PR #22253 完整报告

sgl-project/sglang

[EPD] Support dynamic encoder register

合并时间：2026-06-08 16:05

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/22253

执行摘要

- 一句话：为 EPD 分离添加动态编码器注册引导服务器
- 推荐动作：值得精读。PR 展示了如何设计一个带健康检查的引导服务器，以及与既有静态配置共存的策略。讨论中的 nEmP 支持路径对后续架构有指导意义。

功能与动机

EPD 语言服务器依赖 --encoder-urls 在启动时静态指定编码器地址，无法动态添加或替换编码器。需要一个专用引导服务器使编码器自行注册，实现灵活发现和容错。

实现拆解

1. 配置层：在 server_args.py 添加 encoder_bootstrap_port 和 encoder_register_urls，并将 encoder_urls 从必选改为可选。
2. 引导服务器：在 encode_receiver.py 实现 EncoderBootstrapServer 类（FastAPI + uvicorn），提供 /register_encoder_url、/unregister_encoder_url、/list_encoder_urls、/health 端点，并带健康检查循环（连续 3 次失败自动移除）。
3. 编码器注册：在 encode_server.py 实现 _register_encoder_url_with_bootstrap 函数，启动守护线程以退避重试方式向每个引导服务器注册 URL（最多 30 次、间隔 5 秒）。
4. 管理器集成：在 tokenizer_manager.py 的 init_disaggregation 中实例化 EncoderBootstrapServer，并将共享 URL 列表传给 MMReceiver，使接收器可直接读取同一列表。
5. 动态刷新：在 encode_receiver.py 添加 _refresh_encoder_urls_from_bootstrap，每次请求时（速率限制 5 秒）从引导服务器获取最新 URL 列表覆盖 self.encode_urls。
6. 修复：在 check_server_args 中提前计算 encoder_bootstrap_url 使其被子进程继承；在 _send_encode_request 中处理 encode_urls 为空的情况，避免 AttributeError。同时将 encoder_urls 字段加入 io_struct.py 的请求对象，确保调度器子进程中编码器分配一致性。

关键文件：

- python/sglang/srt/disaggregation/encode_receiver.py（模块 引导服务器；类别 source；类型 dependency-wiring；符号 EncoderBootstrapServer, init, lifespan, _health）：新增 EncoderBootstrapServer 类（核心组件），以及动态刷新逻辑。

- python/sglang/srt/disaggregation/encode_server.py（模块 编码器启动；类别 source；类型 dependency-wiring；符号 _register_encoder_url_with_bootstrap, _try_register_once, _worker）：实现异步编码器注册逻辑，后台重试注册到引导服务器。
- python/sglang/srt/server_args.py（模块 配置层；类别 source；类型 core-logic；符号 encoder_bootstrap_port, encoder_register_urls）：添加编码器引导服务器端口和注册 URL 列表 CLI 参数。
- python/sglang/srt/managers/tokenizer_manager.py（模块 管理器；类别 source；类型 dependency-wiring；符号 init_disaggregation）：在 init_disaggregation 中实例化引导服务器并传递共享 URL 列表。
- python/sglang/srt/managers/io_struct.py（模块 数据结构；类别 source；类型 core-logic；符号 encoder_urls）：在请求结构中添加 encoder_urls 字段，以在子进程间传播编码器列表。
- python/sglang/srt/environ.py（模块 环境配置；类别 source；类型 core-logic；符号 SGLANG_ENCODER_BOOTSTRAP_HEALTH_CHECK_INTERVAL, SGLANG_ENCODER_BOOTSTRAP_HEALTH_CHECK_TIMEOUT）：添加健康检查间隔和超时环境变量。

关键符号：EncoderBootstrapServer.register, EncoderBootstrapServer._health_check_loop, _register_encoder_url_with_bootstrap, _refresh_encoder_urls_from_bootstrap

