doi:10.3772/j.issn.1006-6748.2021.02.004

Blockchain-based verifiable computation with optimized resource allocation①

Yang Ruizhe (杨睿哲)②, Tian tian, Zhang Zheng, Li Meng, Zhang Yanhua
(Faculty of Information Technology, Beijing University of Technology, Beijing 100124, P. R. China)

Abstract

Nowadays, the data that users need to calculate and process increases sharply, however, ordinary users usually lack the required capability. Therefore, resorting to outsourcing computation, they can delegate computing tasks to high-performance nodes over the network to meet their needs. In order to ensure the correctness of outsourcing computations, a verifiable computing scheme based on the blockchain smart contract is proposed, where the primary node and the replica nodes complete the task calculation and verification respectively, and reach a final consensus on the results. Moreover, the computing resources and energy consumption of each node to make the consensus are analyzed, based on which an optimization of resources allocation is proposed to maximize the transaction throughput. The simulation results show the effectiveness of the proposed scheme built on distributed consensus and also the throughput improvement by optimizing.

Key words: blockchain, verifiable computing, practical Byzantine fault tolerance (PBFT)

0 Introduction

In recent years, with the in-depth development of computer, the Internet industry continues to flourish, and the data that needs to be processed is increasing dramatically in an exponential trend. However, due to the limitation of computing power and equipment cost, ordinary users usually cannot complete the huge computing task. To solve this problem, outsourcing computation emerges, which enables users to delegate computing tasks to one or more powerful servers through the Internet in an efficient and cost-effective way. However, the dynamic, randomness, complex and openness of outsourcing computing bring challenges to the reliability of the computation. For example, the server may return an answer without the accurate calculation in order to save its computing resources. In addition, the vulnerability of the software, the instability of the network and the failure of the hardware may also make the correct calculation results failed.

Therefore, how to ensure the correctness of outsourcing computation has become one of the key points in research. Some verifiable computer mechanisms have been designed traditionally based on the theories of computational complexity and cryptography technology, such as interactive proving system[1-2], probabilistic checking of proofs (PCP proving)[3], homomorphic encryption[4-5] and signatures[6]. Besides, Ref. [7] proposed a scheme with adjustable correctness, which required the user to re-execute part of the calculation for verification. Ref. [8] used the redundancy of multiple servers to verify the results, so that users can be guaranteed to get correct results as long as one server was honest. Ref. [9] studied from the perspective of economics by establishing the optimal model of outsourcing contract price on the premise of ensuring the correct calculation, but it required the servers to be completely honest. Ref. [10] introduced game theory and analyzed the preferences of users and servers being rational. Taken together, the interaction between several parts is usually very complex with high communication cost[4-6], and the involvement of users, particularly their partly calculation re-performing still imposes a nonnegligible burden on the ones with low abilities[1-3,7]. Moreover, the assumption in Refs [8-10] that the server is either honest or malicious is not consistent with true scene, where servers usually have different preferences or randomness.

On the other side, blockchain originated from bitcoin[11] to solve the problem of double spending in digital cryptocurrency, has attracted extensive attention due to its decentralized, transparent and tamper-proof features. Soon afterwards, blockchain goes far beyond

① Supported by the National Natural Science Foundation of China (No. 61671029), Foundation of Beijing Municipal Commission of Education (No. KM202010005017) and Doctoral Fund of Ministry of Education of China (No. 2018M640032).

② To whom correspondence should be addressed. E-mail: yangruizhe@bjut.edu.cn

Received on May 19, 2020

its original design and becomes a basic distributed technology in secure manner, which is rapidly applied in the fields of supply chain, credit investigation, product traceability, copyright trading, digital identity, etc., promoting the transition from information Internet to value Internet[12]. Since its own storage and computing capacity of blockchain is insufficient, the existing computing system running on the blockchain has become a widely acceptable approach to meet the huge computing demand, such as the integration of blockchain with edge computing and industrial Internet of Things (IoT) technology[13-14]. However, for computation with assurance of the correctness, its traditional verification mechanism using interaction proof is not suitable to be directly run over blockchain, especially the public chains with proof of work (PoW) consensus, which lacks the mechanism to support interactions between the participants. Therefore, the current scheme mainly uses the incentives of blockchain to ensure the operation of calculation verification[15-17]. Ref. [15] proposed the Ethereum computer scheme, which allowed users to outsource computation to the Ethereum network by paying a certain reward to obtain the correct answer. The Truebit system in Ref. [16] enhances the ability of Ethernet computers. It provides incentives for proper outsourcing computation through a new double-layer validation mechanism, which solves the problem of verifier in Ethereum verification work by imposing forced errors and bonus. Some other studies[18-19] work on the integration of blockchain and the encrypted computation, where multi-party computation (MPC) as an encrypted computation gets the correct outputs and blockchain maintains an open reputation system to overcome a 'denial of service' of malicious adversary.

