PR #49792 完整报告

vllm-project/vllm

[Kernel][SM100] Add a CuTeDSL fused query kernel

合并时间：2026-08-05 06:23

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/49792

执行摘要

- 一句话：SM100 新增 CuTeDSL 融合 query 内核，Triton 兜底
- 推荐动作：值得精读。本 PR 有两点设计可借鉴：一是分派闸门函数把 ' 宁可不走也不要错走 ' 的原则显式化，所有不满足条件的情况一律回退 Triton；二是 ' 非 torch.compile 路径上不要注册 custom op' 这一 review 结论，对后续 kernel 接入方式有指导意义。CuTeDSL 侧的 CTA 特化、PDL launch 与 inline asm 绕 bug 的写法也值得关注。建议阅读时结合 #48597 跟踪页理解其在系列中的定位，并留意后续是否补充与 Triton 输出的数值对比测试。

功能与动机

DSA 稀疏注意力模型（GLM-5.2、DeepSeek-V3.2）在 Blackwell decode 阶段需频繁执行 fused_q 查询预处理，原 Triton 实现未针对 SM100 特性优化。PR body 说明本内核 ' Selected only for the supported dtype/shape combination; the existing Triton implementation stays the fallback everywhere else'，目标是只对 SM100 特定路径加速而不影响其他场景。背景来自 #48597 跟踪页：原始 squash merge 被 #49768 回滚后，以 focused follow-up PR 方式重新拆分，并明确每个 follow-up PR 均在 8×B300 上做过聚焦 GPU 测试与模型评估。PR body 给出的基准显示，全系列可将 output 吞吐从 446.7 提升至 542.5 tok/s（中位 TPOT 1.94→1.56 ms）。

实现拆解

1. 新增核心内核文件 vllm/models/deepseek_v32/nvidia/ops/fused_q_cutedsl.py（537 行）：is_fused_q_cutedsl_supported 作为唯一分派闸门，依次校验 SM100 capability、quantize_mqa、bf16 dtype、RoPE 64 / NoPE 512 维度、MQA head 数为 4 的倍数、indexer 子张量规格；fused_q_cutedsl 负责参数整理与合并两个 scale；FusedQKernel 采用每 warp 处理一个 head group 的布局，mqa 与 indexer 在同一个内核内通过 CTA 特化（bid < num_mqa_ctas 分支）区分，并以 use_pdl=True 加 griddepcontrol_launch_dependents 支持程序化依赖启动，为后续 decode 内核依赖铺路。
2. 在公共入口 vllm/models/deepseek_v32/common/kernels.py 的 fused_q 中接入分派：新增 positions.dtype == torch.int64 与 q_scale 标量断言；因该函数与 ROCm 路径共享且 CuTeDSL 模块顶层 import cutlass，故在 current_platform.is_cuda() 内惰性导入；支持时直接调用 CuTeDSL 内核并返回，否则原样走 Triton _fused_q_kernel，两条实现共用同一套输出张量布局与返回契约。

3. 扩展 vllm/cute_utils 工具层：cvt.py 新增 fp32x2_to_fp8x2（PTX 内联汇编 cvt.rn.satfinite.e4m3x2.f32，绕开 TensorSSA fp32->fp8 转换 bug），并让 bf16x2_to_fp32x2 支持 bf16 张量先 recast_tensor 到 Uint32；__init__.py 的 _TORCH_TO_CUTE_DTYPE 补充 torch.float32 映射。
4. 新增基准脚本 benchmarks/kernels/benchmark_fused_q_cutedsl.py：按 GLM-5.2 DSA 真实维度（NoPE 512、RoPE 64、32 index heads、MTP=5 使并发 N 批为 6N tokens）构造输入，通过 patch.object(C, "is_fused_q_cutedsl_supported", ...) 强制走 Triton 或 CuTeDSL 路径做直接对比。
5. 测试与部署配套：本 PR 无独立单元测试文件，正确性仅靠 8×B300 手工验证与 benchmark；review 阶段移除了最初版本里的 direct_register_custom_op 注册与 fake 实现（模型不做 torch.compile，注册无收益）。

