Rpc原理：

1. 服务消费方（client）调用以本地调用方式调用服务；
2. client stub接收到调用后负责将方法、参数等封装(序列化)成能够进行网络传输的消息体
3. client stub找到服务地址，并将消息发送到服务端；
4. server stub收到消息后进行解码；
5. server stub根据解码结果调用本地的服务
6. 本地服务执行并将结果返回给server stub
7. server stub将返回结果打包成消息并发送至消费方
8. client stub接收到消息，并进行解码（反序列化）；
9. 服务消费方得到最终结果。

公司内使用百度的ral。

Ral配置：

本地ini，配置连接数超时时长。

服务配置:local方式，IP+PORT，用于配置本机

Webfoot：配置一个name，读取noah的配置，得到调用被调用服务的集群，负载均衡策略，数据大包策略，重连策略等。

调用：

同步接口：ral,ral\_multi//所有请求一次串行发送

异步接口：ral\_create\_loop,失败返回false，成功返回一个object可配置回调函数

协议：http/nshead

数据结构：

1）接口名称

2）方法名

3）参数类型以及相应的参数值；

4）超时时间

5）requestID(nio为了回掉而设置的参数)

序列化：

序列化就是将数据结构或对象转换成二进制串的过程

（由于网络协议是基于二进制的，内存中的参数的值要序列化成二进制的形式，也就是序列化）

Rpc：

1. 基于tcp（复杂）nshead+mcpack
2. 基于HTTP（效率低一些）

Ral调用：

Ral/ral\_multi串行请求并行堵塞/ral\_reate\_loop()//epoll

根据参数请求不同的服务组，一个服务组下挂载这多个节点，一个节点可以有多个机器构成

可以配置几种load\_balance策略，一般为random

非阻塞模式：

$loop = ral\_create\_loop(); //新建一个句柄池

if (false === $loop) { return false; }

$header = array(

'pathinfo' => "index.php",

'querystring'=> "user=robin&id=1",

'useragent' => "Mozilla/5.0",

'referer' => "http://www.baidu.com",

'content-type' => "application/octet-stream",

'cookie' => array('uid'=>'1234'),

'Accept-Encoding' => "gzip",

);

$ret = ral\_event\_add("demo1", $loop, "demoService", "post", array('demo'=>1),

rand(), $header); //发送请求新增句柄到句柄池

if (false === $ret) { return false; }

while(1) {

$ret = ral\_loop($loop);

var\_dump($ret);

if (true === $ret) {

break;

}

}

//非阻塞只支持http

**Netty //不了解**

**Tcp层的rpc效率高**