Journal of Systems Architecture 135 (2023) 102807

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

Optimized CPU–GPU collaborative acceleration of zero-knowledge proof for confidential transactions

Ying Huang [a,b], Xiaoying Zheng [a,*], Yongxin Zhu [a,*]

[a] Shanghai Advanced Research Institute, Chinese Academy of Sciences, Shanghai 201210, China
[b] University of Chinese Academy of Sciences, Beijing 100049, China

ARTICLE INFO

Keywords:
Blockchain
Zero-knowledge proofs
Heterogeneous acceleration
Memory optimization

ABSTRACT

The potential application scope of blockchain is much broader than its origin of digital currencies, which may include managing smart contract, copyrighted works, and digitization of commercial or organizational registries. Not surprisingly, the confidentiality of transaction information is desired in these various blockchain applications, and is unfortunately not well supported yet. Zero-knowledge proofs such as *bulletproofs* are good candidate solutions to provide transaction confidentiality in blockchain, due to the ability to verify the truth of information without revealing the information directly. However, zero-knowledge proofs still suffer from high computation overhead and low throughput, and Bulletproofs are still computationally inefficient to be applied in blockchain applications. Edge computing meets confidential transactions' strong need for high performing by providing powerful GPUs and can be a helpful option. In this paper, we first present a CPU–GPU collaborative framework to accelerate the inner-product arguments of Bulletproofs. The experiments show that our implementation achieves an average speedup ratio of 3.7x. On this basis, this paper also proposes to build a high efficient bulletproofs based transaction processing system that supports both confidential and transparent transactions at GPU-enabled edge. Compared with the original bulletproofs version, a GPU accelerated implementation can obtain a speedup ratio up to 4.7x. When multiple bulletproofs are required to execute in bundle, the work focuses on memory optimization and data segmentation, which further increases the speed by 30%. Finally, the overall transaction processing system achieves a 1.5x speedup when both confidential and transparent transactions are parallelized at the edge.

1. Introduction

Blockchain technology [1] has attracted great attention both from the industry and academia due to its unique feature of data immutability. It enables a community of users to record transactions in a shared ledger in a decentralized fashion without an authority, which cannot be changed easily after they are created [2,3]. However, in blockchain, transactions are globally published and are not encrypted in most blockchain applications which are also referred as *transparent transactions*, and there is raised concern of lacking privacy. Data providers often want to keep the sensitive data private, but the transactions on the public chain are open and transparent. This fact causes a hindrance in practice if we wish to apply blockchain in transactions involving privacy-sensitive information. Insufficient financial privacy [4] can have serious security and privacy implications for both commercial and personal transactions [5].

A *confidential transaction* (CT) is a cryptographic method of increasing the privacy and security of blockchains by keeping the transaction information private while preserving the ability of the public network to verify the ledger entries. It conducts a zero-knowledge proof for each piece of information that needs to be verified in the transaction to ensure that the information is legal and can be verified while not disclosing the content of the information [6].

For instance, assume that the two parties to the transaction are A and B, and A has two accounts at the beginning while B has one account with no balance. If A wants to transfer 10-BTC to B now, the transaction without the confidential mechanism is shown at the top of Fig. 1. It can be seen that after A transfers money from account 1, 2 to B, the balance of the original accounts 1, 2 is cleared, and then there is a new account 3 to store the balance after the transaction. As there is no confidential mechanism applied, all balance details are public.

Counter intuitively, the zero-knowledge proof technology can ensure that a valid transaction has taken place without disclosing any information about the account balance. The bottom of Fig. 1 shows the

* Corresponding author.
E-mail addresses: zhengxy@sari.ac.cn (X. Zheng), zhuyongxin@sari.ac.cn (Y. Zhu).

https://doi.org/10.1016/j.sysarc.2022.102807
Received 10 July 2022; Received in revised form 6 November 2022; Accepted 12 December 2022
Available online 15 December 2022

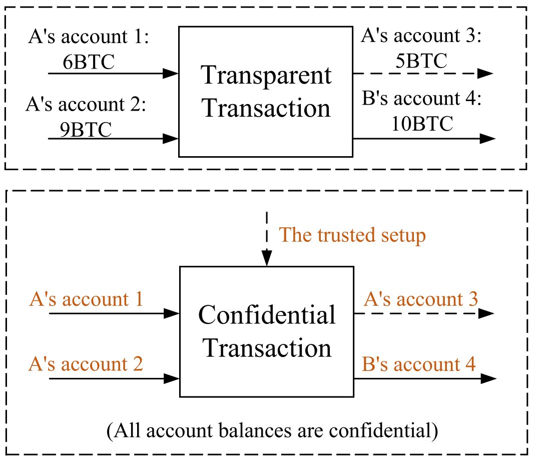

Fig. 1. The transparent transaction (top) and the confidential transaction (bottom) mechanism.

transaction flow after the zero-knowledge proof technology is applied, where it verifies that

- Proof 1: the balance of accounts 3 is no less than 0;
- Proof 2: the balance of accounts 4 is no less than 0;
- Proof 3: the sum of the balances of accounts 1 and 2 is no less than the sum of the balances of accounts 3 and 4.