关键文件：

- vllm/models/deepseek_v32/nvidia/ops/fused_q_cutedsl.py（模块 查询内核；类别 source；类型 core-logic；符号 is_fused_q_cutedsl_supported, fused_q_cutedsl, FusedQKernel, kernel）：新增 537 行 SM100 CuTeDSL 内核，包含分派闸门 is_fused_q_cutedsl_supported、入口 fused_q_cutedsl 与 FusedQKernel（mqa/indexer CTA 特化、PDL 启动），是本 PR 的核心实现。
- vllm/models/deepseek_v32/common/kernels.py（模块 查询预处理；类别 source；类型 core-logic；符号 fused_q）：fused_q 公共入口新增 CuTeDSL 分派分支并保留 Triton fallback，同时兼容 ROCm 共享路径；新增断言收紧输入契约，是理解本 PR 如何接入的关键文件。
- vllm/cute_utils/cvt.py（模块 转换工具；类别 source；类型 core-logic；符号 fp32x2_to_fp8x2, bf16x2_to_fp32x2）：新增 fp32x2_to_fp8x2 PTX 转换，是 fp8 数值正确性的关键；同时扩展 bf16x2_to_fp32x2 支持 bf16 张量输入，供 CuTeDSL 内核复用。
- benchmarks/kernels/benchmark_fused_q_cutedsl.py（模块 内核基准；类别 test；类型 benchmark；符号 make_inputs, run, benchmark）：新增 Triton/CuTeDSL 双路径对比基准，按 GLM-5.2 DSA 真实维度构造输入，是本 PR 唯一的性能验证配套。
- vllm/cute_utils/__init__.py（模块 类型映射；类别 source；类型 configuration）：dtype 映射表补充 torch.float32 -> Float32，是内核参数类型映射的前提。
- vllm/models/deepseek_v32/nvidia/ops/__init__.py（模块 包初始化；类别 infra；类型 infrastructure）：新增 nvidia.ops 包初始化文件，用于承载 nvidia 专属算子模块。

关键符号：is_fused_q_cutedsl_supported, fused_q_cutedsl, FusedQKernel.kernel, FusedQKernel.mqa, FusedQKernel.indexer, fused_q, fp32x2_to_fp8x2, bf16x2_to_fp32x2

关键源码片段

vllm/models/deepseek_v32/nvidia/ops/fused_q_cutedsl.py

新增 537 行 SM100 CuTeDSL 内核，包含分派闸门 is_fused_q_cutedsl_supported、入口 fused_q_cutedsl 与 FusedQKernel（mqa/indexer CTA 特化、PDL 启动），是本 PR 的核心实现。

分派闸门与网格启动

```
# vllm/models/deepseek_v32/nvidia/ops/fused_q_cutedsl.py

def is_fused_q_cutedsl_supported(
    q_pe: torch.Tensor,
    index_q: torch.Tensor | None,
    ql_nope: torch.Tensor,
    *,
    has_indexer: bool,
    quantize_mqa: bool,
) -> bool:
    # 分派闸门：宁可回退 Triton，也不要让内核内部断言在线上触发。
    # 任何一项不满足都应返回 False 走原有路径。
    if not (
        current_platform.has_device_capability(100) # 仅 SM100（B300）
        and quantize_mqa # 仅 fp8 packed query 路径
        and q_pe.dtype == ql_nope.dtype == torch.bfloat16
        and q_pe.shape[-1] == 64 # RoPE 维度固定 64
        and ql_nope.shape[-1] == 512 # NoPE 维度固定 512
        and q_pe.shape[1] % 4 == 0 # 每 warp 一组 head，需 4 的倍数
    ):
        return False
    # 有 indexer 时再校验 indexer 子张量的规格
    return not has_indexer or (
        index_q is not None
        and index_q.dtype == torch.bfloat16
        and index_q.shape[1] % 16 == 0
        and index_q.shape[-1] == 128
    )


# FusedQKernel.__call__：按是否包含 indexer 计算 CTA 网格并启动内核
if cutlass.const_expr(self.idx_dim == 0):
    # 无 indexer：grid 第二维为 mqa CTA 数（每个 CTA 4 个 warps 处理一个 head group）
    grid = (num_tokens, self.num_ctas_per_tok, 1)
else:
    # 有 indexer：一维 grid 拼接 mqa CTA 与 indexer CTA，内核内按 bid 特化
    num_mqa_ctas = num_tokens * self.num_ctas_per_tok
    num_idx_ctas = num_tokens * self.num_ctas_per_idx_tok
    grid = (num_mqa_ctas + num_idx_ctas, 1, 1)

self.kernel(
    positions,
    q_pe,
    q_pe_rope_cache,
    ql_nope,
    q_scale,
    mqa_output,
```

```
    idx_q,
    idx_q_rope_cache,
    idx_weights,
    idx_q_fp8,
    idx_weights_out,
    weight_scale,
).launch(
    grid=grid,
    block=(self.num_warps * 32, 1, 1),
    stream=stream,
    use_pdl=True, # 程序化依赖启动，供后续 decode 内核通过 griddepcontrol 依赖本内核
)
```

