PR #43356 完整报告

vllm-project/vllm

Add Cosmos3 Reasoner model

合并时间：2026-05-29 00:43

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/43356

执行摘要

- 一句话：新增 Cosmos3 Reasoner 模型支持
- 推荐动作：此 PR 是模型集成的良好范例，展示了如何通过 WeightsMapper 和 secondary_weights 机制快速适配非标准 checkpoint 格式。其中的权重映射模式设计值得学习和参考。推荐在引入其他类似架构（如混合双塔模型）时参考此实现。

功能与动机

根据 PR 描述，Cosmos3 是一个尚未发布的混合 transformer 模型，包含 Reasoner 和 Generator 双塔。Reasoner 塔架构与 Qwen-VL 相同，此 PR 实现从统一的 diffusers 格式检查点到 vLLM 的权重映射，以加载 Reasoner 部分。

实现拆解

1. 配置定义：新增 vllm/transformers_utils/configs/cosmos3.py，定义 Cosmos3Config 继承 Qwen3VLConfig，设置 model_type = "cosmos3_omni"，使 vLLM 能识别该架构。
2. 模型实现：新增 vllm/model_executor/models/cosmos3.py，实现 Cosmos3ForConditionalGeneration 继承 Qwen3VLForConditionalGeneration。关键工作是定义 hf_to_vllm_mapper（WeightsMapper 实例），通过三个子映射完成键名转换：
   - orig_to_new_regex：将 layers.*、embed_tokens.*、norm.* 前缀重写为 language_model.model.*，将 blocks.*、merger.* 等视觉前缀重写为 visual.*，并丢弃 audio_modality_embed 和 action_modality_embed。
   - orig_to_new_substr：替换子字符串，例如 _moe_gen 和 add_*_proj 相关键被删除（对应 Generator 塔权重），同时重命名 to_q→q_proj、to_out→o_proj 等。
   - orig_to_new_prefix：删除 proj_in.、time_embedder. 等 Generator 专用前缀，并将 lm_head. 映射到 language_model.lm_head.。
   - 此外设置 allow_patterns_overrides = ["transformer/*.safetensors"] 以指定主权重位于 transformer/ 子目录。
3. 双权重加载：在 __init__ 中通过 self.secondary_weights 指定从同一模型仓库的 vision_encoder/ 子目录加载视觉编码器权重，使用 DefaultModelLoader.Source 精确控制路径。
4. 加载器增强：修改 vllm/model_executor/model_loader/default_loader.py 中的 _prepare_weights 函数，将 pattern == "*.safetensors" 改为

pattern.endswith(".safetensors")，以允许子目录内的 glob 模式匹配安全张量文件。

5. 注册与文档：在 vllm/model_executor/models/registry.py 中注册 Cosmos3ForConditionalGeneration 的映射；更新 vllm/transformers_utils/configs/__init__.py 和 vllm/transformers_utils/config.py 以导入新配置；在 docs/models/supported_models.md 中添加一行。
6. 测试覆盖：
   - tests/models/multimodal/test_mapping.py：新增 test_cosmos3_new_checkpoint_weights_mapper 函数，验证 WeightsMapper 正确转换代表性键名，并确认所有 Generator 塔键被丢弃（返回空列表）。
   - tests/models/multimodal/generation/test_common.py：在 VLMTestInfo 中注册 "cosmos3" 条目，使用 nvidia/Cosmos3-Nano 模型，设置支持 image、multi-image 和 video 输入类型，以及相应的 prompt 格式和预处理后处理钩子。
   - tests/models/registry.py：添加模型条目并标记 is_available_online=False，避免 CI 因模型未公开而失败。

关键文件：

- vllm/model_executor/models/cosmos3.py（模块 模型定义；类别 source；类型 data-contract；符号 Cosmos3ForConditionalGeneration, init）：核心模型实现，定义了权重映射逻辑和 secondary_weights，是 PR 主体。
- vllm/transformers_utils/configs/cosmos3.py（模块 配置；类别 source；类型 core-logic；符号 Cosmos3Config）：定义 Cosmos3Config，使 vLLM 能识别该架构。
- tests/models/multimodal/test_mapping.py（模块 测试；类别 test；类型 test-coverage；符号 test_cosmos3_new_checkpoint_weights_mapper）：包含权重映射单元测试，验证关键映射正确性。
- vllm/model_executor/model_loader/default_loader.py（模块 加载器；类别 source；类型 data-contract）：修改加载器允许子目录 safetensors 匹配，支持主权重在 transformer/ 目录下。
- tests/models/multimodal/generation/test_common.py（模块 测试；类别 test；类型 test-coverage）：注册 Cosmos3 端到端测试配置，集成到测试套件。
- vllm/model_executor/models/registry.py（模块 注册中心；类别 source；类型 data-contract）：注册模型架构到模型工厂，使 vLLM 能够实例化 Cosmos3。

