PR #49718 完整报告

vllm-project/vllm

[Attention] Add FlashInfer XQA decode support on SM12x

合并时间：2026-08-12 07:32

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/49718

执行摘要

- 一句话：SM12x 启用 FlashInfer XQA 解码，支持投机解码与 CUDA 图
- 推荐动作：建议精读 vllm/v1/attention/backends/flashinfer.py 中 draft mask 打包与 _compute_decode_query_lens 的处理，它们展示了投机解码与 CUDA 图 padding 共存时的元数据设计权衡；同时把 use_dedicated_xqa 作为临时方案理解，结合 #49818 与后续统一重构一起追踪最稳妥。

功能与动机

SM12x（RTX 50 系列）此前没有可用的 XQA decode 路径，decode 性能与投机解码能力受限。FlashInfer 0.6.16.post1 通过上游 flashinfer#4137 与 #4199 提供专用 XQA API，使 vLLM 可以在 SM12x 上路由到该内核。PR body 明确目标为“Enable FlashInfer XQA decode on SM120/SM121 through FlashInfer's dedicated XQA API”，并说明这不是 #47306 或 #37592 的重复——那些草稿针对旧共享 decode API 与更早版本。作者还披露 concurrency-1 场景暴露了 CUDA 图 padding 错位，由当前 q_cu_seq_lens 处理修复，但修复后尚缺 GPU 复跑。

实现拆解

1. 平台判定与依赖接入（vllm/utils/flashinfer.py）：新增 flashinfer_xqa_batch_decode_with_kv_cache 懒加载入口并加入 __all__；supports_trtllm_attention() 把 SM12x 并入“仅 decode”分支，与 SM90 同等对待；use_trtllm_attention() 的自动检测在 SM90/SM12x 且 KV cache 为 FP8 时优先选择 XQA decode。该层决定后端路由可行域，配套测试 test_supports_sm12x_decode_only 验证 SM12x 上 decode 为 True、prefill 为 False。
2. 后端路由与元数据扩展（vllm/v1/attention/backends/flashinfer.py）：FlashInferMetadataBuilder.__init__ 新增 use_dedicated_xqa 标记（仅当 SM12x 且 decode kernel 为 XQA 时置真），并将 supports_spec_as_decode 扩展为包含该路径；FlashInferTrtllmAPIDecode 增加 q_len_per_req、q_cu_seq_lens、mask 三个字段，承载 uniform/ragged 投机解码信息。build() 中通过 _compute_decode_query_lens() 根据 qo_indptr 计算每请求有效 query 宽度，CUDA 图 padding 被保留为零长度请求，且 uniform 路径要求“解码 token 数 = 请求数 × 每请求 query 数”的精确乘积，否则退回 ragged 元数据。

3. draft mask 构造与缓存：新增 _pack_draft_block_bool_mask（32-bit 打包成 XQA 的 uint16 布局）、_make_xqa_draft_block_mask 与 _make_xqa_ragged_draft_block_mask（分别处理 uniform/ragged、因果 / 非因果）；builder 维护 _decode_mask_cache，以 (q_len, causal) 为 key 缓存持久 mask，确保 CUDA 图重放时 tensor 地址稳定。
4. CUDA graph 与 sink 能力扩展：get_cudagraph_support() 对 SM12x 放开至 UNIFORM_BATCH（即使启用非因果注意力），但 DCP（decode context parallel）场景保守降级为单 token decode；supports_sink() 对 SM12x 返回 XQA decode 可用性；get_q_data_type() 让 SM12x 在 FP8 KV 下 decode 使用 BF16/FP16 Q（与 SM90 对齐），并避免在只有 FA2 的架构上使用 FP8 Q。forward 路径在 decode_with_dedicated_xqa 分支集中处理断言与 o_sf_scale/use_dcp 参数。
5. 测试与验证配套：tests/v1/attention/test_attention_backends.py 新增 4 类用例——XQA mask 期望值单测、CUDA graph padding 保留、精确 uniform 乘积校验、sink prefill 与 XQA decode 对 SDPA 的数值正确性；同时把 run_attention_backend/_test_backend_correctness 扩展为支持 attention sinks 的参考实现。tests/kernels/attention/test_use_trtllm_attention.py 新增 SM12x 平台判定用例。PR 验证给出 SM121 GSM8K（并发 64）0.8764 flexible / 0.8749 strict 的结果；作者注明本地无 .venv/CUDA，部分 pytest 未实际执行，且 concurrency-1 修复需要 GPU 复跑。