With the above three proofs, the BTC transferring transaction can be committed without revealing the details of the remaining balance.

Zero-knowledge algorithm was originally proposed in the area of cryptography, and it has been widely studied in recent years due to its importance in the field of blockchain. In 2013, Sasson et al. proposed the zk-SNARK algorithm [7], which is now employed by the cryptocurrency Zcash to solve a perceived anonymity problem with Bitcoin-type blockchains. Zk-SNARK proofs require a small amount of calculation and are fast, but rely on an initial “trust system” setup that has been critiqued as an inherent security flaw. Trusted setup refers to the need to set and generate some public parameters before performing protocol proof and authentication. In 2014, Sasson et al. proposed the zk-STARK algorithm [8], which no longer requires a trusted third party, but at the same time, the amount of calculation and the size of the proof file are significantly increased. Bulletproofs was proposed in 2018 [9], and it also does not require a trusted third party. At the same time, it converts many complex calculations into inner product operations, which effectively reduces the amount of calculation and the size of the proof file. What is more, the size of the proof produced by the bulletproofs is logarithmically transformed, that means the size of the proof only becomes a little larger with the sharp increase of transactions. This is beneficial to its application in big data scenarios. In a word, Bulletproofs does not require a trusted setup, and has less computation and has a wider range of applications than zk-STARK.

However, though Bulletproofs is the most efficient zero-knowledge proofs, it is still costly in computing. As confidential transactions usually involve a huge amount of proofs in application, the high computational overhead makes it difficult for confidential transactions to be widely used in practice. Therefore, confidential transactions need to be optimized in parallel to improve execution speed.

Fortunately, edge computing can be a helpful option in such a case. Edge computing [10] often equips with powerful GPUs and capable processors due to their low-latency form and considerably higher bandwidth to satisfy infrastructure applications demand [11]. It meets confidential transactions’ strong need for high performing infrastructure and unique specs for growth, security and decentralization and provides a seamless way to deploy and maintain a geographically diverse presence. In order to enable users to access the systems more reliably, confidential transaction can be used to form interconnection among the participants in Edge Computing systems, so as to realize credit transfer [12], data synchronization and resource sharing [13]. In addition, the data consistency and tamper-proof characteristics of the blockchain can also ensure the integrity of the digital information in systems [14]. Thus, with advancements in both technology, the improved integration of zero-knowledge proofs and edge computing will be a promising solution for confidential transaction systems.

We first propose to accelerate the Bulletproofs algorithm by fully unlocking the power of CPU–GPU systems to parallelize inner product computation. Then, considering the architectural limitation of GPU, we propose to optimize a security system supporting both confidential transactions and transparent transactions, by fully unlocking the power of edge computing to parallelize batches of cryptographic operations including both the bulletproofs and the digital signatures. We first present the common zero-knowledge proofs based transaction process architecture in edge computing for typical applications. We then implement an integrated optimization mechanism including zero-copy memory management, data lookup, data segmentation, and pipelining that is able to further increase the performance. Finally, the security system are tested in a GPU-enabled edge environment and different combinations of individual optimization techniques are evaluated. The overall transaction processing system has achieved a speedup ratio close to 1.5x, in the meanwhile information security is also provided.

We summarize the contribution as follows.

- Accelerate the verifying Bulletproofs algorithm by fully parallelizing inner product computation in CPU–GPU systems and scales down the execution time by 3–4x.
- Optimize the acceleration model of zero-knowledge proof, and improve the operation speed of zero-knowledge proof by 30% through memory optimization and data segmentation methods.
- Propose a security model supporting both confidential and transparent transactions, which implements a bunch of parallel zero-knowledge proofs acceleration.
- Reduce computation time while maintaining data confidentiality. The speed of the optimized zero-knowledge proof algorithm is 4.6x faster than the original Bulletproofs algorithm implementation, and the speed of the overall model is improved by nearly 1.5x.

The rest of the paper is organized as follows. Section 2 discusses potential application prospects and elaborates more related works about zero-knowledge proof algorithm. Section 3 provides basic concepts and general information about inner-product argument and elliptic curve point arithmetic in Bulletproofs. In Section 4, the designed CPU–GPU collaborative algorithm is described, and a transaction processing system based on zero-knowledge proofs is presented, which supporting both confidential and transparent transactions. Then, we describe the optimization mechanism based on three methods in Section 5. Section 6 discusses the detailed implementation of the system and each optimization technique. The system and techniques are evaluated in Section 7 while the conclusion is drawn in Section 8.

