RPC框架

RPC(Remote Procedure Call，远程过程调用)一般用来实现部署在不同集群上的系统之间的方法调用，使得程序能够像访问本地系统资源一样，通过网络传输去访问远端系统资源。RPC实现的架构原理是类似的，如下图所示：



1. RPC Client，客户端调用代码实现，负责发起RPC调用，为调用方用户提供使用API
2. Serialization/Deserialization，负责对RPC调用通过网络传输的内容进行序列化与反序列化，不同的RPC框架有不同的实现机制。主要分为文本和二进制两大类，文本类别的序列化主要有XML与JSON两种格式，二进制类别的序列化常见的有Java原生序列化、Hessian、Protobuf、Thrift、Avro、Kryo等，不同的序列化方式在可读性、码流大小、支持的数据类型及性能等方面有较大差异。
3. Sub Proxy，RPC代理存在客户端，因为要实现客户端对RPC框架的透明调用，客户端不需要去管理消息格式及网络传输协议等，这一切工作在客户端交给RPC框架中的代理层去处理。
4. Message Protocol，完整的Client-Server的交互需要携带两端都能识别、共同约定的消息格式。RPC的消息管理层对网络传输信息进行编码和解码操作。例如RMI框架使用的消息协议为JRMP，Hessian的传输协议基于HTTP，Thrift Protocol等
5. Transport，传输协议层负责管理RPC框架所使用的网络协议、网络IO模型等
6. Selector/Processor，存在于RPC服务端，负责执行RPC接口实现，包括RPC接口的注册、判断客户端的请求权限、控制接口实现类的执行等各项工作
7. Server code，服务端服务业务逻辑具体的实现
8. IDL，接口定义语言，不是必须的，但是需要跨语言的RPC框架会有IDL部分的存在。