**研究报告**

**姓名： 孟 佶**

**学号： 2017104158**

2018年10月

**一、背景**

微服务是一个软件架构形式，在这个架构中，复杂的应用程序是由多个小而独立的服务组成。这些服务较小、高度解耦且专注于完成一个小任务，使得用模块化方法建设系统更加容易，能够很好地应对复杂业务系统的多变需求。**目前，非常多企业的技术架构都开始向微服务方向发展**，例如，猫眼电影作为著名的电影在线票务，其主系统集成了基本电影选座的所有功能。随着业务的发展，在2014年《变形金刚4》上映期间，由于流量过大，耦合业务之间相互影响，导致系统资源耗尽，整体不可用数个小时。这时猫眼电影才发现All-In-One风格的主系统无法应对业务的发展。此后，猫眼电影对其主系统进行改造，进行了订单、支付、券、活动等系统的全面拆分服务化，搭建起一个分布式的服务架构。目前，IBM、谷歌、阿里巴巴和华为等都提出了各自的微服务架构。根据红帽公司（Red Hat）的一项调查，87%的受访者都在使用或者考虑微服务架构技术。**除了企业内部采用服务架构来搭建系统，互联网上的很多资源提供商也将自己的资源封装成服务，外部用户则通过“交易”获得相应的服务**。例如，Programmable Web网站提供了互联网最大的Web服务库，到目前为止该网站已经提供了约2万个Web服务，数量也还在快速增长过程中，实际上现有的互联网上服务的实际数量可能远远大于这个数字。但是这些服务往往有不同提供商，由不同组织管理，开放的互联网环境下往往没有一个可信的中心节点可以对这些服务进行统一的调度。然而，目前流行的微服务架构往往是应用于企业内部，无法直接应用于在缺乏信任、无中心节点的互联网环境下搭建分布式服务架构。因此，**本项目提出一种基于区块链的运行在开放的、缺乏信任和无中心节点的互联网环境下的服务交易平台**。

目前互联网上一些常见的交易平台往往还是需要中心节点的，例如，国内知名的数据交易平台“数据堂”。直接将数据堂模式应用到服务交易上可能有两个问题：1）**集中式交易架构。**交易依赖于用户对中心节点的信任，一旦用户丧失这种信任，交易就可能无法进行；2）**缺乏对服务交易持续性的关注。**能否保证服务提供者持续提供高质量的服务？用户能否从同一个服务提供者不断获取更新更好的服务？

针对第一个问题，**区块链作为一种通过去中心化的方式，集体维护的可靠的分布式数据库的技术方案，是实现互联网环境中交易平台比较好的选择**。具体来说，区块链是一种按照时间顺序将数据区块以顺序相连的方式组合成的一种链式数据结构, 并以密码学方式保证的不可篡改和不可伪造的分布式账本。区块链技术是利用块链式数据结构来验证与存储数据、利用分布式节点共识算法来生成和更新数据、利用密码学的方式保证数据传输和访问的安全，利用由自动化脚本代码组成的智能合约来编程和操作数据的一种全新的分布式基础架构与计算范式。目前在区块链基础之上已经有了一些交易平台，可以解决在无中心节点的环境下交易双方的信任问题。

然而，服务是一种特殊的商品，用户往往希望持续地从服务提供者那里获得符合需求的更新更好的服务，传统的“一次性购买”的交易模式可能无法很好地满足这样的需求。因此当前的基于区块链的交易平台还是没能很好地解决前面提到的第二个问题。区块链采用带有时间戳的链式区块结构存储数据，从而为数据增加了时间维度, 具有极强的可验证性和可追溯性。智能合约是运行在区块链上的代码，由于区块链自身的特性，使得运行在区块链上的智能合约拥有突出的特点：首先，区块链的不可篡改性使得用户不再担心合约内容被恶意修改，提高了合约的信任性；其次，区块链的高可靠性避免了条件满足而未执行合约的情况，合约双方能更好地履行合约义务、更好地享有合约权利；最后，区块链的去中心性使得合约在每一个节点上皆有备份，完备的记录使事后追踪变得可能、方便。我们认为，**用户和服务提供者可以将他们之间的约定写入智能合约，由智能合约来实时监测服务质量，当服务质量不能达到约定的要求时，智能合约就可以按照约定处理。**这样，就可以在缺乏信任和无中心节点的环境中促使服务提供者能按照约定持续提供高质量的服务。

综上所述，本项目提出基于区块链和智能合约实现一个分布式的服务交易平台，使得开放环境中互联网上数量庞大的资源可以得到共享。

**二、代码分析**

一）环境配置（SCSP/artifacts）

## 1、生成公私钥和证书

Fabric中有两种类型的公私钥和证书，一种是给节点之前通讯安全而准备的TLS证书，另一种是用户登录和权限控制的用户证书。这些证书本来应该是由CA来颁发，但是我们这里是测试环境，并没有启用CA节点，所以Fabric帮我们提供了一个工具：cryptogen。

1.1 编译生成cryptogen（SCSP/artifacts/bin/）

直接执行命令make cryptogen即可。

1.2 配置cryptogen.yaml文件（SCSP/artifacts/channel/）

该文件提供了一个Orderer Org和两个Peer Org的配置。其中Name和Domain就是关于这个组织的名字和域名，这主要是用于生成证书的时候，证书内会包含该信息。而Template Count=2是说要生成2套公私钥和证书，一套是peer0.org2的，还有一套是peer1.org2的。最后Users. Count=1是说每个Template下面会有几个普通User（Admin不包含在这个计数中），这里配置了1，也就是说只需要一个普通用户[User1@org2.example.com](mailto:User1@org2.example.com)，可以根据实际需要调整这个配置文件，增删Org、Users等。

