PR #48796 完整报告

vllm-project/vllm

[Core] Keep attention backends eligible for text-only serving of prefix-LM models

合并时间：2026-07-25 18:06

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/48796

执行摘要

- 一句话：修复 prefix-LM 纯文本部署时 attention backend 被错误排除
- 推荐动作：该 PR 设计清晰、测试完善，建议合并。值得关注其缓存设计和对 Registry 的依赖，可作为类似配置动态清除的参考模式。

功能与动机

Prefix-LM multimodal models set is_mm_prefix_lm=True from static model configuration... That constraint is correct while vision inputs can still appear. The problem is that it is applied unconditionally — even when the deployment is configured so that vision inputs can never occur. In that case the gate rejects otherwise-valid attention backends for no reason. (引自 PR body)

实现拆解

1. 在 ModelConfig 中添加 _supports_multimodal_for_mm_prefix 方法(vllm/config/model.py)：该方法通过 MULTIMODAL_REGISTRY.supports_multimodal_inputs 查询部署是否可能接收多模态输入。结果会被缓存以避免重复查询，并确保在 with_hf_config 深拷贝时保持正确决策。在 multimodal_config 尚未初始化时安全返回 True。
2. 在 get_model_arch_config 中集成判断(vllm/config/model.py)：在每次生成架构配置时调用上述方法，将结果通过新参数 supports_multimodal 传递给转换器。同时，在 __post_init__ 中创建 multimodal_config后立即重新获取 model_arch_config，确保清除逻辑生效（并缓存结果供后续 with_hf_config 使用）。
3. 修改 ModelArchConfigConvertorBase(vllm/transformers_utils/model_arch_config_convertor.py)：为 is_mm_prefix_lm 和 convert 方法添加 supports_multimodal 参数。当该参数为 False 时，is_mm_prefix_lm 返回 False 从而清除标志。
4. 更新子类转换器(vllm/transformers_utils/model_arch_config_convertor.py)：Gemma4ModelArchConfigConvertor 等子类同样重写 is_mm_prefix_lm 以支持新参数，并在非多模态时返回 False。
5. 完善单元测试(tests/config/test_multimodal_config.py)：新增多个测试函数，覆盖 registry 调用、无多模态配置时、language_model_only 禁用、转换器清除、以及缓存粘性等场景。

6. 适配模型测试辅助(tests/models/utils.py)：在 DummyConfig 中添加 _supports_multimodal_for_mm_prefix 桩方法，确保架构转换测试仍按默认多模态路径执行。

关键文件：

- vllm/config/model.py（模块 模型配置；类别 source；类型 data-contract；符号 _supports_multimodal_for_mm_prefix）：核心变更文件。新增 _supports_multimodal_for_mm_prefix 方法，修改 get_model_arch_config 以传递多模态支持状态，并在 __post_init__ 中重新获取架构配置。
- vllm/transformers_utils/model_arch_config_convertor.py（模块 架构转换器；类别 source；类型 data-contract；符号 is_mm_prefix_lm, convert）：修改 is_mm_prefix_lm 和 convert 方法以接受 supports_multimodal 参数，在非多模态部署时正确清除 is_mm_prefix_lm。同时更新 Gemma4ModelArchConfigConvertor 子类。
- tests/config/test_multimodal_config.py（模块 Config 测试；类别 test；类型 test-coverage；符号 _make_mm_prefix_model_config, test_supports_multimodal_for_mm_prefix_uses_registry, test_supports_multimodal_for_mm_prefix_before_multimodal_config, test_language_model_only_disables_via_supports_multimodal_inputs）：新增多个单元测试验证 mm_prefix 清除逻辑，包括 registry 调用、无多模态配置时、language_model_only 禁用、转换器清除以及缓存粘性等场景。
- tests/models/utils.py（模块 测试工具；类别 test；类型 test-coverage；符号 _supports_multimodal_for_mm_prefix）：在 DummyConfig 中添加 _supports_multimodal_for_mm_prefix 桩方法，确保架构转换测试路径使用默认多模态模式。

关键符号：_supports_multimodal_for_mm_prefix, is_mm_prefix_lm, convert, _make_mm_prefix_model_config, test_supports_multimodal_for_mm_prefix_uses_registry, test_supports_multimodal_for_mm_prefix_before_multimodal_config, test_language_model_only_disables_via_supports_multimodal_inputs, test_convertor_clears_mm_prefix_when_multimodal_disabled, test_sticky_cache_survives_text_subconfig_regeneration, get_model_arch_config

关键源码片段

vllm/config/model.py

核心变更文件。新增 _supports_multimodal_for_mm_prefix 方法，修改 get_model_arch_config 以传递多模态支持状态，并在 __post_init__ 中重新获取架构配置。

```
# vllm/config/model.py

def _supports_multimodal_for_mm_prefix(self) -> bool:
    """Whether multimodal inputs can still appear for this deployment.

    This runs more than once per config: once early in ``__post_init__``
    (before ``multimodal_config`` exists), again after it is created, and
```

```
        then for every ``get_model_arch_config`` regeneration -- notably
        ``with_hf_config``, which deep-copies this ``ModelConfig`` and swaps
        ``hf_config`` for a text-only submodule (e.g. ``Gemma4ForCausalLM``).

        The result is cached for correctness, not just to save work: on the
        ``with_hf_config`` copy the submodule architecture has no registered
        multimodal processor, so re-querying the registry would raise and be
        treated as text-only, wrongly clearing ``is_mm_prefix_lm`` even when a
        vision modality is still enabled (e.g. ``image=0`` but video allowed).
        The deep-copied cache preserves the top-level decision instead.
        """
        cached = getattr(self, "_supports_multimodal_inputs_cached", None)
        if cached is not None:
            return cached

        if self.multimodal_config is None:
            # Early call before multimodal init — do not clear mm_prefix yet.
            return True

        from vllm.multimodal import MULTIMODAL_REGISTRY

        supports_mm = MULTIMODAL_REGISTRY.supports_multimodal_inputs(self)
        self._supports_multimodal_inputs_cached = supports_mm
        if not supports_mm:
            logger.info_once(
                "Disabled mm_prefix attention mode because multimodal inputs "
                "are configuration-disabled. Attention backends without "
                "mm_prefix support may now be selected."
            )
        return supports_mm

    def get_model_arch_config(self) -> ModelArchitectureConfig:
        convertor_cls = MODEL_ARCH_CONFIG_CONVERTORS.get(
            self.hf_config.model_type, ModelArchConfigConvertorBase
        )
        convertor = convertor_cls(self.hf_config, self.hf_text_config)
        return convertor.convert(
            supports_multimodal=self._supports_multimodal_for_mm_prefix()
        )
```

