(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 108989389 B
(45)授权公告日 2020.09.11

(21)申请号 201810596428.5

(22)申请日 2018.06.11

(65)同一申请的已公布的文献号
申请公布号 CN 108989389 A

(43)申请公布日 2018.12.11

(73)专利权人 北京航空航天大学
地址 100191 北京市海淀区学院路37号

(72)发明人 胡凯 余维 罗戡 丁毅

(51)Int.Cl.
H04L 29/08(2006.01)
H04L 29/06(2006.01)
G06Q 40/04(2012.01)

(56)对比文件
CN 107566153 A,2018.01.09
CN 107102847 A,2017.08.29
CN 106878393 A,2017.06.20
CN 106686132 A,2017.05.17
CN 107579931 A,2018.01.12
US 2017278186 A1,2017.09.28

审查员 胡冰舟

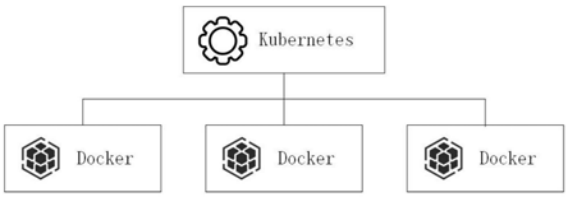
权利要求书1页 说明书4页 附图1页

(54)发明名称

一种建立智能合约微服务化的方法

(57)摘要

本发明提供了一种建立智能合约微服务化的方法,包括:步骤一,设计智能合约微服务的基础容器:(1)使用docker实现单一合约微服务的执行容器;(2)上层使用kubernetes对合约进行管理整体管理;步骤二,设计智能合约微服务间的通信方式,使用远程过程调用的方法或者基于RESTFUL接口规范的方法;步骤三,设计整体智能合约微服务的运行方式:1)用户的请求被微服务的特定网关所获取并进行解析处理;(2)网关检查单一合约以及组合合约重新封装为特定合约服务消息,设定服务访问序列;(3)请求消息完成单一合约的业务逻辑,并对各容器数据更新;(4)最终处理结果返回给网关,随后交付给用户;(5)请求整个处理流程通过区块链接口存入到区块链中。



CN 108989389 B

1.一种建立智能合约微服务化的方法,其特征在于包括:

步骤一,设计智能合约微服务的基础容器,包括:单一合约微服务的执行容器使用docker实现,隔离合约间的资源,使合约成为独立的模块,在上层使用kubernetes对合约进行整体管理,包括合约服务的注册,以及合约容器的管理;

步骤二,设计智能合约微服务间的通信方式,单一合约运行在相互独立的容器之中,合约之间可进行组合使用,完成更为复杂的合约逻辑,在各个合约容器之间建立通信,所述通信方式采用RESTFUL接口方法进行通信;

步骤三,设计整体智能合约微服务的运行方式,智能合约微服务的实现分为部署和运行,每个合约微服务均被部署在单个的基础容器之中,提供给每个合约模版独立的运行、开发、验证的环境,智能合约微服务的运行方式对应的工作流包括:

步骤31,用户的请求将被微服务的特定网关所获取,网关进行对应的解析处理;

步骤32,网关检查所请求的产生的单一合约的合法性以及组合合约的可组合性,在检查完成后将请求信息重新封装为特定的合约服务消息,设定服务访问序列;

步骤33,请求消息按照用户所需逻辑进入各个基础容器之中完成单一合约的业务逻辑,并对各个基础容器数据进行更新;

步骤34,最终处理结果被返回给网关,随后交付给用户;

步骤35,整个请求的处理流程通过区块链接口存入到区块链中;

所述智能合约微服务将应用程序构建为一系列松散耦合的服务,每个合约微服务均被部署在单个基础容器之中,提供给每个合约模版独立的运行、开发、验证的环境,单个合约的所述微服务在所述基础容器中执行,所述基础容器中的单个合约的所述微服务具有功能单一性,可组合性,高融入性,高可扩展性以及高可验证性,所述功能单一性即所述合约微服务所针对的业务是单一的;所述可组合性为将简单合约微服务能通过轻量级通信方式相互组合形成复杂合约,完成特定业务逻辑;所述高融入性即所述合约微服务可将服务暴露,通过轻量级通信服务访问,被封装成API从而被用户使用,并安插在自己的业务逻辑中;所述高可扩展性即所述合约微服务在开发完成后,只需将自身的服务暴露给其他微服务和用户便可被使用,且整体的服务集合通过增加新的微服务或是更新已有微服务来满足各类新的需求;所述高可验证性即单一智能合约微服务的状态机规模简化,形式化验证过程周期减少,验证的准确性提高,且组合合约满足组合验证的标准从而进行验证。