2. Related works

Major privacy protection solutions can be categorized into five types including mixed coins, multi-party secure computing, ring signature, trusted computing, and zero-knowledge proofs. Mixed coins complete user privacy protection through simple obfuscation operations [15]. Ring signature is a digital signature scheme, and its main principle is to hide the user who initiated the behavior in a group for privacy protection [16]. These two methods are in fact partial privacy protection, and are unable to prevent the attack of malicious actors mixed in the group. The process of multi-party secure computing is relatively complex [17], which makes it perform poor, difficult to implement, and difficult to support large-scale applications. Solutions based on trusted computing can take into account correctness, privacy and high performance through hardware [18], but the design and manufacturing costs of hardware are high, and therefore it is difficult to deploy. Zero-knowledge proofs algorithm means that the prover can make the verifier believe that a certain statement is correct without providing any useful information to the verifier. Compared with other algorithms, it provides complete privacy protection and is relatively simple to implement, making it suitable for large-scale applications.

Since zero-knowledge proof algorithms can prove that information is valid without revealing the information, it has the potential to be widely used in many privacy-critical applications to enable secure and verifiable data processing. The characteristics of zero-knowledge proof can lead to some interesting applications, including electronic voting [19], online auction [20], anonymous credentials [21], etc. There are also many interesting applications in the context of distributed ledger technology and blockchain technology, such as Pedersen commitments [22], provisions [23], and various smart contracts on blockchains [24]. Meanwhile, zero-knowledge proof can also be combined with other advanced technologies, which contain verifiable database outsourcing [25] and verifiable machine learning [26], etc.

Zero-knowledge proofs can also be applied in large scientific installations. Shanghai Synchrotron Radiation Facility (SSRF) is the scientific facility with the largest number of users and scientific achievements in China [27]. As the raw data from the SSRF might be traded on the copyright market, the bulletproofs based confidential transaction system is regarded as a candidate solution for the data copyright transactions.

Although Bulletproofs does not need a trusted setup and it is faster than the ZK-STARK algorithm, it still has the problem of large amount of computation and high throughput in practical applications. Therefore, the heterogeneous acceleration framework can be used to accelerate the inner product operation and modular multiplication operation in parallel to improve the operation speed of the algorithm.

Heterogeneous acceleration generally uses the CPU with coprocessor structure. According to the different coprocessors in the zero-knowledge proof heterogeneous acceleration, existing research can be divided into: CPU with dedicated accelerator, CPU with FPGA, CPU with GPU. Dedicated accelerators refer to specific chips designed for fixed scenarios, such as the 96-core blockchain dedicated accelerator chip released by ChainMaker [28], which can increase the speed of digital signatures by 20 times. But the disadvantage of this method includes being difficult to implement, high cost, and poor versatility. In the acceleration of blockchain, accelerating with FPGA is a relatively common research scheme, and the implementation methods are similar, such as accelerating the FFT operation, Montgomery multiplication [29] and signature algorithm [30]. GPU is also a commonly used device in heterogeneous acceleration [31]. For instance, Zhang et al. designed a customized pipelined accelerator with two subsystems to handle the two intensive compute tasks [32]. Compared with FPGA, its advantage is that GPU environment often already exists in zero-knowledge proof scenarios. Therefore, no additional environment configuration is required, and CPU–GPU heterogeneous approach is relatively simple to implement.

Common strategies for GPU parallel optimization include memory optimization, transmission process optimization, and structure optimization. The most important thing about memory optimization is to make reasonable use of all levels of memory, especially cache, shared memory and constant memory [3]. The optimization of transmission is actually to reduce the data exchange between memory and video memory, and it also needs to rely on memory scheduling. Ren et al. proposed the Sentinel architecture, which mainly uses memory allocation to obtain the size and cycle of tensors, and puts hot tensors in shared memory as much as possible [33]. Structure optimization refers to making appropriate changes to the algorithm structure to make it more suitable for acceleration in GPUs, such as loop unrolling or graph unrolling. For example, Morishima [34] et al. proposed the subgraph structure which is suitable for GPU processing so that all of the subgraph making, the feature extraction, and the anomaly detection are performed in GPU. To sum up, the parallel optimization of GPU needs to choose appropriate solutions according to the characteristics of the algorithm and the scene.

