PR #52539 完整报告

vllm-project/vllm

[Kernel][Perf] Support Qwen head ratios in fused GDN MTP

合并时间：2026-08-18 09:17

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/52539

执行摘要

- 一句话：fused GDN MTP 内核支持 Qwen 头比例 1/2/3/4/8，提升 MTP 解码吞吐
- 推荐动作：值得精读。核心看三点：一是如何把运行期整数 ratio 提升为编译期模板参数并保留 10 个特化；二是 host 端用 C++20 templated lambda 消除 dispatch 重复；三是用 config audit 而非拍脑袋决定支持集合，并把“不支持比例回退”显式纳入 guard 测试。对后续内核工作，建议把质量等价性评估（如 GSM8K 等）与性能评估一起纳入 CI 门禁，避免 -1.06 pp 类点估计长期悬而未决。

功能与动机

审计公开 Qwen GDN 配置发现生产 ratio 覆盖 1/2/3/4/8（从 Qwen3.5 0.8B/2B 到 Qwen3.6 27B/35B、Qwen3.8 2.4T），而 #51674 引入的 fused GDN MTP 只支持 HV/H=8，导致大多数真实模型无法走融合路径。PR body 明确说明 "makes the value-head ratio a compile-time kernel parameter and dispatches exact ratios 1/2/3/4/8"，目标是把融合内核的吞吐收益扩展到全部生产模型。reviewer gau-nernst 也主动提出 "Btw, since you are already doing this, any other head ratio we should cover? Can you help check other past Qwen GDN models?"，进一步推动了配置审计。

实现拆解

变更入口

- csrc/libtorch_stable/gdn/fused_gdn_decode_kernel.cu：内核模板从 template <typename StateT> 扩展为 template <typename StateT, int ValueHeadsPerKeyHead>；key_head 推导由字面量 8 改为编译期模板参数。
- vllm/model_executor/layers/mamba/gdn/qwen_gdn_linear_attn.py：_can_use_fused_gdn_mtp_decode 的 guard 从固定 HV/H == 8 放宽。
- tests/kernels/test_fused_gdn_post_conv.py 与 tests/kernels/mamba/test_gdn_fused_mtp.py：测试配套。

实现步骤

1. 内核编译期参数化：在 fused_gdn_decode_kernel.cu 中，gdn_decode_post_conv_mtp_kernel 与 launch_gdn_decode_post_conv_mtp 都增加 int ValueHeadsPerKeyHead 模板参数，内核内 const int key_head = value_head / ValueHeadsPerKeyHead; 使每个 value head 到 key head 的映射、共享内存分块与循环

边界都按精确 ratio 特化展开。对 ratio=8 的既有行为完全兼容。

2. host 校验与分派重构：入口 fused_gdn_decode_post_conv_mtp 把原先 num_value_heads == 8 * num_key_heads 的硬校验改为"整除 + 商在 {1,2,3,4,8} 集合内"；随后按 review 建议用 C++20 templated lambda 组合 StateT 与 ValueHeadsPerKeyHead，配合 switch 将运行期 ratio 映射到编译期常量，最终产出 10 个 ratio × state dtype 特化，消除了最初的 if-else 重复代码。
3. Python guard 联动：QwenGatedDeltaNetAttention._can_use_fused_gdn_mtp_decode 从 self.num_v_heads == 8 * self.num_k_heads 改为先 self.num_v_heads % self.num_k_heads == 0 再判定商属于 (1, 2, 3, 4, 8)，避免非整比被 floor 除法误收，保证 CUDA 端与 Python 端的支持集合一致。
4. 测试覆盖重构：test_fused_gdn_post_conv.py 保留全部 6 个原始 ratio-8 用例（TP16/TP4、ragged、max、BF16/FP32 组合），参数化加入 head_ratio 维度，并新增 ratio 1/2/3/4 的聚焦用例（其中 ratio-2 覆盖两种 state dtype）；test_gdn_fused_mtp.py 新增 test_fused_mtp_head_ratio_guard，显式验证 ratio 1/2/3/4/8 通过、非整比与不支持比例回退。
5. 验证与性能数据：post-conv 文件 75 passed，model-path 文件 12 passed；固定输入 ratio-2 配对测试 60/60 对全部提升（B1 eager 约 2.47x 到 B64 eager 约 1.06x）；真实模型 Qwen3.6-35B native MTP 下 30 层 GDN 全部走 fused 路径，eager B1/B16 吞吐提升 1.117x。GSM8K 5-shot greedy 上 fused 1018/1319 与 fallback 1032/1319，点估计 -1.06 pp，McNemar p=0.335，未建立等价性。

