

PR #53785 完整报告

vllm-project/vllm

[Attention] Enable dense and masked MHA for GLM-5

合并时间: 2026-08-28 05:17

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/53785>

执行摘要

- 一句话: 为 GLM-5 启用 masked MHA prefill, 路由表重构为维度驱动
- 推荐动作: 值得精读。核心看点: 一是 `_use_sparse_mha` 抽取与维度感知策略表的数据契约设计; 二是“策略表只管性能、workspace 管可行性”的双层守卫哲学; 三是对 PR #52512 事故的回归修复方式; 四是 benchmark 驱动的路由阈值方法论与 32K 以上记录策略。适合负责注意力后端、性能路由或 MLA 相关模块的工程师细读。

功能与动机

PR body 明确目标为“enable the sparse-MLA masked-MHA prefill path for the GLM-5 256/256 QK/V head dimensions”, 此前 GLM-5 只能走 sparse MQA, 纯 prefill 延迟明显偏高; 实测 FlashMLA 纯 prefill 在 TP1/2/4/8 的 MHA/MQA 交叉点约为 8.5K/20.8K/52.8K/116.8K tokens, FlashInfer TP4 在 4K 时 MQA 延迟是 masked MHA 的 4.70 倍, 收益巨大。讨论中 simon-veitner-redhat 还指出 PR #52512 曾因 short-prefill 元数据选 dense MHA、indexer 跳过 top-k、sparse MQA 消费 -1 填充缓冲区导致重复 token 而禁用 dense MHA, 本 PR 恢复该路径前必须先保证模型层真正执行 MHA 分支。

实现拆解

1. 路由决策重构 (`vllm/model_executor/layers/attention/mla_attention.py`): 把 `forward_impl` 内联的 dense/masked MHA 判定抽为私有方法 `_use_sparse_mha()`, 并将常量表 `_DSV32_MASKED_MHA_THRESHOLDS` 泛化为 `_MASKED_MHA_THRESHOLDS`, 键为 (`qk_head_dim`, `v_head_dim`), 值为 `backend` → `TP` → (纯 prefill 区间表, 缓存上下文区间表); `_use_masked_mha()` 随之新增维度与 `has_context` 参数。DeepSeek 192/128 旧阈值原样保留, GLM-5 256/256 新增条目, 这让路由策略与模型维度解耦, 后续新维度只需加表项。
2. 模型侧接入 (`vllm/models/deepseek_v32/attention.py`): 移除 `supports_dense_mha_prefill = False` (恢复为 `True`), 并在 `_sparse_indexer_and_attn` 的索引打分之前插入 MHA 分支: FP8 query 时对 `q_pe` 施加 `rotary_emb`, 否则复用解码分支已算好的 `mqa_q`, 拼接 `q_nope` 后调用 `forward_impl`。这不仅打开 GLM-5 的 masked MHA, 也修复 PR #52512 遗留的重复 token 事故。
3. workspace 第二道守卫 (`vllm/model_executor/layers/attention/sparse_mla_attention.py`): 保留固定 128 MiB 的 `masked_mha_workspace_fits()` 校验; 策略表允许记录 32K 以上的获胜区间, 但实际执行仍以 workspace 容量为准, 超限请求自动回退 sparse MQA

，策略表与守卫职责分离。

4. 基准工具配套 (benchmarks/attention_benchmarks/) : benchmark.py 新增 model_parameter_sweep 支持，一次扫描 num_q_heads=64/32/16/8 (对应 TP1/2/4/8) ; mla_runner.py 新增 FLASHINFER_MLA_SPARSE 的 query_format 配置与强制 masked MHA 模式 (自动按 batch 扩容 topk mask workspace) ; 另新增 GLM-5 基准配置 mla_sparse_masked_mha_vs_mqa_glm5.yaml。
5. 依赖与测试: cmake/external_projects/vllm_flash_attn.cmake 临时 pin 到 MatthewBonanni/flash-attention@8f844a0 (含 head-dim-256 mask_mod 支持)，上游合并后需换成 vLLM 官方仓库；测试侧新增 4 组 GLM-5 路由参数化测试、workspace 守卫测试与 DeepSeek V3.2 dispatch 测试，覆盖纯 prefill、缓存上下文、FlashInfer 与维度隔离场景。

