PR #43779 完整报告

vllm-project/vllm

[Rust Frontend] Support streaming `generate` endpoint

合并时间：2026-06-02 03:30

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/43779

执行摘要

本 PR 为 Rust 前端 /inference/v1/generate 端点添加了 SSE 流式响应支持。通过新增异步流式生成函数和 SSE 转换层，使端点能够接受 stream=true 参数并逐 token 输出数据块，同时支持 stream_options 中的 usage 统计功能。非流式路径完全保持向后兼容。

功能与动机

之前 Rust 前端的 generate 端点拒绝 stream=true 参数，无法满足需要逐 token 输出的业务场景。为了实现与 Python vLLM 后端的行为一致，并为用户提供更即时的反馈体验，需要添加流式支持。

实现拆解

1. 类型扩展（types.rs）：新增 GenerateResponseStreamChoice 和 GenerateStreamResponse 结构体，映射 Python 侧流式响应；为 GenerateRequest 添加 stream_options 字段。
2. 请求转换（convert.rs）：在 PreparedRequest 中增加 stream、include_usage、include_continuous_usage 字段，并在 prepare_generate_request 中从请求解析，同时实现连续 usage 门控（continuous_usage 仅当 include_usage 为 true 时启用）。
3. 验证调整（validate.rs）：移除 validate_request_compat 中对 stream=true 的拒绝，改为接受流式请求；添加 stream_options 必须配合 stream=true 的一致性检查。
4. 核心流式逻辑（generate.rs）：在 generate 函数中根据 stream 标志分流：流式则调用 generate_chunk_stream 将引擎输出逐 chunk 转换为内部 GenerateStreamResponse，再由 generate_sse_stream 转换为 SSE 事件，最终使用 axum Sse 响应返回；非流式保留原有 collect 逻辑。
5. 集成测试（tests.rs）：新增多个测试覆盖正常流式、usage 包含与连续、空 finish chunk、引擎错误响应等场景。

rust/src/server/src/routes/inference/generate/validate.rs

调整请求验证逻辑，移除 stream=true 的拒绝，添加 stream_options 与 stream 不一致的验证。

以下展示了 validate_request_compat 函数的完整实现及其核心变更点：

```
// 验证函数：在生成前校验请求的合法性。
// 主要变化：删除对 stream=true 的硬拒绝，改为接受流式请求；
```

```
// 同时如果设置了 stream_options 但没有同时设置 stream=true，则返回错误。
pub(super) fn validate_request_compat(
    request: &GenerateRequest,
    served_model_names: &[String],
) -> Result<(), ApiError> {
    // 1. 模型名校验
    if let Some(model) = request.model.as_ref()
        && !served_model_names.iter().any(|n| n == model)
    {
        return Err(ApiError::model_not_found(model.clone()));
    }

    // 2. stream_options 只有在 stream=true 时才允许
    if request.stream_options.is_some() && !request.stream {
        bail_invalid_request!(
            param = "stream_options",
            "stream_options are only supported when stream=true."
        );
    }

    // 3. token_ids 不能为空
    if request.token_ids.is_empty() {
        bail_invalid_request!(
            param = "token_ids",
            "token_ids must contain at least one token ID."
        );
    }

    // 4. max_tokens 必须大于 0
    if request.sampling_params.max_tokens == Some(0) {
        bail_invalid_request!(
            param = "sampling_params",
            "max_tokens must be greater than 0."
        );
    }

    // 5. prompt_logprobs 合法性（非负或 -1）
    if let Some(prompt_logprobs) = request.sampling_params.prompt_logprobs
        && prompt_logprobs < 0
        && prompt_logprobs != -1
    {
        bail_invalid_request!(
            param = "sampling_params",
            "`prompt_logprobs` must be a non-negative value or -1."
        );
    }

    Ok(())
}
```

这一函数是请求进入路由后的第一道关卡，正确实现可确保不合法的流式请求尽早被拒绝。

评论区精华

- Copilot（关于 prompt_tokens 计算）：“prompt_tokens is only populated on the first iteration, but prompt_info may not be present on the first chunk... leading to incorrect usage reporting.” —— 最终通过捕获延迟的 prompt_info 修复。
- Copilot（关于空 token_ids 跳过）：“Chunks with empty token_ids are skipped before emitting, which means a final chunk that only carries a finish_reason will be dropped.” —— 已通过单独发送 finish chunk 解决。
- Codex（关于错误处理）：“This branch should similarly convert FinishReason::Error into a server error SSE event.” —— BugenZhao 引用 completions 路由的已有模式后，最终实现添加了错误分支。
- BugenZhao（对 Copilot 建议的补充）：“Perhaps we can make prompt_tokens an Option so that we don't need to maintain started anymore.” —— 这一设计简化了状态管理。

风险与影响

- 流式错误处理一致性风险（generate.rs）：如果出现罕见的引擎内部错误，但未被正确转为 SSE 错误事件，可能导致客户端误解。当前已通过 FinishReason::Error 分支处理，但仍需关注其他异常路径。
- 连续 usage 统计性能开销（generate.rs）：当启用 included_continuous_usage 时，每个 chunk 都携带 usage 对象，可能增加响应大小和序列化开销。预计可接受，但需要大规模验证。
- 测试覆盖边界不足（tests.rs）：当前测试主要覆盖 happy path，对引擎中途失败、重连等异常场景覆盖不足，可能留下回归风险。

关联脉络

本 PR 是 Rust 前端功能扩展的延续。此前 PR #43481 添加了 InternLM2 工具解析器，#44153 重构了 generative scoring 入口。此次流式支持的加入进一步完善了 Rust 前端的 generate 端点，使其逐步接近 Python vLLM 的功能等价性。未来可能进一步支持 stream_options 的详细控制参数。