PR #22500 完整报告

sgl-project/sglang

[Observability] Add HTTP sidecar endpoints and FlushCache gRPC RPC for gRPC mode

合并时间：2026-04-23 14:06

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/22500

执行摘要

- 一句话：为 gRPC 模式添加 HTTP sidecar 端点和 FlushCache RPC
- 推荐动作：建议关注以下设计决策：
  - profiling 端点放在 sidecar 而非 gRPC：PD 模式需要 per-worker 定向，不应通过 router fan-out。
  - FlushCache 作为 gRPC RPC：需要 fan-out 到所有 worker，因此通过 router 调度是合理的选择。
  - 移除重复端点：review 中及时删除了 sidecar 中多余的 /server_info，保持了单一实现。这些决策体现了对异构传输模式下功能划分的清晰思考，值得在类似场景复用。

功能与动机

gRPC 模式之前缺乏 profiling、缓存管理和服务器内省端点，阻碍了 torch profiler 工作流与 bench_serving.py 对 gRPC 部署的使用。PR body 明确提到："gRPC mode previously lacked profiling, cache management, and server introspection endpoints. This blocked: Torch profiler workflows… bench_serving.py and bench_one_batch_server against gRPC deployments"。

实现拆解

1. 重构现有 metrics 服务为通用 sidecar：将原来独立的 _start_metrics_server 函数拆分为 _start_sidecar_server（通用启动逻辑）、_add_metrics_routes（添加 /metrics 路由）和 _check_communicator_results（结果校验辅助函数），使 sidecar 可承载多个端点。
2. 新增 profiling 端点：在 _add_admin_routes 中注册 /start_profile 和 /stop_profile 两个 POST 端点。start_profile_handler 解析 JSON 请求体（含环境变量覆盖逻辑，如 SGLANG_PROFILE_WITH_STACK），构造 ProfileReq 对象，通过 request_manager.send_communicator_req 发送给 scheduler，并处理响应错误。超时设置为 600 秒。
3. gRPC FlushCache 集成：通过依赖的 smg-grpc-servicer 包新增 FlushCache RPC，router 在收到 HTTP /flush_cache 请求时对 gRPC worker 调用该 RPC，实现所有 worker 的缓存刷新。
4. 启动流程调整：serve_grpc 函数现在无条件启动 HTTP sidecar（不再仅当 --enable-metrics 时才启动），_on_request_manager_ready 回调用于在 gRPC request

manager 就绪后将 sidecar 的 app 传给 router。

5. 服务器参数变更：将 metrics_http_port 字段重命名为 grpc_http_sidecar_port，CLI 参数同步更改，并更新帮助文本以体现新用途。

关键文件：

- python/sglang/srt/entrypoints/grpc_server.py（模块 gRPC 服务；类别 source；类型 dependency-wiring；符号 _start_sidecar_server, _add_metrics_routes, _check_communicator_results, _add_admin_routes）：核心变更文件，完成 HTTP sidecar 框架搭建、profiling 端点实现、启动流程调整，是本次 PR 的主要逻辑所在。
- python/sglang/srt/server_args.py（模块 配置参数；类别 source；类型 configuration）：将 --metrics-http-port 重命名为 --grpc-http-sidecar-port，反映 sidecar 的新用途（不仅提供 metrics，还提供 profiling 端点）。

关键符号：_start_sidecar_server, _add_metrics_routes, _check_communicator_results, _add_admin_routes, start_profile_handler, stop_profile_handler, _on_request_manager_ready