关键文件：

- vllm/model_executor/layers/attention/mla_attention.py (模块 注意力层；类别 source；类型 data-contract；符号 _use_sparse_mha, _use_masked_mha, _MASKED_MHA_THRESHOLDS) : 核心路由逻辑所在：新增 _use_sparse_mha 与维度感知阈值表 _MASKED_MHA_THRESHOLDS，决定 GLM-5/DeepSeek 的 prefill 走 masked MHA 还是 sparse MQA。
- vllm/models/deepseek_v32/attention.py (模块 模型接入；类别 source；类型 data-contract；符号 _sparse_indexer_and_attn, supports_dense_mha_prefill) : GLM-5 复用 DeepSeek V3.2 注意力实现；恢复 supports_dense_mha_prefill，并在 _sparse_indexer_and_attn 中真正执行 MHA 分支，修复 PR #52512 遗留事故。
- vllm/model_executor/layers/attention/sparse_mla_attention.py (模块 注意力层；类别 source；类型 data-contract；符号 masked_mha_workspace_fits, GLOBAL_TOPK_MASK_MAX_BYTES) : masked_mha_workspace_fits 与 GLOBAL_TOPK_MASK_MAX_BYTES 所在文件，是策略表之外防止 workspace 超限的关键第二道守卫。
- tests/v1/attention/test_mla_backends.py (模块 路由测试；类别 test；类型 test-coverage；符号 test_glm5_masked_mha_pure_prefill_routing, test_glm5_masked_mha_context_routing, test_glm5_flashinfer_masked_mha_routing, test_masked_mha_routing_is_dimension_specific) : 新增 4 组 GLM-5 路由参数化测试与维度隔离测试，固化 FlashMLA/FlashInfer 的纯 prefill 与缓存上下文阈值。
- tests/model_executor/layers/test_mla_short_prefill_indexer.py (模块 索引器测试；类别 test；类型 test-coverage；符号 test_deepseek_v32_dispatches_selected_mha, record_forward_impl) : 覆盖 DeepSeek V3.2 在 selected dense MHA 下正确 dispatch 到 forward_impl 的回归测试，防止重复 token 事故复现。
- benchmarks/attention_benchmarks/benchmark.py (模块 基准工具；类别 source；类型 core-logic；符号 main) : 支持 model_parameter_sweep，一次扫描 TP1/2/4/8 对应的 local q heads，是全部路由阈值的数据来源。
- benchmarks/attention_benchmarks/mla_runner.py (模块 基准工具；类别 source；类型 dependency-wiring；符号 get_prefill_backend_config, _run_mla_benchmark_batched) : 新增 FLASHINFER_MLA_SPARSE 的 query_format 配置与强制 masked MHA 模式，自

动按 batch 扩容 topk mask workspace，支撑 FlashInfer 路由数据采集。

- cmake/external_projects/vllm_flash_attn.cmake（模块 构建脚本；类别 other；类型 core-logic）：临时把 FlashAttention pin 到含 head-dim-256 mask_mod 支持的个人 fork 提交，是功能前置依赖，上游合并后需回切官方仓库。

关键符号：_use_sparse_mha, _use_masked_mha, _sparse_indexer_and_attn, masked_mha_workspace_fits, forward_impl, run_mla_benchmark

关键源码片段

vllm/model_executor/layers/attention/mla_attention.py

核心路由逻辑所在：新增 _use_sparse_mha 与维度感知阈值表 _MASKED_MHA_THRESHOLDS，决定 GLM-5/DeepSeek 的 prefill 走 masked MHA 还是 sparse MQA。

以下是 `MLAAttention._use_sparse_mha` 的完整实现，体现路由策略与 workspace 双层守卫的设计：

```
def _use_sparse_mha(self, attn_metadata: "MLACommonMetadata") -> bool:
    """决定 sparse MLA 的 prefill 阶段是否走 masked/dense MHA 而不是 sparse MQA。
```

```
    该函数把原内联在 forward_impl 中的路由逻辑抽出，并以
    (qk_head_dim, v_head_dim) 维度去查 _MASKED_MHA_THRESHOLDS 策略表，
    使 DeepSeek 192/128 与 GLM-5 256/256 等不同维度可以各自路由。
```

```
    """
```

```
    prefill = attn_metadata.prefill
```

```
    if prefill is None:
```

```
        # 没有 prefill 元数据（例如纯 decode 阶段）时直接回落 MQA
        return False
```

```
    use_masked_mha = (
```

```
        # 后端必须显式支持 masked MHA，且 DCP 场景暂不支持该 mask 语义
```

```
        self.prefill_backend is not None
```

```
        and self.impl.masked_mha_available
```

```
        and self.impl.dcp_world_size <= 1
```

```
        and _use_masked_mha(
```

```
            backend_name=self.attn_backend.get_name(),
```

```
            tensor_parallel_size=(
```

```
                self._vllm_config.parallel_config.tensor_parallel_size
```

```
            ),
```

```
            qk_head_dim=self.qk_nope_head_dim + self.qk_rope_head_dim,
```

```
            v_head_dim=self.v_head_dim,
```

```
            query_len=prefill.max_query_len,
```

```
            seq_len=attn_metadata.prefill_max_seq_len,
```

```
            has_context=prefill.chunked_context is not None,
```

```
        )
```

```
        # 第二道守卫：固定 128 MiB workspace 装不下时，策略命中也要回退 MQA
```

```
        and self.impl.masked_mha_workspace_fits(prefill)
```

```
    )
```

```

# dense MHA 由 short-prefill indexer 标记，masked MHA 由策略表决定；
# 两者任一命中即走 MHA，除非用户显式强制 MQA。
return (prefill.use_dense_mha or use_masked_mha) and not (
    self._vllm_config.attention_config.sparse_mla_force_mqa
)

