PR #34608 完整报告

sgl-project/sglang

Publish per-scheduler load on a dedicated socket for load-aware routers

合并时间：2026-08-27 19:33

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34608

执行摘要

- 一句话：为负载感知路由器新增 per-scheduler 负载发布 socket
- 推荐动作：值得精读。这是为外部路由器提供真实负载数据的基础设施 PR，设计质量高：端口推导与广播单一来源、opt-in 保护升级兼容、去重心跳保证迁移及时性、golden bytes 钉跨语言契约。对计划接入 sgl-router cache_aware_zmq 或自研负载感知路由的团队，这是必读的引擎侧对接规范；对一般模型服务团队，只需确认 --load-publish-endpoint 默认关闭、升级无影响即可。

功能与动机

PR body 明确指出：cache-aware-zmq 路由器用路由器侧的 in-flight 计数器推断 worker 负载，该计数器只统计「当前这个路由器」派发的请求，漏掉其他路由器与直连客户端流量；且流式响应期间计数会一直保持到响应结束，而不是请求真正占据 scheduler 的时间。引擎内部其实已算出正确数字——SchedulerLoadInquirer.get_loads() 每个发布周期产出 LoadSnapshot（喂给 /v1/loads 与 DP-attention 调度），本 PR 就是把这快照暴露给进程外消费者，让路由器按真实队列深度 / KV 占用为 worker 定价。该能力从生产分支移植到 main，路由器侧消费者另行落地。

实现拆解

1. 参数入口与启动校验（server_args.py）：observability 命名空间新增 --load-publish-endpoint（off / auto / 显式 tcp:// 地址），默认关闭；新增 check_load_publish_args 在入口校验——未配 --kv-events-config、不可绑定或与 KV 范围重叠时直接拒绝启动，而不是在 scheduler 子进程里只告警。显式地址仅接受通配主机（* / 0.0.0.0 / [::]），具体主机地址会被判为 connect 风格而拒绝广播。
2. 端口解析收敛到单一 resolver（disaggregation/kv_events.py）：新增 is_kv_publisher_rank（pp / attn-TP / attn-CP rank 0 的统一 gating，KV 发布器与负载发布器共用，否则二者基于 /server_info 推导的端口会不一致）、parse_tcp_port、parse_advertisable_tcp、parse_bindable_tcp 与 resolve_load_pub_range。绑定方 SchedulerLoadPublisher 与广播方 describe_kv_events_publisher 都走同一 resolver，引擎不可能广播一个自己不会绑定的范围；auto 模式紧贴 KV 事件范围之后打包（base + dp_size 起），与相邻 replay ROUTER 范围重叠时自动跳过。
3. 核心发布器（新增 load_publisher.py，267 行）：LoadStat 是 msgspec struct（array_like=True + tag=True，Wire 形状 ["LoadStat", num_running_reqs,

num_waiting_reqs, num_tokens, max_total_num_tokens, attn_dp_rank])，帧格式复用 KV 事件 socket（[b"load", 大端 i64 seq, msgpack LoadStat]）；_open_pub_socket 绑定 PUB socket（HWM=8、LINGER=0、IPv6 开关），刻意不用 get_zmq_socket 因为它设置的 SNDHWM=0 会破坏 HWM 语义；SchedulerLoadPublisher 负责调用节流（LOAD_PUBLISH_INTERVAL=5）、去重心跳（未变化 1 秒重发、变化立即发，迁移永不延迟）与 best-effort 失败处理（每个失败 episode 只记一次日志、不崩溃调度循环，路由器可回退到自己的计数器）。传输是同步 PUB socket（send 只是入队到 ZMQ IO 线程），无后台线程与回放缓冲。

