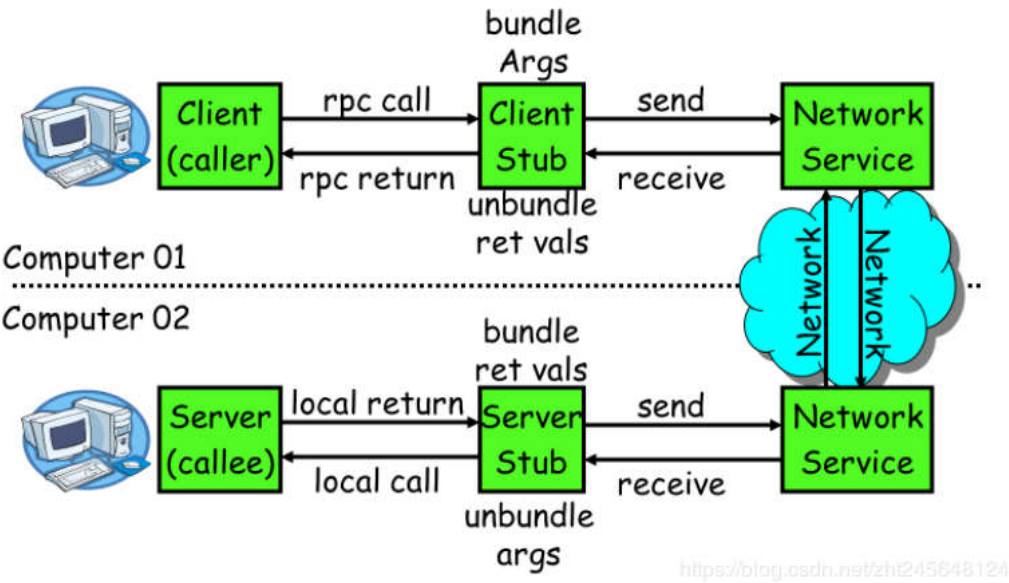


RPC通信基本原理

转载 zht245648124 最后发布于2019-05-19 11:54:37 阅读数 1013 ☆ 收藏

一、RPC简介

- 1) RPC，全称为Remote Procedure Call，即远程过程调用，它是一个计算机通信协议。它允许像调用本地服务一样调用远程服务。它可以有不同的实现，如RMI(远程方法调用)、Hessian、Http invoker等。另外，RPC是与语言无关的。
- 2) RPC示意图



如上图所示，假设Computer1在调用sayHi()方法，对于Computer1而言调用sayHi()方法就像调用本地方法一样，调用->返回。但从后续调用可以看出Computer1调用的是Computer2中的sayHi()方法，RPC屏蔽了底层的实现细节，让调用者无需关注网络通信，数据传输等细节。

二、RPC框架的实现

上面介绍了RPC的核心原理：RPC能够让本地应用简单、高效地调用服务器中的过程（服务）。它主要应用在分布式系统。如Hadoop中的IPC组件。你了解几个RPC框架呢？

从下面几个方面思考，仅供参考：

- 1.通信模型：假设通信的为A机器与B机器，A与B之间有通信模型，在Java中一般基于BIO或NIO；。
- 2.过程（服务）定位：使用给定的通信方式，与确定IP与端口及方法名称确定具体的过程或方法；
- 3.远程代理对象：本地调用的方法(服务)其实是远程方法的本地代理，因此可能需要一个远程代理对象，对于Java而言，远程代理对象可以使用Java的实现，封装了调用远程方法调用；
- 4.序列化，将对象名称、方法名称、参数等对象信息进行网络传输需要转换成二进制传输，这里可能需要不同的序列化技术方案。如:protobuf，Arvo

三、Java实现RPC框架

1、实现技术方案

下面使用比较原始的方案实现RPC框架，采用Socket通信、动态代理与反射与Java原生的序列化。

2、RPC框架架构

RPC架构分为三部分：

- 1) 服务提供者，运行在服务器端，提供服务接口定义与服务实现类。
- 2) 服务中心，运行在服务器端，负责将本地服务发布成远程服务，管理远程服务，提供给服务消费者使用。