vllm/transformers_utils/model_arch_config_convertor.py

修改 is_mm_prefix_lm 和 convert 方法以接受 supports_multimodal 参数，在非多模态部署时正确清除 is_mm_prefix_lm。同时更新 Gemma4ModelArchConfigConvertor 子类。

```
    # vllm/transformers_utils/model_arch_config_convertor.py

    def is_mm_prefix_lm(self, supports_multimodal: bool = True) -> bool:
        """Whether to use bidirectional attention for mm positions.
```

```
    ``supports_multimodal`` is False when the deployment is configuration-
    disabled for multimodal inputs (text-only serving). In that case
    mm_prefix is unnecessary and must stay off so attention backends
    without ``supports_mm_prefix()`` remain eligible.
    """
    if not supports_multimodal:
        return False
    if hasattr(self.hf_config, "is_mm_prefix_lm"):
        return bool(self.hf_config.is_mm_prefix_lm)
    # fallback to list of known models
    MM_PREFIX_LM_MODELS = (
        "bagel",
        "gemma3",
        "molmo2",
        "moondream3",
        "paligemma",
        "umm",
    )
    if not hasattr(self.hf_config, "model_type"):
        return False
    return self.hf_config.model_type in MM_PREFIX_LM_MODELS

def convert(self, supports_multimodal: bool = True) -> ModelArchitectureConfig:
    model_arch_config = ModelArchitectureConfig(
        architectures=self.get_architectures(),
        model_type=self.hf_config.model_type,
        text_model_type=getattr(self.hf_text_config, "model_type", None),
        hidden_size=self.get_hidden_size(),
        total_num_hidden_layers=self.get_num_hidden_layers(),
        total_num_attention_heads=self.get_total_num_attention_heads(),
        head_size=self.get_head_size(),
        vocab_size=self.get_vocab_size(),
        total_num_kv_heads=self.get_total_num_kv_heads(),
        num_experts=self.get_num_experts(),
        quantization_config=self.get_quantization_config(),
        is_deepseek_mla=self.is_deepseek_mla(),
        is_mm_prefix_lm=self.is_mm_prefix_lm(supports_multimodal),
        rswa_window=self.rswa_window(),
        derived_max_model_len_and_key=self.derive_max_model_len_and_key(),
    )
    return model_arch_config
```

评论区精华

- 设计位置争议：DarkLight1337 建议将 mm_prefix 清除逻辑放在 get_model_arch_config 内部而非 __post_init__，以避免在标志更新前过早读取 model_arch_config.is_mm_prefix_lm。作者接受并实施。

- 使用 Registry 而非手动选择模态：DarkLight1337 指出，手动枚举模态过于模型特异性，推荐用 MULTIMODAL_REGISTRY.supports_multimodal_inputs 统一判断。作者同意，代码更简洁通用。
- 缓存必要性确认：DarkLight1337 询问 _supports_multimodal_inputs_cached 是否多余。作者说明该方法会被调用多次（__post_init__ 中前后各一次，以及 with_hf_config），缓存对正确性至关重要。
  - mm_prefix 清除逻辑位置设计 (design): 逻辑移动到 get_model_arch_config 中，并通过 _supports_multimodal_for_mm_prefix 传递。
  - 使用 Registry 而非手动选择模态 (design): 采用 registry 方式，代码更简洁通用。
  - 缓存必要性确认 (correctness): 确认缓存有必要，保证 with_hf_config 深拷贝后复用顶层决策。

风险与影响

- 风险：
  - 缓存一致性的风险：如果未来引入需要动态更新多模态支持状态的场景，当前缓存可能导致过时判断。但当前设计已通过生命周期确保正确性。
  - 依赖 MULTIMODAL_REGISTRY：该 API 若未来发生变化可能影响本逻辑。但由于这是统一入口，风险较低。
  - 仅影响 prefix-LM 模型：变更集中在 is_mm_prefix_lm 为 True 的模型，非 prefix-LM 模型完全不受影响。
  - 测试覆盖充分：单元测试覆盖了主要路径，且包含缓存粘性验证，回归风险较低。
- 影响：
  - 用户影响：部署 prefix-LM 模型但仅用于文本推理的用户将自动受益于更丰富的 attention backend 选择（如 FlashInfer），可能显著提升吞吐量并降低 TTFT。用户无需更改配置，但需了解 --language-model-only 或 --limit-mm-per-prompt 设置才能触发清除。
  - 系统影响：注意力后端选择仍固定在引擎启动时，不改变已运行的请求行为。对视觉多模态部署无影响。
  - 团队影响：配置逻辑的可维护性提高，clear 路径集中在 _supports_multimodal_for_mm_prefix 中，未来可轻松扩展。
  - 风险标记：核心路径变更，缓存一致性，registry 依赖

关联脉络

- 暂无明显关联 PR