计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2020,29(4):97−101 [doi: 10.15888/j.cnki.csa.007341]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于区块链的外包服务公平支付方案①

陈嘉良, 林鸿瑞, 黄钿捷

(福州大学 数学与计算机科学学院, 福州 350108)
通讯作者: 陈嘉良, E-mail: 515629075@qq.com

摘　要: 随着外包计算服务的快速发展, 云计算吸引了越来越多的个人和企业使用外包服务提供商的服务. 而雾计算进一步将云计算扩展到网络边缘, 在雾计算中, 用户由于受计算资源的约束, 所以将计算任务外包给雾节点. 然而, 用户和雾计算节点之间的相互不信任, 将会导致公平支付的问题. 现有的大多数解决方案采用的是传统的支付机制, 需要依赖银行来实现支付. 为了实现外包服务的公平支付问题, 本文提出了基于区块链的外包服务公平支付方案, 通过区块链智能合约支付报酬. 同时本文提出的方案可以确保如果雾计算节点完成了计算任务, 则用户必须支付报酬给雾计算节点. 而如果雾计算节点没有完成计算任务, 则用户可以获得赔偿. 系统分析表明本方案实现了外包服务的正确性和公平性, 并且其消耗在可接受范围内.

关键词: 外包计算; 雾计算; 区块链; 公平支付; 以太坊

引用格式: 陈嘉良,林鸿瑞,黄钿捷.基于区块链的外包服务公平支付方案.计算机系统应用,2020,29(4):97–101. http://www.c-s-a.org.cn/1003-3254/7341.html

Outsourcing Services Fair Payment Scheme Based on Blockchain

CHEN Jia-Liang, LIN Hong-Rui, HUANG Dian-Jie

(College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350108, China)

Abstract: With the rapid development of outsourcing services, cloud computing has attracted an increasing number of individuals and enterprises to enjoy the services from outsourcing service providers. Moreover, fog computing further extends cloud computing to the edge of the network. In fog computing, because the end user is usually resource-constrained, the outsourcing computation tasks can be outsourced to the fog nodes. However, the mutual distrust between users and fog nodes may impede the fair payment of outsourcing services. Nevertheless, most existing solutions adopt the traditional payment mechanism, which needs a trusted authority such as a bank. In this study, in order to realize fair payment of outsourcing services, we introduce a new fair payment framework based on Blockchain in fog computing to directly transfer rewards by smart contract. Meanwhile, we present a construction to guarantee that if there is a malicious user, the honest one can get compensation. Finally, our security analysis indicates that the proposed protocol achieves correctness and fairness, and performance analysis shows that the experimental consumption is acceptable.

Key words: outsourcing computing; fog computing; Blockchain; fair payment; Ethereum

1　引言

由于灵活性和高可用性等好处, 云计算可以提供包括存储和计算在内的服务. 因此, 许多个人和企业将他们的数据外包到云平台上以节省成本. 但是与云的通信通常非常耗时, 而作为云计算的扩展, 雾计算使计算任务能够在网络边缘实现并提供低延迟的服务. 在

① 收稿时间: 2019-08-22; 修改时间: 2019-09-23; 采用时间: 2019-09-29; csa 在线出版时间: 2020-04-05

雾计算中, 资源受限的用户 (由 O 表示的外包用户) 可以将计算任务外包给雾计算节点来完成 (由 W 表示的工作节点) 并支付报酬给他们.

本文考虑能将计算任务分配给雾节点运行的场景, 每个雾计算节点完成相应的任务后将结果返回给用户. 在指定时间内完成任务后, 雾计算节点将从用户处获得报酬. 然而, 由于用户和雾计算节点之间的不信任, 公平支付的问题应该被考虑. 一方面, 雾计算节点可能没有完成计算任务, 也就是说, 雾计算节点可能会发送一些错误的结果给用户. 另一方面, 雾计算节点诚实地完成了任务, 但恶意用户却并不支付报酬.