2.根据权利要求1所述的一种建立智能合约微服务化的方法,其特征在于:所述基于RESTFUL接口规范包括HTTP协议。

一种建立智能合约微服务化的方法

技术领域

[0001] 本发明涉及本涉及区块链、智能合约、微服务领域，特别是涉及到智能合约的微服务化的建立方法，用于提升智能合约的可用性以及可扩展性。

背景技术

[0002] 1997年，智能合约的概念由Nick Szabo提出，发表于“Formalizing and Securing Relationships on Public Networks”中，他首先分析了传统合约(纸质合约)的不足，如不易保存，可执行力不强等。随后提出了智能合约的概念与基本原理，并说明通过使用计算机、互联网、密码协议等新的技术构造智能合约的可能性与优势。虽然智能合约理论几乎与互联网技术(World Wide Web)同时出现，但应用实践却一直严重地落后于理论，缺乏将这个理念转变为现实的清晰路径。主要面临两个方面问题，一是智能合约没有有效的手段来控制实物资产，保证合约的执行；二是单个计算机很难保证执行这些条款以获得合约方的信任，合约方需要可靠的解释和执行代码的可信环境，它无法亲自检查有问题的计算机，也无法直接观察与验证其他合约方的执行动作，只有让第三方审核各方合约执行的记录。而区块链技术的出现解决了这些问题，奠定了智能合约应用的基础，同时，智能合约也扩大了区块链的应用范围。区块链为完全数字化资产的记录和转移奠定了基础，通过完全数字化的资产，区块链给计算机代码提供直接控制资产的方法，使得智能合约具有执行力。在区块链上，资产的控制就是控制资产对应的密钥权限，而不是任何实物。区块链使计算系统成为受信任系统，它已经不仅限于数据库的功能，同时还是可以执行代码和记录数字资产所有权的分布式计算机，数字资产所有权就可以被上传和存储在区块链中，并根据指令执行，一旦区块链记录了合约代码，合约方就可以确定合约不会被更改。

[0003] 现有的基于区块链的智能合约技术还处于一个初级阶段，尚有很多问题还没有解决，包括：

[0004] (1)基于区块链的智能合约执行效率低下，智能合约的一次执行需要全节点共识，且一个区块包含的多个合约需串行执行，导致时间开销过大，效率低下；

[0005] (2)智能合约作为区块链的链上代码执行的安全问题，合约代码缺乏形式化方法对其进行建模验证，易导致代码安全漏洞；

[0006] (3)智能合约的可扩展性问题，目前智能合约独立性极高，业务逻辑基本不对外开放，合约之间的相互调用极少，合约使用完成后很难再次利用。随着合约业务逻辑逐渐向复杂化发展，合约的规模也越来越庞大，而合约内部的逻辑耦合过大，导致合约的维护和更新变得极为复杂。

发明内容

[0007] 本发明的发明构思来自云计算与区块链的快速发展，BaaS(BlockchainasaService，区块链即服务)快速发展，与早期云上的IaaS、PaaS、SaaS技术类似，BaaS技术的目的在于将区块链以及区块链技术包装成为服务，提供快速、便捷的区块链

部署和使用功能，方便企业级用户的使用，而基于区块链的智能合约也被移植到了BaaS上，成为了BaaS上实现各企业业务的方法，而如何将云上现有的技术与智能合约相结合，解决现有智能合约的问题，成为了一个重要的研究点，这也就是本发明的发明构思所在。

