

PR #39004 完整报告

sgl-project/sglang

[router] Resolve a wire protocol per worker at registration

合并时间: 2026-09-16 07:54

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/39004>

执行摘要

本 PR 是 sgl-router ([experimental/sgl-router](#), Rust 实现) 为 cleartext h2c 转发铺路的第一层地基: 在 worker 注册阶段, 把「引擎自报 `enable_http2`」与「路由器是否明文拨号该 worker」两个输入解析为 `WireProtocol` (`Http1 / H2c`), 并作为 不可变字段盖戳进 `Worker`。PR 刻意不消费该值——body 明确说 "Nothing reads the resolved value yet, so this changes no bytes on the wire"——线上转发行为零变化, 但解析规则、存储位置与测试契约全部就位, 供后继 PR #39006 直接消费。核心设计是 per-worker 解析 (避免 fleet-wide 客户端「先写先赢」拖累整集群) 与 `Http1` 全场景兜底 (h2c 是 prior-knowledge 协议, 禁止向不支持者误发)。

功能与动机

PR body 将本变更定位为 h2c 转发的 groundwork: "split out so the resolution rules can be reviewed on their own"——把「决策与存储」从「消费决策」中拆出, 先让解析规则单独过审, 再让 #39006 落地实际转发。选择 per-worker 而非 fleet-wide 解析的动机有二:

- 引擎的 `--enable-http2` 是启动参数, 只有重启才可改, 而重启必然触发 worker 重新注册, 因此解析值在 worker 生命周期内不会过期 (这是字段不可变、无需 `AtomicU8` 的根本原因)。
- h2c 是 prior-knowledge 协议、没有任何协商, 发给不支持它的引擎会让请求全部失败; `Http1` 是唯一能被所有引擎接受的协议, 因此成为默认值与兜底。

此外作者明确接受一个失败模式: `/server_info` 不响应的 worker 无法读取协议, 将以 `Http1` 注册并终生不被重新解析——这是吞吐量成本而非正确性错误, 且被 `warming_worker_without_server_info_registers_on_http1` 测试钉死。

实现拆解

- 协议类型与 Worker 字段 ([experimental/sgl-router/src/workers/worker.rs](#)): 新增 `WireProtocol` 两变体枚举, `Http1` 标记 `#[default]`; `Worker` 增加普通不可变字段 `protocol`, `with_cb_config` 新增第三个参数 `protocol: WireProtocol`, 新增访问器 `protocol()`, `Debug` 输出同步带上协议。 `protocol_is_carried_from_construction` 单测验证默认值与显式值都能从构造携带。
- 内省数据源扩展 ([experimental/sgl-router/src/workers/introspect.rs](#)): `ServerInfoBody` 新增 `enable_http2: Option<bool>` 字段, `#[serde(default)]` 保证老引擎缺字段时反序列化为 `None` (区别于显式 `false`); `fetch` 将其投影进

`ServerInfo.enable_http2`。三个单测分别锁定 `true` / `false` / `absent` 三种取值，其中 `absent` 必须读作 `None` 而非解析失败。

3. 注册前解析决策 (`experimental/sgl-router/src/workers/manager.rs`)：新增 `dials_cleartext` (无 `scheme` URL 先补 `http://` 再解析，规避 `url` 库把 `worker-0.engines.svc` 当 `scheme` 的坑)、`resolve_protocol` (只有 `(Some(true), Some(true))` 组合升级 `H2c`，其余一律 `Http1`)、`log_protocol_resolution` (四类场景分级日志：`h2c` 升级 `info`、`TLS` 降级 `info`、`URL` 不可解析 `warn`、`flag` 未读到 `info`)。`register_one` 在注册落库前完成解析并把协议传给 `add_with_cb`；`reconcile_unresolved_workers` 复用同一入口，临时性 `introspection` 失败后 `reconcile` 重新注册会顺带把协议带回来。
4. 注册表与模块出口 (`experimental/sgl-router/src/workers/registry.rs`、`mod.rs`)：`add_with_cb` 增加 `protocol: WireProtocol` 参数并在构造 `Worker` 时传入，`add` 走 `WireProtocol::default()`；`mod.rs` 对外重新导出 `WireProtocol`。
5. 测试配套：组件测试新增 4 个用例——`manager_resolves_h2c_protocol_from_introspection` (正向升级)、`manager_defaults_http1_when_enable_http2_absent` (缺失兜底)、`manager_registers_worker_that_serves_server_info_only` (无 `/model_info` 的 fallback 链)、`manager_resolves_protocol_per_worker_no_fleet_lock` (核心回归：先注册的 `Http1` worker 不拖累后注册的 `h2c` worker)；`warming_worker_without_server_info_registers_on_http1` 锁定「接受失败模式」且 `reconcile` 间隔调至 600 s，排除修复循环晚到的干扰。辅助函数 `wait_for_protocol` 同时要求 `model_ids` 非空，防止「从不读取协议的实现」因 `Http1` 是默认值而假阳性通过。