Different from PoW scaling to a large number of uncertain nodes, Byzantine fault tolerance (BFT) consensus[20] based on identified community to reach agreement avoids the exhausting computation power consumptions that are used to limit the adversaries, therefore has higher efficiency and throughput. In the widely used hyperledger fabric, practical BFT (PBFT) is employed to achieve consensus. In the evolving of the Ethereum, PBFT consensus runs within the shard, and the hotstuff is developed from BFT consensus in the Facebook Libra[21]. Besides the performance improvements, the interactions between the nodes among the community to reach the BFT consensus, has the similar characteristics with the traditional outsourcing computation. However, the directive combination will make frequent interaction of traditional outsourcing computation too much overhead for PBFT. Therefore, how to contribute PBFT consensus to the verification of outsourcing computation is a key issue to be studied.

A verifiable computing based on blockchain is proposed. Using PBFT consensus and smart contract, users send their computing tasks to the blockchain node in the form of transactions, and the primary node computes the tasks required in the transactions, and then packages the transactions into blocks with the results, which will be verified by the replica nodes according to the Byzantine consensus. Here, the calculation and verification driven by the smart contract ensure the trust of the calculation within blocks on chain as well as the correctness. Furthermore, the resources and energy consumptions of the proposed scheme are analyzed, where the transaction throughput conditioned on energy consumption is optimized. The results show the efficiency of the proposed scheme to realize the correct outsourcing computations by optimizing the resources used.

1 System model

In this section, the model of the blockchain-based verifiable computation system is presented.

Here, users send their data or computing tasks to the blockchain-based verifiable computation system in the form of transactions, where the computation and verification are defined by smart contract to obtain the correctness of calculation, and further the underlying PBFT consensus guarantees their reliable operation so as to ensure the correct results. This final accuracy results will be returned so that users do not need to do extra validation.

Considering that PBFT protocol runs on the view controlled by the primary node and verified by other replica nodes, it can be assumed that the computation is executed by the primary node and then verified by the validators according to the designed smart contract. To achieve a clear explanation, some concepts and mapping relations are firstly illustrated as follows.

(1) Clients and replicas. Clients are the users (end devices) submitting transaction requests, and the replicas are the nodes to execute and verify the requested transactions and finally reach the distributed consensus.

(2) Primary node and validators. Primary node is one of the replicas to execute the computations requested by transactions and then put them into blocks, which will be verified by the other replicas called validator.

(3) Transactions, blocks and smart contract. A transaction is a single event to be permitted or verified

by the underlying blockchain consensus. Different from the financial transaction in Bitcoin and the state changing in Ethereum, this work considers a transaction of completing a computation job. A block is a package of transactions plus a block header containing the metadata, in which the hashes of the current and previous block and Merkle hash of the transactions are all tamper-resistant and make the blocks chained one by one. Smart contract is a self-enforcing agreement embedded in computer code running on top of a blockchain. When the rules in the code are met, the agreement is automatically implemented.

To have an outsourced computation in a secure manner, clients send their transactions to the associated replica nodes with their signatures. On the other side, the primary node firstly verifies the signature and message authentication code (MAC) of each transaction received. If valid, the computation requested in transactions are executed, and then with the results added according to the smart contract, all of which are packaged into a new block and broadcasted to the validators. Each validator verifies the received blocks, including the signatures and MACs of both the block and the transactions within the block. If valid, it continues the verification of the internal calculation results of the transactions, the implementation of which is defined in the smart contract. Afterwards, the validators send the verification messages with their signatures added in to each other. The blocks passing all these verifications will be appended to the blockchain. Table 1 and Table 2 give the form of the transaction and the smart contract.

