

PR #39006 完整报告

sgl-project/sglang

[router] Speak cleartext h2c on both edges: serve it inbound, forward it outbound

合并时间: 2026-09-16 13:30

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/39006>

执行摘要

本 PR 把 #39004 中“已解析但没人读”的 per-worker `WireProtocol` 真正接到线路上，并在同一提交里打开入站半场：出站侧 `Proxy` 从单一客户端拆成 `default_client`（协商客户端）与 `h2c_client`（明文 prior knowledge），逐请求按所选 worker 的协议选择；入站侧显式声明 axum 的 `http2` feature，使同一个明文端口既服务 h2c prior-knowledge 也服务 HTTP/1.1。改动不引入任何用户配置项，对纯 HTTP/1.1 部署保持功能等价，但把 TLS worker 的行为改为经 ALPN 协商到 HTTP/2，并把 router crate 内 `forward_*_to` 的签名变成必带协议。三层测试（进程内转发 / 入站监听 / k8s 混合机群）是本次变更最扎实的部分。

功能与动机

动机在 PR 描述中写得很清楚：用 `--enable-http2` 启动的引擎跑 Granian `HTTPModes.auto`，明文端口上同时提供 h2c 与 HTTP/1.1，而 router 此前两个方向都只说 HTTP/1.1，于是

“a mesh sidecar in front of it and an h2c-capable engine behind it both fell back to one-request-per-connection instead of multiplexing.”

关键约束是“无新增配置”：“inbound is negotiated per connection, outbound is selected per worker from a value already on the worker.”

此外，本 PR 是 1.7k 行原始改动 #38743 拆分栈的第三层：第一层 #38989 因 `/server_info` 已返回 `enable_http2` 而关闭，第二层 #39004（解析规则）已合入，本层只含“转发 + 入站监听”。

实现拆解

1. 出站：单一客户端 → 双客户端 + 逐请求选择

文件 `experimental/sgl-router/src/proxy/mod.rs`，关键符号 `Proxy`、`build_client`、`Proxy::new`、`client_for`、`admin_client`、`forward_json_to`、`forward_streaming_to`。

- pub client: Client 被替换为私有的 `default_client` 与 `h2c_client`；新增自由函数 `build_client(protocol)` 统一连接池（`pool_max_idle_per_host(64)`）与 5 s 建连超时，只在 `Http1/H2c` 分支分叉，后者追加 `http2_prior_knowledge()`。
- `forward_json_to` 与 `forward_streaming_to` 各新增 protocol: `WireProtocol` 形参，把 `self.client.post(...)` 改为 `self.client_for(protocol).post(...)`。选择逻辑是纯 match-vs-field，无 IO、无分配。

- `admin_client()` 固定返回 `default_client`，供跨 worker 扇出使用。作者说明这两个客户端必须在构造期建好：“no per-request client construction, and no single shared client whose first writer decides the protocol for the whole fleet.”
- 单测 `client_for_maps_each_protocol_to_its_own_client` 只断言映射关系，并显式说明“断言两个客户端不相等是空洞的”。

2. 调用点：把协议绑到“它被解析出来的那条腿”

- `server/routes/chat.rs` 的 `chat_completions` 共 5 处转发全部补上协议实参。PD 分支最微妙：`worker` 的所有权会被移进 `tokio::spawn`，所以必须在 `spawn` 之前 `let prefill_protocol = worker.protocol();`；`decode` 腿与普通分支则现场读 `decode_worker.protocol() / worker.protocol()`，流式与非流式各一处。
- `server/routes/cache.rs` 的 `flush_cache` 由 `ctx.proxy.client` 改为 `ctx.proxy.admin_client()`，把“扇出流量不属于任何被选中 worker”编码进 API。
- `src/workers/worker.rs` 与 `src/workers/manager.rs` 属于文档 / 日志同步：
`WireProtocol::Http1` 的语义从“明文与 TLS 一律 HTTP/1.1、不参与协商”改为“明文 HTTP/1.1、TLS 下 ALPN 协商”，`log_protocol_resolution` 的 `(Some(true), Some(false))` 与 `(None, _)` 两条文案同步改写，避免运维日志与实现打架。