关键文件：

- vllm/v1/attention/backends/flashinfer.py（模块 注意力后端；类别 source；类型 core-logic；符号 _pack_draft_block_bool_mask, _make_xqa_draft_block_mask, _make_xqa_ragged_draft_block_mask, _compute_decode_query_lens）：核心实现文件：新增 SM12x 专用 XQA decode 路径，扩展 FlashInferTrtllmAPIDecode 元数据（q_len_per_req/q_cu_seq_lens/mask），新增 draft mask 打包与查询长度推导逻辑，并调整 CUDA graph、sink、Q dtype 等能力判定。
- tests/v1/attention/test_attention_backends.py（模块 后端测试；类别 test；类型 test-coverage；符号 run_attention_backend, _test_backend_correctness, test_flashinfer_xqa_draft_masks, test_flashinfer_xqa_query_lens_preserve_cudagraph_padding）：测试配套关键文件：覆盖 XQA draft mask 的期望打包值、CUDA 图 padding 保留、精确 uniform 乘积校验、sink prefill 与 XQA decode 对 SDPA 的数值正确性，并把参考实现扩展为支持 attention sinks。
- vllm/utils/flashinfer.py（模块 能力判定；类别 source；类型 core-logic；符号 flashinfer_xqa_batch_decode_with_kv_cache, supports_trtllm_attention, use_trtllm_attention）：平台能力判定与 XQA API 懒加载入口：决定 SM12x 是否可自动选择 XQA decode，并同步 supports_trtllm_attention/use_trtllm_attention 的判定口径。
- tests/kernels/attention/test_use_trtllm_attention.py（模块 内核测试；类别 test；类型 test-coverage；符号 test_supports_sm12x_decode_only）：补充 SM12x 平台判定单测，验证新增的 decode-only 分支与 SM90 对称，防止后续平台判定回归。

关键符号：_pack_draft_block_bool_mask, _make_xqa_draft_block_mask, _make_xqa_ragged_draft_block_mask, _compute_decode_query_lens, _get_decode_mask, supports_trtllm_attention, use_trtllm_attention, FlashInferBackend.supports_sink, FlashInferBackend.get_cudagraph_support,

