RPC入门总结（一）RPC定义和原理

转载 Zenhobby 最后发布于2017-11-19 21:01:28 阅读数 47113 ☆ 收藏

转载：深入浅出 RPC - 浅出篇

转载：RPC框架与Dubbo完整使用

转载：深入浅出 RPC - 深入篇

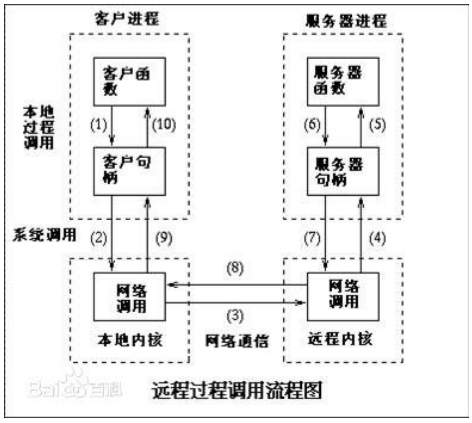
转载：远程调用服务(RPC)和消息队列(Message Queue)对比及其适用/不适用场合分析

一、RPC

1. RPC是什么

RPC（Remote Procedure Call Protocol）——**远程过程调用协议**，它是一种通过网络从远程计算机程序上请求服务，而不需要了解底层网络技术的协议。RPC协议假定某些传输协议的存在，如TCP或UDP，为通信程序之间携带信息数据。在OSI网络通信模型中，RPC跨越了**传输层**和**应用层**。RPC使得开发包括网络分布式多程序在内的应用程序更加容易。

RPC采用**客户机/服务器**模式。请求程序就是一个客户机，而服务提供程序就是一个服务器。首先，客户机调用进程发送一个有进程参数的调用信息到服务进程，然后等待应答信息。在服务器端，进程保持睡眠状态直到调用信息到达为止。当一个调用信息到达，服务器获得进程参数，计算结果，发送答复信息，然后等待下一个调用信息，最后，客户端调用进程接收答复信息，获得进程结果，然后调用执行继续进行。



2. 为什么要用RPC?

其实这是应用开发到一定的阶段的强烈需求驱动的。

1. 如果我们开发简单的单一应用，逻辑简单、用户不多、流量不大，那我们用不着；

2. 当我们的系统访问量增大、业务增多时，我们会发现一台单机运行此系统已经无法承受。此时，我们可以将业务拆分成几个**互不关联的应用**，分别部署在各自机器上，以划清逻辑并减小压力。此时，我们也可以不需要RPC，因为应用之间是互不关联的。
3. 当我们的业务越来越多、应用也越来越多时，自然的，我们会发现有些功能已经**不能简单划分开来或者划分不出来**。此时，可以将公共业务逻辑抽离出来，将之组成独立的服务Service应用 。而原有的、新增的应用都可以与那些独立的Service应用 交互，以此来完成完整的业务功能。所以此时，我们急需**一种高效的应用程序之间的通讯手段**来完成这种需求，所以你看，RPC大显身手的时候来了！
其实3描述的场景也是**服务化 、微服务** 和**分布式系统架构** 的基础场景。即RPC框架就是实现以上结构的有力方式。

