PR #25628 完整报告

sgl-project/sglang

Move queue-load reporting to SchedulerLoadInquirer

合并时间：2026-05-18 18:41

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25628

执行摘要

- 一句话：将队列负载查询逻辑迁移到独立组件
- 推荐动作：值得精读，以理解如何将静态辅助方法从混入类迁移到组合组件，同时保持调用方最小改动。该模式可推广到其他混入类。

功能与动机

继续执行调度器重构链，将队列负载度量报告职责从 SchedulerMetricsMixin 移至独立组件 SchedulerLoadInquirer，以实现关注点分离和模块化，为后续简化调度器提供基础。

实现拆解

1. 在 load_inquirer.py 中新增方法：将 _get_num_pending_tokens 和 get_loads 从 SchedulerMetricsMixin 复制到 SchedulerLoadInquirer 类，移除 @staticmethod 装饰器，将 self: 'SchedulerLoadInquirer' 类型注解简化为 self，并添加必要的导入（io_struct 类型等）。
2. 从 scheduler_metrics_mixin.py 中删除方法及相关导入：删除这两个方法及其所有关联代码，同时移除不再需要的导入（如 GetLoadsReqInput、GetLoadsReqOutput 以及类型检查块中的 SchedulerLoadInquirer 引用）。
3. 更新 scheduler.py 中的调用：在 init_request_dispatcher 的 RPC 分发 lambda 中，将 lambda req: self.get_loads(self.load_inquirer, req) 改为 lambda req: self.load_inquirer.get_loads(req)；在 _get_new_batch_prefill_raw 中，将 self._get_num_pending_tokens(self.load_inquirer, ...) 改为 self.load_inquirer._get_num_pending_tokens(...)。
4. 更新 scheduler_output_processor_mixin.py 中的调用：在 stream_output_generation 中，将 self.get_loads(self.load_inquirer, GetLoadsReqInput(...)) 改为 self.load_inquirer.get_loads(GetLoadsReqInput(...))。

关键文件：

- python/sglang/srt/managers/scheduler_components/load_inquirer.py（模块 调度器组件；类别 source；类型 dependency-wiring；符号 _get_num_pending_tokens, get_loads）：接收了两个核心方法，新增了大量导入，是本次移动的目标组件。
- python/sglang/srt/observability/scheduler_metrics_mixin.py（模块 可观测性；类别 source；类型 dependency-wiring；符号 _get_num_pending_tokens, get_loads）：移除

了两个方法及相关导入，是源出处。

- python/sglang/srt/managers/scheduler.py（模块 调度器；类别 source；类型 core-logic）：更新了 RPC 分发和预填批处理中的两处调用，适配新接口。
- python/sglang/srt/managers/scheduler_output_processor_mixin.py（模块 调度器；类别 source；类型 core-logic）：更新了输出流中的负载获取调用。

关键符号：_get_num_pending_tokens, get_loads

关键源码片段

python/sglang/srt/managers/scheduler_components/load_inquirer.py

接收了两个核心方法，新增了大量导入，是本次移动的目标组件。

```
# python/sglang/srt/managers/scheduler_components/load_inquirer.py
# 新增的方法：计算待填充 token 数量
    def _get_num_pending_tokens(self, chunk_deduct: int = 0) -> int:
        """Get the total number of tokens pending prefill.

        This includes tokens from waiting queue requests plus remaining tokens
        from the currently chunked request.

        Args:
            chunk_deduct: extra tokens to subtract from the chunked request's
                remaining count.  At batch-scheduling time the current chunk
                has been planned but ``prefix_indices`` does not yet include it,
                so callers pass ``extend_input_len`` here.  At load-reporting
                time ``prefix_indices`` is already up-to-date, so the default
                0 is correct.
        """
        # 等待队列中所有请求的序列长度之和
        num_pending_tokens = sum(req.seqlen for req in self.get_waiting_queue())
        # 如果存在分块请求，则加上未处理的 token 数
        if self.get_chunked_req() is not None:
            req = self.get_chunked_req()
            num_pending_tokens += req.seqlen - len(req.prefix_indices) - chunk_deduct
        return num_pending_tokens

    # 新增的方法：获取综合负载度量（用于 /v1/loads 端点）
    def get_loads(self, req: GetLoadsReqInput = None) -> GetLoadsReqOutput:
        # 默认只包含核心字段
        if req is None:
            req = GetLoadsReqInput()
        include = set(req.include) if req.include else {"core"}
        include_all = "all" in include

        num_running_reqs = len(self.get_running_batch().reqs)

        # 根据分离模式收集不同的等待队列
```

```
    waiting_queues = [self.get_waiting_queue()]
    if self.disaggregation_mode == DisaggregationMode.PREFILL:
        waiting_queues.append(self.get_disagg_prefill_bootstrap_queue().queue)
    elif self.disaggregation_mode == DisaggregationMode.DECODE:
        waiting_queues.append(self.get_disagg_decode_prealloc_queue().queue)
        waiting_queues.append(self.get_disagg_decode_transfer_queue().queue)
        waiting_queues.append(
            self.get_disagg_decode_prealloc_queue().retracted_queue
        )

    num_waiting_reqs = sum(len(queue) for queue in waiting_queues)
    # 获取 token 池统计
    num_used_tokens, kv_token_usage = (
        self.pool_stats_observer.get_pool_stats().get_kv_token_stats()
    )
    num_total_tokens = num_used_tokens + sum(
        req.seqlen for queue in waiting_queues for req in queue
    )
    # ... 构建 GetLoadsReqOutput ...
```

评论区精华

该 PR 无人工审核评论，仅有一条机器人生成的每日配额警告。变更本身是机械性移动，未产生设计争议。

- 暂无高价值评论线程

风险与影响

- 风险：变更纯属重构，逻辑零改动，但存在以下风险：
  - 导入丢失：load_inquirer.py 新增了对 io_struct 多个类的导入，若遗漏可能导致运行时 NameError。
  - 调用签名错误：scheduler.py 和 scheduler_output_processor_mixin.py 中的调用方式改变，若出现拼写错误或参数错位会导致崩溃。
  - 测试覆盖缺失：当前变更未包含对应单元测试，回归风险依赖现有集成测试。
  - 影响：用户 / 系统：无功能影响，/v1/loads 端点和调度逻辑行为完全一致。开发者：改善了调度器模块化，未来维护负载查询逻辑只需关注 SchedulerLoadInquirer，无需修改混入类。团队：延续了大规模重构链，需确保后续 PR 兼容。
- 风险标记：缺少测试覆盖，导入变更风险

关联脉络

- PR #25629 Add SchedulerMetricsReporter and route metrics state through it: 同属于调度器重构链，引入了度量报告组件，本 PR 将负载查询进一步拆分。
- PR #25635 Move output streaming to SchedulerOutputStreamer: 属于同一系列解耦混入类的 PR，本 PR 继续遵循相同模式。

- PR #25638 Move module-level helpers out of scheduler.py: 同样是模块级辅助函数重组，共同推进调度器架构简化。