目前已有一些解决问题的方案, 一方面, 用户在支付服务费用时需要验证计算结果. 文献[1]提出了一种基于抽样的方案. 文献[2]提出了一种审计机制, 通过使用计算证明来检测计算节点的恶意行为. 文献[3]提出了一种基于重复计算和 ringer 的方案, 以验证计算结果的正确性. 文献[4–6]提出了概率验证方法检测作弊者. 文献[7]提出了一种基于抽样的解决方案, 该解决方案使用 Merkle 树来防止计算节点作弊. 另一方面, 应考虑支付报酬的问题. 文献 [8,9]基于分割选择方案和秘密共享方案来防止恶意计算节点并考虑支付了问题.

在上述方案中, 要么不考虑支付的问题, 要么采用传统的支付框架, 例如银行. 然而, 传统的支付解决方案存在一些缺点, 银行可能是支付系统的瓶颈. 不同于传统的支付方式, 区块链是一种分布式的系统, 不受任何一方控制, 可以直接转移报酬. 而区块链技术已经被用在了很多外包服务中[10–12], 文献[1]提出了一种基于抽样并结合比特币的方案.

为了解决公平支付的问题, 本文提出了一个用于外包计算的基于以太坊区块链的公平支付方案. 在我们提出的方案中, 外包用户和工作节点可以互不信任. 基于以太坊的智能合约, 本文可以实现诚实的工作节点将会获得报酬, 同时如果工作节点未完成计算任务, 外包用户可以获得赔偿. 本文引入可信第三方 T 来解决外包用户和雾计算节点的冲突.

2 系统模型

系统模型如图 1 所示, 包含外包者 O, 工作节点 W, 第三方 T 和一个区块链.

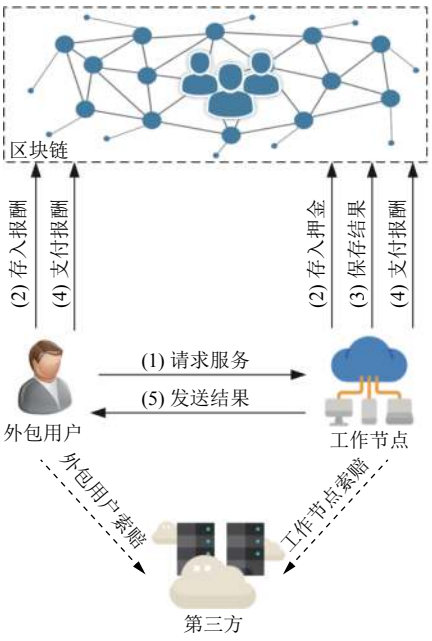

图 1 系统模型

(1) 外包用户 O: 作为外包计算的请求者, O 将一笔报酬存入智能合约中, 并向工作节点 W 请求外包计算服务. 如果 W 提供的结果正确, 则将支付报酬给 W. 否则, O 可以从 W 处获得赔偿.

(2) 工作节点 W: 作为外包计算服务的提供者, W 收到计算服务请求后将一笔押金存入智能合约中. 在完成计算任务后, W 将结果记录到区块链智能合约中, 并将结果发送给 O. 在指定时间 t 之前若 O 未对结果提出异议则从区块链获得报酬.

(3) 第三方 T: 作为第三方, T 接收来自 O 的请求. 一旦 O 发现 W 的计算结果错误, O 发送一个请求给 T. T 验证该请求, 若验证 W 的计算结果错误, 则执行智能合约惩罚 W.

(4) 区块链: 我们使用一个已被广泛使用并支持智能合约的区块链, 如以太坊区块链. 智能合约是在区块链上自我执行的程序.

在我们的系统中, 外包用户和工作节点可以互不信任, 同时它们中的任一个都可能是恶意用户. 具体来说, 恶意外包用户的目的是在不支付报酬的情况下获得外包服务, 而恶意工作节点则希望在不提供有效结果的情况下获得报酬. 我们的设计目标主要包括正确性和公平性.

(1) 正确性: 如果外包用户 O 和工作节点 W 都是诚实的, 那么外包用户 O 可以获得所需的计算结果, 而

工作节点 W 将获得报酬.