二、RPC的原理和框架

Nelson 的论文中指出实现 RPC 的程序包括 5 个部分：

1. User

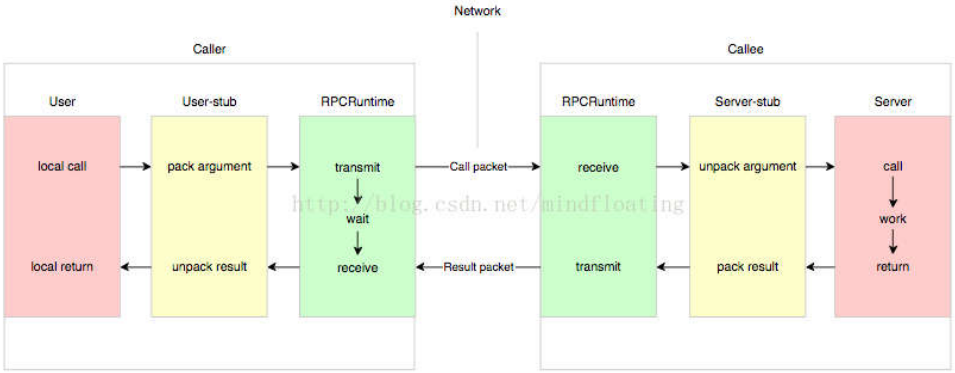
2. User-stub

3. RPCRuntime

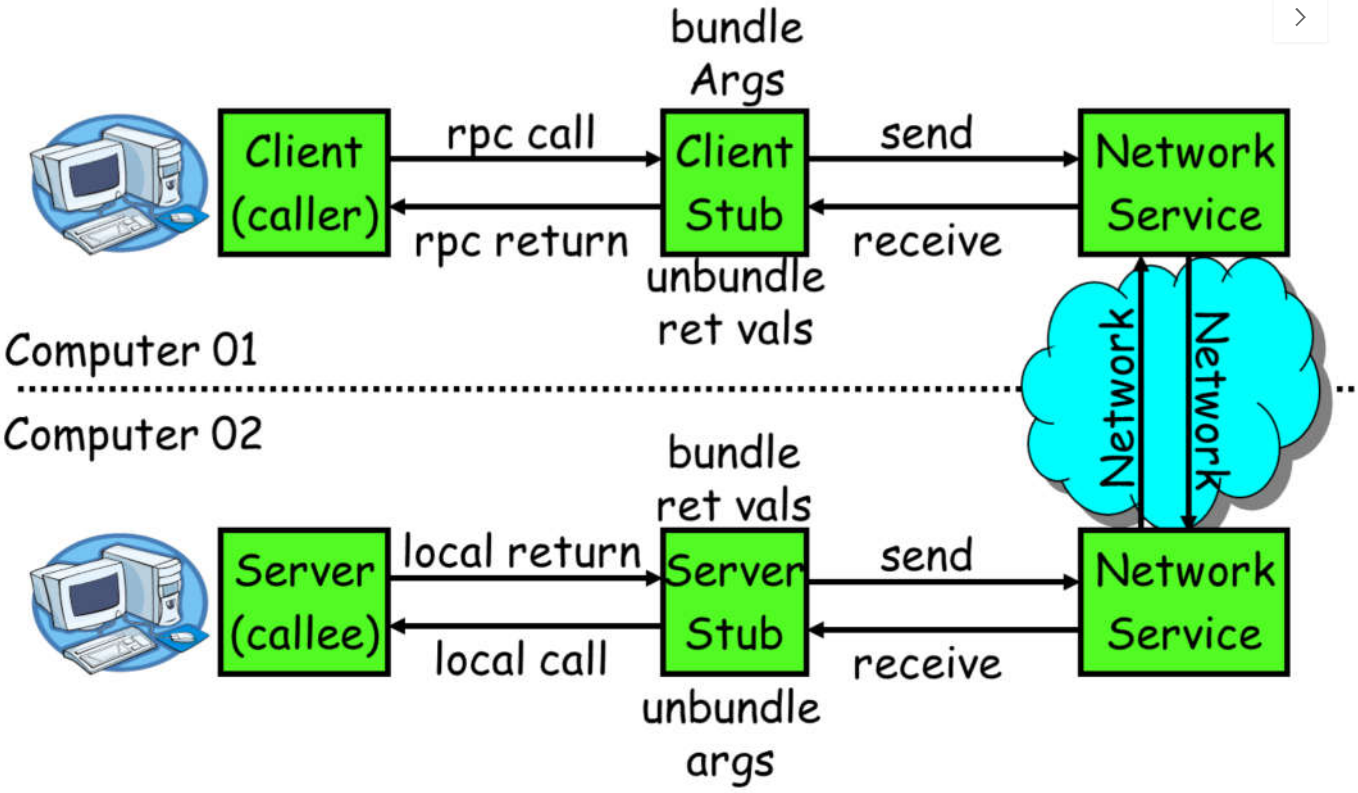
4. Server-stub

5. Server

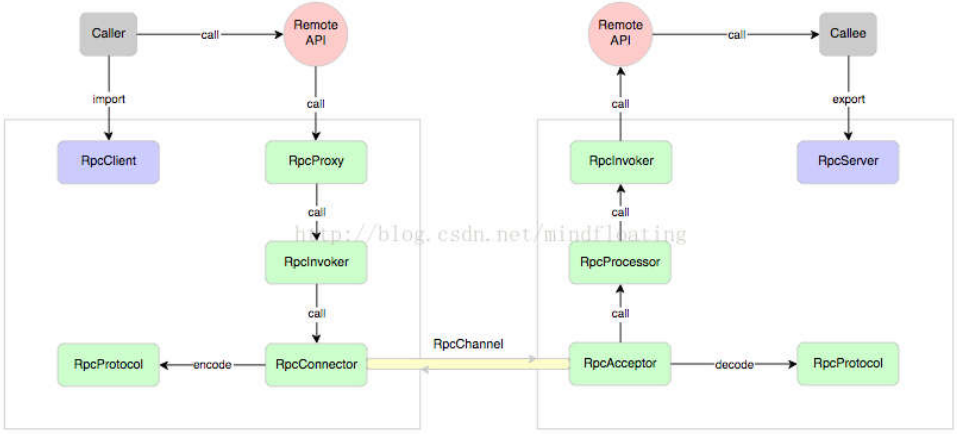
这 5 个部分的关系如下图所示



这里 user 就是 client 端，当 user 想发起一个远程调用时，它实际是通过本地调用**user-stub**。user-stub 负责将调用的接口、方法和参数通过约定的协议规范进行编码并通过本地 RPCRuntime 实例传输到远端的实例。远端 RPCRuntime 实例收到请求后交给 server-stub 进行解码后发起本地端调用，调用结果再返回给 user 端。



粗粒度的 RPC 实现概念结构，这里我们进一步细化它应该由哪些组件构成，如下图所示。



RPC 服务方通过 RpcServer 去导出（export）远程接口方法，而客户方通过 RpcClient 去引入（import）远程接口方法。客户方像调用本地方法一样去调用远程接口方法，RPC 框架提供接口的代理实现，实际的调用将委托给代理RpcProxy 。代理封装调用信息并将调用转交给RpcInvoker 去实际执行。在客户端的RpcInvoker 通过连接器RpcConnector 去维持与服务端的通道RpcChannel，并使用RpcProtocol 执行协议编码（encode）并将编码后的请求消息通过通道发送给服务方。

