PR #48666 完整报告

vllm-project/vllm

[Kernel] Gemma-4 FA4 FP8 Kernel

合并时间：2026-08-14 08:42

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/48666

执行摘要

- 一句话：Gemma-4 接入 SM90 FA4 FP8 内核，修复 MTP scale
- 推荐动作：值得精读。关注点包括：内核接入路径（flash_attn_interface.py）与块大小契约上报的耦合方式；config-aware 后端 API 为未来动态 block size 选择打下的基础；MTP KV scale 复制对 device/float/host 三种表示的细致处理；以及 mgoin 与作者的性能争论揭示的基准口径问题——对比不同 attention 后端时必须先确认负载参数一致。

功能与动机

Gemma-4 使用 256 宽 head 的 sliding_attention 与 512 宽 head 的 full_attention，在 SM90 上 full-attention 层从 FA3 升级到 FA4 CuTeDSL 内核，但此前 FA4 缺少 FP8 KV cache 支持。PR body 明确目标："wires the FA4 FP8-KV-dequant path from vllm-project/flash-attention#164 into vLLM, allowing Gemma-4 to use FP8 KV cache across both FA3 sliding attention and FA4 full attention"。另一个动机是修复 MTP：BF16 draft 模型以 Q-only attention 读取 target 模型的共享 KV cache，仅 alias cache 张量会让 draft attention 模块保留默认的 1.0 K/V scale，导致 FP8 KV 被错误解量化，因此需要复制 target attention 模块的 device 与 host K/V scale。

实现拆解

1. 接入 FA4 FP8-KV 内核与依赖升级：vllm/vllm_flash_attn/flash_attn_interface.py 的 flash_attn_varlen_func 检测 k.dtype == torch.float8_e4m3fn 且设备 capability 为 9（SM90）时，进入 FA4 fp8-KV 反量化分支，仅转发 K/V descale 并设置 fp8_kv_dequant 标志，Q 保持原生 FP16/BF16；cmake/external_projects/vllm_flash_attn.cmake 同步升级 FlashAttention pin 以包含该内核。
2. 上报内核块大小契约：vllm/v1/attention/backends/flash_attn.py 新增 _get_sm90_fa4_fp8_kv_block_size，当满足 SM90 + FP8 KV + head_size 512 + FA4 时返回 64，否则退回 MultipleOf(16)；vllm/v1/attention/backend.py 新增 config-aware 的 get_supported_kernel_block_sizes_for_config 与 get_preferred_block_size_for_config 基类方法，默认委托给无配置版本；vllm/platforms/interface.py 的 update_block_size_for_backend 与 vllm/model_executor/layers/attention/attention.py 的 _largest_kernel_block_within 改为显式传 vllm_config，移除对全局 set_current_vllm_config 上下文的依赖。

3. 修正 FP8 KV 下的量化输入判断：flash_attn.py 的 __init__ 中，FA4 + SM90 + FP8 KV 时把 supports_quant_query_input 置为 False，因为该路径在内核内反量化 FP8 K/V；其他 FA4 路径（如 SM100）仍要求 Q/K/V 同为 FP8 dtype。
4. 修复 Gemma-4 MTP 共享 KV scale：vllm/v1/worker/gpu/spec_decode/gemma4/speculator.py 新增 _copy_target_kv_scales，对 _k_scale/_v_scale 用 detach().clone() 重新 register_buffer（避免 alias target 参数），对 _k_scale_float/_v_scale_float 直接引用，对 _k_scale_cpu/_v_scale_cpu 用 copy_ 保留引用；同时把 _setup_gemma4_kv_sharing 的 target 层解析从前缀与序号拼接改为基于真实层索引的 target_names_by_index 映射。
5. 配套与验证：按 review 建议删除了临时测试文件与多余 clone；PR body 给出 4 项 accuracy 基准（与 BF16 TRITON 差距均在 1.2pp 内）、服务性能扫描（FP8 FA4 最高 261.6 tok/s/GPU）与 MTP 测试（512/512 请求无错，FP8 配置吞吐 +7.4%）。合并前经过多次 rebase main 解决冲突，CI 多次触发通过。

