

PR #55426 完整报告

vllm-project/vllm

[Bugfix][Kimi-K3] Fix KDA projection overlap on Hopper

合并时间: 2026-09-12 03:07

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/55426>

执行摘要

- 一句话: 修复 Hopper 上 Kimi-K3 KDA 投影重叠编译崩溃
- 推荐动作: 值得精读, 特别是关注 CUTLASS/CUTE DSL 内核跨架构兼容的工程师。本 PR 展示了三个可借鉴的设计点: 用内联汇编实现 PTX 指令的可移植降级 (`_fma_f32_bf16_portable`)、用 `cutlass.const_expr` 做编译期内核分支、以及用能力判定函数 (`_kda_qkvg_flashinfer_backend/_has_mixed_precision_bf16_fma`) 把架构差异收敛到单一入口, 避免在调用点散落条件判断。

功能与动机

issue #55350 报告: Kimi-K3 在 H100 上启用 #54697 引入的低 M TP8 KDA 投影重叠路径后, CUDA graph profiling 期间 `kda_skinny_gemm.run_n()` 在 CUTLASS DSL 编译器报 `NVVM_ERROR_COMPILATION`: PTX `fma.f32.bf16` 需要 SM100+, 而 FlashInfer 的 `cute-dsl` QKVG 后端也拒绝 SM90。PR body 明确说明设计取向: *#55108 avoids the failure by disabling KDA projection overlap on Hopper. This PR keeps the overlap structure enabled while selecting architecture-safe operations.*, 即避免用整体禁用换取启动可用, 而是保留重叠结构并做架构安全操作选择。

实现拆解

1. 计算能力判定抽象: 在 `low_latency_gemm.py` 新增 `_kda_qkvg_flashinfer_backend()`, 仅当 `current_platform.is_device_capability((10, 0))` 或 `(10, 3)` 时返回 "cute-dsl", 其余能力返回 `None`; 在 `kda_skinny_gemm.py` 新增 `_has_mixed_precision_bf16_fma()`, 用 `current_platform.has_device_capability(100)` 对应 PTX ISA 中 `fma.f32.bf16` 的 SM100 门槛。
2. skinny GEMM 可移植化: `kda_skinny_gemm.py` 新增 DSL 用户算子 `_fma_f32_bf16_portable`, 通过 LLVM 内联汇编把两个 BF16 操作数先用 `cvt.f32.bf16` 转成 FP32, 再执行 `fma.rn.f32`, 实现 FP32 累加的等价语义; `_KdaSkinnyNGemm` 与 `_KdaSkinnyKGemmm` 构造函数增加 `use_mixed_precision_fma` 标志, `_compile_n/_compile_k` 按当前能力传入, 内核循环里用 `cutlass.const_expr` 做编译期静态分支, SM90 生成可移植版本而不是无条件发射 `fma.f32.bf16`。
3. QKVG 回退与 autotune 联动: `run_kda_projection_overlap` 的 `run_qkvg` 内, 仅在 `_kda_qkvg_flashinfer_backend()` 返回非 `None` 时才调用 `flashinfer.gemm.mm_bf16`, SM90 等架构走 `torch.mm`; `autotune_kda_qkvg` 同步改为只在 backend 可用时采样权重

做 FlashInfer autotune, 避免在 SM90 上触发不支持的 `cute-dsl` 校验, 但仍会把 `_projection_overlap_max_tokens` 置为 14, 保证重叠结构启用。

