

PR #42963 完整报告

vllm-project/vllm

[ModelRunnerV2] Support prompt embeds

合并时间: 2026-08-18 05:58

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/42963>

执行摘要

- 一句话: MRV2 支持 prompt embeds, 新增 Triton 叠加 kernel
- 推荐动作: 值得精读, 尤其是负责 MRV2 / worker 侧架构的工程师。关注三个设计决策: 一是“GPU 常驻 + UVA 指针表 + 单 kernel 叠加”的零热路径方案 (PromptEmbedsState.add_request/apply), 它把逐请求 H2D 拷贝降到每请求一次, 并用 num_computed_tokens 与 embeds_len 做 chunked prefill clamp, 是值得复用的模式; 二是“能力标志 + 启动期拒绝”的快速失败设计 (ModelState.supports_prompt_embeds + resolve_model_state_cls 拆分), 避免用户以为特性生效却静默丢 embedding; 三是 PR 讨论中 njhill 与 qthequartermasterman 关于“该不该移植 / 该不该废弃”的交锋, 是理解 vLLM V2 功能取舍标准的典型案例。阅读时建议同时看 tests/v1/worker/test_prompt_embeds_state.py 的 4 个边界用例, 并留意 mm + prompt embeds 组合仍待 follow-up PR 补齐。

功能与动机

在改动之前, V2 model runner 会在配置校验阶段直接拒绝该特性: PR body 给出的失败日志为 `pydantic_core.ValidationError: Value error, VLLM_USE_V2_MODEL_RUNNER does not yet support: prompt embeds, VLLM_USE_V2_MODEL_RUNNER=1` `pytest tests/basic_correctness/test_basic_correctness.py::test_models -k "True-uni or True-mp"` 结果为 8 failed。作者在测试计划中给出 after 态为 8 passed, 并提供 `vllm serve --enable-prompt-embeds` 的 Qwen3-0.6B 冒烟脚本, `prompt_embeds` 与文本路径输出 MATCH。需求侧还有更关键的论证: maintainer njhill 一度质疑“are we sure we want to support this in MRV2? We should not just port over everything blindly”, 用户 qthequartermasterman 回应其业务完全建立在该特性上——用自训练的多模态编码器 (Nemotron 3 Super 的视觉编码器、Llama 音频编码器、Mistral 图编码器) 把图片 / 音频 / 图直接投到文本模型的 token embedding 空间, 若被废弃则只能 fork vLLM。该反馈最终促成特性保留。

实现拆解

1. 解禁与数据契约: `vllm/config/vllm.py` 删除 V2 runner “不支持 prompt embeds” 的启动期拦截 (-3 行); `vllm/v1/core/sched/output.py` 为 `NewRequestData` 新增 `prompt_len` 属性——`prompt_token_ids` 优先, 其次 `prompt_embeds.shape[0]`, 都没有则 0, 使 `GPUModelRunner.add_requests` 能删掉原来的 `assert new_req_data.prompt_token_ids`

