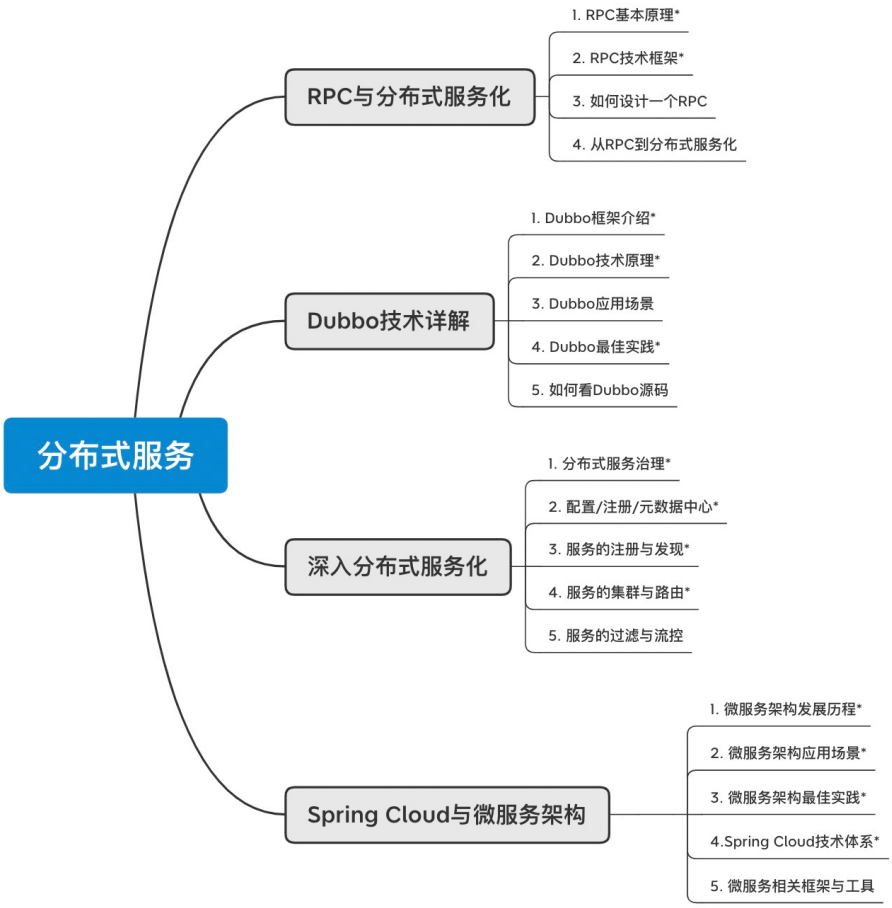
Week9-01 ...第十七课.pdf
2021-05-07 15:39:58 • 861 KB




```
第17课程—RPC与分布式服务化

目录

1. RPC基本原理*

RPC是远程过程调用(Remote Procedure Call)的缩写形式。

分布式基石：rpc  mq

1981年提出；1984年实现；

简单来说，就是“像调用本地方法一样调用远程方法”。webservie 是一种rpc实现
http是网络协议，rest不是rpc ，  本地封装好就是rpc restful 是采用rest的架构的实现
resteasy  jersey  都是rpc
RPC的简化版原理如下图。 核心是代理机制。
1.本地代理存根: Stub
2.本地序列化反序列化
3.网络通信
4.远程序列化反序列化
```

```
5.远程服务存根: Skeleton
6.调用实际业务服务
7.原路返回服务结果
8.返回给本地调用方
rpc原理

1.设计: 接口定义  pojo定义共享


2.代理 动态代理 aop      反射    字节码增强

3.序列化

4.网络传输  tcp

5.查找实现类




查看 tcp协议传输数据  抓包

tcpdum

wireshark




2. RPC技术框架*
java: RMI
。net: Remoting
corba com
3. 如何设计一个RPC
4. 从RPC到分布式服务化
5. 总结回顾与作业实践
```

目录

第17课

1. RPC基本原理

- RPC是什么

  RPC是远程过程调用(Remote Procedure Call)的缩写形式。

RPC的概念与技术早在1981年由Nelson提出。

1984年，Birrell和Nelson把其用于支持异构型分布式系统间的通讯。Birrell的RPC 模型 引入存根进程(stub) 作为远程的本地代理，调用RPC运行时库来传输网络中的调用。 Stub和RPC runtime屏蔽了网络调用所涉及的许多细节，特别是，参数的编码/译码及网 络通讯是由stub和RPC runtime完成的，因此这一模式被各类RPC所采用。

- RPC是什么

  什么叫RPC呢?

