PR #50746 完整报告

vllm-project/vllm

[Bugfix][Frontend] Reject empty gRPC stop strings

合并时间：2026-08-04 04:50

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/50746

执行摘要

本 PR 修复 Rust 前端 gRPC Generate API 中空 stop 字符串导致解码器 panic 并使整个服务进程 abort 的严重问题。修复采用“共享校验点 + 底层防御”双层策略：在 TextRequest::validate() 统一拒绝空 stop 字符串，同时在 decoded.rs 中对缓冲计算和 stop 匹配做饱和减法与零长度守卫，最后补齐转换层与 unary gRPC 回归测试。该修复消除了 gRPC 路径的进程级崩溃风险。

功能与动机

关联 Issue #50725 描述了具体危害：gRPC Generate / GenerateStream 请求中 stopping.stop_strings 含空字符串时，文本解码路径会 panic；由于 release profile 配置 panic = "abort"，panic 会直接终止整个服务进程，所有并发请求都会受影响。REST 路径已有 validate_stop 返回 400 Bad Request，但 gRPC 路径没有等价校验，且解码器存在两个独立 panic 位点，只修一个不够。本 PR 的目标就是在 gRPC 路径上拦截非法输入，并从底层消除 panic 隐患。

实现拆解

1. 新增错误类型：在 rust/src/text/src/error.rs 中添加 EmptyStopString 变体，并将其纳入 Error::is_request_validation_error()，确保上层能正确映射为 InvalidArgument。
2. 共享校验点：在 rust/src/text/src/request.rs 的 TextRequest::validate() 中检查 stop_strings 是否含空字符串，命中则返回 EmptyStopString。gRPC 转换产物 TextRequest 必然经过该校验，且未来新增入口也会自动获得保护。
3. 解码器防御加固：在 rust/src/text/src/output/decoded.rs 中，将缓冲长度计算的 - 1 改为 saturating_sub(1) 防止下溢；在 matches_stop_string 中跳过长度为 0 的 stop 字符串，避免 windows(0) panic。
4. 回归测试：在 request.rs、decoded.rs 和 grpc/tests.rs 中分别增加了单元与端到端测试，覆盖共享校验、两个 panic 位点以及 unary gRPC 返回 InvalidArgument 的完整路径。

rust/src/text/src/output/decoded.rs

文本解码核心路径，修复两个 panic 位点（缓冲下溢和 windows(0)），是本次修复的根本层。

```
/// 判断输出文本中是否命中 stop 字符串列表中的某一个。
///
/// 返回 `(stop 列表索引，命中的起始字节偏移)`。当多个 stop 字符串
/// 在同一段新生成文本中命中时，选择最先结束的那个，平局按列表顺序。
```

```
fn matches_stop_string(stops: &[String], output: &str, new_bytes: usize) -> Option<(usize, usize)
> {
    // 按字节比较，避免 UTF-8 边界问题
    let output = output.as_bytes();
    let next_off = (output.len() + 1) - new_bytes;
    stops
        .iter()
        .map(|ss| (ss.as_bytes(), ss.len(), next_off.saturating_sub(ss.len())))
        .enumerate()
        .filter_map(|(ss_idx, (ss, len, start_off))| {
            // 关键防御：长度为 0 的 stop 字符串无法匹配，直接跳过；
            // 否则后续 windows(0) 会 panic（在 release 下 panic = abort
            // 会终止整个服务进程）。
            if len == 0 {
                return None;
            }
            output[start_off..]
                .windows(len)
                .position(|w| w == ss)
                .map(|pos| (ss_idx, start_off + pos, start_off + pos + len))
        })
        // min_by_key 保留第一个最小值，因此平局时回退到 stop 列表顺序。
        .min_by_key(|&(_, _, end)| end)
        .map(|(ss_idx, start, _)| (ss_idx, start))
}
```

rust/src/text/src/request.rs

新增共享校验点 TextRequest::validate()，将空 stop 字符串校验集中到所有文本请求的必经关口。

```
impl TextRequest {
    /// 在 tokenization 或 request lowering 之前校验最小不变量。
    pub fn validate(&self) -> Result<()> {
        if matches!(&self.prompt, Prompt::TokenIds(ids) if ids.is_empty()) {
            return Err(Error::EmptyPromptTokenIds {
                request_id: self.request_id.clone(),
            });
        }
        // 共享校验点：所有文本请求（包括 gRPC 转换产物）都会经过这里，
        // 空 stop 字符串在此被拒绝，避免进入解码器触发 panic。
        if self
            .decode_options
            .stop_strings
            .as_ref()
            .is_some_and(|stops| stops.iter().any(String::is_empty))
        {
            return Err(Error::EmptyStopString {
                request_id: self.request_id.clone(),
            });
```

```
        }
        Ok(())
    }
}
```

评论区精华

> njhill：这是第三份相同规则的拷贝……真正的问题在使用点未设防。TextDecodeOptions 是纯公开字段结构体，任何新入口都可能绕过校验；解码器有两个独立 panic 位点，在 panic = "abort" 下都会终止进程。建议无论如何都加固 decoded.rs，并考虑把校验放在共享关口。
>
> zcxGGmu：已按建议调整——将校验移入 TextRequest::validate() 共享校验点，移除 gRPC 本地重复校验，并对解码器做了 saturating_sub 与空字符串守卫加固。

最终 njhill 批准通过。

风险与影响

- 风险：decoded.rs 是文本解码热路径，改动可能影响正常 stop 字符串行为，但正常路径不受影响（仅空串场景差异）；新错误类型若未来入口未纳入 is_request_validation_error，可能返回错误状态码而非 InvalidArgument；行为上，之前能发送但会崩溃的请求现在会被显式拒绝，属于合理的 breaking change。
- 影响：主要影响启用 Rust 前端且开启 grpc_port 的部署。修复后 gRPC 路径不再因空 stop 字符串发生进程级 abort，服务可用性显著提升；共享校验点也降低了未来新增入口时遗漏校验的风险。

关联脉络

本 PR 与 Issue #50725 直接对应。在更广阔的视角上，它与历史 PR #50816（通过 schema 强制 cache_salt 非空）同属“前端入口参数校验强化”方向，体现了 vLLM 正在系统性地收紧各协议入口的输入校验，避免非法参数进入核心引擎路径。