(2) 公平性: 对外包用户 O 的公平性意味着恶意工作节点 W 若未能提供正确的结果, 则无法获得报酬. 对工作节点 W 的公平性意味着恶意外包用户 O 在不支付服务费的情况下无法获得正确的结果. 如果恶意工作节点 W 未能提供正确的结果, 则外包用户 O 能够从工作节点处获得相应的赔偿.

3 系统设计

本文设计的方案包含 4 个阶段: 初始化阶段, 计算阶段, 支付阶段和索赔阶段. 同时引入第三方来解决外包用户 O 和工作节点 W 间的冲突. 区块链智能合约确保外包用户 O 要么获得正确的结果, 要么获得赔偿. 此外, 诚实的工作节点 W 可以获得相应的报酬.

3.1 初始化阶段

选择一个哈希函数 H 如 SHA-256, 并且每个参与者生成自己的 ECDSA 公钥/密钥对, 表示为(pk, sk), 并公布自己的公钥 pk 作为账户地址. 外包用户 O 的公私钥对表示为(pk_O, sk_O), 工作节点 W 的公私钥对表示为(pk_W, sk_W), 第三方 T 的公私钥对表示为(pk_T, sk_T). 假设所有参与者都安全地维护每个已发布的公钥, 而密钥 sk 安全的存储在本地, 用于生成签名.

外包用户与工作节点先对计算任务 F 达成协议, 并在区块链上建立智能合约. 其中计算任务表示为 $F = \langle f, D, M\rangle$, 计算函数 f 在数据 D 上所有满足 $f(x) \in M$ 的 x, 工作节点完成相应的计算任务后将包含所有满足要求的 x 结果集 S 返回给外包用户. 智能合约记录外包用户和工作节点还有第三方的账户地址, 并确定计算任务需要完成的时间和任务报酬. 外包用户将任务报酬存入智能合约中, 同时工作节点也将自己的押金存入智能合约.

3.2 计算阶段

在确认了外包用户将报酬存入智能合约后, 工作节点执行计算任务 F 后获得一个结果集$S = \{x_1, \cdots, x_n\}$, 其包含了所有满足$f(x) \in M$的 x. 工作节点将每个结果 x_i的哈希存在智能合约中, 并根据结果集 S 创建一棵 Merkle 树MT_l, 保存 Merkle 树的根节点I_{root}在智能合约中. 其中 l 表示 Merkle 树的树高, 而叶子节点的树高为 0. 在 Merkle 树中, 对于高度为 i 的第 j 个节点的值有$I_{i,j} = H(I_{i-1,j} \| I_{i-1,j+2^{i-1}})$, $I_{i-1,j}$和$I_{i-1,j+2^{i-1}}$表示$I_{i,j}$的两个孩子节点. 当$i = l$时, $I_{l,j}$表示根节点. 当$i = 0$时, $I_{0,j}$表示叶子节点, 即第 j 个结果x_j. 当结果集的 Merkle 树根被保存在区块链后, 工作节点之后将无法对结果集进行更改, 外包用户可以确保工作节点发送给他的结果集出现错误后无法进行否认. 如图 2 是一棵高度为 3 的 Merkle 树. 当结果存入区块链后, 工作节点将结果集 S 发送给外包用户, 并执行一个具有 t 时间锁的支付智能合约, 即在 t 时间后报酬将支付给工作节点.

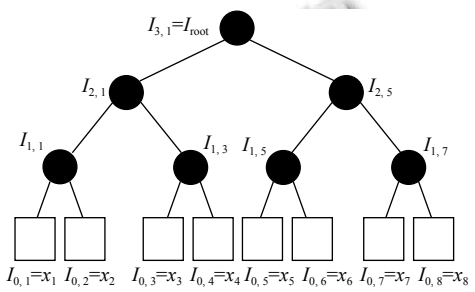

图 2 高度为 3 的结果 Merkle 树

3.3 支付阶段

外包用户将计算结果集 S 的每个元素x_i的哈希与存在区块链上的哈希对比是否一致, 若一致则验证结果是否满足外包任务的要求. 当结果集 S 正确, 则工作节点将在时间 t 后从合约中获得报酬. 当结果集 S 较大时, 可以使用如下的抽样方案, 并验证抽样的结果 S', 使用文献[7]中的方法进行生成 m 个抽样:

$$i_k = (H^k(I_{\text{root}}) \mod n) + 1, \text{ for } k = 1, \cdots, m \quad (1)$$

