PR #25293 完整报告

sgl-project/sglang

[SMG] Add /v1/models fallback for model name discovery

合并时间：2026-05-18 22:02

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25293

PR #25293 分析报告

执行摘要

此PR为SMG（SGLang Model Gateway）的模型名称发现机制添加了/v1/models回退支持。当后端推理引擎未实现 /server_info 端点时，路由器可自动从 /v1/models 获取模型名称，解决了多模型路由因无法发现模型名而中断的问题。新增的集成测试覆盖了正常与回退两种场景，确保功能正确且不破坏现有行为。

功能与动机

SMG 路由器在初始化 worker 时需要发现其服务的模型名称，通常通过 /server_info 或 /model_info 端点。但许多兼容 OpenAI-API 的推理引擎（如 vLLM、TGI）并未实现这些专有端点，只实现了标准的 /v1/models 端点。当发现失败时，模型名称会回退为 UNKNOWN_MODEL_ID，导致多模型路由无法根据模型名称匹配请求。此 PR 旨在大幅降低 SMG 对专有端点依赖性，提升与更多后端的即插即用兼容性。

实现拆解

1. 新增 get_model_name_from_v1_models 函数(discover_metadata.rs +48 行）：该异步函数向 worker 的 /v1/models 发送 GET 请求，从返回 JSON 的 data 数组中查找第一个 object 字段值为 model 的条目，提取其 id 作为模型名称。采用 .and_then() 链式处理，解析失败时返回 Err。
2. 集成到 DiscoverMetadataStep(same file，在 HTTP 分支结尾）：在收集完 /server_info 和 /model_info 的 labels 后，检查 labels 中是否已包含 model_path 或 served_model_name。若两者均不存在，则调用 get_model_name_from_v1_models，将结果插入 labels 作为 served_model_name。该逻辑仅影响 HTTP 连接模式，gRPC 模式不受影响。
3. 构建测试模拟 worker(mock_worker.rs +90 行）：新增 OpenAiOnlyMockWorker 结构体，使用 axum 搭建一个仅暴露 /health、/health_generate 和 /v1/models 端点的最小 HTTP 服务器，模拟那些未实现 /server_info 的后端。支持动态端口绑定和优雅关闭。
4. 编写集成测试(worker_discovery_test.rs +94 行）：
   - test_model_name_discovered_via_server_info：使用完整 MockWorker（包含 /server_info），断言模型名称从标准端点成功获取（预期 mock-model-path）。

- test_model_name_discovered_via_v1_models_fallback：使用 OpenAiOnlyMockWorker，提交 worker 初始化任务，等待 worker 健康后检查注册表中的模型名称是否与预设值（my-model）一致。

5. 注册测试模块(mod.rs +1 行) ：在路由测试模块文件中添加 pub mod worker_discovery_test;。

sgl-model-gateway/src/core/steps/worker/local/discover_metadata.rs

核心变更文件，新增 fallback 函数并集成到步骤中。

```
/// Get model name from /v1/models endpoint (OpenAI-compatible fallback).
/// 当 `/server_info` 和 `/model_info` 均失败时调用此函数。
async fn get_model_name_from_v1_models(url: &str, api_key: Option<&str>) -> Result<String,
String> {
    let base_url = url.trim_end_matches('/');
    let models_url = format!("{}/v1/models", base_url);

    let mut req = HTTP_CLIENT.get(&models_url);
    if let Some(key) = api_key {
        req = req.bearer_auth(key);
    }

    let response = req
        .send()
        .await
        .map_err(|e| format!("Failed to connect to {}: {}", models_url, e))?;

    if !response.status().is_success() {
        return Err(format!(
            "Server returned status {} from {}",
            response.status(),
            models_url
        ));
    }

    let json: Value = response
        .json()
        .await
        .map_err(|e| format!("Failed to parse response from {}: {}", models_url, e))?;

    // 根据 OpenAI API 规范，只取 object 为 "model" 的第一个条目
    json["data"]
        .as_array()
        .and_then(|arr| {
            arr.iter()
                .find(|entry| entry["object"].as_str() == Some("model"))
        })
        .and_then(|entry| entry["id"].as_str())
        .map(|s| s.to_string())
```

```
        .ok_or_else(|| format!("No model found in response from {}", models_url))
}

// 在 DiscoverMetadataStep 的 HTTP 分支中调用：
// 如果已有 labels 不含 model_path 或 served_model_name，则尝试 fallback
if !labels.contains_key("model_path") && !labels.contains_key("served_model_name") {
    if let Ok(model_name) =
        get_model_name_from_v1_models(&config.url, config.api_key.as_deref()).await
    {
        labels.insert("served_model_name".to_string(), model_name);
    }
}
```

