1. 给我简单介绍下你的项目？

这个项目就是一个简易的RPC框架，可以利用此框架进行远程调用。在分布式架构下，每一个部分都需要和其他部分连接并能进行远程调用，这是实现分布式的前提。此RPC项目的原理和分布式常用的dubbo相似。

本项目采用了Netty框架进行了网络通信，并采用zookeeper作为服务注册发现的中间件，用来注册和订阅相关地址。并实现了心跳检测机制保证了RPC框架的高可用。使用了kryo和proto进行了序列化，并使用了注解简化了用户使用框架的流程。

1. 你的项目的流程给我介绍一下
2. 首先需要创建实体类，客户端请求、服务端响应以及用于心跳检测的message实体类
3. 然后先创建了客户端和服务端的初始化类，先完成了一些常用的初始化过程，如客户端初始化类中除了构造方法，还有建立连接、发送请求、获得Channel方法，服务端初始化类中以start()方法作为初始化方法
4. 然后创建所需的心跳检测机制处理类IdleStateHandler、编解码处理类、客户端、服务端处理类，并且需要创建未完成请求的map存放类，还有zookeeper相关服务发现、注册类，
5. 有些类需要使用单例模式保证唯一，所以需要创建单例模式，并且如压缩、负载均衡加载策略、序列化等有多种选择的策略需要灵活调用，所以创建了一个自适应扩展类，用于自动选择相应接口实现类
6. 创建注解，用于方便显示需要调用的函数，及被调用的函数等信息
7. 最后创建客户端服务端测试类测试
8. 你的项目的架构是怎样的？

分为了传输层、协议层、集群层、入口层

传输层主要是用TCP传输客户端和服务端之间的信息

协议层主要是自定义了RPC协议以及序列化、编解码等

集群层是用于提供服务注册和服务发现，以及相关的其他服务

入口层是实现RPC服务的，主要是使用了动态代理屏蔽了内部细节，使我们可以直接调用远程方法就可以使用RPC框架通信，动态代理自动设置代理类在代理类中完成调用过程

1. 你的项目具有哪些特别的点？为什么要做这个项目？

首先是使用了动态代理屏蔽了内部的细节处理，保证调用者可以直接调用就能使用rpc框架进行中间的通信。

其次是对需要调用的服务使用注解标注了需要的版本以及现有的版本，保证调用的服务与需要的服务版本一致

然后就是心跳检测机制，还有使用了自制的单例模式对一些如存放未完成请求的map创建单例，使用了自适应扩展类用于自动选择相应接口实现类，从而使压缩，负载均衡，序列化等部分形成插件模式，可插拔，需要哪种修改相关配置即可使用

1. 你知道dubbo么？那你的项目和dubbo有什么区别？X

Dubbo是阿里开源的RPC框架，是RPC的实现方式。

我的RPC项目相当于一个极简的Dubbo，dubbo在每个模块都有许多可插拔的选择，每个模块都设置成了插件，方便选择，比如注册中心就可以选择zookeeper或者redis，序列化方式也可以有多种选择

1. 短链接，长链接：

短链接有什么问题？如何实现的长连接？ X

短连接在每次请求结束后会马上断开连接，在下次发起请求时重新进行三次握手发起连接，这会导致频繁握手产生不必要的损耗，如果发起200w次请求会连接200w此，性能极差，因此需要使用长连接。

长连接实现：http中在首部字段设置Connection:keep-alive 和 Keep-Alive:timeout = 60，就可以一直保持长连接状态，直到60s内没有任何请求才会断开，如果没有超时限制会一直保持连接不断开

1. 基于TCP 还是http，为什么用TCP实现

基于TCP，http虽说是现成的协议，但是http协议往往需要存储传输大量的数据，并有许多冗余的不需要的首部长度，浪费传输性能，因此可以在TCP连接基础上创建一个私有RPC协议，自定义需要的信息，保证了传输效率最大化

1. 同步阻塞调用性能瓶颈：有什么瓶颈？ X 你怎么解决的，怎么实现异步调用的？

同步调用时准备数据和拷贝数据都会被阻塞，此时其他线程无法执行必须等待此线程完成后才能使用。性能瓶颈：不太清楚，好像是每次发起请求都会发起一个线程，请求：线程=1:1，Java中线程太多十分影响性能

第一种解决就是使用伪异步，虽然仍然是阻塞IO，但是收到请求后会放在一个线程池中。

第二种是IO多路复用，只需要一个Select阻塞轮询多个线程，准备就绪后就可以直接返回处理

第三种就是使用异步调用，使用信号标记，当数据准备好后返回一个信号通知然后从内核态阻塞拷贝数据到用户态

1. 为什么需要心跳检测,客户端发送请求服务端没回应会造成什么(X)？心跳检测机制你是怎么实现的？(X)

当通信中空闲时并不清楚通信的状态是否正常，因此需要在空闲时定期心跳检测保证通信的正常，保证高可用。

如果没有回应会尝试再次发送，最后会抛出通信异常。

心跳检测通过加入心跳检测处理器，在处理器中设置读写空闲的间隔，空闲时间每隔多长时间发送一个心跳检测，如果连续M个周期没有收到响应则表示通信出现异常

1. 什么叫做序列化？Java原生序列化问题(-)？为什么用JSON，怎么实现的JSON序列化，和JAVA序列化速度的差异？(x)

序列化就是把对象转换成二进制流，用来在网络中传输。http协议可以使用JSON，通用各种语言，但是占用容积比较大，在传输大数据时性能会有损耗

1. 为什么用JSON，应该还有其他更快的吧？(-)

使用了kryo和protos

1. 你还知道哪些序列化的方式？

Kryo、protos、hessian

1. 你是怎么基于动态代理进行的请求处理？用的哪一种？为什么使用这种?(X)

