PR #52046 完整报告

vllm-project/vllm

[nv] add pcp support in dsv3.2

合并时间：2026-08-19 05:59

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/52046

执行摘要

- 一句话：DSV3.2 接入 PCP 分片，fused kernel 物化 K 行，prefill 提升 2.65x
- 推荐动作：### 阅读建议

值得精读。重点看两处设计决策：一是 _fused_norm_rope_kernel 如何用输出指针是否为 None 统一 " 物化输出 " 与 " 直插 cache" 两条路径，避免为 PCP 单独维护一份 kernel；二是 _sparse_indexer_and_attn 在 PCP 与 DCP 组合下如何完成 K 行汇聚、query all-gather 与 logsum 合并。也建议关注后续是否有 dense MHA 路径 PCP 支持的跟进 PR，以及 PCP+DCP 组合下的数值一致性测试。

功能与动机

PR body 明确提出 "seq shard is more efficient than head shard for sparse mla and indexer"，即对稀疏 MLA 与稀疏索引器而言，按 seqlen 切分的 PCP 比按 head 切分更高效。性能目标是长 prefill 优化：在 GLM5.2 nvfp4 on B300、32k max batched tokens、16k long prefill threshold 条件下，PCP8 v.s. TP8 full 32k forward p50 = 366.9 ms v.s. 973.9 ms（2.65x）。由于 PCP 分片 seqlen 但 KV cache 像 TP 一样跨 rank 复制，forward 中新生成的 cache 行必须先 gather 再插入，因此需要把 fused_norm_rope 从 " 直接 cache 写入 " 改为 " 物化输出 + 汇聚后写入 "。

实现拆解

实现拆解

1.改造fused_norm_rope的kernel契约（vllm/models/deepseek_v32/common/kernels.py） - _fused_norm_rope_kernel 新增 kv_out_ptr/kv_out_stride、kpe_out_ptr/kpe_out_stride、index_k_out_ptr/index_k_out_stride 三组输出指针；fused_norm_rope 同步新增 kv_c_out、k_pe_out、index_k_out 参数。 - pid==1（KV norm + RoPE）分支改为双模式：输出指针不为 None 时先把 kv_c、k_pe 物化到 buffer；slot_mapping_ptr 存在时才执行原 cache 直插逻辑。pid==0（indexer K）分支类似，indexer_cache_ptr 与 slot_mapping_ptr 同时存在才做 FP8 quant + cache write。 - slot_mapping_ptr is None 的 memory profiling 路径改为：仅当三个输出指针全为 None 时提前 return，保证 PCP 场景下物化仍可执行。

2. 在 DeepseekV32Attention.forward 中按 use_pcp 分流（vllm/models/deepseek_v32/attention.py） - PCP 时 indexer_k_cache 置 None、分配

index_k_out；mla_kv_cache、mla_slot 置 None，分配 kv_c_out、k_pe_out，全部传给 fused_norm_rope。 - _sparse_indexer_and_attn 改为通过函数式 API 调用 sparse_attn_indexer，PCP 时 skip_k_cache_insert=not use_pcp、use_pcp=True 并传入物化后的 index_k，同时透传 dcp_rank、dcp_world_size、cp_kv_cache_interleave_size。

3. 新增 PCP/DCP 汇聚流程（vllm/models/deepseek_v32/attention.py） - 用 get_attention_context 取 metadata 与 cache；PCP 下先 maybe_gather_mla_latent_cache_inputs 汇聚跨 rank 的 K 行，再 do_kv_cache_update 统一插入 cache。 - 当 dcp_world_size > pcp_world_size 时对 MQA query 做 TP all_gather；forward_mqa 后用 dcp_manager.combine 合并 logsum，最后 finalize_mla_pcp_decode 还原头维。
4. 放宽稀疏索引器签名（vllm/model_executor/layers/sparse_attn_indexer.py、vllm/v1/attention/ops/rocm_aiter_mla_sparse.py） - sparse_attn_indexer、sparse_attn_indexer_fake、SparseAttnIndexer.forward_native/forward_cuda/forward_hip 以及 ROCm AITER op 的 k 参数由 torch.Tensor 改为 torch.Tensor | None，支撑 skip K cache insert 时传 None。
5. 测试配套 - tests/kernels/test_fused_deepseek_v32_norm_rope.py：新增 test_fused_norm_rope_materializes_pcp_cache_inputs 验证 PCP 物化输出数值与内存别名；test_fused_q 参数化 num_q_heads，覆盖 GQA 头数变化。 - tests/model_executor/layers/test_mla_short_prefill_indexer.py：新增 test_skipped_k_cache_insert_accepts_no_k，验证 k=None 时索引器定制算子仍可完成 top-k 清空并返回原 buffer。