RPC 服务端接收器 RpcAcceptor 接收客户端的调用请求，同样使用RpcProtocol 执行协议解码（decode）。解码后的调用信息传递给RpcProcessor 去控制处理调用过程，最后再委托调用给RpcInvoker 去实际执行并返回调用结果。如下是各个部分的详细职责：

1. **RpcServer**
   负责导出（export）远程接口
2. **RpcClient**
   负责导入（import）远程接口的代理实现
3. **RpcProxy**
   远程接口的代理实现

4. **RpcInvoker**
客户方实现：负责编码调用信息和发送调用请求到服务方并等待调用结果返回
服务方实现：负责调用服务端接口的具体实现并返回调用结果
5. **RpcProtocol**
负责协议编/解码
6. **RpcConnector**
负责维持客户方和服务方的连接通道和发送数据到服务方
7. **RpcAcceptor**
负责接收客户方请求并返回请求结果
8. **RpcProcessor**
负责在服务方控制调用过程，包括管理调用线程池、超时时间等
9. **RpcChannel**
数据传输通道

三、Java中常用的RPC框架

目前常用的RPC框架如下：

1. **Thrift**：thrift是一个软件框架，用来进行可扩展且跨语言的服务的开发。它结合了功能强大的软件堆栈和代码生成引擎，以构建在 C++, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, C#, Cocoa, JavaScript, Node.js, Smalltalk, and OCaml 这些编程语言间无缝结合的、高效的服务。

