# PR #23010 完整报告

sgl-project/sglang

Merge /get_load into /v1/loads

合并时间：2026-04-18 04:36

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23010

## 执行摘要

- 一句话：合并 /get_load 到 /v1/loads，统一负载报告路径并弃用旧端点。
- 推荐动作：该 PR 值得精读，尤其关注数据结构的统一设计和向下兼容处理。值得注意的设计决策包括：在 GetLoadsReqOutput 中新增 num_total_tokens 字段以区分已使用和总令牌数，以及通过垫片保留旧 API 的平滑迁移路径。

## 功能与动机

PR body 明确指出目标是 'Collapses the duplicate load-reporting path into GetLoadsReqOutput'，即合并重复的负载报告路径以简化代码和维护。旧端点 /get_load 被保留为弃用垫片，确保现有客户端能平稳迁移。

## 实现拆解

1. 移除旧数据结构：在 python/sglang/srt/managers/io_struct.py 中，删除 GetLoadReqInput 和 GetLoadReqOutput 类，并将 BatchTokenIDOutput 和 BatchStrOutput 中的 load 字段类型改为 GetLoadsReqOutput，同时为 GetLoadsReqOutput 添加 num_total_tokens 字段以包含待处理预填充令牌。
2. 统一负载计算逻辑：在 python/sglang/srt/observability/scheduler_metrics_mixin.py 中，移除 get_load 方法，并扩展 get_loads 方法以计算 num_total_tokens（使用令牌数加等待队列中的序列长度），确保负载指标更全面。
3. 更新 API 端点：在 python/sglang/srt/entrypoints/http_server.py 中，修改 /get_load 端点，使其作为垫片调用 /v1/loads 并投影数据到旧格式（如 num_tokens 映射为 num_total_tokens），同时记录弃用警告。
4. 调整调用方和路由：修改 python/sglang/srt/managers/tokenizer_control_mixin.py 移除 get_load 通信器；更新 python/sglang/srt/managers/data_parallel_controller.py 中的 DPBudget.update_budget 以使用 num_total_tokens；其他文件如 scheduler.py 做微小调整。
5. 测试覆盖：新增单元测试文件 test/registered/unit/managers/test_dp_budget.py 和 test/registered/unit/entrypoints/test_v1_loads_aggregate.py，锁定字段映射和聚合逻辑，防止回归。

关键文件：

- python/sglang/srt/observability/scheduler_metrics_mixin.py（模块 调度器指标；类别 source；类型 core-logic；符号 get_load）：核心负载计算逻辑变更，移除旧 get_load 方法并扩展 get_loads 以包含 num_total_tokens，直接影响调度器指标报告。
- python/sglang/srt/managers/io_struct.py（模块 数据结构；类别 source；类型 core-logic；符号 GetLoadReqInput, GetLoadReqOutput）：数据结构核心变更，删除旧负载类并更新输出结构，为 GetLoadsReqOutput 添加 num_total_tokens 字段，影响数据契约。
- python/sglang/srt/entrypoints/http_server.py（模块 HTTP 接口；类别 source；类型 entrypoint）：API 入口点变更，更新 /get_load 端点作为弃用垫片，确保向下兼容，影响客户端调用。
- test/registered/unit/managers/test_dp_budget.py（模块 预算测试；类别 test；类型 test-coverage；符号 _load, TestDPBudgetUpdateBudget, test_maps_running_plus_waiting_to_total_requests, test_maps_num_total_tokens_not_num_used_tokens）：新增单元测试，锁定 DPBudget.update_budget 对 GetLoadsReqOutput 字段的映射，防止回归。
- test/registered/unit/entrypoints/test_v1_loads_aggregate.py（模块 聚合测试；类别 test；类型 test-coverage；符号 _load, TestComputeAggregate, test_multi_dp_rank_sums, test_averages_over_dp_count）：新增单元测试，验证 /v1/loads 聚合逻辑，区分 total_used_tokens 和 total_tokens，确保负载报告准确性。

