

PR #39322 完整报告

sgl-project/sglang

[Router] Extract worker selection into policies::selection (no behavior change)

合并时间: 2026-09-15 15:49

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/39322>

执行摘要

本 PR 把 sgl-router 中 prefill 与 decode 两条 worker 选路阶梯, 从约 900 行的 `chat_completions` HTTP handler 中原样抽到新模块 `experimental/sgl-router/src/policies/selection.rs` (+845 行), 对外暴露 `select_prefill_worker` (返回 `Result`) 与 `select_decode_peer` (返回 `Option`) 两个 `pub(crate)` 入口, 并声明零行为变更。handler 侧删除 461 行闭包与辅助函数, `std::cell::Cell` 被 `Selector` 结构体字段取代, 状态码映射仍留在路由层。该 PR 是 #39168 / #39169 / #39170 三个新选路层级的公共地基, 同时把原本只能写成端到端 HTTP 测试的阶梯级断言变为可写的单元测试。

功能与动机

为什么要做: 两条选路阶梯此前是约 900 行 `chat_completions` handler 里唯一的非测试调用者。作者在 PR 描述中给出的核心理由是——

The assertions worth writing are about a ladder as a whole ("a saturated fleet still routes", "a domain that fits is never passed over for one that only fits by relaxing capacity"), not about any single rung. Inside the handler those can only be written as end-to-end HTTP tests, or not at all.

要解决的具体问题:

1. 可测性: 阶梯的价值断言是整体性质的, 写在 HTTP handler 内部只能退化为端到端测试, 或者根本写不出来。
2. `Cell` 只是返回类型问题的伪装: `Cell<PolicySelectionFailureReason>` 存在的唯一目的, 是把失败原因从闭包里 "偷带" 出来。
3. 后续三个层级无处安放: #39168、#39169、#39170 每个都会给 prefill 阶梯加一层; 按原位置继续写, 就是 1100 行 handler 加第二个 `Cell`。
4. 既有不对称已经发生过: 把两条阶梯分置在不同代码位置, 正是 queue gate 在一条路径上带两级兜底、另一条只有一级而无人察觉的原因。

范围声明: `selection.rs` 新增 +845, `policies/mod.rs` +1, `server/routes/chat.rs` +39 / -461; 两个 commit 中第一个是机械验证过的纯搬迁, 第二个是 review 追加的测试与结构收口。

实现拆解

1. 新增模块骨架 (`policies/selection.rs`) : 定义 `PrefillSelectionInputs<'a>` 与 `DecodeSelectionInputs<'a>` 两个 `pub(crate)` 输入结构体, 按 " 协作者在后、单请求事实在前 " 排列字段, 显式持有 `policy / bucket_selector / metrics`, 使 `policies/` 不再依赖 `server::app_context`。入口为 `select_prefill_worker(...) -> Result<Arc<Worker>, PolicySelectionFailureReason>` 与 `select_decode_peer(...) -> Option<Arc<Worker>>`。
2. 用状态结构体替换 `Cell`: `struct Selector { inputs, bucket_request, failure_reason }` 跨越原先的闭包边界, `run()` 返回 `Option<Arc<Worker>>`, `select_prefill_worker` 用 `selected.ok_or(selector.failure_reason)` 收口。语义约定为 " 每层放弃时覆写 `reason`, 未记录失败的层返回 `None` 会保留上一层的原因 ", 因此上报的永远是最后一个记录过原因的层级。
3. 阶梯逻辑逐层搬迁: `Selector::run` 依次执行 `cache_winner()` (`Cache-Aware` 的有界全局候选、指标记录与 `commit_prefill_selection`)、`global_affinity_probe` (全局会话亲和探测, 命中后经 `prefill_affinity_domain` 在本 `Bucket` 内以 `select_in_domain(..., true, false, false)` 重建 `backup`), 以及按 `PolicyKind / SessionAffinityMode` 分流的两遍 `select_domains`; `base_context()` 统一构造 `SelectionContext`, `select_in_domain()` 调 `propose_prefill` 加 `resolve_prefill_admitted / resolve_prefill`, 域耗尽时写 `PrefillAdmissionExhausted`。 `prefill_policy_reason` 与 `projected_decode_kv_tokens` 随各自测试一并迁入。
4. handler 瘦身与模块注册: `server/routes/chat.rs` 删除 461 行闭包、两个辅助函数与相关导入 (含 `std::cell::Cell`), `chat_completions` 现在只组装输入结构体再调用新入口; `prefill` 失败经 `.map_err(|reason| policy_selection_failed(&ctx, &model_str, reason))?` 映射 `ApiError`, `decode` 失败映射 `ApiError::NoDecodeWorkersAvailable`。`policies/mod.rs` 仅新增 `pub mod selection;` 一行, 可见性保持 `pub(crate)`。
5. 测试与验证配套: 搬迁 `commit` 保持两文件 `#[test]` 数守恒 (33), 并用 `token` 流逐段对比证明纯移动; `review commit` 追加 5 个阶梯级测试 (饱和 `prefill / decode` 集群经容量兜底仍可路由、采样结果不会落到被 `admission` 拒绝的 `worker`、空 `fleet` 报 `ProposalEmpty`), 逐个做 `mutation check`。 `cargo test 702 pass / 0 fail`, `cargo fmt --check` 与 `cargo clippy --all-targets` 干净, `buckets.rs:167` 的既存 `lint` 已由 `#39259` 在 `main` 修复。