关键文件：

- vllm/v1/attention/backends/flash_attn.py（模块 注意力后端；类别 source；类型 core-logic；符号 _get_sm90_fa4_fp8_kv_block_size, get_supported_kernel_block_sizes, get_supported_kernel_block_sizes_for_config, get_preferred_block_size_for_config）：核心变更文件：为 SM90 FA4 FP8-KV 内核定义 64-token 块大小契约，并调整 supports_quant_query_input，决定 Gemma-4 上 FA4 与 FA3 如何协同使用 FP8 KV cache。
- vllm/v1/worker/gpu/spec_decode/gemma4/speculator.py（模块 投机解码；类别 source；类型 core-logic；符号 _copy_target_kv_scales, _setup_gemma4_kv_sharing）：修复 BF16 MTP draft 模型共享 target FP8 KV 时 K/V scale 未同步的问题，是 FP8 MTP 精度的关键。
- vllm/v1/attention/backend.py（模块 后端基类；类别 source；类型 core-logic；符号 get_supported_kernel_block_sizes_for_config, get_preferred_block_size_for_config）：新增 config-aware 基类 API，使块大小选择可在有具体配置时被后端覆盖，是此次管道重构的基础。
- vllm/model_executor/layers/attention/attention.py（模块 注意力层；类别 source；类型 data-contract；符号 _largest_kernel_block_within, get_kv_cache_spec）：数据契约调整：_largest_kernel_block_within 增加 vllm_config 参数，使滑动窗口切片也能感知后端的内核块大小契约。
- vllm/vllm_flash_attn/flash_attn_interface.py（模块 内核接口；类别 source；类型 core-logic；符号 flash_attn_varlen_func）：FA4 FP8-KV 反量化路径的接入点，决定内核如何被 varlen 接口调用。
- vllm/platforms/interface.py（模块 平台层；类别 source；类型 dependency-wiring；符号 update_block_size_for_backend）：移除 set_current_vllm_config hack，改用新增的 config-aware API 获取首选块大小。
- vllm/v1/attention/backends/fa_utils.py（模块 辅助工具；类别 source；类型 core-logic）：配合新内核路径的辅助调整。
- cmake/external_projects/vllm_flash_attn.cmake（模块 构建脚本；类别 infra；类型 configuration）：升级 FlashAttention 依赖 pin，以包含 FA4 FP8-KV 反量化内核。

关键符号：_get_sm90_fa4_fp8_kv_block_size, get_supported_kernel_block_sizes_for_config, get_preferred_block_size_for_config, _copy_target_kv_scales, _setup_gemma4_kv_sharing, _largest_kernel_block_within, flash_attn_varlen_func, update_block_size_for_backend

关键源码片段

vllm/v1/attention/backends/flash_attn.py

核心变更文件：为 SM90 FA4 FP8-KV 内核定义 64-token 块大小契约，并调整 supports_quant_query_input，决定 Gemma-4 上 FA4 与 FA3 如何协同使用 FP8 KV cache。

```
# vllm/v1/attention/backends/flash_attn.py
# SM90 FA4 FP8-KV 内核使用固定 64-token 的 TMA tile/page 作为页大小契约，
# 与通用 FlashAttention 的 16 的倍数能力不同，必须按具体引擎配置上报，
# 否则 sliding-window 缓存规划会选 32 导致内核无法使用。

@staticmethod
def _get_sm90_fa4_fp8_kv_block_size(
    vllm_config: VllmConfig | None = None,
) -> int | None:
    # 没有显式配置时回落到当前全局配置，兼容无 vllm_config 的调用点。
    if vllm_config is None:
        vllm_config = get_current_vllm_config_or_none()
    if vllm_config is None or vllm_config.model_config is None:
        return None

    head_size = vllm_config.model_config.get_head_size()
    if (
        current_platform.is_device_capability_family(90)
        and vllm_config.cache_config.cache_dtype in ('fp8', 'fp8_e4m3')
        and head_size == 512
        and get_flash_attn_version(head_size=head_size) == 4
    ):
        # SM90 上 FA4 的 FP8-KV 反量化内核要求页大小为 64 token。
        return 64
    return None

@classmethod
def get_supported_kernel_block_sizes(cls) -> list[int | MultipleOf]:
    # sliding-window 缓存规格会挑选最小的可上报 size，因此这里必须
    # 返回内核的真实页大小契约，而不是通用的 MultipleOf(16)。
    if block_size := cls._get_sm90_fa4_fp8_kv_block_size():
        return [block_size]
    return [MultipleOf(16)]

@classmethod
def get_supported_kernel_block_sizes_for_config(
    cls, vllm_config: VllmConfig
```

```
) -> list[int | MultipleOf]:
    # 有明确引擎配置时直接判定，避免依赖全局 current config 上下文。
    if block_size := cls._get_sm90_fa4_fp8_kv_block_size(vllm_config):
        return [block_size]
    return [MultipleOf(16)]

@classmethod
def get_preferred_block_size_for_config(
    cls, default_block_size: int, vllm_config: VllmConfig
) -> int:
    # 配置感知版本，供缓存规划拿到 vllm_config 时直接调用。
    if block_size := cls._get_sm90_fa4_fp8_kv_block_size(vllm_config):
        return max(default_block_size, block_size)
    if current_platform.is_xpu():
        return max(default_block_size, 64)
    return super().get_preferred_block_size(default_block_size)

# 初始化中修正 quant query 判断：
# FA4 的 SM90 FP8-KV 路径直接消费原生 FP16/BF16 Q，并在内核内部完成
# FP8 K/V 的反量化；而其他 FA4 路径（如 SM100）仍要求 Q/K/V 为同一
# FP8 dtype，因此仅在该场景禁用 quant query 输入，避免重复量化。
uses_sm90_fa4_fp8_kv_dequant = (
    self.vllm_flash_attn_version == 4
    and current_platform.is_device_capability_family(90)
    and self.kv_cache_dtype in ('fp8', 'fp8_e4m3')
)
self.supports_quant_query_input = (
    flash_attn_supports_quant_query_input()
    and not uses_sm90_fa4_fp8_kv_dequant
)
```