2. **Dubbo**：Dubbo是一个分布式服务框架，以及SOA治理方案。其功能主要包括：高性能NIO通讯及多协议集成，服务动态寻址与路由，软负载均衡与容错，依赖分析与降级等。 Dubbo是阿里巴巴内部的SOA服务化治理方案的核心框架，Dubbo自2011年开源后，已被许多非阿里系公司使用。

3. **Spring Cloud**：Spring Cloud由众多子项目组成，如Spring Cloud Config、Spring Cloud Netflix、Spring Cloud Consul 等，提供了搭建分布式系统及微服务常用的工具，如配置管理、服务发现、断路器、智能路由、微代理、控制总线、一次性token、全局锁、选主、分布式会话和集群状态等，满足了构建微服务所需的所有解决方案。Spring Cloud基于Spring Boot，使得开发部署极其简单。

四、RPC和消息队列的差异

1. 功能差异

在架构上，RPC和Message的差异点是，Message有一个中间结点Message Queue，可以把消息存储。
消息的特点
1. Message Queue把请求的压力保存一下，逐渐释放出来，让处理者按照自己的节奏来处理。
2. Message Queue引入一下新的结点，系统的可靠性会受Message Queue结点的影响。
3. Message Queue是异步单向的消息。发送消息设计成是不需要等待消息处理的完成。
所以对于有同步返回需求，用Message Queue则变得麻烦了。
RPC的特点
同步调用，对于要等待返回结果/处理结果的场景，RPC是可以非常自然直觉的使用方式(RPC也可以是异步调用)。
由于等待结果，Consumer（Client）会有线程消耗。如果以异步RPC的方式使用，Consumer（Client）线程消耗可以去掉。但不能做到像消息一样暂存消息/请求，压力会直接传导到服务Provider。

2. 适用场合差异

1. 希望同步得到结果的场合，RPC合适。
2. 希望使用简单，则RPC；RPC操作基于接口，使用简单，使用方式模拟本地调用。异步的方式编程比较复杂。
3. 不希望发送端（RPC Consumer、Message Sender）受限于处理端（RPC Provider、Message Receiver）的速度时，使用Message Queue。
随着业务增长，有的处理端处理量会成为瓶颈，会进行同步调用到异步消息的改造。这样的改造实际上有调整业务的使用方式。比如原来一个操作页面提交后就下一个页面会看到处理结果；改造后异步消息后，下一个页面就会变成“操作已提交，完成后会得到通知”。

3. 不适用场合说明

1. RPC同步调用使用Message Queue来传输调用信息。 上面分析可以知道，这样的做法，发送端是在等待，同时占用一个中间点的资源。变得复杂了，但没有对等的收益。
2. 对于返回值是void的调用，可以这样做，因为实际上这个调用业务上往往不需要同步得到处理结果的，只要保证会处理即可。（RPC的方式可以保证调用返回即处理完成，使用消息方式后这一点不能保证了。）
3. 返回值是void的调用，使用消息，效果上是把消息的使用方式Wrap成了服务调用（服务调用使用方式成简单，基于业务接口）。

五、RPC框架的核心技术点

RPC框架实现的几个核心技术点：

（1）服务暴露：

远程提供者需要以某种形式提供**服务调用相关的信息**，包括但不限于**服务接口定义**、**数据结构**、或者中间态的**服务定义文件**。例如Facebook的Thrift的**IDL文件**，Web service的**WSDL文件**；服务的调用者需要通过一定的途径获取远程服务调用相关的信息。

目前，大部分跨语言平台 RPC 框架采用根据 IDL 定义通过 code generator 去生成 stub 代码，这种方式下实际导入的过程就是**通过代码生成器在编译期完成的**。代码生成的方式对跨语言平台 RPC 框架而言是必然的选择，而对于同一语言平台的 RPC 则可以通过**共享接口定义**来实现。这里的导入方式本质也是一种代码生成技术，只不过是在**运行时生成**，比静态编译期的代码生成看起来更简洁些。

java 中还有一种比较特殊的调用就是**多态**，也就是一个接口可能有多个实现，那么远程调用时到底调用哪个？这个本地调用的语义是通过 jvm 提供的引用多态性隐式实现的，那么RPC来说跨进程的调用就没法隐式实现了。如果前面DemoService 接口有 2 个实现，那么在导出接口时就需要**特殊标记不同的实现需要**，那么远程调用时也需要传递该标记才能调用到实现类，这样就解决了**多态调用**的语义问题。