FlashInferBackend.get_q_data_type, FlashInferMetadataBuilder.build

关键源码片段

vllm/v1/attention/backends/flashinfer.py

核心实现文件：新增 SM12x 专用 XQA decode 路径，扩展 FlashInferTrtllmAPIDecode 元数据（q_len_per_req/q_cu_seq_lens/mask），新增 draft mask 打包与查询长度推导逻辑，并调整 CUDA graph、sink、Q dtype 等能力判定。

```
def _pack_draft_block_bool_mask(
    bool_mask: torch.Tensor,
    num_packed: int,
) -> torch.Tensor:
    """把布尔 draft mask 打包成 XQA 要求的 uint16 位布局。

    XQA 的 draft mask 按每 32 个 key 位置压缩成一个位段，而不是逐个
    bool；这里用 int64 位权求和完成折叠，最后按 uint16 重排。
    """
    num_rows = bool_mask.shape[0]
    # 32 位块内每位的权重，用于把最后一维折叠成整数
    bits = 1 << torch.arange(32, device=bool_mask.device, dtype=torch.int64)
    mask_u32 = (
        (bool_mask.view(num_rows, num_packed, 32).to(torch.int64) * bits)
        .sum(dim=-1)
        .to(torch.uint32)
    )
    # 每个 32 位段转成两个 uint16，得到 XQA 的 packed 布局
    return mask_u32.view(torch.uint16).reshape(num_rows, num_packed * 2)


def _make_xqa_draft_block_mask(
    q_len: int,
    causal: bool,
    device: torch.device,
) -> torch.Tensor:
    """构建 uniform 请求的 packed XQA draft mask。

    causal 时第 i 个 query 只能看到 <= i 的 key 位置；非 causal 时
    所有 query 看到全部 32 位对齐前的 key。补零到 32 的倍数只为对齐位打包。
    """
    num_packed = (q_len + 31) // 32
    padded = num_packed * 32
    q_idx = torch.arange(q_len, device=device).unsqueeze(1)
    kv_idx = torch.arange(padded, device=device).unsqueeze(0)
    bool_mask = kv_idx <= q_idx if causal else (kv_idx < q_len).expand(q_len, padded)
    return _pack_draft_block_bool_mask(bool_mask, num_packed)


def _make_xqa_ragged_draft_block_mask(
```

```
    q_lens: list[int],
    max_q_len: int,
    causal: bool,
    device: torch.device,
) -> torch.Tensor:
    """构建 ragged 批次的 packed draft mask。

    投机解码时各请求可携带不同数量的 draft token：row_lens 用
    repeat_interleave 把每请求的 query 数展开到对应行，request_starts
    定位每个请求在总 query 序列中的起始偏移。
    """
    num_packed = (max_q_len + 31) // 32
    padded = num_packed * 32
    q_lens_t = torch.tensor(q_lens, device=device)
    row_lens = torch.repeat_interleave(q_lens_t, q_lens_t)
    request_starts = torch.cumsum(q_lens_t, dim=0) - q_lens_t
    row_starts = torch.repeat_interleave(request_starts, q_lens_t)
    q_idx = torch.arange(sum(q_lens), device=device) - row_starts
    kv_idx = torch.arange(padded, device=device).unsqueeze(0)
    # 先按行裁剪 key 长度，再叠加因果约束
    bool_mask = kv_idx < row_lens.unsqueeze(1)
    if causal:
        bool_mask &= kv_idx <= q_idx.unsqueeze(1)
    return _pack_draft_block_bool_mask(bool_mask, num_packed)
```

vllm/utils/flashinfer.py

平台能力判定与 XQA API 懒加载入口：决定 SM12x 是否可自动选择 XQA decode，并同步 supports_trtllm_attention/use_trtllm_attention 的判定口径。

```
@functools.cache
def supports_trtllm_attention(is_prefill: bool = False) -> bool:
    """判断当前平台是否可用 TRTLLM 注意力（按 prefill/decode 分阶段）。

    SM90 与 SM12x 都只有 XQA decode 内核；SM100/SM103 才同时支持
    TRTLLM prefill 与 decode。其余架构一律不支持。
    """
    # Batch-invariant 模式关闭 TRTLLM 注意力，保证 CUDA 图 shape 固定
    if envs.VLLM_BATCH_INVARIANT:
        return False

    # TRTLLM 内核需要从 NVIDIA artifactory 下载 cubin
    if not has_nvidia_artifactory():
        return False

    # SM90 和 SM12x 只有 XQA decode（SM12x 为本 PR 新增的家族分支）
    if current_platform.is_device_capability(
        90,
    ) or current_platform.is_device_capability_family(120):
        return not is_prefill
```

```
# SM100/SM103 同时具备 TRTLLM prefill 与 decode 内核
return current_platform.is_device_capability_family(100)
```

评论区精华

- amirkl94 在 _compute_decode_query_lens 调用处质疑：函数在 use_dedicated_xqa 为 False 时返回不同的值是否是有意行为，建议要么统一返回 (1, None, None)，要么在函数内加断言；作者回复“Added assertion”。
- pavanimajety 两次提出对 use_dedicated_xqa 的疑问：为何不直接为 SM120 选择 XQA 内核，并提示自己正在重构该文件、担心路径越来越多；作者解释这是临时方案，避免改动 Blackwell 既有执行路径，后续单独 PR 合并两条路径。
- pavanimajety 最终批准时留言：“Merging for now since the CI is clean, but follow on task to ensure XQA is cleanly picked for SM120 with the flashinfer reactor.”
- seanyourhighness 在 issue 侧补充 NVFP4 集成说明：当前版本仅在 kv_cache_dtype.startswith('fp8') 时启用 SM90/SM12x XQA，NVFP4 初始化仍要求 TRTLLM prefill 与 decode 双支持，wrapper 路径仍把 NVFP4 视为 trtllm-gen-only，因此本 PR 不会启用 SM120 NVFP4 KV；FlashInfer 0.6.15 的 XQA 已有 SM120 NVFP4-KV 路径，正文中“XQA 不支持 NVFP4 KV”的说法需要收窄，相关 vLLM 侧门控由 #49818 补齐。
- _compute_decode_query_lens 返回值不一致与断言 (correctness): 作者补充断言，确认该函数只在 use_dedicated_xqa 下被调用。
- 是否维护 use_dedicated_xqa 临时标记 (design): 接受临时方案并合入，批准评论中留下 follow-up 任务，要求后续在 flashinfer reactor 重构中确保 SM120 干净选择 XQA。
- SM12x NVFP4 KV 能力边界与 #49818 集成 (correctness): 本 PR 明确不启用 SM120 NVFP4 KV，vLLM 侧门控由 #49818 补齐；评论作为集成注意事项记录，未在本 PR 内解决。
- forward 中 XQA 断言收敛位置 (style): 按作者方案保留集中断言。