关键文件：

- csrc/libtorch_stable/gdn/fused_gdn_decode_kernel.cu（模块 融合内核；类别 other；类型 core-logic；符号 gdn_decode_post_conv_mtp_kernel, launch_gdn_decode_post_conv_mtp, fused_gdn_decode_post_conv_mtp）：功能核心：内核模板增加 ValueHeadsPerKeyHead 编译期参数，host 入口从 HV/H=8 硬校验放宽为 {1,2,3,4,8} 集合，并用 C++20 templated lambda + switch 完成 10 个特化分派。
- vllm/model_executor/layers/mamba/gdn/qwen_gdn_linear_attn.py（模块 模型层；类别 source；类型 data-contract；符号 _can_use_fused_gdn_mtp_decode）：Python 侧融合路径准入 guard 的单一改动点，决定哪些 Qwen GDN 模型能启用 fused MTP。
- tests/kernels/test_fused_gdn_post_conv.py（模块 内核测试；类别 test；类型 test-coverage；符号 test_fused_gdn_decode_post_conv_mtp_ratio8, test_fused_gdn_decode_post_conv_mtp_head_ratios）：内核直接测试：保留 6 个原始 ratio-8 用例并新增生产 ratio 聚焦用例，是验证 10 个特化正确性的主要载体。
- tests/kernels/mamba/test_gdn_fused_mtp.py（模块 模型测试；类别 test；类型 test-coverage；符号 test_fused_mtp_head_ratio_guard）：模型路径测试：新增 guard 边界测试，显式覆盖 ratio 1/2/3/4/8 通过与非整比 / 不支持比例回退，防止 guard 与 CUDA 端集合漂移。

关键符号：gdn_decode_post_conv_mtp_kernel, launch_gdn_decode_post_conv_mtp, fused_gdn_decode_post_conv_mtp, QwenGatedDeltaNetAttention._can_use_fused_gdn_mtp_decode, test_fused_gdn_decode_post_conv_mtp_head_ratios, test_fused_mtp_head_ratio_guard

