PR #27073 完整报告

sgl-project/sglang

[router] Configure experimental sgl-router via CLI flags instead of a config file

合并时间：2026-06-05 10:02

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27073

PR 分析报告：sgl-router CLI 标志驱动改造

执行摘要

将实验性 Rust sgl-router 从配置文件驱动完整迁移到 CLI 标志驱动，移除 serde 依赖，统一多模型为单模型，并在同一 PR 中集成了 bigram 哈希和 metrics 增强。变更影响 43 个文件，测试全通过，风险可控。

功能与动机

PR 描述明确指出："The experimental Rust sgl-router was config-file-only: its sole CLI arg was --config <path> pointing at a TOML/YAML file." 这种设计缺乏灵活性，也与 Python 版本的 CLI 接口不一致。本 PR 的目标是让 sgl-router 像 Python sglang_router.launch_router 一样通过 --host/--port/--service-discovery/--selector 等标志启动，并固定为单一模型服务。

实现拆解

1. CLI 入口定义：新增 config/cli.rs，使用 clap 派生定义所有命令行标志，并在 Cli::into_config 中完成互斥校验、条件参数检查和 K8s 选择器解析。
2. 类型系统简化：config/types.rs 移除所有 serde 派生，Config.models 改为 Config.model，DiscoveryConfig 包装层内联为 DiscoveryBackend；PolicyKind 和 LogFormat 从 serde::Deserialize 改为 clap::ValueEnum，使非法值在解析时即被拒绝。K8s 选择器组合在构建时解析为 K8sDiscoveryMode 枚举，消除无效状态。
3. 验证适配：config/mod.rs 删除 Config::from_path，简化 validate 仅检查 model.id 非空和静态 URL 列表合法性（URL 格式、去重、scheme 检查）。测试代码改为直接构造 Config 实例，不再通过临时文件。
4. 全模块适配：更新策略工厂、API 路由（/v1/models）、tokenizer 加载器、观测性配置等所有引用旧类型的地方，删除 serde_yaml/toml/humantime-serde 依赖。
5. 附带增强（PR 后继提交）：
   - 支持 bigram token_ids 解码（wire.rs），适应 DeepSeek-V4 等 EAGLE 模型发布的 [t_i, t_{i+1}] 格式。
   - 添加 per-request 访问日志和 sgl_router_overlap_blocks 指标（cache_aware_zmq.rs + metrics registry）。

- 实现 compute_block_hashes_bigram（hash.rs），使 cache-aware 路由在 EAGLE 模型上也能正确计算块哈希匹配。

experimental/sgl-router/src/config/cli.rs

CLI 入口，新增全部命令行标志定义和 into_config 核心转换逻辑，是本次重构的中心文件。

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

//! 命令行界面。路由器完全通过标志配置 —— 没有配置文件。
//! [`Cli::into_config`] 将标志解析为验证后的 [`Config`]。

use anyhow::{anyhow, Result};
use clap::Parser;
use std::num::NonZeroU32;

use crate::config::{
    default_cb_cool_down, default_proxy_request_timeout_secs, default_stale_request_timeout_
    secs,
    resolve_mode, ActiveLoadConfig, CacheAwareConfig, CircuitBreakerConfig, Config,
    DiscoveryBackend, K8sDiscoveryConfig, LogFormat, ModelConfig, ObservabilityConfig,
    PolicyKind,
    ProxyConfig, ServerConfig, StaticUrlsDiscoveryConfig,
};

/// `sgl-router` — 轻量 KV 感知 OpenAI 兼容 SGLang worker 路由器。
///
/// 发现方式互斥：传递 `--worker-urls`（静态地址）或 `--service-discovery`
///  （Kubernetes EndpointSlice）—— 必须且仅需一种。
#[derive(Parser, Debug)]
#[command(
    name = "sgl-router",
    version,
    about = "Slim KV-aware OpenAI-compatible router for SGLang workers"
)]
pub struct Cli {
    // ---- server ----
    #[arg(long, default_value = "127.0.0.1")]
    pub host: String,
    #[arg(long, default_value_t = 30000)]
    pub port: u16,

    // ---- model (exactly one) ----
    #[arg(long)]
    pub model_id: String,
    /// Tokenizer 来源：本地路径或 HuggingFace repo id；省略时降级为 `--model-id`
    #[arg(long)]
    pub tokenizer_path: Option<String>,
    /// 路由策略：round_robin / random / power_of_two / cache_aware_zmq
```

```
    #[arg(long, value_enum, default_value = "round_robin")]
    pub policy: PolicyKind,

