PR #26348 完整报告

sgl-project/sglang

Optimize get load calls (/v1/loads) using shared-memory load snapshots

合并时间：2026-05-30 04:40

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26348

执行摘要

- 一句话：使用共享内存快照优化 /v1/loads 性能，延迟降低 10-100 倍
- 推荐动作：建议团队精读 load_snapshot.py 中双后端的设计模式（策略模式选择），以及 refresh_load_budget 中的 20ms 节流逻辑，这是性能与准确性权衡的典型实践。该 PR 为未来扩展实时监控和负载均衡提供了基础架构。

功能与动机

旧的负载获取方法依赖 ZMQ 通信，在 scheduler 执行长 chunked prefill 时会产生额外延迟。本 PR 的目标是将 get_loads 调用的延迟降低 10-100 倍，并减少不必要的 IPC 开销。

实现拆解

1. 新增负载快照模块（load_snapshot.py）：定义 LoadSnapshot 数据类（使用 msgspec 序列化）和两种传输后端：SHM（mmap 写入 /dev/shm）和 ZMQ fallback。提供工厂函数根据配置选择后端。
2. Scheduler 发布快照（scheduler.py）：Scheduler 初始化时创建 writer，并在每次调度循环后调用 publish_load_snapshot，根据配置的间隔（默认 15 次调度周期）决定是否发布。
3. 重构 /v1/loads 端点（v1_loads.py）：移除旧的基于 dataclass 的序列化、聚合计算以及多种度量 dataclass 的导入。改为调用 tokenizer_manager.get_loads 获取 LoadSnapshot 列表，利用 to_dict 方法返回。新增 include 和 format 查询参数。
4. DP 控制器使用共享内存（data_parallel_controller.py）：移除原有的 handle_load_update_req（ZMQ 接收负载更新），改为在 dispatching_with_trace 中可选地调用 refresh_load_budget。该方法读取 load_snapshot_reader 的所有快照，并调用 DPBudget.update_budget（增加 last_timestamp 跳过陈旧数据）。刷新有 20ms 节流，避免高并发时每个 dispatch 都读。
5. 配置与测试：在 server_args.py 添加 --load-snapshot-publish-interval 参数；在 environ.py 添加 SGLANG_LOAD_SNAPSHOT_USE_ZMQ 环境变量。新增单元测试 test_load_snapshot_backends.py（SHM 和 ZMQ 后端的读写、多 rank）、集成测试 test_load_snapshot_server.py（真实服务器无 DP 和普通 DP 场景），修改 test_v1_loads_aggregate.py 和 test_data_parallel_controller.py 适配新逻辑。

关键文件：

- python/sglang/srt/managers/load_snapshot.py（模块 负载快照；类别 source；类型 core-logic；符号 _native, should_use_zmq, zmq_reader_owner, LoadSnapshot）：核心新增文件，实现共享内存负载快照机制，包含双后端策略和序列化
- python/sglang/srt/entrypoints/v1_loads.py（模块 负载端点；类别 source；类型 core-logic；符号 _loads_dict_factory, _compute_aggregate, _format_loads_prometheus）：核心修改文件，重构 /v1/loads 端点，移除聚合逻辑，改为使用 LoadSnapshot.to_dict
- python/sglang/srt/managers/data_parallel_controller.py（模块 数据并行控制；类别 source；类型 entrypoint；符号 update_budget, handle_load_update_req, dispatching_with_trace, refresh_load_budget）：DP 控制器从 ZMQ 改为共享内存读取负载，增加 refresh_load_budget 和节流机制
- python/sglang/srt/managers/scheduler.py（模块 调度器；类别 source；类型 dependency-wiring；符号 publish_load_snapshot, handle_get_loads_req）：Scheduler 新增 publish_load_snapshot 方法，定期向共享内存发布负载
- test/registered/unit/managers/test_load_snapshot_backends.py（模块 后端测试；类别 test；类型 test-coverage；符号 _temp_path, _ipc_addr, _warmup_zmq, TestShmRoundTrip）：全面单元测试 SHM 和 ZMQ 后端的读写、多 rank 场景

关键符号：publish_load_snapshot, refresh_load_budget, create_load_snapshot_writer, should_use_zmq, zmq_reader_owner, _format_loads_prometheus, get_loads, update_budget, dispatching_with_trace

关键源码片段

python/sglang/srt/entrypoints/v1_loads.py

核心修改文件，重构 /v1/loads 端点，移除聚合逻辑，改为使用 LoadSnapshot.to_dict

```
@router.get("/v1/loads")
async def get_loads(
    dp_rank: Optional[int] = None,
    include: Optional[str] = None,
    format: Optional[str] = None,
    tokenizer_manager=Depends(_get_tokenizer_manager),
):
    """
    获取所有 DP rank 的负载指标。

    查询参数:
        dp_rank: 过滤指定 DP rank（可选）
        include: 逗号分隔的 section 过滤（可选，如 core, memory 等）
        format: 响应格式，json（默认）或 prometheus
    """
    include_list = [s.strip() for s in include.split(",")] if include else None

    load_results = await tokenizer_manager.get_loads(
        include=include_list,
```

```
        dp_rank=dp_rank,
    )

    if format == "prometheus":
        # 使用 to_dict 生成 Prometheus 格式，支持 include 过滤
        return _format_loads_prometheus(load_results, include_set)

    # JSON 格式：每个 LoadSnapshot 通过 to_dict 转换为字典
    loads = [load.to_dict(include_set) for load in load_results]
    return {"loads": loads, "timestamp": ..., "version": ...}
```

评论区精华

本 PR 无 review 评论。原始 commit 来自 #25841（由 @cctry 提交），@merrymercy 在此基础上整合并修复了循环导入和格式化问题。

- 暂无高价值评论线程

风险与影响

- 风险：共享内存文件在进程异常退出时可能残留，需确保清理逻辑（已有文件锁和自动清理机制）。多节点 ZMQ 模式依赖于网络和进程健康，PULL 套接字 owner 进程崩溃会导致快照中断。DPBudget 的 20ms 节流阈值在极端负载下可能导致短暂的调度决策使用陈旧数据，但这是经过权衡的设计。/v1/loads API 返回格式移除了 *aggregate* 字段，现有客户端需要适配。新代码引入了 msgspec 依赖和更多系统调用，但影响可控。
- 影响：正面影响：/v1/loads 延迟大幅降低，scheduler 阻塞不再影响负载读取；DP 负载均衡数据获取更及时；减少 ZMQ 通信量。负面影响：API 返回格式变化，客户端需更新；共享内存路径在 Windows 上可能受限（未测试）；团队需要维护两套后端（SHM 和 ZMQ），增加测试负担。
- 风险标记：共享内存残留风险，API 响应格式变更，多节点 ZMQ 依赖，节流阈值可能不准确

关联脉络

- PR #25841 [cctry] original load snapshot POC: 本 PR 基于 #25841 的 commit 进行整合、修复和最终集成