PR #46839 完整报告

vllm-project/vllm

[Bugfix][Rust Frontend] Reject prompt_logprobs for streaming generate

合并时间：2026-06-30 13:10

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/46839

执行摘要

该 PR 修复 Rust 前端 /inference/v1/generate 路由的一个验证漏洞：之前流式请求（stream=true）中设置 prompt_logprobs 会被静默忽略，现在返回 400 错误。涉及两个文件，改动量小，测试充分，是常规的路由层修复。

功能与动机

根据 PR 描述，/inference/v1/generate 在接受 stream=true 和 sampling_params.prompt_logprobs > 0 或 -1 时返回 200 并开始 SSE 流，但流式响应形状不含 prompt_logprobs 字段，导致该参数被静默丢弃。这会使客户端误以为 prompt logprobs 已生效，是不符合设计契约的隐式行为。因此需要在该参数组合出现时及时返回错误提示。

实现拆解

1. 核心验证逻辑修改（rust/src/server/src/routes/inference/generate/validate.rs）：在 validate_request_compat 函数中，将原有的 if let 与链（&&）转换为嵌套 if，并新增分支：当 request.stream 为 true 且 prompt_logprobs 被设置（即 Some(_)）时，返回 400 Bad Request。原本的值合法性检查（非负或 -1）保持不变。
2. 单元测试覆盖（同一文件）：新增两个测试函数——validate_request_compat_rejects_streaming_prompt_logprobs 验证流式请求被拒绝（包含 0、1、-1 三种取值），validate_request_compat_accepts_non_stream_prompt_logprobs 验证非流式请求仍然通过。
3. 集成测试补充（rust/src/server/src/routes/tests.rs）：新增 raw_generate_rejects_streaming_prompt_logprobs 测试，直接模拟 HTTP 请求并断言状态码、错误参数和消息内容，确保端到端行为符合预期。

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

核心验证逻辑，新增针对流式与 prompt_logprobs 冲突的拒绝检查，并调整条件判断结构。

```
// 验证生成请求的兼容性
pub(super) fn validate_request_compat(
    request: &GenerateRequest,
    served_model_names: &[String],
) -> Result<(), ApiError> {
    // 检查模型名是否在服务列表中
    if let Some(model) = request.model.as_ref()
```

```
    && !served_model_names.iter().any(|n| n == model)
{
    return Err(ApiError::model_not_found(model.clone()));
}

// stream_options 只能在 stream=true 时设置
if request.stream_options.is_some() && !request.stream {
    bail_invalid_request!(
        param = "stream_options",
        "stream_options are only supported when stream=true."
    );
}

// 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."
    );
}

// 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."
    );
}

// 当 prompt_logprobs 被显式设置时:
if let Some(prompt_logprobs) = request.sampling_params.prompt_logprobs {
    // 检查值是否合法（非负数或 -1）
    if prompt_logprobs < 0 && prompt_logprobs != -1 {
        bail_invalid_request!(
            param = "sampling_params",
            "`prompt_logprobs` must be a non-negative value or -1."
        );
    }

    // 拒绝流式生成时设置 prompt_logprobs，因为流式输出不包含 prompt_logprobs 字段
    if request.stream {
        bail_invalid_request!(
            param = "sampling_params",
            "`prompt_logprobs` are not available when `stream=true`."
        );
    }
}

Ok(())
```

```
}
```

rust/src/server/src/routes/tests.rs

集成测试，验证 HTTP 路由层对非法请求返回 400 状态，确保端到端正确性。

```
#[tokio::test(flavor = "multi_thread", worker_threads = 2)]
#[serial]
async fn raw_generate_rejects_streaming_prompt_logprobs() {
    let mut app = test_app().await;

    // 测试 prompt_logprobs = 0 和 =1 两种情况（ -1 已在单元测试中验证）
    for prompt_logprobs in [0, 1] {
        let response = app
            .call(
                Request::builder()
                    .method("POST")
                    .uri("/inference/v1/generate")
                    .header("content-type", "application/json")
                    .body(Body::from(
                        json!({
                            "model": "Qwen/Qwen1.5-0.5B-Chat",
                            "token_ids": [11, 22],
                            "stream": true,
                            "sampling_params": {
                                "prompt_logprobs": prompt_logprobs
                            }
                        })
                        .to_string(),
                    ))
                    .expect("build request"),
            )
            .await
            .expect("call app");

        // 验证返回 400 错误以及错误信息
        assert_eq!(response.status(), StatusCode::BAD_REQUEST);
        let body = to_bytes(response.into_body(), usize::MAX).await.expect("read body");
        let json: serde_json::Value = serde_json::from_slice(&body).expect("decode json");
        assert_eq!(json["error"]["param"], "sampling_params");
        assert_eq!(
            json["error"]["message"],
            "`prompt_logprobs` are not available when `stream=true`."
        );
    }
}
```

评论区精华

在 review 中，chatgpt-codex-connector[bot] 指出最初的变更未拒绝 prompt_logprobs=0，因为 0 也是显式请求 prompt logprobs，同样不应在流式请求中允许。作者 reidliu41 迅速确认并修复，最终版本拒绝所有 Some(prompt_logprobs) 的流式请求。这个反馈确保了修复的完整性。

风险与影响

- 风险：变更很小，风险低。向后兼容性方面，之前使用流式且传入 prompt_logprobs 的客户端将从静默忽略变为 400 错误，但这是预期行为变更，且更早暴露错误，避免歧义。无性能影响。
- 影响：影响范围限于 Rust 前端的单一路径 /inference/v1/generate。影响程度小，变更清晰，测试覆盖充分。对系统其他部分无影响。

关联脉络

该 PR 是 Rust 前端持续完善参数验证的一部分。近期还有 #46833 等 Rust 前端修复，但本 PR 专注于参数互斥检查，没有直接依赖其他改动。