关键文件：

- vllm/models/deepseek_v32/common/kernels.py（模块 核心算子；类别 source；类型 data-contract；符号 _fused_norm_rope_kernel, fused_norm_rope）：核心 Triton kernel 契约改造：fused_norm_rope 从直接写 cache 变为支持物化输出，是 PCP 接入的基础。
- vllm/models/deepseek_v32/attention.py（模块 注意力层；类别 source；类型 data-contract；符号 DeepseekV32Attention.forward, DeepseekV32Attention._sparse_indexer_and_attn）：PCP 逻辑接入点：forward 分流、稀疏索引器调用、K 行汇聚与 cache 更新、PCP/DCP 组合的 combine 流程都在此文件。
- vllm/model_executor/layers/sparse_attn_indexer.py（模块 稀疏索引；类别 source；类型 data-contract；符号 sparse_attn_indexer, sparse_attn_indexer_fake, SparseAttnIndexer.forward_cuda, SparseAttnIndexer.forward_hip）：稀疏索引器 Op 的 k 参数放宽为 Optional，支撑 PCP 路径在 skip_k_cache_insert 时传 None。
- tests/kernels/test_fused_deepseek_v32_norm_rope.py（模块 算子测试；类别 test；类型 test-coverage；符号 test_fused_norm_rope_materializes_pcp_cache_inputs, test_fused_q）：新增 PCP 物化路径的数值与别名测试，并扩展 fused_q 对 GQA 头数的覆盖。
- tests/model_executor/layers/test_mla_short_prefill_indexer.py（模块 索引测试；类别 test；类型 test-coverage；符号 test_skipped_k_cache_insert_accepts_no_k）：验证稀

疏索引器在 k=None 且 skip_k_cache_insert=True 时可正常运行并清空 top-k buffer。

- vllm/v1/attention/ops/rocm_aiter_mla_sparse.py（模块 ROCm 算子；类别 infra；类型 infrastructure；符号 rocm_aiter_sparse_attn_indexer, rocm_aiter_sparse_attn_indexer_fake）：ROCm AITER 稀疏索引器 op 的 k 参数同步放宽为 Optional，保持与 CUDA 侧 Op 契约一致。

关键符号：_fused_norm_rope_kernel, fused_norm_rope, DeepseekV32Attention.forward, DeepseekV32Attention._sparse_indexer_and_attn, sparse_attn_indexer, maybe_gather_mla_latent_cache_inputs, finalize_mla_pcp_decode

