PR #47787 完整报告

vllm-project/vllm

[Rust Frontend] Stamp `arrival_time` at the frontend entry

合并时间：2026-07-07 11:27

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/47787

执行摘要

本 PR 将 Rust 前端的 arrival_time 标记时机从渲染 / 分词后前移到请求入口处，修复了 TTFT 和 e2e 直方图偏短的问题，使 Rust 前端的时间指标与 Python 前端完全对齐。变更涉及 10 个文件，主要是在 generate_inner 和 chat 方法开头 stamp 时间戳，并在下游透传。

功能与动机

PR body 明确说明 Python stamps it at the renderer entry, before both. So Rust's TTFT and e2e histograms run short. 该变更是为了匹配 Python 前端的指标收集行为。关联 RFC #44757 和 roadmap #44280，是 Rust 前端请求追踪功能的第一步。

实现拆解

1. 可见性提升：在 request_metrics.rs 中将 current_unix_timestamp_secs 的可见性从 pub(crate) 提升为 pub，并删除 request.rs 中的私有实现，改为统一导入。
2. 字段新增：在 TextRequest 结构体中增加 arrival_time: Option<f64> 字段，默认为 None。
3. 入口 stamp：在 text/src/lib.rs 的 generate_inner 方法开头，若未设置 arrival_time 则调用 current_unix_timestamp_secs 填充；在 chat/src/lib.rs 的 chat 方法开头直接计算时间戳并传入。
4. 下游透传：lower.rs 中将 arrival_time 从硬编码 None 改为从 request 直接透传，并添加两项单元测试。
5. 服务层适配：gRPC 和 HTTP 入口点（server/src/ 下的三个 convert.rs）在构建 TextRequest 时传入 arrival_time。

rust/src/text/src/lower.rs

核心变更：在 lower_text_request 中透传 arrival_time，并添加两项测试验证透传和缺失行为

```
/// Convert a high-level TextRequest into GenerateRequest.
pub fn lower_text_request(
    request: TextRequest,
    prompt_token_ids: Vec<u32>,
    sampling_hints: SamplingHints,
    sampling_limits: SamplingLimits,
    tokenizer: &dyn Tokenizer,
) -> Result<PreparedTextRequest> {
    // … 省略前置验证 …
    let generate_request = GenerateRequest {
```

```
            request_id: request.request_id.clone(),
            prompt_token_ids,
            mm_features: request.mm_features.clone(),
            sampling_params: lower_sampling_params(/* ... */)?,
            cache_salt: request.cache_salt.clone(),
            priority: request.priority,
            data_parallel_rank: request.data_parallel_rank,
            reasoning_parser_kwargs: request.reasoning_parser_kwargs.clone(),
            lora_request: request.lora_request.clone(),
            arrival_time: request.arrival_time, // 透传，不再硬编码为 None
            trace_headers: None,
        };
        Ok(PreparedTextRequest { text_request: request, generate_request })
    }

    // 测试：验证透传
    #[test]
    fn lower_text_request_passes_arrival_time_through() {
        let request = TextRequest { arrival_time: Some(42.5), ..sample_request() };
        let prepared = lower_text_request(
            request,
            vec![1, 2, 3],
            sample_sampling_hints(),
            sample_sampling_limits(),
            &stub_tokenizer(),
        )
        .unwrap();
        assert_eq!(prepared.generate_request.arrival_time, Some(42.5));
    }

    // 测试：验证缺失时保持 None
    #[test]
    fn lower_text_request_leaves_arrival_time_unset_when_absent() {
        let request = TextRequest { arrival_time: None, ..sample_request() };
        let prepared = lower_text_request(
            request,
            vec![1, 2, 3],
            sample_sampling_hints(),
            sample_sampling_limits(),
            &stub_tokenizer(),
        )
        .unwrap();
        assert_eq!(prepared.generate_request.arrival_time, None);
    }
```

rust/src/text/src/lib.rs

在 generate_inner 入口处填充默认 arrival_time，实现前移标记

```
async fn generate_inner(
```

```
        &self,
        mut request: TextRequest,
    ) -> Result<(TextRequest, GenerateOutputStream)> {
        request.validate()?;

        // 在 render 和 tokenization 之前标记 arrival_time，以匹配 Python 行为
        if request.arrival_time.is_none() {
            request.arrival_time = Some(vllm_llm::current_unix_timestamp_secs());
        }

        let tokenizer = self.backend.tokenizer();
        let prompt_token_ids = match take(&mut request.prompt) {
            Prompt::Text(text) => tokenizer.encode(&text, request.add_special_tokens)?,
            Prompt::TokenIds(token_ids) => token_ids,
        };
        // ... 后续 lower 和 generate ...
    }
```

评论区精华

无实质讨论，维护者 BugenZhao 直接批准并回复 “Thanks!”。

风险与影响

- 风险：低。TTFT 和 *e2e* 直方图会因包含渲染和分词时间而变长，但这是预期行为。如果监控系统基于旧阈值告警，需要更新基线。
- 影响：所有 Rust 前端请求的时间指标将被修正，与 Python 前端一致；无性能或协议影响。

关联脉络

本 PR 是 RFC #44757（Rust 前端请求追踪）Phase 1 的第一项变更，后续会有更多 PR 引入 OpenTelemetry tracing 支持。