（2）远程代理对象：

服务调用者用的服务实际是远程服务的本地代理。说白了就是通过**动态代理**来实现。

java 里至少提供了两种技术来提供动态代码生成，一种是 jdk 动态代理，另外一种是字节码生成。动态代理相比字节码生成使用起来更方便，但动态代理方式在性能上是要逊色于[illegible]字节码生成的，而字节码生成在代码可读性上要差很多。两者权衡起来，个人认为牺牲一些性能来获得代码可读性和可维护性显得更重要。

（3）通信：

RPC框架与具体的协议无关。RPC 可基于 **HTTP 或 TCP 协议**，Web Service 就是基于 HTTP 协议的 RPC，它具有良好的跨平台性，但其性能却不如基于 TCP 协议的 RPC。

1. TCP/HTTP：众所周知，TCP 是传输层协议，HTTP 是应用层协议，而传输层较应用层更加底层，在数据传输方面，越底层越快，因此，在一般情况下，TCP 一定比 HTTP 快。

2. 消息ID：RPC 的应用场景实质是一种**可靠的请求应答消息流**，和 HTTP 类似。因此选择长连接方式的 TCP 协议会更高效，与 HTTP 不同的是在协议层面我们**定义了每个消息的[illegible]ID**，因此可以更容易的复用连接。

3. IO方式：为了支持高并发，传统的阻塞式 IO 显然不太合适，因此我们需要**异步的 IO**，即 NIO。Java 提供了 NIO 的解决方案，Java 7 也提供了更优秀的 NIO.2 支持。

4. 多连接：既然使用长连接，那么第一个问题是到底 client 和 server 之间需要多少根连接？实际上单连接和多连接在使用上没有区别，对于**数据传输量较小**的应用类型，**单连接**[illegible]多。单连接和多连接最大的区别在于，每根连接都有自己私有的发送和接收缓冲区，因此**大数据量传输时分散在不同的连接缓冲区会得到更好的吞吐效率**。所以，如果你的数据传输量[illegible]让单连接的缓冲区一直处于饱和状态的话，那么使用多连接并不会产生任何明显的提升，反而会增加**连接管理**的开销。

5. 心跳：连接是由 client 端发起建立并维持。如果 client 和 server 之间是直连的，那么连接一般不会中断（当然物理链路故障除外）。如果 client 和 server 连接经过一些负载中转设备，有可能连接一段时间不活跃时会被这些中间设备中断。为了保持连接有必要定时为每个连接发送心跳数据以维持连接不中断。心跳消息是 RPC 框架库使用的内部消息，在前文协议头结构中也有一个专门的**心跳位**，就是用来标记心跳消息的，它对业务应用透明。

（4）序列化：

两方面会直接影响 RPC 的性能，一是传输方式，二是序列化。

1. **序列化方式**：毕竟是远程通信，需要将对象转化成二进制流进行传输。不同的RPC框架应用的场景不同，在序列化上也会采取不同的技术。 就序列化而言，Java 提供了默认的序列化方式，但在高并发的情况下，这种方式将会带来一些性能上的瓶颈，于是市面上出现了一系列优秀的序列化框架，比如：Protobuf、Kryo、Hessian、Jackson 等，它们可以取代 Java 默认的序列化，从而提供更高效的性能。

2. **编码内容**：出于效率考虑，编码的信息越少越好（传输数据少），编码的规则越简单越好（执行效率高）。如下是编码需要具备的信息：

```
-- 调用编码 --
1. 接口方法
   包括接口名、方法名
2. 方法参数
   包括参数类型、参数值
3. 调用属性
   包括调用属性信息，例如调用附件隐式参数、调用超时时间等

-- 返回编码 --
1. 返回结果
   接口方法中定义的返回值
2. 返回码
   异常返回码
3. 返回异常信息
   调用异常信息
```

