基于Netty的服务网关

依赖

- Netty 4.1.19.Final
- Thrift 0.10.0
- CuratorFramework

目标功能

没有验证过文件上传下载的正确性。因为在alphadog里没有看到文件上传的api，文件下载返回的只是文件地址。

1. NIO
2. 可动态更新路由

   路由加载方式

   - 网关启动时通过加载配置文件 `routing.yaml`

     ```
     thriftServices:
       - http: POST /echo # http请求
         convertor: # 解析请求和生成响应的类
         service: echoservice # 下游服务名（zookeeper中的名字）
         method: echo # 下游服务方法名
         clazz: # thrift生成的类名
       - http: ....
       ...
     xxxServices:
     ```

   - 支持模版匹配 @PathVar注入

     ```
         - http: POST /echo/{id}
     ```

   - 网关启动时通过扫描package加载@Router类
   - 通过客户端工具运行时更改路由配置（**未完成**）

     作为一种特殊的请求，同样在netty中处理。

   - 路由的数据结构保证读取的速度，对路由的更新通过创建一个新的hashmap实例的方式防止线程阻

塞。

3. http和rpc转换

   thrift: 默认和alphadog配置保持一致

   - protocol：TMultiplexedProtocol --内嵌-> TCompactProtocol
   - transport： TFramedTransport

4. 服务发现/负载均衡

   - 服务发现和netty线程分开，作为一个background thread运行。
   - 使用一个zookeeper连接，配置所有的服务，每个服务对应一个本地缓存实例，缓存实时从zookeeper获得更新。
   - 负载均衡默认采用轮询。负载均衡的数据结构保证获取服务实例的速度，节点更新的操作在zookeeper线程中串行执行。

5. 动态添加（删除）过滤器

   网关提供两种类型的过滤器。preRouting, routing和postRouting过滤器。（和Zuul过滤器类似）

   - preRouting: 刚解析完http请求后
   - routing: 调用下游服务前 **（未完成）**
   - postRouting: 生成http响应后

   加载过滤器方式： * 网关启动前通过代码配置 * 运行时通过客户端工具 **（未完成成）** 1. 加载新的filter 2. disable已加载的filter 3. enable被disable的filter

6. 熔断 **（未完成）**

   熔断完全基于过滤器实现。通过记录下游服务响应的不同状态的次数（成功，失败，超时等），决定对于之后请求的操作（例：直接返回）。（原理和hystrix类似）

7. 错误处理

   request运行过程中抛出异常 和 请求被过滤器过滤等特殊情况，自动快速产生一个对应的http响应。

请求-响应生命周期

1. 从数据流中解码，得到的http请求
2. preRouting过滤器。
3. 查询http请求对应的下游服务，选择服务节点（服务发现，负载均衡）
4. routing过滤器
5. 将http请求转换成thrift请求参数（这里内部调用配置的http请求解析器）
6. 连接服务节点（生成rpc channel），将thrift请求参数编码成数据流发送至下游服务

7. 从数据流中解码，得到的thrift响应
8. 将thrift响应转换成http响应（这里内部调用配置的http响应生成器）
9. postRouting过滤器
10. 将http响应编码成数据流发送回客户端

扩展

关于如何扩展

Demo

sgw.demo.DemoServer