0.分布式服务框架和RPC参考博客

<https://github.com/tang-jie/NettyRPC>

<https://www.cnblogs.com/jietang/p/5615681.html>

**分布式服务框架设计文档**

**1. 分布式服务框架的简要架构图**



**2. RPC私有协议设计**

**2.1服务注册**

服务端某一具体的服务用ProviderBean描述(具备唯一的服务名称ServiceName、版本号Version、权限定实例类名称、ip地址和端口号)

Provider:

String serviceName

int version

String interfaceName

String ip:port

**上述服务向Zookeeper的注册信息如下**

**serviceName/ip:port**

**2.2 Request消息体**

int type(消息类型)

String methondName(方法名称)

Map<String,param>param携带参数

int version 版本号

int messageId (消息请求id，保证并发条件下的唯一性质)

**2.3 Response 消息体**

int type(消息返回类型)

int messageId(与Reqeust中的messageId对应)

Object resultData(调用的返回结构)

**2.4 序列化协议**

1.MessagePack二进制序列化协议

2.Json序列化协议(目前支持)

**3. 客户端调用逻辑**

**3.1 消费对象Cosumer，对外提供缴费接口**

**错误！嵌入对象无效。**

**3.1.1调用形式**

3.1.1.1 同步调用

Consumer.comsume(ServiceName, methondName, version)

3.1.1.2 future异步调用

3.1.1.3 callback异步调用(请求队列，回调队列)

**3.1.2 MessageRouter:**

1.保持和Zookeeper的连接，缓存ServiceName – socketIP的列表数据,Zookeeper实时推送列表数据，更新缓存列表。

2.ServiceName 不存在时，请求Zookeeper的数据

3.执行负载均衡逻辑

**3.1.3 ConnectionPool**

缓存到多个ip的的Socket连接，避免重复建立连接，实现连接复用

**4.服务端处理调用逻辑**



**1. 服务注册表**

**2. 服务隔离**

**3. 动态配置，是否在IO线程中处理业务逻辑**

**4. 流量控制(选做)**

**5. 分布式服务的部署方式**

**(1)JAR包的部署方式**

**(2)spring的管理方式**