PR #44539 完整报告

vllm-project/vllm

[mamba] unify KDA conv states into one cache to match 2-state SSM layout

合并时间：2026-06-05 02:38

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/44539

执行摘要

- 一句话：统一 KDA 卷积状态为一个缓存
- 推荐动作：建议精读。这是一个典型的数据结构统一重构，展示了如何在不改变内核计算逻辑的前提下，通过调整状态分组来适配更通用的架构。对于理解 vLLM 中 Mamba 系列（特别是 GDN/KDA）的缓存设计很有帮助。

功能与动机

PR body 明确说明："Purpose align with GDN for future PD support, see https://github.com/vllm-project/vllm/pull/44064"。即为了与 GDN 架构对齐，以便将来支持前缀缓存（PD），需统一 KDA 的卷积状态布局。

实现拆解

1. mamba_utils.py: 修改 MambaStateDtypeCalculator.kda_state_dtype 返回类型从 (dtype, dtype, dtype, float32) 变为 (dtype, float32)；修改 MambaStateShapeCalculator.kda_state_shape：将三个卷积形状（conv_state_shape、conv_state_k_shape、conv_state_k_shape）合并为一个聚合形状 conv_dim = proj_size + 2 * proj_k_size，返回类型从 4 元组变为 (conv_state_shape, recurrent_state_shape)；修改 kda_state_copy_func 从返回 4 个复制函数变为 2 个。
2. kimi_gdn_linear_attn.py: 更新 KimiGatedDeltaNetAttention.get_state_dtype 和 get_state_shape 返回值类型从 4 元组变为 2 元组；在 _forward 中将解包缓存从 (conv_state_q, conv_state_k, conv_state_v, recurrent_state) 改为 (conv_state, recurrent_state)，然后通过 conv_state.chunk(3, dim=-2) 拆出 q/k/v 的 conv state，保持下层 kernel 调用不变。
3. kimi_linear.py: 更新 get_mamba_state_dtype_from_config、get_mamba_state_shape_from_config 和 get_mamba_state_copy_func 的返回类型从 4 元组调整为 2 元组，与上述接口匹配。

关键文件：

- vllm/model_executor/layers/mamba/mamba_utils.py（模块 Mamba 工具；类别 source；类型 data-contract；符号 kda_state_dtype, kda_state_shape, kda_state_copy_func）：核心变更文件：修改了 KDA 的 dtype、shape 和 copy_func，从 4 状态压缩为 2 状态，定义了新的数据契约。

- vllm/model_executor/layers/mamba/gdn/kimi_gdn_linear_attn.py（模块 Kimi GDN；类别 source；类型 data-contract；符号 get_state_dtype, get_state_shape, _forward）：消费端：更新了接口返回值类型并修改缓存解包逻辑，实际使用拆分后的 conv state。
- vllm/model_executor/models/kimi_linear.py（模块 Kimi 模型；类别 source；类型 data-contract；符号 get_mamba_state_dtype_from_config, get_mamba_state_shape_from_config, get_mamba_state_copy_func）：模型入口：同步更新了三个类方法的返回类型签名。

关键符号：kda_state_dtype, kda_state_shape, kda_state_copy_func, get_state_dtype, get_state_shape, _forward

关键源码片段

vllm/model_executor/layers/mamba/mamba_utils.py

核心变更文件：修改了 KDA 的 dtype、shape 和 copy_func，从 4 状态压缩为 2 状态，定义了新的数据契约。

```
# vllm/model_executor/layers/mamba/mamba_utils.py
# 修改后：kda_state_dtype 返回 (state_dtype, float32)
@classmethod
def kda_state_dtype(cls, model_dtype: ModelDType | torch.dtype, mamba_cache_dtype:
MambaDType) -> tuple[torch.dtype, torch.dtype]:
    state_dtype = get_kv_cache_torch_dtype(mamba_cache_dtype, model_dtype)
    return (state_dtype, torch.float32)

# 修改后：kda_state_shape 返回 (conv_state_shape, recurrent_state_shape)
# conv_dim = proj_size + 2 * proj_k_size，合并原三个卷积状态为一个
@classmethod
def kda_state_shape(cls, tp_world_size: int, num_heads: int, head_dim: int,
                    num_k_heads: int | None = None, head_k_dim: int | None = None,
                    conv_kernel_size: int = 4, num_spec: int = 0) -> tuple[tuple[int, int], tuple[int, int,
                    int]]:
    if num_k_heads is None:
        num_k_heads = num_heads
    if head_k_dim is None:
        head_k_dim = head_dim
    proj_size = num_heads * head_dim
    proj_k_size = num_k_heads * head_k_dim
    conv_dim = proj_size + 2 * proj_k_size
    conv_state_shape = cls._orient_conv_shape(divide(conv_dim, tp_world_size), conv_kernel_size
    - 1)
    recurrent_state_shape = (divide(num_heads, tp_world_size), head_dim, head_dim)
    return (conv_state_shape, recurrent_state_shape)

# 修改后：kda_state_copy_func 返回 2 个函数
@classmethod
def kda_state_copy_func(cls) -> tuple[MambaStateCopyFunc, MambaStateCopyFunc]:
    return (get_conv_copy_spec, get_temporal_copy_spec)
```

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

消费端：更新了接口返回值类型并修改缓存解包逻辑，实际使用拆分后的 conv state。

```
# vllm/model_executor/layers/mamba/gdn/kimi_gdn_linear_attn.py
# 修改后：从缓存解包 conv_state 和 recurrent_state
(conv_state, recurrent_state) = constant_caches
# 如果布局不是 DS（dim first），则转置
if not is_conv_state_dim_first():
    conv_state = conv_state.transpose(-1, -2)
# 从合并的 conv_state 中拆出 q/k/v 各自的卷积状态
conv_state_q, conv_state_k, conv_state_v = conv_state.chunk(3, dim=-2)
# 后续使用 conv_state_q, conv_state_k, conv_state_v 调用 causal_conv1d_fn
```

评论区精华

PR 没有 review 评论，reviewer WoosukKwon 和 tdoublep 均直接 approve，tdoublep 评论 "nice clean up"。

- 暂无高价值评论线程

风险与影响

- 风险：风险较低。改动集中在 KDA 内部状态管理，对上层模型接口（kimi_linear.py）仅涉及返回类型签名变更，且被调用方（如 prefix caching 逻辑）在合并依赖此接口的 PR 之前已经适配。测试结果（GSM8K 0.89 accuracy）表明功能正确。潜在风险：若其他未包含在此 PR 中的模块直接解包 4 元组返回值，会因解包数量不匹配而报错。但基于仓库代码范围，所有使用点均已同步修改。
- 影响：直接影响 Kimi-Linear 模型系列（如 moonshotai/Kimi-Linear-48B-A3B-Instruct）的推理缓存管理。缓存从 4 个 tensor 变为 2 个，降低了内存占用和复制开销，提升性能。对用户透明，无需修改配置或代码。对团队而言，此次重构为后续 GDN/PD 功能扫清了障碍。
- 风险标记：数据契约变更，缺少测试文件变更

关联脉络

- PR #44064 GDN related PR (referenced in PR body): 本 PR 旨在对齐 GDN 架构以支持 future PD，PR body 引用了该 PR 作为上下文。
- PR #42554 [PD][Nixl] Mamba prefix caching mode support: 涉及 Mamba 前缀缓存功能，与本 PR 的 PD 支持方向一致。