

PR #54917 完整报告

vllm-project/vllm

[Bugfix][Gemma] Conditionally create KV projections/norms on KV-shared layers

合并时间: 2026-09-07 22:04

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/54917>

执行摘要

- 一句话: 修复 Gemma 4/3n 模型 KV 共享层条件创建, 解决微调检查点加载失败问题。
- 推荐动作: 此 PR 值得精读。它修复了一个影响面广的微调模型加载 bug, 且采用的条件创建方案比最初的“权重标记”hack 更优雅、更符合模型设计意图。核心设计决策 (在 `__init__` 中根据配置条件创建模块, 而非在 `forward` 或 `load_weights` 中打补丁) 值得学习和借鉴。审查者推动的重构建议直接提升了代码质量。建议关注其权重映射 (`_kv_sharing_weights_mapper`) 的设计, 以及如何通过分叉的前向路径和加载逻辑来同时兼容原始检查点和微调检查点。

功能与动机

核心动机是解决微调检查点加载失败。如 PR body 所述, HuggingFace transformers 在 Gemma 4 E2B/E4B 和 Gemma 3n 的 KV 共享层上不创建 `k_proj`、`v_proj`、`k_norm`。任何通过 `save_pretrained` 保存的检查点 (如来自 HF Trainer、TRL、verl) 都会省略这些张量。原始 Google 发布的检查点恰好冗余地包含了它们, 因此问题仅在微调后显现。vLLM 原先无条件创建这些模块, 导致加载微调检查点时出现 `ValueError`。

实现拆解

1. 模块结构重构 (Gemma 4) : 在 `vllm/model_executor/models/gemma4.py` 的 `Gemma4Attention.__init__` 中, 根据 `self.is_kv_shared_layer` 标志条件创建 `k_proj/v_proj/k_norm/v_norm`。KV 共享层仅创建 `q_proj` (`ColumnParallelLinear`), 非共享层创建 `qkv_proj` (`QKVParallelLinear`) 及其相关的 `norm` 层。
2. 模块结构重构 (Gemma 3n) : 在 `vllm/model_executor/models/gemma3n.py` 的 `Gemma3nAttention.__init__` 中实施相同的条件创建逻辑。KV 共享层创建 `q_proj`, 非共享层创建 `qkv_proj`、`k_norm` 和 `v_norm`。
3. 权重映射适配 (Gemma 3n) : 新增 `_kv_sharing_weights_mapper` 函数, 为 Gemma 3n 生成权重映射。该映射将非 KV 共享层的 `q/k/v_proj` 打包到 `qkv_proj`, 同时将 KV 共享层的冗余 `k_proj`、`v_proj`、`k_norm` 映射为 `None` (丢弃)。在 `Gemma3nTextModel.__init__` 中, 将此映射与原有的 `hf_to_vllm_mapper` 合并。
4. 前向传播适配: 在两个模型的 `forward` 方法中, 根据 `self.is_kv_shared_layer` 标志分叉执行路径。共享层仅投影 Q 并应用 RoPE, K/V 从目标层的缓存中获取; 非共享层执行完整的 QKV 投影、`norm` 和 RoPE。

5. 权重加载逻辑微调 (Gemma 4) : 在 `Gemma4Model.load_weights` 中, 调整了堆叠参数映射的注释和回退逻辑, 使其在 KV 共享层 (仅有 `q_proj`) 无法找到堆叠的 `qkv_proj` 时, 能优雅地跳过并回退到直接加载路径, 从而跳过原始检查点中冗余的 K/V 张量。
6. 测试覆盖: 在 `tests/models/language/generation/test_gemma.py` 中新增两个单元测试: `test_gemma4_kv_shared_layer_loads_plain_q_proj` 验证 Gemma 4 的权重加载行为; `test_gemma3n_kv_shared_layer_mapper` 验证 Gemma 3n 的权重映射逻辑。

关键文件:

