PR #39823 完整报告

vllm-project/vllm

[Model] Add block-local attention and YaRN for local layers to Gemma3

合并时间：2026-04-22 14:34

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/39823

执行摘要

- 一句话：为 Essential AI 的 Rnj-1 系列模型添加支持，引入块局部注意力和全层 YaRN 配置。
- 推荐动作：建议精读 vllm/model_executor/models/rnj1.py 以理解新模型架构，并关注 vllm/v1/attention/ops/triton_unified_attention.py 中的掩码实现，了解块局部注意力与滑动窗口的设计权衡。对于维护者，需注意后端限制和未来扩展可能性。

功能与动机

根据 PR body，Essential AI 的 Rnj-1 系列模型与 Gemma3 架构相似，但需要两个关键变更：全局和局部层均使用 YaRN（Gemma3 仅全局层使用），以及局部层使用块局部注意力代替滑动窗口。这些变更是为了适配新模型的架构需求，确保在 vLLM 中正确运行。

实现拆解

1. 新增 Rnj1 模型文件：创建 vllm/model_executor/models/rnj1.py，实现 Rnj1MLP、Rnj1Attention、Rnj1DecoderLayer 和 Rnj1Model 类，支持块局部注意力和全层 YaRN，基于 Gemma3 架构但独立维护。
2. 修改注意力后端检查：在 vllm/model_executor/layers/attention/attention.py 的 __init__ 方法中添加断言，确保块局部注意力仅使用 TRITON_ATTN 后端，避免不兼容。
3. 扩展 Triton 注意力后端：修改 vllm/v1/attention/backends/triton_attn.py，在 __init__ 和 forward 方法中添加 chunk_lookback 参数，传递给内核。
4. 更新 Triton 统一注意力内核：修改 vllm/v1/attention/ops/triton_unified_attention.py 的 kernel_unified_attention_2d 和 kernel_unified_attention_3d 函数，引入 CHUNK_LOOKBACK 和 CHUNK_SIZE 常量，实现块局部注意力的掩码计算，替代滑动窗口逻辑。
5. 更新注册表和测试：在 vllm/model_executor/models/registry.py 中添加 Rnj1ForCausalLM 映射，并在 tests/models/registry.py 中补充测试覆盖，确保模型可被正确加载。
6. 更新文档：修改 docs/models/supported_models.md，添加 Rnj1 模型支持说明。

关键文件：

- vllm/model_executor/models/rnj1.py（模块 模型层；类别 source；类型 core-logic；符号 Rnj1MLP, Rnj1Attention, Rnj1DecoderLayer, Rnj1Model）：新增 Rnj1 模型实现文件，

包含核心类如 Rnj1Attention 和 Rnj1Model，支持块局部注意力和全层 YaRN，是 PR 的主要变更。

- vllm/v1/attention/ops/triton_unified_attention.py（模块 注意力内核；类别 source；类型 core-logic）：修改 Triton 统一注意力内核，实现块局部注意力的掩码逻辑，是关键的性能和正确性变更。
- vllm/model_executor/layers/attention/attention.py（模块 注意力层；类别 source；类型 core-logic）：修改注意力层初始化，添加后端检查，确保块局部注意力仅使用 Triton 后端，避免运行时错误。
- vllm/v1/attention/backends/triton_attn.py（模块 注意力后端；类别 source；类型 core-logic）：扩展 Triton 注意力后端，添加 chunk_lookback 参数并传递给内核，支持块局部注意力配置。
- vllm/model_executor/models/registry.py（模块 注册表；类别 source；类型 data-contract）：更新模型注册表，添加 Rnj1ForCausalLM 映射，使 vLLM 能识别和加载新模型。
- tests/models/registry.py（模块 测试；类别 test；类型 test-coverage）：更新测试文件，添加 Rnj1ForCausalLM 的测试覆盖，确保模型注册正确且可测试。
- docs/models/supported_models.md（模块 文档；类别 docs；类型 documentation）：更新文档，添加 Rnj1 模型支持说明，保持文档与代码同步。

关键符号：Rnj1MLP.forward, Rnj1Attention.init, Rnj1Model.forward, kernel_unified_attention_2d, kernel_unified_attention_3d