mqa 方法的 NoPE / RoPE 处理核心

```
# FusedQKernel.mqa：每个 warp 处理一个 MQA head group 的 NoPE 与 RoPE 两大块，
# 展示单 token 单 head 的核心计算与访存重叠。

# 尽早发出所有加载，最大化访存与计算重叠
rQ_nope_bf16 = cute.make_rmem_tensor(16, BFloat16) # NoPE 部分 16 个 bf16 元素
rQ_rope_bf16 = cute.make_rmem_tensor(2, BFloat16) # RoPE 部分 2 个 bf16 元素
src_ql_nope = cute.local_tile(ql_nope[token_id, head_id, None], (16,), (lane_id,))
src_q_rope = cute.local_tile(q_pe[token_id, head_id, None], (2,), (lane_id,))
cute.copy(cp_32B, src_ql_nope, rQ_nope_bf16) # 32 字节向量化加载
cute.copy(cp_4B, src_q_rope, rQ_rope_bf16) # RoPE 只有 4 字节
rCos_raw = q_pe_rope_cache[pos, 0 + lane_id]
rSin_raw = q_pe_rope_cache[pos, 32 + lane_id]

# NoPE 块：bf16 -> fp32 提升、除以 q_scale 后量化成 fp8 e4m3，写出到 mqa_output 前段
rQ_nope_f32 = cvt.bf16x2_to_fp32x2(rQ_nope_bf16).load() * inv_scale
rQ_nope_f8 = cute.make_rmem_tensor(16, Float8E4M3FN)
rQ_nope_f8.store(rQ_nope_f32.to(Float8E4M3FN))
cute.copy(cp_16B, rQ_nope_f8, dst_Q_nope)

# RoPE 块：fp32 旋转（cos/sin）再缩放；随后通知依赖本内核的 decode 内核可启动
rQ_rope_f32 = cvt.bf16x2_to_fp32x2(rQ_rope_bf16)
r0 = (rQ_rope_f32[0] * rCos - rQ_rope_f32[1] * rSin) * inv_scale
r1 = (rQ_rope_f32[1] * rCos + rQ_rope_f32[0] * rSin) * inv_scale
cute.arch.griddepcontrol_launch_dependents()
# TensorSSA 的 fp32->fp8 转换存在 bug，这里依赖直接 PTX（见 cvt.fp32x2_to_fp8x2）
rQ_rope_f8 = cute.make_rmem_tensor(2, Float8E4M3FN)
cute.recast_tensor(rQ_rope_f8, Uint16)[0] = cvt.fp32x2_to_fp8x2(r0, r1)
cute.copy(cp_2B, rQ_rope_f8, dst_Q_rope) # 写出到 mqa_output 的 RoPE 段
```

vllm/models/deepseek_v32/common/kernels.py

fused_q 公共入口新增 CuTeDSL 分派分支并保留 Triton fallback，同时兼容 ROCm 共享路径；新增断言收紧输入契约，是理解本 PR 如何接入的关键文件。

