PR #26480 完整报告

sgl-project/sglang

feat(agentic router, 1/N): Add LoadBasedPolicy

合并时间：2026-06-06 00:13

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26480

执行摘要

- 一句话：新增 LoadBasedPolicy 路由策略，按最低活跃负载选择 worker
- 推荐动作：该 PR 值得精读，尤其是 LoadBasedPolicy 的实现简洁且符合 Policy trait 约定，可作为自定义路由策略的范本。Review 中提出的 herd effect 和测试 flaky 问题是关注的焦点，建议在实际部署前评估并解决这些潜在风险。此外，SelectionContext 中的 routing_key 字段暗示了未来的扩展方向，值得留意后续工作。

功能与动机

虽然 PR body 未填写具体的动机描述，但从代码变更意图判断，引入负载感知策略是为了使路由器能在多个 worker 之间基于实时负载更均匀地分配请求，避免部分 worker 过载而其它空闲，从而提升集群的整体吞吐和资源利用率。PR 标题中的 agentic router, 1/N 表明这是系列工作中的第一步。

实现拆解

1. 配置类型扩展（types.rs）：在 PolicyKind 枚举中增加 LoadBased 变体，对应 CLI 参数 --policy load_based，并更新相关文档注释。
2. 模块注册与上下文预扩展（mod.rs）：公开声明 pub mod load_based，使新策略模块可被发现；同时在 SelectionContext 结构体中预留 routing_key 字段（通过 with_routing_key 构造），为后续 cache-aware 等策略提供路由键信息（当前 LoadBasedPolicy 未使用该字段）。
3. 核心策略实现（load_based.rs）：定义 LoadBasedPolicy 结构体，实现 Policy trait 的 select 方法，内部调用 pick_min_load —— 使用 min_by_key 从候选 worker 中选出 active_load() 最小的实例；当多个 worker 负载相同时，选择切片中顺序靠前的（确定性的）。同时附带两个单元测试：空列表返回 None、负载倾斜时正确选中最空闲的 worker。
4. 工厂函数注册（factory.rs）：在 build_policy 和 build_policy_kind_only 中增加 PolicyKind::LoadBased 分支，使其能够构造 LoadBasedPolicy 实例。对应的单元测试 load_based_builds_via_factory 验证工厂能正确创建并注册该策略。
5. CLI 解析测试（cli.rs）：新增 parses_load_based_policy 测试，确保 --policy load_based 能正确解析为 PolicyKind::LoadBased。
6. 端到端验收测试（test_load_based_policy.py）：编写需要真实 GPU 的慢测试 test_load_based_routes_to_the_cooler_worker，通过启动两个 worker、模拟一个长请求

使一个 worker 进入繁忙状态，然后发送短请求验证路由器确实将短请求路由到了空闲 worker。

关键文件：

- *experimental/sgl-router/src/policies/load_based.rs*（模块 路由策略；类别 source；类型 entrypoint；符号 new, pick_min_load, select, empty_returns_none）：核心策略实现文件，定义了 LoadBasedPolicy 结构体、pick_min_load 方法以及 Policy trait 的实现，并包含单元测试。
- *experimental/sgl-router/tests/e2e/chat_completions/test_load_based_policy.py*（模块 集成测试；类别 test；类型 test-coverage；符号 _chat, _active_prefill_loads, _success_counts, _wait_for_busy_worker）：端到端验收测试，验证在真实 GPU 环境中 load_based 策略能正确将请求路由到空闲 worker，确保功能正确性。
- *experimental/sgl-router/src/policies/mod.rs*（模块 路由策略；类别 source；类型 entrypoint；符号 with_routing_key, routing_key）：新增 pub mod load_based 声明，并在 SelectionContext 中增加 routing_key 字段，为后续策略扩展提供基础。
- *experimental/sgl-router/src/policies/factory.rs*（模块 路由策略；类别 source；类型 entrypoint；符号 load_based_builds_via_factory）：在 build_policy 和 build_policy_kind_only 中增加 LoadBased 分支，确保工厂能构造新策略。
- *experimental/sgl-router/src/config/cli.rs*（模块 配置；类别 source；类型 configuration；符号 parses_load_based_policy）：新增 CLI 解析测试，验证 --policy load_based 能正确映射到 PolicyKind::LoadBased。
- *experimental/sgl-router/src/config/types.rs*（模块 配置；类别 source；类型 configuration）：定义 PolicyKind 枚举的新变体 LoadBased，使配置系统识别新策略。

关键符号：LoadBasedPolicy::new, LoadBasedPolicy::pick_min_load, LoadBasedPolicy::select (impl Policy), SelectionContext::with_routing_key, SelectionContext::routing_key, build_policy (factory.rs), build_policy_kind_only (factory.rs), test_load_based_routes_to_the_cooler_worker (E2E test), empty_returns_none (unit test), picks_lowest_active_load (unit test), parses_load_based_policy (CLI test), load_based_builds_via_factory (factory test)