除了以上这些必须的调用信息，我们可能还需要一些**元信息**以方便程序编解码以及未来可能的扩展。这样我们的编码消息里面就分成了两部分，一部分是**元信息**、另一部分是调用的**必要信息**。如果设计一种 RPC 协议消息的话，元信息我们把它放在协议消息头中，而必要信息放在协议消息体中。下面给出一种概念上的 RPC 协议消息设计格式：

```
-- 消息头 --
magic       : 协议魔数，为解码设计
header size: 协议头长度，为扩展设计
version     : 协议版本，为兼容设计
st          : 消息体序列化类型
hb          : 心跳消息标记，为长连接传输层心跳设计
ow          : 单向消息标记，
rp          : 响应消息标记，不置位默认是请求消息
status code: 响应消息状态码
reserved    : 为字节对齐保留
message id : 消息 id
body size   : 消息体长度

-- 消息体 --
采用序列化编码，常见有以下格式
xml    : 如 webservie soap
json   : 如 JSON-RPC
binary: 如 thrift; hession; kryo 等
```

以上就是RPC的核心技术点，包含内容繁多，下面本博就在学习的基础上以三种RPC框架为例介绍RPC的各项关键技术。

Zenhobby
发布了17 篇原创文章 · 获赞 174 · 访问量 56万+

他的留言板

想对作者说点什么

September_Lhi 1年前 查看回复 1

文中“RPC 的应用场景实质是一种可靠的请求应答消息流，和 HTTP 类似”，RPC不是基于HTTP/TCP协议吗，为什么还会拿来和HTTP做比较，菜鸟问答

z越秀越强 6个月前

请问一下,客户端编码后把请求发送给服务端，服务端解码后处理完返回数据给客户端是不是也得再编码发送，然后客户端再解码

我是一只小小小小小鸟 9个月前 赞

登录 查看 6 条热评

RPC入门总结（二）RMI的原理和使用 阅读数 4295
转载：架构设计：系统间通信（8）——通信管理与RMI 上篇转载：一、RMIRMI（Remote Method I... 博文 来自： Invoker's Tower

RPC服务和HTTP服务对比 阅读数 19万+
很长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... 博文 来自： 王云朋的专栏

RPC基本原理 阅读数 343
转载：https://blog.csdn.net/zkp_java/article/details/81879577RPC基本原理RPC(Remote Proced... 博文 来自： 你的博客

RPC入门总结（四）RPC IO基础：Netty原理和使用 阅读数 2024
转载：Java 编程思想（七） BIO/NIO/AIO的区别(Reactor和Proactor的区别)转载：Java 编程思想（... 博文 来自： Invoker's Tower

RPC入门总结（九）Dubbo框架实现细节 阅读数 7328
转载：转载：转载：转载：转载：转载：转载：一、Dubbo的注册中心服务注册中心是Dubbo中的重... 博文 来自： Invoker's Tower

终于明白阿里百度这样的大公司，为什么面试经常拿ThreadLocal考验求职者了 阅读数 22万+
点击上面↑「爱开发」关注我们每晚10点，捕获技术思考和创业资源洞察什么是ThreadLocalThreadLo... 博文 来自： 爱开发

60 个让程序员崩溃的瞬间，哈哈哈哈哈哈哈哈 阅读数 14万+
阅读本文大概需要 2.3333 分钟。前方高能，每一个程序员看完，你不笑死个人，你来找我，我自己看... 博文 来自： stormzhang的...

RPC 工作原理 阅读数 16
一、RPC1. RPC是什么RPC（Remote Procedure Call Protocol）——远程过程调用协议，它是一种通... 博文 来自： weixin_30872...

五年程序员记流水账式的自白。 阅读数 2万+
不知觉已中码龄已突破五年，一路走来从起初铁憨憨到现在的十九线程序员，一路成长，虽然不能成为... 博文 来自： Hello__Sunsh...