Note that although the rules in contracts depend on the specific computation task, the computation complexity and its resources exhausted respectively in the execution and verification are usually substantially different, such as the hashing and matrix operation. For example, this work considers the inverse calculation of

Table 1 The form of the transaction

Item	Description
Transaction number	The number of the transaction in the block
Transaction ID	Hash of this transaction
Scope	Account scopes
Message/Action	Task to be calculated (Ex. the inverse of a matrix) Call smart contract Calculation results
Signature	The signature of this transaction
MAC	The MAC of this transaction

Table 2 The form of the smart contract

Item	Description
Version	Current version
Address	Call address of smart contract
Value	Matrix to be calculated
Functions	Primary node (Ex. inverse of a matrix) Validator: (Ex. verification by the product of matrix and its inverse)

matrix, where it has the complexity of $O(L^3)$ for obtaining the inverse of the matrix $\boldsymbol{C}_{L\times L}$ and the complexity of $O(L^2)$ for computing $\boldsymbol{C}^{-1}\boldsymbol{C}$. Therefore, generally it can be assumed that the central processing unit (CPU) cycles for the computation task of one transaction at the primary node is α and the CPU cycles required for the verification at the validator is α', $\eta = \alpha'/\alpha \leqslant 1$.

2 Consensus and performance analysis

In this section, the consensus steps of outsourcing computing based on the specific steps of PBFT protocol is analyzed.

There are N nodes (the servers), with the computation capacity of each node denoted by f_n (CPU cycles per second), $n = 1, \cdots, N$. It is note that PBFT is safe in asynchronous environments such as the Internet, with the recovery mechanisms to tolerate any number of faults over the lifetime of the system provided fewer than $(N-1)/3$ replicas become faulty[20]. Therefore, the verifiable computation over PBFT consensus inherits this security to ensure the correctness of the computing results that pass the computation and verifications even the arbitrary replica presents faulty at any time unless more than 1/3 of the replicas are faulty at a time.

The consensus protocol consists of 5 steps based on PBFT, as shown in Fig. 1, where generating or verifying one signature requires β cycles, and generating or verifying one MAC requires θ cycles.

Step 1 (request→pre-prepare) The clients submit requests (transactions) to the primary node n', and the primary node picks up a batch of K transactions from the transaction pool with time interval of T. Firstly, the signatures and MACs of these transactions will be verified, which are considered as an uncivil execution, and a fraction g of transactions can pass the verification. Next, the computation task contained in the valid transactions will be executed by smart contracts

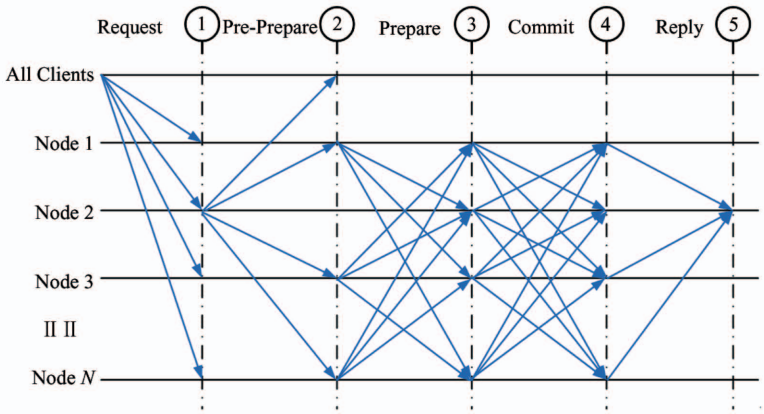

Fig. 1 PBFT protocol

(Table 2). Further, these transactions with computing results and other important information will be packaged into a new block, as shown in Table 1. And multicast as the prepared block along with a pre-prepare message to other nodes for validation. The pre-prepare message contains the signature and MAC of the primary node and hashed result of the block.

Therefore, for each prepared block produced at the primary node, the computation cost Δ_1 and time $T_{1n'}$ can be written as

$$\begin{aligned} \Delta_1 &= \frac{K}{g}(\beta+\theta) + K\alpha + \beta + (N-1)\theta \\ T_{1n'} &= f_{1n'}^{-1}\Delta_1 \end{aligned} \tag{1}$$

Step 2 (pre-prepare→prepare) The validator $n'' \neq n'$ receives the new block with the pre-prepare message, and first verifies the signature and MAC of the block, then the signatures and MACs of the transactions and finally the computation result inside the transaction via the code in smart contracts. Afterwards, the validator sends a prepare message with the verification results to all the other replicas with its signature and MAC added in. The cost and time of this step can be written as

$$\begin{aligned} \Delta_2 &= \beta + \theta + K(\beta+\theta) + K\alpha' + \beta + (N-1)\theta \\ T_{2n''} &= f_{2n''}^{-1}\Delta_2 \end{aligned} \tag{2}$$