其中, I_{root}表示结果集 Merkle 树的根节点, 并且:

$$H^k(I_{\text{root}}) = \begin{cases} H(I_{\text{root}}), & \text{for } k = 1 \\ H(H^{k-1}(I_{\text{root}})), & \text{for } k = 2, \cdots, m \end{cases} \quad (2)$$

外包用户验证抽样的 m 个结果, 文献[7]中证明了只要 m 足够, 则能确保整个结果的正确性. 在外包用户验证了结果正确后, 工作节点将在时间 t 后获得报酬. 然而, 如果外包用户发现存在不正确的结果, 则外包用户将发送一个裁决请求给第三方.

3.4 索赔阶段

当第三方 T 从外包用户处收到裁决请求时, 则进入索赔阶段. 第三方 T 裁决这个请求的正确性, 如果接受这个裁决请求, 则第三方 T 执行智能合约中止支付报酬. 考虑以下两种情况.

(1) 若工作节点发送给外包用户的结果集计算的 Merkle 树根与存在区块链上的I_{root}不一致, 则表明工作节点发送的结果不正确. 外包用户发送一个裁决请求

给第三方包含结果集与区块链上的I_{root}, 第三方验证后要求工作节点返回正确的结果集 S, 并验证该结果集的 Merkle 树根是否与I_{root}一致. 若一致, 则发送该结果集给外包用户 O. 若不一致, 则调用合约 *judge* 函数中止支付报酬.

(2) 若结果集 S 的元素x_i验证结果为$f(x_i) \notin M$, 即结果集 S 存在不正确的结果, 表明工作节点 W 并未完成计算任务. 当外包用户 O 发现工作节点返回的结果中存在错误结果x_i, 则外包用户将外包任务$F = \langle f, D, M \rangle$, 错误的结果$x_i$与该元素哈希存在区块链上的位置包含在请求中发送给第三方 T. 第三方 T 先将元素x_i与该元素在区块链上的哈希进行验证, 若不相同, 则拒绝该请求. 当确定该元素为工作节点的计算结果后, 第三方 T 验证元素x_i在外包任务$F = \langle f, D, M \rangle$中结果是否正确, 即将结果$x_i$代入外包任务中计算验证$f(x_i)$是否满足 M. 若该结果不满足外包任务要求, 则第三方 T 调用合约中的 *judge* 函数修改变量 *payService* 为 false 使得 callback 函数中报酬支付中止. 使用的部分伪代码如图 3.

```
contract tlocked_servicepay{
    mapping (address => int) public balanceOf; //余额
    mapping (address => int) depositpool; //押金池
    address OUTSOURCER,WORKER,THIRDPARTY,DEPOSIT; //参与方地址
    int fee; // 报酬
    int T; //区块数
    int MerkleRoot; //任务结果的Merkle树根
    bool paysearch=true; // 值为false时取回报酬
    //外包用户存放押金
    function depositOutsourcer(int _value) returns (bool){
        // 验证外包用户身份
        assert (msg.sender == Outsourcer);
        // 检测外包用户是否有足够的余额
        assert (balanceOf[Outsourcer] >= _value);
        balanceOf[Outsourcer] -= _value;
        balanceOf[DEPOSIT] += _value;
        depositpool[Outsourcer] += _value;
    }
    //工作节点存放押金
    function depositWorker(int _value) returns (bool) {
        assert (msg.sender == Worker);
        assert (balanceOf[Worker] >= _value);
        balanceOf[Worker] -= _value;
        balanceOf[DEPOSIT] += _value;
        depositpool[Worker] += _value;
    }
    function Pay(string _root) {//报酬支付
        if(msg.sender == Worker && MerkleRoot ==_root){
            assert(depositpool[Worker] >= fee);
            T = block.number + t; // 现在的区块数加t
            //T区块之后调用
            Alarm Clock Service(T, callback());
            depositpool[Outsourcer] -= fee;
        }
    }
    function judge() returns (bool) {//第三方裁决函数
        assert (msg.sender==THIRDPARTY);
        payService = false;
        depositpool[Outsourcer] += fee;
    }
    function callback() returns (bool) {
        if (payService == true && block.number > T)
            Transfer (DEPOSIT,Worker, 2*fee);
    }
    function TxClaim() {//索赔函数
        if(msg.sender == Outsourcer && payService == false ){
            Transfer (DEPOSIT, Outsourcer, 2*fee);
        }
    }
}
```