[0008] 本发明的目的在于提供一种建立智能合约微服务化的方法，包括：

[0009] 步骤一，设计智能合约微服务的基础容器；

[0010] 步骤二，设计智能合约微服务间的通信方式；

[0011] 步骤三，设计整体智能合约微服务的运行方式。

[0012] 优选的，所述微服务将应用程序构建为一系列松散耦合的服务，每个合约微服务均被部署在单个基础容器之中，提供给每个合约模版独立的运行、开发、验证的环境，单个合约的所述微服务在所述基础容器中执行，所述基础容器中的单个合约的所述微服务具有功能单一性，可组合性，高融入性，高可扩展性以及高可验证性，所述功能单一性即所述合约微服务所针对的业务是单一的；所述可组合性为将简单合约微服务能通过轻量级通信方式相互组合形成复杂合约，完成特定业务逻辑；所述高融入性即所述合约微服务可将服务暴露，通过轻量级通信服务访问，被封装成API从而被用户使用，并安插在自己的业务逻辑中；所述高可扩展性即所述合约微服务在开发完成后，只需将自身的服务暴露给其他微服务和用户便可被使用，且整体的服务集合通过增加新的微服务或是更新已有微服务来满足各类新的需求；所述高可验证性即单一智能合约微服务的状态机规模简化，形式化验证过程周期减少，验证的准确性提高，且组合合约满足组合验证的标准从而进行验证。

[0013] 优选的，所述步骤一设计智能合约微服务的基础容器包括：

[0014] (1)使用docker实现单一合约微服务的执行容器，从而隔离合约间的资源，使合约成为独立的模块；

[0015] (2)上层使用kubernetes对合约进行管理整体管理，包括合约服务的注册，以及合约容器的管理。

[0016] 优选的，所述步骤二设计智能合约微服务的通信方式，用于实现合约的可扩展性以及合约之间可进行组合使用，完成更为复杂的合约逻辑，采用两种方式实现智能合约微服务间的通信，一种是使用远程过程调用(Remote ProcedureCall)的方法，而另一种是使用基于RESTFUL接口规范的方法。

[0017] 优选的，所述基于RESTFUL接口规范包括HTTP协议。

[0018] 优选的，所述步骤三设计智能合约微服务运行方式包括：

[0019] (1)用户的请求被微服务的特定网关所获取，网关进行对应的解析处理；

[0020] (2)网关检查所请求的产生的单一合约的合法性以及组合合约的可组合性，在检查完成后将所述拥护的请求作为信息重新封装为特定的合约服务消息，设定服务访问序列；

[0021] (3)请求消息按照用户所需逻辑进入各个容器之中完成单一合约的业务逻辑，并对各个容器数据进行更新；

[0022] (4)最终处理结果被返回给网关，随后交付给用户；

[0023] (5)所述用户的请求的整个处理流程通过区块链接口存入到区块链中。

[0024] 采用本发明的方法的有益效果在于，将现有云技术与智能合约结合，形成一套将智能合约微服务化的方法，可有效提升合约的可扩展性，对于智能合约的开发与应用具有

重要的意义。

[0025] 根据下文结合附图对本发明具体实施例的详细描述，本领域技术人员将会更加明了本发明的上述以及其他目的、优点和特征。

附图说明

[0026] 后文将参照附图以示例性而非限制性的方式详细描述本发明的一些具体实施例。附图中相同的附图标记标示了相同或类似的部件或部分。本领域技术人员应该理解，这些附图未必是按比例绘制的。本发明的目标及特征考虑到如下结合附图的描述将更加明显，附图中：

[0027] 附图1为根据本发明实施例的智能合约微服务的基础容器的结构视图；

[0028] 附图2为根据本发明实施例的智能合约微服务的工作流程图。

具体实施方式

[0029] 云上的微服务技术是面向服务架构(SOA)风格的一种变体，它将应用程序构建为一系列松散耦合的服务，有效的提升项目的可扩展性。在微服务体系结构中，服务应该是细粒度的，协议应该是轻量级的。将应用程序分解为不同的小型服务的好处是它提高了模块性，使应用程序更易于理解、开发和测试。它还通过使小型自主小组独立开发，部署和扩展各自的服务来平行发展。同时这种技术也允许个体服务的架构通过不断的重构而出现，基于微服务的体系架构可以实现持续交付和部署。将云技术中的微服务架构应用到智能合约中，将云上的智能合约以微服务的形式进行开发可有效的解决其可扩展性差的问题。