Step 3 (prepare→commit) Each replica receives and checks the prepare message to make sure that it is consistent with the pre-prepare message. Once upon receiving $2f$ matching prepare messages with passing verification results, the replica n'' sends out a commit message to all the others, which includes the signature and MAC. The computation cost and time are

$$\begin{aligned} \Delta_3 &= 2f(\beta+\theta) + \beta + (N-1)\theta \\ T_{3n} &= f_{3n}^{-1}\Delta_3 \end{aligned} \tag{3}$$

Step 4 (commit→reply) Each replica receives and checks the commit message to make sure that it is consistent with the prepare message. Once having $2f$ matching commit messages from the other replicas, a reply message with the signature and MAC will be delivered to the primary node. The computation cost and time are

$$\begin{aligned} \Delta_4 &= 2f(\beta+\theta) + K(\beta+\theta) \\ T_{4n''} &= f_{4n''}^{-1}\Delta_4 \end{aligned} \tag{4}$$

Step 5 (reply → be appended to the chain) The primary node receives and checks the reply message. After the primary node receives $2f$ matching reply messages, the new block becomes valid and will be appended to the blockchain. The computation cost and time are

$$\begin{aligned} \Delta_5 &= 2f(\beta+\theta) \\ T_{5n'} &= f_{5n'}^{-1}\Delta_5 \end{aligned} \tag{5}$$

3 Optimized resource allocation

As analyzed in Section 2, the efficiency of block producing is limited by the computation time of each consensus step at each replica, which depends on the complexity of the computing and the afforded computation resources f_{sn}, $s = 1, \cdots, 5$. Here, this work aims to maximize the throughput of blockchain K/T by optimally using the computation resources under the constraint of the energy used.

For the energy cost, this work follows the model in Ref. [18]. When the CPU cycle frequency is f_{sn}, the main energy consumption in the consensus calculation process is denoted as

$$F_{e_{sn}}(f_{sn}, K) = \delta_{sn}\Delta_{sn}\gamma(f_{sn})^2 \tag{6}$$

where, γ is a constant related to the hardware architecture and $\delta_{sn} = 0, 1$ is used to indicate whether the node n participates in the step s. Thus, the vector $\delta_{n'} = [1, 0, 1, 1, 1]$ represents the participation of the primary node n' in the steps of the consensus realization, and $\delta_{n''\neq n'} = [0, 1, 1, 1, 0]$ is used for the validators. Sub-

sequently, the energy cost of the system can be denoted as

$$F_e(\boldsymbol{f}, K) = \sum_{n=1}^{N}\sum_{s=1}^{5} F_{e_{sn}}(f_{sn}, K) \tag{7}$$

where the vector $\boldsymbol{f} = [f_{sn}]^{\mathrm{T}}_{1\times 5N}$ represent the whole computing resources.

Here, It can be defined the average time of being on chain for each transaction as

$$F(K, T) = \frac{T}{K} \tag{8}$$

which is actually the inverse of the system throughput. Therefore, to achieve maximum throughput, the target is designed to be

$$\begin{aligned} &\min F(K, T)\\ &\text{s.t.} \quad C1: \sum_{s=1}^{5} f_{sn} \leqslant f_n, \quad \forall n\\ &\qquad C2: \frac{\Delta_{sn}}{f_{sn}} \leqslant T, \quad \forall s, n\\ &\qquad C3: F_e(\boldsymbol{f}, K) \leqslant P \end{aligned} \tag{9}$$

where P is the total energy consumption provided in the system.

4 Optimization solution

For the constrained optimization problem in Eq. (9), the interior point method[22], an effective algorithm to solve nonlinear convex optimization problems, can be used. First, according to the principle of interior point method, the optimization function is rewritten as

$$\begin{aligned} &\min F(K, T) = \frac{T}{K}\\ &\text{s.t.} \quad C1: g_n^1(\boldsymbol{f}) = \sum_{s=1}^{5} f_{sn} - f_n \leqslant 0\\ &\qquad C2: g_{sn}^2(K, T, \boldsymbol{f}) = \frac{\Delta_{sn}}{f_{sn}} - T \leqslant 0\\ &\qquad C3: g^3(K, \boldsymbol{f}) = \sum_{n=1}^{N}\sum_{s=1}^{5} \Delta_{sn}\delta_{sn}\gamma(f_{sn})^2 - P \leqslant 0 \end{aligned} \tag{10}$$

