

PR #51592 完整报告

vllm-project/vllm

[Rust][Benchmark] Align speed-bench CLI flags with Python and add flag parity test

合并时间: 2026-08-20 16:50

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/51592>

执行摘要

本 PR 修复 `VLLM_USE_RUST_BENCH=1` 委托模式下 Rust `vllm-bench` 与 Python `vllm bench serve` 之间的 CLI 参数漂移: 补齐缺失的 `--speed-bench-output-len` (默认 4096, 与 Python 对齐)、把 `--speed-bench-dataset-subset` 注册为 `--speed-bench-config` 的可见别名, 并移植 Python 侧 #41916 的 `spec-decode /metrics` 解析修复 (按 metric 名匹配、只读 `_total` 计数)。同时新增“Rust 快照比对 + Python 快照生成”的双向一致性测试, 把参数漂移变成任一 CI 必现失败。变更全部收敛在基准测试客户端, 不触及模型服务与推理路径。

功能与动机

`VLLM_USE_RUST_BENCH=1` 委托 (#50081) 会以原始 `argv` 直接 `exec vllm-bench`, 因此 Python 文档化的每个参数都必须在 Rust CLI 中可解析。PR body 指出 `speed-bench` 参数已经漂移:

- `--speed-bench-output-len` 在 Rust 中不存在, 直接报 `unexpected argument`; 即便用户没传, 输出长度也会静默回退到 `--sharegpt-output-len` → `--random-output-len` (默认 128, 而 Python 为 4096), 委托出的基准结果不可比。Slack 上已有用户反馈 `vllm bench serve` 拒绝该参数而 `--help=all` 却列出它。
- 子集参数 Python 拼写为 `--speed-bench-dataset-subset`, Rust 侧却是 `--speed-bench-config`。
- 复查 `spec-decode bench` 路径发现, Rust 的 `/metrics` 解析器早于 Python #41916 的修复, `_created` 时间戳序列与 `label` 值会污染计数, 同类潜在缺陷仍在。

实现拆解

1. CLI 参数对齐 (`rust/src/bench/src/cli.rs`): 为 `BenchServeArgs` 新增 `--speed-bench-output-len` (usize, 默认 4096, 与 Python 默认一致); 把 `--speed-bench-dataset-subset` 挂为 `speed_bench_config` 的 `visible_alias`, 两种拼写在委托下都能解析。
2. 数据流接线 (`rust/src/bench/src/config.rs` + `rust/src/bench/src/benchmark.rs`): `BenchConfig` 新增 `speed_bench_output_len` 字段并在 `from_args` 透传; `run_benchmark` 的 `DatasetName::SpeedBench` 分支改用 `config.speed_bench_output_len` 传入 `load_speed_bench_dataset`, 删除 `sharegpt_output_len.unwrap_or(random_output_len)` 的静默回退, 根除“默认 128 vs 4096”的可比性问题。

3. 移植 spec-decode metrics 解析修复 (`rust/src/bench/src/benchmark.rs`) : 把解析逻辑从 `fetch_spec_decode_metrics` 抽成纯函数 `parse_spec_decode_metrics(text: &str)`: 先取行首 token、在第一个 `{` 处截断得到 metric 名, 再要求以 `_total` 结尾, 最后按 metric 名分派累加。配套 3 个单元测试: 跨 engine 求和、跳过 `_created` 系列与 label 值干扰、无 spec 指标返回 `None`。
4. 双向 CLI 一致性测试 (新增 `rust/src/bench/tests/cli_parity.rs`、`tests/benchmarks/test_rust_bench_cli_parity.py`、`rust/src/bench/tests/python_serve_flags.txt`) : Rust 侧通过 `clap::CommandFactory` 枚举全部 long flag 与别名, 逐项与 Python 快照比对, 未覆盖的 Python flag 必须是 `PYTHON_ONLY` 白名单项 (24 个刻意缺口: `asr/bfcl/blazedit/spec-bench/timed-trace` 数据集、绘图、`self-timed` 等); 另有两条反向测试防白名单“陈旧”与“悬空”。Python 侧 `pytest` 从 `vllm.benchmarks.serve.add_cli_args` 实时收集 flag 与快照比对, 并支持一键再生成。BugenZhao 的追加提交将其挂入 `.buildkite/test_areas/benchmarks.yaml`。
5. 透明化 `--tokenizer-mode` (`rust/src/bench/src/config.rs` + `rust/src/bench/README.md`) : 该参数仅为兼容 Python CLI 而接受, 实际未生效 (此前是死代码字段)。现在非 `auto` 值在配置期触发 `tracing::warn`, 说明真实解析链: `HF tokenizer.json` → `tiktoken` → 服务端 `/tokenize` 回退 (`mistral_common` 不支持本地分词), `README` 同步说明。