[0030] 这种智能合约微服务化的实现方法，包括如下步骤：

[0031] 步骤一，设计智能合约微服务的基础容器；

[0032] 步骤二，设计智能合约微服务间的通信方式；

[0033] 步骤三，设计整体智能合约微服务的运行方式。

[0034] (一)关于智能合约微服务容器设计：

[0035] 微服务是面向服务架构(SOA)风格的一种变体，它将应用程序构建为一系列松散耦合的服务，可有效的提升项目的可扩展性。智能合约本质上是实现业务逻辑的代码，可很好的与微服务相结合。单个合约微服务在容器中执行，容器中的智能合约微服务具备以下特点：

[0036] 功能单一性：满足面向对象的原则，为保证高可重用率，智能合约微服务所针对的业务应是单一的。

[0037] 可组合性：用户所需业务通常较为复杂，简单合约微服务能通过轻量级通信方式相互组合形成复杂合约，完成特定业务逻辑。

[0038] 高融入性：智能合约微服务可将服务暴露，通过轻量级通信服务访问，可被封装成API方便被用户使用，并安插在自己的业务逻辑中。

[0039] 高可扩展性：智能合约微服务在开发完成后，只需将自身的服务暴露给其他微服务和用户便可被使用，且整体的服务集合可通过增加新的微服务或是更新已有微服务来满足各类新的需求。

[0040] 高可验证性：单一智能合约微服务由于其功能单一，简化了其生成的状态机的规

模,减少了形式化验证过程周期,提升了验证的准确性;组合合约满足组合验证的标准,可很好的通过该方法进行验证。

[0041] 单一合约微服务的执行容器使用docker实现,可很好的隔离合约间的资源,使合约成为独立的模块,在上层使用kubernetes对合约进行管理整体管理,包括合约服务的注册,以及合约容器的管理,如图1所示。

[0042] (二)关于智能合约微服务的通信设计:

[0043] 单一合约运行在相互独立的容器之中,为实现合约的可扩展性,合约之间可进行组合使用,完成更为复杂的合约逻辑。故须在各个合约容器之间建立通信。业界目前主要有两种方式实现微服务间的通信,一种是使用远程过程调用(Remote Procedure Call)的方法,而另一种是使用基于RESTFUL接口规范的方法(如HTTP等)。两种方法各有长处,RPC在性能上表现出色,而基于RESTFUL接口的方法则在调用和测试上简单方便。由于后期合约的组合频繁,故本实施例使用更为轻量级的RESTFUL接口方法进行通信。

[0044] (三)关于智能合约微服务的运行方式设计:

[0045] 智能合约微服务的实现分为部署和运行,每个合约微服务均被部署在单个的容器之中,提供给每个合约模版独立的运行、开发、验证的环境。智能合约微服务工作流如图2所示。

[0046] (1)用户的请求将被微服务的特定网关所获取,网关进行对应的解析处理。

[0047] (2)网关会检查所请求的产生的单一合约的合法性以及组合合约的可组合性,在检查完成后将请求信息重新封装为特定的合约服务消息,设定服务访问序列。

[0048] (3)请求消息会按照用户所需逻辑进入各个容器之中完成单一合约的业务逻辑,并对各个容器数据进行更新。

[0049] (4)最终处理结果将被返回给网关,随后交付给用户。

[0050] (5)整个请求的处理流程通过区块链接口存入到区块链中。

[0051] 在本实施例中,智能合约被微服务化,形成一个个的微服务合约,将复杂合约逻辑解耦,极大提升合约可重用性和可扩展性。该实施例的应用场景为网上购物。将网上的购物过程逻辑全部写入到一个合约之中,通常情况下,购物的合约逻辑为买家和卖家在网上签订完成购物合约,买家将预付金注入到合约之中,卖家发货后买家收到货物,若买家对货物满意,则触发合约将合约中的资金交予卖家,若买家不满意,则申请退货,合约中的资金返还给买家。现实中各类不同购物逻辑形成不同的合约,其中合约的大部分条款逻辑均与上述逻辑大体相似,只有少部分进行了更改。若针对不同购物逻辑形成不同合约,则合约很难被重用或是扩展,且会出现大量合约冗余。使用合约微服务化思想,将上述通用逻辑使用合约微服务表示,合约微服务可在不同的购物逻辑中被调用,则会极大提升了合约可用性和可扩展性,并降低了合约冗余。

