PR #34796 完整报告

sgl-project/sglang

Add --http2-max-concurrent-streams server arg

合并时间：2026-08-16 01:34

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34796

执行摘要

- 一句话：为 HTTP/2 服务新增可配置的并发流上限参数
- 推荐动作：值得快速浏览：适合关注 HTTP/2 部署、ServerArgs 校验模式和 Granian 配置接线的开发者。可以学习用 FakeEmbeddedServer 拦截 Server 构造来做配置传递断言的测试手法，以及 " 服务器显式拥有连接级容量并广播给客户端 " 的设计思路。该 PR 不涉及核心推理路径，无模型性能影响，不需要精读。

功能与动机

PR body 明确指出：--enable-http2 下 Granian 的 SETTINGS_MAX_CONCURRENT_STREAMS 值 is not configurable today, and it is not reported anywhere。对池化 HTTP/2 连接的客户端来说，需要知道服务器广告的每连接流上限来 size its own admission control；否则超发请求会在 HTTP/2 传输层内部静默排队，无法在连接池边界快速失败；准入过少则连接容量闲置。因此 The server should own this value and make it explicit，把并发流上限变成服务器可配置、可报告的显式属性。

实现拆解

1. 参数定义与校验（python/sglang/srt/server_args.py）：在 ServerArgs 中紧随 enable_http2 之后新增 http2_max_concurrent_streams 字段（int，默认 200，归属 NS("serving")）；在 _handle_ssl_validation 的 enable_http2 分支内新增范围校验 0 < value < 2**32，非法值抛 ValueError。校验放在模型加载前的参数处理阶段，保证快速失败。
2. 参数接线（python/sglang/srt/entrypoints/http_server.py）：_run_granian_server 新增必填位置参数 http2_max_concurrent_streams（位于 log_level 之后），从 granian.http 导入 HTTP2Settings，在构造 granian_kwargs 时注入 http2_settings=HTTP2Settings(max_concurrent_streams=...)；_setup_and_run_http_server 在单 tokenizer worker 与多 tokenizer worker 两条 Granian 分支都显式传入 server_args.http2_max_concurrent_streams。
3. 测试配套：新增 test/registered/unit/entrypoints/test_http2_server_config.py（注册 CPU CI，est_time=2），用 unittest.mock.patch 将 granian.server.embed.Server 替换为 FakeEmbeddedServer，捕获构造参数并断言 http2_settings.max_concurrent_streams == 37；test/registered/openai_server/basic/test_http2_server.py 在单 / 多 tokenizer worker 两种启动配置中显式加入 --http2-max-concurrent-streams 64，确保真

实服务器路径消费该标志。

4. 文档配套：docs/docs/advanced_features/server_arguments.mdx 新增一行参数说明，标注类型 int、默认值 200 与生效条件（仅 --enable-http2 时）。

关键文件：

- python/sglang/srt/server_args.py（模块 参数配置；类别 source；类型 core-logic；符号 ServerArgs, http2_max_concurrent_streams, _handle_ssl_validation）：新增 http2_max_concurrent_streams 字段（默认 200）并在 _handle_ssl_validation 中做 HTTP/2 合法范围校验，是本次变更的配置入口。
- python/sglang/srt/entrypoints/http_server.py（模块 入口服务；类别 source；类型 dependency-wiring；符号 _run_granian_server, _setup_and_run_http_server）：_run_granian_server 新增必填参数并构造 HTTP2Settings 注入 Granian，单 / 多 tokenizer worker 两条分支完成接线，是功能落地的关键路径。
- test/registered/unit/entrypoints/test_http2_server_config.py（模块 配置测试；类别 test；类型 test-coverage；符号 TestGranianHTTP2Config, test_passes_explicit_max_concurrent_streams, FakeEmbeddedServer）：新增 CPU 单元测试，用 FakeEmbeddedServer 拦截 Granian Server 构造，断言显式值进入 HTTP2Settings，是该参数接线正确性的核心验证。
- test/registered/openai_server/basic/test_http2_server.py（模块 端到端测试；类别 test；类型 test-coverage；符号 TestHTTP2Server, TestHTTP2ServerMultiTokenizer）：在单 / 多 tokenizer worker 端到端启动参数中显式加入 --http2-max-concurrent-streams 64，确保真实服务器路径消费该标志。
- docs/docs/advanced_features/server_arguments.mdx（模块 文档；类别 docs；类型 documentation）：补充参数使用文档，标注类型、默认值与生效条件，方便用户发现该配置。

