PR #32166 完整报告

sgl-project/sglang

[XPU] Use SYCL kernels for DeepSeek V4 MHC on XPU

合并时间：2026-08-25 10:27

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/32166

执行摘要

- 一句话：DeepSeek V4 MHC 在 XPU 上改用 SYCL 内核，提速最高 3 倍
- 推荐动作：建议熟悉 DeepSeek V4 或关注 XPU/NPU 内核集成的工程师精读此 PR。值得关注的设计决策：1）用 functools.cache + 延迟导入避免模型注册表急切导入触发可选 CUDA runtime；2）用 NamedTuple 统一多后端内核句柄，调用方不感知硬件差异；3）fail-fast 优于 fallback，避免 CUDA-only 文件被跨后端逻辑污染。后续建议补充 XPU 端到端回归测试，并在文档中说明 SGLANG_OPT_FUSE_MHC_POST_PRE 的默认值与行为。

功能与动机

PR body 明确说明：DeepSeek V4 的 MHC 路径需要启用 XPU 专属优化内核，因为现有 CUDA 路径内核（deep_gemm、CUDA-first device selection）在 Intel XPU 上不可用或次优，会造成错误的设备调度或缺失性能机会。因此需要把 hc_head 从 Triton 实现切换到 sgl_kernel.mhc.fused_hc_head（SYCL 内核），把 mhc_fused_post_pre 切换到 XPU 专属内核，并修复设备选择避免在 XPU 可用时回退到 CUDA。PR 附带的 BMG benchmark（hc_mult=4、bfloat16、50 次迭代中位数）显示 SYCL 内核相对 Triton 有 1.288x–2.983x 的加速。

实现拆解

变更入口为 python/sglang/srt/models/deepseek_v4.py，核心是建立“按硬件后端分发 MHC 内核”的统一机制：

1. 扩展 MhcOps 契约并重构 _get_mhc_ops()：在 MhcOps NamedTuple 中新增 mhc_pre、mhc_post、fused_hc_head 三个可选字段；XPU 分支从 sgl_kernel 懒加载 fused_hc_head、hc_post、hc_split_sinkhorn、mhc_fused_post_pre、mhc_pre 五个 SYCL 符号，非 XPU 分支保持从 sglang.kernels.ops.layernorm.mhc 导入 TileLang/CUDA 实现并将新字段置为 None。这样调用方只依赖字段名不感知硬件差异，同时保持“仅当 DeepSeek V4 层执行时才导入”的懒加载语义，避免模型注册表急切导入触发可选 CUDA runtime。
2. 接入 XPU mhc_pre 调用：在 hc_pre_torch_impl 的 XPU 分支中，将 norm.weight.data 与 norm.variance_epsilon 组装为 norm_kwargs，调用 _get_mhc_ops().mhc_pre(...)（SYCL 版），返回 (y, post, comb, norm is not None)；返回形参与非 XPU 路径对齐，后续算子无需区分内核来源。

3. 新增 fused post/pre 开关：新增 _is_fused_mhc_post_pre_enabled_xpu()，仅在 _is_xpu 时读取 envs.SGLANG_OPT_FUSE_MHC_POST_PRE；use_fused_mhc_post_pre 改为 is_cross_layer_mhc_fusion_enabled() or _is_fused_mhc_post_pre_enabled_xpu()，保证非 XPU 行为不变的同时允许 XPU 单独开启融合后处理。
4. 修复设备选择与后端隔离：XPU 上避免回退 CUDA 设备、不使用 deep_gemm；_prewarm_mhc_kernels 在 XPU 提前返回，跳过 TileLang prewarm；按 review 要求删除了对 _FP8_WO_A_GEMM 的 XPU 特殊屏蔽，待 FP8 支持就绪后再启用。
5. Review 收敛与测试取舍：按维护者要求，XPU 分支从 python/sglang/kernels/ops/layernorm/mhc.py 全部移除，该文件保持 CUDA/TileLang-only；_compute_num_split_for_mhc_pre 仅做等价的 n_sms 变量提取（+2 行）。原计划新增的 test/registered/xpu/test_hc_head.py 因含 benchmark 代码和多余 XPU skip，最终未合入，只改动 2 个源码文件。

关键文件：

- python/sglang/srt/models/deepseek_v4.py（模块 模型层；类别 source；类型 data-contract；符号 MhcOps, _get_mhc_ops, _is_fused_mhc_post_pre_enabled_xpu, hc_pre_torch_impl）：核心变更文件：扩展 MhcOps 契约，XPU 下从 sgl_kernel 懒加载 fused_hc_head、hc_post、mhc_pre、mhc_fused_post_pre 等 SYCL 内核，新增 _is_fused_mhc_post_pre_enabled_xpu() 开关，修改 use_fused_mhc_post_pre 与 hc_pre 分发，并修复 XPU 设备选择与 CUDA 回退问题。
- python/sglang/kernels/ops/layernorm/mhc.py（模块 内核层；类别 infra；类型 infrastructure；符号 _compute_num_split_for_mhc_pre）：按 review 要求收敛后的最终形态：从该文件移除所有 XPU 分支，保持 CUDA/TileLang-only；_compute_num_split_for_mhc_pre 仅做等价的 n_sms 变量提取，避免污染 CUDA-only 文件。