3) 服务消费者，运行在客户端，通过远程代理对象调用远程服务。

3、具体实现

服务提供者接口定义与实现，代码如下：

```
1 package com.rpc.test;
2
3 /**
4  * 服务提供者接口定义与实现
5  */
6 public interface HelloService {
7
8     String sayHi(String name);
9
10 }
11
```

HelloServices接口实现类：

```
1 package com.rpc.test;
2
3 public class HelloServiceImpl implements HelloService {
4
5     public String sayHi(String name) {
6         return "Hi, " + name;
7     }
8
9 }
10
```

服务中心代码实现，代码如下：

```
1 package com.rpc.test;
2
3 import java.io.IOException;
4
5 public interface Server {
6     public void stop();
7
8     public void start() throws IOException;
9
10    public void register(Class serviceInterface, Class impl);
11
12    public boolean isRunning();
13
14    public int getPort();
15 }
16
```

服务中心实现类：

```
1 package com.rpc.test;
2
3 import java.io.IOException;
4 import java.io.ObjectInputStream;
5 import java.io.ObjectOutputStream;
6 import java.lang.reflect.Method;
7 import java.net.InetSocketAddress;
8 import java.net.ServerSocket;
9 import java.net.Socket;
10 import java.util.HashMap;
11 import java.util.concurrent.ExecutorService;
12 import java.util.concurrent.Executors;
13
14 public class ServiceCenter implements Server {
15     private static final HashMap<String, Class> serviceRegistry = new HashMap<String, Class>();
16     private static ExecutorService executor = Executors.newFixedThreadPool(Runtime.getRuntime().availableProcessors());
17     private static boolean isRunning = false;
18
```



1



```
19     private static int port;
20
21     public ServiceCenter(int port) {
22         this.port = port;
23     }
24
25     public void stop() {
26         isRunning = false;
27         executor.shutdown();
28     }
29
30     public void start() throws IOException {
31         ServerSocket server = new ServerSocket(port);
32         // server.bind(new InetSocketAddress(port));
33         System.out.println("start server");
34         try {
35             while (true) {
36                 // 1.//2、调用accept()方法开始监听，等待客户端的连接，监听客户端的TCP连接，接到TCP连接后将其封装
37                 Socket socket = server.accept();
38                 executor.execute(new ServiceTask(socket));
39             }
40         } finally {
41             server.close();
42         }
43     }
44
45     public void register(Class serviceInterface, Class impl) {
46         serviceRegistry.put(serviceInterface.getName(), impl);
47     }
48
49     public boolean isRunning() {
50         return isRunning;
51     }
52
53     public int getPort() {
54         return port;
55     }
56
57     private static class ServiceTask implements Runnable {
58         Socket client = null;
59
60         public ServiceTask(Socket client) {
61             this.client = client;
62         }
63
64         public void run() {
65             ObjectInputStream input = null;
66             ObjectOutputStream output = null;
67             try {
68                 // 2.将客户端发送的码流反序列化成对象，反射调用服务实现者，获取执行结果
69                 input = new ObjectInputStream(client.getInputStream());
70                 String serviceName = input.readUTF();
71                 String methodName = input.readUTF();
72                 Class<?>[] parameterTypes = (Class<?>[]) input.readObject();
73                 Object[] arguments = (Object[]) input.readObject();
74                 Class serviceClass = serviceRegistry.get(serviceName);
75                 if (serviceClass == null) {
76                     throw new ClassNotFoundException(serviceName + " not found");
77                 }
78                 Method method = serviceClass.getMethod(methodName, parameterTypes);
79                 Object result = method.invoke(serviceClass.newInstance(), arguments);
80
81                 // 3.将执行结果反序列化，通过socket发送给客户端
82                 output = new ObjectOutputStream(client.getOutputStream());
83                 output.writeObject(result);
84             } catch (Exception e) {
85                 e.printStackTrace();
86             } finally {
87                 if (output != null) {
88                     try {
89                         output.close();
```



