PR #48408 完整报告

vllm-project/vllm

[KV Connector] Add per-layer canonical KV page mappings for parallelism-agnostic offload

合并时间：2026-07-31 19:42

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/48408

执行摘要

- 一句话：KV 卸载新增按层 canonical 页映射，支持跨并行配置复用
- 推荐动作：值得精读。核心看 canonical_mapping.py 如何用 ByteRegion/CopyRun 表达 NHD/HND、packed/split、TP/GQA/MLA、DCP/PCP 的组合映射，以及 CanonicalPageMapping.is_writer 按 block 轮转的负载均衡设计；这是 vLLM KV offload 走向并行无关格式的关键一步。建议结合 follow-up #48414 一起阅读，了解消费端如何使用这些映射。

功能与动机

PR body 说明：在一种并行配置（TP/DCP/PCP）下卸载的 KV cache，希望能在另一种并行配置下被重排或复用；而此前 CanonicalKVCacheRef 只记录 tensor 与页大小，无法表达字节级映射关系。为此引入 canonical（无并行）页作为中间格式，让所有并行推理只发生在 sharding.py（最终落位 canonical_mapping.py），下游消费统一的字节映射。Issue 评论中 oandreeva-nv 追问 MambaSpec 的支持复杂度，说明 canonical 映射需要尽量覆盖更多层类型，而当前实现只认证 Attention 类层。

实现拆解

1. 变更入口与接线：vllm/distributed/kv_transfer/kv_connector/v1/offloading_connector.py 在 WORKER 分支构造 OffloadingConnectorWorker 时新增传参 vllm_config；vllm/distributed/kv_transfer/kv_connector/v1/offloading/worker.py 的 register_kv_caches 在构建 CanonicalKVCaches 之前调用 derive_canonical_mappings(vllm_config, kv_cache_config, kv_caches)，并把得到的 mapping 挂到每个 CanonicalKVCacheRef 上。
2. 数据契约扩展：vllm/v1/kv_offload/base.py 新增 CopyRun（带步长的字节对应关系：fragment 大小、数量、本地 /canonical 步长）和 CanonicalPageMapping（canonical 页大小、本地页大小、runs、num_writers/writer_index、parallelism_agnostic 标志），并实现 is_writer(block_id) 按 block 轮转写者；CanonicalKVCacheRef 增加可空字段 mapping（None 表示未认证）。
3. 映射推导核心：新增 canonical_mapping.py（444 行）。_RankContext 汇总单 rank 的分片参数（tp/dcp/pcp/interleave/total_kv_heads），_layer_mapping 按层认证：MLA 走 latent 单副本 + CP token 交错；普通 Attention 依据 total_kv_heads 与 tp 的关系区分 head sharding 与 GQA 复制，再叠加 DCP/PCP 的 token 交错；未知布局、

per-token-head 量化或非 Attention 层返回 None，由 _opaque_fallback_mapping 生成 rank-private、parallelism_agnostic=False 的兜底映射（fail closed）。

4. 物理布局识别：_attention_byte_regions 根据 tensor shape/stride 区分 packed/split 与 NHD/HND 四种布局，_packed_kv_regions/_split_kv_regions 计算每个 token 重复一次的 ByteRegion（本地偏移、canonical 偏移、每 token 字节数、canonical token 步长）；_coalesce_runs 合并连续片段以减少 copy op 数量，单 rank 场景会折叠为整页一次拷贝。
5. 验证与测试配套：_verify_tiling 在启动期对 worker group 的每条 layer mapping 做整页 tiling 校验（每个 canonical 页恰好被覆盖一次）。新增 tests/v1/kv_connector/unit/offloading_connector/test_canonical_mapping.py，26 个无 GPU 测试覆盖四种布局的 placement、GQA/MLA writer 轮转、DCP/PCP 交错、跨 TP store/load 字节回环以及 fail-closed 门禁；tests/v1/kv_connector/unit/offloading_connector/test_worker.py 适配新构造签名，并断言每层都有 certified 或 opaque 的 mapping。

