

PR #2598 完整报告

radixark/miles

fix(metrics): propagate and correctly aggregate speculative decoding counters

合并时间: 2026-09-12 06:10

原文链接: <http://prhub.com.cn/radixark/miles/pull/2598>

执行摘要

- 一句话: 统一 SGLang 投机解码计数并修复跨会话聚合
- 推荐动作: 值得精读。该 PR 展示了跨模块数据契约演进的标准姿势: 类型层 fallback 兼容 → 传输层打通 → 权威聚合 → 驱动校验 → 聚合消费, 每一步都有测试锚定。特别值得借鉴的是 session v2 的『服务端聚合一次 + 按 session_id 去重』模式, 它解决的是树状采样结构下共享节点重复计数的通用问题, 可推广到其他以树 / 图结构回传样本的指标场景。阅读建议聚焦 miles/utils/types.py 的 SpecInfo 与 miles/ray/rollout/metrics.py 的 _compute_spec_metrics 两处实现。

功能与动机

PR body 直接点明两个根因: "The old Miles field names no longer matched the SGLang response contract, and averaging per-request ratios gives each request equal weight regardless of its number of drafts or verification steps. The session samples codec also dropped SpecInfo before it reached rollout metrics." 即旧字段名 (spec_accept_token_num 等) 在 SGLang 新响应中已不存在, 导致计数器读不到值; 逐请求比率平均会稀释大 draft 数请求的贡献; session codec 在指标到达 rollout 前丢弃了 SpecInfo。session v2 还有一个额外边界: 树会物化多片共享生成节点的样本, "Summing leaf-local SpecInfo would count shared nodes more than once", 因此需要服务端发布一份权威聚合、驱动侧按 session id 去重。

实现拆解

1. 统一数据契约: miles/utils/types.py 中 Sample.SpecInfo 字段重命名为 SGLang 规范命名 (spec_num_correct_drafts / spec_num_proposed_drafts / spec_verify_ct / completion_tokens), add() 增加 spec_verify_ct <= 0 守卫 (未经历投机验证的普通生成不计数), to_dict() / from_dict() 成对更新且 from_dict() 保留旧字段 fallback, 历史序列化样本仍可读取。配套同步 miles/rollout/generate_utils/sample_utils.py 与 miles/utils/test_utils/mock_sglang_server.py 的字段读取点。
2. 打通 session v1/v2 共享 wire: miles/rollout/session/samples/codec.py 在共享 samples wire 上序列化 / 反序列化 SpecInfo, 此前 session 服务端已汇总计数但 codec 在编码时丢弃, 导致驱动侧解出来全是 0。分别由 tests/fast/router/test_session_samples_op.py 与 test_session_samples_op_v2.py 锚定 v1/v2 的精确计数器往返。

3. session v2 权威聚合一次：新增 `miles/rollout/session/v2/metrics.py`, `build_session_rollout_metrics(session_id, nodes)` 对所有已提交生成节点（`session.tree.nodes`）汇总一份 `SpecInfo`；`SessionCoreV2.collect_samples` 在投机解码配置开启时、hooks 处理完 metadata 之后的 wire 边界注入 `SESSION_ROLLOUT_METRICS_KEY`。测试覆盖共享前缀树与 `max_seq_len` 截断排除节点仍计入聚合。
4. 驱动侧校验与替换：`miles/rollout/generate_hub/agentive_tool_call.py` 的 `generate` 在开启投机解码且 session v2 时：成功收集后校验 carrier 的 `session_id` 与 `tracer.session_id` 一致、`metrics` 非空，再以权威 carrier 覆盖每片叶子 metadata（清除旧叶子携带的陈旧 carrier）；收集失败或空回复返回 ABORTED 时移除陈旧 carrier，避免悬空归属。
5. 聚合层重写与配套：`miles/ray/rollout/metrics.py` 的 `_compute_spec_metrics` 重写为：无 carrier 的普通样本（如训练中经 `generate_function_path` 覆盖到 `single_turn` 生成器的样本）直接贡献本地 `spec_info`；有 carrier 的按 `session_id` 去重（同一 session 多片叶子只取一次）；所有计数先求和再算 `spec_accept_rate` 与 `spec_accept_length`。配套更新 dashboard `dump_reader` 字段读取、mock SGLang server 输出，并在 11 个测试文件补齐用例（types、metrics、agentive、codec、dashboard、mock、fixtures 等），PR 验证为 224 passed / 1 skipped。