4. 调度器接线与快照复用（scheduler.py）：init_load_publisher 创建发布器并使用与 DP 均衡 writer 相同的发布间隔；publish_load_snapshot 改为返回实际发布的 LoadSnapshot（对下游 override 是破坏性变更），使 load socket 直接复用 DP 均衡 sink 已算好的快照，避免二次 O(queue) 遍历；on_idle 在 stalled（无 batch 但未 fully idle，如 KV 压力下队列停摆、disagg 转移）路径也发布负载，并用 LOAD_STALL_REFRESH_S = 0.05 墙钟下限约束该自旋路径上两个 sink 的 get_loads 频率。
5. 测试与文档配套：test_loadstat_wire.py 用 golden hex 钉死 Wire 编码并覆盖 port/rank gating（CPU-only，socket bind 在 _open_pub_socket 处 stub）；test_server_info.py 覆盖 auto 跳过 replay 范围、显式地址迁移广播 base、connect 风格 KV 端点省略 load 键、IPv6 括号保留与裸 IPv6 拒绝等；test_kv_events.py 覆盖 resolve_load_pub_range 的 off / auto / 冲突 / u16 溢出分支；test_scheduler_on_idle_load.py 验证 stalled 自旋路径 50ms 节流；docs 的 server arguments 表格补充端口占用与放置说明。

关键文件：

- python/sglang/srt/managers/scheduler_components/load_publisher.py（模块 负载发布；类别 source；类型 core-logic；符号 LoadStat, _open_pub_socket, SchedulerLoadPublisher, init）：核心新组件：定义 Wire 契约 LoadStat 与 SchedulerLoadPublisher 发布器，含调用节流、去重心跳、HWM 语义与 best-effort 失败处理。
- python/sglang/srt/disaggregation/kv_events.py（模块 端口解析；类别 source；类型 dependency-wiring；符号 is_kv_publisher_rank, LOAD_TOPIC, parse_tcp_port, parse_advertisable_tcp）：端口推导中枢：新增 is_kv_publisher_rank、parse_* 与 resolve_load_pub_range，绑定与广播共用同一 resolver，防止引擎广播自己不会绑定的范围。
- python/sglang/srt/managers/scheduler.py（模块 调度器；类别 source；类型 core-logic；符号 init_load_publisher, publish_load_snapshot）：接线与快照复用：init_load_publisher 接线新组件，publish_load_snapshot 改为返回快照供发布器复用，stalled no-batch 路径发布负载并限流。
- python/sglang/srt/server_args.py（模块 启动参数；类别 source；类型 configuration；符号 check_load_publish_args, load_publish_endpoint）：新增 --load-publish-endpoint 参数（默认 off）与 check_load_publish_args 启动校验，防止未绑定或重叠范围。
- python/sglang/srt/managers/scheduler_components/kv_events_publisher.py（模块 KV 发布器；类别 source；类型 core-logic）：移除重复的 rank gating 内联判断，复用新抽出的 is_kv_publisher_rank，保持两个发布器门控一致。

- test/registered/unit/managers/test_loadstat_wire.py（模块 负载协议；类别 test；类型 test-coverage；符号 TestLoadStatWire, test_loadstat_golden_bytes, test_loadstat_msgpack_array_shape, test_loadstat_tag_is_class_name）：golden bytes 钉死跨语言 Wire 编码，覆盖 port/rank gating 与默认关闭语义，是 Wire 契约的权威测试。
- test/registered/unit/entrypoints/test_server_info.py（模块 服务信息；类别 test；类型 test-coverage；符号 test_load_port_skips_an_overlapping_replay_range, test_explicit_load_publish_endpoint_moves_the_advertised_base, test_load_keys_omitted_for_connect_style_kv_endpoint, test_load_keys_omitted_when_explicitly_off）：覆盖 /server_info load 端口广播的各类边界：replay 冲突、显式地址迁移、IPv6 括号、connect 风格省略与 off 显式关闭。
- test/registered/unit/disaggregation/test_kv_events.py（模块 KV 事件；类别 test；类型 test-coverage；符号 TestResolveLoadPubRange, test_off_by_default, test_auto_packs_after_kv_range, test_auto_skips_an_overlapping_replay_range）：覆盖 resolve_load_pub_range 各分支：off 默认、auto 打包、replay 冲突跳过、connect 风格拒绝与 u16 溢出拒绝。
- test/registered/unit/managers/test_scheduler_on_idle_load.py（模块 空闲发布；类别 test；类型 test-coverage；符号 TestOnIdleStallPublish, test_spinning_stall_publishes_once_within_the_floor, test_publishes_again_after_the_floor_elapses）：验证 stalled 自旋路径的 50ms 墙钟节流，确保 on_idle 高频调用不会放大 get_loads 开销。
- docs/docs/advanced_features/server_arguments.mdx（模块 文档；类别 docs；类型 documentation）：补充 --load-publish-endpoint 文档，说明端口占用规模与显式地址放置方式，是部署规划的依据。