For inequality constraints, penalty function $\mu\sum_i \ln(p_i)$ is introduced to approximate the original objective function, so that the optimization problem can be transformed into convex optimization under equality constraints:

$$\begin{aligned} &\min F_\mu(K, T) = \min F(K, T) - \mu\sum_i \ln(p_i)\\ &\text{s.t.} \quad \boldsymbol{g}(K, T, \boldsymbol{f}) + \boldsymbol{p} = 0 \Rightarrow \boldsymbol{p} = -\boldsymbol{g}(K, T, \boldsymbol{f}) \end{aligned} \tag{11}$$

where,

$$\begin{aligned} \boldsymbol{g}(\cdot) &= \left[g_1^1(\cdot), \cdots, g_N^1(\cdot), g_{11}^2(\cdot), \cdots, g_{51}^2(\cdot), \right.\\ &\qquad \left. \cdots g_{1N}^2(\cdot), \cdots, g_{5N}^2(\cdot), g^3(\cdot)\right]^{\mathrm{T}}\\ \boldsymbol{p} &= [p_1, \cdots, p_i, \cdots, p_{6N+1}]^{\mathrm{T}} \end{aligned} \tag{12}$$

It should be noted that the elements in the relaxed vector $\boldsymbol{p}$ map one-to-one to the elements in the inequality constraint vector $\boldsymbol{g}$. Besides, p_i is required to be positive to keep boundaries, and the smaller parameter $\mu > 0$ brings $F_\mu(\cdot)$ closer to $F(\cdot)$. Therefore, this work can give the approximation by adjusting the value of μ. In other words, when μ is reduced to 0, the minimum value of $F_\mu(\cdot)$ should be equal to the minimum value of $F(\cdot)$. When the constraints are satisfied, the objective function can be written as

$$\min F_\mu(K, T, \boldsymbol{f}) = \frac{T}{K} - \mu\boldsymbol{I}^{\mathrm{T}}\ln(-\boldsymbol{g}(K, T, \boldsymbol{f})) \tag{13}$$

To get the minimum value of $F_\mu(\cdot)$, this work has to find the variables making the gradient of $F_\mu(\cdot)$ being 0. Thus, for $F_\mu(\cdot)$, the gradient $\nabla F_\mu(\cdot)$ and Hessian matrix $\boldsymbol{H}(F_\mu(\cdot))$ are given as

$$\nabla F_\mu(\cdot) = \left[\frac{\partial F_\mu(\cdot)}{\partial K}, \frac{\partial F_\mu(\cdot)}{\partial T}, \frac{\partial F_\mu(\cdot)}{\partial f_{11}}, \cdots, \frac{\partial F_\mu(\cdot)}{\partial f_{sn}}\right]^{\mathrm{T}}$$

$$\boldsymbol{H}(F_\mu(\cdot)) = \begin{bmatrix} \frac{\partial^2 F_\mu(\cdot)}{\partial K\partial K} & \frac{\partial^2 F_\mu(\cdot)}{\partial K\partial T} & \cdots & \frac{\partial^2 F_\mu(\cdot)}{\partial K\partial f_{sn}}\\ \frac{\partial^2 F_\mu(\cdot)}{\partial T\partial K} & \frac{\partial^2 F_\mu(\cdot)}{\partial T\partial T} & \cdots & \frac{\partial^2 F_\mu(\cdot)}{\partial T\partial f_{sn}}\\ \vdots & \vdots & \ddots & \vdots\\ \frac{\partial^2 F_\mu(\cdot)}{\partial f_{sn}\partial K} & \frac{\partial^2 F_\mu(\cdot)}{\partial f_{sn}\partial T} & \cdots & \frac{\partial^2 F_\mu(\cdot)}{\partial f_{sn}\partial f_{sn}} \end{bmatrix} \tag{14}$$

Here, this work needs to choose different solutions according to whether Hessian matrix is invertible.

If it is an invertible matrix, the Newton iteration method can be used. The relationship between the parameters of iteration v and the ones of iteration $v-1$ is given as

$$\begin{aligned} [K^{(v)}, T^{(v)}, (\boldsymbol{f}^{(v)})^{\mathrm{T}}]^{\mathrm{T}} &= [K^{(v-1)}, T^{(v-1)}, (\boldsymbol{f}^{(v-1)})^{\mathrm{T}}]^{\mathrm{T}}\\ &\quad - \boldsymbol{H}(F_\mu^{(v-1)}(\cdot))^{-1}\nabla F_\mu^{(v-1)}(\cdot) \end{aligned} \tag{15}$$

In this iteration, the parameter μ is fixed. For every μ, it can be found that a minimum value $F_\mu^*(\cdot)$, and the smallest μ brings the most accurate minimum value.