rust/src/bench/src/benchmark.rs

核心逻辑改动: 抽出 `parse_spec_decode_metrics` 纯函数并移植 #41916 修复, 同时将 `speed-bench` 输出长度改为使用新参数, 消除静默回退。

```
// rust/src/bench/src/benchmark.rs —— spec-decode 指标解析 (移植 Python 侧 vllm#41916 修复)。
```

```
//
```

```
// 修复要点:
```

```
// 1. 用“花括号前的 metric 名”做匹配, label 值 (如 engine="0") 不再干扰判名;
```

```
// 2. 只读取以 `_total` 结尾的 counter 样本, 跳过 `_created` 时间戳序列,
```

```
// 避免 unix 时间戳被错误累加进计数 (改动前会整行 contains 匹配)。
```

```
/// 解析 Prometheus 文本格式中的 spec decode 计数器。
```

```
///
```

```
/// 返回 `None` 表示未启用 spec decode 或没有可用指标。
```

```
pub(crate) fn parse_spec_decode_metrics(text: &str) -> Option<SpecDecodeMetrics> {
```

```
    let mut num_drafts: u64 = 0;
```

```
    let mut num_draft_tokens: u64 = 0;
```

```
    let mut num_accepted_tokens: u64 = 0;
```

```
    let mut accepted_per_pos: HashMap<u64, u64> = HashMap::new();
```

```
    let mut found_spec_decode = false;
```

```
    for line in text.lines() {
```

```
        let line = line.trim();
```

```
        // 跳过空行与 `# HELP` / `# TYPE` 注释行
```

```
        if line.is_empty() || line.starts_with('#') {
```

```
            continue;
```

```
        }
```

```

// 取行首 token, 并在第一个 `{` 处截断得到纯 metric 名
let first_token = match line.split_whitespace().next() {
    Some(t) => t,
    None => continue,
};
let metric_name = first_token.split('{').next().unwrap_or(first_token);
// 只关心 spec_decode 且以 `_total` 结尾的 counter 系列
if !metric_name.starts_with("vllm:spec_decode") || !metric_name.ends_with("_total") {
    continue;
}
found_spec_decode = true;
let val = match line.split_whitespace().last().and_then(|s| s.parse::<f64>().ok()) {
    Some(v) => v as u64,
    None => continue,
};

// 注意匹配顺序: `num_accepted_tokens_per_pos` 必须在 `num_accepted_tokens` 之前,
// 否则 per-position 系列会被前者截胡
if metric_name.contains("num_drafts") {
    num_drafts += val;
} else if metric_name.contains("num_draft_tokens") {
    num_draft_tokens += val;
} else if metric_name.contains("num_accepted_tokens_per_pos") {
    // position 标签仍需从原始行解析: position="N"
    if let Some(start) = line.find("position=\"") {
        let start = start + "position=\"".len();
        if let Some(end) = line[start..].find("\"")
            && let Ok(pos) = line[start..start + end].parse::<u64>()
        {
            *accepted_per_pos.entry(pos).or_insert(0) += val;
        }
    }
} else if metric_name.contains("num_accepted_tokens") {
    num_accepted_tokens += val;
}
}

if !found_spec_decode {
    return None;
}

Some(SpecDecodeMetrics {
    num_drafts,
    num_draft_tokens,
    num_accepted_tokens,
    accepted_per_pos,
})
}

```

rust/src/bench/src/cli.rs

CLI 入口：新增 `--speed-bench-output-len`（默认 4096）并给 `--speed-bench-config` 加 `--speed-bench-dataset-subset` 可见别名，是委托场景解析的关键。