关键符号：Cosmos3ForConditionalGeneration.init, test_cosmos3_new_checkpoint_weights_mapper

关键源码片段

vllm/model_executor/models/cosmos3.py

核心模型实现，定义了权重映射逻辑和 secondary_weights，是 PR 主体。

```
# SPDX-License-Identifier: Apache-2.0
# SPDX-FileCopyrightText: Copyright contributors to the vLLM project

import regex
```

```
from vllm.config import VllmConfig
from vllm.model_executor.model_loader.default_loader import DefaultModelLoader
from vllm.model_executor.models.qwen3_vl import Qwen3VLForConditionalGeneration
from vllm.model_executor.models.utils import WeightsMapper


class Cosmos3ForConditionalGeneration(Qwen3VLForConditionalGeneration):
    # Cosmos3 统一检查点以一种扁平键布局存储 Qwen3-VL 理解塔和生成塔的权重。
    # 此映射器丢弃生成塔权重，并将理解塔键重写为 Qwen3VLForConditionalGeneration
    # 期望的嵌套形式。
    hf_to_vllm_mapper = WeightsMapper(
        orig_to_new_regex={
            # 语言模型部分：layers.* → language_model.model.layers.*
            regex.compile(
                r"^(layers\.|embed_tokens\.|norm\.)(.+)$"
            ): r"language_model.model.\1\2",
            # 视觉部分：blocks.* → visual.blocks.*
            regex.compile(
                r"^(blocks\.|merger\.|patch_embed\.|pos_embed\.|deepstack_merger_list\.)"
            ): r"visual.\1",
            # 丢弃多模态嵌入：音频和动作
            regex.compile(r"^audio_modality_embed(?:\..*)?$"): None,
            regex.compile(r"^action_modality_embed(?:\..*)?$"): None,
        },
        orig_to_new_substr={
            # 删除生成塔专有子字符串
            "_moe_gen": None,
            ".add_q_proj.": None,
            ".add_k_proj.": None,
            ".add_v_proj.": None,
            ".to_add_out.": None,
            ".norm_added_q.": None,
            ".norm_added_k.": None,
            # 重命名注意力投影：to_q → q_proj，等
            ".to_q.": ".q_proj.",
            ".to_k.": ".k_proj.",
            ".to_v.": ".v_proj.",
            ".to_out.": ".o_proj.",
            ".norm_q.": ".q_norm.",
            ".norm_k.": ".k_norm.",
        },
        orig_to_new_prefix={
            # 丢弃生成塔专用前缀
            "proj_in.": None,
            "proj_out.": None,
            "time_embedder.": None,
            "audio_proj_in.": None,
            "audio_proj_out.": None,
            "action_proj_in.": None,
```

```
            "action_proj_out.": None,
            # lm_head 重定向到语言模型内
            "lm_head.": "language_model.lm_head.",
        },
    )

    # 指定主权重位于 checkpoint 的 transformer/ 子目录
    allow_patterns_overrides = ["transformer/*.safetensors"]

    """
    Cosmos3 checkpoint 将 transformer 权重和 vision_encoder 权重分离到不同目录，
    采用了 diffusers 格式。这里使用 secondary_weights 加载 Reasoner 部分所需的全部权重。
    """

    def __init__(self, *, vllm_config: VllmConfig, prefix: str = "") -> None:
        super().__init__(vllm_config=vllm_config, prefix=prefix)
        # 从 vision_encoder 子目录加载视觉编码器权重作为次要权重
        self.secondary_weights = [
            DefaultModelLoader.Source(
                model_or_path=vllm_config.model_config.model,
                revision=vllm_config.model_config.revision,
                prefix="",
                allow_patterns_overrides=["vision_encoder/*.safetensors"],
            ),
        ]
```

vllm/transformers_utils/configs/cosmos3.py

定义 Cosmos3Config，使 vLLM 能识别该架构。

```
# SPDX-License-Identifier: Apache-2.0
# SPDX-FileCopyrightText: Copyright contributors to the vLLM project

# 继承 Qwen3VL 配置以实现快速集成
from transformers.models.qwen3_vl.configuration_qwen3_vl import Qwen3VLConfig


class Cosmos3Config(Qwen3VLConfig):
    model_type = "cosmos3_omni" # 用于模型注册的标识符
```