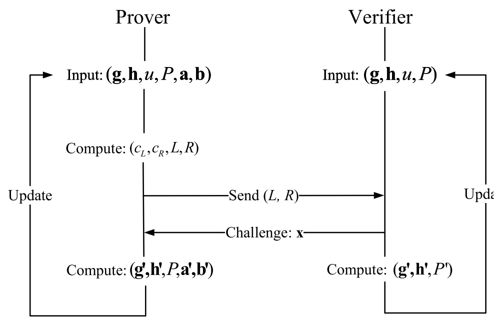
Prover
Verifier
Input: $(\mathbf{g},\mathbf{h},u,P,\mathbf{a},\mathbf{b})$
Input: $(\mathbf{g},\mathbf{h},u,P)$
Compute: (c_L,c_R,L,R)
Send (L, R)
Challenge: x
Update
Update
Compute: $(\mathbf{g}',\mathbf{h}',P,\mathbf{a}',\mathbf{b}')$
Compute: $(\mathbf{g}',\mathbf{h}',P')$

Fig. 2. The procedure of inner-product argument in Bulletproofs [42].

3. Background

The current acceleration schemes in the field of zero-knowledge proofs are all accelerating the zk-snark algorithm and the zk-stark algorithm [35], and most of them are based on FPGA to achieve acceleration [36] or by changing the hardware structure, such as adding pipeline accelerators to achieve hardware acceleration [32]. As for GPU acceleration, the blockchain project Filecoin uses GPU acceleration to shorten the PoST proof time in zk-snark, and Long et al. [37] proposed a CPU–GPU systems to accelerate the computation tasks of IPR solution in matrix factorization. Privacy protection of cryptography is currently a hot research topic. Qiu et al. [38] proposed a novel privacy-preserving method using bipartite matching and attracted much follow-up work. Dai et al. [39] presented some important research on privacy protection in smartphones.

Bulletproofs present a new zero-knowledge argument to prove that a secret committed value lies in a given interval relying only on the discrete logarithm assumption. An argument is a proof that holds only if the prover is computationally bounded and certain computational hardness assumptions hold. An inner-product argument proves that an inner product relation holds between committed vectors [40]. Bulletproofs build on the communication efficient inner-product argument proposed by Bootle et al. [41]. The argument proves that the prover knows the openings of two binding Pedersen vector commitments that satisfy a given inner product relation.

3.1. Inner-product argument

Fig. 2 describes the overall interactive procedure of inner-product argument in Bulletproofs. There is a prover and a verifier in Bulletproofs. As shown in Fig. 2, the verifier repeatedly sends a random challenge x to the prover and the prover will compute two variables L, R according to the received x and feedback L, R to the verifier.

We first introduce the notations used in the interaction between the prover and the verifier. Let $\mathbf{g},\mathbf{h} \in \mathbb{G}^n$ denote two independent generators, and $u, P \in \mathbb{G}$ denote the binding vector commitments. The variable $c \in \mathbb{Z}_p$ is a scalar. In the inner-product argument, the prover proves that there exists two vectors $\mathbf{a},\mathbf{b} \in \mathbb{Z}_p^n$ that satisfy $P = \mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}$ and $c = \langle \mathbf{a},\mathbf{b}\rangle$, i.e.,

$$\begin{aligned}&(\mathbf{g},\mathbf{h} \in \mathbb{G}^n, P \in \mathbb{G}, c \in \mathbb{Z}_p; \mathbf{a},\mathbf{b} \in \mathbb{Z}_p^n) : \\ &P = \mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}} \wedge c = \langle \mathbf{a},\mathbf{b}\rangle\end{aligned} \tag{1}$$

Let us further restrict P to be a Pedersen's commitment of the vector $(\mathbf{a}, \mathbf{b}, c)$. The formula (1) is modified as follows.

$$(\mathbf{g}, \mathbf{h} \in \mathbb{G}^n, P \in \mathbb{G}, c \in \mathbb{Z}_p; \mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n) : \quad P = \mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}u^{\langle \mathbf{a},\mathbf{b}\rangle} \tag{2}$$

Initially, when $n > 1$, the input parameters of the prover are $(\mathbf{g}, \mathbf{h}, u, P, \mathbf{a}, \mathbf{b})$ and set $n' = \frac{n}{2}$, then P can be rewritten as $P = H(\mathbf{a}_{[:n']}, \mathbf{a}_{[n':]}, \mathbf{b}_{[:n']}, \mathbf{b}_{[n':]}, c)$. In order to ensure that the verifier can calculate a new Pedersen commitment, the prover needs to provide the verifier with the L and R, which are defined as follows.