    // ---- circuit breaker (opt-in) ----
    #[arg(long)]
    pub cb_threshold: Option<NonZeroU32>,
    #[arg(long)]
    pub cb_cool_down_secs: Option<u64>,
    // ... 其余字段（缓存调优、发现、代理、日志等）
}

impl Cli {
    /// 将解析后的标志转换为验证过的 [`Config`]。
    pub fn into_config(self) -> Result<Config> {
        // 构建发现后端（互斥检查 + K8s 选择器解析）
        let discovery = self.build_discovery()?;
        // 构建模型配置（包含 tokenizer_path 降级逻辑）
        let model = self.with_model();
        // 可选断路器
        let circuit_breaker = self.cb_threshold.map(|threshold| CircuitBreakerConfig {
            threshold,
            cool_down_secs: self.cb_cool_down_secs.unwrap_or_else(default_cb_cool_down),
        });
        // ... 组装 Config 并验证
        let cfg = Config { /* ... */ };
        cfg.validate()?;
        Ok(cfg)
    }
}
```

experimental/sgl-router/src/config/types.rs

类型核心定义，移除了 serde 派生，Config.models → Config.model，DiscoveryBackend 直接内联，PolicyKind/LogFormat 改用 clap::ValueEnum。

```
use std::num::NonZeroU32;

/// 内存中路由器配置，由 [`cli::Cli::into_config`] 从 CLI 标志构建。
/// 路由器仅服务一个模型。
#[derive(Debug, Clone)]
pub struct Config {
    pub server: ServerConfig,
    pub observability: ObservabilityConfig,
    pub model: ModelConfig, // 单一模型（原 Vec<ModelConfig>）
    pub discovery: DiscoveryBackend, // 直接使用后端枚举（原 DiscoveryConfig 包装）
    pub proxy: ProxyConfig,
    pub active_load: ActiveLoadConfig,
}

/// 模型配置，每个路由器实例仅一个。
```

```rust
#[derive(Debug, Clone)]
pub struct ModelConfig {
    pub id: String,
    /// Tokenizer 来源路径或 HuggingFace repo id；省略时降级为 id
    pub tokenizer_path: String,
    pub policy: PolicyKind,
    pub circuit_breaker: Option<CircuitBreakerConfig>,
    /// cache_aware_zmq 策略专有调优
    pub cache_aware: Option<CacheAwareConfig>,
}

/// 路由策略枚举，改用 clap::ValueEnum 而非 serde，在 CLI 解析时失效。
#[derive(Debug, Clone, Copy, PartialEq, Eq, Default, clap::ValueEnum)]
pub enum PolicyKind {
    #[default]
    #[value(name = "round_robin")]
    RoundRobin,
    #[value(name = "random")]
    Random,
    #[value(name = "power_of_two")]
    PowerOfTwo,
    /// KV cache 感知路由，需要 tokenizer 和 ZMQ 事件
    #[value(name = "cache_aware_zmq")]
    CacheAwareZmq,
}
```

评论区精华

该 PR 未产生 review 讨论。作者在描述中自述了设计决策，其中值得注意的包括：

- “Discovery is mutually exclusive — --worker-urls (static) XOR --service-discovery (k8s), exactly one required.”
- “Knobs that require another flag (--cb-cool-down-secs without --cb-threshold; cache-aware knobs without --policy cache_aware_zmq) are rejected, not silently ignored.”

这些设计体现了“使无效状态不可表示”和“尽早报错”的原则。

风险与影响

- 配置兼容性：现有用户必须从 TOML/YAML 配置文件迁移到 CLI 标志，可能需要调整部署脚本。
- 单一模型：之前可以配置多个模型（尽管实际使用可能单一），现在强制单例，可能影响后续扩展。
- 附带变更风险：bigram 和 metrics 增强与核心 CLI 重构混合同一 PR，增大审阅和回退粒度。
- 测试覆盖：cargo test --all-targets 387 个测试全部通过，clippy 和 fmt 检查干净，CI（基础）通过。
- 影响范围：局限于 experimental/sgl-router 模块，不影响主推理路径。

关联脉络

本 PR 替代了 fork 分支的 #27069，是路由器现代化的关键一步。之前的历史 PR #27193（用 batch 携带的 attention plan marker 替换 skip_attn_backend_init）与本 PR 无直接关系，但反映了项目持续简化配置和减少特殊状态的趋势。另外 #27300（完善 CustomSpecAlgo 接口）和 #27247（AMD 采样修复）等展示了对实验性模块的持续维护风格，本 PR 与之相似，致力于提升代码质量和接口一致性。