1



, 由线程池执行



举报

```
90         } catch (IOException e) {
91             e.printStackTrace();
92         }
93     }
94     if (input != null) {
95         try {
96             input.close();
97         } catch (IOException e) {
98             e.printStackTrace();
99         }
100     }
101     if (clent != null) {
102         try {
103             clent.close();
104         } catch (IOException e) {
105             e.printStackTrace();
106         }
107     }
108 }
109
110 }
111 }
112 }
113 }
```



1



客户端的远程代理对象:

```
1 package com.rpc.test;
2
3 import java.io.ObjectInputStream;
4 import java.io.ObjectOutputStream;
5 import java.lang.reflect.InvocationHandler;
6 import java.lang.reflect.Method;
7 import java.lang.reflect.Proxy;
8 import java.net.InetSocketAddress;
9 import java.net.Socket;
10
11 /**
12  * 客户端的远程代理对象
13  * @param <T>
14  */
15 public class RPCClient<T> {
16     public static <T> T getRemoteProxyObj(final Class<?> serviceInterface, final InetSocketAddress addr) {
17         // 1. 将本地的接口调用转换成JDK的动态代理, 在动态代理中实现接口的远程调用
18         return (T) Proxy.newProxyInstance(serviceInterface.getClassLoader(), new Class<?>[]{serviceInterface},
19             new InvocationHandler() {
20                 public Object invoke(Object proxy, Method method, Object[] args) throws Throwable {
21                     Socket socket = null;
22                     ObjectOutputStream output = null;
23                     ObjectInputStream input = null;
24                     try {
25                         // 2. 创建Socket客户端, 根据指定地址连接远程服务提供者
26                         socket = new Socket();
27                         socket.connect(addr);
28
29                         // 3. 将远程服务调用所需的接口类、方法名、参数列表等编码后发送给服务提供者
30                         output = new ObjectOutputStream(socket.getOutputStream());
31                         output.writeUTF(serviceInterface.getName());
32                         output.writeUTF(method.getName());
33                         output.writeObject(method.getParameterTypes());
34                         output.writeObject(args);
35
36                         // 4. 同步阻塞等待服务器返回应答, 获取应答后返回
37                         input = new ObjectInputStream(socket.getInputStream());
38                         return input.readObject();
39                     } finally {
40                         if (socket != null) socket.close();
41                         if (output != null) output.close();
42                         if (input != null) input.close();
43                     }
44                 }
45             }
46         );
47     }
48 }
```



举报

```
44         }
45     });
46 }
47 }
48 }
```

👍 1

🔗

💬

📖

☆

📱

<

>

最后为测试类：

```
1 package com.rpc.test;
2
3 import java.io.IOException;
4 import java.net.InetSocketAddress;
5
6 public class RPCTest {
7
8     public static void main(String[] args) throws IOException {
9         new Thread(new Runnable() {
10             public void run() {
11                 try {
12                     Server serviceServer = new ServiceCenter(8088);
13                     serviceServer.register>HelloService.class, HelloServiceImpl.class);
14                     serviceServer.start();
15                 } catch (IOException e) {
16                     e.printStackTrace();
17                 }
18             }
19         }).start();
20        >HelloService service = RPCClient.getRemoteProxyObj>HelloService.class, new InetSocketAddress("localhost", 8088);
21         System.out.println(service.sayHi("test"));
22     }
23 }
24 }
```

运行结果：

```
1 regeist service>HelloService
2 start server
3 Hi, test
```

四、总结

- 1
- RPC本质为消息处理模型，RPC屏蔽了底层不同主机间的通信细节，让进程调用远程的服务就像是本地的服务一样。

五、可以改进的地方

这里实现的简单RPC框架是使用Java语言开发，与Java语言高度耦合，并且通信方式采用的Socket是基于 BIO 实现的，IO 效率不高，还有Java原生的序内存太多，运行效率也不高。可以考虑从下面几种方法改进。


