PR #29684 完整报告

sgl-project/sglang

[passthrough] engine: zstd request-body decompression + header overrides

合并时间：2026-07-01 14:48

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/29684

执行摘要

- 一句话：为 HTTP 入口添加 zstd 解压缩与头部覆盖
- 推荐动作：此 PR 设计清晰，采用环境变量门控、ASGI 中间件模式，值得参考。建议阅读实现中的错误处理和类型转换细节，特别是 wrapped_receive 模拟 ASGI 完整体的技巧。

功能与动机

允许上游路由器转发请求时无需处理请求体（解压缩、解析、重序列化），减少 upstream→engine 跳的序列化 / 压缩开销。引用 PR 描述："Adds two optional, env-gated HTTP request-ingress features so an upstream can forward request bodies without rewriting them" 以及 "This avoids decompressing, parsing, and re-serializing/re-compressing the body at the upstream on every request, cutting the serialization/compression overhead on the upstream→engine hop."

实现拆解

1. 解压缩中间件(http_request_decompression.py)：新建 ASGI 中间件类 RequestDecompressionMiddleware，在 __call__ 中检查请求头 x-body-compressed。若存在且值为 zstd，则异步收集完整请求体、在 run_in_executor 中释放 GIL 调用 C 库解压缩，之后删除原 content-length 和 x-body-compressed 头，设置正确的 content-length，并提供一个 wrapped_receive 让后续应用层读取解压后的完整体。不支持的压缩方法返回 400；非 HTTP 请求直接透传。
2. 头部覆盖函数(request_headers.py)：定义字典 _HEADER_OVERRIDES 映射头部名到（属性名，类型）对，包括 rid、bootstrap_host、routed_dp_rank 等字段。apply_header_overrides 遍历头部，若存在则类型转换后 setattr，转换失败引发 HTTPException (400)。
3. 入口集成(http_server.py)：在 lifespan 中条件性添加 RequestDecompressionMiddleware；在 generate_request 端点中条件性调用 apply_header_overrides。
4. 配置开关(environ.py)：新增 SGLANG_ENABLE_REQUEST_DECOMPRESSION 和 SGLANG_ENABLE_REQUEST_HEADER_OVERRIDES 两个 EnvBool，默认 False。
5. 依赖声明(pyproject.toml)：添加 zstandard 依赖。测试文件（test_request_decompression.py、test_request_headers.py）覆盖正常 / 异常 / 分块 / 非

法输入等场景。

关键文件：

- python/sglang/srt/entrypoints/http_request_decompression.py（模块 入口层；类别 source；类型 dependency-wiring；符号 _zstd_decompress, _rewrite_headers, RequestDecompressionMiddleware, init）：核心实现：解压缩中间件，是 PR 最重要的新文件。
- python/sglang/srt/entrypoints/request_headers.py（模块 入口层；类别 source；类型 dependency-wiring；符号 apply_header_overrides）：核心实现：定义请求字段与头部映射关系，提供带类型转换的覆盖函数。
- python/sglang/srt/entrypoints/http_server.py（模块 入口层；类别 source；类型 dependency-wiring）：集成点：将中间件和头部覆盖挂载到 FastAPI 应用和生成请求端点。
- python/sglang/srt/environ.py（模块 配置层；类别 source；类型 core-logic）：配置入口：添加两个环境变量开关，默认关闭。
- test/registered/cpu/test_request_decompression.py（模块 测试；类别 test；类型 test-coverage；符号 _drive, receive, send, app）：完整覆盖解压缩中间件的各种路径，是质量保障的关键。

关键符号：RequestDecompressionMiddleware.call, apply_header_overrides, _zstd_decompress, _rewrite_headers