experimental/sgl-router/src/workers/manager.rs

协议解析决策的核心载体：`dials_cleartext` 处理无 `scheme` URL 的规范化坑，`resolve_protocol` 定义完整输入矩阵，`log_protocol_resolution` 按四类场景分级打日志；`register_one` 与 `reconcile_unresolved_workers` 共用同一解析入口。

```
/// 判断路由器是否以明文方式拨号 `worker_url`。
///
/// 不能直接写 `Url::parse(url)?.scheme() == "http"`：无 scheme 的
/// `worker-0.engines.svc:8000` 会被 `url` 解析成 scheme 为
/// `worker-0.engines.svc` 的 URL，从而对“明文短名”这种形态误报
/// “非 cleartext”。这里沿用 `parse_bootstrap_host` 的规范化思路——
/// 补上显式 `http://` 后重试解析，再读 scheme。无 scheme 的 URL
/// 因此被判定为明文：该结论不会真正到达线上（proxy 侧仍是裸
/// `Url::parse`，无 scheme 请求会在 `join(path)` 阶段失败），
/// 所以返回 `Some(true)` 只影响日志归因，不会误发 h2c。
fn dials_cleartext(worker_url: &str) -> Option<bool> {
    if worker_url.contains(":/") {
        return url::Url::parse(worker_url)
            .ok()
            .map(|u| u.scheme() == "http");
    }
    url::Url::parse(&format!("http://{worker_url}"))
        .ok()
}
```

```

        .map(!_! true)
    }

    /// 推导某个 worker 的转发协议，输入为引擎自报的 `enable_http2`
    /// 与 `dials_cleartext` 的结果（URL 解析失败时为 `None`）。
    ///
    /// 两个输入在 worker 生命周期内都固定：引擎不重启就无法改
    /// `--enable-http2`，而重启必然触发重新注册，所以本函数每个
    /// worker 只执行一次、且在注册落库之前完成，解析值不会过期。
    ///
    /// 只有 `(Some(true), Some(true))` 才升级到 `H2c`——h2c 是
    /// prior-knowledge 协议、没有协商，发给不支持它的引擎会让每个
    /// 请求失败，cleartext 门槛就是控制该风险的关键。其余所有组合
    /// 一律回落到 `Http1`，即所有引擎都接受的绝对安全默认值。
    fn resolve_protocol(enable_http2: Option<bool>, cleartext: Option<bool>) -> WireProtocol {
        match (enable_http2, cleartext) {
            (Some(true), Some(true)) => WireProtocol::H2c,
            _ => WireProtocol::Http1,
        }
    }
}

```

experimental/sgl-router/src/workers/worker.rs

WireProtocol 两变体枚举与 Worker 不可变 protocol 字段的落点，是整个 PR 的数据契约；注释明确解释了 h2c 的 prior-knowledge 语义与 Http1 兜底的不对称性。

```

    /// 路由器发给某个 worker 的转发协议。
    ///
    /// 在 worker 生命周期内固定不变：它由 `manager::resolve_protocol`
    /// 从引擎的 `--enable-http2` 启动参数和拨号 URL 的 scheme 推导而来，
    /// 两者在进程运行期间都不会改变。默认值的不对称性是关键设计：
    /// HTTP/1.1 被所有引擎接受，而 h2c 属于 prior-knowledge 模式，
    /// 发给不支持它的引擎会直接失败——所以默认永远落在 `Http1`。
    #[derive(Debug, Clone, Copy, PartialEq, Eq, Default)]
    pub enum WireProtocol {
        /// HTTP/1.1，明文与 TLS 下都可用，是所有引擎的安全兜底。
        ///
        /// 注意它不是 ALPN 协商出来的：`Cargo.toml` 用
        /// `default-features = false` 构建 request 且未启用 `http2`
        /// feature，转发客户端只会通告 `http/1.1`，因此运行
        /// `--enable-http2` 的 TLS 引擎依然走 HTTP/1.1。
        #[default]
        Http1,
        /// 明文 HTTP/2 with prior knowledge (h2c)。仅当引擎自报
        /// `enable_http2 = true` 且 URL 为 cleartext 时才使用。
        H2c,
    }
}