关键符号：_get_mhc_ops, MhcOps, _is_fused_mhc_post_pre_enabled_xpu, hc_pre_torch_impl, _compute_num_split_for_mhc_pre

关键源码片段

python/sglang/srt/models/deepseek_v4.py

核心变更文件：扩展MhcOps契约，XPU下从sgl_kernel懒加载fused_hc_head、hc_post、mhc_pre、mhc_fused_post_pre 等 SYCL 内核，新增 _is_fused_mhc_post_pre_enabled_xpu() 开关，修改 use_fused_mhc_post_pre 与 hc_pre 分发，并修复 XPU 设备选择与 CUDA 回退问题。

```
# python/sglang/srt/models/deepseek_v4.py
# MHC 内核句柄统一契约：不同硬件后端（CUDA/TileLang、NPU、XPU/SYCL）
# 通过同一个 NamedTuple 暴露各自可用的内核，调用方只依赖字段名，不感知硬件差异。
class MhcOps(NamedTuple):
    hc_split_sinkhorn: Callable[..., Any]
    mhc_fused_post_pre: Optional[Callable[..., Any]]
    npu_hc_pre: Optional[Callable[..., Any]]
```

```
    # 以下三个字段为 XPU SYCL 内核预留；CUDA/TileLang 与 NPU 路径恒为 None
    mhc_pre: Optional[Callable[..., Any]]
    mhc_post: Optional[Callable[..., Any]]
    fused_hc_head: Optional[Callable[..., Any]]


@functools.cache
def _get_mhc_ops() -> MhcOps:
    """按需加载 MHC 内核：只有 DeepSeek V4 层真正执行时才导入。

    模型模块会被注册表急切导入，若在此处直接导入持有 TileLang 内核的
    `sglang.kernels.ops.layernorm.mhc`，可能在无关模型初始化通信工作区
    之前就触发可选 CUDA runtime，因此必须延迟到层执行时再导入。
    DeepSeek V4 是唯一消费者。
    """
    if _is_xpu:
        # XPU 专属路径：从 sgl_kernel（sgl-kernel-xpu）懒加载 SYCL 内核，
        # 符号缺失时直接导入失败（fail fast），不做 try/except fallback，
        # 避免在 CUDA-only 文件里引入跨后端逻辑。
        from sgl_kernel import (
            fused_hc_head,
            hc_post,
            hc_split_sinkhorn,
            mhc_fused_post_pre,
            mhc_pre,
        )

        return MhcOps(
            hc_split_sinkhorn=hc_split_sinkhorn,
            mhc_fused_post_pre=mhc_fused_post_pre,
            npu_hc_pre=None,
            mhc_pre=mhc_pre,
            mhc_post=hc_post,
            fused_hc_head=fused_hc_head,
        )

    # CUDA/TileLang 与 NPU 共用原实现，新字段留空
    from sglang.kernels.ops.layernorm.mhc import (
        hc_split_sinkhorn,
        mhc_fused_post_pre,
        npu_hc_pre,
    )

    return MhcOps(
        hc_split_sinkhorn=hc_split_sinkhorn,
        mhc_fused_post_pre=mhc_fused_post_pre,
        npu_hc_pre=npu_hc_pre,
        mhc_pre=None,
        mhc_post=None,
```

```
        fused_hc_head=None,
    )


def _is_fused_mhc_post_pre_enabled_xpu() -> bool:
    # 仅在 XPU 上读取融合后处理开关，非 XPU 一律返回 False，
    # 保证 CUDA/NPU 原有行为完全不变
    if _is_xpu:
        return envs.SGLANG_OPT_FUSE_MHC_POST_PRE.get()

    return False
```

python/sglang/kernels/ops/layernorm/mhc.py

按 review 要求收敛后的最终形态：从该文件移除所有 XPU 分支，保持 CUDA/TileLang-only；_compute_num_split_for_mhc_pre 仅做等价的 n_sms 变量提取，避免污染 CUDA-only 文件。

```
# python/sglang/kernels/ops/layernorm/mhc.py
# 按 review 最终要求保持 CUDA-only：split count 仍使用 CUDA multi-processor
# count（等价提取 n_sms 变量），XPU 不再走 TileLang mhc_pre，本文件无需感知 XPU。
def _compute_num_split_for_mhc_pre(num_tokens: int, hc_hidden_size: int) -> int:
    block_m, block_k = 64, 64
    grid_size = (num_tokens + block_m - 1) // block_m
    num_block_k = (hc_hidden_size + block_k - 1) // block_k
    n_sms = torch.cuda.get_device_properties(0).multi_processor_count
    return max(1, min(n_sms // max(grid_size, 1), num_block_k // 4))
```