关键源码片段

python/sglang/srt/entrypoints/http_request_decompression.py

核心实现：解压缩中间件，是 PR 最重要的新文件。

```
"""Pure-ASGI middleware that decompresses compressed request bodies.

Gated on `SGLANG_ENABLE_REQUEST_DECOMPRESSION` and request header
`x-body-compressed`, whose value names the method. For example, a caller that
compressed the body with zstd sets the `x-body-compressed: zstd` header.
"""

import asyncio
import io
import logging

import zstandard
from fastapi.responses import Response
from starlette.datastructures import Headers

logger = logging.getLogger(__name__)


# zstd 解压缩函数，使用 stream_reader 以支持大流
# run_in_executor 会释放 GIL，避免阻塞事件循环
def _zstd_decompress(raw: bytes) -> bytes:
```

```
    return zstandard.ZstdDecompressor().stream_reader(io.BytesIO(raw)).read()


# 可扩展的解压缩方法注册表，目前仅支持 zstd
_DECOMPRESSORS = {"zstd": _zstd_decompress}


def _rewrite_headers(headers, new_len):
    """Remove old content-length and x-body-compressed, add new content-length."""
    out = [
        (k, v)
        for (k, v) in headers
        if k not in (b"content-length", b"x-body-compressed")
    ]
    out.append((b"content-length", str(new_len).encode()))
    return out


class RequestDecompressionMiddleware:
    """Decompress request body per request header `x-body-compressed`."""

    def __init__(self, app):
        self.app = app

    async def __call__(self, scope, receive, send):
        # 非 HTTP 请求直接透传，不干扰 WebSocket 或 lifespan
        if scope["type"] != "http":
            return await self.app(scope, receive, send)

        # 无压缩请求头则透传
        method = Headers(scope=scope).get("x-body-compressed")
        if method is None:
            return await self.app(scope, receive, send)

        # 不支持的压缩方法返回 400
        decompress = _DECOMPRESSORS.get(method)
        if decompress is None:
            return await Response(
                f"unsupported x-body-compressed {method!r}; "
                f"supported: {sorted(_DECOMPRESSORS)}",
                status_code=400,
            )(scope, receive, send)

        # 收集所有请求体分片
        body = b""
        more_body = True
        while more_body:
            message = await receive()
            if message["type"] != "http.request":
```

```
                return await self.app(scope, receive, send)
            body += message.get("body", b"")
            more_body = message.get("more_body", False)

        # 在单独的线程中解压缩，避免阻塞事件循环 (zstd 释放 GIL)
        try:
            loop = asyncio.get_running_loop()
            body = await loop.run_in_executor(None, decompress, body)
        except Exception as e:
            logger.warning("request body decompress failed: %s", e)
            return await Response("decompress failed", status_code=400)(
                scope, receive, send
            )

        # 更新 scope 中的头部（副本），替换 content-length，删除 x-body-compressed
        scope = dict(scope)
        scope["headers"] = _rewrite_headers(scope["headers"], len(body))

        # 提供伪造的 receive 使下游读到完整的解压后 body
        body_sent = False

        async def wrapped_receive():
            nonlocal body_sent
            if not body_sent:
                body_sent = True
                return {"type": "http.request", "body": body, "more_body": False}
            return await receive()

        await self.app(scope, wrapped_receive, send)
```

python/sglang/srt/entrypoints/request_headers.py

核心实现：定义请求字段与头部映射关系，提供带类型转换的覆盖函数。

```
"""Override object fields based on _HEADER_OVERRIDES from header values.

This mechanism allows upstream callers to leave the body opaque
(no parse/merge/re-serialize).
"""

from fastapi import HTTPException

# 请求头部 -> (目标属性名，值类型)
_HEADER_OVERRIDES = {
    "x-override-rid": ("rid", str),
    "x-override-bootstrap-host": ("bootstrap_host", str),
    "x-override-bootstrap-port": ("bootstrap_port", int),
    "x-override-bootstrap-room": ("bootstrap_room", int),
    "x-override-conversation-id": ("conversation_id", str),
    "x-override-routed-dp-rank": ("routed_dp_rank", int),
```

```
    "x-override-disagg-prefill-dp-rank": ("disagg_prefill_dp_rank", int),
}


def apply_header_overrides(obj, headers) -> None:
    """Override request based on header values. Fail the request when any override has issues.
    """
    for header, (attr, cast) in _HEADER_OVERRIDES.items():
        value = headers.get(header)
        if value is None:
            continue
        try:
            setattr(obj, attr, cast(value))
        except ValueError as e:
            # 类型不匹配时向客户端返回清晰错误，避免默默忽略
            raise HTTPException(
                status_code=400, detail=f"invalid {header} header {value!r}: {e}"
            ) from e
```

评论区精华

无实质 review 讨论，PR 由合并者直接批准。

- 暂无高价值评论线程

风险与影响

- 风险：风险较低：两个功能均默认关闭，且无推理路径改动。潜在风险：新依赖 zstandard 增加安装体积（但属于常见库）；解压缩中间件将请求体全部收集到内存后解压，若请求体过大可能引发 OOM（但 FastAPI 已有默认请求大小限制，且这是入口中间件常见模式）；头部覆盖仅作用于预定义的属性列表，类型转换错误会直接返回 400 拒绝请求，不会导致错误状态。测试覆盖了主要路径和异常分支。
- 影响：对用户：默认无影响，需主动设置环境变量启用。启用后，上游可发送压缩请求体并利用头部覆盖，减少网络传输量和上游处理开销。对系统：不会影响推理性能，仅请求入口增加少量分支检查。对团队：增加了两个独立模块，需维护头部映射列表与压缩方法注册表。
- 风险标记：新增外部依赖，可选功能默认关闭，请求体内存收集

关联脉络

- 暂无明显关联 PR