common/kernels.py 中的分派片段

```
# vllm/models/deepseek_v32/common/kernels.py 中的 fused_q 分派片段
# fused_q 是 DSA 稀疏注意力的查询预处理公共入口，与 ROCm 路径共享；
# 而 CuTeDSL 模块在模块顶层 import cutlass，因此只能在 CUDA 平台惰性导入，
# 避免破坏 ROCm 等非 CUDA 后端的导入链。

# 本 PR 新增的输入契约校验：
# assert positions.dtype == torch.int64
# assert q_scale.dtype == torch.float32 and q_scale.numel() == 1

# 分派：先把候选内核置空，未命中任何条件时保持 None，继续走 Triton 内核。
cutedsl_kernel: Callable[..., None] | None = None
if current_platform.is_cuda():
    from vllm.models.deepseek_v32.nvidia.ops.fused_q_cutedsl import (
        fused_q_cutedsl,
        is_fused_q_cutedsl_supported,
    )

    if is_fused_q_cutedsl_supported(
        q_pe,
        index_q,
        ql_nope,
        has_indexer=has_indexer,
        quantize_mqa=quantize_mqa,
    ):
        cutedsl_kernel = fused_q_cutedsl

# 输出张量分配逻辑与 Triton 路径完全共用：quantize_mqa=True 时
# mqa_q 为 fp8 packed 的 [ql_nope; q_pe] 输出，否则为 bf16 的 RoPE 输出；
# index_q_fp8 与 index_weights_out 同样由这条路径统一分配，
# 因此两条实现的返回张量布局保持一致。

if cutedsl_kernel is not None:
    cutedsl_kernel(
        positions,
        q_pe,
        q_pe_cos_sin_cache,
        ql_nope,
        q_scale,
        mqa_q,
        index_q,
        index_q_cos_sin_cache,
        index_weights,
        index_weights_softmax_scale,
        index_weights_head_scale,
        index_q_fp8,
        index_weights_out,
        has_indexer=has_indexer,
        index_rope_interleave=index_rope_interleave,
    )
```

```
    return index_q_fp8, index_weights_out, mqa_q

# 未命中 CuTeDSL 时回落原始 Triton 网格启动，行为与合入前完全一致。
_fused_q_kernel[(3, num_tokens, grid_heads)](...)
```

vllm/cute_utils/cvt.py

新增 fp32x2_to_fp8x2 PTX 转换，是 fp8 数值正确性的关键；同时扩展 bf16x2_to_fp32x2 支持 bf16 张量输入，供 CuTeDSL 内核复用。

cvt.py 新增与调整的转换 op

```
# vllm/cute_utils/cvt.py —— 为 CuTeDSL 内核新增的 fp32 转 fp8 打包转换
# 说明：cute 的 TensorSSA fp32->fp8 转换存在 bug，因此依赖 PTX 内联汇编。
@dsl_user_op
def fp32x2_to_fp8x2(a0: Float32, a1: Float32, *, loc=None, ip=None) -> Uint16:
    out = llvm.inline_asm(
        T.i16(),
        [a0.ir_value(loc=loc, ip=ip), a1.ir_value(loc=loc, ip=ip)],
        "cvt.rn.satfinite.e4m3x2.f32 $0, $2, $1;", # 两个 fp32 打包为 fp8 e4m3 对
        "=h,f,f", # 输出 16 位寄存器，两个输入为 fp32
        has_side_effects=False,
        is_align_stack=False,
    )
    return Uint16(out)


# bf16x2_to_fp32x2 对 bf16 张量输入的新增处理：先 recast 成 Uint32 再逐元素拆解
elif isinstance(data, (cute.Tensor, cute.TensorSSA)):
    if data.element_type == BFloat16:
        data = cute.recast_tensor(data, Uint32)
    assert data.element_type == Uint32
    size = cute.size(data.shape)
    out = cute.make_rmem_tensor(size * 2, Float32)
    for i in range(size):
        out[i * 2], out[i * 2 + 1] = bf16x2_to_fp32x2(data[i])
    return out
```