is not None, 只在 pooling 路径保留该断言。

2. GPU 侧状态与叠加 kernel (核心) : 新增 `vllm/v1/worker/gpu/model_states/prompt_embeddings.py`。 `PromptEmbedsState.__init__` 维护 `gpu_tensors: dict[req_id, (embeds, mask)]` 保活引用, 以及三张按 `req_state index` 索引的 `UvaBackedTensor (embeds_ptrs / mask_ptrs / embeds_lens)` ; `add_request` 用 `async_tensor_h2d` 做一次性拷贝并写入指针, 无 `embeds` 时必须把 `embeds_lens.np[req_index]` 清零以避免复用 `index` 时读悬垂指针; `apply_staged_writes` 统一下刷 UVA; `apply` 只在 `gpu_tensors` 非空时启动 `_apply_prompt_embeddings_kernel`, `grid` 为 `(num_reqs, cdiv(max_query_len, TOKEN_BLOCK))`。
3. ModelState 契约重构: `interface.py` 把抽象方法 `get_mm_embeddings` 重命名为 `prepare_inputs_embeddings` (统一承载 MM embedding 与 prompt embeds), 并新增 `supports_prompt_embeddings: ClassVar[bool] = False`; `model_states/__init__.py` 把 `init_model_state` 拆成 `resolve_model_state_cls` + 实例化两步, 以便在构造前检查类级能力标志并对不支持的实现抛 `ValueError`; `encoder_decoder.py` 额外显式拒绝; `vllm/model_executor/models/diffusion_gemma.py`、`longcat_flash_ngram.py` 等调用点跟随更名。
4. Model Runner 骨干适配: `model_runner.py` 新增 `uses_inputs_embeddings = supports_mm_inputs or enable_prompt_embeddings`, dummy run 统一走 `model_state.dummy_inputs_embeddings(...)`, `execute_model` 在 `inputs_embeddings` 可用且模型不 `requires_raw_input_tokens` 时把 `input_ids` 置空, LoRA 分支收窄为 `self.supports_mm_inputs and self.lora_config is not None`; `_remove_request` 改为先按 `req_id` 清理 `model_state` 中与索引关联的辅助状态, 规避 review 指出的 `index` 复用竞态; 并补上 `prompt logprobs` 与 `prompt embeds` 互斥的守卫。
5. MM 直通过程与 M-RoPE 过滤: `encoder_runner.prepare_mm_inputs` 对 `modality == "prompt_embeddings"` 的特征不再送入编码器, 而是用 `async_tensor_h2d` 直接写入 `encoder_cache.encoder_outputs`, 让后续 `gather_mm_embeddings` 通过标准 `is_mm_embed` 路径拼接 (HF renderer 会把 `prompt_embeddings` 当成普通 MM 模态混在真实媒体中投递); `DefaultModelState.add_request` 在调用 `rope_state.init_prefill_positions` 前过滤掉 `prompt_embeddings` 模态, 因为它是无 `grid` 信息的直通模态, 而 M-RoPE 假设每个特征都带 `grid`。
6. 测试与验证配套: 新增 `tests/v1/worker/test_prompt_embeddings_state.py` (4 个用例: chunked prefill 与解码边界、跨 `embeds` 末尾的 `clamp`、`is_token_ids` 掩码、`index` 复用残留清理), 扩展 `tests/v1/worker/test_encoder_runner.py` (直通缓存、缓存命中跳过、纯 `embeds` 批次不调用编码器), 并微调 `tests/v1/worker/test_gpu_model_runner.py`; PR body 的端到端验证覆盖 8 个 `basic_correctness` 用例与 HTTP 冒烟。注意 `mm + prompt embeds` 组合仍被列为不支持, 作者表示将在 follow-up PR 中处理。

关键文件:

- `vllm/v1/worker/gpu/model_states/prompt_embeddings.py` (模块 提示嵌入; 类别 source; 类型 core-logic; 符号 `PromptEmbedsState`, `PromptEmbedsState.init`, `PromptEmbedsState.add_request`, `PromptEmbedsState.remove_request`) : 本 PR 的核心新增文件: 定义 `PromptEmbedsState` (UVA 指针表 + 每请求一次 H2D) 与