  简单来说，就是“像调用本地方法一样调用远程方法”。

  UserService service = new UserService(); User user = service.findById(1);

  UserService service = Rpcfx.create(UserService.class, url); User user = service.findById(1);

  如何能做到本地方法调用时转换成远程?

- RPC原理

  RPC的简化版原理如下图。 核心是代理机制。

  1.本地代理存根: Stub

  2.本地序列化反序列化

  3.网络通信

  4.远程序列化反序列化

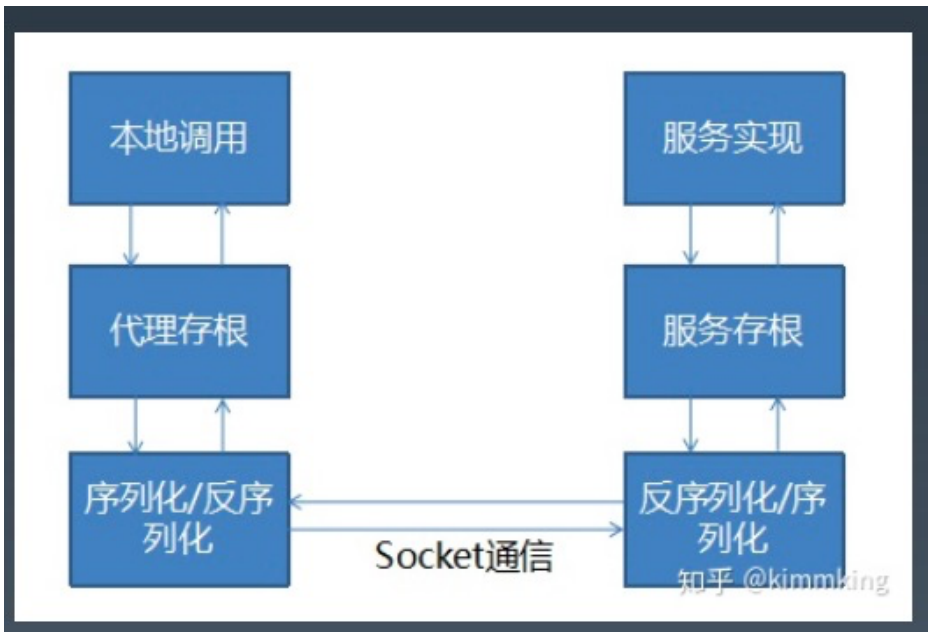
  5.远程服务存根: Skeleton

  6.调用实际业务服务

  7.原路返回服务结果

  8.返回给本地调用方

  注意处理异常。



- RPC原理--1.设计

  RPC是基于接口的远程服务调用。

  本地应用程序 与 远程应用程序，分别需要共享什么信息，角色有什么不同?

  共享:POJO实体类定义，接口定义。 REST/PB下，真的不需要嘛?另一种选择:WSDL/WADL/IDL 远程->服务提供者，本地->服务消费者。

- RPC原理--2.代理

  RPC是基于接口的远程服务调用。

  Java下，代理可以选择动态代理，或者AOP实现

  - C#直接有远程代理

  - Flex可以使用动态方法和熟悉

- RPC原理--3.序列化

  序列化和反序列化的选择:

  1、语言原生的序列化，RMI，Remoting

  2、二进制平台无关，Hessian，avro，kyro，fst等

  3、文本，JSON、XML等

- RPC原理--4.网络传输

  最常见的传输方式:

  - TCP/SSL

  - HTTP/HTTPS

- RPC原理--5.查找实现类

  通过接口查找服务端的实现类。

  一般是注册方式，

  例如 dubbo 默认将接口和实现类配置到Spring

2. RPC技术框架

- RPC技术框架

  很多语言都内置了RPC技术。

  Java RMI

  .NET Remoting

  远古时期，就有很多尝试，

  - Corba(Common ObjectRequest Broker Architecture)公共对象请求代理体系结 构，OMG组织在1991年提出的公用对象请求代理程序结构的技术规范。底层结构是基于 面向对象模型的，由OMG接口描述语言(OMG Interface Definition Language，OMG IDL)、对象请求代理(Objec tRequest Broker，ORB)和IIOP标准协议(Internet Inter ORB Protocol，也称网络ORB交换协议)3个关键模块组成。

  - COM(Component Object Model，组件对象模型)是微软公司于1993年提出的一种 组件技术，它是一种平台无关、语言中立、位置透明、支持网络的中间件技术。很多老 一辈程序员心目中的神书《COM本质论》。

- 常见的RPC技术

  - Corba/RMI/.NET Remoting

  - JSON RPC, XML RPC，WebService(Axis2, CXF) - Hessian, Thrift, Protocol Buffer, gRPC

  Hessian

  Thrift gRPC

3. 如何设计一个RPC框架

- 假如我们自己设计一个RPC框架

  从哪些方面考虑? 基于共享接口还是IDL?