关键符号：_run_granian_server, _setup_and_run_http_server, _handle_ssl_validation

关键源码片段

python/sglang/srt/server_args.py

新增 http2_max_concurrent_streams 字段（默认 200）并在 _handle_ssl_validation 中做 HTTP/2 合法范围校验，是本次变更的配置入口。

```
# python/sglang/srt/server_args.py 中的关键片段（合并展示）
class ServerArgs:
    # ... 其他字段 ...

    enable_http2: A[
        bool,
        "Use Granian instead of Uvicorn as the ASGI server, enabling HTTP/1.1 "
        "and HTTP/2 auto-negotiation. Clients may use h2c (cleartext HTTP/2) "
        "or plain HTTP/1.1. Requires 'pip install sglang[http2]'.",
        NS("serving"),
    ] = False
```

```
    # 新增：HTTP/2 每连接最大并发流数，默认 200
    # 该值会通过 Granian 的 HTTP2Settings 在连接建立时广播给客户端，
    # 客户端据此决定连接池的准入上限，避免超发请求在传输层内排队。
    http2_max_concurrent_streams: A[
        int,
        "Maximum number of concurrent streams advertised on each HTTP/2 "
        "connection (1 to 2^32 - 1). Only applies with --enable-http2.",
        NS("serving"),
    ] = 200

    def _handle_ssl_validation(self) -> None:
        # ... 其他 SSL 校验 ...

        if self.enable_http2:
            # HTTP/2 协议规定 SETTINGS_MAX_CONCURRENT_STREAMS 合法范围是
            # 1 ~ 2^32 - 1，在模型加载前做前置校验以便快速失败
            if not 0 < self.http2_max_concurrent_streams < 2**32:
                raise ValueError(
                    "--http2-max-concurrent-streams must be between 1 and "
                    "4294967295."
                )

            try:
                import granian # noqa: F401
            except ImportError:
                raise ValueError(
                    "--enable-http2 requires the 'granian' package. "
                    'Install it with: pip install "sglang[http2]"'
                )
```

python/sglang/srt/entrypoints/http_server.py

_run_granian_server 新增必填参数并构造 HTTP2Settings 注入 Granian，单 / 多 tokenizer worker 两条分支完成接线，是功能落地的关键路径。

```
# python/sglang/srt/entrypoints/http_server.py 中的关键片段（合并展示）
def _run_granian_server(
    host,
    port,
    log_level,
    http2_max_concurrent_streams, # 新增必填参数，所有调用方必须显式传入
    tokenizer_worker_num=1,
    ssl_certfile=None,
    ssl_keyfile=None,
    ssl_ca_certs=None,
    ssl_keyfile_password=None,
    ssl_verify=False, # MTLS is not supported
    backlog=2048,
    backpressure=2048,
):
```

```
    import signal

    from granian import Granian
    from granian.constants import HTTPModes, Interfaces, Loops
    from granian.http import HTTP2Settings
    from granian.server.embed import Server as GranianEmbeddedServer

    # 单 tokenizer worker 走进程内嵌 server，多 worker 走 Granian 多进程模式
    Server = GranianEmbeddedServer if tokenizer_worker_num == 1 else Granian
    target = (
        app if tokenizer_worker_num == 1 else "sglang.srt.entrypoints.http_server:app"
    )
    granian_kwargs = dict(
        target=target,
        address=host,
        port=port,
        interface=Interfaces.ASGI,
        http=HTTPModes.auto,
        # 将用户配置的并发流上限写入 Granian 的 HTTP/2 设置，
        # 连接建立时通过 SETTINGS 帧广播给客户端
        http2_settings=HTTP2Settings(
            max_concurrent_streams=http2_max_concurrent_streams
        ),
        log_level=log_level,
        ssl_cert=ssl_certfile,
        ssl_key=ssl_keyfile,
        ssl_key_password=ssl_keyfile_password,
        ssl_ca=ssl_ca_certs,
        ssl_client_verify=ssl_verify,
        backlog=backlog,
        backpressure=backpressure,
    )

    if tokenizer_worker_num > 1:
        granian_kwargs["workers"] = tokenizer_worker_num
        granian_kwargs["loop"] = Loops.uvloop

    server = Server(**granian_kwargs)
    # ... 后续 SIGINT/SIGTERM 信号处理与 serve 逻辑与之前一致 ...
```

