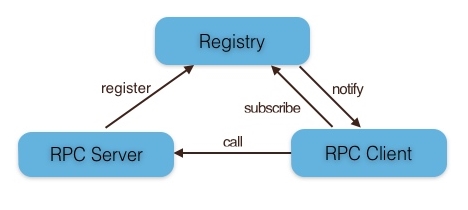
**Motan RPC**

官方github地址：https://github.com/weibocom/motan

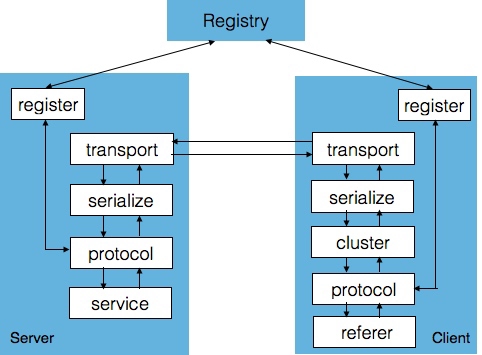
简单的同步时调用实例，参照github上的例子，很容易跑起来，这里就不在赘述。

**Motan 的架构及模块设计**

　　架构设计，分为服务提供方(RPC Server)、服务调用方(RPC Client)、注册中心(Registry)三个角色，Server 向 Registry 注册声明所提供的服务；Client 向 Registry 订阅指定服务，与 Registry 返回的服务列表的 Server 建立连接，进行 RPC 服务调用。Client 通过 Registry 感知 Server 的状态变更。三者的交互关系如下图：



　　服务模块化设计方便灵活扩展，Motan 主要包括 register、transport、serialize、protocol、cluster 等，各个模块都是支持通过SPI 进行扩展，各个模块的交互图如下：



**register 模块**

　　用来和注册中心进行交互，包括注册服务、订阅服务、服务变更通知、服务心跳发送等功能；Server 端会在系统初始化时通过 register 模块注册服务，Client 端在系统初始化时会通过 register 模块订阅到具体提供服务的 Server 列表，当 Server 列表发生变更时也由 register 模块通知 Client。

**protocol 模块**

　　用来进行 RPC 服务的描述和 RPC 服务的配置管理，这一层还可以添加不同功能的 filter 用来完成统计、并发限制等功能。

　　serialize 模块将 RPC 请求中的参数、结果等对象进行序列化与反序列化，即进行对象与字节流的互相转换；默认使用对 Java 更友好的 hessian2 进行序列化。

　　transport 模块用来进行远程通信，默认使用 Netty NIO 的 TCP 长链接方式。

**cluster 模块**

　　Client 端使用的模块，cluster 是一组可用的 Server 在逻辑上的封装，包含若干可以提供 RPC 服务的 Server，实际请求时会根据不同的高可用与负载均衡策略选择一个可用的 Server 发起远程调用。

　　在进行 RPC 请求时，Client 通过代理机制调用 cluster 模块，cluster 根据配置的 HA 和 LoadBalance 选出一个可用的 Server，通过 serialize 模块把RPC 请求转换为字节流，然后通过 transport 模块发送到 Server 端。

　　服务配置化增强了 Motan 的易用性，Motan 框架中将功能模块抽象为四个可配置的元素，分别为：

　　protocol：服务通信协议。服务提供方与消费方进行远程调用的协议，默认为 Motan 协议，使用 hessian2 进行序列化，Netty 作为 Endpoint 以及使用Motan 自定义的协议编码方式。 registry：注册中心。服务提供方将服务信息(包含 IP、端口、服务策略等信息)注册到注册中心，服务消费方通过注册中心发现服务。当服务发生变更，注册中心负责通知各个消费方。 service：服务提供方提供的服务。使用方将核心业务抽取出来，作为独立的服务。通过暴露服务并将服务注册至注册中心，从而使调用方调用。 referer：服务消费方对服务的引用，即服务调用方。

Motan 推荐使用 Spring 配置 RPC 服务，目前 Motan 扩展了 6 个自定义 Spring XML 标签

motan:protocol motan:registry motan:basicService

　　motan:service motan:basicReferer motan:referer

　　高可用方面是 Motan 的一大特点，支持多种服务治理和高可用机制，包括：

灵活多样的集群负载均衡策略，支持

ActiveWeight/Random/RoundRobin/LocalFirst/Consistent 等 6 种策略，并支持自定义扩展； 自动集成 Failover、Failfast 容错策略，实现故障节点自动摘除，自动探测恢复，有效进行服务故障隔离，远离服务卡死及雪崩； 连接池自定义控制，根据业务场景灵活配置； 支持多机房间调用流量压缩、动态流量调整，实现真正的跨 IDC 的高可用。

　　基于高并发、高负载场景的优化，具备在高压力场景下的高可用能力。

**集群调用示例**

在集群环境下使用Motan需要依赖外部服务发现组件，目前支持consul或zookeeper。

这里只谈使用ZooKeeper作为注册中心

#### ZooKeeper安装与启动([官方文档](https://zookeeper.apache.org/doc/trunk/zookeeperStarted.html))

下载url ：<http://apache.org/dist/zookeeper/zookeeper-3.4.9/>

选择 **zookeeper-3.4.9.tar.gz**

修改zoo\_sample.cfg 文件名(D:\soft\zookeeper-3.4.8\conf) 为 zoo.cfg主要修改一下日志位置，具体配置文件如下：

# The number of milliseconds of each tick

tickTime=2000

# The number of ticks that the initial

# synchronization phase can take

initLimit=10

# The number of ticks that can pass between

# sending a request and getting an acknowledgement

syncLimit=5

# the directory where the snapshot is stored.

# do not use /tmp for storage, /tmp here is just

# example sakes.

dataDir=D:\\mine\\project\\zk\\data1

dataLogDir=D:\\mine\\project\\zk\\log1

# the port at which the clients will connect

clientPort=2181

# the maximum number of client connections.

# increase this if you need to handle more clients

#maxClientCnxns=60

#

# Be sure to read the maintenance section of the

# administrator guide before turning on autopurge.

#

# http://zookeeper.apache.org/doc/current/zookeeperAdmin.html#sc\_maintenance

#

# The number of snapshots to retain in dataDir

#autopurge.snapRetainCount=3

# Purge task interval in hours

# Set to "0" to disable auto purge feature

#autopurge.purgeInterval=1

Win 下直接运行 zkServer.cmd 即可；

Linux 环境，到bin目录下， 执行 ./zkServer.sh

带着一下问题来探寻：

1. 服务端用什么来服务来运行并怎么运行？
2. Client 端与服务端怎么通信？
3. Client 端提交过去的数据做过哪些处理？
4. Zookeeper 是如何注册服务？
5. Zookeeper 是如何管理以及维护服务？
6. Client 端是如何远程调用服务？