阿里面试官问我：如何设计秒杀系统？我的回答让他比起大拇指 阅读数 6万+
csdn把秒杀设计得最好的文章了。 博文 来自： 放丙

程序员必须掌握的核心算法有哪些？ 阅读数 41万+
由于我之前一直强调数据结构以及算法学习的重要性，所以就有一些读者经常问我，数据结构与算法应... 博文 来自： 帅地

Java入门学习路线目录索引（持续更新中） 阅读数 4万+

新增：使用IDEA搭建SpringBoot框架整合Mybatis、MySQL、Thymeleaf实现用户查询、注册、登录... 博文 来自： oneStar的博客

关于RPC协议的通俗理解 阅读数 2万+

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.c... 博文 来自： huangmr的专栏

了解这些后，再去决定要不要买mac 阅读数 2万+

当时买mac的初衷，只是想要个固态硬盘的笔记本，用来运行一些复杂的扑克软件。而看了当时所有的... 博文 来自： Diana5253的...

初识RPC 阅读数 44

作者：洪春涛链接：https://www.zhihu.com/question/25536695/answer/221638079来源：知乎著... 博文 来自： weixin_30322...

RPC入门总结（六）Thrift的介绍和用法 阅读数 3319

转载：由浅入深了解Thrift（一）——Thrift介绍与用法转载：Thrift源码分析（八）--总结加一个完整... 博文 来自： Invoker's Tower

RPC框架（一）RPC简介 阅读数 8466

一、概述二、RPC2.1、RPC定义2.2、RPC主要组成部分三、影响RPC框架性能的因素四、工业界的RP... 博文 来自： keep_trying的...

Linux(服务器编程):15---两种高效的事件处理模式（reactor模式、proactor模式） 阅读数 2万+

前言同步I/O模型通常用于实现Reactor模式 异步I/O模型则用于实现Proactor模式 最后我们会使用同步... 博文 来自： 江南、董少

从入门到精通，Java学习路线导航（附学习资源） 阅读数 11万+

引言最近也有很多人来向我"请教"，他们大都是一些刚入门的新手，还不了解这个行业，也不知道从何... 博文 来自： java_sha的博客

RPC基础入门：原理介绍和简单示例 阅读数 1830

一、RPC1. RPC是什么2. 为什么要用RPC？ 二、RPC的原理和框架三、Java中常用的RPC框架四、RPC... 博文 来自： weixin_38327...

【RPC入门】RPC概念、原理及开发 阅读数 115

RPC基础概念RPC（Remote Procedure Call）—远程过程调用，它是一种通过网络从远程计算机程序... 博文 来自： signal926的专栏

从零开始实现RPC框架 - RPC原理及实现 阅读数 3万+

RPC概述RPC(RemoteProcedureCall)即远程过程调用，允许一台计算机调用另一台计算机上的程序得... 博文 来自： Ricky

开源一个功能完整的SpringBoot项目框架 阅读数 7万+

福利来了，给大家带来一个福利。最近想了解一下有关Spring Boot的开源项目，看了很多开源的框架... 博文

C语言魔塔游戏 阅读数 6万+

很早就很想写这个，今天终于写完了。游戏截图：编译环境: VS2017游戏需要一些图片，如果有想要的... 博文 来自： 张宜强的博客

HTML5适合的情人节礼物有纪念日期功能 阅读数 1万+

前言利用HTML5，css，js实现爱心树 以及 纪念日期的功能 网页有播放音乐功能 以及打字倾诉感情的... 博文 来自： weixin_43853...

作为程序员的我，大学四年一直自学，全靠这些实用工具和学习网站！ 阅读数 2万+

我本人因为高中沉迷于爱情，导致学业荒废，后来高考，毫无疑问进入了一所普普通通的大学，实在惭... 博文 来自： 编码之外的技...

Python新型冠状病毒疫情数据自动爬取+统计+发送报告+数据屏幕（三）发送篇 阅读数 8323

今天介绍的项目是使用 Itchat 发送统计报告项目功能设计：定时爬取疫情数据存入Mysql进行数据分析... 博文 来自： Fantasy!

什么是RPC协议？RPC协议与HTTP协议的区别 阅读数 2111

什么是RPC协议？RPC是一种远程过程调用的协议，使用这种协议向另一台计算机上的程序请求服务，... 博文 来自： BraveHeart

什么是 RPC 阅读数 723

RPC：远程调用。通过RPC框架，使得我们可以像调用本地方法一样地调用远程机器上的方法：1、本... 博文 来自： 坚持，让梦想...