[0052] 本实施例将现有云技术与智能合约结合,形成一套将智能合约微服务化的方法,可有效提升合约的可扩展性,对于智能合约的开发与应用具有重要的意义。

[0053] 虽然本发明已经参考特定的说明性实施例进行了描述,但是不会受到这些实施例的限定而仅仅受到附加权利要求的限定。本领域技术人员应当理解可以在不偏离本发明的保护范围和精神的情况下对本发明的实施例能够进行改动和修改。

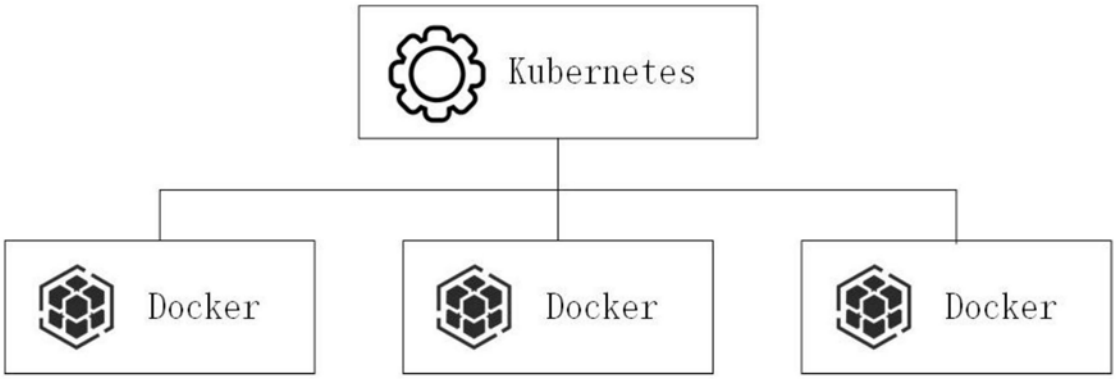

Kubernetes
Docker
Docker
Docker

图1

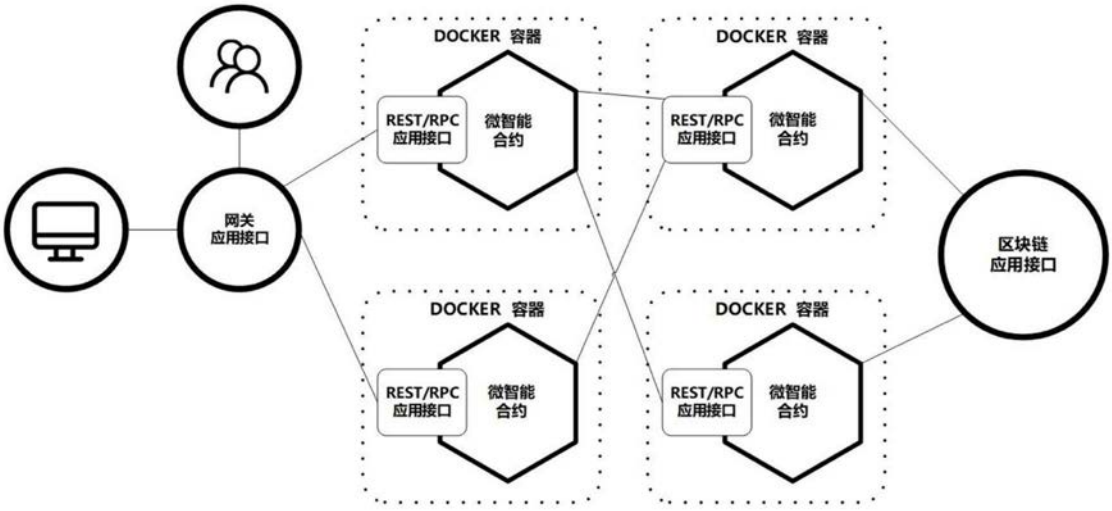

DOCKER 容器
REST/RPC 应用接口
微智能合约
DOCKER 容器
REST/RPC 应用接口
微智能合约
网关 应用接口
区块链 应用接口
DOCKER 容器
REST/RPC 应用接口
微智能合约
DOCKER 容器
REST/RPC 应用接口
微智能合约

图2