- 1.可以采用基于JSON数据传输的RPC框架；
- 2.可以使用NIO或直接使用Netty替代BIO实现；
- 3.使用开源的序列化机制，如Hadoop Avro与Google protobuf等；
- 4.服务注册可以使用Zookeeper进行管理，能够让应用更加稳定。

👍 点赞 1

☆ 收藏

🔗 分享

...

 zht245648124

发布了28 篇原创文章 · 获赞 4 · 访问量 1万+

私信



🔊

举报

 想对作者说点什么

关于RPC协议的通俗理解

根据网上搜索的一些资料摘抄汇总的，如果有误，欢迎斧正。作者：肖继潮链接：http://www.zhihu.com/question... 博文 来自： huangmr0610 1万+

RPC 通信和 RMI 区别

RPC（Remote Procedure Call Protocol） 远程过程调用协议，通过网络从远程计算机上请求调用某种服务。一个R... 博文 来自： 张花生的博客 746

RPC进程间通信的一种实现

客户端项目中不可避免的要用到进程间的通信，方式也多种多样。单就开发而言，RPC这种模式最方便来做进程间通... 博文 来自： 最爱吹吹风 331

rpc基本原理

1.基本概念Remote Procedure Call,即远程过程调用,是一个计算机通信协议。该协议允许运行于一台计算机的程序调... 博文 来自： 技术改变生活 195

RPC进程通信小模型

这几天因为工作的需要，学习了一下RPC进程通信，RPC确实是个好东西啊， 博文 来自： 独孤龙城的专栏 957

RPC基本原理

转载：https://blog.csdn.net/zkp_java/article/details/81879577RPC基本原理RPC(Remote Procedure Call)，远... 博文 来自： 你的博客 344

RPC框架底层通信原理

RPC（Remote Procedure Call）即远程过程调用，它是一种通过网络从远程计算机程序上请求服务，而不需要了解... 博文 来自： aloysjun的博客 871

RPC原理介绍

面向服务架构SOA任何大型网站的发展都伴随着网站架构的演进。网站架构一般最初是单应用设计，然后逐渐经历面... 博文 来自： elricboa的专栏 8675

RPC服务和HTTP服务对比

很久时间以来都没有怎么好好搞清楚RPC（即RemoteProcedureCall，远程过程调用）和HTTP调用的区别，不都是... 博文 来自： 王云朋的专栏 19万+

如何满足她？每次60分钟，多吃它，想多硬就多硬！

星之·猎媒

Gradle入门二——创建task

1.在项目的build.gradle中添加新的tasktask hello1{ doLast{ println 'hello from other script' }}添加后... 博文 来自： zht245648124的... 62

RPC与REST的区别

一：RPCRPC即远程过程调用,很简单的概念,像调用本地服务(方法)一样调用服务器的服务(方法).通常的实现有XML-R... 博文 来自： 十五楼亮哥 2万+



huangmr0610
281篇文章
排名:2000+
[关注](#)



Chackca
85篇文章
排名:千里之外
[关注](#)



最爱吹吹风
155篇文章
排名:千里之外
[关注](#)



是小萌子呀
13篇文章
排名:千里之外
[关注](#)

RPC定义(一)

在分布式环境中，客户机和服务器在不同的机器上运行，客户端调用在服务器端运行的过程，并把结果发送回客户机... 博文 来自： yaotai8135的博客 507

Http与RPC通信协议的比较

OSI网络结构的七层模型各层的具体描述如下： 第七层：应用层 定义了用于在网络中进行通信和数据传输的接口 ... 博文 来自： tiandirensoon 94

orientDB导入CSV数据

此示例描述了将CSV文件作为图形导入OrientDB的过程。为简单起见，仅考虑以下两个实体：贴吧（post）评论（C... 博文 来自： zht245648124的... 1001



