微服务部署之ZeroRpc简介

ZeroRpc是一款高性能的Rpc服务器.用于在部署微服务时提供远程调用功能.让程序员以接近函数调用的方式执行远程的代码.

和时下流行的Restful相比

RPC的优点

- 以接近函数调用的方式执行远程的代码,让程序员专注于代码逻辑

Restful的优点

- 前后端通用.无语言障碍

性能方面RPC VS RESTFUL 测试下来有接近1倍的性能优势

部署方式

部署时,代码主要分3个部分

- **业务逻辑代码** 提供业务逻辑的具体实现.
- **Restful接口代码** 提供基于htpp的api请求服务.
- **zero-rcp服务代码** 提供基于rpc的远程调用服务

实例

下面我们举个例子进行说明:

为了简便起见,本例使用flask+zeorpc进行的展示,换成django+zerorpc亦可

安装zerorpc

```
pip install zerorpc
或者
pip3 install zerorpc
```

创建一个目录/项目文件夹.新建4个py文件.

- my_service.py 存放业务逻辑的具体实现
- rest_server.py 提供基于htpp的api请求服务
- zerorpc_server.py 提供基于rpc的远程调用服务
- zerorpc_client.py 客户端文件.演示了调用方式.进行简单的性能对比

my_service.py

```python
# -*- coding: utf-8 -*-
import os
import sys
__project_dir__ = os.path.dirname(os.path.dirname(os.path.realpath(__file__)))
if __project_dir__ not in sys.path:
    sys.path.append(__project_dir__)


"""业务逻辑"""

users = []


class Server1:

    def add_user(self, user_name: str):
        if user_name in users:
            return "{} exists!".format(user_name)
        else:
            users.append(user_name)
            return "ok"


class Server2:
    def find_user(self, user_name):
        if user_name in users:
            return user_name
        else:
            return "not found!"

    def all_user(self):
        return ",".join(users) if len(users) > 0 else ""


if __name__ == "__main__":
    pass
```

rest_server.py

```python
from flask import Flask
from flask import request
import json
from zerorpc_test.my_service import *


app = Flask(__name__)


"""restful接口服务器"""


def check_auth(auth):
    """
    检查token的装饰器.
    这里只是示范,实际工作中代码要复杂一些
    :param auth:  authorization
    :return:
    """
    """
    检查的结果 :
    0表示没有authorization
    -1表示权限不足
    1表示检查通过
    """
    validated = 0

    if auth is None:
```

```
        pass
    else:
            """
            调用authorization检查的微服务根据检查结果确认validated的值.
            """
            pass
    return validated


@app.route("/some_api")
def index_func():
    """
    视图函数
    :return:
    """
    user_name = request.args.get("user_name")
    a = Server1()
    if user_name is not None:
        r = a.add_user(user_name=user_name)
    b = Server2()
    return b.all_user()


@app.before_request
def permission_check():
    """
    在每次请求前进行检查.
    :return:
    """
    auth = request.headers.get("authorization")
    validated = check_auth(auth=auth)
    validated = 1  # 测试的时候人为赋值
    """
    0表示没有authorization
    -1表示权限不足
    1表示检查通过
    """
    if validated == 0:
        return json.dumps({"message": "401"})
    elif validated == -1:
        return json.dumps({"message": "rejected"})
    else:
        """放行"""
        pass


if __name__ == "__main__":
    app.run(host="0.0.0.0", port=8003)
    pass
```

zerorpc_server.py

```
# -*- coding: utf-8 -*-
import os
import sys
__project_dir__ = os.path.dirname(os.path.dirname(os.path.realpath(__file__)))
if __project_dir__ not in sys.path:
    sys.path.append(__project_dir__)
import zerorpc
from zerorpc_test.my_service import *

"""zero-rpc的服务器"""


class Cooler(
                        Server1,
                        Server2
                    ):
    """
    服务器,只需要继承具体的服务类即可.注意这些服务类中不可有同名的实例方法
    """


if __name__ == "__main__":
    port = 4242
    s = zerorpc.Server(Cooler())
    s.bind("tcp://0.0.0.0:{}".format(port))
    print("zero-rpc running on {} ...".format(port))
    s.run()
```

zerorpc_client.py

```
# -*- coding: utf-8 -*-
import zerorpc
import requests
import datetime


c = zerorpc.Client()
c.connect("tcp://0.0.0.0:4242")  # 连接到rpc服务器

user_name1 = "jack"
user_name2 = "tom"
b1 = datetime.datetime.now()
n = 500
for i in range(n):
    c.add_user(user_name1)
    c.add_user(user_name2)
    print(c.all_user(), end="    ")
e1 = datetime.datetime.now()
print()
print("{}次rpc调用耗时: {}秒".format(n, (e1 - b1).total_seconds()))
b2 = datetime.datetime.now()
ses = requests.Session()
u = "http://127.0.0.1:8003/some_api"
params = {"user_name": user_name1}
for i in range(n):
    r = ses.get(url=u, params=params)
    print(r.text, end="    ")
e2 = datetime.datetime.now()
print()
print("{}次restful调用耗时: {}秒".format(n, (e2 - b2).total_seconds()))
```

测试的时候,先启动zerorpc_server.py和rest_server.py.然后运行zerorpc_client.py进行测试

实际部署时.每个人将自己负责的模块部分,需要开放给前端的Api以Restful的方式暴露出来;只允许后端调用的Api以Rpc的方式

部署.
在以Rpc方式部署时,大家需要先协商号端口号的分配以避免端口号冲突.