**问题：**在分布式环境下如何监控服务提供者提供的服务调用情况（QPS）？

**背景：**为了应对日益复杂的业务场景，通常将整个系统业务拆分成不同的子系统或者子应用，子应用之间通过消息队列进行通信，当然也可以通过访问同一个数据存储系统来构成一个关联的完整系统。在横向拆分上，将可以复用的业务子应用拆分出来，独立部署为分布式服务，新增业务只需要调用这些分布式服务就可以完成内部逻辑。

**图一 分布式系统架构**

**总体思路：**首先在ZooKeeper集群中注册服务调用者和服务提供者以及服务监控服务，当服务调用者调用了需要监控的服务时，通过动态代理动态生成服务实现者的代理对象，其中代理对象中注入一个监听服务接口引用，通过回调机制起到对调用者调用某服务接口的拦截作用。而监听服务接口的实现就是通过拦截方式无侵入式的给服务接口的调用情况进行统计和分析。



**图二 服务框架架构**

**图三 ZooKeeper注册中心**

**举例说明：**

**步骤一**

(1)服务提供者启动时,向/Demo/SayHelloService/providers目录下注册自己的URL地址；  
(2)服务消费者启动时,订阅/Demo/SayHelloService/providers目录下的提供者URL地址，并向/Demo/SayHelloService/consumers目录下注册自己的URL地址；  
(3)监控中心启动时，订阅/Demo/SayHelloService目录下的所有提供者和消费者URL地址。

**步骤二**

暴露的服务接口

Interface SayHello {

public void sayHello();

}

服务提供者实现类：

Class SayHelloImpl implements SayHello {

public void sayHello() {

System.out.println(“hello world”);

}

}

**步骤三**

服务消费者调用服务时，动态生成的动态代理对象(字节码)

Class ProxySayHelloImpl implements SayHello {

private Filter monitor;

private SayHello target;

public void sayHello() {

monitor.invoke // 拦截回调

}

}

**步骤四**

控制权转到监听实现类时

public class MonitorFilter implements Filter {

public Result invoke(Method method, Object[] args, Object target) {

1.  获取调用者的上下文信息

2.  并发计数加一

3.  记录起始时间戳

4. 执行实际的调用

5. 采集调用信息比如计算调用耗时、获取并发数、获取实际服务名称和服务方法

6. 并发计数减一

7. 将收集到的信息通过任务交给监控中心处理

}

}

**步骤五**

展示监控服务调用情况