关键源码片段

python/sglang/srt/entrypoints/grpc_server.py

核心变更文件，完成 HTTP sidecar 框架搭建、profiling 端点实现、启动流程调整，是本次 PR 的主要逻辑所在。

```
# python/sglang/srt/entrypoints/grpc_server.py
# HTTP sidecar 核心框架

def _add_admin_routes(app, request_manager):
    """添加 /start_profile、/stop_profile 等 admin 端点。
    Business logic（请求构造、环境变量处理、响应解释）在此处；
    request_manager 仅提供与 scheduler 的通信传输。
    """

    async def start_profile_handler(request):
        try:
            # 解析 JSON body，允许空 body
            if request.content_length and request.content_length > 0:
                try:
                    body = await request.json()
                except json.JSONDecodeError as e:
                    return web.Response(status=400, text=f"Invalid JSON: {e}")
            else:
                body = {}

            # 环境变量覆盖逻辑（与 tokenizer_communicator_mixin 一致）
            with_stack = body.get("with_stack")
            env_with_stack = get_bool_env_var("SGLANG_PROFILE_WITH_STACK", "true")
            with_stack = (with_stack is not False) and env_with_stack
```

```
        record_shapes = body.get("record_shapes")
        env_record_shapes = get_bool_env_var("SGLANG_PROFILE_RECORD_SHAPES", "true")
        record_shapes = (record_shapes is not False) and env_record_shapes

        # 构造 ProfileReq，通过 communicator 发送给 scheduler
        req = ProfileReq(
            type=ProfileReqType.START_PROFILE,
            output_dir=body.get("output_dir"),
            start_step=body.get("start_step"),
            num_steps=body.get("num_steps"),
            activities=body.get("activities"),
            with_stack=with_stack,
            record_shapes=record_shapes,
            profile_by_stage=body.get("profile_by_stage", False),
            profile_id=str(time.time()),
            merge_profiles=body.get("merge_profiles", False),
            profile_prefix=body.get("profile_prefix"),
            profile_stages=body.get("profile_stages"),
        )
        # 超时 600 秒以支持长时间 profiling
        results = await request_manager.send_communicator_req(
            req, "profile_communicator", timeout=600.0
        )
        err = _check_communicator_results(results, "Start Profile")
        if err:
            return err
        return web.Response(text="Start profiling.\n")
    except Exception as e:
        logger.exception("Failed to start profile")
        return web.Response(status=500, text="Internal error")

# 类似实现 stop_profile_handler，此处省略

app.router.add_post("/start_profile", start_profile_handler)
app.router.add_post("/stop_profile", stop_profile_handler)
```

评论区精华

Review 中主要讨论点：

- 移除 /server_info 端点：slin1237 指出 server_info 已原生支持于 smg-grpc-servicer 的 gRPC RPC 中，建议移除重复的 HTTP 端点。作者采纳并在第二个 commit 中移除，并在 commit message 中致谢。
- 向后兼容保留 GET /metrics：尽管 /metrics 本应是 GET，但作者指出“一些测试工具使用 GET，为向后兼容保留”，故未改动。
  - HTTP sidecar 中的 /server_info 端点是否多余 (design): 作者同意并在 commit f2a4198 中删除了该端点，commit message 说明移除原因并致谢 reviewer。
  - /metrics 端点保持 GET 方法以向后兼容 (design): 接受，未修改。

风险与影响

- 风险：
  1. 依赖外部服务：本 PR 的功能依赖 smg-grpc-servicer 包的更新，若对等 PR (lightseekorg/smg#1088) 未及时合并或存在兼容性问题，FlushCache RPC 和 request_manager 回调可能不可用。
  2. 配置向后兼容：--metrics-http-port 被移除，使用该参数的启动脚本会报错。需确保所有部署脚本已更新为 --grpc-http-sidecar-port。
  3. 无新增测试覆盖：PR body 提到"CI tests"未完成（勾选框未选），且没有添加对应的单元测试或集成测试文件，存在回归风险。
  4. sidecar 端口冲突：sidecar 默认使用 port+1，如果该端口已被占用，启动会失败并回退到无 metrics 状态（原有逻辑），但 profiling 端点也会同时不可用。 - 影响：影响范围：针对使用 gRPC 模式的用户，特别是需要 profiling 和缓存管理场景。用户影响：gRPC 模式用户现在可以通过熟悉的 HTTP 端点启动 / 停止 torch profiler，并可通过 router 的 /flush_cache 刷新所有 worker 缓存。对等请求的超时设置为 600 秒，适合长时间 profiling。系统影响：sidecar 会增加一个额外的 HTTP 监听端口，但仅当 gRPC 模式启用时才会启动。团队影响：维护了 HTTP 和 gRPC 两种模式的可观察性一致性，减少了社区因缺少 gRPC profiling 而提出的支持请求。
- 风险标记：配置向后兼容，缺少测试覆盖，依赖外部包 smg-grpc-servicer

关联脉络

- PR #1088 feat(grpc): add FlushCache RPC and profiling support for gRPC mode: 对等 PR，需要在 smg-grpc-servicer 中添加 FlushCache RPC 和 request_manager 回调支持，本 PR 才能完整工作。