图 3 智能合约部分伪代码

4 系统分析

4.1 安全分析

根据本系统的安全目标, 给出了本文的安全分析.

(1) 正确性: 如果使用的哈希函数 H 是抗碰撞的并且 ECDSA 签名是不可伪造的, 则我们的协议满足正确性. 假设外包用户和工作节点都是诚实的, 并且遵循方案的步骤. 在计算阶段, 工作节点将每个结果的哈希和结果集构成的 Merkle 树根节点存在区块链上. 在支付阶段, 外包用户将验证结果集元素的哈希是否与区块链保存的一致, 保证结果集 S 是工作节点保存在区块链上的结果, 之后验证所有或是抽样的结果是否满足外包计算任务的要求. 只有在满足了结果是符合外包任务要求后, 工作节点才能在时间 t 后收到报酬. 换句话说, 如果使用的哈希函数 H 是抗碰撞的并且 ECDSA 签名是不可伪造的, 由于区块链的不可篡改, 则外包用户收到的结果集 S 一定是未被篡改的, 且工作节点只有在提供的结果集 S 满足外包任务的要求后才能获得相应的报酬.

(2) 公平性: 我们首先证明对诚实工作节点的公平性, 然后证明在恶意工作节点下考虑对外包用户的公平性.

情况 1: 假设工作节点是诚实的, 而外包用户是恶意的, 即恶意的外包用户想要获得一个有效的结果而不支付报酬. 这种情况下, 在计算阶段, 工作节点只有在确认了外包用户将报酬存入智能合约中才会执行计算任务. 当完成计算任务后, 只有在工作节点的结果出现问题时支付才会被第三方 T 通过合约中止. 由于智能合约的强制执行特性, 所以只要工作节点提供的结果是符合外包计算任务要求的, 则工作节点一定能够在时间 t 后获得报酬.

情况 2: 假设外包用户是诚实的, 而工作节点是恶意的, 即恶意的工作节点想要在不提供正确结果的情况下获得报酬. 首先, 若工作节点未将结果的哈希或结果集的 Merkle 树根保存在区块链上, 则外包用户发送一个裁决请求给第三方, 第三方要求工作节点将结果集哈希保存于区块链中. 若工作节点未能将结果集哈希保存在区块链上, 表示工作节点未完成外包计算任务. 第三方 T 调用如图 3 的合约 *judge* 中止报酬的支付, 则外包用户可以获得赔偿并取回报酬. 其次, 若工作节点提供的结果不符合外包计算任务的要求, 外包用户将错误的结果包含到裁决请求中发送给第三方. 当第三方 T 验证了该错误结果后, 将中止报酬的支付,

则外包用户可以获得赔偿并将报酬取回.

通过以上的分析, 如果使用的哈希函数 H 是抗碰撞的并且 ECDSA 签名是不可伪造的, 则本系统满足正确性和公平性.

4.2 消耗分析

本系统在以太坊官方测试网络上实现了一个智能合约来分析性能. 本文使用的哈希函数是 SHA-256. 当我们进行实验时, gas 价格设置为 2 Gwei, 其中 1 Gwei = 10^9 wei = 10^{-9} ether, 目前 1 ether=168 USD. 我们将它部署在以太坊官方测试网络 Ropsten 上, 使用的伪代码表示的算法如图 3. 表 1 是智能合约消耗的实验结果, 合同创建操作仅执行一次以完成初始化其消耗 267 202 gas=0.0898 USD. 外包用户的报酬存入和工作节点的押金存入分别消耗 21 485 gas=0.0072 USD 和 21 397 gas=0.0072 USD, 而支付操作消耗 41 533 gas=0.0140 USD. 当出现恶意工作节点时, 进入索赔阶段, 第三方 T 裁决操作 *judge* 消耗 22 086 gas=0.0074 USD. 同时索赔操作消耗 29 383 gas=0.0099 USD. 而本实验当未出现错误结果智能合约共需消耗 351 617 gas, 约为 0.1182 USD. 当出现错误结果时, 需执行裁决操作与索赔操作, 智能合约共需消耗 403 086 gas, 约为 0.1355 USD. 第三方 T 在智能合约上的消耗只有执行裁决操作 *judge* 的消耗, 而对与链下的验证操作需根据具体的外包任务来确定.