风险与影响

- 风险：
  - 核心路径回归风险：vllm/v1/attention/backends/flashinfer.py 的 build()/forward() 是 V1 推理热路径，use_dedicated_xqa 分支与既有 SM90/SM100 分支并存，元数据或 mask 构造错误会直接导致 decode 数值错误。
  - CUDA 图 padding 待复测：作者自述 concurrency-1 复现了图 padding 错位，修复（零长度请求 + 精确 uniform 乘积校验）后尚未在 GPU 上复跑，存在未验证回归窗口。
  - 依赖版本多端联动：需要 FlashInfer 0.6.16.post1 及以上（提交中还有 post2 更新），Python/cubin/JIT 缓存 /Docker 多处 pin 必须保持一致；pavanimajety 也因此要求跑全量 CI。
  - 与 #49818 文件重叠：两个 PR 都改动 vllm/utils/flashinfer.py 与 vllm/v1/attention/backends/flashinfer.py，合入顺序影响 SM12x NVFP4 门控；正文中“XQA 不支持 NVFP4”的断言需随 #49818 收窄，避免误导后续开发者。

    - 设计债与维护成本：use_dedicated_xqa 是临时双路径标记，维护者正在重构该文件，短期增加理解与合并成本。
- 影响：
    - 用户侧：RTX 50 系（SM120/121）用户在 decode 阶段可自动选中 XQA 内核，获得统一的投机解码、滑动窗口、sink 与 uniform-batch CUDA 图支持；SM90/SM100 行为不变，NVFP4 KV 暂不启用。
    - 系统侧：平台能力判定（supports_trtllm_attention/use_trtllm_attention）对 SM12x 的口径与 SM90 对齐，影响后续所有后端的自动选择逻辑；decode 元数据字段的扩展为向后兼容的增量。
    - 团队侧：引入临时路由标记，需与 flashinfer 后端重构及 #49818 协调；FlashInfer 依赖升级会传导到 CI、Docker 与 wheel 构建流程。
    - 风险标记：核心路径变更，CUDA 图 padding 待 GPU 复测，依赖版本多端联动，与 #49818 文件重叠，临时双路径设计债

关联脉络

- PR #49818 标题未在材料中提供（PR 讨论中引用为 SM12x NVFP4 enablement）：与 #49718 同改 vllm/utils/flashinfer.py 与 vllm/v1/attention/backends/flashinfer.py，承担 SM12x NVFP4 KV 门控；seanyourhighness 指出两者需明确合入顺序。
- PR #47306 标题未在材料中提供（PR body 引用为早期 XQA 草稿）：PR body 说明该草稿面向旧共享 decode API 与更早 FlashInfer 版本，与本 PR 功能线相同但不重复。
- PR #37592 标题未在材料中提供（PR body 引用为早期 XQA 草稿）：PR body 中与 #47306 并列列为早期草稿，本 PR 限定 SM12x 专用 XQA API，避免与其混淆。
- PR #51865 [Bugfix][MRV2] Require all requests to be decoding for uniform-decode dispatch: 同属投机解码 + CUDA 图统一批处理语义；本 PR 的 q_len_per_req/q_cu_seq_lens 与 uniform decode 判定相互影响，历史修复确保 uniform-decode 图谱不混入 prefill。