评论区精华

review 中两条核心讨论均来自 gau-nernst，且都已被作者采纳：一是质疑 torch custom op 注册的必要性——vllm/models 下的模型不经过 torch.compile，direct_register_custom_op 及其 fake impl 没有收益，zhou9402 回复 'make sense, remove for fix.' 并在后续提交中移除；二是建议把 has_cutedsl() 与 SM100 能力检查折叠进 is_fused_q_cutedsl_supported（NVIDIA 平台因 FlashAttention 4 与 FlashInfer 已保证具备 cutlass-dsl），作者回复 'fixed'，最终 gate 内直接检查 has_device_capability(100)，lazy import 仍保留 current_platform.is_cuda() 保护以兼容 ROCm 共享路径。此外 Claude Code Review 因 fork 自动禁用，最终由 gau-nernst 与 WoosukKwon 分别 approve。

- 是否注册为 torch custom op (design): 作者在后续提交中移除了 custom op 注册与 fake 实现，改为在 fused_q 内直接调用 fused_q_cutedsl。
- SM100/cutedsl 能力检查的位置 (design): 最终 is_fused_q_cutedsl_supported 内部直接检查 current_platform.has_device_capability(100)；lazy import 仍以 current_platform.is_cuda() 保护，避免影响 ROCm 共享路径。

风险与影响

- 风险：正确性风险：fused_q 处于 decode 热路径，分派条件与内核内部断言（num_heads % 4、index heads % 16 等）必须严格一致，否则线上可能 ' 走了新路径却触发内核 assert 或产生错误结果 '；当前靠 is_fused_q_cutedsl_supported 兜底，改动后需要同步维护两份约束。数值风险：fp8 转换依赖 PTX 内联汇编绕开 TensorSSA bug，行为与 cutlass-dsl 版本耦合，升级可能暴露或修复该 bug 导致行为漂移。回归风险：common/kernels.py 的 fused_q 与 ROCm 路径共享，若未来移除 current_platform.is_cuda() 保护，非 CUDA 平台会在 import 阶段加载 cutlass 而出错；此外自定义 op 注册被移除后，函数调用走普通 Python 路径，后续若引入 torch.compile 需重新评估。测试缺口：无自动化测试，SM100 依赖使常规 CI 无法覆盖，回归只能依赖 B300 手工验证。性能风险：单独合入仅 +0.4%，主要收益依赖 #48597 系列其余 PR，若系列后续不落地，本 PR 的维护成本大于直接收益。
- 影响：影响范围：仅 SM100（B300）+ DSA 模型（GLM-5.2、DeepSeek-V3.2 等）+ fp8 packed query + MTP decode 且满足 shape 约束的场景走新内核；其他硬件、模型、dtype 组合完全回退 Triton，行为不变。性能影响：单独合入 output tok/s 446.7→448.4、中位 TPOT 1.94→1.93 ms；与 #49790、#49793、#50230 等配合达到 542.5 tok/s（约 +21%）。团队影响：新增 537 行 CuTeDSL 内核维护点与 benchmark 基线，对 cutlass-dsl 版本升级敏感，同时为后续 SM100 DSA 优化提供了可对照的参考实现。影响程度整体低 - 中，受硬件与模型组合限制，但位于 decode 热路径且缺少测试保护。
- 风险标记：缺少测试覆盖，SM100 硬件依赖，内联汇编依赖，热路径变更

关联脉络

- PR #48597 [Perf][GLM-5.2] Blackwell decode optimizations: 跟踪页：本 PR 是 #48597 重拆分（re-split）中的 'CuTeDSL fused-query kernel' 一项，原始 squash merge 被 #49768 revert；PR body 中的基准数据与合并顺序均来自该 tracker。
- PR #49790 SM100 sparse-model integration and routing: tracker 明确 #49790 必须先合入，main 上才有 deepseek_v32 包可达性；本 PR 的内核依赖其 SparseAttention 模型接入。
- PR #50230 Programmatic dependent launch for the decode kernels: 本 PR 内核已采用 use_pdl=True 与 griddepcontrol 原语，与 #50230 的程序化依赖启动方案紧密配合，是全系列 decode 优化的一部分。
- PR #49793 MTP/speculative-decoding optimizations: tracker 中与 fused_q 同属 decode 热路径的优化项，全系列吞吐提升（542.5 tok/s）依赖二者叠加。