$$\begin{aligned} c_L &= \langle \mathbf{a}_{[:n']}, \mathbf{b}_{[n':]} \rangle \in {}_p \\ c_R &= \langle \mathbf{a}_{[n':]}, \mathbf{b}_{[:n']} \rangle \in {}_p \\ L &= \mathbf{g}_{[n':]}^{\mathbf{a}_{[:n']}} \mathbf{h}_{[:n']}^{\mathbf{b}_{[n':]}} u^{c_L} \in \mathbb{G} \\ R &= \mathbf{g}_{[:n']}^{\mathbf{a}_{[n':]}} \mathbf{h}_{[n':]}^{\mathbf{b}_{[:n']}} u^{c_R} \in \mathbb{G} \end{aligned} \tag{3}$$

Subsequently, the verifier generates and sends a random challenge $x \overset{\$}{\leftarrow} \mathbb{Z}_p^*$ to the prover, and the prover need to compute

$$\begin{aligned} \mathbf{a}' &= \mathbf{a}_{[:n']} \cdot x + \mathbf{a}_{[n':]} \cdot x^{-1} \in \mathbb{Z}_{n'}^p \\ \mathbf{b}' &= \mathbf{b}_{[:n']} \cdot x^{-1} + \mathbf{b}_{[n':]} \cdot x \in \mathbb{Z}_{n'}^p \\ \mathbf{g}' &= \mathbf{g}_{[:n']}^{x^{-1}} \circ \mathbf{g}_{[n':]}^{x} \in \mathbb{G}^{n'} \\ \mathbf{h}' &= \mathbf{h}_{[:n']}^{x} \circ \mathbf{h}_{[n':]}^{x^{-1}} \in \mathbb{G}^{n'} \\ P' &= L^{x^2} P R^{x^{-2}} \in \mathbb{G} \end{aligned} \tag{4}$$

and update the input $(\mathbf{g}', \mathbf{h}', P', \mathbf{a}', \mathbf{b}')$. In the meanwhile, after receiving (L, R), the verifier computes $\mathbf{g}', \mathbf{h}', P'$ and updates the input. The interactive procedure is repeated until $n' = 1$. Finally, the verifier compares P' with $H(x^{-1}\mathbf{a}', x\mathbf{a}', x\mathbf{b}', x^{-1}\mathbf{b}', \langle \mathbf{a}', \mathbf{b}' \rangle)$. If the two values are equal, the verification is considered to be successful.

There are many computation steps of the inner-product argument as in Eqs. (3)–(4) that can be performed in parallel. Furthermore, as Bulletproofs usually require a large number of inner-product arguments, the performance of Bulletproofs will be greatly improved if the inner-product argument is effectively accelerated.

3.2. Elliptic curve point arithmetic

In Bulletproofs, the implementation of the inner-product argument uses secure Elliptic Curve Cryptography (ECC), which is based on the algebraic structure of elliptic curves over finite (prime) fields. Thus, the elliptic curve points instead of numbers are used as basic mathematical objects such as addition and multiplication.

An elliptic curve is a set of points defined by $y^2 = x^3 + ax + b$ and a, b satisfying $4a^3 + 27b^2 \neq 0$, which ensures the curve does not contain singularities. Note that the elliptic curve is symmetric with respect to the x-axis. The elliptic curve point addition is shown in Figs. 3 and 4.

Let us assume a straight line and the elliptic curve intersect at points (P, Q, R), and define $P + Q + R = 0$ (0 represents the identity element, which is a point at infinity). The three points (P, Q, R) form an elliptic curve group. The elliptic curve point addition is regarded as solving R given the two points P and Q, i.e., $P + Q = -R$.

Let $\{(x_i, y_i), i \in P, Q, R\}$ represent the coordinates of P, Q, R, respectively. Assume a real field, when $P \neq Q$, the slope of the line λ is as follows.

$$\lambda = \frac{y_Q - y_P}{x_Q - x_P}, P \neq Q. \tag{5}$$

When $P = Q$, the slope λ is calculated as

$$\lambda = \frac{3x_P^2 + a}{2y_P}, P = Q. \tag{6}$$

Then the coordinates of R can be calculated as

$$x_R = \lambda^2 - x_P - x_Q, y_R = \lambda(x_P - x_R) - y_P. \tag{7}$$

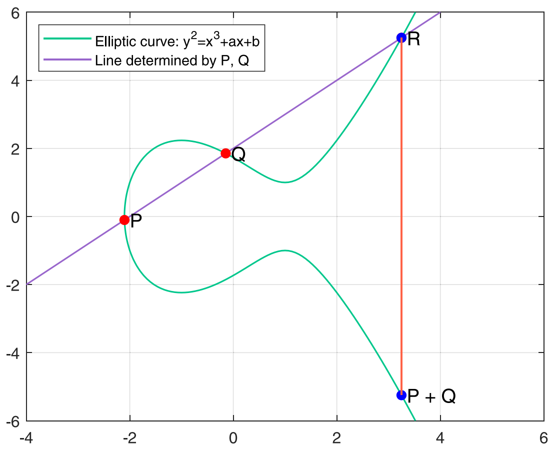

Fig. 3. Elliptic curve point addition ($P \neq Q$).