_apply_prompt_embeds_kernel 叠加 kernel，决定了 prompt embeds 在 MRV2 中的全部运行时语义。

- vllm/v1/worker/gpu/model_states/default.py (模块 模型状态; 类别 source; 类型 core-logic; 符号 supports_prompt_embeds, add_request, remove_request, dummy_inputs_embeds) : DefaultModelState 成为 prompt embeds 的实际接入点: 声明 supports_prompt_embeds、初始化 PromptEmbedsState 与纯文本常驻缓冲区、把 get_mm_embeddings 扩写为统一入口 prepare_inputs_embeds, 并新增无 MM 输入时的 embed_input_ids 分支。
- vllm/v1/worker/gpu/model_states/interface.py (模块 状态接口; 类别 source; 类型 data-contract; 符号 supports_prompt_embeds, prepare_inputs_embeds) : 抽象接口的契约变更点: get_mm_embeddings 更名为 prepare_inputs_embeds, 并新增 supports_prompt_embeds 类级能力标志, 是所有 ModelState 实现与第三方扩展必须跟进的破坏性 API 变化。
- vllm/v1/worker/gpu/model_runner.py (模块 模型运行器; 类别 source; 类型 core-logic; 符号 uses_inputs_embeds, execute_model, add_requests, _remove_request) : Model Runner 骨干适配: uses_inputs_embeds 统一输入形态分支、改用 NewRequestData.prompt_len、删除对 prompt_token_ids 的强断言、修正 _remove_request 的索引清理顺序。
- tests/v1/worker/test_prompt_embeds_state.py (模块 嵌入测试; 类别 test; 类型 test-coverage; 符号 _NewReqData, _Batch, _make_state, _apply) : 覆盖叠加 kernel 的全部关键语义: chunked prefill 偏移、跨 embeds 末尾的 clamp、is_token_ids 掩码与 index 复用后的残留清理, 是评估该特性正确性的主要依据。
- vllm/v1/worker/gpu/mm/encoder_runner.py (模块 编码器; 类别 source; 类型 dependency-wiring; 符号 EncoderRunner.prepare_mm_inputs) : MM 直通路径的关键改动: prompt_embeds 模态不再进入编码器批次, 而是直接写入 encoder_cache, 使后续 gather 能通过标准 is_mm_embed 路径拼接。
- vllm/v1/worker/gpu/model_states/__init__.py (模块 状态工厂; 类别 source; 类型 entrypoint; 符号 init_model_state, resolve_model_state_cls) : 启动期校验入口: 把 init_model_state 拆出 resolve_model_state_cls, 使不支持的 ModelState 实现在构造前即可被拒绝, 是快速失败设计的关键落点。
- vllm/v1/core/sched/output.py (模块 调度输出; 类别 source; 类型 data-contract; 符号 NewRequestData.prompt_len) : 新增 NewRequestData.prompt_len 属性, 为“没有 prompt_token_ids 只有 prompt_embeds”的请求形态建立正式数据契约, 是 worker 侧删除强断言的前提。
- vllm/v1/worker/gpu/model_states/encoder_decoder.py (模块 编解码状态; 类别 source; 类型 data-contract; 符号 prepare_inputs_embeds) : 显式拒绝 encoder-decoder 模型使用 prompt embeds, 并在方法更名中同步, 体现了对不兼容组合的快速失败策略。
- vllm/config/vllm.py (模块 配置校验; 类别 source; 类型 configuration; 符号 VllmConfig) : 特性“解禁”入口: 删除 V2 runner 不支持 prompt embeds 的启动期拦截, 是 8 个 basic_correctness 用例从失败转为通过的直接原因。

- tests/v1/worker/test_encoder_runner.py (模块 编码器测试; 类别 test; 类型 test-coverage; 符号 _model_state, _embeds_item, _pe_feature, test_prepare_mm_inputs_passes_prompt_embeds_through) : 验证 prompt_embeds 作为 MM 模态的直通语义: 混入真实媒体时只把真实媒体送进编码器、已缓存条目不得重复上传、纯 embeds 批次不得触发编码器。

关键符号: PromptEmbedsState.add_request, PromptEmbedsState.remove_request, PromptEmbedsState.apply_staged_writes, PromptEmbedsState.apply, _apply_prompt_embeds_kernel, DefaultModelState.add_request, DefaultModelState.remove_request, DefaultModelState.dummy_inputs_embeds, DefaultModelState.prepare_inputs_embeds, ModelState.prepare_inputs_embeds, init_model_state, resolve_model_state_cls, NewRequestData.prompt_len, EncoderRunner.prepare_mm_inputs, GPUModelRunner.execute_model, GPUModelRunner._remove_request