- `vllm/model_executor/models/gemma4.py` (模块 注意力模块; 类别 source; 类型 core-logic; 符号 `Gemma4Attention.init`, `Gemma4Attention.forward`, `Gemma4Model.load_weights`) : 核心变更文件。重构了 `Gemma4Attention` 的 `__init__` 和 `forward` 方法, 条件创建 KV 相关模块并分叉前向路径, 是修复的主要实现。
- `vllm/model_executor/models/gemma3n.py` (模块 注意力模块; 类别 source; 类型 core-logic; 符号 `Gemma3nAttention.init`, `Gemma3nAttention.forward`, `_kv_sharing_weights_mapper`) : 另一个核心变更文件, 对 `Gemma3nAttention` 实施了与 Gemma 4 类似的结构重构, 并新增了专用的权重映射函数 `_kv_sharing_weights_mapper`。
- `tests/models/language/generation/test_gemma.py` (模块 模型测试; 类别 test; 类型 test-coverage; 符号 `test_gemma4_kv_shared_layer_loads_plain_q_proj`, `test_gemma3n_kv_shared_layer_mapper`) : 关键测试文件, 新增了两个单元测试以验证修复的正确性, 覆盖了 Gemma 4 和 Gemma 3n 的 KV 共享层加载与映射逻辑。

关键符号: `Gemma4Attention.init`, `Gemma4Attention.forward`, `Gemma4Model.load_weights`, `Gemma3nAttention.init`, `Gemma3nAttention.forward`, `_kv_sharing_weights_mapper`, `test_gemma4_kv_shared_layer_loads_plain_q_proj`, `test_gemma3n_kv_shared_layer_mapper`

关键源码片段

`vllm/model_executor/models/gemma4.py`

核心变更文件。重构了 `Gemma4Attention` 的 `__init__` 和 `forward` 方法, 条件创建 KV 相关模块并分叉前向路径, 是修复的主要实现。

```
# vllm/model_executor/models/gemma4.py # Gemma4Attention.__init__ 中的条件创建逻辑
layer_idx = extract_layer_index(prefix) num_kv_shared_layers = getattr(config, "
num_kv_shared_layers", 0) first_kv_shared_layer_idx = config.num_hidden_layers -
num_kv_shared_layers self.is_kv_shared_layer = (    num_kv_shared_layers > 0 and
layer_idx >= first_kv_shared_layer_idx ) if self.is_kv_shared_layer:    # KV 共享层: 仅创
建 q_proj, K/V 来自目标层的缓存    # 与 HuggingFace transformers 中
Gemma4Attention 的行为一致    self.q_proj = ColumnParallelLinear(        hidden_size,
        self.total_num_heads * self.head_dim,        bias=config.attention_bias,
quant_config=quant_config,        prefix=f"{prefix}.q_proj",    ) else:    # 非共享层: 创
建完整的 QKV 投影和 norm 层    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.k_norm = RMSNorm(self.head_dim,
eps=config.rms_norm_eps)    self.v_norm = RMSNorm(        self.head_dim,
eps=config.rms_norm_eps, has_weight=False    ) #
vllm/model_executor/models/gemma4.py # Gemma4Attention.forward 中分叉的执行路径
defforward(    self,    positions: torch.Tensor,    hidden_states: torch.Tensor,
**kwargs, ) -> torch.Tensor:    if self.is_kv_shared_layer:        # 共享 KV: 仅投影 Q,
K/V 来自目标层        q, _ = self.q_proj(hidden_states)        q = q.unflatten(-1,
(self.num_heads, self.head_dim))        q = self.q_norm(q)        q = q.flatten(-2, -1)
        q, _ = self.rotary_emb(positions, q, None)        attn_output = self.attn(q, None,
None)    else:        # 完整路径: 投影 QKV, 应用 norm 和 RoPE        qkv, _ =
self.qkv_proj(hidden_states)        q, k, v = qkv.split([self.q_size, self.kv_size,
self.kv_size], dim=-1)        # ... (norm 和 RoPE 应用 ) ...        attn_output =
self.attn(q, k, v)    output, _ = self.o_proj(attn_output)    return output

```

vllm/model_executor/models/gemma3n.py

另一个核心变更文件，对 `Gemma3nAttention` 实施了与 Gemma 4 类似的结构重构，并新增了专用的权重映射函数 `_kv_sharing_weights_mapper`。

```

# vllm/model_executor/models/gemma3n.py
# 新增的权重映射函数，用于处理 KV 共享层的权重
def _kv_sharing_weights_mapper(config: Gemma3nTextConfig) -> WeightsMapper:
    """KV-shared layers only have q_proj, so qkv_proj packing applies to the
    other layers. Original checkpoints still ship K/V tensors for the shared
    layers (fine-tuned ones omit them); those are dropped."""
    first_kv_shared_layer_idx = config.num_hidden_layers - config.num_kv_shared_layers
    return WeightsMapper(
        # 对于 KV 共享层 (i >= first_kv_shared_layer_idx):
        # 将 k_proj, v_proj, k_norm 映射为 None，即丢弃这些权重
        orig_to_new_substr={
            f"layers.{i}.self_attn.{name}": None
            for i in range(first_kv_shared_layer_idx, config.num_hidden_layers)
            for name in ("k_proj", "v_proj", "k_norm")
        },
        # 对于非 KV 共享层 (i < first_kv_shared_layer_idx):
        # 将 q/k/v_proj 打包到 qkv_proj，与原始行为一致
        orig_to_new_stacked={
            f"layers.{i}.self_attn.{shard}_proj": (
                f"layers.{i}.self_attn.qkv_proj.",
                shard,
            )
            for i in range(first_kv_shared_layer_idx)
            for shard in ("q", "k", "v")
        },
    )