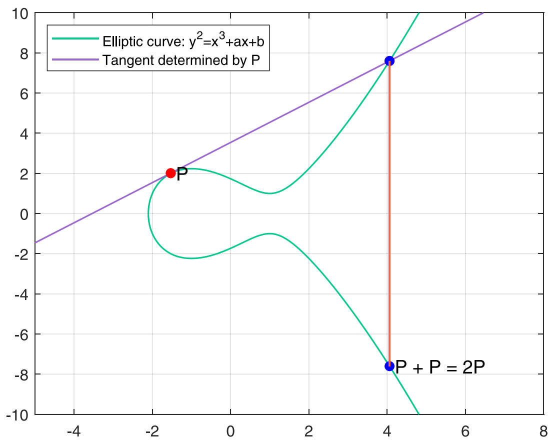

Fig. 4. Elliptic curve point addition ($P = Q$).

However, the elliptic curve is defined in a finite field F_p instead of the real field in encryption. The finite field arithmetic follows the modulus rules, thus the slope of the line and the coordinates of R are calculated as follows, where p is a prime number.

$$\begin{aligned} \lambda &= \frac{y_Q - y_P}{x_Q - x_P} \pmod p, P \neq Q \\ \lambda &= \frac{3x_P^2 + a}{2y_P} \pmod p, P = Q \\ &\begin{cases} x_R = \lambda^2 - x_P - x_Q \pmod p \\ y_R = \lambda(x_P - x_R) - y_P \pmod p \end{cases} \end{aligned} \tag{8}$$

Finally, given the coordinates of R, the elliptic curve point addition operation $P + Q = -R$ can be obtained by finding the symmetric point of R, i.e., $(x_R, -y_R)$.

The elliptic curve point multiplication can be implemented by repeatedly applying addition as $nP = P + P + \cdots + P$.

In the inner product proof and verification of Bulletproofs, there is a large amount of elliptic curve point addition and multiplication operations that can be parallelized. It will effectively improve the speed of inner-product arguments in Bulletproofs.

4. System description

The first is the acceleration algorithm design based on Bulletproofs. In Bulletproofs, the most significant parts are the circuit proof and rangeproof, which are both implemented based on inner product arguments. The procedure of the circuit proof and rangeproof contains

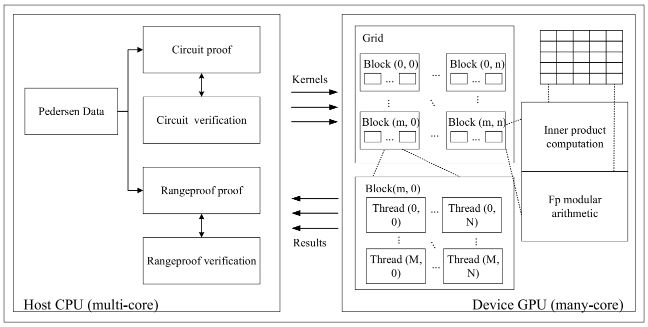

Fig. 5. The CPU–GPU collaborative operation framework.

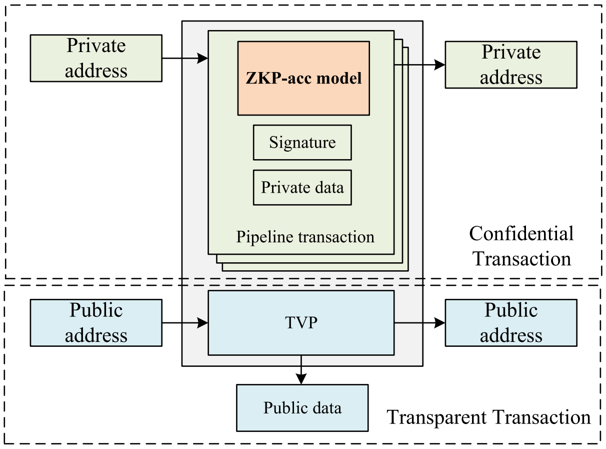

Fig. 6. Zero-knowledge proofs based security system architecture.

a large number of vector inner product operations and modular operations based on elliptic curves in finite fields, that can be accelerated by parallel computing. The framework of the entire CPU–GPU collaborative acceleration is shown in Fig. 5.

