PR #46808 完整报告

vllm-project/vllm

[GLM-5] Add DSV3.2/GLM5 to `vllm/models/`

合并时间：2026-06-27 05:09

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/46808

执行摘要

- 一句话：添加 DeepSeek V3.2/GLM-5 硬件特定模型实现
- 推荐动作：建议架构师和模型开发者精读 attention.py 中的 DeepseekV32Indexer 实现，理解稀疏注意力索引生成逻辑，以及 model.py 中的层间残差连接设计，为后续优化奠定基础。

功能与动机

根据 PR 描述："该 PR 增加了一个新的、无需 torch.compile、硬件特定的 DeepSeek V3.2 和 GLM-5 实现……目的是先落地模型结构，以便后续优化工作。"（原文：to land the model structure first and enable follow-up optimization work）

实现拆解

1. 创建目录结构和入口：新增 vllm/models/deepseek_v32/ 包，__init__.py 对非 NVIDIA SM100 平台抛出 NotImplementedError，并从 nvidia/ 子包导入模型类 DeepseekV32ForCausalLM 和 DeepseekV32MTP。
2. 实现注意力层（attention.py）：定义了 DeepseekV32Indexer 类，用于生成稀疏注意力所需的 top-k 索引和键值缓存；DeepseekV32Attention 类封装了 MLA 注意力与稀疏索引的集成，支持 MLAAttention 和页式 KV 缓存。
3. 实现解码器和主模型（model.py）：DeepseekV32DecoderLayer 组合了 DeepseekV32Attention 和 前馈网络（复用 DeepseekV2MoE/DeepseekV2MLP）；DeepseekV32Model 和 DeepseekV32ForCausalLM 构建完整模型，支持流水线并行和权重加载。
4. 实现多 token 预测（mtp.py）：DeepseekV32MultiTokenPredictorLayer 和 DeepseekV32MultiTokenPredictor 支持 speculative decoding 的多步预测，DeepseekV32MTP 作为顶层入口，包含 embed_input_ids、compute_logits 等方法。
5. 注册表回滚：在最终提交 ff028d6 中移除了对 registry.py 的修改，确保现有 DeepSeek V3.2 和 GLM-5 实现继续默认使用。

关键文件：

- vllm/models/deepseek_v32/nvidia/attention.py（模块 注意力层；类别 source；类型 data-contract；符号 DeepseekV32Indexer, init, forward, DeepseekV32Attention）：核心注意力层，定义 DeepseekV32Indexer 和 DeepseekV32Attention，实现稀疏注意力索

引生成与 MLA 前向。

- vllm/models/deepseek_v32/nvidia/model.py（模块 模型主干；类别 source；类型 data-contract；符号 DeepseekV32DecoderLayer, init, forward, DeepseekV32Model）：模型主干，包含 DeepseekV32DecoderLayer、DeepseekV32Model 和 DeepseekV32ForCausalLM，组合注意力与 MoE。
- vllm/models/deepseek_v32/nvidia/mtp.py（模块 推测解码；类别 source；类型 data-contract；符号 DeepseekV32MultiTokenPredictorLayer, init, forward, DeepseekV32MultiTokenPredictor）：多 token 预测模块，支持 speculative decoding 的多步预测。
- vllm/models/deepseek_v32/__init__.py（模块 入口；类别 source；类型 data-contract）：包入口，包含平台检查与模型类导出，关键讨论点。
- vllm/models/deepseek_v32/nvidia/__init__.py（模块 子包；类别 source；类型 data-contract）：子包空 init，包结构必需。

关键符号：DeepseekV32Indexer.init, DeepseekV32Indexer.forward, DeepseekV32Attention.init, DeepseekV32Attention.forward, DeepseekV32Attention._sparse_attention, DeepseekV32DecoderLayer.init, DeepseekV32DecoderLayer.forward, DeepseekV32Model.embed_input_ids, DeepseekV32Model.load_weights, DeepseekV32ForCausalLM.set_moe_parameters, DeepseekV32ForCausalLM.forward, DeepseekV32MultiTokenPredictorLayer.forward, DeepseekV32MultiTokenPredictor.set_skip_topk, DeepseekV32MultiTokenPredictor.forward, DeepseekV32MultiTokenPredictor.compute_logits, DeepseekV32MTP.embed_input_ids, DeepseekV32MTP.compute_logits