关键符号：SchedulerLoadPublisher, publish_load_stat, LoadStat, _open_pub_socket, resolve_load_pub_range, parse_tcp_port, parse_advertisable_tcp, parse_bindable_tcp, is_kv_publisher_rank, init_load_publisher, publish_load_snapshot, check_load_publish_args

关键源码片段

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

核心新组件：定义 Wire 契约 LoadStat 与 SchedulerLoadPublisher 发布器，含调用节流、去重心跳、HWM 语义与 best-effort 失败处理。

```
# 每个 scheduler 在独立 ZMQ PUB socket 上周期性发布 LoadStat 仪表值，
# 供进程外的 load-aware 路由器（如 sgl-router 的 cache_aware_zmq 策略）
# 基于真实队列深度与 KV 占用做路由决策。帧格式与 KV 事件 socket 一致：
# [b"load", 大端 i64 seq, msgpack LoadStat]，订阅方一套循环处理两类消息。

from itertools import count
from typing import Optional

import msgspec
```

```
import zmq

# 除 force 外最多每 5 次调用发布一次；若被 DP 快照间隔覆盖则与其同相。
LOAD_PUBLISH_INTERVAL = 5

# 未变化的仪表值最多每 1 秒重发一次；变化值总是立即发出，
# 状态迁移永远不被延迟。约束 idle 自旋循环的发送速率。
LOAD_PUBLISH_HEARTBEAT_S = 1.0

# 小 HWM：load 是仪表值，管道满时丢弃的读数会被下一次心跳覆盖。
# ZMQ_CONFLATE（真正的 newest-wins）不可用——只保留单帧，
# 会破坏三帧帧格式——所以有界积压是最接近的折中。
LOAD_PUB_HWM = 8

_encoder = msgspec.msgpack.Encoder()


class LoadStat(msgspec.Struct, array_like=True, gc=False, tag=True):
    # Wire 形状（tag + array_like）：
    # ["LoadStat", num_running_reqs, num_waiting_reqs, num_tokens,
    # max_total_num_tokens, attn_dp_rank]
    # 路由器按位置解码前四个计数；array_like 总是发射末尾字段
    # （未设置时为 null），解码方必须容忍。不设 omit_defaults：
    # 它可能裁掉末尾默认值，缩短路由器解码的形状。
    num_running_reqs: int
    num_waiting_reqs: int
    num_tokens: int # 在用的 KV tokens
    max_total_num_tokens: int # KV 容量；未知时为 0
    # DP attention 下为 attn_dp_rank，否则为普通 dp_rank；
    # 仅信息用途（路由器按 socket rank 主键）。命名跟随 EventBatch。
    attn_dp_rank: Optional[int] = None


def _open_pub_socket(endpoint: str) -> zmq.Socket:
    # 绑定 load PUB socket。模块级便于测试只 stub 这一处副作用，
    # 同时覆盖真实的 gating 与端口推导。不用 get_zmq_socket：
    # 它会设置 SNDHWM=0，破坏 LOAD_PUB_HWM 的语义。
    sock = zmq.Context.instance().socket(zmq.PUB)
    try:
        sock.set_hwm(LOAD_PUB_HWM)
        sock.setsockopt(zmq.LINGER, 0)
        if is_zmq_endpoint_ipv6(endpoint): # 来自 sglang.srt.utils.network
            sock.setsockopt(zmq.IPV6, 1)
        sock.bind(endpoint)
    except Exception:
        sock.close() # 不泄漏共享 context 上的句柄
        raise
    return sock
```