关键源码片段

`experimental/sgl-router/src/server/routes/chat.rs`

`chat_completions` 是原阶梯的唯一非测试调用者, 本 PR 在此删除 461 行闭包与辅助函数、移除 `Cell` 导入, 改为组装输入结构体并调用新入口, 同时保留状态码映射职责。

```
// server/routes/chat.rs: handler 现在只做编排 —— parse -> tokenize -> select -> proxy。  
// 两条阶梯的输入在这里组装, 失败原因在这里映射成 HTTP 状态码 (模块不碰 HTTP)。
```

```
// `select_prefill_worker` 会在 Bucket 分区关闭时把该模式归约为 Bucket,  
// 因此这里直接传配置值, 无需再复制一份归约逻辑。
```

```
let session_affinity_mode = ctx  
    .config  
    .model  
    .affinity
```

```

.as_ref()
.map(|config| config.session_affinity_mode)
.unwrap_or(SessionAffinityMode::Bucket);

let worker = select_prefill_worker(&PrefillSelectionInputs {
    policy: policy.as_ref(),
    policy_kind: ctx.config.model.policy,
    bucket_selector: ctx.bucket_selector.as_ref(),
    metrics: ctx.metrics.as_ref(),
    model_id: &model_id,
    body: Some(&body),
    routing_key,
    session_id,
    request_input_tokens,
    request_tokens: request_tokens.as_ref().map(|tokens| tokens.ids.as_slice()),
    external_prefix: external_prefix.as_ref(),
    load_snapshot: load_snapshot.as_ref(),
    workers: &workers,
    ttft_slo_ms,
    tps_slo,
    session_affinity_mode,
})
// 失败原因到状态码的映射留在路由层：这是模块划界的边界条件。
.map_err(|reason| policy_selection_failed(&ctx, &model_str, reason));

// Decode 选择发生在 Final P 之后；普通聚合 worker 直接跳过。
//
// PD 模式下解析不到 decode peer (NoDecodeWorkersAvailable) 会冒泡成 503,
// 便于运维对 prefill / decode 池失衡告警。
let decode_peer: Option<Arc<Worker>> = if worker.mode() == WorkerMode::Prefill {
    let decode_workers = resolver.decode_candidates(&model_id).map_err(|e| match e {
        PdResolveError::NoHealthyWorkers => ApiError::NoHealthyWorkers {
            model: model_str.clone(),
        },
        PdResolveError::NoDecodeWorkersAvailable => ApiError::NoDecodeWorkersAvailable {
            model: model_str.clone(),
        },
        PdResolveError::NoPrefillWorkersAvailable => ApiError::NoPrefillWorkersAvailable {
            model: model_str.clone(),
        },
    });
} else {
    None
};
Some(
    select_decode_peer(&DecodeSelectionInputs {
        decode_policy_kind: ctx.config.model.decode_policy,
        bucket_selector: ctx.bucket_selector.as_ref(),
        model_id: &model_id,
        prefill_url: &worker.url,
        decode_workers: &decode_workers,
        request_input_tokens,
    })
)

```

```
        requested_max_output_tokens,
        ttft_slo_ms,
        tps_slo,
        load_snapshot: load_snapshot.as_ref(),
    })
    // decode 阶梯只有一种失败形态，所以入口返回 Option 而不是 Result:
    // 为对称而造错误类型只会成为噪音。
    .ok_or_else(|| ApiError::NoDecodeWorkersAvailable {
        model: model_str.clone(),
    })?,
)
} else {
    None
};
```

评论区精华

sherlockwu (APPROVED) : "refactor looks good"

ShangmingCai (APPROVED) : "Logic looks good. Need to fix conflict before merge.
CC: @Bo-Vincent for a double-check review, will this change break your policy?"