关键源码片段

vllm/models/deepseek_v32/common/kernels.py

核心 Triton kernel 契约改造：fused_norm_rope 从直接写 cache 变为支持物化输出，是 PCP 接入的基础。

```
# 关键分支：pid==1 处理 KV RMS Norm + RoPE + 物化 / 直插 cache。
# PCP 模式下 normed kv_c 与 roped k_pe 先落到输出 buffer，
# 由上层 all-gather 汇聚后再统一写入复制的 KV cache。
elif pid == 1:
    kv_block = tl.arange(0, KV_DIM)
    kv_c = tl.load(kv_ptr + tok_idx * kv_stride + kv_block)
    kv_c = _rms_norm(kv_c, kv_rms_w, kv_rms_eps, KV_DIM)

    dim_off = tl.arange(0, KPE_HALF_ROT_DIM)
    x1 = tl.load(kpe_ptr + tok_idx * kpe_stride + dim_off * 2).to(tl.float32)
    x2 = tl.load(kpe_ptr + tok_idx * kpe_stride + dim_off * 2 + 1).to(tl.float32)
    r1 = x1 * cos - x2 * sin
    r2 = x2 * cos + x1 * sin

    # PCP 物化路径：把寄存器里的 norm/rope 结果写出，供跨 rank 汇聚。
    if kv_out_ptr is not None:
        tl.store(kv_out_ptr + tok_idx * kv_out_stride + kv_block, kv_c)
    if kpe_out_ptr is not None:
        # interleaved 布局：相邻两维为一组旋转对
        tl.store(kpe_out_ptr + tok_idx * kpe_out_stride + dim_off * 2,
                 r1.to(kpe_out_ptr.dtype.element_ty))
        tl.store(kpe_out_ptr + tok_idx * kpe_out_stride + dim_off * 2 + 1,
                 r2.to(kpe_out_ptr.dtype.element_ty))

    # 非 PCP 路径：slot_mapping 存在时仍直接写入 MLA cache，
    # 避免为 PCP 单独维护一份 kernel。
    if slot_mapping_ptr is not None:
        slot_idx = tl.load(slot_mapping_ptr + tok_idx)
        dst = (mla_cache_ptr + mla_block_idx * mla_cache_block_stride
               + mla_block_off * mla_cache_entry_stride)
        if MLA_CACHE_FP8:
            scale = tl.load(mla_cache_scale_ptr)
```

```
            tl.store(dst + kv_block,
                     (kv_c.to(tl.float32) / scale).to(tl.float8e4nv))
        else:
            tl.store(dst + kv_block, kv_c)
        # k_pe_roped 按 interleaved 布局写入 cache 尾部
        if MLA_CACHE_FP8:
            tl.store(dst + KV_DIM + dim_off * 2,
                     (r1 / scale).to(tl.float8e4nv))
            tl.store(dst + KV_DIM + dim_off * 2 + 1,
                     (r2 / scale).to(tl.float8e4nv))
        else:
            tl.store(dst + KV_DIM + dim_off * 2, r1)
            tl.store(dst + KV_DIM + dim_off * 2 + 1, r2)
```

vllm/models/deepseek_v32/attention.py

PCP 逻辑接入点：forward 分流、稀疏索引器调用、K 行汇聚与 cache 更新、PCP/DCP 组合的 combine 流程都在此文件。

```
# _sparse_indexer_and_attn 中的 PCP 关键路径。
# PCP 按 seqlen 分片、KV cache 跨 rank 复制（类似 TP），
# 因此新生成的 K 行必须先汇聚再插入 cache。
if self.use_pcp:
    assert kv_c is not None and k_pe is not None
    kv_for_cache, kpe_for_cache, cache_slot_mapping = (
        maybe_gather_mla_latent_cache_inputs(
            kv_c,
            k_pe.unsqueeze(1),
            layer_slot_mapping,
            attn_metadata.num_decode_tokens,
            True,
        )
    )
    # 汇聚后的 K 行统一写入复制的 MLA cache
    self.impl.do_kv_cache_update(
        kv_for_cache,
        kpe_for_cache,
        kv_cache,
        cache_slot_mapping,
        self.kv_cache_dtype,
        self._k_scale,
    )

num_actual = attn_metadata.num_actual_tokens
if num_actual == 0:
    output.zero_()
    return

# PCP 与 DCP 组合：MQA query 需要跨 TP rank all-gather
if self.use_pcp and self.impl.dcp_world_size > self.impl.pcp_world_size:
```

```
        if isinstance(mqa_q_arg, tuple):
            mqa_q_arg = torch.cat(mqa_q_arg, dim=-1)
        mqa_q_arg = get_tp_group().all_gather(mqa_q_arg, dim=1)

    attn_out, lse = self.impl.forward_mqa(mqa_q_arg, kv_cache, attn_metadata, self)

    # DCP 下用 logsum 合并各分片注意力输出，并还原 PCP 头部布局
    if self.use_pcp and self.impl.dcp_world_size > 1:
        assert lse is not None and self.dcp_manager is not None
        attn_out = self.dcp_manager.combine(
            attn_out,
            lse,
            seq_lens=seq_lens,
            query_start_loc=query_start_loc,
        )
        attn_out = finalize_mla_pcp_decode(attn_out, self.num_heads)
```

