PR #43606 完整报告

vllm-project/vllm

[Render] Add `/derender` endpoints for disaggregated postprocessing

合并时间：2026-06-13 13:55

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/43606

执行摘要

- 一句话：为 render 服务器添加 /derender 端点完成解聚后处理
- 推荐动作：推荐精读 vllm/entrypoints/serve/render/serving.py 和 vllm/entrypoints/openai/engine/serving.py，理解设计决策：如何在保持 generate server 无 tokenizer 的情况下实现端到端文本响应。特别是 format_token_id_placeholder/resolve_token_id_placeholder 的对称设计值得参考。审查中关于代码合并与解耦的讨论也具有学习价值。

功能与动机

遵循 RFC #42729，解聚服务将 render（预处理）和 generate（推理）分离到不同服务器。generate 服务器为保持轻量和 GPU-only，不携带 tokenizer，因此无法直接输出文本。derender 端点作为后处理，在 render 服务器上闭环，填补了生成输出到标准响应格式的缺失环节。

实现拆解

1. 协议模型（vllm/entrypoints/serve/disagg/protocol.py）：新增 DerenderChatRequest 和 DerenderCompletionRequest，封装 GenerateResponse、可选的 prompt_tokens 计数以及原始请求对象（供未来解析器使用）。DerenderCompletionRequest 利用 pydantic model_validator 校验 prompt_tokens 长度与 generate_responses 一致。
2. 辅助函数提取（vllm/entrypoints/openai/engine/serving.py）：将 format_token_id_placeholder 和 resolve_token_id_placeholder 从 render 服务器私有实现中提取为公共函数。前者将 token ID 格式化为 token_id:N 字符串；后者解码占位符，通过 tokenizer.convert_ids_to_tokens 和 tokenizer.convert_tokens_to_string 正确获取文本，并编码为 UTF-8 字节。同时改造 _get_decoded_token 使用此函数，保持一致性。
3. 核心 derender 逻辑（vllm/entrypoints/serve/render/serving.py）：在 OpenAIServingRender 类中新增 derender_chat_response 和 derender_completion_response 方法。辅助函数 _resolve_logprobs 遍历 logprob 条目解析每个 token 占位符；_convert_chat_logprobs_to_completion_logprobs 适配 Completion 响应 schema；_build_chat_choice 构建单个 choice。主方法从 GenerateResponse 中提取每个 choice，detokenize 得到文本，解析 logprobs，组装完整的 ChatCompletionResponse/CompletionResponse。

4. API 路由（vllm/entrypoints/serve/render/api_router.py）：添加 POST /v1/chat/completions/derender 和 POST /v1/completions/derender，使用相同的 validate_json_request 依赖和错误处理模式，调用 handler 对应方法。
5. 现有 serving 路径适配（vllm/entrypoints/openai/chat_completion/serving.py 和 completion/serving.py）：将硬编码的 f'token_id:{token_id}' 替换为 format_token_id_placeholder 函数调用，统一占位符格式。
6. 测试配套（tests/entrypoints/serve/render/test_derender.py）：新增 488 行测试，覆盖 chat 和 completion 的 roundtrip、usage 计数、logprob 解析、输入验证等场景，使用 RemoteLaunchRenderServer 启动真实 render 服务器进行集成测试。

关键文件：

- vllm/entrypoints/serve/render/serving.py（模块 渲染服务；类别 source；类型 core-logic；符号 _resolve_logprobs, _convert_chat_logprobs_to_completion_logprobs, _build_chat_choice, derender_chat_response）：核心实现：包含 derender_chat_response、derender_completion_response 方法以及辅助函数 _resolve_logprobs、_convert_chat_logprobs_to_completion_logprobs、_build_chat_choice。新增 244 行，是整个 derender 服务的业务逻辑所在。
- tests/entrypoints/serve/render/test_derender.py（模块 集成测试；类别 test；类型 test-coverage；符号 server, client, _render_chat, _make_generate_response）：新增 488 行集成测试，覆盖 chat 和 completion derender 的 roundtrip、usage、logprobs 解析等场景，是验证新功能正确性的核心测试。
- vllm/entrypoints/serve/disagg/protocol.py（模块 协议层；类别 source；类型 data-contract；符号 DerenderChatRequest, DerenderCompletionRequest, _validate_prompt_tokens_length）：定义 DerenderChatRequest 和 DerenderCompletionRequest 数据模型，是 derender API 的协议基础。同时新增模型验证器确保 prompt_tokens 长度一致性。
- vllm/entrypoints/serve/render/api_router.py（模块 路由层；类别 source；类型 entrypoint；符号 derender_chat_completion, derender_completion）：注册两个新 HTTP 路由 /v1/chat/completions/derender 和 /v1/completions/derender，将请求分发到 OpenAIServingRender 的对应方法。
- vllm/entrypoints/openai/engine/serving.py（模块 引擎服务；类别 source；类型 core-logic；符号 format_token_id_placeholder, resolve_token_id_placeholder）：提取 format_token_id_placeholder 和 resolve_token_id_placeholder 作为公共函数，供 render、chat completion、completion 多个模块使用，确保占位符格式统一。
- vllm/entrypoints/openai/chat_completion/serving.py（模块 聊天服务；类别 source；类型 core-logic）：将硬编码的 token_id 占位符字符串替换为 format_token_id_placeholder 函数调用，与 derender 路径保持一致的 placeholder 格式。
- vllm/entrypoints/openai/completion/serving.py（模块 补全服务；类别 source；类型 core-logic）：与 chat_completion/serving.py 类似，将硬编码的 token_id 占位符替换为 format_token_id_placeholder 调用。