表 1 智能合约的消耗

操作	gas 消耗	实际消耗 (ehter)
合约创建	267 202	0.000 534 404
报酬存入	21 485	0.000 042 970
押金存入	21 397	0.000 042 794
支付操作	41 533	0.000 083 066
裁决操作	22 086	0.000 044 172
索赔操作	29 383	0.000 058 766

5 结语

随着外包服务的快速发展, 为了解决外包计算的支付问题, 本文提出了基于区块链的外包服务公平支付方案. 通过使用区块链将外包任务的结果进行保存, 使其不能篡改. 只有在结果正确时, 外包用户才支付服务报酬给工作节点, 若结果不正确, 外包用户将可以获得赔偿. 本协议的安全分析和消耗分析表明本协议是正确的且公平的, 同时本协议的消耗是可接受的.

参考文献

1 Huang H, Chen XF, Wu QH, *et al*. Bitcoin-based fair payments for outsourcing computations of fog devices. Future Generation Computer Systems, 2018, 78: 850–858. [doi: 10.1016/j.future.2016.12.016]

2 Wei LF, Zhu HJ, Cao ZF, *et al*. Security and privacy for storage and computation in cloud computing. Information Sciences, 2014, 258: 371–386. [doi: 10.1016/j.ins.2013.04.028]

3 Golle P, Mironov I. Uncheatable distributed computations. Proceedings of the Cryptographers' Track at the RSA Conference. San Francisco, CA, USA. 2001. 425–440.

4 Chen XF, Li J, Susilo W. Efficient fair conditional payments for outsourcing computations. IEEE Transactions on Information Forensics and Security, 2012, 7(6): 1687–1694. [doi: 10.1109/TIFS.2012.2210880]

5 Küpçü A. Incentivized outsourced computation resistant to malicious contractors. IEEE Transactions on Dependable and Secure Computing, 2017, 14(6): 633–649. [doi: 10.1109/TDSC.2015.2499738]

6 Ulusoy H, Kantarcioglu M, Pattuk E. TrustMR: Computation integrity assurance system for MapReduce. Proceedings of 2015 IEEE International Conference on Big Data. Santa Clara, CA, USA. 2015. 441–450.

7 Du W, Jia J, Mangal M, *et al*. Uncheatable grid computing. Proceedings of the 24th International Conference on Distributed Computing Systems. Tokyo, Japan. 2004. 4–11.

8 Carbunar B, Tripunitara M. Fair payments for outsourced computations. Proceedings of 2010 7th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks. Boston, MA, USA. 2010. 1–9.

9 Carbunar B, Tripunitara MV. Payments for outsourced computations. IEEE Transactions on Parallel and Distributed Systems, 2012, 23(2): 313–320. [doi: 10.1109/TPDS.2011.163]

10 Zhang YH, Deng RH, Shu JG, *et al*. TKSE: Trustworthy keyword search over encrypted data with two-side verifiability via blockchain. IEEE Access, 2018, 6: 31077–31087. [doi: 10.1109/ACCESS.2018.2844400]

11 Do HG, Ng WK. Blockchain-based system for secure data storage with private keyword search. Proceedings of 2017 IEEE World Congress on Services. Honolulu, HI, USA. 2017. 90–93.

12 Zhang YH, Deng R, Liu XM, *et al*. Outsourcing service fair payment based on blockchain and its applications in cloud computing. IEEE Transactions on Services Computing, 2018: 1. [doi: 10.1109/TSC.2018.2864191]