 参考：<http://blog.csdn.net/shan9liang/article/details/8995023>

远程通信的几种选择（RPC，Webservice，RMI，JMS的区别）

* RPC（Remote Procedure Call Protocol）-远程过程调用

RPC使用C/S方式，采用http协议,发送请求到服务器，等待服务器返回结果。这个请求包括一个参数集和一个文本集，通常形成“classname.methodname”形式。优点是跨语言跨平台，C端、S端有更大的独立性，缺点是不支持对象，无法在编译器检查错误，只能在运行期检查。

* Web Service

Web Service提供的服务是基于web容器的，底层使用http协议，类似一个远程的服务提供者，比如天气预报服务，对各地客户端提供天气预报，是一种请求应答的机制，是跨系统跨平台的。就是通过一个servlet（server applet，即java编写的server端应用程序），提供服务出去。

首先客户端从服务器的到WebService的WSDL，同时在客户端声称一个代理类(Proxy Class)， 这个代理类负责与WebService服务器进行Request 和Response。 当一个数据（XML格式的）被封装成SOAP（简单对象访问协议，基于xml的数据交换格式）格式的数据流发送到服务器端的时候，就会生成一个进程对象并且把接收到这个Request的SOAP包进行解析，然后对事物进行处理，处理结束以后再对这个计算结果进行SOAP包装，然后把这个包作为一个Response发送给客户端的代理类(Proxy Class)，同样地，这个代理类也对这个SOAP包进行解析处理，继而进行后续操作。这就是WebService的一个运行过程。

Web Service大体上分为5个层次:

 1. Http传输信道

 2. XML的数据格式

 3. SOAP封装格式

 4. WSDL的描述方式

 5. UDDI  UDDI是一种目录服务，企业可以使用它对Webservices进行注册和搜索

* RMI （Remote Method Invocation）

RMI 采用stubs 和 skeletons 来进行远程对象(remote object)的通讯。stub 充当远程对象的客户端代理，有着和远程对象相同的远程接口，远程对象的调用实际是通过调用该对象的客户端代理对象stub来完成的，通过该机制RMI就好比它是本地工作，采用tcp/ip协议，客户端直接调用服务端上的一些方法。优点是强类型，编译期可检查错误，缺点是只能基于**[Java](http://lib.csdn.net/base/javase" \o "Java SE知识库" \t "_blank)**语言，客户机与服务器紧耦合。

* JMS（[**Java**](http://lib.csdn.net/base/java)Messaging Service）

JMS是Java的消息服务，JMS的客户端之间可以通过JMS服务进行异步的消息传输。JMS支持两种消息模型：Point-to-Point（P2P）和Publish/Subscribe（Pub/Sub），即点对点和发布订阅模型。

**几者的区别与联系**

**1、RPC与RMI**

（1）RPC（remote procedure call） 跨语言，而 RMI（remote method invocation）只支持Java。

（2）RMI 调用远程对象方法，允许方法返回 Java 对象以及基本数据类型，而RPC 不支持对象的概念，传送到 RPC 服务的消息由外部数据表示 (External Data Representation, XDR) 语言表示，这种语言抽象了字节序类和数据类型结构之间的差异。只有由 XDR 定义的数据类型才能被传递， 可以说 RMI 是面向对象方式的 Java RPC 。

（3）在方法调用上，RMI中，远程接口是每个远程方法都具有方法签名。如果一个方法在服务器上执行，但是没有相匹配的签名被添加到这个远程接口上，那么这个新方法就不能被RMI客户方所调用。

在RPC中，当一个请求到达RPC服务器时，这个请求就包含了一个参数集和一个文本值，通常形成“classname.methodname”的形式。这就向RPC服务器表明，被请求的方法在为 “classname”的类中，名叫“methodname”。然后RPC服务器就去搜索与之相匹配的类和方法，并把它作为那种方法参数类型的输入。这里的参数类型是与RPC请求中的类型是匹配的。一旦匹配成功，这个方法就被调用了，其结果被编码后返回客户方。