3. 入站：把已经“白捡”的 h2 能力钉到直接依赖

- `experimental/sgl-router/Cargo.toml`：声明 `axum` 的 `http2` feature，并为 `request` 打开 `http2` (`http2_prior_knowledge()` 本身是 `#[cfg(feature = "http2")]`)。作者解释 `axum` 侧“看似冗余而实则不是”：`axum::serve` 跑在 `hyper-util` 的 `auto builder` 上，HTTP/2 分支由 `hyper-util/http2` 编译，而这条路径此前只通过 `sgl-kv-indexer -> tonic -> hyper-util/server-auto` 传递获得，随时可能因该 gRPC 依赖离开依赖图而退回 HTTP/1.1-only。
- `experimental/sgl-router/src/main.rs` 配合入站服务路径 +4 行（未提供该文件 patch 细节，具体行文不确定）。

4. 测试与 CI 配套

文件	作用
<code>tests/proxy/h2c_forward.rs</code>	用 <code>hyper http2::Builder</code> 起 HTTP/2-only 服务端，验证 h2c 客户端可达、HTTP/1.1 客户端必然失败、 <code>forward_streaming_to</code> 的 multi-frame SSE 不截断且 <code>on_first_byte</code> 触发
<code>tests/proxy/inbound_h2c.rs</code>	走真实 socket + <code>axum::serve</code> （协商在 <code>tower service</code> 之下， <code>oneshot</code> 够不到），断言同端口双协议共存
<code>tests/proxy/pd_protocol_bindings</code>	<code>prefill mock</code> 只讲 HTTP/2、 <code>decode mock</code> 只讲 HTTP/1.1，用线上互斥反证“解析值 = 使用值”

文件	作用
tests/e2e/k8s_integration/test_h2c_forwarding.py	三期 uvicorn + 一台 Granian worker 的混合机群, 用 <code>x_http_version</code> 断言两种协议同时在用
tests/e2e/k8s_integration/fake_worker.py、Dockerfile.fake_worker	新增 <code>FAKE_WORKER_HTTP2</code> 开关 (对 "False" 等写法做归一)、 <code>enable_http2</code> 上报、 <code>x_http_version</code> 回显, 并切换到 Granian <code>HTTPModes.auto</code>
tests/proxy/main.rs、tests/proxy/chat_routing.rs	注册新模块与少量适配
README.md	面向运维补充双端 h2c 语义 (+44/-1)

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

协议真正落到线路上的地方: `chat_completions` 的 5 处转发调用全部补上协议实参, 其中 PD 分支的 `prefill_protocol` 必须在 `tokio::spawn` 之前捕获, 是“解析值 → 使用点”绑定最易出错的接缝。

// experimental/sgl-router/src/server/routes/chat.rs —— 把协议绑到它被解析出来的那条腿

// PD 分支: worker 的所有权会被移进 spawn 出去的任务, 所以协议必须在 spawn

// 之前就取下来 (``prefill_protocol``), spawn 之后就再也拿不到 worker 了。

```
let prefill_url = worker.url.clone();
let prefill_protocol = worker.protocol();
let prefill_breaker = Arc::clone(&worker.breaker);
let prefill_headers = headers.clone();
let prefill_body = outgoing_body.clone();
let prefill_proxy = Arc::clone(&ctx.proxy);
```

```
tokio::spawn(async move {
    match prefill_proxy
        .forward_json_to(
            &prefill_url,
            prefill_protocol, // 绑定 prefill worker 自己解析出的协议
            &prefill_breaker,
            "/v1/chat/completions",
            &prefill_headers,
            prefill_body,
        )
        .await
    {
        // ... prefill 结果处理保持不变 ...
    }
});
```

// decode 腿: worker 仍在作用域内, 直接现取 ``decode_worker.protocol()``;

// 流式与非流式两条路径各一处, 保证与 decode worker 注册时解析的协议一致。

```
let fetch = ctx.proxy.forward_streaming_to(  
  &decode_worker.url,  
  decode_worker.protocol(),  
  &decode_worker.breaker,  
  "/v1/chat/completions",  
  &headers,  
  /* stream_guards, on_first_byte, on_stream_end ... */  
);
```

```
let fetch = ctx.proxy.forward_json_to(  
  &decode_worker.url,  
  decode_worker.protocol(),  
  &decode_worker.breaker,  
  "/v1/chat/completions",  
  &headers,  
  body,  
);
```

// 非 PD 的普通分支同理: `worker.protocol()` 随 `&worker.url` 一起传入,
// 流式与非流式各一处, 避免任何“借用别的 worker 协议”的可能。

评论区精华

本 PR 没有留下任何 review 评论 (`review_comments_count = 0`、`comments_count = 0`) , 只有 sherlockwu 的一条空 body APPROVED。因此真正的“讨论”发生在 PR 描述里, 作者以自问自答的方式预先回应了几处关键质疑:

“The call sites pass `worker.protocol()` alongside `worker.url`, which is what binds the resolved value to the leg it was resolved for.”