关键源码片段

vllm/model_executor/models/rnj1.py

新增 Rnj1 模型实现文件，包含核心类如 Rnj1Attention 和 Rnj1Model，支持块局部注意力和全层 YaRN，是 PR 的主要变更。

```
class Rnj1Attention(nn.Module):
    def __init__(
        self,
        config: Gemma3TextConfig,
        hidden_size: int,
        num_heads: int,
        num_kv_heads: int,
        head_dim: int,
        max_position_embeddings: int,
        cache_config: CacheConfig | None = None,
        quant_config: QuantizationConfig | None = None,
        attn_logits_soft_cap: float | None = None,
        prefix: str = "",
    ) -> None:
        super().__init__()
        self.config = config
        self.hidden_size = hidden_size
        tp_size = get_tensor_model_parallel_world_size()
```

```
self.total_num_heads = num_heads
assert self.total_num_heads % tp_size == 0
self.num_heads = self.total_num_heads // tp_size
self.total_num_kv_heads = num_kv_heads
if self.total_num_kv_heads >= tp_size:
    assert self.total_num_kv_heads % tp_size == 0
else:
    assert tp_size % self.total_num_kv_heads == 0
self.num_kv_heads = max(1, self.total_num_kv_heads // tp_size)
self.head_dim = head_dim
self.q_size = self.num_heads * self.head_dim
self.kv_size = self.num_kv_heads * self.head_dim
self.scaling = config.query_pre_attn_scalar**-0.5
# 初始化 QKV 投影层，支持量化配置
self.qkv_proj = QKVParallelLinear(
    hidden_size,
    self.head_dim,
    self.total_num_heads,
    self.total_num_kv_heads,
    bias=config.attention_bias,
    quant_config=quant_config,
    prefix=f"{prefix}.qkv_proj",
)
self.o_proj = RowParallelLinear(
    self.total_num_heads * self.head_dim,
    hidden_size,
    bias=config.attention_bias,
    quant_config=quant_config,
    prefix=f"{prefix}.o_proj",
)
# 获取旋转位置编码，使用 YaRN 类型
self.rotary_emb = get_rope(
    head_dim=self.head_dim,
    rotary_dim=self.head_dim,
    max_position=max_position_embeddings,
    base=config.rope_theta,
    is_neox_style=True,
    rope_type="yarn", # 所有层均使用 YaRN，与 Gemma3 不同
    config=config,
)
# 注意力层，传递额外参数如 chunk_lookback 以支持块局部注意力
self.attn = Attention(
    self.num_heads,
    self.head_dim,
    self.scaling,
    num_kv_heads=self.num_kv_heads,
    cache_config=cache_config,
    quant_config=quant_config,
    attn_logits_soft_cap=attn_logits_soft_cap,
```

```
            rope_rotary_emb=self.rotary_emb,
            prefix=prefix,
            # 通过 extra_impl_args 传递 chunk_lookback，在 attention.py 中处理
            extra_impl_args={"chunk_lookback": 1}, # 默认 lookback 为 1
        )
```

vllm/v1/attention/ops/triton_unified_attention.py

修改 Triton 统一注意力内核，实现块局部注意力的掩码逻辑，是关键的性能和正确性变更。

```
# 在 kernel_unified_attention_2d 函数中，添加块局部注意力掩码计算
if CHUNK_LOOKBACK > -1:
    # 块局部注意力：每个查询只能关注当前块和前 CHUNK_LOOKBACK 个块
    # 计算查询的绝对位置和块索引
    q_abs = context_len + query_pos[:, None]
    seq_mask = seq_mask & (
        (
            (q_abs // CHUNK_SIZE) # 查询所在块索引
            - (seq_offset[None, :] // CHUNK_SIZE) # 键所在块索引
        )
        <= CHUNK_LOOKBACK # 允许的块偏移量
    )
elif SLIDING_WINDOW > 0:
    # 滑动窗口注意力：保持原有逻辑
    seq_mask = seq_mask & ((query_abs_pos - seq_offset) < SLIDING_WINDOW)
# 注意：块局部注意力与滑动窗口互斥，通过条件分支实现
```

评论区精华

review 中主要讨论了：

1) 参数配置读取：gemini-code-assist[bot] 建议从模型配置读取 chunk_lookback 参数以提高灵活性，但作者 defer 给 reviewer，最终未采纳； 2) 重命名与对齐：tdoublep 询问是否向 transformers 提交 PR 添加新层类型，作者重命名为 'chunked_attention' 以与现有术语对齐； 3) 新模型文件创建：tdoublep 建议创建独立模型文件而非修改 gemma3.py，以避免代码交织，最终采纳并新增 rnj1.py； 4) 实现细节：讨论块局部注意力与滑动窗口的互斥性，以及 BLOCK_LOCAL_LOOKBACK=0 的含义（表示仅关注当前块）。

- 参数配置读取灵活性 (design): 作者 defer 给 reviewer，最终未采纳，保持硬编码以简化实现。
- 重命名为 chunked attention (design): 采纳重命名，从 'block-local' 改为 'chunked'，并在代码中更新。
- 创建新模型文件而非修改现有 (design): 采纳，新增 rnj1.py 文件，恢复 gemma3.py 到原始状态。
- 块局部注意力与滑动窗口互斥性 (correctness): 确认互斥，内核中通过 CHUNK_LOOKBACK 和 SLIDING_WINDOW 分支实现。

风险与影响

- 风险：技术风险包括： 1) 后端限制：块局部注意力仅支持 TRITON_ATTN 后端，在 vllm/model_executor/layers/attention/attention.py 中通过断言强制，可能在其他后端上导致运行时错误； 2) 维护复杂度：新增 rnj1.py 文件虽减少与 gemma3.py 的耦合，但增加模型文件数量，需长期维护； 3) 测试覆盖不足：尽管有 registry 测试，但缺少针对块局部注意力内核的专项测试，可能隐藏边界条件问题。
  - 影响：对用户：支持 Essential AI 的新模型，扩展 vLLM 的模型生态，用户可加载 Rnj-1 系列模型进行推理。对系统：新增注意力模式可能轻微影响性能，但测试显示对 Gemma3 无回归，且块局部注意力优化了长序列处理。对团队：需维护新模型文件，但设计为最小化干扰，遵循代码分离原则。
  - 风险标记：核心路径变更，后端限制，新增模型维护

关联脉络

- PR #36268 [Audio] Bundle get_generation_prompt() params into SpeechToTextParams: 类似地，该 PR 引入新的数据类以统一参数，展示 vLLM 中模型接口的演进模式。
- PR #39750 [Refactor] Remove unused param: 涉及模型参数清理，与本 PR 新增模型文件形成对比，反映代码维护策略。