关键文件：

- `miles/utils/types.py`（模块 类型层；类别 source；类型 core-logic；符号 `Sample.SpecInfo`, `SpecInfo.add`, `SpecInfo.from_dict`, `SpecInfo.to_dict`）：投机解码计数器的数据契约核心：`SpecInfo` 字段名对齐 SGLang 规范命名，`add()` 增加无验证不计数守卫，`from_dict()` 保留旧字段 fallback，是整个 PR 的根基。
- `miles/ray/rollout/metrics.py`（模块 指标聚合；类别 source；类型 dependency-wiring；符号 `_compute_spec_metrics`）：指标聚合核心：从逐样本平均改为 carrier 去重 + 全局计数，修复混合批次中普通生成样本被漏计的问题（review 后补齐）。
- `miles/rollout/session/v2/metrics.py`（模块 会话聚合；类别 source；类型 core-logic；符号 `build_session_rollout_metrics`, `SESSION_ROLLOUT_METRICS_KEY`）：新增的 session 侧权威聚合模块：定义 `SESSION_ROLLOUT_METRICS_KEY` 与 `build_session_rollout_metrics`，解决树共享节点重复计数问题。
- `miles/rollout/session/v2/core.py`（模块 会话核心；类别 source；类型 dependency-wiring；符号 `SessionCoreV2.collect_samples`）：wire 边界发布点：`collect_samples` 只在投机解码开启时注入权威 carrier，且放在 hooks 之后，保证 hook 可检视但最终以服务端值为准。
- `miles/rollout/generate_hub/agentive_tool_call.py`（模块 生成调度；类别 source；类型 dependency-wiring；符号 `generate`）：驱动侧接管 carrier：校验 `session_id` 归属与 `metrics` 非空、替换叶子上的陈旧 carrier，失败路径清理悬空归属。
- `miles/rollout/session/samples/codec.py`（模块 样本编解码；类别 source；类型 core-logic）：共享 wire 编解码：保证 `SpecInfo` 在 session v1/v2 与驱动之间往返不丢失，是计数器到达 rollout 指标层的前提。

- tests/fast/router/test_session_samples_op_v2.py (模块 会话测试; 类别 test; 类型 test-coverage; 符号 spec_core, test_spec_info_crosses_samples_wire, test_session_rollout_metrics_are_absent_when_spec_is_disabled, test_session_rollout_metrics_count_every_tree_node_once) : 覆盖最全的 session v2 树状路径测试: spec_info 穿 wire、每节点计一次、被 max_seq_len 排除的节点仍入聚合。
- tests/fast/rollout/generate_hub/test_agentic_v2.py (模块 生成测试; 类别 test; 类型 test-coverage; 符号 _session_metadata, test_transport_collection_error_has_no_metrics_owner, test_v2_replaces_stale_metrics_with_shared_authoritative_carrier, test_v2_rejects_missing_server_session_metrics) : 覆盖驱动侧 carrier 校验与替换的异常路径: 错误 session、缺失 metrics、传输失败清理。
- tests/fast/ray/rollout/test_metrics.py (模块 指标测试; 类别 test; 类型 test-coverage; 符号 TestComputeSpecMetrics, test_aggregates_sglang_counters_before_computing_ratios, test_v2_deduplicates_session_carriers_and_includes_ordinary_samples) : 验证聚合语义: 先聚合后比值、session carrier 去重、普通样本混入。
- tests/fast/utils/test_types.py (模块 类型测试; 类别 test; 类型 test-coverage; 符号 TestSpecInfo, test_ratio_properties, test_add_ignores_completion_without_speculative_verification, test_from_dict_reads_legacy_miles_fields) : SpecInfo 单元测试: 比率属性、add 守卫、旧字段兼容读取。

关键符号: Sample.SpecInfo.add, Sample.SpecInfo.from_dict, Sample.SpecInfo.to_dict, Sample.SpecInfo.spec_accept_rate, _compute_spec_metrics, build_session_rollout_metrics, agentic_tool_call.generate, SessionCoreV2.collect_samples