If $\boldsymbol{H}(F_\mu(\cdot))$ is irreversible, this work uses the Conjugate gradient method to complete the iteration with the optimization direction $\boldsymbol{r}^{(v)}$ and the optimization step size $\lambda^{(v)}$.

The initial direction is $r^{(1)} = -\nabla F_\mu^{(1)}(\cdot)$, and

the relationship between the $\boldsymbol{r}^{(v)}$ and the $\boldsymbol{r}^{(v-1)}$ is

$$\boldsymbol{r}^{(v)} = -\nabla F_{\mu}^{(v)}(\cdot) + \frac{(\nabla F_{\mu}^{(v)}(\cdot))^{\mathrm{T}} \nabla F_{\mu}^{(v)}(\cdot)}{(\nabla F_{\mu}^{(v-1)}(\cdot))^{\mathrm{T}} \nabla F_{\mu}^{(v-1)}(\cdot)} \boldsymbol{r}^{(v-1)} \tag{16}$$

The corresponding optimization step size $\lambda^{(v)}$ is

$$\lambda^{(v)} = -\frac{(\boldsymbol{r}^{(v)})^{\mathrm{T}} \nabla F_{\mu}^{(v)}(\cdot)}{(\boldsymbol{r}^{(v)})^{\mathrm{T}} A \boldsymbol{r}^{(v)}} \tag{17}$$

Then the relationship between iteration v and iteration $v-1$ is

$$[K^{(v)}, T^{(v)}, (\boldsymbol{f}^{(v)})^{\mathrm{T}}]^{\mathrm{T}} = [K^{(v-1)}, T^{(v-1)}, (\boldsymbol{f}^{(v-1)})^{\mathrm{T}}]^{\mathrm{T}} + \lambda^{(v-1)} \boldsymbol{r}^{(v-1)} \tag{18}$$

Repeat the calculation according to the above steps until the minimum value is found.

5 Simulation results and discussions

This work simulates the proposed scheme to verify its performance. First, it gives some simulation parameters, then the simulation results and analysis.

The simulation parameters are shown in Table 3.

Table 3 The simulation parameters

μ	γ	g	s	$\bar{f}_n$	β	θ
0.1	10^{-15}	0.7	5	200 M cycles	0.008 M cycles	0.0005 M cycles

Fig. 2 and Fig. 3 show the system throughput vs. energy consumption in the case of different nodes and different task complexity. First, it can be seen that the increased energy offers higher system throughput. Second, at the same cost of energy, the proposed scheme by allocating the computation resources adaptive to the task has higher throughput than the traditional scheme, where the resources at each node are distributed evenly to each step of the consensus without considering the actual requirements. Specifically, Fig. 2 shows the performance comparison when the number of nodes is $N = 7$, and the ratio α/α' is 10 or 100 with α' fixed to be 0.02 M cycles. The performance improvement of the proposed algorithm via optimized resources is higher when the ratio of calculation to verification is lower and the energy provided by the system is higher. Fig. 3 compares the throughput with the number of service nodes increased from 7 to 10 for α/α' fixed at 100. It shows that more participating nodes sharing the energy result in the reduced throughput, however, the proposed algorithm can still achieve much higher throughput compared to the scheme without optimization. This performance increase is big in the case of less nodes while it is a little bit lower with the number of nodes increasing.

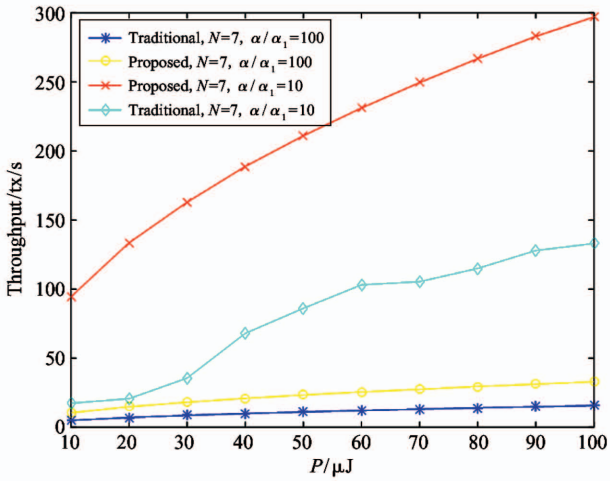

Fig. 2 The system throughput vs. energy ($N=7$, $\alpha/\alpha'=100$, 10)

