PR #2028 完整报告

radixark/miles

session: collect speculative-decoding counters

合并时间：2026-07-31 14:45

原文链接：http://prhub.com.cn/radixark/miles/pull/2028

执行摘要

- 一句话：补齐 session 路径 spec 计数器采集
- 推荐动作：值得快速精读（约 10-15 分钟）。这是一个教科书式的"双路径重复实现导致指标失真"案例：PR body 对根因、实测数据、修复后效果和数据可用性边界的分析非常扎实，尤其"修完 spec_accept_length 但 spec_accept_rate 仍为 0，因为字段缺失而非 bug"的区分，展示了严谨的排查方法论。后续建议以 # TODO unify with Sample.update_from_meta_info 为线索，推动 session 与直接 rollout 路径的元数据采集逻辑合并，从根本上消除此类分叉。

功能与动机

PR body 明确指出：rollout/spec_accept_rate 和 rollout/spec_accept_length 在 agentic（session/TITO）rollouts 中恒为 0.0，即使 EAGLE 在运行且引擎记录了真实 accept length。根因是 session 路径在 _compute_sample_from_openai_record 中重新派生元数据（上方留有 # TODO unify with Sample.update_from_meta_info），但只拷贝了 prefix_cache_info 和 weight_versions，spec_info 保持零值，最终被 metrics.py 平均成 0。实测数据：16 节点 GLM-5.2 运行中 64 个 dumped samples 全部 spec_info = {0, 0, 0, 0}，而引擎日志显示 accept len: 1.77。

实现拆解

1. 定位根因

对比直接 rollout 路径的 Sample.update_from_meta_info 与 session 路径的 _compute_sample_from_openai_record，发现后者只调用 sample.prefix_cache_info.add(...) 与 weight_versions.append(...)，遗漏 sample.spec_info.add(...)，导致 spec_info 恒为零值被 metrics.py 平均。

2. 修复核心逻辑

在 miles/rollout/session/samples/merge.py 的 _compute_sample_from_openai_record 中，于 prefix_cache_info.add 之前新增条件采集：当 args.sglang_speculative_algorithm 为真时调用 sample.spec_info.add(choice.get("meta_info", {}))，与直接 rollout 路径行为对齐。改动仅 +2 行，属于控制流调整。

3. 测试配套

tests/fast/rollout/session/test_samples.py 顶部的两个 SimpleNamespace fixture（_ARGS 与 _ARGS_RECORDING）补上 sglang_speculative_algorithm=None。这是必需改动而非装饰：新代码直接读取该属性而非 getattr 兜底，缺属性会导致该文件每个用例抛 AttributeError。

4. 验证

本地运行 tests/fast/rollout/session 与 tests/fast/rollout/session/test_lifecycle_metadata.py 共 45 个用例全部通过；16 节点 GLM-5.2 实测 spec_accept_length 从恒 0.0 恢复到 1.86-1.88，且 spec_verify_ct（362549）与 completion_token_num（682896）均能正常获取。

关键文件：

- miles/rollout/session/samples/merge.py（模块 会话采样；类别 source；类型 core-logic；符号 _compute_sample_from_openai_record, Sample.spec_info.add）：修复核心：在 _compute_sample_from_openai_record 中补充 spec_info 采集，使 session 路径与直接 rollout 路径的 Sample.update_from_meta_info 行为对齐，直接决定指标正确性。
- tests/fast/rollout/session/test_samples.py（模块 测试用例；类别 test；类型 test-coverage；符号 _ARGS, _ARGS_RECORDING）：两个 SimpleNamespace fixture 补 sglang_speculative_algorithm=None，防止新属性直读导致整文件用例 AttributeError，是修复的必需测试配套。

关键符号：_compute_sample_from_openai_record

关键源码片段

miles/rollout/session/samples/merge.py

修复核心：在 _compute_sample_from_openai_record 中补充 spec_info 采集，使 session 路径与直接 rollout 路径的 Sample.update_from_meta_info 行为对齐，直接决定指标正确性。

```
# 函数位于 miles/rollout/session/samples/merge.py，是 session（TITO）路径构造 Sample
的收尾逻辑。
# 注意：这段代码与直接 rollout 路径的 `Sample.update_from_meta_info` 高度重复
# （函数上方留有 `# TODO unify with Sample.update_from_meta_info` 注释），
# 此前只复制了 `prefix_cache_info` 与 `weight_versions`，漏掉 `spec_info`，
# 导致 agentic rollout 的 `rollout/spec_accept_length` 恒为 0.0。

match choice["finish_reason"]:
    case "stop" | "tool_calls":
        sample.status = Sample.Status.COMPLETED
    case "length":
        sample.status = Sample.Status.TRUNCATED
    case "abort":
        sample.status = Sample.Status.ABORTED

# 修复点：与 `Sample.update_from_meta_info` 对齐，
# 仅在启用 speculative 算法（如 EAGLE）时采集计数。
# `meta_info` 可能缺失，用 `get` 兜底；
```

```
    # 若不采集，`spec_info` 保持零值，被 `metrics.py` 平均后输出 0.0。
    if args.sglang_speculative_algorithm:
        sample.spec_info.add(choice.get("meta_info", {}))
    sample.prefix_cache_info.add(choice.get("meta_info", {}))
    if "weight_version" in choice["meta_info"]:
        sample.weight_versions.append(choice["meta_info"]["weight_version"])

    return sample
```

