PR #22959 完整报告

sgl-project/sglang

fix(loads): preserve include filtering after watching mode switch

合并时间：2026-04-16 18:04

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/22959

执行摘要

- 一句话：修复 /v1/loads 端点切换 watching 模式后 include 过滤失效的问题。
- 推荐动作：该 PR 是重要的 bugfix，值得精读以理解 watching 模式下的客户端过滤模式。关注 get_loads 方法中的映射字典设计和 watching_call 中的引用捕获时机，这些是异步通信模式下的典型设计决策。

功能与动机

PR body 明确指出，这是对 #22919 的跟进修复。在 #22919 将 get_loads_communicator 切换到 watching 模式后，由于并发调用者共享同一个请求，调度器总是收到 include=["all"]，这使得 include 查询参数完全失效——调用者无论请求什么，都会收到所有负载指标节。需要添加客户端过滤来保持原有的 API 契约。

实现拆解

1. 修复 watching_call 中的引用捕获顺序：在 watching_call 方法中，将 values = self._result_values 和 event = self._result_event 的赋值顺序调整到 await event.wait() 之前，并移动注释以明确捕获引用的时机，确保后续等待者能正确获取结果。
2. 在 get_loads 方法中添加客户端过滤：在 get_loads 方法中，当 include 参数存在且不包含 "all" 时，创建一个 include_set 和映射字典 _section_attrs，遍历所有结果对象，将不在 include_set 中的对应属性设置为 None。
3. 配套调整：本次变更仅涉及一个源码文件，没有测试、配置或部署配套改动。

关键文件：

- python/sglang/srt/managers/tokenizer_communicator_mixin.py（模块 管理器；类别 source；类型 core-logic；符号 watching_call, get_loads）：这是本次 PR 唯一修改的文件，包含了修复 include 过滤失效的核心逻辑。

关键符号：watching_call, get_loads

关键源码片段

python/sglang/srt/managers/tokenizer_communicator_mixin.py

这是本次 PR 唯一修改的文件，包含了修复 include 过滤失效的核心逻辑。

```
async def get_loads(
```

```
    self,
    include: Optional[List[str]] = None,
    dp_rank: Optional[int] = None,
) -> List[GetLoadsReqOutput]:
    """
    获取 /v1/loads 端点的综合负载指标。
    """
    self.auto_create_handle_loop()
    # 由于 watching 模式在并发调用者间共享结果，我们总是请求所有节，然后在这里过滤。
    req = GetLoadsReqInput(include=["all"], dp_rank=None)
    results = await self.get_loads_communicator(req)

    # 如果指定了 dp_rank，则按 dp_rank 过滤
    if dp_rank is not None:
        results = [r for r in results if r.dp_rank == dp_rank]

    # 客户端过滤可选节（调度器总是返回所有节）
    if include and "all" not in include:
        include_set = set(include)
        # 映射客户端查询参数到结果对象属性名
        _section_attrs = {
            "memory": "memory",
            "spec": "speculative",
            "lora": "lora",
            "disagg": "disaggregation",
            "queues": "queues",
        }
        for r in results:
            for key, attr in _section_attrs.items():
                if key not in include_set:
                    setattr(r, attr, None) # 将未请求的节设置为 None

    return results
```

评论区精华

review 中只有一条来自 gemini-code-assist[bot] 的评论，建议将 _section_attrs 字典重构为模块级常量以提高性能和可维护性。评论指出该字典在每次 get_loads 调用时都会重新创建，应遵循 PEP 8 指南定义为常量。但 PR 作者在后续提交中并未采纳此建议，最终代码仍保留了方法内定义的字典。

- 将 _section_attrs 字典重构为模块级常量 (performance): PR 作者未采纳该建议，最终代码仍保留方法内定义的字典。

风险与影响

- 风险：1. 回归风险：本次变更修复了 include 过滤失效的问题，但未修改核心通信逻辑，风险较低。不过，在 watching_call 中调整引用捕获顺序可能影响极端并发场景下的行为，需

确保不会引入新的竞态条件。 2. 性能风险：在 get_loads 中添加了 O(nm) 的循环过滤（n 为结果数，m 为节属性数），对于大量结果可能增加开销，但考虑到负载端点调用频率不高，影响有限。 3.兼容性风险*：无，本次变更恢复了原有 API 行为，对用户透明。

- 影响：1. 用户影响：外部调用 /v1/loads 端点时，include 参数将重新生效，用户可以按需获取特定负载指标节，减少不必要的数据传输。 2. 系统影响：仅影响负载监控端点，不涉及推理、调度等核心路径。 3. 团队影响：修复了 #22919 引入的副作用，确保了 API 一致性，维护了系统可观测性模块的可靠性。
- 风险标记：客户端过滤开销，异步引用捕获

关联脉络

- PR #22919 fix(loads): switch get_loads_communicator to watching mode：本次 PR 是 #22919 的直接跟进修复，解决了 #22919 引入的 include 过滤失效问题。