test/registered/unit/entrypoints/test_http2_server_config.py

新增 CPU 单元测试，用 FakeEmbeddedServer 拦截 Granian Server 构造，断言显式值进入 HTTP2Settings，是该参数接线正确性的核心验证。

```
# test/registered/unit/entrypoints/test_http2_server_config.py（新增文件）
import importlib.util
import unittest
from unittest.mock import patch
```

```
from sglang.srt.entrypoints.http_server import _run_granian_server
from sglang.test.ci.ci_register import register_cpu_ci

register_cpu_ci(est_time=2, suite="base-a-test-cpu")


@unittest.skipUnless(
    importlib.util.find_spec("granian"), "granian is required for HTTP/2"
)
class TestGranianHTTP2Config(unittest.TestCase):
    def test_passes_explicit_max_concurrent_streams(self):
        # FakeEmbeddedServer 拦截 Granian 的内嵌 Server 构造，
        # 在不开真实 socket 的情况下捕获传给 Server 的所有配置
        configured = {}

        class FakeEmbeddedServer:
            def __init__(self, **kwargs):
                configured.update(kwargs)

            async def serve(self):
                return None

            def stop(self):
                return None

        with patch("granian.server.embed.Server", FakeEmbeddedServer):
            _run_granian_server(
                host="127.0.0.1",
                port=30000,
                log_level="info",
                http2_max_concurrent_streams=37,
            )

        # 断言显式传入的值最终进入了 HTTP2Settings
        self.assertEqual(configured["http2_settings"].max_concurrent_streams, 37)


if __name__ == "__main__":
    unittest.main()
```

评论区精华

PR 本身没有任何 review 评论（review_comments_count = 0）。唯一讨论发生在 issue 评论区：作者 cctry 在 CI 失败后指出 startup_weight_load.py 第 122 行读取 server_args.torchao_config 报 AttributeError，判定为与本次改动无关的失败，并两次 /tag-and-rerun-ci 重跑 CI。这段讨论说明本次改动不触碰模型加载路径，主要风险在 CI 排障而非功能逻辑。

- CI 失败是否为本次改动引入 (other): 作者判定为无关的既有失败，重跑 CI 后继续合入流程。

风险与影响

- 风险：
  1. _run_granian_server 签名破坏性变更：http2_max_concurrent_streams 作为必填位置参数插入在 log_level 与 tokenizer_worker_num 之间，仓库内 _setup_and_run_http_server 的两个调用分支已同步更新，但若存在其他直接调用该函数的测试或外部脚本，会因缺参直接 TypeError。
  2. Granian 版本兼容性：from granian.http import HTTP2Settings 依赖较新版本 Granian；若用户环境的 granian 版本不支持 HTTP2Settings.max_concurrent_streams，--enable-http2 启动会失败。PR 未声明最低 granian 版本，也未在 pyproject.toml 中锁定依赖。
  3. 默认值行为变化：默认 200 是否与 Granian 内置默认一致未在 PR 中说明；如果不一致，现有 --enable-http2 用户的连接级并发行为会静默变化。
  4. 校验边界：范围校验只发生在 enable_http2 分支内，若用户设置了非法值但未开启 --enable-http2，参数不会报错，属于静默失效的误配场景。
  5. 测试盲区：端到端测试只验证服务器能启动并正常响应，未断言实际广播的 SETTINGS_MAX_CONCURRENT_STREAMS 值等于 64；该断言目前只存在于 mock 单元测试中。 - 影响：用户侧：对使用 HTTP/2 连接池的客户端（长连接代理、SDK 连接池）提供了可预期的准入控制依据，可以把并发流上限对齐到模型服务能力，减少传输层静默排队。系统侧：影响仅限 Granian 连接设置，不涉及调度、KV Cache、模型推理等核心路径；但连接级并发数会影响内存占用与调度压力，大值配置需谨慎。团队侧：新增参数遵循 ServerArgs 既有模式（字段 + NS 分组 + handle* 校验），维护成本低；FakeEmbeddedServer 拦截构造的测试模式可复用于后续 Granian 配置项。 - 风险标记：Granian HTTP2Settings 版本兼容性，_run_granian_server 签名破坏性变更，默认值行为变化，端到端测试未断言广播值

关联脉络

- PR #34892 feat: add safeguards for remote media URLs: 同样在 server_args.py 新增参数并配套 server_arguments.mdx 文档，印证了参数 + 校验 + 文档的标准流程。
- PR #34265 config: a named entry point for the resolution pipeline, and the last dynamic config read: 同为 ServerArgs 配置体系演进的一部分，说明本次新增参数落在配置迁移大背景下，后续需要纳入配置 bag 读取体系。