关键源码片段

csrc/libtorch_stable/gdn/fused_gdn_decode_kernel.cu

功能核心：内核模板增加 ValueHeadsPerKeyHead 编译期参数，host 入口从 HV/H=8 硬校验放宽为 {1,2,3,4,8} 集合，并用 C++20 templated lambda + switch 完成 10 个特化分派。

```
// host 入口：先做整除校验，再按编译期常量分派。
const int num_key_heads = static_cast<int>(key_width / (2 * kDimK));
const int value_heads_per_key_head = num_value_heads / num_key_heads;
STD_TORCH_CHECK(
    num_value_heads % num_key_heads == 0 &&
    ((value_heads_per_key_head >= 1 && value_heads_per_key_head <= 4) ||
     value_heads_per_key_head == 8),
    "GDN decode MTP fusion requires HV/H in {1, 2, 3, 4, 8}");

// C++20 templated lambda：把 StateT 与 ratio 组合提升为编译期参数。
const auto launch = [&]<typename StateT, int ValueHeadsPerKeyHead>() {
    launch_gdn_decode_post_conv_mtp<StateT, ValueHeadsPerKeyHead>(
        mixed_qkv, a_log, dt_bias, state_indices, cu_seqlens,
        num_accepted_tokens, state, norm_weight, out, a_ptr, b_ptr,
        output_gate_ptr, num_key_heads, num_value_heads, scale, norm_eps,
        strides);
};
// 先按 state dtype 分派，再用 switch 固定 ratio，共 10 个特化。
const auto dispatch_state_type = [&]<int ValueHeadsPerKeyHead>() {
    if (state_scalar_type == ScalarType::Float) {
        launch.template operator()<float, ValueHeadsPerKeyHead>();
    } else {
        launch.template operator()<__nv_bfloat16, ValueHeadsPerKeyHead>();
    }
};
switch (value_heads_per_key_head) {
    case 1: dispatch_state_type.template operator()<1>(); break;
    case 2: dispatch_state_type.template operator()<2>(); break;
    case 3: dispatch_state_type.template operator()<3>(); break;
    case 4: dispatch_state_type.template operator()<4>(); break;
    default: dispatch_state_type.template operator()<8>(); break; // ratio 8 兼容路径
}

// ratio 变为编译期模板参数后，key_head 映射可被编译器精确优化。
template <typename StateT, int ValueHeadsPerKeyHead>
__global__ __launch_bounds__(kThreads, 2) void gdn_decode_post_conv_mtp_kernel(
    // ... 参数列表省略 ...
) {
    // 每个 value head 映射到对应的 key head；ratio=8 时与原实现完全一致。
    const int key_head = value_head / ValueHeadsPerKeyHead;
    // 共享内存与循环展开均按编译期 ratio 特化，避免运行期除法与分支开销。
    // ... 主体计算省略 ...
}
```

vllm/model_executor/layers/mamba/gdn/qwen_gdn_linear_attn.py

Python 侧融合路径准入 guard 的单一改动点，决定哪些 Qwen GDN 模型能启用 fused MTP。

```
def _can_use_fused_gdn_mtp_decode(
    self, attn_metadata: GDNAttentionMetadata
) -> bool:
    state_indices = attn_metadata.spec_state_indices_tensor
    return (
        attn_metadata.spec_sequence_masks is not None
        and attn_metadata.num_decodes == 0
        and attn_metadata.num_spec_decodes > 0
        and self.kv_cache[1].dtype in FUSED_GDN_STATE_DTYPES
        and self.gdn_decode_kernel == "cuda"
        # 先做整除校验，避免非整比被下面的 floor 除法误收。
        and self.num_v_heads % self.num_k_heads == 0
        # 支持集合来自公开 Qwen GDN checkpoint 审计，包含 1/2/3/4/8。
        and self.num_v_heads // self.num_k_heads in (1, 2, 3, 4, 8)
        and state_indices is not None
        and state_indices.size(1) <= MAX_FUSED_GDN_MTP_TOKENS
        and hasattr(torch.ops._C, "fused_gdn_decode_post_conv_mtp")
    )
```

评论区精华

核心讨论集中在分派代码结构、测试保留策略与支持集合的确定上：

- 分派去重（design）：gau-nernst 建议 "you can either use a macro or lambda template (requires C++20, which is set by vLLM) for dispatch to reduce repeated code"。作者采用 C++20 templated lambda，最终以 switch + 两层 lambda 完成 10 个特化分派。
- 测试保留策略（testing）：gau-nernst 要求 "keep the original test cases, and add maybe only 1 or 2 new cases for head_ratio=2"。作者恢复全部 6 个原始 ratio-8 用例，并为每个生产 ratio 各加一个聚焦用例。
- guard 写法（style）：gau-nernst 提出 nit "self.num_v_heads // self.num_k_heads in (2, 8)"；作者改为模整除 + 商集合形式，并说明 "a non-integral ratio cannot be accepted by floor division"，支持集合最终为 1/2/3/4/8。
- 生产比例审计（question）：gau-nernst 追问是否还有其他生产 head ratio；作者盘点出 HV/H=1/2/3/4 的多个 checkpoint，gau-nernst 补充 "Qwen/Qwen3.8-2.4T-A95B uses Hv/H=8"，作者承认漏掉了 Qwen3.8 并修正总结。
- 分派代码去重：宏 vs C++20 templated lambda (design): 作者采用 C++20 templated lambda，最终以 switch + 两层 lambda 完成 10 个特化分派，消除了重复代码。
- 测试用例保留与聚焦策略 (testing): 作者恢复全部 6 个原始 ratio-8 用例及其 ID，config audit 后为每个生产 ratio 各加一个聚焦用例，另加一个 ratio-2 用例覆盖第二种 state dtype，post-conv 文件 75 passed。