关键源码片段

miles/utils/types.py

投机解码计数器的数据契约核心: SpecInfo 字段名对齐 SGLang 规范命名, add() 增加无验证不计数守卫, from_dict() 保留旧字段 fallback, 是整个 PR 的根基。

```
@dataclass
class SpecInfo:
    # 字段名与 SGLang 响应契约对齐, 取代旧的 spec_accept_token_num /
    # spec_draft_token_num / completion_token_num 命名 (旧名字在新版
    # SGLang 响应中已不存在, 导致此前指标读不到值) 。
    spec_num_correct_drafts: int = 0
    spec_num_proposed_drafts: int = 0
    spec_verify_ct: int = 0
    completion_tokens: int = 0

    @property
    def spec_accept_rate(self) -> float:
        # 比率属性只负责单条 SpecInfo 的换算; 跨样本聚合时由
        # _compute_spec_metrics 先求和再算比值, 避免小样本被平均稀释。
        if self.spec_num_proposed_drafts == 0:
            return 0.0
```

```

    return self.spec_num_correct_drafts / self.spec_num_proposed_drafts

@property
def spec_accept_length(self) -> float:
    return self.completion_tokens / self.spec_verify_ct if self.spec_verify_ct > 0 else 0.0

def add(self, meta_info: dict) -> None:
    # spec_verify_ct 缺失或为 0 说明该响应未经历投机验证；
    # 只携带 completion_tokens 的普通生成不应污染聚合计数。
    spec_verify_ct = meta_info.get("spec_verify_ct") or 0
    if spec_verify_ct <= 0:
        return
    self.spec_num_correct_drafts += meta_info.get("spec_num_correct_drafts", 0)
    self.spec_num_proposed_drafts += meta_info.get("spec_num_proposed_drafts", 0)
    self.spec_verify_ct += spec_verify_ct
    self.completion_tokens += meta_info.get("completion_tokens", 0)

def to_dict(self) -> dict:
    # 持久化与 wire 传输统一使用 SGLang 规范命名。
    return {
        "spec_num_correct_drafts": self.spec_num_correct_drafts,
        "spec_num_proposed_drafts": self.spec_num_proposed_drafts,
        "spec_verify_ct": self.spec_verify_ct,
        "completion_tokens": self.completion_tokens,
    }

@staticmethod
def from_dict(data: dict) -> "SpecInfo":
    # 历史序列化样本使用旧 Miles 字段名，这里保留 fallback 映射，
    # 保证旧 dump 仍可被正确读取（新数据则直接命中规范键）。
    info = Sample.SpecInfo()
    info.spec_num_correct_drafts = data.get(
        "spec_num_correct_drafts", data.get("spec_accept_token_num", 0)
    )
    info.spec_num_proposed_drafts = data.get(
        "spec_num_proposed_drafts", data.get("spec_draft_token_num", 0)
    )
    info.spec_verify_ct = data.get("spec_verify_ct", 0)
    info.completion_tokens = data.get("completion_tokens", data.get("completion_token_num", 0))
    return info

```

miles/ray/rollout/metrics.py

指标聚合核心：从逐样本平均改为 carrier 去重 + 全局计数，修复混合批次中普通生成样本被漏计的问题（review 后补齐）。

```

def _compute_spec_metrics(args, all_samples: list[Sample]):
    # 未配置投机解码时完全不产出 spec 相关指标。
    if args.sglang_speculative_algorithm is None:

```

```

    return {}

carriers = {}
spec_infos = []
for sample in all_samples:
    carrier = sample.metadata.get(SESSION_ROLLOUT_METRICS_KEY)
    if carrier is None:
        # 普通生成样本（如训练中经 generate_function_path 覆盖到
        # single_turn 生成器）没有 session carrier，直接贡献本地计数，
        # 避免混合批次里这部分生成被漏掉。
        spec_infos.append(sample.spec_info)
    else:
        # session v2 的多片叶子共享同一个权威 carrier:
        # 按 session_id 去重，保证共享生成节点只计一次。
        carriers[carrier["session_id"]] = carrier
spec_infos.extend(
    Sample.SpecInfo.from_dict(carrier["metrics"]["spec_info"])
    for carrier in carriers.values()
)

# 先对所有计数器求和、再计算比值，而不是平均每个样本的比率；
# 这样每个 draft / 验证步骤的权重与其实际数量成正比。
num_correct_drafts = sum(info.spec_num_correct_drafts for info in spec_infos)
num_proposed_drafts = sum(info.spec_num_proposed_drafts for info in spec_infos)
spec_verify_ct = sum(info.spec_verify_ct for info in spec_infos)
completion_tokens = sum(info.completion_tokens for info in spec_infos)
return {
    "spec_accept_rate": num_correct_drafts / num_proposed_drafts if num_proposed_drafts >
    0 else 0.0,
    "spec_accept_length": completion_tokens / spec_verify_ct if spec_verify_ct > 0 else 0.0,
}