tests/fast/rollout/session/test_samples.py

两个 SimpleNamespace fixture 补 sglang_speculative_algorithm=None，防止新属性直读导致整文件用例 AttributeError，是修复的必需测试配套。

```
    # tests/fast/rollout/session/test_samples.py 顶部的全局 fixture。
    # 修复后的生产代码从 `args` 直接读取 `sglang_speculative_algorithm` 属性
    # （而非 `getattr` 兜底），因此这两个 fixture 必须同步补上该属性，
    # 否则本文件所有用例都会在构造样本时抛出 `AttributeError`。

    _ARGS = SimpleNamespace(
        save_debug_trajectory_data=None,
        sglang_speculative_algorithm=None, # 新增：默认关闭 speculative 计数采集
    )

    _ARGS_RECORDING = SimpleNamespace(
        save_debug_trajectory_data="/unused/{rollout_id}.jsonl",
        sglang_speculative_algorithm=None, # 新增：与 `_ARGS` 保持一致
    )
```

评论区精华

review 本身没有实质讨论，guapisolo 直接批准（“LGTM.”）。真正有价值的分析在 PR body 中：

- 作者明确区分“bug”与“数据不可用”两个层面：spec_accept_length = completion_token_num / spec_verify_ct 在修复后可正常计算；而 spec_accept_rate 依赖 spec_accept_token_num 与 spec_draft_token_num，当前 sglang 版本的 meta_info 不提供这两个字段，因此修复后仍为 0.0，属 engine 侧缺失，需要上游改动。
- 实测前后对照：修复前 64 个样本全部 spec_info = {0, 0, 0, 0}，引擎日志 accept len: 1.77；修复后指标为 1.86-1.88，确认根因判断正确。
- 另有一条 gemini-code-assist 机器人的公告评论（Gemini Code Assist 已停服），与本次变更无关。
- spec_accept_rate 修复后仍为 0.0 是否仍是 bug (question): 认定为数据可用性限制而非 bug；补齐这两个字段需要 sglang engine 侧改动，超出本 PR 范围。
- 测试 fixture 补属性是必需改动而非装饰 (testing): 通过测试配套保证修复后的代码在测试环境中不回归，45 个用例全部通过。

风险与影响

- 风险：
  - 属性直读风险：新代码用 args.sglang_speculative_algorithm 直接取属性（替代 getattr 模式），任何未配置该属性的调用方会在构造样本时抛 AttributeError；PR 已同步更新测试 fixture，但需确认生产环境所有调用 compute_samples_from_openai_records 的入口都设置了该属性。
  - 指标口径变化：spec_accept_length 从恒 0 变为真实值，会改变依赖它的下游报表与历史基线对比，纵向比较旧数据时需注意口径不一致。
  - 残留误导风险：spec_accept_rate 继续为 0.0，容易被误读为“EAGLE 无收益”，建议在指标面板或文档标注其为数据不可用（当前 sglang 版本缺字段）而非真实为零。
  - 代码重复风险：本修复延续了 # TODO unify with Sample.update_from_meta_info 指出的双路径重复实现，未来两条路径再次分叉仍可能重演此类 bug，属于未根治的隐患。
- 影响：
  - 影响功能：agentic/session（TITO）rollouts 的 speculative-decoding 指标统计，rollout/spec_accept_length 由错误的恒 0 变为真实值，对模型评估与 EAGLE 收益分析有直接改善。
  - 影响范围：改动仅 2 个文件、净增 6 行，属于低风险小改动；但影响所有 session 路径采样与指标上报。
  - 对团队：为 spec_accept_rate 的上报差距（engine 侧缺字段）留下明确的后续工作项，并暴露 session 与直接 rollout 两条路径需要统一实现的技术债。
  - 风险标记：属性直读新增 AttributeError 风险，指标口径变化影响历史对比，spec_accept_rate 仍为 0 易误读，双路径重复实现未根治

关联脉络

- 暂无明显关联 PR