“HTTP/1.1 stays the fallback wherever an upgrade is not positively safe, and that fallback is the negotiating client, not a downgrade.”

“Admin fan-out (`/flush_cache`) always uses the default client, since it addresses every worker at once rather than a selected one — hence `Proxy::admin_client` rather than reaching for any one worker's protocol.”

关于 axum 的 `http2` feature“看似冗余”, 作者也主动披露了测试的真实界限:

“these do not fail if axum's `http2` feature is dropped (other dependencies also enable `hyper-util/http2`) — verified by removing it. Their job is the property an operator depends on, whatever the feature graph looks like.”

风险与影响

风险	具体位置	说明
协议错配 即硬失败	<code>proxy::client_for</code> / <code>chat.rs</code> PD 分支	h2c 是 prior knowledge, 不做协商; 对不提供 h2c 的端点会直接失败。缓解依赖 <code>resolve_protocol</code> 只在 <code>enable_http2 == true</code> 且明文时才选 H2c
feature 静默降级	<code>Cargo.toml</code> 的 <code>reqwest</code> / <code>axum http2</code>	<code>worker.rs</code> 文档明言“Dropping that feature silently reduces this variant to HTTP/1.1 everywhere”, 且入站测试在 feature 图被其他依赖补上时不会失败
TLS 行为 变更	<code>manager.rs</code> / <code>worker.rs</code> 注释 与日志	<code>http2</code> feature 打开后, <code>https://</code> worker 会经 ALPN 协商到 HTTP/2, 是线路级行为变化, 需真实环境验证
PD 绑定 覆盖偏薄	<code>chat_completions</code> 与 <code>pd_protocol_binding.rs</code>	三条表达式传错任一都能编译通过, 却只有一处测试兜底; <code>prefill</code> 腿依赖 3 s 轮询, 属弱同步
已知未解 决取舍	<code>reconcile_unresolved_workers</code> (#39004 引入)	<code>/server_info</code> 不可达的 worker 终生走 HTTP/1.1, 本 PR 未改变该模式, 仅是吞吐代价
e2e 环境 耦合	<code>test_h2c_forwarding.py</code>	依赖 kind 集群、 <code>--cb-threshold 1</code> 的 router manifest、fake-worker 镜像重建与 90 s 收敛窗口

影响面：对已启用 `--enable-http2` 的部署，router→引擎、mesh→router 两段都获得 HTTP/2 多路复用；对纯 HTTP/1.1 部署功能等价（`inbound_still_accepts_http1` 与 `http1_client_cannot_reach_http2_only_worker` 双向兜底）。crate 内部 API 有 breaking: `Proxy::client` 变私有、两个 `forward_*_to` 新增必填参数，仓库内调用点已在同一 PR 内全部更新。

关联脉络

- #39004（已合入）：WireProtocol 枚举、`manager::resolve_protocol`、`registry::add_with_cb` 的协议盖章，当时“changes no bytes on the wire”；本 PR 是它的执行层，并回头修正了它对 ALPN 的错误描述。
- #38743（被取代）：1.7k 行单 commit 的原始 h2c 改动，本 PR 是拆分栈的第三层。
- #38989（已关闭）：原计划让引擎在 `/model_info` 上报 `enable_http2`，后确认 `/server_info` 的 `resolved_dict()` 早已包含该字段，故本 PR 直接从 `/server_info` 读取。
- #39000 / #39014（同期 router PR）：同属 router 的配置面与连接生命周期系列，采用同样的拆栈推进方式；#39014 的“在途 HTTP 计数”与本 PR 的多路复用目标都指向连接效率这条主线。

总体判断：这是一个动机清楚、边界自律（落单层、改文档、不改配置面）、测试设计讲究的 PR；主要遗留是 PD 协议绑定的测试覆盖偏薄，以及 `http2` feature 缺失时的静默降级只靠文档和断言性质型测试兜底。