关键源码片段

python/sglang/srt/disaggregation/encode_receiver.py

新增 EncoderBootstrapServer 类（核心组件），以及动态刷新逻辑。

```
# python/sglang/srt/disaggregation/encode_receiver.py
# EncoderBootstrapServer 类的核心注册与健康检查逻辑
class EncoderBootstrapServer:
    """Lightweight bootstrap server for dynamic encoder discovery.

    Built on FastAPI + uvicorn, runs in a daemon thread.
    The registered URLs are stored in a list shared with MMReceiver.
    """
    def __init__(
        self,
        host: str,
        port: int,
        urls: Optional[List[str]] = None,
        health_check_interval: Optional[float] = None,
        health_check_timeout: Optional[float] = None,
    ):
        self.host = host
        self.port = port
        # _urls 与外部共享列表引用，register/unregister 直接修改此列表
        self._urls: List[str] = urls if urls is not None else []
        self._lock = threading.Lock()
```

```
        self._health_check_interval = (
            health_check_interval
            if health_check_interval is not None
            else envs.SGLANG_ENCODER_BOOTSTRAP_HEALTH_CHECK_INTERVAL.get()
        )
        self._health_check_timeout = (
            health_check_timeout
            if health_check_timeout is not None
            else envs.SGLANG_ENCODER_BOOTSTRAP_HEALTH_CHECK_TIMEOUT.get()
        )
        self._consecutive_failures: Dict[str, int] = {}
        self._max_consecutive_failures = 3

        # FastAPI app and routes (omitted for brevity)
        # ...

    def register(self, url: str) -> None:
        """线程安全添加编码器 URL，避免重复。"""
        with self._lock:
            if url not in self._urls:
                self._urls.append(url)

    async def _health_check_loop(self) -> None:
        """周期性健康检查，连续失败 3 次则移除 URL。"""
        while True:
            await asyncio.sleep(self._health_check_interval)
            async with ClientSession() as session:
                for url in self.list_urls():
                    try:
                        async with session.get(
                            f"{url}/health",
                            timeout=ClientTimeout(self._health_check_timeout),
                        ) as resp:
                            if resp.status == 200:
                                self._consecutive_failures[url] = 0
                            else:
                                self._consecutive_failures[url] = (
                                    self._consecutive_failures.get(url, 0) + 1
                                )
                    except Exception:
                        self._consecutive_failures[url] = (
                            self._consecutive_failures.get(url, 0) + 1
                        )
                    if self._consecutive_failures[url] >= self._max_consecutive_failures:
                        logger.warning(f"Removing unhealthy encoder: {url}")
                        self.unregister(url)
                        self._consecutive_failures.pop(url, None)
```

python/sglang/srt/disaggregation/encode_server.py

实现异步编码器注册逻辑，后台重试注册到引导服务器。

```
# python/sglang/srt/disaggregation/encode_server.py
# 编码器启动时向引导服务器注册自身的函数

def _register_encoder_url_with_bootstrap(server_args: ServerArgs) -> None:
    """在后台线程中向所有配置的引导服务器注册本编码器 URL。

    每个引导服务器独立重试，最多 30 次，间隔 5 秒，
    避免序列化等待导致阻塞。
    """
    encoder_url = server_args.url()
    payload = {"url": encoder_url}
    bootstrap_urls = list(server_args.encoder_register_urls)
    if not bootstrap_urls:
        return

    def _try_register_once(bootstrap_url: str) -> bool:
        try:
            resp = http_requests.post(
                f"{bootstrap_url}/register_encoder_url",
                json=payload,
                timeout=5.0,
            )
            if resp.status_code == 200:
                logger.info(
                    "Registered encoder URL '%s' with bootstrap at %s",
                    encoder_url,
                    bootstrap_url,
                )
                return True
            logger.warning(
                "Bootstrap %s returned %d: %s",
                bootstrap_url,
                resp.status_code,
                resp.text,
            )
        except Exception as e:
            logger.debug("Register attempt to %s failed: %s", bootstrap_url, e)
        return False

    def _worker():
        pending = list(bootstrap_urls)
        retry_count = {url: 0 for url in pending}
        while pending:
            still_pending = []
            for bootstrap_url in pending:
                if _try_register_once(bootstrap_url):
                    continue
```

```
                retry_count[bootstrap_url] += 1
                if retry_count[bootstrap_url] >= 30:
                    logger.error(
                        "Giving up on bootstrap %s after 30 attempts."
                        " Encoder discovery via this bootstrap will be incomplete.",
                        bootstrap_url,
                    )
                    continue
                still_pending.append(bootstrap_url)
            pending = still_pending
            if pending:
                time.sleep(5.0)

    threading.Thread(
        target=_worker, daemon=True, name="encoder-bootstrap-register"
    ).start()
```

