PR #48937 完整报告

vllm-project/vllm

[Rust][Benchmark] Use `tracing` for logs

合并时间：2026-07-21 13:18

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/48937

执行摘要

本 PR 将 Rust vllm-bench 中的原始 println / eprintln 调用全面替换为 tracing 宏，并统一日志输出到 stderr，使 stdout 只保留纯净的基准测试结果。新增 RowDownloadReporter 优雅处理非 TTY 场景，确保 CI 等环境下仍有定期日志输出。讨论中修复了初始化顺序和独立二进制 subscriber 缺失两个关键问题。属于 Rust 日志基础设施的规范化和可观测性改进。

功能与动机

PR body 明确说明目标：替换 raw println / eprintln 为 tracing 和结构化字段，与 Rust 前端约定一致。同时将日志输出目标改为 stderr，避免 stdout 被日志污染，兼容解析脚本。这一改动提升了日志的可检索性、可过滤性，为未来集成日志采集系统打下基础。

实现拆解

1. 日志基础设施调整(rust/src/cmd/src/logging.rs)：init_tracing 接受 process_label 参数，输出写入 std::io::stderr，移除硬编码常量。
2. 入口初始化(rust/src/cmd/src/main.rs)：先调用 init_tracing 再解析 CLI，通过 std::env::args() 手动匹配子命令以确定标签，保证解析警告不被吞没。
3. 批量替换打印语句（约 18 个文件）：在 benchmark.rs、multi_turn.rs、hf_dataset.rs、speed_bench.rs、tokenizer.rs、tiktoken.rs、config.rs、sharegpt.rs 等中将 println! / eprintln! 替换为 tracing::info! / tracing::warn!，并附带结构化字段，如 host、addresses、prompts、elapsed_seconds 等。
4. 提取 log_failed_requests(rust/src/bench/src/metrics/calculator.rs)：将两个计算函数中重复的失败请求输出逻辑提炼为独立函数，统一使用 tracing::warn!，输出 failed_requests 和每条错误详情。
5. 新增 RowDownloadReporter(rust/src/bench/src/datasets/progress.rs)：封装 indicatif 进度条，当 is_hidden() 为 true（非 TTY）时，以 10 秒为间隔通过 tracing::info! 报告行数。附带单元测试验证时间间隔逻辑。

rust/src/bench/src/datasets/progress.rs

新文件，实现了 RowDownloadReporter 结构体，在非 TTY 下降级为 tracing 日志报告进度，替代之前直输出到 stderr 的 eprint!。

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

```
use std::time::{Duration, Instant};
use indicatif::{ProgressBar, ProgressStyle};

const REPORT_INTERVAL: Duration = Duration::from_secs(10);

/// Reports row download progress to an interactive progress bar, or through
/// periodic tracing events when the progress bar is hidden on a non-TTY.
pub(super) struct RowDownloadReporter {
    progress: ProgressBar,
    next_report: Instant,
}

impl RowDownloadReporter {
    /// Creates a reporter that emits non-TTY updates every 10 seconds.
    pub fn new() -> Self {
        let progress = ProgressBar::new(0);
        progress.set_style(
            ProgressStyle::with_template(
                "{spinner:.green} Fetching rows [{bar:30.cyan/blue}] {pos}/{len}",
            )
            .unwrap()
            .progress_chars("#>-"),
        );
        Self {
            progress,
            next_report: Instant::now() + REPORT_INTERVAL,
        }
    }

    /// Updates the current row count and reports progress when due.
    pub fn update(&mut self, rows: usize, total: u64) {
        let rows = rows as u64;
        let total = total.max(rows);
        self.progress.set_length(total);
        self.progress.set_position(rows);

        if self.should_report(Instant::now()) {
            // 在非 TTY 模式下，通过 tracing 输出结构化日志，避免静默
            tracing::info!(rows, total, "fetching dataset rows");
        }
    }

    /// Clears the interactive progress bar after the download completes.
    pub fn finish(self) {
        self.progress.finish_and_clear();
    }

    // 判断是否应通过 tracing 上报：仅当进度条隐藏（非 TTY）且距上次报告已超 10 秒
    fn should_report(&mut self, now: Instant) -> bool {
```

```
        if !self.progress.is_hidden() || now < self.next_report {
            return false;
        }
        self.next_report = now + REPORT_INTERVAL;
        true
    }
}

#[cfg(test)]
mod tests {
    use super::*;

    #[test]
    fn hidden_reporter_uses_ten_second_deadline() {
        // 验证时间间隔逻辑：9 秒内不报，10 秒内报，19 秒内不报，20 秒报
        let start = Instant::now();
        let mut reporter = RowDownloadReporter {
            progress: ProgressBar::hidden(),
            next_report: start + REPORT_INTERVAL,
        };

        assert!(!reporter.should_report(start + Duration::from_secs(9)));
        assert!(reporter.should_report(start + Duration::from_secs(10)));
        assert!(!reporter.should_report(start + Duration::from_secs(19)));
        assert!(reporter.should_report(start + Duration::from_secs(20)));
    }
}
```

评论区精华

- 初始化顺序：tahsintunan 指出 init_tracing 后移会导致 CLI 解析时的警告丢失。作者回复“Good catch! Reverted to original ordering”并修复。
- 非 TTY 静默：tahsintunan 提醒 Indidatif 隐藏进度条后下载过程会完全静默。作者“Good catch. Extracted a reporter to handle both TTY progress bar and non-TTY fallback logging.” 最终实现了 RowDownloadReporter。
- 独立二进制订阅者缺失：chatgpt-codex-connector[bot] 提到 vllm-bench 独立运行时 tracing 事件被丢弃。作者后续提交修复。

风险与影响

- 日志可见性提升：stderr/stdout 分离，对解析脚本友好，CI 日志更结构化。
- 无功能破坏：仅影响内部日志输出，不涉及 API、模型或调度。
- 非 TTY 降级机制：确保自动化环境不会因进度条隐藏而长时间无输出。
- 兼容性：由于最终输出仍保留在 stdout，使用 vllm-bench 的流程不受影响。

关联脉络

本 PR 是 Rust bench 工具功能堆栈的一部分，前置 PR #48930 引入了基准测试框架。后续可能继续围绕可观测性（如 opentelemetry 集成）和性能分析进行增强。团队内 Rust 前端和 bench 工具的日志规范现已统一。