统一迁入 namespace sglang，并让 load_jit 在命名空间内输出 TVM_FFI_DLL_EXPORT_TYPED_FUNC，因此 Python 侧调用 kernel_name 不再带 sglang:: 前缀。这意味着任何仍停留在全局或匿名命名空间的内核，都会在 JIT 首次编译时出现符号解析失败。

被遗漏的两个文件各有原因：align_single_token.cuh 是 #32541 在迁移之后新增的，天然不参与 #33400 的扫描；modulate_scale_shift.cuh 在 #33400 中只做了内部伪命名空间（sglang_modulate_scale_shift）的嵌套化，少包了一层最外层 sglang。

实现拆解

1. 补全 MoE 单 token 对齐内核命名空间 文件 python/sglang/kernels/jit/csrc/moe/align_single_token.cuh：将顶部匿名 namespace { ... } 改为 namespace sglang { ... }，并在文

件末尾把 } // namespace 的注释更新为 } // namespace sglang。改动后 AlignSingleTokenParams、AlignSingleTokenKernel 与其余 JIT 内核一样处于 sglang 命名空间下，load_jit 生成的无前缀导出即可正确解析。

2. 补齐 diffusion modulation 内核外层命名空间 文件 python/sglang/kernels/jit/csrc/diffusion/modulate_scale_shift.cuh：在原有 namespace sglang_modulate_scale_shift { namespace { ... } } 结构之外，再包裹一层 namespace sglang，并在文件尾闭合。这样该内核路径从全局命名空间变为 sglang::sglang_modulate_scale_shift，与 #33400 对 csrc/diffusion/ 伪命名空间的处理约定一致。
3. 验证与配套 无新增测试文件，但复用现有 test_modulate_scale_shift.py：H200 上清空 TVM-FFI 缓存后 12 项测试全部通过，覆盖 BF16/FP16 生产形状与边界形状，并要求与 eager PyTorch 逐位相等，证明修复不改变数值。pr-test 与 pr-test-extra 两个 CI 均通过。

python/sglang/kernels/jit/csrc/moe/align_single_token.cuh

32541 新增的单 token MoE 对齐内核，在 #33400 迁移时被遗漏，仍是匿名 namespace；本 PR 将其改为 namespace sglang，补齐 JIT 命名空间约定。

```
// python/sglang/kernels/jit/csrc/moe/align_single_token.cuh
// 修复 #33400 的命名空间迁移遗漏：该文件是 #32541 新增的
// 单 token MoE 对齐内核，原为匿名命名空间，改为 namespace sglang
#include <cstdint>

namespace sglang {

struct AlignSingleTokenParams {
    const int32_t* __restrict__ topk_ids; // [1, topk]
    // ... 其余字段未改动
};

struct AlignSingleTokenKernel {
    // ... 内核实现未做任何计算改动
};

} // namespace sglang
```

python/sglang/kernels/jit/csrc/diffusion/modulate_scale_shift.cuh

33400 只把内部伪命名空间 sglang_modulate_scale_shift 嵌套化，遗漏了最外层 namespace sglang；本 PR 补齐外层包裹，恢复 diffusion adaLN modulation 内核的首次编译。

```
// python/sglang/kernels/jit/csrc/diffusion/modulate_scale_shift.cuh
// 在伪命名空间 sglang_modulate_scale_shift 外层补齐 namespace sglang
#include <cstdint>
```

```
namespace sglang {

namespace sglang_modulate_scale_shift {
namespace {

struct ModulateScaleShiftKernel {
    // ... 内核实现未改动，bit-exact 测试保证与 eager PyTorch 逐位一致
};

} // namespace
} // namespace sglang_modulate_scale_shift

} // namespace sglang
```

评论区精华

该 PR 没有 review 评论，BBuf 直接 APPROVED。Issue 评论区只有两条 CI 机器操作：

> mmangkad: /tag-and-rerun-ci BBuf: /rerun-test test/registered/radix_cache/unified_radix_tree/test_unified_radix_cache_kl_dsv4.py

重跑结果通过。没有设计权衡或未解决疑虑。

风险与影响

- 回归风险低但存在：两个文件都是被多处 include 的 .cuh 头文件，若某调用方此前依赖匿名命名空间的全局可见性，补上 namespace sglang 后可能引发编译错误。不过 #33400 后所有 JIT 内核已统一该约定，方向正确。
- 数值零变化：modulate 内核有 bit-exact 测试兜底；align_single_token 虽无直接测试变更，但纯包装改动不涉及逻辑。
- 影响面限定在 JIT 首次编译：Restore first-use JIT compilation 意味着只有 fresh TVM-FFI cache 的场景受影响，已缓存部署无感知。
- 可维护性警示：此次遗漏说明 #33400 这类全局迁移很难覆盖迁移之后新增的文件，仓库已有的 test_kernels_namespace（#33400 中提到）这类扫描测试应保持运行，以把 namespace 约定固化为 CI 防线。

关联脉络

- #33400（Move JIT kernels into namespace sglang）：本 PR 是其直接遗漏补丁。该迁移当时虽做了全树 ::identifier 扫描，但无法覆盖后来新增的文件，也漏掉了 modulate_scale_shift.cuh 的外层包裹。
- #32541（[Kimi] Support kimi-k3）：单 token MoE 对齐内核 align_single_token.cuh 正是随 K3 支持引入的；K3 这条功能线随后还关联了 #33903（Inkling silu_and_mul 迁移纯 Triton）和 #34045（短卷积后端扩展）等 JIT/ 内核工作。

整体看，本 PR 是 JIT 内核 namespace 统一演进中的一次快速收尾，价值在于提醒团队：大范围机械迁移需要配合可自动扫描的防回归机制。