In Fig. 5, there are two computing components, i.e., the CPU host and the GPU device. The Bulletproofs circuit and rangeproof module are initialized in the host, which includes data input, memory allocation, and parameter setup for the proof and verification stages. When the Bulletproofs algorithm steps into the parallel calculation part of the inner-product and elliptic modular operations, the host copies and transmits the data and kernels that need to be calculated to the GPU device for parallel execution based on the CUDA platform [43,44]. After the calculation is completed, the result will not be transmitted back to the host immediately but temporarily stored in the global memory of the GPU until the result needs to be used by the host.

In Fig. 6, we present a common zero-knowledge proofs based security system architecture for typical applications such as scientific data sharing and tracing. In real applications, there usually exist two kinds of transactions, i.e., the confidential transactions and the transparent transactions. In confidential transactions, data are kept disclosed during the entire procedure; in the meanwhile, data are made public during transparent transactions. As described in Fig. 6, the system is split into two components. The first carries confidential transactions, and the second supports transparent transactions. In real applications, a bunch of transactions stream into the system and are executed concurrently. Therefore multiple zero-knowledge proofs instead of a single proof are implemented in parallel in the system. The high system parallelism needs to be explored to further improve the performance.

4.1. Confidential transactions

In confidential transactions, zero-knowledge proofs are used to prove that transactions can be legally concluded without disclosing the specific transaction contents. Furthermore, multiple zero-knowledge proofs are applied in parallel at different functional units of confidential transactions, such as data verification and address verification. In order to facilitate parallelism in execution of multiple proofs and increase the performance, the confidential transactions are pipelined with four stages. The four stages are: transaction construction, zero-knowledge proof computation, signature and transaction sending.

- Transaction construction: transaction construction is a procedure of defining transaction data structures, assigning values, initializing the input and output queues of the transaction, and creating objects according to the transaction addresses and adding them to the address queue.
- Zero-knowledge proof computation: when the transaction construction is finished, the transaction process is verified by the zero-knowledge proofs. For each transaction, multiple zero-knowledge proofs are required for different components of information. For instance, these information including (1) the ownership of the address, (2) whether the calculation process of the transaction is correct, (3) whether there is a conflict in the transaction itself are verified by zero-knowledge proofs independently. After all components have been verified as legitimate, the transaction is signed and sent to the destination.
- Signature: signature is a procedure of attaching additional data to the data unit or the cryptographic transformation made on the data unit. The additional data attached are effective proofs of the data authenticity. In addition to ensure immutability, the data privacy is protected. Therefore, there are two pairs of signature keys required. The first pair of keys is used to ensure immutability. More specifically, the private key encrypts the data to ensure that the data cannot be tampered with, and then the signed data needs to be verified by the public key. The second pair of symmetric keys is used to encrypt private data.
- Transaction sending: the verified and signed transaction is sent to the destination.

4.2. Transparent transactions

Transparent transactions are the peripheral supplement of confidential transactions and help to complete the transaction of public data. In transparent transactions, all information is publicly traceable, which is more concise than confidential transactions. Therefore, many encryption processes can be omitted. The transparent transaction process is pipelined with three stages including transaction construction, signature and transaction sending. Compared with confidential transactions, zero-knowledge proof is not required in transparent transactions. The data structures required in transaction construction are also relatively simple. In Fig. 6, a transparent value pool (TVP) is used to store public data and addresses in transparent transactions.

5. Optimization mechanism for data fetching

Generally, the architecture of a GPU is very similar to that of a CPU. They both make use of memory constructs of cache layers, global memory and memory controller. A high-level GPU architecture is all about data-parallel throughput computations and putting available cores to work instead of focusing on low latency cache memory access like a CPU. Fig. 7 presents a typical GPU memory architecture.

GPU devices have several different memory spaces: Global, local, texture, constant, shared and register memory. Each type of memory has its advantages and disadvantages. The only two types of on-chip memory are register and shared memory. Local, Global, Constant, and Texture memory all reside off chip.

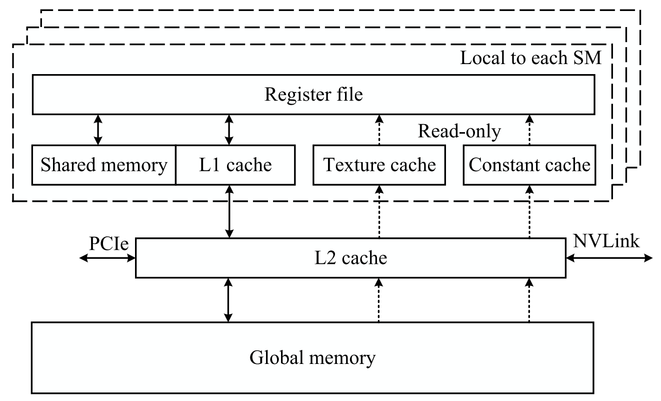

Fig. 7. GPU memory architecture.