**2、JMS和RMI**

采用JMS 服务，对象是在物理上被异步从网络的某个JVM 上直接移动到另一个JVM 上（是消息通知机制）

而RMI 对象是绑定在本地JVM 中，只有函数参数和返回值是通过网络传送的（是请求应答机制）。

RMI一般都是同步的，也就是说，当client调用Server的一个方法的时候，需要等到对方的返回，才能继续执行client端，这个过程与调用本地方法感觉上是一样的，这也是RMI的一个特点。

JMS 一般只是一个点发出一个Message到Message Server,发出之后一般不会关心谁用了这个message。

所以，一般RMI的应用是紧耦合，JMS的应用相对来说是松散耦合应用。

**3、Webservice与RMI**

RMI是在tcp协议上传递可序列化的java对象，只能用在java虚拟机上，绑定语言，客户端和服务端都必须是java

webservice没有这个限制，webservice是在http协议上传递xml文本文件，与语言和平台无关

**4、Webservice与JMS**

Webservice专注于远程服务调用，jms专注于信息交换。

大多数情况下Webservice是两系统间的直接交互（Consumer <--> Producer），而大多数情况下jms是三方系统交互（Consumer <- Broker -> Producer）（类似于kafka）。当然，JMS也可以实现request-response模式的通信，只要Consumer或Producer其中一方兼任broker即可。

JMS可以做到异步调用（kafka即为异步调用），完全隔离了客户端和服务提供者，能够抵御流量洪峰； WebService服务通常为同步调用（请求-响应模式），需要有复杂的对象转换，相比SOAP（简单对象封装），现在JSON，rest都是很好的http[**架构**](http://lib.csdn.net/base/architecture)方案；（举一个例子，电子商务的分布式系统中，有支付系统和业务系统，支付系统负责用户付款，在用户在银行付款后需要通知各个业务系统，那么这个时候，既可以用同步也可以用异步，使用异步的好处就能抵御网站暂时的流量高峰，或者能应对慢消费者。）

JMS是java平台上的消息规范。一般jms消息不是一个xml，而是一个java对象，很明显，jms没考虑异构系统（连个并非完全一样的系统），说白了，JMS就没考虑非java的东西。但是好在现在大多数的jms provider（就是JMS的各种实现产品）都解决了异构问题。相比WebService的跨平台各有千秋吧。

参考：<http://blog.csdn.net/a363722188/article/details/41147675>

RPC和RMI的区别

远程对象方法调用并不是新概念，远程过程调用 (RPC-remote procedurecall) 已经使用很多年了。远程过程调用被设计为在应用程序间通信的平台中立的方式，它不理会操作系统之间（异构系统）以及语言之间的差异。即 RPC 支持多种语言，而 RMI(Remote Method Invocation，远程对象方法调用)只支持 Java 写的应用程序。

  另外 RMI 调用远程对象方法，允许方法返回 [**Java**](http://lib.csdn.net/base/javase) 对象以及基本数据类型。而 RPC 不支持对象的概念，传送到 RPC 服务的消息由外部数据表示 (ExternalData Representation,XDR) 语言表示，这种语言抽象了字节序类和数据类型结构之间的差异。只有由 XDR 定义的数据类型才能被传递， RPC 不允许传递对象。可以说 RMI 是面向对象方式的 JavaRPC 。

RMI和RPC之间最主要的区别在于方法是如何被调用的。在RMI中，远程接口使每个远程方法都具有方法签名。如果一个方法在服务器上执行，但是没有相匹配的签名被添加到这个远程接口上，那么这个新方法就不能被RMI客户方所调用。

在RPC中，当一个请求到达RPC服务器时，这个请求就包含了一个参数集和一个文本值，通常形成“classname.methodname”的形式。这就向RPC服务器表明，被请求的方法在为“classname”的类中，名叫“methodname”。然后RPC服务器就去搜索与之相匹配的类和方法，并把它作为那种方法参数类型的输入。这里的参数类型是与RPC请求中的类型是匹配的。一旦匹配成功，这个方法就被调用了，其结果被编码后返回客户方。