评论区精华

核心争议与决策集中在三处：1）mingfeima 明确反对在 CUDA/TileLang-only 的 mhc.py 中做 try/except + fallback，要求 XPU 分发全部收口到 deepseek_v4.py 的 _get_mhc_ops()，符号缺失直接 fail fast；2）mingfeima 提醒 NPU 兼容性问题（“is this going to break NPU？”），要求保留 npu_hc_pre 字段并把 XPU 内核统一放入 _get_mhc_ops()，cyxlily 据此重构；3）对于 _compute_num_split_for_mhc_pre，polisettyvarma 曾建议改用 get_device_core_count，cyxlily 说明 XPU 上 multi_processor_count 不可用（需用 gpu_eu_count），但 mingfeima 最终要求保持 CUDA-only 文件不动，该改动被回退。测试文件 test_hc_head.py 因含 benchmark 代码和多余 XPU skip 被要求清理，最终未合入。所有讨论线程均已解决，无遗留未解决疑虑。

- XPU 分发是否该进入 CUDA/TileLang-only 的 mhc.py (design): XPU 分支全部收口到 deepseek_v4.py 的 _get_mhc_ops()，mhc.py 保持 CUDA-only，缺失内核 fail fast。
- 替换 npu_hc_pre 是否破坏 NPU (correctness): cyxlily 恢复 npu_hc_pre 字段，并将 mhc_pre、hc_post、fused_hc_head 移入 _get_mhc_ops()；NPU 分支与原有行为保持一致。
- split count 计算是否改用 get_device_core_count (design): 保留 torch.cuda.get_device_properties(0).multi_processor_count 原逻辑，仅做等价的 n_sms 变量提取。

- UT 中移除 benchmark 代码与多余 XPU skip (testing): cyxlily 逐一移除相关代码；该测试文件最终未合入本 PR。
- XPU 上 FP8 flag 的特殊处理 (question): 最终删除对 XPU 的特殊屏蔽，保留原逻辑。

风险与影响

- 风险：1）NPU 回归风险：MhcOps 契约新增字段并调整 hc_pre 分发，review 中曾出现“是否破坏 NPU”的讨论；最终保留 npu_hc_pre 字段，但 deepseek_v4.py 属高频变更文件，后续合入需关注 NPU 相关 CI。2）依赖 sgl-kernel-xpu 版本：XPU 路径直接导入 fused_hc_head、hc_post、mhc_pre、mhc_fused_post_pre，若安装的 sgl_kernel 缺失这些 SYCL 符号会直接 ImportError；fail fast 是有意设计，但部署时需保证版本匹配（对应 sgl-kernel-xpu PR#345、PR#302）。3）测试覆盖缺口：原计划的 test_hc_head.py 最终未合入，XPU 路径缺少仓库内端到端回归测试，主要依赖 sgl-kernel-xpu 侧的 accuracy 验证。4）配置项默认值不确定：SGLANG_OPT_FUSE_MHC_POST_PRE 的默认值在本次材料中未给出，若默认开启会改变 XPU 下 fused post/pre 行为，建议核对 envs 定义。5）mhc.py 的 2 行改动仅为等价的 n_sms 变量提取，对 CUDA 路径无行为影响。
- 影响：影响范围集中在 DeepSeek V4 模型 + Intel XPU 硬件：对用户而言，在 XPU（BMG）上运行 DeepSeek V4 可获得 MHC 内核 1.29x–2.98x 的延迟改善，并修复 XPU 上误回退 CUDA 的设备调度问题；对系统而言，CUDA/TileLang 与 NPU 路径保持不变，改动不触及调度器、显存与通信模块。对工程团队而言，本 PR 确立了硬件专属内核的接入模式：通过 _get_mhc_ops() 懒加载 + NamedTuple 契约做后端分发，且坚持 CUDA-only 文件不被跨后端逻辑污染，该模式可复用于后续 XPU/NPU 内核集成。影响程度中等：功能开关由新环境变量控制，非 XPU 用户无感知。
- 风险标记：XPU 路径缺少合入测试，MhcOps 契约变更波及 NPU/CUDA 路径，依赖 sgl-kernel-xpu SYCL 符号版本，deepseek_v4.py 高频变更文件

关联脉络

- PR #35719 [AMD] Fix Qwen3.5 MTP dropping fused shared-expert weights: 同为 DeepSeek 系模型在非 CUDA 平台上的内核 / 权重处理修复，体现平台专属 dispatch 的演进脉络。
- PR #32039 [AMD][Fix] Route MoRI through the Qwen MoE all-to-all path: 同为修复非 CUDA 平台（AMD）上的路由与精度问题，与本次 XPU 设备选择与内核路由修复属于同类平台适配工作。