4. 测试配套: `tests/kernels/test_bf16_skinny_gemm.py` 新增 `test_kda_mixed_precision_bf16_fma_capability` (表驱动验证 (9,0)/(10,0)/(10,3)/(12,0) 判定)、`test_kda_skinny_gemms_sm90_cuda_graph` (在 SM90 上编译 `run_n/run_k` 并做 CUDA graph 捕获与重放, 与 FP32 参考做余弦校验)、`test_kda_qkv_flashinfer_backend` (验证各能力下 backend 选择), 并把 `test_kda_qkv_autotune_enables_full_overlap` 参数化为 `backend=None` 与 `"cute-dsl"` 两种情况。
5. 验证结果: 作者在 H100 单卡上以 `VLLM_BATCH_INVARIANT=0` 验证 rows 2-14 编译、2/8/14 的 CUDA graph 捕获重放、3/14 的完整重叠投影, 数值最大绝对误差 portable FMA 为 $1.22e-4$ 、QKVG `torch.mm` 分支为 $4.88e-4$, 余弦相似度均大于 0.9999998; 另有 32 x H100 四节点端到端 4000/4000 请求通过。

关键文件:

- `vllm/models/kimi_k3/nvidia/low_latency_gemm.py` (模块 投影调度; 类别 source; 类型 core-logic; 符号 `_kda_qkv_flashinfer_backend`, `run_kda_projection_overlap`, `run_qkv`, `autotune_kda_qkv`): 投影重叠的调度入口: 新增 `_kda_qkv_flashinfer_backend()` 按计算能力决定 QKVG 是否走 FlashInfer `cute-dsl`, 并联动 `run_qkv` 与 `autotune_kda_qkv`, 在 SM90 上回退 `torch.mm` 但保留 `F_A/beta` 与 `F_B` 并发分支, 是整个修复的决策中枢。
- `vllm/models/kimi_k3/nvidia/ops/cute_dsl/kda_skinny_gemm.py` (模块 GEMM 内核; 类别 source; 类型 kernel-impl; 符号 `_fma_f32_bf16_portable`, `_has_mixed_precision_bf16_fma`, `_KdaSkinnyNGemm`, `_KdaSkinnyKGemm`): 内核层修复: 新增 `_fma_f32_bf16_portable` 与 `_has_mixed_precision_bf16_fma()`, 让 skinny N/K GEMM 在 SM90 上编译 FP32 转换后的 FMA, 避免 sm_90a 上 libNVVM 编译失败, 是本 PR 真正消除编译崩溃的底层改动。
- `tests/kernels/test_bf16_skinny_gemm.py` (模块 内核测试; 类别 test; 类型 test-coverage; 符号 `test_kda_mixed_precision_bf16_fma_capability`, `test_kda_skinny_gemms_sm90_cuda_graph`, `test_kda_qkv_flashinfer_backend`): 新增 SM90 专项回归测试: `capability` 判定表驱动、真实 CUDA graph 捕获与重放、QKVG 后端选择与 `autotune` 参数化, 直接覆盖 issue #55350 的失败场景, 防止 Hopper 回归。

关键符号: `_kda_qkv_flashinfer_backend`, `run_kda_projection_overlap`, `run_qkv`, `autotune_kda_qkv`, `_fma_f32_bf16_portable`, `_has_mixed_precision_bf16_fma`, `test_kda_mixed_precision_bf16_fma_capability`, `test_kda_skinny_gemms_sm90_cuda_graph`, `test_kda_qkv_flashinfer_backend`

关键源码片段

`vllm/models/kimi_k3/nvidia/low_latency_gemm.py`

投影重叠的调度入口: 新增 `_kda_qkv_flashinfer_backend()` 按计算能力决定 QKVG 是否走 FlashInfer `cute-dsl`, 并联动 `run_qkv` 与 `autotune_kda_qkv`, 在 SM90 上回退 `torch.mm`

但保留 F_A/beta 与 F_B 并发分支，是整个修复的决策中枢。

```
def _kda_qkvg_flashinfer_backend() -> Literal["cute-dsl"] | None:
    """返回当前 GPU 计算能力下可用的 FlashInfer QKVG 后端。
```