关键符号：format_token_id_placeholder, resolve_token_id_placeholder, _resolve_logprobs, _convert_chat_logprobs_to_completion_logprobs,

_build_chat_choice, derender_chat_response, derender_completion_response,derender_chat_completion, derender_completion, _validate_prompt_tokens_length

关键源码片段

vllm/entrypoints/serve/render/serving.py

核心实现：包含 derender_chat_response、derender_completion_response 方法以及辅助函数 _resolve_logprobs、_convert_chat_logprobs_to_completion_logprobs、_build_chat_choice。新增 244 行，是整个 derender 服务的业务逻辑所在。

```
# 逐个解析 ChatCompletionLogProbs 中的 token_id:N 占位符
def _resolve_logprobs(
    logprobs: ChatCompletionLogProbs, tokenizer: TokenizerLike
) -> ChatCompletionLogProbs:
    # 如果没有 content 则直接返回
    if logprobs.content is None:
        return logprobs
    resolved_content = []
    for entry in logprobs.content:
        # 解析当前 token 的占位符，返回真实字符串和 UTF-8 bytes
        token_str, token_bytes = resolve_token_id_placeholder(entry.token, tokenizer)
        # 同时解析 top_logprobs 中每个候选 token
        resolved_top = []
        for top in entry.top_logprobs:
            top_str, top_bytes = resolve_token_id_placeholder(top.token, tokenizer)
            resolved_top.append(
                top.model_copy(update={"token": top_str, "bytes": top_bytes})
            )
        resolved_content.append(
            entry.model_copy(
                update={
                    "token": token_str,
                    "bytes": token_bytes,
                    "top_logprobs": resolved_top,
                }
            )
        )
    return ChatCompletionLogProbs(content=resolved_content)


# 将 ChatLogProbs（per-token 对象）转换为 CompletionLogProbs（并行扁平列表）
def _convert_chat_logprobs_to_completion_logprobs(
    logprobs: ChatCompletionLogProbs,
) -> CompletionLogProbs:
    if logprobs.content is None:
        return CompletionLogProbs()
    tokens: list[str] = []
```

```
    token_logprobs: list[float | None] = []
    top_logprobs_list: list[dict[str, float] | None] = []
    text_offset: list[int] = []
    offset = 0
    for entry in logprobs.content:
        text_offset.append(offset)
        tokens.append(entry.token)
        token_logprobs.append(entry.logprob)
        top_logprobs_list.append(
            {t.token: t.logprob for t in entry.top_logprobs}
            if entry.top_logprobs
            else None
        )
        offset += len(entry.token)
    return CompletionLogProbs(
        text_offset=text_offset,
        token_logprobs=token_logprobs,
        tokens=tokens,
        top_logprobs=top_logprobs_list,
    )
```

vllm/entrypoints/serve/disagg/protocol.py

定义 DerenderChatRequest 和 DerenderCompletionRequest 数据模型，是 derender API 的协议基础。同时新增模型验证器确保 prompt_tokens 长度一致性。

```
class DerenderChatRequest(BaseModel):
    '''Request for the /v1/chat/completions/derender endpoint.

    Wraps a GenerateResponse and caller-supplied metadata needed to produce
    a fully-formed ChatCompletionResponse without a GPU.
    '''
    model: str
    generate_response: GenerateResponse
    prompt_tokens: int | None = None # prompt token 数，用于 usage
    chat_request: ChatCompletionRequest | None = None # 原始请求，供 parser 使用


class DerenderCompletionRequest(BaseModel):
    '''Request for the /v1/completions/derender endpoint.

    Parallel to DerenderChatRequest but handles multi-prompt completions.
    '''
    model: str
    generate_responses: list[GenerateResponse]
    prompt_tokens: list[int] | None = None # 每个 response 的 prompt token 数
    completion_request: CompletionRequest | None = None

    @model_validator(mode='after')
    def _validate_prompt_tokens_length(self) -> 'DerenderCompletionRequest':
```

```
        # 确保 prompt_tokens 长度与 generate_responses 一致
        if self.prompt_tokens is not None and len(self.prompt_tokens) != len(
            self.generate_responses
        ):
            raise ValueError(
                f'prompt_tokens length ({len(self.prompt_tokens)}) must equal '
                f'generate_responses length ({len(self.generate_responses)})'
            )
        return self
```