2、生成创世区块和Channel配置区块

2.1 编译生成configtxgen（SCSP/artifacts/bin/）

## 与1.1类似，我们可以通过make configtxgen命令生成configtxgen

### 2.2 配置configtx.yaml文件（SCSP/artifacts/channel/）

该文件里面配置了由2个Org参与的Orderer共识配置TwoOrgsOrdererGenesis，以及由2个Org参与的Channel配置：TwoOrgsChannel。Orderer可以设置共识的算法是Solo还是Kafka，以及共识时区块大小，超时时间等。而Peer节点的配置包含了MSP的配置，锚节点的配置。

### 2.3 生成创世区块

配置修改好后，就用configtxgen生成创世区块。

### 2.4生成Channel配置区块

最终，我们在channel文件夹中，应该是能够看到：

channel/   
├── softwarechannel.tx   
├── genesis.block   
├── crypto-config

├── ordererOrganzitions  
└── peerOrganzations

## 3、配置Fabric环境的docker-compose文件

### 3.1 配置Orderer（SCSP/artifacts/base.yaml）

其中ORDERER\_GENERAL\_GENESISFILE=/etc/hyperledger/configtx/genesis.block，这个创世区块就是我们之前创建的创世区块，这里就是Host到Docker的映射：../channel/genesis.block:/etc/hyperledger/configtx/genesis.block

另外的配置主要是TL，Log等，最后暴露出服务端口7050。

### 3.2 配置Peer（SCSP/artifacts/base.yaml和SCSP/artifacts/docker-compose.yaml）

在Peer的配置中，主要是给Peer分配好各种服务的地址，以及TLS和MSP信息。

### 3.3 配置CLI（SCSP/artifacts/docker-compose.yaml）

CLI在整个Fabric网络中扮演客户端的角色，我们在开发测试的时候可以用CLI来代替SDK，执行各种SDK能执行的操作。CLI会和Peer相连，把指令发送给对应的Peer执行。

## 二）运行

### 1、启动Fabric网络

我们通过执行脚本（SCSP/runApp.sh）即可启动fabric网络。在该文件中，首先通过以下命令将正在启动的网络关闭：

docker-compose -f ./artifacts/docker-compose.yaml down

接着，通过以下命令将正在运行的docker容器删除：

CONTAINER\_IDS=$(docker ps -aq)

docker rm -f $CONTAINER\_IDS

同样，通过以下命令将生成的镜像删除：

DOCKER\_IMAGE\_IDS=$(docker images | grep "dev\|none\|test-vp\|peer[0-9]-" | awk '{print $3}')

docker rmi -f $DOCKER\_IMAGE\_IDS

最后，重新启动fabric网络：

docker-compose -f ./artifacts/docker-compose.yaml up -d

### 2、创建Channel（SCSP/scripts/install.sh）

curl -s -X POST \

http://localhost:4000/channels \

-H "authorization: Bearer $ADMIN\_GFE\_TOKEN" \

-H "content-type: application/json" \

-d '{

"channelName":"softwarechannel",

"channelConfigPath":"../artifacts/channel/softwarechannel.tx"

}'

### 3、将Peer加入到Channel中（SCSP/scripts/install.sh）

### curl -s -X POST \

### http://localhost:4000/channels/softwarechannel/peers \

### -H "authorization: Bearer $ADMIN\_DEKE\_TOKEN" \

### -H "content-type: application/json" \

### -d '{

### "peers": ["peer0.fabric.deke.com","peer1.fabric.deke.com"]

### }'

### 4、安装Chaincode（SCSP/scripts/install.sh）

### curl -s -X POST \

### http://localhost:4000/chaincodes \

### -H "authorization: Bearer $ADMIN\_GFE\_TOKEN" \

### -H "content-type: application/json" \

### -d "{

### \"peers\": [\"peer0.fabric.gfe.com\",\"peer1.fabric.gfe.com\"],

### \"chaincodeName\":\"$CC\_NAME\",

### \"chaincodePath\":\"$CC\_SRC\_PATH\",

### \"chaincodeType\": \"$LANGUAGE\",

### \"chaincodeVersion\":\"v0\"

### }"

5、Chaincode实例化（SCSP/scripts/install.sh）

curl -s -X POST \

http://localhost:4000/channels/softwarechannel/chaincodes \

-H "authorization: Bearer $ADMIN\_GFE\_TOKEN" \

-H "content-type: application/json" \

-d "{

\"peers\": [\"peer0.fabric.gfe.com\"],

\"chaincodeName\":\"currency\",

\"chaincodeVersion\":\"v0\",

\"chaincodeType\": \"$LANGUAGE\",

\"args\":[]

}"

6、用户和服务提供方进行注册（SCSP/scripts/regist.sh）

7、服务提供方加入候选队列（SCSP/scripts/listener.sh）

8、服务协商过程（SCSP/scripts/request.sh）

三、运行结果

以下界面是由经过服务协商过程选择出来的四个服务（ticket-airline、hotel、rent-car以及scenic-spot）组成的旅游服务界面