关键源码片段

vllm/v1/worker/gpu/model_states/default.py

DefaultModelState 成为 prompt embeds 的实际接入点: 声明 supports_prompt_embeds、初始化 PromptEmbedsState 与纯文本常驻缓冲区、把 get_mm_embeddings 扩写为统一入口 prepare_inputs_embeds, 并新增无 MM 输入时的 embed_input_ids 分支。

vllm/v1/worker/gpu/model_states/default.py (重新整理后的关键实现)

```
class DefaultModelState(ModelState):
    # 能力标志: 只有声明为 True 的 ModelState 才允许 --enable-prompt-embeds。
    supports_prompt_embeds = True

    def __init__(self, vllm_config, model, encoder_cache, device):
        super().__init__(vllm_config, model, encoder_cache, device)

        self.prompt_embeds_state: PromptEmbedsState | None = None
        if self.model_config.enable_prompt_embeds:
            self.prompt_embeds_state = PromptEmbedsState(
                self.max_num_reqs, self.inputs_embeds_size, self.dtype, self.device
            )
        if not self.supports_mm_inputs:
            # 纯文本模型也需要一条常驻 inputs_embeds 缓冲区,
            # 作用等价于 encoder_runner.inputs_embeds。
            embeds_buffer_size = (self.max_num_tokens, self.inputs_embeds_size)
            self.inputs_embeds = torch.zeros(
                embeds_buffer_size, dtype=self.dtype, device=self.device
            )
        # ... (rope_state / mm_pruner 初始化保持不变)

    def add_request(self, req_index: int, new_req_data: NewRequestData) -> None:
        if self.rope_state is not None:
            assert new_req_data.prefill_token_ids is not None
```

```

# prompt_embeds 是无 grid 信息的直通模态，而 M-RoPE 假设每个特征
# 都带 grid，因此先把它过滤掉，避免位置编码算错。
mm_features = [
    f for f in new_req_data.mm_features if f.modality != 'prompt_embeds'
]
self.rope_state.init_prefill_positions(
    req_index,
    self.model,
    new_req_data.prefill_token_ids,
    mm_features=mm_features,
)
if self.prompt_embeds_state is not None:
    self.prompt_embeds_state.add_request(req_index, new_req_data)

def remove_request(self, req_id: str) -> None:
    # 按 req_id 清理，避免与 req index 复用发生竞态。
    if self.prompt_embeds_state is not None:
        self.prompt_embeds_state.remove_request(req_id)

def dummy_inputs_embeds(self, num_tokens: int) -> torch.Tensor:
    # dummy run 用的缓冲区，只有形状有意义。
    if self.supports_mm_inputs:
        return self.encoder_runner.inputs_embeds[:num_tokens]
    return self.inputs_embeds[:num_tokens]

def prepare_inputs_embeds(self, scheduled_encoder_inputs, input_batch, req_states):
    # CUDA graph 下 input_batch.input_ids 可能被 padding，这里取未 padding
    # 的前缀，以匹配 is_mm_embed 的 (num_tokens,) 形状。
    input_ids_unpadded = input_batch.input_ids[: input_batch.num_tokens]

    if self.supports_mm_inputs:
        self.execute_mm_encoder(scheduled_encoder_inputs)
        mm_embeds, is_mm_embed = super().gather_mm_embeddings(input_batch)
        if self.mm_pruner is not None and mm_embeds:
            # EVS: 为被裁剪的媒体重算 mrope 位置。
            mm_embeds = self.mm_pruner.recompute(mm_embeds, input_batch, req_states)
            # 必须先 flush staged writes, prepare_inputs() 才能读到新位置。
            self.apply_staged_writes()
        inputs_embeds = self.encoder_runner.get_inputs_embeds(
            input_ids_unpadded, mm_embeds, is_mm_embed
        )
    else:
        # 无 MM 输入时的新分支：直接查 embedding 表。启用 prompt embeds 后
        # model_runner 会把 input_ids 置为 None，模型只能从 inputs_embeds 取输入。
        input_embeddings = self.model.embed_input_ids(input_ids_unpadded)
        self.inputs_embeds[: input_embeddings.shape[0]] = input_embeddings
        inputs_embeds = self.inputs_embeds

    if self.prompt_embeds_state is not None:

```

```

# 在基础 embedding 之上做叠加, chunk/ 掩码语义由 kernel 内部判断。
self.prompt_embeds_state.apply(
    input_batch, req_states.num_computed_tokens.gpu, inputs_embeds
)

```

```

return inputs_embeds[: input_batch.num_tokens_after_padding]