tests/models/multimodal/test_mapping.py

包含权重映射单元测试，验证关键映射正确性。

```
def test_cosmos3_new_checkpoint_weights_mapper():
    # 导入 Cosmos3 模型类以获取映射器
    from vllm.model_executor.models.cosmos3 import Cosmos3ForConditionalGeneration

    mapper = Cosmos3ForConditionalGeneration.hf_to_vllm_mapper

    # 验证语言模型与视觉键的转换
```

```
assert mapper.apply_list(
    [
        "layers.0.self_attn.to_q.weight",
        "layers.0.self_attn.to_k.weight",
        "layers.0.self_attn.to_v.weight",
        "layers.0.self_attn.to_out.weight",
        "layers.0.self_attn.norm_q.weight",
        "layers.0.self_attn.norm_k.weight",
        "embed_tokens.weight",
        "norm.weight",
        "lm_head.weight",
        "blocks.0.attn.qkv.weight",
    ]
) == [
    "language_model.model.layers.0.self_attn.q_proj.weight",
    "language_model.model.layers.0.self_attn.k_proj.weight",
    "language_model.model.layers.0.self_attn.v_proj.weight",
    "language_model.model.layers.0.self_attn.o_proj.weight",
    "language_model.model.layers.0.self_attn.q_norm.weight",
    "language_model.model.layers.0.self_attn.k_norm.weight",
    "language_model.model.embed_tokens.weight",
    "language_model.model.norm.weight",
    "language_model.lm_head.weight",
    "visual.blocks.0.attn.qkv.weight",
]

# 验证生成塔相关键被丢弃（返回空列表）
assert (
    mapper.apply_list(
        [
            "layers.0.self_attn.add_q_proj.weight",
            "layers.0.self_attn.add_k_proj.weight",
            "layers.0.self_attn.add_v_proj.weight",
            "layers.0.self_attn.to_add_out.weight",
            "layers.0.self_attn.norm_added_q.weight",
            "layers.0.self_attn.norm_added_k.weight",
            "layers.0.self_attn.q_proj_moe_gen.weight",
            "layers.0.mlp_moe_gen.gate_up_proj.weight",
            "norm_moe_gen.weight",
            "proj_in.weight",
            "proj_out.weight",
            "time_embedder.linear_1.weight",
            "audio_proj_in.weight",
            "audio_proj_out.weight",
            "action_proj_in.weight",
            "action_proj_out.weight",
            "audio_modality_embed",
            "action_modality_embed",
        ]
```

```
        )
        == []
    )
```

评论区精华

- 权重映射路径的正确性：gemini-code-assist[bot] 质疑 regex 替换目标路径不正确，但作者澄清 WeightsMapper 会依次应用 regex、substr 和 prefix 映射，最终路径正确，测试通过。
- 避免全局配置副作用：早期版本修改 vllm_config.load_config.load_format 导致副作用，作者根据反馈改为硬编码 allow_patterns_overrides。
- 子目录加载支持：Isotr0py 建议使用 subfolder 参数简化 secondary_weights 加载，但作者说明主权重同样需要子目录，因此保留对 default_loader.py 的修改。
- 测试可用性：Isotr0py 建议在 registry 测试中设置 is_available_online=False 以避免 CI 失败，作者采纳。
  - 权重映射 regex 路径是否正确 (correctness): 作者解释并坚持原实现，审核人不再质疑，状态 resolved。
  - 避免全局配置副作用 (design): 作者接受建议并修改，删除了全局副作用。
  - allow_patterns 安全风险 (security): 作者硬编码了 allow_patterns_overrides，避免了风险。
  - 测试可用性处理 (testing): 接受建议并修改。
  - secondary_weights 使用 subfolder (design): 双方达成一致，维持当前实现，追加子目录支持。

风险与影响

- 风险：
  - 权重映射正确性：映射规则遗漏或错误可能导致加载失败或静默精度下降。当前单元测试覆盖了主要场景，降低了风险。
  - 加载器模式变更：将 _prepare_weights 从精确匹配改为后缀匹配，可能影响其他使用自定义 allow_patterns 的模型，但影响范围有限（仅在自定义 override 路径时触发）。
  - 模型未公开：测试依赖 nvidia/Cosmos3-Nano 仓库，在模型公开前 CI 无法运行端到端测试，仅依赖单元测试。
- 影响：
  - 用户影响：用户可加载 Cosmos3 模型进行推理，但模型尚未公开发布；新增模型不影响现有模型使用。
  - 系统影响：无性能或安全回归风险；新代码集中在特定模型类，与核心调度、缓存等模块解耦。
  - 团队影响：维护成本低，因 Cosmos3 基于 Qwen3VL，未来底座更新时可自动受益。
  - 风险标记：模型未公开，加载器模式变更可能影响其他模型

关联脉络

- 暂无明显关联 PR