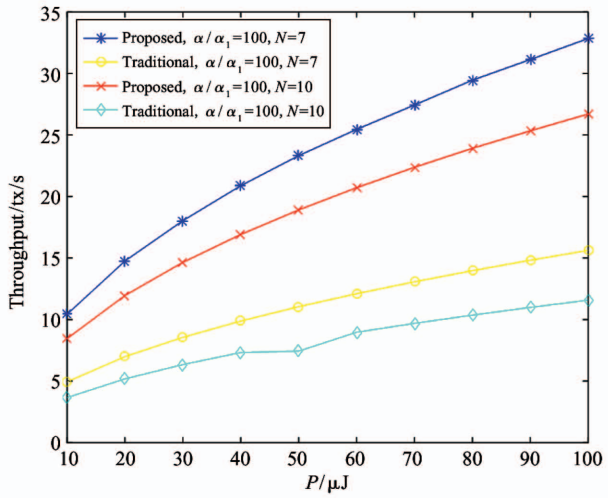

Fig. 3 The system throughput vs. energy ($\alpha/\alpha'=100$, $N=7$, 10)

Fig. 4 gives the system throughput vs. the ratio of calculation to verification, where the verification computation is set to be $\alpha' = 0.02$ M cycles. It can be seen

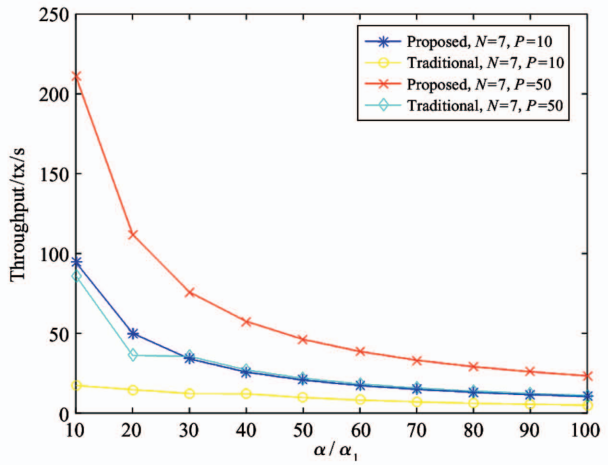

Fig. 4 The system throughput vs. the computation ratio of calculation to verification ($N=7$, $P=10$, 50)

that the throughput of the system gradually decreases along with the increase of α/α', which means the increase of the task complexity. Moreover, it is noticed that the improvement by optimizing is more outstanding when the energy of the system is relatively small and the calculation complexity is relatively low.

Fig. 5 verifies the system throughput vs. the number of nodes participating in the consensus. Since the energy per node decreases along with the increase of the number of nodes, the throughput decreases accordingly, however, the remarkable improvement of the proposed scheme remains. In addition, the increase of the system energy brings a large rise for the proposed scheme while the traditional scheme has little benefit.

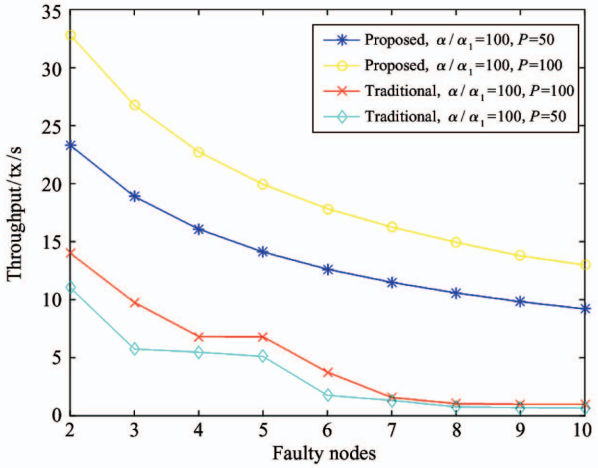

Fig. 5 The system throughput vs. the number of nodes ($\alpha/\alpha' = 100$, $P = 50, 100$)

6 Conclusions

To ensure the correctness of the outsourced computation, a verifiable computing scheme with optimization based on blockchain is proposed, which makes the calculation and verification respectively operated by the primary and replica nodes via the smart contract, reaching the consensus on the calculation results. Further, the complexity of each consensus step is analyzed, based on which the energy and computing resources are optimally allocated to maximize the system throughput. The simulation results show the effectiveness of the proposed scheme.