```

vllm/models/deepseek_v32/attention.py

GLM-5 复用 DeepSeek V3.2 注意力实现；恢复 supports_dense_mha_prefill，并在 _sparse_indexer_and_attn 中真正执行 MHA 分支，修复 PR #52512 遗留事故。

以下是 DeepseekV32Attention._sparse_indexer_and_attn 中新增的 MHA 执行分支，位于索引器打分逻辑之前：

```

# 位于 DeepseekV32Attention._sparse_indexer_and_attn 中、索引器打分逻辑之前。
# 短 prefill 若被路由选中走 dense/masked MHA，必须先在这里真正执行 MHA，
# 否则 indexer 会跳过 top-k 打分，后续 sparse MQA 将消费 -1 填充的缓冲区
# 并产生重复 token (PR #52512 曾因此禁用 dense MHA) 。
if self._use_sparse_mha(attn_metadata):
    assert kv_c is not None and k_pe is not None
    # FP8 query 场景 q_pe 还需叠加旋转编码；非 FP8 场景直接复用解码分支的 mqa_q
    mha_q_pe = self.rotary_emb(positions, q_pe)[0] if self._fp8_query else mqa_q
    # masked MHA 需要把 nope 与 rope 两部分拼回 256 维的 QK head
    mha_q = torch.cat((q_nope, mha_q_pe), dim=-1)
    # 与 decode 共用 forward_impl，保证 CUDA graph 捕获与重放行为一致
    self.forward_impl(
        mha_q,
        kv_c,
        k_pe.unsqueeze(1),
        kv_cache,
        attn_metadata,
        output,
    )
return

```

评论区精华

核心交锋集中在 simon-veitner-redhat 与 MatthewBonanni 对选择逻辑的讨论：

- workspace 与策略表一致性：simon-veitner-redhat 指出生产环境固定分配 128 MiB workspace，实际路由要求“策略命中且 workspace 够用”双条件成立，而策略表记录了超过 32K 的交叉点，二者存在表竞合。MatthewBonanni 回应称策略表是有意记录 32K 以上获胜区间，以便 workspace 上限调整后策略仍可用，masked_mha_workspace_fits() 是独立最终守卫，并新增 test_masked_mha_workspace_guards_long_routing_policy 验证长策略条目会被守卫拒绝，reviewer 最终 LGTM。
- FlashInfer 路由前提：simon-veitner-redhat 指出 GLM-5 复用 DeepSeek V3.2 的 DSA 实现，SM10 上 local heads ≤ 16 （即 TP4/TP8）才会优先 FlashInfer，所以只有 TP4/TP8 有 FlashInfer 基准条目，与实测数据一致。

