PR #2070 完整报告

THUDM/slime

[docker] expose sglang load inflight details

合并时间：2026-06-12 19:16

原文链接：http://prhub.com.cn/THUDM/slime/pull/2070

执行摘要

- 一句话：暴露 sglang 负载 inflight 细节
- 推荐动作：建议运维团队关注该能力，可用于请求生命周期分析和调度调优。设计上请求触发采集，性能影响可控。

功能与动机

当前 /v1/loads 端点仅返回聚合统计数据，运维人员难以定位请求堆积或卡死问题。暴露 inflight 细节可帮助排查请求级瓶颈。

实现拆解

1. 在 docker/patch/latest/sglang.patch 中修改 sglang 源码：在 GetLoadsReqInput 的 VALID_SECTIONS 加入 'inflight'，使请求时可以指定包含 inflight 明细。
2. 在 GetLoadsReqOutput 中新增 inflight 字段（Optional[List[Dict]]），用于返回每队列的请求级详情（rid、bootstrap_room、seqlen、age_s、stage 等）。
3. 在 SchedulerMetricsMixin 中实现 inflight 数据的采集逻辑：遍历 running batch、waiting queue 和 bootstrap 队列，从请求的 time_stats 中读取 entry_time 并基于 perf_counter 计算 age_s，构造 dict 列表。
4. 更新 docker/version.txt 从 nightly-dev-20260608b 到 nightly-dev-20260612a，对应新补丁版本。

关键文件：

- docker/patch/latest/sglang.patch（模块 推理引擎；类别 source；类型 observability；符号 GetLoadsReqInput, GetLoadsReqOutput, inflight, SchedulerMetricsMixin）：核心变更：修改 sglang 源码增加 inflight 监控
- docker/version.txt（模块 部署脚本；类别 config；类型 configuration）：同步更新 Docker 版本号指向新补丁

关键符号：update_device_timer

关键源码片段

docker/patch/latest/sglang.patch

核心变更：修改 sglang 源码增加 inflight 监控

```
# GetLoadsReqInput 中新增 inflight 枚举值
VALID_SECTIONS = frozenset({
    "core", "memory", "spec", "lora", "disagg", "queues",
    "inflight", # 新增：请求级队列明细
    "all"
})

# GetLoadsReqOutput 中新增 inflight 字段
@dataclass
class GetLoadsReqOutput(BaseReq):
    core: Optional[CoreMetrics] = None
    memory: Optional[MemoryMetrics] = None
    spec: Optional[SpecMetrics] = None
    lora: Optional[LoRAMetrics] = None
    disaggregation: Optional[DisaggregationMetrics] = None
    queues: Optional[QueueMetrics] = None
    # 请求级队列明细，仅当 include 包含 "inflight" 或 "all" 时填充
    # 每个元素为 {"name": str, "num_reqs": int, "reqs": List[Dict]}
    # reqs 中的字典包含 rid、bootstrap_room、seqlen、age_s、stage 等字段
    inflight: Optional[List[Dict[str, Any]]] = None

# SchedulerMetricsMixin 中 inflight 数据采集（部分）
if include_all or "inflight" in include:
    now_perf = time.perf_counter()
    inflight_queues = [
        ("running", self.running_batch.reqs, None),
    ]
    if self.disaggregation_mode == DisaggregationMode.PREFILL:
        inflight_queues += [
            ("waiting", self.waiting_queue, "wait_queue_entry_time"),
            ("bootstrap", self.disagg_prefill_bootstrap_queue.queue, "prefill_queue_entry_time"),
        ]
    inflight = []
    for name, req_list, entry_time_field in inflight_queues:
        reqs_detail = []
        for req in req_list:
            age_s = now_perf - getattr(req.time_stats, entry_time_field, now_perf)
            reqs_detail.append({
                "rid": req.rid,
                "age_s": age_s,
                "stage": req.stage if hasattr(req, 'stage') else None,
                # ... 其他字段
            })
        inflight.append({"name": name, "num_reqs": len(reqs_detail), "reqs": reqs_detail})
    output.inflight = inflight or None
```

评论区精华

本 PR 无公开 review 评论。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 性能风险：inflight 数据采集在每次请求端点时遍历所有队列，高并发时可能增加 scheduler 延迟。但数据仅在显式请求时采集，非持续运行，风险可控。
  2. 兼容性：新增 'inflight' 枚举和输出字段，不影响旧客户端。
  3. 测试覆盖：补丁修改未经自动化测试验证，主要依赖手工测试和镜像构建。 - 影响：用户可通过 ?include=inflight 参数获取更细粒度的请求状态，提升运维可观测性。影响范围限于使用更新后 Docker 镜像的部署。 - 风险标记：潜在性能开销，仅 Docker 环境验证，缺少自动化测试

关联脉络

- PR #2056 Use /v1/loads to re-abort server: PR 2056 同样修改了 /v1/loads 端点相关逻辑，添加 inflight 细节后可配合更精确的重试策略。