FlashInfer 的 cute-dsl BF16 GEMM 在后台校验时拒绝 SM90，只对 SM100 与 SM103 返回 cute-dsl；其他能力（含 SM90）返回 None，让调用方回退到 torch.mm，从而保留并发投影重叠结构，避免 CUDA graph profiling 阶段的内核编译失败。

```
"""
```

```
if current_platform.is_device_capability(
    (10, 0)
) or current_platform.is_device_capability((10, 3)):
    return "cute-dsl"
return None
```

```
def run_qkvg() -> torch.Tensor:
```

```
    # M=1/2 走预测量的 shape_dynamic_skinny_gemm 专用配置
```

```
    if config := KDA_QKVG_CONFIGS.get(num_tokens):
```

```
        return shape_dynamic_skinny_gemm(
            hidden_states,
            qkvg_weight,
            config,
            None,
        )
```

```
    # 仅当 FlashInfer 后端可用（SM100/SM103）时才走 mm_bf16；
```

```
    # SM90 等架构这里会拿到 None，直接落入 torch.mm 回退
```

```
    if num_tokens <= KDA_PROJECTION_OVERLAP_MAX_TOKENS and (
        backend := _kda_qkvg_flashinfer_backend()
```

```
    ):
```

```
        from flashinfer.gemm import mm_bf16
```

```
        return mm_bf16(
            hidden_states,
            qkvg_weight.t(),
            pdl=True,
            backend=backend,
        )
```

```
    # 架构安全回退：保持并发结构但使用通用 GEMM
```

```
    return torch.mm(hidden_states, qkvg_weight.t())
```

[vllm/models/kimi_k3/nvidia/ops/cute_dsl/kda_skinny_gemm.py](#)

内核层修复：新增 `_fma_f32_bf16_portable` 与 `_has_mixed_precision_bf16_fma()`，让 skinny N/K GEMM 在 SM90 上编译 FP32 转换后的 FMA，避免 `sm_90a` 上 libNVVM 编译失败，是本 PR 真正消除编译崩溃的底层改动。

```
@dsl_user_op
```

```
def _fma_f32_bf16_portable(
```

```

a: BFloat16,
b: BFloat16,
acc: Float32,
*,
loc=None,
ip=None,
) -> Float32:
    """BF16 乘加 + FP32 累加的可移植实现，面向 pre-SM100 GPU。

```

PTX fma.f32.bf16 要求 SM100 或更新；在 sm_90a 上无条件发射会让 libNVVM 编译失败（即 issue #55350 的报错）。这里通过内联汇编把两个 BF16 操作数先用 cvt.f32.bf16 转成 FP32，再执行一次 FP32 fma.rn.f32，语义与混合精度 FMA 等价但可在 Hopper 上编译。

```

    """
    a_bits = llvm.bitcast(T.i16(), a.ir_value(loc=loc, ip=ip), loc=loc, ip=ip)
    b_bits = llvm.bitcast(T.i16(), b.ir_value(loc=loc, ip=ip), loc=loc, ip=ip)
    result = llvm.inline_asm(
        T.f32(),
        [a_bits, b_bits, acc.ir_value(loc=loc, ip=ip)],
        "{\n\t"
        ".reg .f32 a_f32, b_f32;\n\t"
        "cvt.f32.bf16 a_f32, $1;\n\t"
        "cvt.f32.bf16 b_f32, $2;\n\t"
        "fma.rn.f32 $0, a_f32, b_f32, $3;\n\t"
        "}",
        "=f,h,h,f",
        has_side_effects=False,
        is_align_stack=False,
        asm_dialect=llvm.AsmDialect.AD_ATT,
        loc=loc,
        ip=ip,
    )
    return Float32(result)

```

```

def _has_mixed_precision_bf16_fma() -> bool:
    """PTX fma.f32.bf16 是否受当前 GPU 支持（SM100 及以上）。"""
    return current_platform.has_device_capability(100)