眼睛换人工晶体需要多少费用
眼睛手术费用多少

C# RPC远程方法调用框架thrift

首先项目创建windows控制台程序，项目里面引用 写这篇文章时用的是thrift-csharp版本0.10.0 项目结构 服务端代... 博文 来自： u011511011的专栏 1943

RPC与其实现方式概念笔记

一，消息队列服务一般用于设计多系统之间的信息传输，一般这种传输不需要对方对数据做出回应。它最常见的方式... 博文 来自： silyvin 5249

RPC通信原理	阅读数 4
一句话总结:RPC: remote procedure call Protocol 远程过程调用调用远程服务, 就像调用本地的服务一样, 不用关... 博文 来自: weixin_3c4a7701...	<div><div>1</div><div>562</div></div>
Kafka集群搭建详细步骤 (基于版本2.2.0)	
1、Kafka的安装需要java环境, cent os 7最好自己重新安装jdk1.8以上; 2、准备zookeeper搭建zookeeper集群详... 博文 来自: zht245648124的...	<div><div></div><div>924</div></div>
一篇文章了解RPC框架原理	
1.RPC框架的概念RPC (Remote Procedure Call) -远程过程调用, 通过网络通信调用不同的服务, 共同支撑一个软... 博文 来自: 山月记	<div><div></div><div>56</div></div>
高考状元被发现惊人存款, 母亲: 做梦都没想到这是真的	
盛源·猎媒	
RPC研究	
深入浅出RPC——深入篇mindwind · 2014-09-22出处: http://mindwind.me/blog/2014/09/22/深入浅出RPC... 博文 来自: iteye_150	<div><div></div><div>万+</div></div>
Dubbo原理简述一: RPC原理和Netty通信原理	
一、RPC原理: 一次完整的RPC调用流程 (同步调用, 异步另说) 如下: 1) 服务消费方 (client) 调用以本地调用方... 博文 来自: 永远年轻, 永远...	
kafka api 操作	阅读数 126
1. 生产者apiProducer是Kafka三大组件中的一个, 用于发送消息到kafka集群中, Producer提供了丰富的配置 (见... 博文 来自: zht245648124的...	
【转】你应该知道的 RPC 原理	阅读数 327
本文作者: 伯乐在线 - meituanlibaba 链接在校期间大家都写过不少程序, 比如写个hello world服务类, 然后本地... 博文 来自: wangeshen的博客	
RPC通信功能实现	阅读数 9431
Table of ContentsRPC通信功能实现配置参数调用方法RPC通信功能实现HBase的RPC通信功能主要基于Protobuf... 博文 来自: JavaMan_chen的...	
架构设计: 系统间通信 (10) ——RPC的基本概念	阅读数 49
1、概述经过了详细的信息格式、网络IO模型的讲解, 并且通过JAVA RMI的讲解进行了预热。从这篇文章开始我们将... 博文	
分布式下的远程通信技术 (RPC) 的一些理解	阅读数 25
前言为什么需要RPC, 而不是简单的HTTP接口? 刚开始还是菜鸟的时候, 时常把RPC和HTTP搞混淆, 本身概念还没... 博文 来自: weixin_33725807...	
linux系列之常用运维命令整理笔录	阅读数 22万+
本博客记录工作中需要的linux运维命令, 大学时候开始接触linux, 会一些基本操作, 可是都没有整理起来, 加上是... 博文 来自: Nicky's blog	
通俗易懂地给女朋友讲: 线程池的内部原理	阅读数 10万+
餐盘在灯光的照耀下格外晶莹剔透, 女朋友拿起红酒杯轻轻地抿了一小口, 对我说: “经常听你说线程池, 到底线程... 博文 来自: 万猫学社	
用 java 简单实现 rpc 通信	阅读数 343
代码不一定能够运行起来, 这是在之前的代码中抽象出来的。这里只是说说基本的思路定义消息: package com.xia... 博文 来自: chaojilaji的博客	
RPCs及QName小结	阅读数 3289
1 RPCs在常规的Netconf/YANG使用情况下, RPCs被用来模块化Netconf服务器向Netconf客户端提供的功能和AP... 博文 来自: Phoenix	
考察Hadoop的底层rpc通信(一)	阅读数 2478
简介IPC: inter process communication 即进程间通信 RPC: remote procedure call 即远程过程调用 IPC是进程间... 博文 来自: 呼呼的小窝	
有哪些让程序员受益终生的建议	阅读数 15万+
从业五年多, 辗转两个大厂, 出过书, 创过业, 从技术小白成长为基层管理, 联合几个业内大牛回答下这个问题, 希... 博文 来自: 启舰	
Python 基础 (一) : 入门必备知识	阅读数 14万+
Python 入门必备知识, 你都掌握了吗? 博文 来自: 程序之间	
一个小巧的rpc通信组件 (C++和python)	603
C++的网络库一般都很重, 这里基于zmq这个高性能的消息中间件用C++和python写了一个客户端/服务器互相异步... 博文 来自: 踏莎行的博客	<div><div>举报</div><div>78</div></div>
通信协议 (五) - RPC	
1.RPC (Remote Procedure Call远程过程调用) 协议1.1.RPC (Remote Procedure Call远程过程调用) 简介远程过... 博文 来自: weixin_42366378...	