关键文件：

- vllm/distributed/kv_transfer/kv_connector/v1/offloading/canonical_mapping.py（模块 映射推导；类别 source；类型 core-logic；符号 _RankContext, ByteRegion, _coalesce_runs, _local_to_canonical_token）：新增 444 行的核心推导模块，集中全部并行推理逻辑（TP/DCP/PCP），向下游输出字节级 CopyRun 映射，并实现 fail-closed 的 opaque 兜底。
- tests/v1/kv_connector/unit/offloading_connector/test_canonical_mapping.py（模块 映射测试；类别 test；类型 test-coverage；符号 _ctx, _full_spec, _mla_spec, _split_nhd_cache）：新增 474 行测试，26 个无 GPU 用例覆盖四种物理布局的 placement、GQA/MLA writer 轮转、DCP/PCP 交错、跨 TP store/load 字节回环与 fail-closed 门禁。
- vllm/v1/kv_offload/base.py（模块 卸载核心；类别 source；类型 core-logic；符号 CopyRun, CanonicalPageMapping, is_writer, CanonicalKVCacheRef）：定义 CopyRun 与 CanonicalPageMapping 数据契约，并给 CanonicalKVCacheRef 增加可空 mapping 字段，是下游所有消费者依赖的公共类型。
- vllm/distributed/kv_transfer/kv_connector/v1/offloading/worker.py（模块 连接器；类别 source；类型 dependency-wiring；符号 OffloadingConnectorWorker, register_kv_caches）：接线关键点：构造新增 vllm_config 依赖，register_kv_caches 中调用 derive_canonical_mappings 并把 mapping 挂到 CanonicalKVCacheRef。
- tests/v1/kv_connector/unit/offloading_connector/test_worker.py（模块 连接器测试；类别 test；类型 test-coverage；符号 _single_rank_vllm_config）：适配新构造签名并提供 _single_rank_vllm_config，断言每层都有 certified 或 opaque mapping，防回归。
- vllm/distributed/kv_transfer/kv_connector/v1/offloading_connector.py（模块 连接器；类别 source；类型 core-logic；符号 OffloadingConnector）：WORKER 分支构造 OffloadingConnectorWorker 时传入 vllm_config，完成接线闭环。

关键符号：derive_canonical_mappings, _layer_mapping, _interleave_cp_tokens, _local_to_canonical_token, _coalesce_runs, _attention_byte_regions, _packed_kv_regions, _split_kv_regions, _opaque_fallback_mapping, _verify_tiling, CanonicalPageMapping.is_writer