```

```

for mi in cutlass.range_constexpr(num_rows):
    for ni in cutlass.range_constexpr(split_n):
        for vi in cutlass.range_constexpr(_FAB_VECTOR_WIDTH):
            # 编译期常量分支：SM100+ 用原生混合精度 FMA,
            # Hopper 等旧架构用先转 FP32 再 FMA 的可移植实现
            if cutlass.const_expr(self.use_mixed_precision_fma):
                acc[mi, ni] = _fma_f32_bf16(
                    a_regs[mi, vi],
                    weight_regs[ni, vi],
                    acc[mi, ni],

```

```

    )
else:
    acc[mi, ni] = _fma_f32_bf16_portable(
        a_regs[mi, vi],
        weight_regs[ni, vi],
        acc[mi, ni],
    )

```

tests/kernels/test_bf16_skinny_gemm.py

新增 SM90 专项回归测试：capability 判定表驱动、真实 CUDA graph 捕获与重放、QKVG 后端选择与 autotune 参数化，直接覆盖 issue #55350 的失败场景，防止 Hopper 回归。

```
@pytest.mark.parametrize("num_tokens", [2, 8, 14])
```

```
def test_kda_skinny_gemms_sm90_cuda_graph(num_tokens: int) -> None:
```

```
    """Hopper 必须能编译并捕获 pre-SM100 的 FMA 特化内核。
```

该测试直接覆盖 issue #55350 的失败场景：在 SM90 上编译 run_n/run_k，并在 CUDA graph 捕获与重放后与 FP32 参考做余弦相似度校验，防止 sm_90a 编译回归。

```
    """
```

```
    _require_capability_and_cute((9, 0))
```

```
    from vllm.models.kimi_k3.nvidia.ops.cute_dsl.kda_skinny_gemm import (
```

```
        KdaSkinnyGemm,
```

```
)
```

```
    torch.manual_seed(43 + num_tokens)
```

```
    hidden_states = torch.randn(num_tokens, 7168, dtype=torch.bfloat16, device="cuda")
```

```
    f_ab_weight = torch.randn(144, 7168, dtype=torch.bfloat16, device="cuda")
```

```
    f_b_weight = torch.randn(1536, 128, dtype=torch.bfloat16, device="cuda")
```

```
    gemm = KdaSkinnyGemm()
```

```
    gemm.run_k(gemm.run_n(hidden_states, f_ab_weight), f_b_weight)
```

```
    torch.accelerator.synchronize()
```

```
    # 在独立 stream 上捕获 CUDA graph，随后重放验证内核可被 graph 化
```

```
    capture_stream = torch.cuda.Stream()
```

```
    capture_stream.wait_stream(torch.cuda.current_stream())
```

```
    graph = torch.cuda.CUDAGraph()
```

```
    with torch.cuda.graph(graph, stream=capture_stream):
```

```
        projected_fab = gemm.run_n(hidden_states, f_ab_weight)
```

```
        projected_fb = gemm.run_k(projected_fab, f_b_weight)
```

```
    torch.cuda.current_stream().wait_stream(capture_stream)
```

```
    graph.replay()
```

```
    torch.accelerator.synchronize()
```

```
    expected_fab = torch.nn.functional.linear(
```

```
        hidden_states.float(), f_ab_weight.float()
```

```
)
```

```
    expected_fb = torch.nn.functional.linear(
```

```

    expected_fab[:, :128].to(torch.bfloat16).float(), f_b_weight.float()
)
# 与 FP32 参考比较, 保证 portable FMA 精度足够
for actual, expected in (
    (projected_fab, expected_fab),
    (projected_fb, expected_fb),
):
    cosine = torch.nn.functional.cosine_similarity(
        actual.float().flatten(), expected.flatten(), dim=0
    ).item()
    assert cosine > 0.999