首先创建了SpringBeanPostProcessor实现BeanPostProcessor类，重写了postProcessBeforeInitialization和postProcessAfterInitialization，在处理Bean前获取相关信息为服务器发布服务到zookeeper，在处理Bean后利用动态代理创建客户端代理，把调用的方法的相关参数等信息放在客户端代理中，然后代替原有的Bean

其次客户端代理也使用了动态代理，使用的JDK动态代理，实现了InvokationHandler接口，然后重写了invoke方法，和getProxy方法，通过getProxy中返回Proxy.newProxyInstance(clazz.getClassLoader(), new Class<?>[]{clazz}, this);创建代理类。并在invoke中调用相关方法及前后增加的逻辑。最终通过代理类调用方法时会直接调用invoke方法

1. 解决进程通信问题这个我不太懂，你的项目哪里遇到进程通信问题了？ (X)

就比如说客户端发出请求，此时服务端收到后需要先处理请求然后发出服务端响应给客户端

1. 你是怎么解决进程通信问题的？(X)

这里先使用了ReentrantLock和Condition条件锁，在主线程获取数据前，先要等待结果；在获取时候结果时，先上锁，防⽌多个线程同时获取结果；如果还没有结果，则进⼊Condition的等待室进⾏等待（condition.await），并在finally块 中释放锁；当获取到消息时，通过signal⽅法唤醒在condition等待室中等待的线程

后来使用了CompleteFuture作为改进，使用了类似的内部机制，在传输完成后使用complete方法传输表明传输完成，然后通过get()方法可以获得传输完成的管道继续执行。

1. 你还知道哪些解决进程通信问题的方式？

管道/匿名管道

消息队列/缓存，放入消息后可立刻返回

共享/虚拟内存，内核态和用户态共用一块内存，节省了拷贝时间

信号量

Socket

1. TCP/IP 拆包粘包能简单介绍下么？你怎么解决的呢？有没有更好的解决方式？(X Header Body)

拆包粘包就是在接收消息时由于某些原因导致接受的一个包附带了另一个包的一部分数据，或者一个包被拆分成了两个包，导致收到的消息不完整

原因是1、应用程序write吸入字节大小大于发送缓冲区大小；2、MSS大小的TCP分段导致超出长度的必须分段3、MTU长度的IP分片导致超出长度的必须分片

解决：1、消息定长，不够补空格；2、利用换行符在包尾分割，如FTP协议；3、发送的消息分为消息头和消息体，消息头标注消息的总长度

1. 你是怎么基于BeanPostProcessor 机制和 ApplicationListener 机制实现客户端的自启 动与基于注解的服务调用
2. 注册中心你是怎么实现的？用的什么？

使用了zookeeper及Curator客户端实现的

1. 为什么用Zk？(X)

Zk是当前主流的注册中心中间件，Zookeeper的数据模型很简单，由一系列被称为ZNode的数据节点组成，与传统的磁盘文件系统不同的是，zk将全量数据存储在内存中，可谓是高性能，而且支持集群，可谓高可用，另外支持事件监听。这些特点决定了zk特别适合作为注册中心(数据发布/订阅)。

1. 你的服务结点结构是怎样的？

/my-rpc+服务器名字+服务器IP地址+端口，如/my-rpc/cyt/127.0.0.1:9998

1. 那么当有新的服务上线或者旧服务下线的时候，你怎么保证得到最新结果？(X)

通过注解保证，使用了注解标注了当前的组合版本以及需要的组和版本，调用前会根据此判断，如果版本一致就说明调用的是一致的，保证了得到的是最新结果，如果不一致说明版本不同，调用就会失败。

并且zk有强一致性，采用消息总线通知机制保证注册中心数据的一致性

1. 客户端怎么发现服务？

创建发现服务类，重写lookupService发现服务方法，根据服务器名在zk中查询该名字对应子节点中的ip和端口号，如果服务器地址很多，选择一种负载均衡策略来选择服务器地址。然后根据此向服务器发出请求

1. 怎么动态监听链接？(X)
2. 给我讲一讲Watcher机制 ( X )
3. 给我讲一讲你项目中你认为最难的点以及你的解决方式?(-)

似乎难点还挺多，主要是不熟练逻辑和api的使用。最后测试启动的时候，总会报错，并且报错也没有明确说明是哪里的问题，然后就得一步步debug进Spring体系里慢慢找，debug了半天，后来从Spring里的单例模式中创建完单例后跳到了错误产生的代码部分，然后一步步发现了都是小问题，都是某个地方需要扫描包时包名写错了，但是这种地方还很多。

1. 你的项目有哪些改进地方？你想改进哪里？

有一个已经改进的地方，原来使用的重入锁+Condition条件锁进行的线程通信，但是这样做比较麻烦，并且后来发现有一个现成的CompleteFuture类，可以complete直接完成，并只有在完成后get()才会返回结果。

然后是相比于Dubbo，提供的序列化方式太少，我只提供了两种，其中一种还报错运行不了。

我感觉可以使用Spring整合，会更方便，直接使用配置文件对相关序列化、压缩方式进行选择，现在需要我自己创建一个扩展类，然后自己创建一个文件标注使用的哪个配置。

1. 你这个项目的创新点有哪些？
2. 做这个项目的意义是什么？

主要是为了提升自己的技术，CRUD项目做完了感觉就那样，但是看到一些开源项目时没有头绪，这个RPC项目在开发时对开源的定义越来越清晰了，提高了自己的技术广度同时也提高了对技术的热情，全程使用的git，并有项目文档可以使用。

1. 项目的性能如何？哪些地方可以继续优化从而提高性能？