- 历史事故回归：simon-veitner-redhat 提醒 PR #52512 曾因 short-prefill 元数据选 dense MHA 但模型层未执行、indexer 跳过 top-k、sparse MQA 消费 -1 缓冲区导致重复 token；本 PR 恢复 dense MHA 时必须真正执行 MHA 分支，最终通过 `_sparse_indexer_and_attn` 显式分支与 `test_deepseek_v32_dispatches_selected_mha` 覆盖。
- 次要意见：TP8 独立配置并入主配置、plot 脚本是否保留及 B200 特定性，均按建议处理。
 - 128 MiB workspace 与策略表 32K 以上条目的一致性 (design): MatthewBonanni 回应：策略表有意记录 32K 以上获胜区间以便 workspace 上限调整后仍可用，workspace 检查是独立最终守卫，并新增 `test_masked_mha_workspace_guards_long_routing_policy` 验证长策略条目会被守卫拒绝。
 - FlashInfer 路由依赖 local head 数与 GLM-5 复用 DeepSeek V3.2 (question): 接受该前提：TP4/TP8 的 FlashInfer 阈值按实测加入，TP1/TP2 不配置 FlashInfer 条目。
 - 恢复 dense MHA 必须避免 PR #52512 重复 token 事故回归 (correctness): 本 PR 在 `_sparse_indexer_and_attn` 中显式增加 MHA 分支并让 `forward_impl` 真正执行，新增 `test_deepseek_v32_dispatches_selected_mha` 覆盖；reviewer 最终 LGTM。
 - 测试充分性：路由测试未覆盖 workspace 场景 (testing): MatthewBonanni 补充 `test_masked_mha_workspace_guards_long_routing_policy`，验证 `masked_mha_workspace_fits` 会拒绝超长策略条目。
 - 基准配置与脚本整理 (style): 已合并配置；plot 脚本与 B200 数据按需保留。

风险与影响

- 风险：
 - 策略表与固定 workspace 表竞合：策略表记录 32K 以上获胜区间，但 128 MiB workspace 固定上限意味着纯 prefill 超过 32K 的条目实际不会命中；当前由 `masked_mha_workspace_fits` 兜底，未来若调整 workspace 大小需同步审计表语义（`mla_attention.py` 与 `sparse_mla_attention.py`）。
 - 临时 FlashAttention fork pin: cmake 依赖 MatthewBonanni/flash-attention@8f844a0，若上游 vLLM 官方 FlashAttention 仓库合并变基，pin 可能失效，合并前必须回切，否则影响所有构建。
 - dense MHA 恢复对 DeepSeek V3.2 全系生效：DeepseekV32Attention 被 GLM-5 复用，`_use_sparse_mha` 或 `_sparse_indexer_and_attn` 新分支的任何错误都会同时影响两类模型，属于核心 prefill 路径变更。
 - CUDA graph 捕获 / 重放一致性：提交历史中有专门修复“Fix GLM dense MHA routing under CUDA graphs”，表明 graph 捕获期与运行期 metadata 变化可能导致 kernel 分支不一致。
 - 测试覆盖局限：新增测试多为 CPU 构造型单元测试，真实 GPU 端到端覆盖依赖 Buildkite CI 与 B200 benchmark，路由回归主要靠参数化单元测试。
- 影响：
 - GLM-5/GLM-5.2 用户：sparse prefill 显著加速，FlashMLA 在 TP1 8K 以内、TP2 20K 以内、TP4 48K 以内、TP8 112K 以内纯 prefill 走 masked MHA；FlashInfer

TP4 36K、TP8 64K 以内同样受益；缓存上下文场景按赢区路由，超过阈值自动回退 MQA，行为保守。

- DeepSeek V3.2 用户：dense MHA 重新可用，短 prefill 获得加速，同时修复历史重复 token 事故。
- 团队与工程：路由表从单一 DeepSeek 常量表演进为按维度、后端、TP 组织的策略表，新模型接入成本降低；基准工具补齐 local head sweep 能力，可复用于其他 MLA 模型；CI 增加 FlashAttention fork 构建，需跟进回切。
- 影响范围：注意力 prefill 核心路径，但仅对 sparse MLA（DeepSeek/GLM 系）生效，其他模型无行为变化。
- 风险标记：核心注意力路径变更，临时依赖 flash-attention fork pin，策略表与固定 workspace 表竞合，CUDA graph 捕获 / 重放路由一致性，历史事故回归（PR #52512 类），真实 GPU 端到端测试依赖 CI

关联脉络

- PR #53878 [Perf][GLM5.2] Fuse sparse MLA Q concatenation with head padding: 同为 GLM sparse MLA prefill 性能优化，改动 sparse_mla_attention 相关路径，与本 PR 构成 GLM-5 注意力加速的功能线。
- PR #54012 [Attention][DCP] Use FlashInfer native CP for MLA decode: FlashInfer MLA 后端能力演进，本 PR 为 FLASHINFERENCE_MLA_SPARSE 增加 masked MHA 路由与基准支持，二者共享 FlashInfer MLA 后端上下文。
- PR #53183 [Model Runner V2] Use MRV2 for all models by default: MRV2 已成为默认模型运行路径，本 PR 的 forward_impl/_use_sparse_mha 正是 MRV2 下 DeepSeek/GLM 系列 prefill 的主路径。