python/sglang/srt/disaggregation/kv_events.py

端口推导中枢：新增 is_kv_publisher_rank、parse_* 与 resolve_load_pub_range，绑定与广播共用同一 resolver，防止引擎广播自己不会绑定的范围。

```
# 广播为 /server_info 中的 load_topic；load socket 只承载 load，
# 订阅方可以 subscribe-all。
LOAD_TOPIC = 'load'

# PUB socket 可以 BIND 而非 CONNECT 的主机集合。按解析后的主机精确匹配
# 而非子串："::" 出现在每个 IPv6 地址内部，子串测试会把具体远端主机
# 误判为可绑定。
_BIND_WILDCARD_HOSTS = frozenset({'*', '0.0.0.0', '::'})


def parse_tcp_port(endpoint: Optional[str]) -> Optional[int]:
    # 与主机无关地回答 "tcp:// 端点占用哪个合法端口 "，用于冲突检查
    #  （replay ROUTER 可绑定任意主机拼写）。
    if not endpoint or not endpoint.startswith('tcp://'):
        return None
    try:
        port = NetworkAddress.parse(endpoint[len('tcp://'):]).port
    except ValueError:
        return None
    return port if 0 < port <= 65535 else None


def parse_advertisable_tcp(endpoint: Optional[str]) -> Optional[tuple[str, int]]:
    # 适合放进 /server_info 的 (host, port)，否则 None。任意主机都可
    #  （KV 事件走 connect 风格）；IPv6 重新加括号，消费者可拼接
    # tcp://{host}:{port}。裸的无括号 IPv6 被拒绝——与 resolver 相同的
    # 解析，保证描述与绑定一致。
    if not endpoint or not endpoint.startswith('tcp://'):
        return None
    try:
        addr = NetworkAddress.parse(endpoint[len('tcp://'):])
    except ValueError:
        return None
    if not addr.host or not (0 < addr.port <= 65535):
        return None
    host = f'[{addr.host}]' if addr.is_ipv6 else addr.host
    return host, addr.port


def parse_bindable_tcp(endpoint: Optional[str]) -> Optional[tuple[str, int]]:
    # 如果 PUB socket 可以 BIND 该 tcp:// 端点则返回 (host, port)，
    # 否则 None。具体主机在此是 connect 风格：load PUB 绑上去
    # 不会有人连接，而且不会报任何错误。
    if not endpoint or not endpoint.startswith('tcp://'):
```

```
            return None
        try:
            addr = NetworkAddress.parse(endpoint[len('tcp://'):])
        except ValueError:
            return None
        if addr.host not in _BIND_WILDCARD_HOSTS or not (0 < addr.port <= 65535):
            return None
        return addr.host, addr.port
```

python/sglang/srt/managers/scheduler.py

接线与快照复用：init_load_publisher 接线新组件，publish_load_snapshot 改为返回快照供发布器复用，stalled no-batch 路径发布负载并限流。

```
    def publish_load_snapshot(self, force: bool = False):
        # 返回它发布的 LoadSnapshot；禁用、被节流或失败时返回 None——
        # 同址 sink（面向路由器的 load publisher）可复用该快照，
        # 避免再次遍历队列。这是对 override 点的破坏性变更：
        # 下游覆盖实现返回 None 仍功能正确，但会静默丢失共享快照优化。
        writer = self.load_snapshot_writer
        if writer is None:
            return None
        if not force:
            writer.publish_counter += 1
            if writer.publish_counter < writer.publish_interval:
                return None
        writer.publish_counter = 0
        try:
            load = self.load_inquirer.get_loads()
            writer.write(load)
            return load
        except Exception as e:
            logger.warning('load snapshot publish failed: %s', e)
            return None
```

评论区精华

PR 无行内 review 评论（review_comments 为 0），hzh0425 给出 APPROVED；讨论主要沉淀在 22 个 commit 的 review 迭代与 PR Notes 中：

- 端口推导必须单一来源（commit 66116dc）：首轮 review 要求把 load 端口推导收敛到 kv_events.py 的共享 helper，绑定与广播共用；auto 打包必须越过相邻 replay ROUTER 范围（replay = kv + 1 时总是冲突）。
- 去重心跳优于固定发布下限（commit 88841ea）：50ms 墙钟下限可能丢掉 busy→idle 迁移发布（最后一个 batch 的强制发布落在下限前微秒），路由器会看到最长约 1 秒的过期 busy 值；改为「未变化值按 1 秒心跳去重、变化值立即发」，迁移永不延迟。
- 与 #28599 方案收敛（commit 6c8ac210）：采纳并行 PR 中「严格更优」的部分——bindable 端点 resolver、显式 endpoint override、同步 socket，保留本 PR 的 Wire 契约与发布节奏。