Bo-Vincent (issue comment) : "Took a look at the code and ran a quick test. It
preserves the existing policy order and selection logic. LGTM!"

除了正式 review，PR 描述本身承载了本轮最有价值的设计论证，几处关键表态值得直接引用：

The **Cell** was a return-type problem in disguise. **Cell<PolicySelectionFailureReason>** existed to smuggle a value out of a closure; it is now a struct field, and **std::cell::Cell** is no longer imported by the handler at all.

Splitting their ladders across a module boundary is what let the queue gate (#39168) ship with a two-tier fallback on one path and a single tier on the other, with nobody reading either file able to see the asymmetry.

Decode returns **Option** rather than prefill's **Result**: it has exactly one failure mode, so there is no reason to name it, and a symmetric-for-symmetry's-sake error type would be noise.

风险与影响

风险点	具体位置	说明
第二个 commit 并非纯搬迁	commit 1f645cad	Result 取代 invalid-state 结构体、对称输入结构体、domain 惰性物化 (prefill_domains 闭包) 都是搬迁之外的新动作，其中惰性物化改变了 domain 的分配时机，是本 PR 唯一需要逐行核对的语义面

风险点	具体位置	说明
失败原因依赖覆写约定	<code>Selector.failure_reason</code>	" 未记录失败者保留上一层原因 " 是口头约定；后续 #39168 / #39169 / #39170 新增层级若忘记赋值，会报出过期原因，且只在最终选路失败时可见
隐式 panic 契约	<code>select_in_domain</code> 中的 <code>expect("shared prefill admission requires a load snapshot")</code>	模块成为入口后，若出现非 HTTP 调用方且未传 <code>load_snapshot</code> 会直接 panic；目前靠 <code>needs_load_snapshot</code> 与 <code>uses_shared_prefill_admission</code> 默认同步来保证
分支基线落后于 main	<code>buckets.rs:167</code>	该 lint 需由 #39259 修复，ShangmingCai 已要求先解冲突；重解冲突时可能引入偏差
兼容性	无	不涉及公开 API、配置键或线格式变更， <code>pub(crate)</code> 可见性与 <code>PolicySelectionFailureReason</code> 现有可见性一致

影响面：用户侧完全无感知，路由结果、状态码与 `x-ssl-decode-url` header 行为均保持；系统侧 router 的 chat handler 从约 900 行降到约 440 行编排层，两条阶梯首次获得独立模块边界与单元测试能力；团队侧这是 4 步 stack 的第 1 步，直接决定后续三个选路层级的落点质量，并把 " 饱和集群仍能路由 " 这类横切断言固化为可回归测试。

关联脉络

- 向前的 stack：本 PR 是描述中 4 步计划的第 1 步，后续为 #39168 (queue gate / --worker-queue-limit)、#39169 (saturation pin / --saturation-queue-floor) 与 #39170 (candidate sampling / --min-load-choices)，后两者是 #39168 之上的并列兄弟。本 PR 提供的 Selector 层级结构与 " 采样不落到被拒 worker " 测试，正是为它们预置的护栏。
- 基线依赖：#39259 在 main 上修复了 `buckets.rs:167` 的 clippy lint，而本分支尚未包含，这就是 ShangmingCai 要求先解冲突的直接原因。
- 同区域的重构序列：#39108 (Router KV-event 存储层级，标题自带 1/4 编号) 同属 `experimental/ssl-router/src/policies/` 的分解式改造；两者共同体现该仓库对路由核心的演进策略——把大改拆成可审计的小步 PR，每一步保持行为不变并补齐测试。
- 外部对照：作者引用 NVIDIA Dynamo 的 `lib/kv-router/` 作为同类分层先例——worker 选择独立成模块与测试、worker role 作为一条选路路径的维度而非 " 每个 role 一条阶梯 "，与本 PR 把 prefill / decode 放进同一模块的取舍方向一致。