评论区精华

评论区精华

- WoosukKwon 对 slot_mapping_ptr is None 分支的疑问（kernels.py）："What is this for? Shouldn't we skip the entire kernel?"——既然 slot 为空，为何不整体跳过 kernel。
- GirasoleY 的解释："We still need it to fall through to run the actual logic. Set to 0 is actually a placeholder so following calculation (i.e. mla_block_idx = slot_idx // mla_block_size) does not evaluate None"。即占位符 0 是为了避免后续指针运算对 None 求值；随后改为仅在 slot_mapping_ptr is not None 时加载真实 slot。
- WoosukKwon 要求回归检查："Please check whether dense_mha is working!"，指向 q_nope/ql_nope 计算顺序调整与 dense MHA 组合路径。
- GirasoleY 的决策："Removed mha + pcp support, revert back sparse_attn_indexer() changes in 0a8636cd"——最终不支撑 dense MHA 路径的 PCP，与 PR body 的 "disable PCP for mha path" 一致。
  - slot_mapping 为 None 时 kernel 分支的设计 (design)：改为仅在 slot_mapping_ptr 非 None 时加载真实 slot 索引，物化路径与 paged-cache 寻址解耦。
  - dense MHA 与 PCP 组合的正确性 (correctness)：dense MHA 路径禁用 PCP，与 PR body 的 disable PCP for mha path 一致。

风险与影响

- 风险：### 风险分析
- 核心算子契约变更：fused_norm_rope 增加了三个输出参数并改变了 cache 写入的触发条件。任何未同步更新的调用方（如后续复用的 DeepSeek 系模型或 GLM 系列 kernel）都可能静默丢失 cache 写入或物化输出，需以 attention.py 为唯一适配点并保持测试覆盖。
- PCP/DCP 组合新逻辑：maybe_gather_mla_latent_cache_inputs + do_kv_cache_update + dcp_manager.combine 是首个版本，依赖 MLACommonMetadata 的

num_decode_tokens、seq_lens 等字段；若 metadata 在 CUDAGraph 捕获或 profiling 阶段不完整，可能出现空 slot 或长度不匹配。

- dense MHA 路径禁用 PCP：用户若显式配置 dense MHA 与 PCP，本 PR 不会走 PCP 加速，行为会回退到非 PCP 路径，需文档或日志提示，避免预期落差。
- ROCm 侧仅签名同步：rocm_aiter_mla_sparse.py 只放宽了 k 为 Optional，没有 AMD 平台的 PCP 行为验证；AITER 索引器收到 k=None 时是否安全取决于调用侧 skip_k_cache_insert 的传递。
- 影响：### 影响分析
- 用户侧：启用 PCP 的 DSV3.2 稀疏 MLA 长 prefill 场景获得 2.65x 时延收益；未启用 PCP 或走 dense MHA 的短 prefill 场景不受益但也不回退。
- 系统侧：新增跨 rank 的 K 行汇聚、cache 统一写入、MQA query all-gather 与 logsum combine，通信模式比纯 TP 更复杂，需要在大规模 rank 数下验证扩展效率。
- 团队侧：融合 kernel 的 " 物化 vs 直插 cache" 双模式成为后续模型（如 GLM-5.2 相关分支、DeepSeek 后续版本）接入 PCP 的参考范式，索引器 Op 签名放宽也降低了复用门槛。
- 风险标记：核心算子契约变更，PCP/DCP 组合新逻辑，dense MHA 路径禁用 PCP，ROCm 侧仅签名同步

关联脉络

- PR #52381 Harden DeepSeek V3.2 fused kernel grids: 同一文件 vllm/models/deepseek_v32/common/kernels.py 的 grid 越界修复，本 PR 再次改动该 kernel 的契约与分支逻辑，需关注后续越界风险回归。
- PR #52512 [Bugfix][MLA] Do not use Dense MHA for GLM-5.2: 处理 GLM-5.2 短 prefill 误走 dense MHA 的问题，与本 PR 的 dense MHA 路径 PCP 禁用决策直接相关，且 head 分支名同为 glm52-dsv32 系列。
- PR #49790 GLM-5.2 to DeepSeek V3.2 routing (stacked baseline): PR body 明确说明性能数据是在 #49790 堆叠下测得，是本 PR 的评估基准。