```

vllm/v1/worker/gpu/model_states/__init__.py

启动期校验入口：把 init_model_state 拆出 resolve_model_state_cls，使不支持的 ModelState 实现在构造前即可被拒绝，是快速失败设计的关键落点。

vllm/v1/worker/gpu/model_states/__init__.py (重新整理后的完整片段)

```

def init_model_state(vllm_config, model, encoder_cache, device) -> ModelState:
    cls = resolve_model_state_cls(vllm_config, model)

    # 对会静默忽略 prompt embeds 的 ModelState 在启动时直接报错,
    # 避免用户以为特性生效、实际 embedding 被丢弃。
    if vllm_config.model_config.enable_prompt_embeds and not cls.supports_prompt_embeds:
        raise ValueError(f'--enable-prompt-embeds not supported with {cls.__name__}.')

    return cls(vllm_config, model, encoder_cache, device)

```

```

def resolve_model_state_cls(vllm_config, model) -> type[ModelState]:
    # 把“选类”与“实例化”拆开：调用方可以在真正构造之前
    # 先检查类级别的能力标志 (supports_prompt_embeds) 。
    if hasattr(model, 'get_model_state_cls'):
        return model.get_model_state_cls()

    # Cross-attention 的 encoder-decoder 模型 (Whisper、CohereASR、NemotronParse 等) 。
    if any(isinstance(m, CrossAttention) for m in model.modules()):
        from vllm.v1.worker.gpu.model_states.encoder_decoder import (
            EncoderDecoderModelState,
        )

        return EncoderDecoderModelState

    # Encoder-only attention 是非因果的，不需要 KV cache。
    if any(
        layer.attn_type == AttentionType.ENCODER_ONLY
        for layer in get_layers_from_vllm_config(vllm_config, Attention).values()
    ):
        from vllm.v1.worker.gpu.model_states.encoder_only import EncoderOnlyModelState

        return EncoderOnlyModelState

    if vllm_config.model_config.is_hybrid or vllm_config.model_config.is_attention_free:
        from vllm.v1.worker.gpu.model_states.mamba_hybrid import MambaHybridModelState

```

```
return MambaHybridModelState
```

```
from vllm.v1.worker.gpu.model_states.default import DefaultModelState
```

```
return DefaultModelState
```

评论区精华

1) 是否应该支持（最重要的设计争论）：njhill 质疑「But are we sure we want to support this in MRV2? We should not just port over everything blindly, we would like to deprecate as much as possible.」，并追问功能取舍的判据；作者 gcanlin 回应「Agree. But I'm not sure how to decide whether any feature is needed. Do we have any target or plan for MRV2?」。qthequartermasterman 从业务角度强烈反对废弃：「I for one would be greatly disappointed if prompt_embeds support was deprecated. My business is built on it, and if it were removed, I'd have to fork vLLM to continue doing business.」，并列举自训练视觉 / 音频 / 图编码器投到文本 embedding 空间的用法。njhill 回「that's useful feedback!」，随后亲自 rebase、rework 并合并，事实上的结论是保留该特性。

2) 索引回收顺序的竞态：gemini-code-assist 指出 `_remove_request` 顺序脆弱——`req_states.remove_request(req_id)` 会把 index 归还空闲池，若随后 `model_state.remove_request(req_index)` 之前该 index 被新请求复用，新请求状态会被误清，建议先清理 `model_state`。gcanlin 回复「Fixed.」；从最终 head 看，`PromptEmbedsState.remove_request(req_id)` 改为按 `req_id` 出栈，从设计上消除了 index 复用问题。

3) prompt logprobs 互斥守卫：yewentao256 引用 V1 既有代码 `if request.prompt_token_ids is None: # Prompt logprobs is incompatible with prompt embeddings / continue`，问「We have this in v1, should we add it as well?」，gcanlin 答「Yes. I add the guard before add_request.」。