关键源码片段

vllm/distributed/kv_transfer/kv_connector/v1/offloading/canonical_mapping.py

新增 444 行的核心推导模块，集中全部并行推理逻辑（TP/DCP/PCP），向下游输出字节级 CopyRun 映射，并实现 fail-closed 的 opaque 兜底。

```
def _layer_mapping(
    spec: KVCacheSpec,
    kv_cache: torch.Tensor | list[torch.Tensor] | None,
    num_blocks: int,
    ctx: _RankContext,
) -> CanonicalPageMapping | None:
    """Certified mapping for one layer at one rank, or None (fail closed)."""
    # 只有 Attention 类 spec 能认证；Mamba 等类型直接返回 None，走 opaque 兜底
    if not isinstance(spec, AttentionSpec):
        return None
    bs = spec.block_size
    page = spec.real_page_size_bytes
    # CP 场景要求 interleave 能整除 block_size，否则无法按 chunk 交错 token
    if ctx.cp_size > 1 and (ctx.interleave > bs or bs % ctx.interleave):
        return None

    if isinstance(spec, MLAAttentionSpec):
        # MLA 的 latent 是 TP 复制、CP 切分的，先做前置条件检查
        if (
            spec.compress_ratio != 1
            or page % bs
            or ctx.tp_size % ctx.dcp_size
            or spec.kv_quant_mode.is_per_token_head
        ):
            return None
        row = page // bs # 每个 token 的 latent 字节数
        return CanonicalPageMapping(
            canonical_page_size_bytes=ctx.cp_size * page,
            local_page_size_bytes=page,
            runs=_interleave_cp_tokens([ByteRegion(0, 0, row, row)], bs, ctx),
            num_writers=ctx.tp_size // ctx.dcp_size, # 复制份数
            writer_index=ctx.tp_rank // ctx.dcp_size,
            parallelism_agnostic=ctx.cp_size == 1,
        )

    # per-token-head 量化（如某些 FP8）无法用字节映射表达，直接拒绝认证
    if spec.kv_quant_mode.is_per_token_head or not isinstance(kv_cache, torch.Tensor):
        return None
    total, tp = ctx.total_kv_heads, ctx.tp_size
    if spec.num_kv_heads != max(1, total // tp):
        return None
```

```
    if total >= tp:
        if total % tp:
            return None
        num_head_shards, replication = tp, 1 # 纯 head sharding，无复制
    else:
        if tp % total:
            return None
        num_head_shards, replication = total, tp // total # GQA：head 复制
    # DCP 在持有相同 KV 的复制组内切分 token
    if replication % ctx.dcp_size:
        return None

    head_shard = ctx.tp_rank // replication
    regions = _attention_byte_regions(
        kv_cache, spec, num_blocks, head_shard, num_head_shards, ctx.cp_size
    )
    if regions is None:
        return None # 物理布局无法识别，fail closed
    return CanonicalPageMapping(
        canonical_page_size_bytes=ctx.cp_size * num_head_shards * page,
        local_page_size_bytes=page,
        runs=_interleave_cp_tokens(regions, bs, ctx),
        num_writers=replication // ctx.dcp_size,
        writer_index=(ctx.tp_rank % replication) // ctx.dcp_size,
        parallelism_agnostic=ctx.cp_size == 1,
    )
```

vllm/v1/kv_offload/base.py

定义 CopyRun 与 CanonicalPageMapping 数据契约，并给 CanonicalKVCacheRef 增加可空 mapping 字段，是下游所有消费者依赖的公共类型。

```
@dataclass(frozen=True)
class CopyRun:
    """带步长的字节对应关系：第 i 个 fragment 覆盖本地页
    [local_offset + i * local_stride, +fragment_size) 与 canonical 页
    [canonical_offset + i * canonical_stride, +fragment_size)。
    例如单 rank 场景会被 _coalesce_runs 折叠成整页一次拷贝。
    """
    local_offset: int
    canonical_offset: int
    fragment_size: int
    num_fragments: int
    local_stride: int
    canonical_stride: int


@dataclass(frozen=True)
class CanonicalPageMapping:
    """本 worker 的物理页如何映射到 canonical（无并行）页。
```

```
    仅在进程内使用，从不序列化；runs 双向覆盖完整本地页，
    持有相同字节的多个 rank 按 block 轮转分担写入。
    """
    canonical_page_size_bytes: int # canonical 页大小（含所有 TP/CP 分片）
    local_page_size_bytes: int # 本 worker 未 padding 的页大小
    runs: tuple[CopyRun, ...] # 本地页 <-> canonical 页的字节对应
    num_writers: int # 持有完全相同字节的 rank 数量
    writer_index: int # 本 worker 在这些 rank 中的序号
    parallelism_agnostic: bool # 该块是否与并行配置无关（cp_size == 1）

    def is_writer(self, block_id: int) -> bool:
        """判断本 worker 是否为该 block 的写入者。
        按 block_id 轮转，让持有相同字节的 rank 分摊 store 流量。
        """
        return block_id % self.num_writers == self.writer_index
```