vllm/entrypoints/openai/engine/serving.py

提取 format_token_id_placeholder 和 resolve_token_id_placeholder 作为公共函数，供 render、chat completion、completion 多个模块使用，确保占位符格式统一。

```
def resolve_token_id_placeholder(
    token: str, tokenizer: TokenizerLike
) -> tuple[str, list[int] | None]:
    '''Decode a 'token_id:N' placeholder back to a token string and UTF-8 bytes.

    Returns (token, None) unchanged if token is not a placeholder.
    This is the inverse of format_token_id_placeholder / _get_decoded_token
    when return_as_token_id=True.
    '''
    # 尝试移除前缀 'token_id:'，若 token 不以前缀开头则返回原值
    suffix = token.removeprefix('token_id:')
    if suffix == token:
        return token, None
    try:
        token_id = int(suffix)
    except ValueError:
        return token, None
    # 通过 tokenizer 获取 token 的内部表示
    token_repr = tokenizer.convert_ids_to_tokens([token_id])[0]
    if token_repr is None:
        logger.warning_once(
            'resolve_token_id_placeholder: token_id %d has no vocab entry; '
            'substituting empty string',
            token_id,
        )
        return '', None
    # 将内部表示转换为真实文本字符串
    token_str = tokenizer.convert_tokens_to_string([token_repr])
    # 编码为 UTF-8 字节序列（errors='replace' 避免非 UTF-8 数据崩溃）
    return token_str, list(token_str.encode('utf-8', errors='replace'))
```

评论区精华

1. token resolution 正确性（gemini-code-assist[bot]）：指出 convert_ids_to_tokens 返回内部表示（如 、Ġ），直接 .encode('utf-8') 产生错误 bytes。作者修正为先用

convert_tokens_to_string 获取真实文本再编码。

2. 代码维护（DarkLight1337）：要求合并 derender 与现有 serving 路径的 logprob 构建逻辑，避免 drift。作者权衡后坚持当前设计：generate 服务器无 tokenizer，协议传递占位符更干净；若改为传递原始 Logprob 数据会增加网络开销并破坏无 tokenizer 假设。审查者接受此解释。
3. 辅助函数风格（DarkLight1337）：建议使用 removeprefix 替代 startswith+ 切片；使用 warning_once 避免重复日志。作者均采纳。
4. 日志等级（DarkLight1337）：提醒 derender 端点的日志可能频繁触发，作者改为 debug 级别。

- token_id 占位符解码正确性 (correctness): 作者采纳并修复为使用 convert_tokens_to_string。
- derender 与现有 serving 代码合并 (design): 维持当前设计，不合并。
- 代码风格：removeprefix 和 warning_once (style): 作者采纳，代码已修改。
- 日志级别过高 (performance): 作者改为 debug。

风险与影响

- 风险：
  1. Token resolution 缺陷：已合并后报告了 U+FFFD 替换字符问题（issue comment #43606#issuecomment-），表明 resolve_token_id_placeholder 在处理非 ASCII token 时可能产生错误字节。aoshen02 正在 PR #45919 中修复。
  2. 同步阻塞：derender 端点是同步且无状态，如果请求量大，可能阻塞 render 服务器的异步事件循环。虽然有 run_in_executor 的潜在改进，但当前未实现。
  3. 代码维护：_resolve_logprobs、_convert_chat_logprobs_to_completion_logprobs 等函数与 chat_completion/serving.py 中的 logprob 构建逻辑有重叠，存在维护不一致的风险。
  4. 协议耦合：DerenderChatRequest 已包含 ChatCompletionRequest 字段但尚未被解析器使用，未来协议调整可能涉及多 PR 联动。 - 影响：对用户：提供了解聚部署场景下完整的前后端分离能力，用户可通过 vllm launch render 同时得到 render 和 derender 端点。对系统：render 服务器现在承担后处理 CPU 负载，但无需 GPU；请求延迟增加一个心跳（detokenize+ 解析）。对团队：新增 488 行测试，维护责任明确；但代码与现有 serving 路径有重叠，需在后续重构中关注统一。 - 风险标记：token resolution 缺陷，同步阻塞，代码维护风险，协议耦合

关联脉络

- PR #42433 [EC Connector] Add EC Transfer Params: 共享 disaggregated serving 架构，EC Connector 负责跨服务器传输，derender 负责后处理，共同构成解聚服务完整流程。
- PR #48102 [Bugfix][KV Offloading] Fix stale transfer_jobs after reset_cache + harden job completion: KV offloading 也属于解聚组件，修复可能影响 derender 的竞态条件，体现同一功能域的维护活动。