4) mm + prompt embeds 组合：yewentao256 建议把 `inputs_embeds` 与 `prompt_embeds` 字典放进 `elif` 分支，gcanlin 先是想保留以支持组合，实测 `test_chat_completion_with_mixed_image_embeds.py` 在 V2 下报 `AttributeError: 'DefaultModelState' object has no attribute 'prompt_embeds'`，最终确认「V2 hasn't supported mm + prompt embeds even if not changing this」，将其加入 `unsupported` 列表并承诺 follow-up PR。

5) 方法重命名：gcanlin 提议「Would renaming this method to `prepare_inputs_embeds` be better so that we can have less intrusive code for model runner backbone?」，让一个方法同时承载 mm embed 与 prompt embed，该建议在 njhill 的重构中落地。

6) 流程性反馈：yewentao256 要求补充“具体解决了什么问题”的前后对比（作者补上了 `basic_correctness` 与 `test_deep_sleep` 的 before/after 日志）并要求精简 diff；mergify 多次提示 pre-commit 失败，最终由 njhill 完成 rebase 与简化。

- MRV2 是否应该支持 prompt embeds，还是随其他 V1 特性一起废弃 (design): 保留并支持该特性，njhill 认可业务论证后主导重构并合并；同时确立了“V2 特性取舍需要有明确业务理由”的讨论范式。

- `_remove_request` 中 `req_index` 归还与 `model_state` 清理的顺序竞态 (correctness): 先清理与索引关联的 `model_state` 状态; 最终 `head` 进一步把 `PromptEmbedsState.remove_request` 改为按 `req_id` 出栈, 从设计上消除索引复用问题。
- `prompt logprobs` 与 `prompt embeddings` 的互斥守卫 (correctness): 在 `add_request` 之前补上守卫, 行为与 V1 对齐。
- `mm + prompt embeds` 组合在 V2 中的支持范围 (design): V2 暂不支持 `mm + prompt embeds`, 通过不支持列表快速失败, 作者承诺在 follow-up PR 中补齐。
- `get_mm_embeddings` 是否重命名为 `prepare_inputs_embeddings` (design): 方案被采纳, njhill 的重构中落地为 `ModelState.prepare_inputs_embeddings`, 并让 `default.py` 在其中合流 MM 与纯文本两条路径。
- 补充问题描述、精简 diff 与 pre-commit 失败 (style): 作者补齐了 `basic_correctness` 与 `test_deep_sleep` 的 before/after 日志, 并在 njhill 接手后完成 rebase、重构与精简。

风险与影响

- 风险: 1) 接口重命名的兼容性风险: `ModelState.get_mm_embeddings` → `prepare_inputs_embeddings` 属于跨文件契约变更, 除仓库内 `default.py`、`encoder_decoder.py`、`encoder_only.py`、`diffusion_gemma.py`、`longcat_flash_ngram.py` 外, 任何自定义模型 / 第三方 `ModelState` 子类或通过 `get_model_state_cls()` 注册自有 `state` 的模型都会在升级后立刻报 `AttributeError`/ 抽象方法未实现; 同时 `supports_prompt_embeds` 默认 `False`, 第三方实现若不显式声明, `--enable-prompt-embeds` 将直接抛 `ValueError` (这是有意为之的快速失败, 但需要在 release note 中说明)。
- 2) 新 Triton kernel 的覆盖与正确性: `_apply_prompt_embeds_kernel` 只在 CUDA 上被验证, `tests/v1/worker/test_prompt_embeds_state.py` 顶部用 `pytest.importorskip("triton") + torch.cuda.is_available()` 做模块级跳过, 非 NVIDIA 平台 (ROCm/XPU/CPU CI) 默认不执行; kernel 通过 `tl.pointer_type` 重解释裸 device pointer, `self.dtype` 与 `inputs_embeddings.dtype` 不一致时只有 `assert` 兜底, 属性级错误会退化为静默读错数据或非法访存。
- 3) 指针表生命周期依赖: 指针表指向 `self.gpu_tensors` 中张量的显存, 保活完全依赖该 dict 的引用; `remove_request` 与 `add_request(None)` 路径都只清 dict 与 `embeds_lens`, 若未来有人在别处提前释放或让 `embeds` 张量被复用 (如 pinned memory 池), kernel 会按悬垂指针读取。 `test_index_reuse_clears_stale_entry` 只覆盖了 `index` 复用这一种残留场景。
- 4) 能力组合缺失: `mm + prompt embeds` 在 V2 仍不支持 (讨论中明确记录), 启用后可能报错或需要回退 V1 runner; `encoder_decoder` / 混合模型等路径被硬拒绝, 用户语义上是“降级可用性”。
- 5) 性能与显存影响: 启用 `prompt embeds` 后纯文本模型会常驻一条 (`max_num_tokens`, `inputs_embeddings_size`) 的 `inputs_embeddings` 缓冲区, 并在每步走 `model.embed_input_ids + kernel` 叠加, 相比纯 `input_ids` 路径增加一次 `embedding` 查表与一次 `kernel launch` (`gpu_tensors` 为空时会跳过)。