- opt-in 是兼容性红线（commit 4d3c29e）：若默认开启，任何 --kv-events-config 用户升级都会保留 kv_base + dp_size 端口，可能撞掉同宿邻居未防护的 KV bind；改为默认关闭后升级零影响。
- hzh0425 的 CI 失败提醒（issue 评论）：某次 CI 运行失败，作者随后修复并 /tag-and-rerun-ci 重跑。
- load 端口推导收敛为单一 resolver，并跳过 replay 范围 (design): 新增 resolve_load_pub_range，绑定与广播统一走该函数；auto模式在重叠时跳过replay范围。
- 去重心跳 vs 固定发布下限 (design): 采用 LOAD_PUBLISH_HEARTBEAT_S=1.0 去重心跳，迁移发布永不延迟；同时保留调用节流并在 dedup 命中后继续累计计数。
- 与 PR #28599 并行方案的收敛 (design): 完成方案合并，形成最终的 resolve_load_pub_range / 显式 override / 同步 PUB 设计。
- 默认关闭以避免升级端口碰撞 (design): 特性改为 opt-in（--load-publish-endpoint），默认不预留端口；启动校验在入口拒绝重叠范围。
- publish_load_snapshot 返回快照的破坏性变更 (design): 两个调用点均把快照喂给 load publisher；文档要求 override 返回所写快照以保留优化。
- stalled no-batch 路径的发布与限流 (performance): on_idle 在 fully-idle 门控前发布，并引入 LOAD_STALL_REFRESH_S=0.05 墙钟下限约束两个 sink。

风险与影响

- 风险：
  1. 端口冲突（启用时）：auto 模式每个 worker 从 KV base 起额外占用 2 * dp_size 个端口（相邻 replay 时为 2 * dp_size + 1），同宿 worker 必须拉开 KV base 间距或改用显式地址；这是 opt-in 而非默认开启的直接原因。
  2. 跨语言 Wire 契约未闭环：test_loadstat_golden_bytes 只钉死 Python 侧编码，Rust 路由器侧消费方尚未合入，字段重排或改名在闭环前仍是静默跨语言破坏风险；测试注释明确要求 Rust 侧断言同一 hex。
  3. 破坏性 override 变更：Scheduler.publish_load_snapshot 返回值从无变为 LoadSnapshot，下游自定义 scheduler 若未同步返回快照，load socket 退化为自行遍历队列——功能正确但静默丢失共享快照优化。
  4. 严格端点解析的用户可见变更：裸无括号 IPv6 KV 端点（如 tcp://::1:5557）会使 /server_info 整体丢弃 kv_events 块，旧逻辑会广播一个不可用地址；依赖旧行为的部署需改写为 tcp://[::1]:5557。
  5. 热路径新增逻辑：发布路径在每个调度循环运行，失败日志按 episode 限流；默认关闭时每个 batch 仅一次 bound method 加 None 检查；stalled 自旋路径由 50ms 墙钟下限约束 get_loads，避免放大 O(queue) 开销。
- 影响：
  1. 用户与部署：默认零影响；启用后为 sgl-router 提供真实负载源，但端口占用需纳入部署规划；裸 IPv6 KV 端点用户面临有意的行为变更。
  2. 系统：调度器新增一个可选 PUB socket 与发布逻辑，与 DP 均衡 sink 共享快照避免重复遍历，stalled 路径获得 50ms 级负载刷新。

3. 团队：为该功能线定义了跨语言 Wire 契约（msgpack 形状、帧格式、golden 测试方法），后续 router PR 需对齐；server args 表格新增参数，observability 命名空间继续扩充；多人协作痕迹明显（hzh0425 合入 main 分支并修复 server arg，最终审批），属于团队共同维护的重点特性（带 high priority 标签）。 - 风险标记：热路径新增发布逻辑，跨语言 Wire 契约未闭环，端口占用规划，破坏性 override 变更，默认关闭的 opt-in 特性，严格端点解析变更

关联脉络

- PR #28599 Per-scheduler load publisher（并行方案，提交中提及）：commit 6c8ac210 明确「采纳 #28599 中严格更优的部分」（bindable 端点 resolver、显式 override、同步 socket），与本 PR 属于同一功能线的并行方案收敛。
- PR #36586 [Core] Refactor server argument choices: 同期重构 server_args.py 参数注册与校验路径，与本 PR 新增的 --load-publish-endpoint / check_load_publish_args 处于同一耦合面。
- PR #36681 Move server args config parser under utils: server_args.py 解析器迁移至 utils/server_args_config_parser.py，本 PR 的启动校验依赖同一套参数基础设施，后续迁移需随迁。