```

miles/rollout/session/v2/metrics.py

新增的 session 侧权威聚合模块：定义 SESSION_ROLLOUT_METRICS_KEY 与 build_session_rollout_metrics，解决树共享节点重复计数问题。

```

# 该 key 是 session 服务端发布的权威 rollout 指标载体，
# 驱动侧 (agentic_tool_call) 与聚合侧 (_compute_spec_metrics) 共用。
SESSION_ROLLOUT_METRICS_KEY = "session_rollout_metrics"

```

```

def build_session_rollout_metrics(session_id: str, nodes: list[TrajectoryNode]) -> dict:
    # 在服务端对所有已提交生成节点汇总一次，而不是让每片返回的
    # 叶子各自累加：v2 树中多个样本会共享同一段生成节点，叶子局部
    # 求和会把共享节点重复计数。
    spec_info = Sample.SpecInfo()
    for node in nodes:
        spec_info.add(node.record.response["choices"][0]["meta_info"])
    return {

```

```
"session_id": session_id,
"metrics": {"spec_info": spec_info.to_dict()},
}
```

评论区精华

核心交锋是 guapisolo 的 codex 审查评论 (P2) 指出的混合批次漏计问题，以及 guapisolo 对 session v1/v2 测试覆盖的补强。审查确认了 `use_session_server` 只是默认生成器选择而非样本来源，训练中 `Sample.generate_function_path` 可把样本覆盖到 `single_turn.generate`，这些样本有 `spec_info` 但无 session carrier，原有只聚合 carrier 的实现会让全普通批次的两个指标输出 0.0。nanjiangwill 用 ac647b0e0 提交修复聚合器，并补充混合批次与全普通批次测试。另外 guapisolo 主动补上了 session v1/v2 的 wire 传输测试，并在收尾时判定 tito 检查失败为既有 CI 失败，不阻塞合并。

- 混合批次中普通生成样本的 spec 计数被遗漏 (correctness): nanjiangwill 在 ac647b0e0 提交修复: `_compute_spec_metrics` 按 session_id 去重 carrier，同时纳入无 carrier 样本的 spec_info，并新增混合批次与全普通批次测试。
- session v1/v2 的 SpecInfo 传输缺失测试覆盖 (testing): codec 补齐 SpecInfo 序列化，v1/v2 分别新增 `test_spec_info_crosses_samples_wire` 精确计数器往返测试。
- CI 失败是否为既有问题 (other): 非本 PR 引入，合并未受阻。

风险与影响

- 风险:
 1. 数据契约变更: miles/utils/types.py 的 from_dict 有旧字段 fallback，历史 dump 可读；但新序列化数据字段名变化会影响所有读写路径，dashboard dump_reader 与 mock server 已同步，外部消费 dump 的脚本需留意。
 2. 新增硬校验: miles/rollout/generate_hub/agent_tool_call.py 在 spec 开启且 session v2 成功返回时，carrier 缺失 / session id 不匹配 / metrics 为空都会抛异常 (KeyError / ValueError)。这是有意的 fail-fast，但要求 session server 与 driver 同步部署，旧 server + 新 driver 组合会直接失败。
 3. 聚合语义变化: 从逐样本平均改为全局计数后，spec_accept_rate 数值会系统性变化，依赖旧基线的监控与看板需要重新校准解读。
 4. 测试依赖 mock: 数百个测试与复现均基于 mock SGLang 与手工构造的树节点，未覆盖真实 GPU 端到端；真实 SGLang 响应若字段缺失，SpecInfo.add 的 spec_verify_ct <= 0 守卫可防崩，但计数会偏低。- 影响: 对用户与系统: 投机解码指标从 "恒 0 或失真" 变为可置信的全局计数，直接改善训练与评估监控的可观测性；session v1/v2 行为一致，wire 上新增 session_rollout_metrics 权威载体。对团队: session server 与 driver 需同步升级，dashboard 展示字段随之更新；树状采样下 "服务端聚合一次 + 按 session_id 去重" 的语义具有通用借鉴价值。影响范围是 rollout 指标、session 服务、dashboard 与 mock 测试基建，属跨模块但边界清晰的指标正确性修复。- 风险标记: 数据契约变更，跨模块联动，新增硬校验，测试依赖 mock

关联脉络

- PR #3115 Scan full responses for repetition metrics: 同属 rollout 指标正确性修复脉络，均改动指标计算与聚合路径，可对照理解 metrics 模块的持续演进。
- PR #2825 fix: chunk true-on-policy logprob computation: 同为 rollout 指标计算路径的 bugfix，涉及样本级数据在聚合前的处理，共享 rollout 测试基建。