vllm/v1/worker/gpu/spec_decode/gemma4/speculator.py

修复 BF16 MTP draft 模型共享 target FP8 KV 时 K/V scale 未同步的问题，是 FP8 MTP 精度的关键。

```
# vllm/v1/worker/gpu/spec_decode/gemma4/speculator.py
# BF16 draft 模型以 Q-only 方式读取 target 模型的共享 KV cache，
# 仅 alias cache 张量会让 draft attention 模块保留默认的 K/V scale=1.0，
# 对 FP8 KV cache 会得到错误的解量化结果，因此需要把 target 侧的
# K/V scale（含 device、float 与 host 三份表示）完整复制到 draft 层。

def _copy_target_kv_scales(attn: nn.Module, target_attn: nn.Module) -> None:
    # 默认 scale 是标量 buffer，而部分量化方法会替换成 length-one 或
    # per-head 参数；用 detach().clone() 重新注册 buffer，既保留 target
    # 的值与 shape，又不 alias target 的参数（避免状态污染）。
    for scale_name in ('_k_scale', '_v_scale'):
        target_scale = getattr(target_attn, scale_name)
        attn.register_buffer(scale_name, target_scale.detach().clone())
    # float 表示形式与 device 张量值一致，直接引用 target 的可读属性。
```

```
    for scale_name in ('_k_scale_float', '_v_scale_float'):
        setattr(attn, scale_name, getattr(target_attn, scale_name))
    # host 侧同步副本需要已存在的 buffer，用 copy_ 原地覆盖保持引用稳定。
    for scale_name in ('_k_scale_cpu', '_v_scale_cpu'):
        getattr(attn, scale_name).copy_(getattr(target_attn, scale_name))
```

vllm/vllm_flash_attn/flash_attn_interface.py

FA4 FP8-KV 反量化路径的接入点，决定内核如何被 varlen 接口调用。

```
# vllm/vllm_flash_attn/flash_attn_interface.py
# SM90 FA4 fp8-KV 路径：paged FP8 e4m3 K/V 由内核反量化，K/V descale
# 在 kernel 内折叠；该路径接受 bf16/fp16 Q，并以原生 dtype 写出 O
#（无需 Q 转换与输出拷贝）。只有 (batch, num_kv_heads) 形状的 f32
# K/V descale 需要转发给内核。
fa4_fp8_kv_dequant = (
    k.dtype == torch.float8_e4m3fn
    and torch.cuda.get_device_capability()[0] == 9
)
if fa4_fp8_kv_dequant:
    fa4_q_descale = None
    fa4_k_descale = k_descale
    fa4_v_descale = v_descale
else:
    fa4_q_descale = None
    fa4_k_descale = None
    fa4_v_descale = None

out, softmax_lse, _, _ = _flash_attn_fwd(
    q, k,
    # 其余参数保持原样 ...
    q_descale=fa4_q_descale,
    k_descale=fa4_k_descale,
    v_descale=fa4_v_descale,
    output_scale=output_scale,
    fp8_kv_dequant=fa4_fp8_kv_dequant,
)
```