- Python guard 的 ratio 判定写法 (style): 作者改为模整除 + 商集合形式，支持集合定为 1/2/3/4/8，避免非整比被 floor 除法误接受。
- model-path 测试改动是否必要 (question): 最终保留新增的 test_fused_mtp_head_ratio_guard，因为它提供了 Python guard 判定边界的独立覆盖。
- 生产 head ratio 审计与支持集合确定 (question): 最终支持集合确定为 {1,2,3,4,8}，ratio-8 既是兼容路径也是生产路径；作者修正总结并纳入最终矩阵。

风险与影响

- 风险：
  1. 数值质量风险：GSM8K 5-shot greedy 上 fused 路径 1018/1319（77.18%）低于 fallback 1032/1319（78.24%），点估计 -1.06 pp，PR body 明确声明 "This does not establish equivalence or non-inferiority"。融合路径被无条件启用后，该点估计需要后续积累更多质量数据确认。
  2. 索引与共享内存布局风险：key_head = value_head / ValueHeadsPerKeyHead 影响 shared memory 分块与循环边界，10 个特化虽由编译期保证整除，但不同 ratio 下的 bank conflict 与吞吐模式未被逐一基准验证。
  3. 平台验证不完整：PR body 说明 H20 只验证了早期 ratio-2-only host dispatch，最终 multi-ratio dispatch 仅在 PRO 6000 上验证；H20 真实模型与质量评估未运行，跨 GPU 性能不可直接对比。
  4. 编译产物扩容：内核特化从 2 个增长到 10 个，虽无 ABI/ 构建系统变更，但会延长 CUDA 编译时间并增大二进制体积。
  5. 双端校验一致性：CUDA 端 STD_TORCH_CHECK 与 Python guard 都按 {1,2,3,4,8} 校验，顺序上先做模整除再做商判定，能拒绝非整比，误用风险较低。 - 影响：用户影响：Qwen3.5/3.6/3.8 系列 GDN 模型启用 native MTP 时，解码路径从 fallback 切到 fused，真实模型吞吐提升约 1.1x（eager 与 FULL_AND_PIECEWISE 模式均受益），固定输入最多提升约 2.5x。

系统影响：CUDA 编译特化翻倍至 10 个；融合路径覆盖面扩大，意味着内核数值缺陷的影响面也随之扩大，因此 GSM8K 的 -1.06 pp 点估计需要持续关注。

团队影响：形成了“配置审计 → 编译期特化 → 聚焦测试”的内核扩展范式，对后续其他 head ratio 或架构（如 MoE MTP）有模板价值；review 中确立的“保留原用例 + 聚焦新增”测试策略值得沿用。

- 风险标记：数值质量未建立等价性，H20 平台验证不完整，内核特化扩容编译产物

关联脉络

- PR #51674 Fused GDN MTP verification for HV/H=8（PR body 引用的基础实现）：PR body 明确引用 #51674 是仅支持 HV/H=8 的 fused GDN MTP verification 基础实现，本 PR 是对它的比例扩展与内核泛化。
- PR #52197 Support DSpark configs with architectures=DSparkDraftModel + model_type=qwen3: 同属 Qwen 推测解码 / 草稿配置方向，且都涉及 Qwen 模型在

speculative/MTP 路径下的能力扩展，可视为同一功能线的演进。