

PR #54303 完整报告

vllm-project/vllm

[Rust Frontend] Bound recursive argument parsers

合并时间: 2026-09-02 03:45

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/54303>

执行摘要

此 PR 为 Rust 解析器中的递归参数解析添加深度限制，防止 Gemma4 和 MiniMax M3 解析器因无界递归导致栈溢出崩溃。通过引入线程本地 RAII 守卫，将最大递归深度设为 128，确保超限时返回错误而非终止进程，显著提升服务健壮性。

功能与动机

修复 Issue #50927 报告的栈溢出崩溃：Gemma4 统一解析器的递归下降语法没有深度限制，导致深度嵌套的模型输出耗尽工作线程栈并 abort 进程。仅需约 6 KB 的生成文本即可触发崩溃，严重影响生产环境稳定性。MiniMax M3 解析器存在类似风险，因此一并修复。

实现拆解

- 递归守卫模块：新增 `rust/src/parser/src/utils/recursion.rs`，实现 `ParserRecursionGuard` 结构体。使用线程本地 `Cell<usize>` 跟踪深度，`enter()` 方法在未超限时递增深度并返回守卫，超限时返回 `Cut` 错误。`Drop` 实现恢复前一层深度，确保 `unwind` 安全。
- 解析器集成：在 `gemma4.rs` 中修改 `gemma4_object` 和 `gemma4_array_value` 函数，在解析 `{[` 后调用守卫；在 `minimax_m3.rs` 中修改 `parameter_element` 函数，在解析元素前调用守卫。守卫的生命周期与递归帧绑定。
- 模块导出：在 `utils.rs` 中添加 `pub(crate) mod recursion;` 使守卫可被其他解析器使用。
- 测试覆盖：为守卫添加单元测试验证深度限制和 `unwind` 恢复；为 Gemma4 和 MiniMax M3 解析器添加测试验证递归限制生效（如 `gemma4_parser_rejects_excessive_recursion_and_recovers_input`）。

`rust/src/parser/src/utils/recursion.rs`

新增核心守卫模块，定义递归深度限制和 RAII 逻辑

```
// 递归深度守卫，使用线程本地存储跟踪当前递归层级
pub(crate) struct ParserRecursionGuard {
    previous_depth: usize,
    // 标记非 Send，确保守卫在创建线程上 drop
    _not_send: PhantomData<Rc<()>>,
}
```

```
impl ParserRecursionGuard {
    // 进入递归帧，超限返回 Cut 错误
```

```

pub(crate) fn enter() -> ModalResult<Self> {
    PARSER_RECURSION_DEPTH.with(|depth| {
        let previous_depth = depth.get();
        if previous_depth >= MAX_PARSER_RECURSION_DEPTH {
            let mut error = ContextError::new();
            error.push(StrContext::Label("parser recursion limit exceeded"));
            return Err(ErrMode::Cut(error));
        }
        depth.set(previous_depth + 1);
        Ok(Self { previous_depth, _not_send: PhantomData })
    })
}

// Drop 时恢复前一层深度
impl Drop for ParserRecursionGuard {
    fn drop(&mut self) {
        PARSER_RECURSION_DEPTH.with(|depth| depth.set(self.previous_depth));
    }
}

```

rust/src/parser/src/unified/gemma4.rs

修改 Gemma4 解析器，在对象和数组解析中集成守卫

```

// 解析嵌套 Gemma4 对象，添加递归深度检查
fn gemma4_object(input: &mut &str) -> ModalResult<Map<String, Value>> {
    literal("{").parse_next(input)?;
    let _guard = ParserRecursionGuard::enter()?; // 进入递归帧
    terminated(gemma4_args, literal("}")).parse_next(input)
}

// 解析 Gemma4 数组值，添加递归深度检查
fn gemma4_array_value(input: &mut &str) -> ModalResult<Vec<Value>> {
    literal("[").parse_next(input)?;
    let _guard = ParserRecursionGuard::enter()?; // 进入递归帧
    terminated(gemma4_array_content, literal("]")).parse_next(input)
}

```

评论区精华

无实质技术讨论，仅有自动化评论和批准。NJ Hill 评论 'Thanks @BugenZhao' 表示认可。

风险与影响

- 兼容性风险：深度限制可能拒绝合法但嵌套极深的输出，但 128 层在实际场景中足够，且错误信息明确提示 'parser recursion limit exceeded'。
- 性能影响：守卫引入线程本地访问和分支，但解析器非热路径，开销可忽略。
- 测试覆盖：已添加单元测试验证边界条件，但未覆盖所有解析路径的集成测试，建议后续补充。

关联脉络

此 PR 与 #47562（工具调用标记截断修复）和 #53870（MCP SDK 兼容性修复）同属工具调用解析器改进线。近期历史 PR 显示工具调用模块持续维护，此修复填补了递归安全漏洞，是模块健壮性提升的一部分。