评论区精华

review 的核心交锋围绕三点：一是 zyongye 质疑 attention.py 中 with set_current_vllm_config(vllm_config): 包装仅为测试用途，作者先解释其必要性（sliding-window 缓存规划会得到 32，而 SM90 FA4 FP8-KV 内核需要 64 页大小），随后接受建议，通过新增 config-aware 基类方法完成管道并移除包装；二是 zyongye 指出 flash_attn_interface.py 中多余 clone，作者直接删除；三是 zyongye 认为新增测试文件不必要，作者移除。PR 合并前还有一次重要的性能争论：mgoin 报告 nightly 上 auto-selected FA4 明显慢于 Triton（3,067.7 vs 4,713.8 tok/s），作者回应内核面向 static quantization with kv scale、可考虑 opt-in，随后复测定位为 CI 负载缺少 skip_special_tokens=False 参数，补上后 FA4 反而快 60%。

- set_current_vllm_config 包装是否为临时测试代码 (design): 接受重构建议，引入 get_supported_kernel_block_sizes_for_config 与 get_preferred_block_size_for_config ，移除 with set_current_vllm_config。
- flash_attn_interface 中额外 clone 是否必要 (question): 多余克隆被删除。
- 新增测试文件是否必要 (testing): 测试文件从 PR 中移除。
- FA4 自动选择在 Hopper nightly 上性能回退 (performance): 性能回退缩因是 CI 负载未传 skip_special_tokens=false 导致比较失真；真实 FLASH_ATTN 场景 FA4 明显更快，自动选择逻辑保留。

风险与影响

- 风险：主要风险集中在四处：其一，flash_attn.py 的 FA4 自动选择逻辑可能在一些量化变体（如 per-tensor vs per-block scale）或特定负载形态上引入性能回退，mgoin 的 nightly 报告说明该风险真实存在，且当前完全依赖手工 benchmark 验证；其二，flash_attn_interface.py 新增 fp8_kv_dequant 参数依赖 FlashAttention pin 升级，若用户侧构建未同步 pin，接口可能不兼容；其三，_copy_target_kv_scales 假定 target attention 一定存在 _k_scale/_v_scale 等属性，若未来量化方案改变属性集合会抛 AttributeError，且 _k_scale_cpu 的原地 copy_ 要求 draft 侧 buffer 已初始化；其四，本 PR 缺少直接对应的自动化测试，精度与 MTP 行为主要依赖手工 benchmark，回归风险较高。
- 影响：对用户：启用 FP8 KV cache 的 Gemma-4 在 H100/H200（SM90）上会自动使用 FA4 内核，decode 吞吐显著提升（作者数据最多 2.21x），但 BF16 用户需要显式 --attention-config.flash_attn_version=4 才能走 FA4，FP8 用户则应保持 version 不设或 3，避免 sliding attention 层被错误升级。对系统：新增 config-aware block size 选择 API，影响 KV cache 页大小规划（SM90 FA4 FP8 场景强制 64 token 页）与显存占用粒度，所有继承 AttentionBackend 的后端都会看到新的基类方法。对团队：需要与 flash-attention 仓库保持 pin 同步，性能基准对比必须统一负载参数（如 skip_special_tokens），否则容易重现 mgoin 的误判。
- 风险标记：核心注意力路径变更，FA4 自动选择潜在性能回退，依赖 FlashAttention pin 升级，MTP scale 复制隐含属性假设，缺少自动化回归测试

关联脉络

- PR #53017 [Model Runner V2][Spec Decode] Fix draft logits cache column stride in gumbel_sample: 同属 v1 spec_decode 与 draft 采样路径的近期修复，与本 PR 的 MTP draft/KV 共享改造在同一子系统，可对照验证采样与缓存正确性。
- PR #52078 [Attention] Avoid redundant mask compute in GDN metadata build: 同为 v1 注意力后端的近期性能优化，与 flash_attn 后端的 block/ 页规划逻辑相邻。
- PR #52839 [refactor] consolidate cp attn ops: 对 v1 attention ops 与 MLA 注意力层的大范围重构，与本 PR 扩展注意力后端 API 处于同一演进线，后续后端都会继承 config-aware 基类方法。
- PR #52998 [Distributed] Enable FlashInfer all-reduce by default: 与 FA4 自动选择类似，属于 v1 后端默认行为切换，性能验证容易受基准口径影响，mgoin 的对比方法可作为参

考。