// rust/src/bench/src/cli.rs —— BenchServeArgs 中与 Python 对齐的 speed-bench 参数定义。

/// SPEED-Bench 数据集配置/切分 (qualitative、throughput_1k、throughput_2k、
/// throughput_8k、throughput_16k、throughput_32k) 。

///

/// Python 侧参数名为 `--speed-bench-dataset-subset`，Rust 侧原名

/// `--speed-bench-config`；通过 `visible_alias` 同时接受两种拼写，

/// 保证 `VLLM_USE_RUST_BENCH=1` 委托时原始 `argv` 能原样解析。

```
#[arg(
    long,
    visible_alias = "speed-bench-dataset-subset",
    default_value = "qualitative"
)]
```

pub speed_bench_config: SpeedBenchConfig,

/// 每个请求的输出 token 数（仅 SPEED-Bench 数据集）。

///

/// 默认 4096 与 Python `speed_bench_output_len` 一致；此前 Rust 侧

/// 缺失该参数，长度会静默回退到 `sharegpt/random` 的 128，

/// 导致委托出的基准结果不可比。

```
#[arg(long, default_value_t = 4096)]
```

pub speed_bench_output_len: usize,

评论区精华

codex (P1, testing)：“当贡献者按文档用裸 `python` 执行快照再生成命令时，可能解析到系统解释器，绕过仓库虚拟环境要求；同一错误命令还会被写进快照头。”—— 已修复：统一改为 `.venv/bin/python`。

codex (P1, style)：AGENTS.md 指令文件“只增不减”违反编辑守则，要求用删减 / 合并抵消新增内容 —— `commit “lean AGENTS.md”` 将说明并入既有 `Build & Test` 段落。

claude[bot] (黄标, documentation)：“`BenchServeArgs::tokenizer_mode` 的 `clap doc` 注释仍写着 `(auto, hf, slow, mistral)`，让 `vllm-bench --help` 呈现为可用参数，而本 PR 实际加了‘非 auto 被忽略’的运行时告警。”—— 已在本 PR 内同步更新 `help` 文案。

BugenZhao (design)：“The parity tests look quite clever! 🧐” —— 批准并合并，还补了 CI 触发条件。

风险与影响

- 默认行为变更：依赖旧回退（128 输出长度）跑 `speed-bench` 的 Rust 用户，升级后默认变为 4096，单请求时长与吞吐口径会变化。这是刻意修复，但对比历史基准时需注意。
- 快照同步负担：Python 侧新增参数后，若快照与白名单未同步更新，`cargo` 或 `pytest` 任一 CI 会失败。团队需接受“每次动参数都要跑一次再生成脚本”的流程成本；BugenZhao 已把

测试挂入 benchmarks CI 兜底。

- metrics 解析兼容面：只认 `_total` 后缀的 counter；若某观测端点用非标准命名暴露 spec 指标，统计会静默变为 `None`。另外 position 标签仍靠整行 `line.find("position=\\")` 解析，label 顺序变化时仍可能漏解析（相比改动前已大幅收敛）。
- 影响范围：仅 `rust/src/bench/` 客户端与 CI 配置；模型服务、Python bench 主路径不受影响。委托路径从“可用但结果不可比”变为“参数与 Python 等价”。

关联脉络

- #50081 (VLLM_USE_RUST_BENCH 委托，标题未在材料中提供)：本 PR 的直接动因——委托把 Python 文档化参数变成 Rust CLI 的硬契约，参数漂移因此从“不痛不痒”升级为“结果不可比”。
- #41916 (Fix spec decode benchmark metrics)：Python 侧修复的源头，本 PR 将其移植到 Rust /metrics 解析器，并补了单元测试。
- #52844 (Rust 前端 n>1 拒绝)：同属“Rust 入口必须与 Python 契约一致”的修复线，表明团队正在系统性地收敛 Rust 与 Python 的行为差异。
 - 结合本仓库近期动态（#53017 修 gumbel 采样 stride、#52839 整合 CP 注意力算子），可以看到 Rust 基准与 spec-decode 采样路径都在被持续加固。