关键源码片段

vllm/models/deepseek_v32/nvidia/attention.py

核心注意力层，定义 DeepseekV32Indexer 和 DeepseekV32Attention，实现稀疏注意力索引生成与 MLA 前向。

```
class DeepseekV32Indexer(nn.Module):
    """DeepSeek V3.2 稀疏注意力索引器：生成 top-k token 索引用于稀疏 MLA 注意力。"""
    def __init__(self, vllm_config, config, hidden_size, q_lora_rank, quant_config, cache_config,
    topk_indices_buffer, prefix=""):
        super().__init__()
        # 从配置中读取索引相关参数
        self.topk_tokens = config.index_topk
        self.n_head = config.index_n_heads
        self.head_dim = config.index_head_dim
        self.rope_dim = config.qk_rope_head_dim
        # 不使用张量并行，直接复制
        self.wq_b = ReplicatedLinear(self.q_lora_rank, self.head_dim * self.n_head, bias=False,
        quant_config=quant_config, prefix=f"{prefix}.wq_b")
        # 融合的 wk + weights_proj 线性层：一次 GEMM 产生 [head_dim + n_head]
        # FP8 wk 权重在加载时被提升至 BF16 以保持融合
        self.wk_weights_proj = MergedColumnParallelLinear(hidden_size, [self.head_dim, self.n_
```

```
        head], bias=False, quant_config=None, disable_tp=True, prefix=f"{prefix}.wk_weights_proj")
        self.k_norm = LayerNorm(self.head_dim, eps=1e-6)
        self.softmax_scale = self.head_dim ** -0.5
        # FP8 缓存的缩放格式与量化块大小
        self.scale_fmt = "ue8m0"
        self.quant_block_size = 128
        self.topk_indices_buffer = topk_indices_buffer
        # FP8 类型缓存：值以 FP8 格式存储，每个 quant_block_size 元素附带 FP32 缩放因子
        self.k_cache = DeepseekV32IndexerCache(head_dim=self.head_dim + self.head_dim // self.
        quant_block_size * 4, dtype=torch.uint8, prefix=f"{prefix}.k_cache", cache_config=cache_
        config)
        self.max_model_len = vllm_config.model_config.max_model_len
        self.indexer_op = SparseAttnIndexer(self.k_cache, self.quant_block_size, self.scale_fmt,
        self.topk_tokens, self.head_dim, self.max_model_len, self.max_total_seq_len, self.topk_
        indices_buffer)

    def forward(self, hidden_states, qr, positions, rotary_emb):
        # 通过 wq_b 得到 query，并分割为 rope 和非 rope 部分
        q, _ = self.wq_b(qr)
        q = q.view(-1, self.n_head, self.head_dim)
        q_pe, q_nope = torch.split(q, [self.rope_dim, self.head_dim - self.rope_dim], dim=-1)
        # 融合的 wk + weights_proj：一次 GEMM 产出 [head_dim + n_head]
        kw, _ = self.wk_weights_proj(hidden_states)
        k = kw[:, :self.head_dim] # 前半为 key
        weights = kw[:, self.head_dim:] # 后半为 attention 权重
        k = self.k_norm(k)
        k_pe, k_nope = torch.split(k, [self.rope_dim, self.head_dim - self.rope_dim], dim=-1)
        # 应用旋转位置编码
        q_pe, k_pe = rotary_emb(positions, q_pe, k_pe)
        # ... 后续代码处理稀疏索引计算、缓存写入等（省略）
```

vllm/models/deepseek_v32/nvidia/model.py

模型主干，包含 DeepseekV32DecoderLayer、DeepseekV32Model 和 DeepseekV32ForCausalLM，组合注意力与 MoE。

```
class DeepseekV32DecoderLayer(torch.nn.Module):
    """DeepSeek V3.2 解码器层：包含自注意力与混合专家前馈网络。"""
    def __init__(self, vllm_config, prefix, config=None, topk_indices_buffer=None):
        super().__init__()
        config = config or vllm_config.model_config.hf_config
        quant_config = vllm_config.quant_config
        self.hidden_size = config.hidden_size
        layer_idx = int(prefix.split(".")[-1])
        self.layer_idx = layer_idx
        # 自注意力层
        self.self_attn = DeepseekV32Attention(vllm_config, config, prefix=f"{prefix}.self_attn", topk_
        indices_buffer=topk_indices_buffer)
        # 根据层索引决定使用 MoE 还是密集 MLP
        if config.n_routed_experts is not None and layer_idx >= config.first_k_dense_replace and
```

```
            layer_idx % config.moe_layer_freq == 0:
                self.mlp = DeepseekV2MoE(config, parallel_config, quant_config, prefix=f"{prefix}.mlp")
            else:
                self.mlp = DeepseekV2MLP(config.hidden_size, config.intermediate_size, config.hidden_
                act, quant_config=quant_config, prefix=f"{prefix}.mlp")
            self.input_layernorm = RMSNorm(config.hidden_size, eps=config.rms_norm_eps)
            self.post_attention_layernorm = RMSNorm(config.hidden_size, eps=config.rms_norm_eps)

        def forward(self, positions, hidden_states, residual):
            # 残差连接与层归一化
            if residual is None:
                residual = hidden_states
                hidden_states = self.input_layernorm(hidden_states)
            else:
                hidden_states, residual = self.input_layernorm(hidden_states, residual)
            hidden_states = self.self_attn(positions=positions, hidden_states=hidden_states)
            hidden_states, residual = self.post_attention_layernorm(hidden_states, residual)
            hidden_states = self.mlp(hidden_states)
            return hidden_states, residual
```

评论区精华

- 平台限制与注册表变更：tjtanaa 指出 __init__.py 中硬编码的平台检查会导致非 CUDA 平台崩溃，且注册表变更会破坏已有测试。WoosukKwon 解释这仅为开发阶段暂用，并在 commit ff028d6 中回滚了注册表修改。
- 性能询问：yewentao256 请求运行 e2e benchmark 比较新旧实现性能，但未得到答复，预期在后续优化后再评估。
- 平台限制与注册表变更 (design): WoosukKwon 承认后注册表变更已回滚，平台限制保留为当前开发状态，未来计划支持 AMD/XPU。
- 性能比较请求 (question): 未得到答复，预期在后续优化后再评估。

风险与影响

- 风险：
  1. 无测试覆盖：所有新增文件无配套测试，回归风险高。
  2. 平台限制：当前仅支持 NVIDIA SM100，其他平台若未及时适配将不可用。
  3. 性能未优化：未包含 op fusion，当前性能不具竞争力，需后续优化。
  4. 与旧实现并存：未来切换可能带来兼容性问题（如权重格式、配置差异）。
  5. 注册表回滚风险：开发期间对注册表的修改若未完全回滚可能影响模型加载。 - 影响：当前对用户无直接影响，因为新实现未激活；对系统新增约 1170 行代码，增加维护负担；对团队提供了模型架构基础，为后续性能优化和硬件适配铺路。 - 风险标记：无测试覆盖，平台受限仅 NVIDIA, 性能未优化，与旧实现并存

关联脉络

- 暂无明显关联 PR