PR #47435 完整报告

vllm-project/vllm

[Rust Frontend] Improve scheduler stats logging parity

合并时间：2026-07-02 22:55

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/47435

执行摘要

此 PR 填补 Rust 前端周期性日志统计与 Python 版的多个差距，新增 deferred 请求数、预占次数、外部 prefix cache 命中率、投机解码吞吐与接受率、以及 MFU 估计值。为支持非 Prometheus 数据类型，引入 SchedulerLogStatsAccumulator 作为内部累积器。同时修复 engine index 始终从 0 硬编码的问题，正确支持混合 DP 场景。

功能与动机

PR 作者指出：先前 Rust 前端的 stats logger 从未直接观测 scheduler stats，仅靠定期抓取全局 Prometheus registry。但 per-position speculation acceptance 和 CUDAGraph sample buckets 等信息无法用 Prometheus 数据表达。因此引入非 Prometheus 的 per-engine SchedulerLogStatsAccumulator，挂载到全局 registry 以便跨 engine 捕获。此外修复 engine index 始终为 0 的问题。

实现拆解

1. 定义 SchedulerLogStatsAccumulator 及关联类型（rust/src/metrics/src/scheduler.rs）：新增 SchedulerLogStatsInterval 存储投机解码草稿数、per-position 接受 token 数、以及 CUDAGraph 采样计数。SchedulerLogStatsAccumulator 使用 Arc<Mutex<SchedulerLogStatsInterval>> 允许并发观测，并提供 observe_spec_decode、observe_cudagraph 和 drain 方法。
2. 扩展记录函数 record_scheduler_stats（rust/src/engine-core-client/src/metrics.rs）：在原有的 Prometheus counter 更新之后，额外调用 log_stats.observe_spec_decode 和 log_stats.observe_cudagraph，将原始调度器数据灌入 accumulator。
3. 扩展 EngineMetrics 和 CounterSnapshot（rust/src/llm/src/log_stats.rs）：新增 external prefix cache 查询 / 命中、预占次数、投机解码相关计数器、MFU 计数器以及 scheduler_deferred gauge。
4. 实现日志计算与格式化函数（rust/src/llm/src/log_stats.rs）：新增 read_deferred_waiting、cache_hit_rate、spec_decoding_log_stats、mfu_log_stats、drain_scheduler_log_stats、format_position_rates 等函数。在 run_stats_logger 的主循环中定期 drain accumulator，计算衍生指标并按 Python 风格输出日志字符串。
5. 修复 engine index 硬编码并改用真实索引（rust/src/engine-core-client/src/client.rs 和 rust/src/llm/src/lib.rs）：新增 engine_indices() 方法；StatsLogger::start 参数从 engine_count 改为 engine_indices，确保混合 DP 下标签正确。

配套改动：CudagraphStat 重命名为 CudagraphStats（protocol/stats.rs），语义更清晰。

rust/src/metrics/src/scheduler.rs

新增 SchedulerLogStatsAccumulator 及相关类型，定义非 Prometheus 累积器基础设施。

```
/// Internal, non-Prometheus accumulator for periodic text logs that need raw
/// scheduler DTOs.
#[derive(Clone, Default)]
pub struct SchedulerLogStatsAccumulator {
    inner: Arc<Mutex<SchedulerLogStatsInterval>>,
}

impl SchedulerLogStatsAccumulator {
    /// Observe spec-decoding fields needed for per-position text-log rates.
    pub fn observe_spec_decode(&self, num_drafts: u64, accepted_tokens_per_pos: &[u64]) {
        // 加锁获取内部 interval，然后累加草稿数和 per-position 接受 token 数
        let mut inner = self.inner.lock().expect("scheduler log stats accumulator poisoned");
        inner.spec_num_drafts += num_drafts;

        // 确保 vector 长度足够，逐位置累加
        if inner.spec_accepted_tokens_per_pos.len() < accepted_tokens_per_pos.len() {
            inner.spec_accepted_tokens_per_pos
                .resize(accepted_tokens_per_pos.len(), 0);
        }
        for (position, accepted_tokens) in accepted_tokens_per_pos.iter().copied().enumerate() {
            inner.spec_accepted_tokens_per_pos[position] += accepted_tokens;
        }
    }

    /// Observe one CUDA graph runtime sample for the interval table.
    pub fn observe_cudagraph(
        &self,
        num_unpadded_tokens: u64,
        num_padded_tokens: u64,
        num_paddings: u64,
        runtime_mode: &str,
    ) {
        // 构建 CudagraphLogKey，在 map 中计数
        let mut inner = self.inner.lock().expect("scheduler log stats accumulator poisoned");
        let key = CudagraphLogKey {
            num_unpadded_tokens,
            num_padded_tokens,
            num_paddings,
            runtime_mode: runtime_mode.to_string(),
        };
        *inner.cudagraph_counts.entry(key).or_default() += 1;
    }

    /// Drain and reset the current text-log interval.
```

```
    pub fn drain(&self) -> SchedulerLogStatsInterval {
        // 用 take 替换出当前 interval，重置为空
        let mut inner = self.inner.lock().expect("scheduler log stats accumulator poisoned");
        std::mem::take(&mut *inner)
    }
}
```

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

新增 engine_indices 方法，修复 engine index 硬编码问题。

```
/// Return the engine-side indices connected to this client.
pub fn engine_indices(&self) -> Vec<u32> {
    // 从每个 engine 的 connection 中提取 engine_index，确保支持混合 DP 下非连续索引
    self.engines
        .iter()
        .map(|engine| {
            engine
                .engine_id
                .engine_index()
                .expect("engine id must encode as u16")
        })
        .collect()
}
```

评论区精华

- Codex（P2）：指出 record_scheduler_stats 调用了 observe_cudagraph，但日志 drain 后未读取 cudagraph_counts，导致 CUDAGraph 样本被丢弃，用户看不到对应日志。
- BugenZhao回复：已知未完成，因担心打印 Markdown 表格过于冗长，后续根据反馈再考虑添加。

风险与影响

- 风险：CUDAGraph 样本记录但未被日志输出（P2）；Mutex 锁粒度较粗但调用频率极低（每 10 秒一次），性能风险可忽略；无新增测试覆盖；engine index 修复可能影响 DP 场景，但方向正确。
- 影响：Rust 前端日志得到显著丰富，与 Python 版保持平价；为非 Prometheus 指标建立可复用的 accumulator 模式；已知 CUDAGraph 日志缺失待后续补充。

关联脉络

此 PR 属于 Rust 前端日志统计增强系列，后续 PR #47444 将在此基础上进一步演进。其他历史相关 PR 包括 #47283（枚举域类型重构）和 #46306（profiling 控制路由），共同提升 Rust 前端的可观测性和可维护性。