sgl-model-gateway/tests/common/mock_worker.rs

新增 OpenAiOnlyMockWorker 用于测试回退路径。

```
/// A minimal OpenAI-compatible mock worker that does not implement /server_info or /model_
info.
pub struct OpenAiOnlyMockWorker {
    port: u16,
    model_name: String,
    shutdown_handle: Option<tokio::task::JoinHandle<()>>,
    shutdown_tx: Option<tokio::sync::oneshot::Sender<()>>,
}

impl OpenAiOnlyMockWorker {
    pub fn new(model_name: impl Into<String>) -> Self {
        Self { port: 0, model_name: model_name.into(), shutdown_handle: None, shutdown_tx:
        None }
    }

    pub async fn start(&mut self) -> Result<String, Box<dyn std::error::Error>> {
        // 绑定随机端口，仅注册 /health, /health_generate, /v1/models
        // /v1/models 返回包含 model_name 的简化 OpenAI 响应
        // 详细实现见完整文件
        Ok(format!("http://127.0.0.1:{}", self.port))
    }

    pub async fn stop(&mut self) {
        if let Some(tx) = self.shutdown_tx.take() { let _ = tx.send(()); }
        if let Some(h) = self.shutdown_handle.take() {
            let _ = tokio::time::timeout(tokio::time::Duration::from_secs(5), h).await;
        }
    }
}

impl Drop for OpenAiOnlyMockWorker {
    fn drop(&mut self) {
        if let Some(tx) = self.shutdown_tx.take() { let _ = tx.send(()); }
```

```
    }
}
```

sgl-model-gateway/tests/routing/worker_discovery_test.rs

新增集成测试文件，覆盖正常和回退两种场景。

```
use smg::{config::RouterConfig, core::Job};
use crate::common::{create_test_context, mock_worker::{HealthStatus, MockWorkerConfig,
OpenAiOnlyMockWorker, WorkerType}, AppTestContext};

#[cfg(test)]
mod worker_discovery_tests {
    use super::*;

    #[tokio::test]
    async fn test_model_name_discovered_via_server_info() {
        let ctx = AppTestContext::new(vec![MockWorkerConfig {
            port: 0, worker_type: WorkerType::Regular,
            health_status: HealthStatus::Healthy, response_delay_ms: 0, fail_rate: 0.0,
        }]).await;
        let discovered = ctx.app_context.worker_registry.get_models();
        assert!(discovered.contains(&"mock-model-path".to_string()), "Expected mock-model-path");
        ctx.shutdown().await;
    }

    #[tokio::test]
    async fn test_model_name_discovered_via_v1_models_fallback() {
        let mut worker = OpenAiOnlyMockWorker::new("my-model");
        let url = worker.start().await.unwrap();
        let config = RouterConfig::builder().regular_mode(vec![url.clone()]).random_policy()
            .host("127.0.0.1").port(0).max_payload_size(256 * 1024 * 1024)
            .request_timeout_secs(600).worker_startup_timeout_secs(5)
            .worker_startup_check_interval_secs(1).max_concurrent_requests(64)
            .queue_timeout_secs(60).build_unchecked();
        let app_context = create_test_context(config.clone()).await;
        app_context.worker_job_queue.get().unwrap()
            .submit(Job::InitializeWorkersFromConfig { router_config: Box::new(config) }).await.
            unwrap();
        // 等待 worker 健康
        let start = tokio::time::Instant::now();
        loop {
            if app_context.worker_registry.get_all().iter().any(|w| w.is_healthy()) { break; }
            tokio::time::sleep(tokio::time::Duration::from_millis(100)).await;
        }
        let discovered = app_context.worker_registry.get_models();
        assert!(discovered.contains(&"my-model".to_string()), "Fallback failed");
        worker.stop().await;
    }
}
```

评论区精华

- @alexnails 审查：要求从 /v1/models 提取模型名称时验证 object 字段为 model，避免从其他资源类型（如 engine）中误取 ID。该请求在代码中得到实现：arr.iter().find(|entry| entry["object"].as_str() == Some("model"))。
- @Gruner-atero 回复：标记为 done，并在后续 commit 中添加了该验证逻辑。
- 最终审批：@alexnails 批准该 PR。

风险与影响

- 风险：极低。回退逻辑仅在两个主力端点均失败时执行，不影响正常路径。潜在风险是目标后端的 /v1/models 响应格式与 OpenAI 规范略有偏差，但函数已内置友好错误处理（返回 Err），不会传播异常或导致 panic。
- 影响：正面。提升 SMG 对标准 OpenAI-API 后端的兼容性，用户无需为每个后端专门配置模型名称。测试覆盖完善，回归风险低。

关联脉络

此 PR 是 SMG 组件独立演进的一部分，不直接依赖于本次提供的近期历史 PR（主要涉及 scheduler 重构）。它增强了模型发现模块的鲁棒性，未来可能与其他 SMG 功能（如多模型自动路由）形成组合优势。