RPC的原理总结 阅读数 18

一.RPC的引入早期单机时代，一台电脑上运行多个进程，大家各干各的，老死不相往来。假如A进程需... 博文 来自： weixin_34335...

RPC意义和用法 阅读数 938

“本地过程调用，就好比你现在在家里，你要想洗碗，那你直接把碗放进洗碗机，打开洗碗机开关就可... 博文 来自： 好记性不如烂...

走进高并发（二）Java并行程序基础 阅读数 1万+

一、进程和线程在操作系统这门课程中，对进程的定义是这样的：进程是计算机中的程序关于某数据集... 博文 来自： 脚踏实地，慢...

一文详尽系列之模型评估指标 阅读数 5988

点击上方“Datawhale”，选择“星标”公众号第一时间获取价值内容在机器学习领域通常会根据实际... 博文 来自： Datawhale

RPC 基本原理和实现方式 阅读数 674

近几年随着微服务化项目的崛起，逐渐成为许多公司中大型分布式系统架构的主流方式，而今天所说的 ... 博文 来自：云深不知处

给Python初学者的一些编程技巧 阅读数 8921

这篇文章主要介绍了给Python初学者的一些编程技巧,皆是基于基础的一些编程习惯建议,需要的朋友可... 博文 来自：python核心编程

大学四年自学走来，这些私藏的实用工具/学习网站我贡献出来了 阅读数 46万+

大学四年，看课本是不可能一直看课本的了，对于学习，特别是自学，善于搜索网上的一些资源来辅助... 博文 来自：帅地

Http和RPC区别 阅读数 1万+

RPC（即RemoteProcedureCall，远程过程调用）和HTTP（HyperTextTransferProtocol，超文本... 博文 来自：蓬莱道人的博客

RPC快速入门（GO） 阅读数 72

RPC是什么？在分布式计算，远程过程调用（英语：Remote Procedure Call，缩写为 RPC）是一个计... 博文 来自：haya的博客

Mybatis学习（2）史上最全的 自定义mybatis 阅读数 1万+

这篇博客介绍了一下手写 mybatis 的全部过程，并且有完整的代码实现。 博文 来自：扬帆向海的博客

RPC——一切架构的基础 阅读数 325

RPC——一切架构的基础现代架构设计离不开分布式系统，而远程过程调用（Remote Process Call，... 博文 来自：zhangbijun12...

python json java mysql pycharm android linux json格式 c#影院售票系统有哪些 c#鼠标相对窗体的坐标 c# 如何快速的求和 c# 界面设计 c#窗口隐藏 c# 动态注入il 测试c#程序的软件 加入队列c# c# 模型验证取消 c# 小数点后保留4位

Zenhobby

TA的个人主页 >

原创	粉丝	获赞	评论	访问
17	372	174	50	56万+

等级: 博客 5 周排名: 9万+

积分: 5300 总排名: 8779

勋章:

关注 私信

最新文章

我在ThoughtWorks学软开（一）敏捷之于开发如同蜜糖，甜到发腻齁到忧伤

Node.js从入门到实战（八）Solr的层级

Node.js从入门到实战（七）Solr查询规则总结

DevOps入门（三）自动化构建工具Gradle

DevOps入门（二）包管理工具yarn与npm对比

分类专栏

我在ThoughtWorks... 1篇

MFC 40篇

归档

热门文章

最新评论

C++中引用（&）的用法和...
small21：mark,经常看经常忘

五大算法思想：分治、动态规划、贪心...
Chloe__：这篇总结简直是宝藏，感谢~

Java 数据库连接池的实现
qq_22038259：[reply]huahangwanghao[/reply] hia,hia

RPC入门总结（一）RPC定义和原理
qq_41643060：请问一下,客户端编码后把请求发送给服务端，服务端解码后处理完返回数据给？...

MySql从一窍不通到入门（五）S...
Vibugs：虽然是转载的，但是文章质量很高。谢谢作者