Dubbo底层实现原理和机制

Dubbo底层 用到Socket

1.通信原理
计算机于外界的信息交换成为通信， 基本的通信方法有两种 并行通信和串行通信
并行通信：Parallel communication{并行是指多比特数据同时通过并行线进行传送，这样数据传送速度大大提高，但并行传送的线路长度受到限制，因为长度增加，干扰就会增加，数据也就容易出错。}

```
穿行通信:serial communication 串行通信作为计算机通信方式之一，主要起到主机与外设以及主

1.一组数据 (通常是字节) 的各位数据被同时传送的通信方法称为并行通信，并行通信依靠I/O接
```

·多，只适用于近距离.（距数公尺 的通信

2.一组信息的各位数据被逐位顺序传送的通信方式称为串行通信.串行通信可以通过串行接口来实现，串行通信传输速度慢，但是传输线少。适宜 长距离通信。

串行通信按信息传送方向分为三种
1.单工 ——————只能一个方向传输数据
A ---------->B
2.半双工
信息能双向传输，但不能同时双向传输
3.全双工
能双向传输并且可以同时双向传输

```
        2.Socket 是一种应用接口，TCP/IP 是网络传输协议，虽然接口相同，但是不同的协议有不同
        Socket可以支持不同的传输层协议(TCP或者UDP， 当使用TCP协议进行连接时候，该Socket链
        、
    所以。Sockets(TCP)通信是全双工的方式
```

Dubbo远程同步调用原理分析

从Dubbo开源文档上了解到一个调用过程如下图

http://code.alibabatech.com/wiki/display/dubbo/User+Guide#UserGuide-APIReference

另外文档里有说明：Dubbo缺省协议采用单一长连接和NIO异步通讯，适合于小数据量大并发的服务调用，以及服务消费者机器数远大于服务提供者机器数的情况。

Dubbo缺省协议，使用基于mina1.1.7+hessian3.2.1的tbremoting交互。

· 连接个数：单连接

· 连接方式：长连接

· 传输协议：TCP

· 传输方式：NIO异步传输

· 序列化：Hessian二进制序列化

· 适用范围：传入传出参数数据包较小（建议小于100K），消费者比提供者个数多，单一消费者无法压满提供者，尽量不要用dubbo协议传输大文件或超大字符串。

· 适用场景：常规远程服务方法调用

通常，一个典型的同步远程调用应该是这样的：

1， 客户端线程调用远程接口，向服务端发送请求，同时当前线程应该处于“暂停“状态，即线程不能向后执行了，必需要拿到服务端给自己的结果后才能向后执行

2， 服务端接到客户端请求后，处理请求，将结果给客户端

3， 客户端收到结果，然后当前线程继续往后执行

Dubbo里使用到了Socket（采用apache mina框架做底层调用）来建立长连接，发送、接收数据，底层使用apache mina框架的IoSession进行发送消息。

查看Dubbo文档及源代码可知，Dubbo底层使用Socket发送消息的形式进行数据传递，结合了mina框架，使用IoSession.write()方法，这个方法调用后对于整个远程调用(从发出请求到接收到结果)来说是一个异步的，即对于当前线程来说，将请求发送出来，线程就可以往后执行了，至于服务端的结果，是服务端处理完成后，再以消息的形式发送给客户端的。于是这里出现了2个问题：

· 当前线程怎么让它“暂停”，等结果回来后，再向后执行?

· 正如前面所说，Socket通信是一个全双工的方式，如果有多个线程同时进行远程方法调用，这时建立在client server之间的socket连接上会有很多双方发送的消息传递，前后顺序也可能是乱七八糟的，server处理完结果后，将结果消息发送给client，client收到很多消息，怎么知道哪个消息结果是原先哪个线程调用的?

分析源代码，基本原理如下：

1. client一个线程调用远程接口，生成一个唯一的ID（比如一段随机字符串，UUID等），Dubbo是使用AtomicLong从0开始累计数字的

2. 将打包的方法调用信息（如调用的接口名称，方法名称，参数值列表等），和处理结果的回调对象callback，全部封装在一起，组成一个对象object

3. 向专门存放调用信息的全局ConcurrentHashMap里面put(ID, object)

4. 将ID和打包的方法调用信息封装成一对象connRequest，使用IoSession.write(connRequest)异步发送出去

5. 当前线程再使用callback的get()方法试图获取远程返回的结果，在get()内部，则使用synchronized获取回调对象callback的锁， 再先检测是否已经获取到结果，如果没有，然后调用callback的wait()方法，释放callback上的锁，让当前线程处于等待状态。

6. 服务端接收到请求并处理后，将结果（此结果中包含了前面的ID，即回传）发送给客户端，客户端socket连接上专门监听消息的线程收到消息，分析结果，取到ID，再从前面的ConcurrentHashMap里面get(ID)，从而找到callback，将方法调用结果设置到callback对象里。

7. 监听线程接着使用synchronized获取回调对象callback的锁（因为前面调用过wait()，那个线程已释放callback的锁了），再notifyAll()，唤醒前面处于等待状态的线程继续执行（callback的get()方法继续执行就能拿到调用结果了），至此，整个过程结束。

这里还需要画一个大图来描述，后面再补了

需要注意的是，这里的callback对象是每次调用产生一个新的，不能共享，否则会有问题；另外ID必需至少保证在一个Socket连接里面是唯一的。

现在，前面两个问题已经有答案了，

· 当前线程怎么让它“暂停”，等结果回来后，再向后执行?

```
答：先生成一个对象obj，在一个全局map里put(ID,obj)存放起来，再用synchronized获取obj锁，再
```

· 正如前面所说，Socket通信是一个全双工的方式，如果有多个线程同时进行远程方法调用，这时建立在client server之间的socket连接上会有很多双方发送的消息传递，前后顺序也可能是乱七八糟的，server处理完结果后，将结果消息发送给client，client收到很多消息，怎么知道哪个消息结果是原先哪个线程调用的?

```
答：使用一个ID，让其唯一，然后传递给服务端，再服务端又回传回来，这样就知道结果是原先哪个线程
```