阅读数 164

博文 来自: [大数据容器](#)



1 万+

博文 来自: [CSDN资讯](#)

数 6

博文 来自: [weixin_3](#)

106

博文 来自: zhang10_

293

博文 来自: Oscar Che

python json java mysql pcharm android linux json格式 c# mvc 接口开发 c#与mysql数据库 c#截取字符串倒序数第 c# 滚动抽奖 c# 变量栈大小 c#前端界面栏位有数据值 c# 文件保存到数据库 c# 打印窗口 c#项目开发实例 c# orm框架对比



zhzt245648124

TA的个人主页 >

原创28

粉丝7

获赞4

评论0

访问1万+

等级: 博客 3

周排名: 6万+

积分: 414

总排名: 18万+

勋章:  

关注

私信



便宜的主机

最新文章

java8新特性

Gradle入门——概念理解

Gradle入门二——创建task

mysql死锁

java面试总结--算法

分类专栏

 Gremlin3篇

 shiro

 java设计模式2篇

 java基础11篇

 orientDB3篇

展开

归档

2019年9月1篇

2019年7月2篇

2019年6月1篇

2019年5月11篇

2019年4月2篇

2019年3月2篇

2019年1月8篇

2018年9月3篇

展开

热门文章

tinkerpop / gremlin图遍历简单示例
阅读数 1930

CentOS7下超详细搭建完全分布式集群——hadoop2.7.7
阅读数 1257

RPC通信基本原理
阅读数 1002

orientDB导入CSV数据
阅读数 999

【1】tinkerpop / gremlin定义一个属性图
阅读数 807

1 域名永久购买

2 舆情监测平台

3 程序员外包平台

4 全网舆情监测

5 6元虚拟主机

6 菠菜推广

7 终生免费云主机

8 呼叫中心系统方

9 大数据培训机构

10 多用户商城系统

11 票务系统

12 大数据分析平台

13 hr人事管理系统

14 crm客户管理系

15 抠图兼职

16 麦克纳姆轮

17 程序员外包

18 会员卡管理系统

19 程序员月薪

20 什么叫ui设计

21 网站开发公司

22 人力资源咨询

23 第三方支付排名

24 云通信平台

 QQ客服

 kefu@csdn.net

 客服论坛

 400-660-0108

工作时间 8:30-22:00

关于我们 招聘 广告服务 网站地图

京ICP备19004658号 经营性网站备案信息

 公安备案号 11010502030143

©1999-2020 北京创新乐知网络技术有限公司

网络110报警服务

北京互联网违法和不良信息举报中心

中国互联网举报中心 家长监护 版权申诉

 1


















举报