6) 日志与调试体验: prompt logprobs 与 prompt embeds 互斥, 只会跳过对应请求, 需要在文档中明确, 否则用户会看到“请求了 prompt logprobs 却拿不到”的静默行为。 - 影响: 对用户: 启用 `VLLM_USE_V2_MODEL_RUNNER=1` 且使用 `--enable-prompt-embeds` 的部署不再启动失败, `/v1/completions`、`/v1/chat/completions` 的 `prompt_embeds` 输入在 MRV2 下与 V1 行为对齐 (作者提供的 Qwen3-0.6B 冒烟中 `prompt_embeds` 与文本路径输出完全一致), 依赖自训练多模态编码器的业务场景 (qthequartermasterman 类用户) 得以继续演进而不必 fork。对系统: 改动集中在 v1 worker 与 `model_states` 层, `prepare_inputs_embeds` 成为 mm 与 prompt embeds 的统一入口, 后续任何新增输入形态都会走这条主路径, MRV2 与 V1 的功能差距进一步缩小; `NewRequestData.prompt_len` 让调度输出对“无 token id 请求”正式建模, 消除了 worker 侧的强断言。对团队: `ModelState` 抽象方法更名 + 能力标志是面向模型作者的破坏性契约变更, 内部模型 (`diffusion_gemma`、`longcat_flash_ngram`) 已同步, 但外部扩展需要跟进; 同时该 PR 确立了“V2 特性取舍需有明确业务论证”的协作范式。影响程度评估: 中等偏上, 属于 MRV2 的关键功能补全而非全系统迁移。 - 风险标记: 接口重命名破坏兼容, Triton kernel 仅有 CUDA 覆盖, mm + prompt embeds 组合缺失, 指针表保活依赖引用, 端到端 CI 依赖人工验证

关联脉络

- PR #51700 [2/2][Model Runner V2] FULL CUDA graph capture for microbatched steps (DBO): 同属 MRV2 主干演进, 改动 `vllm/v1/worker/gpu/model_runner.py`、`cuda_graph_utils.py` 与 `vllm/config/vllm.py`; 本 PR 的 `uses_inputs_embeds / dummy_inputs_embeds` 路径与之紧邻, 需确认形状逻辑在 FULL CUDA graph 捕获下仍成立。
- PR #56707 [CI] Initialize warmup registry in V2 QSA runner fixture: 同属 `tests/v1/worker` 下的 V2 model runner 测试体系, 本 PR 新增的 `test_prompt_embeds_state.py` 与 `test_gpu_model_runner.py` 微调共享同一批夹具与 CI 作业。
- PR #56305 [Attention] Add Triton/FlashInfer composite for multimodal prefix attention: 该 PR 建立了 v1 多模态前缀注意力路径; 本 PR 的 `prompt_embeds` 直通 (`encoder_runner` 写入 `encoder_cache`、经 `is_mm_embed` 拼接) 复用了同一套多模态输入语义, 二者共同决定多模态前缀的正确性。
- PR #42168 MTP compatibility with prompt embeds input (讨论中链接的 PR, 标题按讨论描述): qthequartermasterman 在讨论中指出该 MTP + prompt embeds 兼容工作因 MRV2 相关变更被搁置约一个月, 与本 PR 直接构成前后依赖关系。