python/sglang/srt/server_args.py

添加编码器引导服务器端口和注册 URL 列表 CLI 参数。

```
# python/sglang/srt/server_args.py 新增参数定义
# (partial)
class ServerArgs:
    # ...
    encoder_bootstrap_port: int = 8997 # 语言服务器端引导服务器端口
    encoder_register_urls: List[str] = dataclasses.field(default_factory=list)
    # ...
```

评论区精华

- nEmP 支持：ShangmingCai 询问是否支持多编码器多预填充，作者表示逐步支持，当前仅 nE1P。
- 引导服务器位置：ShangmingCai 建议将启动从 http_server.py 移至 tokenizer_manager，作者采纳。
- 参数冗余：ShangmingCai 认为 encoder_bootstrap_url 可由端口派生，作者最终移除该参数。
- 健康检查配置：ZhengWG 建议通过环境变量配置，作者添加了 SGLANG_ENCODER_BOOTSTRAP_HEALTH_CHECK_INTERVAL 和 TIMEOUT。
- 编码器 DP 兼容性：ShangmingCai 指出需在 DP 模式启动中也调用注册函数，作者已添加。
- 自动注销：ZhengWG 提出心跳自动注销，健康检查循环已实现但无主动心跳。
- nEmP 支持范围 (design): 作者表示逐步支持，当前不实现 nEmP。
- 引导服务器启动位置 (design): 作者采纳并迁移。
- encoder_bootstrap_url 参数冗余 (design): 作者移除了 encoder_bootstrap_url，仅保留端口。

- 健康检查可配置性 (design): 作者添加了 SGLANG_ENCODER_BOOTSTRAP_HEALTH_CHECK_INTERVAL 和 TIMEOUT。
- 编码器 DP 模式兼容性 (correctness): 作者在 DP 模式路径中增加了注册调用。
- 自动注销机制 (design): 健康检查循环已实现自动移除，但未实现基于主动心跳的注销。

风险与影响

- 风险:
  - 单点故障：引导服务器嵌在语言管理器进程中，若该进程异常，新编码器无法注册，但已注册编码器仍可继续处理。
  - 最终一致性：编码器注册不保证立即到达，健康检查间隔（默认 10s）内可能发出请求到不健康编码器。
  - 静态与动态混合：同时使用 --encoder-urls 和动态注册时，列表可能包含重复，代码已去重。
  - DP 模式覆盖：虽已添加注册调用，但未在测试中验证 DP 模式下的注册路径。
  - 安全暴露：引导服务器端点无认证，在生产环境需配置网络隔离。
  - 影响：用户可灵活部署编码器，无需在启动前枚举所有 URL；编码器可以动态扩容和缩容。需要理解新的 CLI 参数。此 PR 向后兼容（原 --encoder-urls 仍可用）。对现有 EPD 用户推荐升级以获得动态能力。
  - 风险标记：引导服务器进程耦合语言管理器，单点故障，静态 encoder_urls 与动态注册混合可能产生重复，健康检查间隙可能使用不健康编码器，编码器 DP 模式注册路径未充分测试，缺少主动心跳注销机制

关联脉络

- 暂无明显关联 PR