评论区精华

orozery 在 vllm/v1/kv_offload/base.py 上指出：store_runs 将等于 load_runs，且无法在持有相同字节的 rank 间做写负载均衡，建议把 writer election 移到 copy loop 并合并两个字段。最终实现合并为单一 runs 字段，并新增 num_writers/writer_index 与 is_writer(block_id) 按 block_id 轮转分担 store 流量。

orozery 随后给出基于 Claude 的一整套命名与可读性建议（CopyRun、_interleave_cp_tokens、ByteRegion、_opaque_fallback_mapping、_verify_tiling 等），作者全部采纳并表示 “Better naming then I / claude could give ;)”；同时按建议把文件从 vllm/v1/kv_offload/sharding.py 移动到 vllm/distributed/kv_transfer/kv_connector/v1/offloading/ 下。

orozery 提出 canonical_schema_id 目前未使用，建议 defer 到真正使用它的 follow-up，作者回复 “Removed — deferred to the follow-up”。

Issue 评论中 oandreeva-nv 询问 MambaSpec 的 canonical 映射支持复杂度与后续计划，评论中没有作者回复；当前实现里非 AttentionSpec 一律返回 None 走 opaque fallback。

- store_runs/load_runs 合并与按 block 轮转的 writer election (design): 合并为单一 runs 字段，新增 num_writers/writer_index 与 is_writer(block_id) 按 block 轮转，作者回复 Done。
- 命名与文件位置重构 (style): 作者全部采纳，并迁移文件到 vllm/distributed/kv_transfer/kv_connector/v1/offloading/ 下。
- 可读性改进：ByteRegion 与布局检测拆分 (design): 实现拆分为 _packed_kv_regions/_split_kv_regions 并引入 ByteRegion，作者回复 Done。
- canonical_schema_id 未使用，推迟到 follow-up (design): 作者移除该符号及相关常量，回复 Removed — deferred to the follow-up。
- MambaSpec 的 canonical 映射支持复杂度与计划 (question): 评论中无作者回复；当前实现非 AttentionSpec 直接返回 None 走 opaque fallback，Mamba 层暂无认证路径。

风险与影响

- 风险： 1) 字节级偏移推导风险集中在 canonical_mapping.py 的 _packed_kv_regions/_split_kv_regions，stride 判断错误会导致 KV 数据错位；已有 26 个测试覆盖四种布局与 DCP/PCP 组合，但未覆盖所有后端（如 per-token-head 量化的 FP8 会直接被拒绝认证）。

2) 本 PR 只是登记惰性元数据，真实 store/load 路径在 follow-up #48414 才消费 mapping；中间态若后续实现有 bug，已认证层会按新布局读写，fail-closed 兜底无法保护已认证层。

3) worker.py 的 register_kv_caches 仅在 len(tensors_per_block[first_layer_name]) == 1 时挂 mapping，共享同一 tensor 的跨层（cross-layer）场景会得到 None，后续消费者需要明确处理该分支。

4) Mamba 等非 Attention 层退化为 rank-private 映射，在启用这些层的模型上无法获得跨并行配置复用收益。 - 影响：对用户暂无直接功能变化（mapping 是 inert metadata），但为 KV 卸载 / 重载跨 TP、DCP、PCP 配置复用打下基础，未来可显著提升多机缓存复用效率与冷启动命中率。对系统而言，把分散在offload栈中的并行布局知识收敛到单一模块，connector、kv_offload 下游只需按 CopyRun 做字节拷贝；对团队确立了 Certified/opaque 双轨认证策略，后续新增后端只需扩展 _attention_byte_regions 的布局识别即可。 - 风险标记：核心路径变更，字节级偏移复杂度高，消费者在 follow-up 中，Mamba 等层仅 fallback

关联脉络

- PR #48414 Stacked consumer (referenced in PR body): PR body 提到这是 stacked consumer，实际消费这些 canonical page mapping 的 CPU layout 实现；4x H100 端到端验证（tp1/tp2/tp4 + dcp2）依赖它完成。
- PR #50301 [KV Offload] Enable single-copy MLA layout for CPUOffloadingSpec: 同一 KV offload 演进线，默认启用 MLA 单副本布局，与 canonical mapping 中 MLA latent 单写（single-writer election）设计相互呼应。