  动态代理 or AOP? 序列化用什么?文本 or 二进制? 基于TCP还是HTTP?

还有没有要考虑的?

- RPC原理--1.设计

  共享:POJO实体类定义，接口定义。

  REST/PB下，真的不需要嘛?另一种选择:WSDL/WADL/IDL 远程->服务提供者，本地->服务消费者。

  ---------- rpcfx 里的 api 子项目

- RPC原理--2.代理

  RPC是基于接口的远程服务调用。

  Java下，代理可以选择动态代理，或者AOP实现 ---------- rpcfx 里的 默认使用 动态代理

- RPC原理--3.序列化

  序列化和反序列化的选择:

  1、语言原生的序列化，RMI，Remoting

  2、二进制平台无关，Hessian，avro，kyro，fst等

  3、文本，JSON、XML等

  ---------- rpcfx 里的 默认使用 JSON

- RPC原理--4.网络传输

  最常见的传输方式:

  - TCP/SSL

  - HTTP/HTTPS

  ---------- rpcfx 里的 默认使用 HTTP

- RPC原理--5.查找实现类

  通过接口查找具体的业务服务实现。

  ---------- rpcfx 里的 默认使用 Spring getBean

4. 从RPC到分布式服务化

- 从RPC走向服务化->微服务架构

  具体的分布式业务场景里，除了能够调用远程方法，我们还需要考虑什么?

  1、多个相同服务如何管理?

  2、服务的注册发现机制?

  3、如何负载均衡，路由等集群功能?

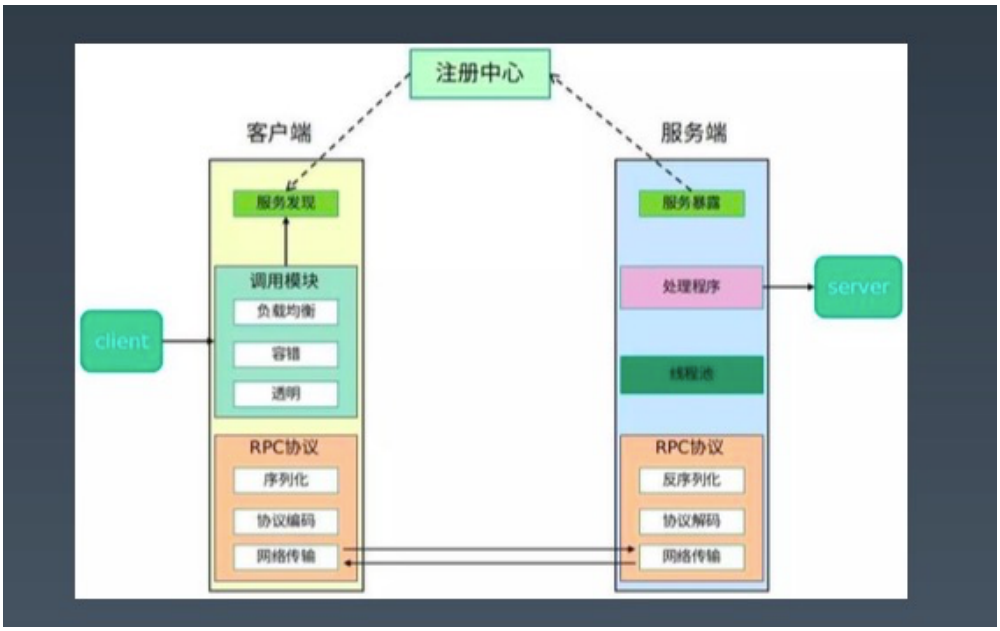
  4、熔断，限流等治理能力。

  5、重试等策略

  6、高可用、监控、性能等等。

- 从RPC走向服务化

  一个典型的分布式服务化架构



第17课 5.总结回顾与作业实践

第 17 课作业实践

1、(选做)实现简单的Protocol Buffer/Thrift/gRPC(选任一个)远程调用demo。

2、(选做)实现简单的WebService-Axis2/CXF远程调用demo。

3、(必做)改造自定义RPC的程序，提交到github:

1)尝试将服务端写死查找接口实现类变成泛型和反射

2)尝试将客户端动态代理改成AOP，添加异常处理

3)尝试使用Netty+HTTP作为client端传输方式

4、(选做☆☆)升级自定义RPC的程序:

1)尝试使用压测并分析优化RPC性能

2)尝试使用Netty+TCP作为两端传输方式

3)尝试自定义二进制序列化

4)尝试压测改进后的RPC并分析优化，有问题欢迎群里讨论

5)尝试将fastjson改成xstream

6)尝试使用字节码生成方式代替服务端反射