```

评论区精华

- H200 大规模验证 (elvircrn) : 在 Kimi K3 P/D TP8+DP/EP 4 的 64 GPU H200 集群上实测 gsm8k: exact_match 0.9689 (flexible-extract 5-shot) 与 0.9697 (strict-match 5-shot) , 并给出 lgtm, 说明 SM90 另一代表架构上无精度回归。
- CI 与合入确认 (tlrmchlsmth) : 两次 /ci run 分别触发 Buildkite CI #88371 (commit 9f53c14be251) 与 #88373 (commit 1c45deca2695) , 随后 APPROVED 合入。
- 设计权衡留待后续 (PR body) : 作者明确声明 A matched comparison against #55108's full-disable approach remains useful follow-up work, 即与 #55108 整体禁用方案的定量性能对比尚未完成, 属于未解决事项。
- 自动化审阅无异议: CodeRabbit 两次自动 review 均未产生 actionable comments。
 - H200 大规模端到端验证 gsm8k (testing): elvircrn 给出 lgtm, SM90 另一代表架构验证通过。
 - CI 触发与合并批准 (other): 两轮 CI 通过后合入 main。
 - 与 #55108 全禁用方案的定量对比留待后续 (design): 未解决, 作为后续工作; 本 PR 先保证正确性与可用性。

风险与影响

- 风险:
 - 性能回退风险: SM90 上 QKVG 从 FlashInfer mm_bf16/cute-dsl 变为 torch.mm, 且 skinny kernels 的 BF16 混合精度 FMA 变为先转 FP32 再 FMA, 低 M 投影的理论峰值可能下降; 正确性优先, 收益尚未与 #55108 全禁用方案做定量对比。
 - 精度变化: portable FMA 最大绝对误差 1.22e-4, QKVG torch.mm 分支 4.88e-4, 均来自 FP32 参考对比且余弦相似度大于 0.9999998; gsm8k 无回归, 但长文本生成等场景仍建议关注。
 - 覆盖缺口: 测试表确认 (10,7) 与 (12,0) 等未来能力也返回 None 走 torch.mm, 这些架构没有实机验证; 若后续新架构支持 fma.f32.bf16, 需同步维护 _has_mixed_precision_bf16_fma()。
 - CUDA graph 路径: SM90 测试仅覆盖 rows 2/8/14 的 graph 捕获重放, 作者 preflight 验证了 rows 2-14 编译, 但完整矩阵的 graph 行为仍依赖有限采样。

- 影响：对 H100/H200 (SM90) 的 Kimi-K3 用户，本 PR 直接消除启动期 NVVM_ERROR_COMPILATION 崩溃，且相比 #55108 的禁用方案保留投影重叠的并发结构，避免完全放弃该优化；SM100/SM103 行为保持不变。对系统而言，修复横跨内核编译 (kda_skinny_gemm.py)、后端选择与 autotune (low_latency_gemm.py) 三层，形成按计算能力选择架构安全操作的通用模式，可复用到其他模型的跨架构内核。对团队而言，新增的 SM90 CUDA graph 回归测试为后续 Hopper 内核变更提供直接防线。
- 风险标记：架构特化分支，核心路径变更，精度变化，CUDA graph 覆盖

关联脉络

- PR #54697 Overlap low-M TP8 KDA projections: 引入 KDA projection overlap 与 kda_skinny_gemm.py 的原始 PR；issue #55350 明确该路径由 #54697 引入，本 PR 修复其 SM90 编译兼容性。
- PR #55108 Avoid KDA projection failure on Hopper by disabling overlap: PR body 提到 #55108 用禁用 KDA projection overlap 的方式规避 SM90 失败；本 PR 是保留 overlap 结构的替代修复方案，二者需做性能对比。
- PR #53948 PP prototype (本 PR 集成镜像包含)：PR body 说明四节点 32 x H100 端到端镜像基于 main + 本 PR + #53948，将 PP 原型作为启动 / 正确性回归的伴随变更。
- PR #55356 [Kimi Perf] Group fp8 mla cahche insertion, 4~6x kernel level performance improvement for small batch: 同仓库近期 Kimi-K3 nvidia 性能与内核工作，说明该模型在 Hopper 之后的 Kernel 层持续演进。