References

[1] Goldwasser S, Micali S, Rackoff C. The knowledge complexity of interactive proof systems[J]. *SIAM Journal on Computing*, 1989, 18(1): 186-208

[2] Babai L. Trading group theory for randomness[C]// Proceedings of the 17th Annual ACM Symposium on Theory of Computing, Rhode Island, USA, 1985: 421-429

[3] Arora S, Safra S. Probabilistic checking of proofs: a new characterization of NP[J]. *Journal of the ACM*, 1998, 45(1):70-122

[4] Craig G. A Fully Homomorphic Encryption Scheme[M]. San Francisco: Stanford University, 2009

[5] Li Q, Feng D, Zhang L, et al. Enhanced attribute-based authenticated key agreement protocol in the standard model [J]. *Chinese Journal of Computers*, 2013, 36(10):2156-2167

[6] Johnson R, Molnar D, Song D, et al. Homomorphic signature schemes[C]// Proceedings of CT-RSA, Berlin, Germany, 2002: 244-262

[7] Monrose F, Wycko P, Rubin A D. Distributed execution with remote audit[C]// Proceedings of Network and Distributed System Security Symposium, San Diego, USA, 1999: 3-5

[8] Canetti R, Riva B, Rothblum G N. Practical delegation of computation using multiple servers[C]// Proceedings of the 18th ACM Conference on Computer and Communications Security, Chicago, USA, 2011: 445-454

[9] Pham V, Khouzani M H R, Cid C. Optimal contracts for outsourced computation[C]// Proceedings of the Decision and Game Theory for Security, Los Angeles, USA, 2014:79-98

[10] Yue C, Tian Y, Zhang D, et al. Game-theoretic mechanism of rational outsourcing computation[J]. *Journal of Cryptologic Research*, 2019, 6(1):112-122

[11] Nakamoto S. Bitcoin: a peer-to-peer electronic cash system[EB/OL]. https://bitcoin.org/:Bitcoin, 2008

[12] Chen S, Di Q, Fan Q, et al. White Paper on China's Blockchain Industry in 2018[M]. Beijing: Information Center of MIIT, 2018:1-157

[13] Qiu C, Yu F R, Yao H, et al. Blockchain-based software-defined industrial Internet of Things: a dueling deep Q-learning approach[J]. *IEEE Internet of Things Journal*, 2019, 6(3): 4627-4639

[14] Yang R, Yu F R, Si P, et al. Integrated blockchain and edge computing systems: a survey, some research issues and challenges[J]. *IEEE Communications Surveys and Tutorials*, 2019, 21(2):1508-1532

[15] Luu L, Teutsch J, Kulkarni R, et al. Demystifying incentives in the consensus computer[C]// ACM SIGSAC Conference, New York, USA, 2015:706-719

[16] Teutsch J, Reitwießner C. A scalable verification solution for blockchains [J]. *arXiv*:1908.04756, 2017

[17] Hu S, Cai C, Wang Q, et al. Searching an encrypted cloud meets blockchain: a decentralized, reliable and fair realization[C]// Proceedings of the IEEE INFOCOM 2018-IEEE Conference on Computer Communications, Honolulu, USA, 2018:792-800

[18] Kumaresan R, Vaikuntanathan V, et al. Improvements to secure computation with penalties[C]// Proceedings of the ACM SIGSAC CCS'16, Vienna, Austria, 2016:406-417

[19] Gao H, Ma Z, Luo S, et al. BFR-MPC: a blockchain-based fair and robust multi-party computation scheme [J]. *IEEE Access*, 2019, (99):1-1

[20] Castro M, Liskov B. Practical Byzantine fault tolerance [C]// Proceedings of the 3rd Symposium on Operating Systems Design and Implementation, New Orleans, USA, 1999: 173-186

[21] Yin M, Malkhi D, Reiter M K, et al. HotStuff: BFT consensus in the lens of blockchain[J]. *arXiv*:1803.05069, 2018

[22] Mao Y, You C, Zhang J, et al. A survey on mobile edge computing: the communication perspective[J]. *IEEE Communications Surveys and Tutorials*, 2017, 19(4): 2322-2358

Yang Ruizhe, born in 1982. She received the Ph. D degree from the Beijing University of Posts and Telecommunications and joined the Beijing University of Technology in 2009. Her research interests include blockchain, resource management, and channel estimation.