关键源码片段

experimental/sgl-router/src/policies/load_based.rs

核心策略实现文件，定义了 LoadBasedPolicy 结构体、pick_min_load 方法以及 Policy trait 的实现，并包含单元测试。

```
// SPDX-FileCopyrightText: Copyright (c) 2026 The SGLang Authors
// SPDX-License-Identifier: Apache-2.0

use crate::policies::{Policy, SelectionContext};
use crate::workers::Worker;
use std::sync::Arc;

/// Deterministic load-based policy.
```

```rust
/// Chooses the candidate with the lowest current `Worker::active_load`.
/// Ties follow the candidate slice order, which is registry-dependent.
#[derive(Debug, Default)]
pub struct LoadBasedPolicy;

impl LoadBasedPolicy {
    /// 创建一个新的空策略实例
    pub fn new() -> Self {
        Self
    }

    /// 从 worker 切片中选出 `active_load` 最小的实例
    /// 平局时选择切片中靠前的 worker（确定性的，可能引起羊群效应）
    pub fn pick_min_load(workers: &[Arc<Worker>]) -> Option<Arc<Worker>> {
        workers
            .iter()
            .min_by_key(|w| w.active_load())
            .map(Arc::clone)
    }
}

impl Policy for LoadBasedPolicy {
    fn select(&self, workers: &[Arc<Worker>], _ctx: &SelectionContext<'_>) ->
    Option<Arc<Worker>> {
        Self::pick_min_load(workers)
    }
}

#[cfg(test)]
mod tests {
    use super::*;
    use crate::discovery::{ModelId, WorkerId, WorkerMode, WorkerSpec};

    fn worker(id: &str) -> Arc<Worker> {
        Arc::new(Worker::new(WorkerSpec {
            id: WorkerId(id.into()),
            url: format!("http://{id}:30000"),
            mode: WorkerMode::Plain,
            model_ids: vec![ModelId("tiny".into())],
            bootstrap_port: None,
        }))
    }

    #[test]
    fn empty_returns_none() {
        let policy = LoadBasedPolicy::new();
        let model = ModelId("tiny".into());
        let ctx = SelectionContext::new(&model, None);
        assert!(policy.select(&[], &ctx).is_none());
```

```
    }

    #[test]
    fn picks_lowest_active_load() {
        let policy = LoadBasedPolicy::new();
        let model = ModelId("tiny".into());
        let ctx = SelectionContext::new(&model, None);
        let w0 = worker("w0");
        let w1 = worker("w1");
        let _g0 = w0.load_guard(); // 模拟 w0 负载增加
        assert_eq!(
            policy.select(&[w0, Arc::clone(&w1)], &ctx).unwrap().id,
            w1.id
        );
    }
}
```

experimental/sgl-router/src/policies/mod.rs

新增 pub mod load_based 声明，并在 SelectionContext 中增加 routing_key 字段，为后续策略扩展提供基础。

```
// 在 mod.rs 中新增模块声明
pub mod load_based;

// SelectionContext 中新增 routing_key 字段及构造 / 访问方法
pub struct SelectionContext<'a> {
    model: &'a ModelId,
    request_body: Option<&'a [u8]>,
    routing_key: Option<&'a str>, // 新增：用于缓存感知等策略的路由键
}

impl<'a> SelectionContext<'a> {
    // 保留原有 new
    pub fn new(model: &'a ModelId, request_body: Option<&'a [u8]>) -> Self {
        Self { model, request_body, routing_key: None }
    }
    // 新增构造方法：同时提供 routing_key
    pub fn with_routing_key(
        model: &'a ModelId,
        request_body: Option<&'a [u8]>,
        routing_key: Option<&'a str>,
    ) -> Self {
        Self { model, request_body, routing_key }
    }

    pub fn routing_key(&self) -> Option<&str> {
        self.routing_key
    }
}
```

评论区精华

Review 评论主要提出两个中等优先级问题：

- 高并发下的羊群效应（herd effect）：pick_min_load 使用 min_by_key，当多个 worker 具有相同最低负载时始终选择切片中第一个，在高并发场景下可能导致多个请求几乎同时被路由到同一个 worker（因为活跃负载计数还未更新），从而造成瞬时倾斜。建议引入随机平局打破机制（random tie-breaking）。
- 测试潜在的不稳定性：E2E 测试通过轮询 sgl_router_active_load{kind="prefill_tokens"} 指标来判断 worker 是否繁忙，但小模型预填阶段极快（<10ms），而轮询间隔为 200ms，很可能错过瞬态预填阶段导致超时失败。建议改为基于并发请求数或其他更稳定的指标。
  - 高并发下的羊群效应（Herd Effect）（design）: 建议引入随机 t ie-breaking，但 PR 中未修复，评论未得到作者回应，状态为未解决。
  - E2E 测试潜在的不稳定性 (testing): 建议改用基于并发请求数或其他更鲁棒的指标，PR 中未修改，状态未解决。

风险与影响

- 风险：
  1. 羊群效应（herd effect）：在高并发且多个 worker 负载相同时，确定性选择第一个 worker 可能导致负载集中，需要后续引入随机化或加权策略缓解。
  2. 测试 flaky 风险：E2E 测试依赖的预填指标窗口过窄，在多变的 CI 环境下可能失败，需降低测试可靠性或改用更鲁棒的忙判断逻辑。
  3. 未使用新字段：SelectionContext 中新增的 routing_key 字段未被 LoadBasedPolicy 使用，存在一定的误导性，但属于预留设计。
  4. 影响范围有限：仅在配置为 load_based 策略时生效，不会影响已有路由策略，风险可控。
  - 影响：用户：运行 sgl-router 的用户现在可以通过 --policy load_based 启用负载感知路由，有助于在多个 backend worker 之间实现更均衡的流量分配。系统：新策略作为可选方案集成，不影响现有策略（round_robin、random、power_of_two、cache_aware_zmq）的执行路径。团队：作为 agentic router 特性的第一步，为后续更复杂的路由策略（如缓存感知、亲和性路由）提供了可扩展的框架和模式参考。
- 风险标记：高并发下潜在羊群效应，E2E 测试可能 flaky，未使用的 routing_key 字段可能造成误解

关联脉络

- 暂无明显关联 PR