关键符号：get_load, get_loads, _compute_aggregate, update_budget

## 关键源码片段

### python/sglang/srt/observability/scheduler_metrics_mixin.py

核心负载计算逻辑变更，移除旧 get_load 方法并扩展 get_loads 以包含 num_total_tokens，直接影响调度器指标报告。

```
def get_loads(self: Scheduler, req: GetLoadsReqInput = None) -> GetLoadsReqOutput:
    """
    获取综合负载指标用于 /v1/loads 端点。
    关键变更：移除旧 get_load 方法，统一计算 num_total_tokens 以包含等待队列中的待处理令牌。
    """
    num_running_reqs = len(self.running_batch.reqs)
    waiting_queues = [self.waiting_queue] # 基础等待队列
    # 根据解聚模式添加额外队列
    if self.disaggregation_mode == DisaggregationMode.PREFILL:
        waiting_queues.append(self.disagg_prefill_bootstrap_queue.queue)
    elif self.disaggregation_mode == DisaggregationMode.DECODE:
        waiting_queues.append(self.disagg_decode_prealloc_queue.queue)
        waiting_queues.append(self.disagg_decode_transfer_queue.queue)
        waiting_queues.append(self.disagg_decode_prealloc_queue.retracted_queue)

    num_waiting_reqs = sum(len(queue) for queue in waiting_queues)
    num_used_tokens, kv_token_usage = self.get_pool_stats().get_kv_token_stats()
    # 计算总令牌数：已使用令牌 + 等待队列中所有请求的序列长度
```

```
    num_total_tokens = num_used_tokens + sum(
        req.seqlen for queue in waiting_queues for req in queue
    )

    return GetLoadsReqOutput(
        dp_rank=self.dp_rank,
        num_running_reqs=num_running_reqs,
        num_waiting_reqs=num_waiting_reqs,
        num_used_tokens=num_used_tokens,
        num_total_tokens=num_total_tokens, # 新增字段，用于数据并行负载均衡
        max_total_num_tokens=self.max_total_num_tokens,
        token_usage=kv_token_usage,
        # 其他字段省略
    )
```

### python/sglang/srt/managers/io_struct.py

数据结构核心变更，删除旧负载类并更新输出结构，为 GetLoadsReqOutput 添加 num_total_tokens 字段，影响数据契约。

```
@dataclass
class GetLoadsReqOutput(BaseReq):
    """
    每个数据并行 rank 的负载指标，用于 /v1/loads 端点。
    关键变更：新增 num_total_tokens 字段，包含已使用令牌和待处理预填充令牌。
    """
    dp_rank: int
    timestamp: float
    num_running_reqs: int = field(
        metadata={"metric": ("gauge", "运行中的请求数")}
    )
    num_waiting_reqs: int = field(
        metadata={"metric": ("gauge", "等待中的请求数")}
    )
    num_used_tokens: int = field(
        metadata={"metric": ("gauge", "已使用的令牌数")}
    )
    # num_total_tokens 包含已使用令牌和待处理预填充令牌（等待队列序列长度）
    # 用于数据并行负载均衡，避免长提示工作负载下低估负载
    num_total_tokens: int = field(
        metadata={"metric": ("gauge", "总令牌数（已使用 + 待处理）")}
    )
    max_total_num_tokens: int = field(
        metadata={"metric": ("gauge", "最大令牌容量")}
    )
    # 其他字段省略
```

## 评论区精华

Review 中仅有一条批准评论，无具体讨论。但从 commit 历史可见，初始尝试完全移除 /get_load 垫片（commit 'drop get_load shim'）后被回退，最终决定保留垫片以确保向后兼容。这体现了设计权衡：简化代码 vs. 客户端兼容性。

- 是否完全移除 /get_load 垫片 (design): 决定保留 /get_load 作为弃用垫片，提供平滑迁移路径，平衡代码简洁性和客户端兼容性。

## 风险与影响

- 风险：主要风险包括：1) 回归风险：字段映射错误（如 DPBudget.update_budget 错误使用 num_used_tokens 而非 num_total_tokens）可能导致数据并行负载均衡失效，尤其影响长提示工作负载；测试已覆盖此点。2) 兼容性风险：旧客户端依赖 /get_load 端点，垫片虽提供过渡，但弃用警告可能被忽略，需文档说明。3) 性能影响：新负载计算增加了 num_total_tokens 求和，但开销可忽略。
- 影响：用户影响：/get_load 端点被标记为弃用，用户应迁移到 /v1/loads；负载报告数据更准确，包含待处理令牌。系统影响：统一了负载指标数据结构，减少代码重复，提升可维护性；数据并行调度依赖新字段，可能改善负载均衡。团队影响：简化了 API 表面，降低未来开发负担；但需更新相关文档和客户端代码。
- 风险标记：核心路径变更，数据契约变更，向后兼容性风险，测试覆盖关键

## 关联脉络

- PR #23006 [Pipeline Parallelism][Bug] Fix scheduler hang in pipeline parallelism setup: 同样修改了 scheduler.py 文件，涉及调度器逻辑，可能共享负载报告和调度上下文。