```

评论区精华

核心讨论围绕修复策略展开。评审者 [hmellor](#) 最初要求变更请求，明确指出应该“有条件地创建层”，而不是无条件创建后再用权重加载“hack”来规避问题。他指出最初的修复没有修正 Gemma3n，因为 `k_norm` 仍在 `Gemma3nAttention.forward` 中被无条件调用，并且将过于 Gemma 特定的逻辑放在了通用的 `utils.py` 文件中。作者 [Josephasafg](#) 接受了这一更优的架构方案，并提交了新版本，将 `k_norm/v_norm` 的创建移入 `else` 块（即非共享层分支）。最终 [hmellor](#) 批准了修改后的 PR。

- 修复策略：条件创建 vs. 权重标记 hack (design): 作者 [Josephasafg](#) 接受了该建议，提交新版本，将 `k_norm/v_norm` 的创建移入 `else` 块（非共享层分支），使模块结构与 HuggingFace transformers 一致。最终 [hmellor](#) 批准了此方案。
- AMD CI 测试失败 (correctness): AMD CI 失败被识别为与另一个问题相关，需要合并 #55887 来解决，与本 PR 的核心逻辑无直接关联。

风险与影响

- 风险：正确性风险：核心风险在于确保 KV 共享层的前向传播路径完全正确。修改后的 `forward` 方法在共享层中传入 `None` 作为 K/V，依赖底层的 Attention 模块正确处理这种情况并从共享目标层获取缓存。已通过单元测试和端到端 GSM8K 准确度测试验证。兼容性风险：此变更旨在恢复与 HuggingFace 检查点的兼容性，但需确保原始 Google 检查点（包含冗余张量）仍能正确加载。Gemma 4 的 `load_weights` 通过堆叠参数映射的回退逻辑处理此问题，Gemma 3n 通过新的 `_kv_sharing_weights_mapper` 显式丢弃冗余张量。性能风险：无。条件创建减少了共享层的参数数量和计算量，性能测试显示无回归。AMD CI 延迟：合并后才发现 AMD CI (MI300) 上的语音到文本集成测试失败，但这似乎是由另一个已知问题 (#55887) 引起，与本 PR 的核心修改无直接关联。
- 影响：用户影响：直接影响所有使用微调后 Gemma 4 E2B/E4B 或 Gemma 3n 模型的用户，修复了模型加载失败的阻断性问题，使这些模型可以在 vLLM 中正常部署。系统影响：修改了 vLLM 中 Gemma 系列模型的核心注意力模块的数据契约（模块创建逻辑），使模型结构更贴近参考实现（HuggingFace transformers）。这影响了模型加载、权重初始化和前向计算路径。团队影响：为处理类似的“KV 共享层”问题提供了更清晰、更符合模型原生结构的模式。PR 作者与评审者约定将后续的加载逻辑重构（迁移到 `AutoWeightsLoader`）和分片级验证作为独立跟进任务，表明此修复是解决更广泛问题的关键一步。
- 风险标记：核心路径变更，模型数据契约变更，跨模型一致性修复

关联脉络

- PR #41504 [Bugfix][Gemma] conditionally create KV projections/norms on KV-shared layers: 被本 PR 明确提及的“前身”PR，因过期且作者未响应而被取代。两者试图解决相同的根因问题。
- PR #55887 [Bugfix][ROCm][DSv4] Skip launch_pdl=True JIT warmup when PDL is unsupported: 在讨论中被提及，作为解决合并后发现的 AMD MI300 CI 测试失败（语音到文本）所需的相关修复。
- PR #41184 improve(gemma): add fallback for missing kv heads on attention: 在评论中被引用，其引入的 `if name not in params_dict: continue` 改进使得本 PR 可以安全地处理原始检查点中冗余的 `k_norm` 张量（跳过而非报错）。