# RPC简介

## 介绍

套接字是客户机-服务器网络的基本组成部分。它们为程序提供了一种相对简单的机制，可以在远程或本地机器上建立与另一个程序的连接，并来回发送消息（我们甚至可以使用读写系统调用）。然而，这个接口迫使我们使用读/写（输入/输出）接口来设计分布式应用程序，这不是我们通常设计非分布式应用程序的方式。在设计集中应用程序时，过程调用通常是标准接口模型。如果我们想让分布式计算看起来像集中式计算，基于输入/输出的通信并不是实现这一点的方法。

## 定义

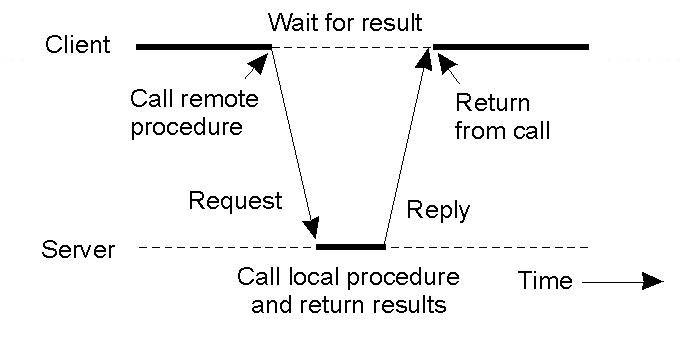
RPC的全称是Remote Procedure Call，它是一种进程间通信方式。允许像调用本地服务一样通过网络调用远程服务，它的具体实现方式可以不同。

* RMI（远程方法调用）JAVA自带的远程方法调用工具，不过有一定的局限性，毕竟是JAVA语言最开始时的设计，后来很多框架的原理都基于RMI
* WebService：SOAP
* Spring的HTTP Invoker
* Thrift：Thrift是Apache的一个项目(http://thrift.apache.org)，前身是Facebook开发的一个RPC框架，采用thrift作为IDL (Interface description language)。
* Netty：Netty框架不局限于RPC，更多的是作为一种网络协议的实现框架，比如HTTP，由于RPC需要高效的网络通信，就可能选择以Netty作为基础。
* Brpc：是一个基于protobuf接口的RPC框架，在百度内部称为“baidu-rpc”，它囊括了百度内部所有RPC协议，并支持多种第三方协议，从目前的性能测试数据来看，brpc的性能领跑于其他同类RPC产品。
* Dubbo：是Alibaba开发的一个RPC框架，远程接口基于Java Interface, 依托于Spring框架。
* HSF

## Wiki百科的定义

RPC是一个请求-响应协议。RPC请求由客户机创建，发送请求消息到远程的服务器，使用提供的参数来执行指定的进程。远程服务器向客户机发送响应，应用程序继续其进程。当服务器正在处理调用时，除非客户端向服务器发送异步请求（如xmlhttprequest），否则客户端将被阻止（在服务器完成处理后再继续执行）。在各种实现中有许多变化和差别，导致了各种不同（不兼容）的RPC协议产生。

远程过程调用和本地调用之间的一个重要区别是，由于不可预知的网络问题，远程调用可能会失败。此外，调用者通常必须在不知道远程过程是否被实际调用的情况下处理这些故障。*幂等进程（如果多次调用就没有额外效果）很容易处理，但仍然存在足够的困难，调用远程过程的代码通常仅限于精心编写的低级子系统。*



# RPC框架原理

RPC概念术语是在1980s由Bruce Jay Nelson提出，他对RPC进行了总结：

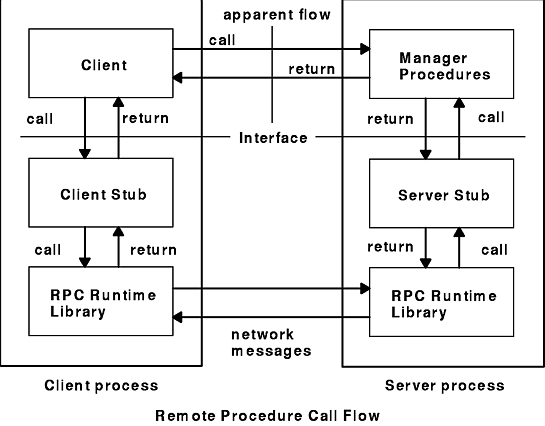
简单：

高效：

通用：

RPC框架的目标就是让远程过程调用更加简单、透明。RPC框架负责屏蔽底层的传输方式（TCP、UDP）、序列化方式（xml、json、二进制）和通信细节，开发者只需要知道在什么地方提供了什么样的远程服务接口即可。

## RPC模型



1. The client calls a local procedure, called the client stub. To the client process, this appears to be the actual procedure, because it is a regular local procedure. It just does something different since the real procedure is on the server. The client stub packages the parameters to the remote procedure (this may involve converting them to a standard format) and builds one or more network messages. The packaging of arguments into a network message is called marshaling and requires serializing all the data elements into a flat array-of-bytes format.
2. Network messages are sent by the client stub to the remote system (via a system call to the local kernel using *sockets* interfaces).
3. Network messages are transferred by the kernel to the remote system via some protocol (either connectionless or connection-oriented).
4. A server stub, sometimes called the **skeleton**, receives the messages on the server. It unmarshals the arguments from the messages and, if necessary, converts them from a standard network format into a machine-specific form.
5. The server stub calls the server function (which, to the client, is the remote procedure), passing it the arguments that it received from the client.
6. When the server function is finished, it returns to the server stub with its return values.
7. The server stub converts the return values, if necessary, and marshals them into one or more network messages to send to the client stub.
8. Messages get sent back across the network to the client stub.
9. The client stub reads the messages from the local kernel.
10. The client stub then returns the results to the client function, converting them from the network representation to a local one if necessary.

## RPC框架实现的核心技术

服务提供形式

远程代理

通信

序列化

Java RMI

简单RPC框架的实现