

PR #53939 完整报告

vllm-project/vllm

[Bugfix][Rust Frontend] Fix LogprobsTensors wire schema mismatch

合并时间: 2026-08-27 07:16

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/53939>

PR #53939 分析: 修复 Rust 前端 LogprobsTensors 线协议字段不匹配

执行摘要

本 PR 修复了 #52242 引入的跨语言线协议回归: Python 侧 `LogprobsTensors` 从 4 个位置字段增至 5 个后, Rust 前端 `WireLogprobs` 仍按 4 元素解码, 导致所有带 `prompt_logprobs` 的请求在 Rust Frontend 上返回 HTTP 500。修复方式是在 Rust 侧为 `WireLogprobs` 追加带 `#[serde(default)]` 的第五字段 `cu_num_generated_tokens_tensor`, 并在 `resolve()` 中显式拒绝非 `None` 值, 同时配套两个 Rust 回归测试与 Python 兼容 fixtures。该方案保留了 #52242 的 DSpark 特性, 是 #53886 回滚方案的更窄替代。

功能与动机

53932 记录了 H200 Rust Frontend OpenAI Coverage 中 `test_prompt_logprobs_e2e_server` 的稳定失败: 请求 `/v1/completions` 返回 HTTP 500, 根因是 `NamedTuple` 序列化为固定长度数组, `LogprobsTensors` 新增 `cu_num_generated_tokens_tensor` 后负载变为 5 元素, 而 Rust `Deserialize_tuple` 解码仍期望 4 个元素。PR body 明确说明选择保留 #52242、只做 Rust 侧窄修复, 并引用了 #53886 自身的建议。

实现拆解

1. 线协议结构体扩字段: 在 `wire.rs` 的 `WireLogprobs` 尾部新增 `#[serde(default)]` `cu_num_generated_tokens_tensor: Option<OpaqueValue>`。 `OpaqueValue` 只承接槽位、不解读内容; `serde(default)` 保证旧 4 字段负载继续可解码。文档中补充了「`NamedTuple` 位置序列化要求 Rust 结构体同步追加尾部字段」的兼容性约定。
2. 语义解码拒绝非 `None`: 在 `logprobs.rs` 的 `WireLogprobs::resolve()` 中新增校验, 非 `None` 的第五字段直接报错; 错误消息刻意不打印字段值, 因为 `opaque` 值可能内嵌完整 `tensor` 数据。

3. 构造路径配套：同一文件中构造 WireLogprobs 的位置补上
cu_num_generated_tokens_tensor: None, 保证 Rust 侧发出的负载也是 5 元素形状。
4. 回归测试：tests.rs 中 prompt-logprobs fixture inline_prompt_logprobs_value 增加尾部 Value::Nil; 新增 decodes_five_field_prompt_logprobs_payload 与 rejects_non_none_cu_num_generated_tokens_tensor 两个用例。
5. Python 兼容 fixtures: python_compat.py 中两个 prompt-logprobs fixtures 增加尾部 None 注释, 与真实负载对齐; new_logprobs fixtures 保持 4 字段 (LogprobsLists 未受 #52242 影响)。

rust/src/engine-core-client/src/protocol/logprobs/wire.rs

核心线协议结构体修改：为 WireLogprobs 增加第五个 #[serde(default)] 字段，是本次修复的关键所在。

```
// SPDX-License-Identifier: Apache-2.0
// SPDX-FileCopyrightText: Copyright contributors to the vLLM project

use serde_tuple::{Deserialize_tuple, Serialize_tuple};

use crate::protocol::OpaqueValue;
use crate::protocol::tensor::WireNdArray;

/// Python 侧的 `LogprobsLists` / `LogprobsTensors` 在解析 aux frame 引用和
/// raw view 之前对应的 wire 表示。
///
/// 原 Python 定义:
/// <https://github.com/vllm-project/vllm/blob/d5cadcee8641a9fcec15facb5a9157d157daa207/vllm/v1/outputs.py#L30-L106>
///
/// # Wire compatibility (线协议兼容性约定)
///
/// Python 的 `LogprobsLists` / `LogprobsTensors` 是 `NamedTuple`：每个字段
/// 都会按位置序列化，而这里的 `Deserialize_tuple` 会在负载比本结构体更长时
/// 直接硬失败。为保持位置兼容，新增 Python 字段时必须在同一变更中为这里的
/// 结构体追加匹配的尾部 `#[serde(default)]` 字段。
#[derive(Debug, Clone, PartialEq, Serialize_tuple, Deserialize_tuple)]
pub struct WireLogprobs {
    /// 形状为 `[num_positions, max_num_logprobs + 1]` 的 wire 数组。
    pub logprob_token_ids: WireNdArray,
    /// 形状为 `[num_positions, max_num_logprobs + 1]` 的 wire 数组。
    pub logprobs: WireNdArray,
    /// 形状为 `[num_positions]` 的 wire 数组。
    ///
    /// Python 侧 sample logprobs 的字段名是 `sampled_token_ranks`，prompt
    /// logprobs 则是 `selected_token_ranks`；Rust 统一用中性字段名，
    /// 因为两种 payload 共用同一 wire 表示。
    pub token_ranks: WireNdArray,
    /// 仅为兼容 batch 级 Python tensor 而保留。
```

```

/// 按调度切分后的 per-request 输出应在此处传 `None`，语义解码器
/// 会拒绝任何其他值。
#[serde(default)]
pub cu_num_generated_tokens: Option<Vec<usize>>,
/// `cu_num_generated_tokens_tensor` 是 #52242 给 `LogprobsTensors`
/// 追加的设备侧生成边界。仅为 wire 兼容保留：`NamedTuple` 即使该槽位
/// 为 `None` 也会编码出来，因此 5 元素 payload 必须有此字段才能解码。
/// per-request 输出应为 `None`，语义解码器会拒绝其他值。
#[serde(default)]
pub cu_num_generated_tokens_tensor: Option<OpaqueValue>,
}