```

评论区精华

唯一审阅人 [sherlockwu](#) 直接 APPROVED ("lgm!")，未产生逐行评论。实质性的设计论证集中在 PR body 中作者主动标出的两个「值得审阅者注意」的细节，构成一次高质量的设计自审：

dials_cleartext 不能写成 `Url::parse(url)?.scheme() == "http"`：无 scheme 的 `worker-0.engines.svc:8000` 会被 url 库解析成 scheme 为 `worker-0.engines.svc` 的形态，对「明文短名」误判为「非 cleartext」。作者选择与 `parse_bootstrap_host` 一致的规范化——补显式 `http://` 重试再读 scheme。

Http1 在 TLS 下真的就是 HTTP/1.1：Cargo.toml 以 `default-features = false` 构建 request 且未启用 `http2`，转发客户端只通告 `http/1.1`，因此 `(Some(true), Some(false)) => Http1` 分支目前只是 fallback 而非协商；#39006 将启用 request 的 `http2` feature 并恢复 ALPN 行为。

风险与影响

技术风险

1. 接受的失败模式（作者明确承认）：`/server_info` 不响应的 worker 会以 `Http1` 注册进模型池，且 `reconcile_unresolved_workers` 以空 `model_ids` 为键、不会重新解析其协议——该 worker 终生走 HTTP/1.1。这是吞吐量损失而非正确性错误，但运维上难以感知。
2. TLS 引擎的 ALPN 缺口：`https:// + --enable-http2` 的 worker 会继续走 HTTP/1.1，直到 #39006 启用 request `http2` feature；合入顺序颠倒可能出现协议字段与客户端能力不一致的中间态。
3. 信任引擎自报字段：协议决策完全依赖 `/server_info` 的 `enable_http2`，若引擎误报 `true` 且 URL 为明文，消费该值的转发层会向不支持 h2c 的端点发 h2c；本 PR 尚未消费该值，风险推迟到 #39006。
4. API 签名变更：`add_with_cb` 与 `with_cb_config` 均增加参数，波及全部调用点，属编译期可见变更。

影响范围

- 范围限定在 `experimental/sgl-router` 的 worker 注册核心路径（`manager / registry / introspect / worker` 四文件联动），引擎侧与 Python 侧零改动，线上转发行为零变化。
- 对团队：这是 stacked PR 的第一层，合入后 #39006 自动 retarget 到 main，h2c 转发能力随之进入可消费状态。
- 对运维：`log_protocol_resolution` 的 `warn` 分支会在「URL 不可解析且引擎上报 `enable_http2`」时给出可操作告警，有助于暴露错误配置。

关联脉络

- #39006（后继）：本 PR body 明确关联，启用 request `http2` feature 并消费 `WireProtocol`，恢复 TLS 下的 ALPN 协商，是本 PR 唯一直接下游。
- 同仓库近期 Rust 侧演进（#39329 Rust 前端扩展点、#39677 DSV4.1 Rust 模块）显示 Rust 服务端在持续扩展能力面，但与本 PR 无直接耦合；本 PR 属于 `sgl-router` 协议能力扩展的独立线程，以栈式 PR（base 分支叠加）模式推进，是仓库 Rust 侧常见的协作方式。

- 若后续要修复「introspection 失败永久降级」问题，作者已给出方向：把 kv-events 配置与 PD role 覆盖从「可选增强」变为「硬前置」，但那会改变所有 worker 的准入语义，需要独立 PR 论证。