```

rust/src/engine-core-client/src/protocol/logprobs.rs

语义解码路径：`resolve()` 新增对非 `None` 第五字段的 fail-fast 校验，并同步构造路径。

```

/// 将 wire 格式的 logprobs 解析为语义化的 [`Logprobs`] 记录：
/// 查找 aux frames、解码 raw views，并把每一行归并为一个
/// [`PositionLogprobs`]。
fn resolve<Frame>(self, frames: &[Frame], field_prefix: &str) -> Result<Logprobs>
where
    Frame: AsRef<[u8]>,
{
    // batch 级索引 cu_num_generated_tokens 不允许出现在 per-request 输出中
    if let Some(indices) = self.cu_num_generated_tokens {
        bail_ext_value_decode!(
            "{field_prefix}.cu_num_generated_tokens: expected None for per-request engine-
            core logprobs payload, got {indices:?}"
        );
    }

    // #52242 追加的 cu_num_generated_tokens_tensor 同样只能为 `None`。
    // 注意不要像上面那样 Debug 打印值：非 `None` 的不透明值可能内嵌
    // 完整的 tensor 数据块，打印会导致日志爆炸。
    if self.cu_num_generated_tokens_tensor.is_some() {
        bail_ext_value_decode!(
            "{field_prefix}.cu_num_generated_tokens_tensor: expected None for per-request
            engine-core logprobs payload"
        );
    }

    // 通过 aux frame 查找与 raw view 解码，把 wire 数组转成语义记录
    let token_ids = array::decode_array2_u32(
        self.logprob_token_ids,
        &format!("{field_prefix}.logprob_token_ids"),
        frames,
    )?;
    let logprobs = array::decode_array2_f32(
        self.logprobs,
        &format!("{field_prefix}.logprobs"),
    )?;
}

```

```

        frames,
    )?;
    let token_ranks = array::decode_array1_u32(
        self.token_ranks,
        &format!("{field_prefix}.token_ranks"),
        frames,
    )?;

    // 形状一致性校验: token 行 / 列与 logprobs 对齐、token_ranks 长度行数
    if token_ids.rows != logprobs.rows || token_ids.cols != logprobs.cols {
        bail_ext_value_decode!(
            "{field_prefix}: row shape mismatch between token ids ({}, {}) and logprobs ({}, {})",
            token_ids.rows,
            token_ids.cols,
            logprobs.rows,
            logprobs.cols
        );
    }
    if token_ids.rows != token_ranks.len() {
        bail_ext_value_decode!(
            "{field_prefix}: token_ranks length {} does not match row count {}",
            token_ranks.len(),
            token_ids.rows
        );
    }

    // 后续逻辑: 处理空位列表编码 ([0, 0] 或 [0, k + 1]) , 并将每行
    // 分组为 PositionLogprobs, 这里省略。
}

```

评论区精华

- therealnaveenkamal 指出 `python_compat.py` 中 `prompt-logprobs fixtures` 仍是 4 元素, 建议补充尾部 `None` 以匹配 #52242 之后的真实负载:

"bumping them to 5 (trailing None) would make the fixtures match the real payloads that Python sends"

- 作者 jungjiyu 随即更新 fixtures 并运行 pre-commit 确认通过。
- njhill 与 tlrnchlsmith 都通过 `/ci run` 触发 Buildkite CI, 并核准本 PR, 说明修复方案得到维护者认可。

风险与影响

- 跨语言协议同步风险: `#[serde(default)]` 仅能容忍「旧的、更短的负载」, 若 Python 侧未来再新增字段而 Rust 未同步, `Deserialize_tuple` 仍会硬失败。PR 已将同步约定写入 `WireLogprobs` 文档, 后续变更需遵守。
- fail-fast 策略: `resolve()` 对非 `None` 第五字段直接报错, 而当前解码器并不支持该 tensor ; 这是刻意的保守选择, 避免静默丢弃数据。若 DSpark 后续真在 per-request 输出中填充

该字段，Rust 前端需要另行实现解码。

- 测试覆盖：Rust 单元测试（101 个）全部通过，Python 兼容 fixtures 也已对齐；但作者未在本机运行 H200 E2E，端到端恢复依赖 CI 的 `test_prompt_logprobs_e2e_server` 覆盖。
- 范围外事项：dispatcher 的 `decode-error` 处理与 CI 依赖覆盖仍由 #53932 跟踪，不因本 PR 关闭。

关联脉络

本 PR 与 #52242、#53886、#53932 构成一条完整的功能演进链：#52242 引入 DSpark Logprobs 自适应验证特性并埋下跨语言协议隐患；#53932 暴露 CI 失败并完成根因定界（bisect 锁定在 `d5cadcee`）；#53886 以回滚快速解阻塞但明确推荐 Rust 侧窄修复；本 PR 采纳推荐方案，保留特性并补齐兼容与回归测试。对后续跨语言协议演进来说，本 PR 建立了「Python 字段追加 + Rust 尾部 `serde(default)` 字段 + fixtures 同步」的可复用模式。