- Register memory: registers are local to a thread, and each thread has exclusive access to its own registers.
- Local memory: it has the same scope rules as register memory, but performs slower.
- Shared memory: data stored in shared memory are visible to all threads within that block and lasts for the duration of the block, which allows for threads to communicate and share data between one another.
- Global memory: global memory is allocated and managed by the host, and it is accessible to both the host and the GPU. It is the largest memory space available, and is also slower to access.
- Constant memory: constant memory is used for data that is read only. Compared with global memory, constant memory can reduce the required memory bandwidth.
- Texture memory: texture memory is another variety of read-only memory on the device. When all reads in a warp are physically adjacent, using texture memory can reduce memory traffic and increase performance compared to global memory.

As a side note, the edge devices such as GPUs and FPGAs have different computing power and memory resources. The proposed memory access optimization techniques including zero-copy memory access,[1] data lookup table, data segmentation, and pipelining generally fit various GPUs and FPGAs, help reduce the number of memory accesses required, and eventually reduce energy overhead at the edge.

5.1. Zero-copy memory

Data transfers between the host and the GPU device often lead to idle computing resources and are the performance bottleneck. An asynchronous transfer version enables overlap of data transfers with computation, which can be used to hide data transfers between the host and the device. Zero-copy memory is an asynchronous memory-mapping method that maps pinned (non-pageable) host memory directly to GPU memory and implicitly transfers it to the GPU. It enables GPU threads to directly access host memory. When a GPU thread reads a host-mapped variable, it submits a PCIe read transaction, and the host will return the data over the PCIe bus. Compared with other cudaMemcpy methods, zero-copy memory allows to overlap CPU–GPU memory transfers with computation.

Fig. 8 shows the zero-copy memory procedure applied in the zero-knowledge proofs. When a GPU thread reads data, it sends a signal in advance and continues to do computation; and the data will be mapped

[1] The zero-memory access is naturally supported by FPGAs without additional codes.

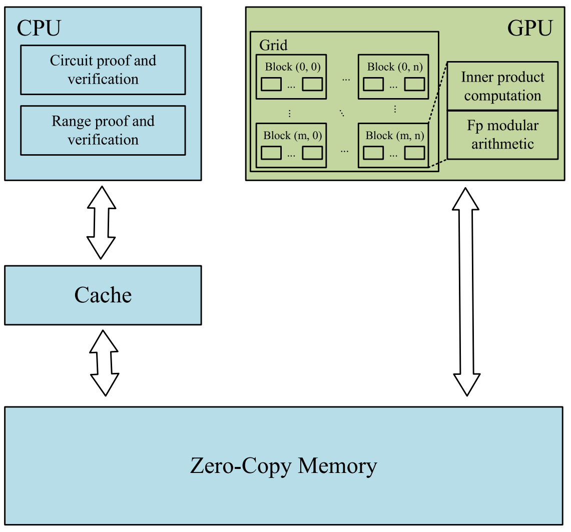

Fig. 8. Schematic of zero-copy memory.

from CPU memory to the zero-copy memory in the background. Eventually, the GPU thread reads the data directly from the zero-copy memory instead of the GPU cache. With suitable workloads where the GPU can overlap computation with PCIe transfers, higher performance may be achieved as well as obviating the need to allocate a GPU memory buffer.

However, there are a few things to consider. First, since the mapped buffer is shared by the CPU and GPU, developers must synchronize access using existing context, stream or event APIs. Synchronization typically is needed when the application wants to read data written to a mapped pinned buffer by the GPU. If the application is using multi-buffering to maximize concurrent CPU/GPU execution, it may have to synchronize before starting to write to a buffer that may still be getting read by the GPU. Secondly, because the data is not cached on the GPU, mapped pinned memory should be read or written only once, and the global loads and stores that read and write the memory should be coalesced. Since any repeated access to such memory areas causes repeated CPU–GPU transfers and there are differences in the number of memory accesses of data in our model, it is necessary to use zero-copy memory for data that are accessed only once in order to effectively increase the transfer rate and effectively utilize the GPU memory structure. For data that need to be accessed repeatedly, other solutions are required.

5.2. Lookup table

In zero-knowledge proofs, some data are accessed repeatedly and cannot be transferred by the zero-copy memory method. A lookup table is proposed to store the data that are accessed repeatedly. A lookup table is a collection of data of the same type and can support operations of searching, reading, inserting and deleting data in the table. There are two types of lookup table, static and dynamic, respectively. The static lookup table only supports searching and reading, and the dynamic lookup table supports all the four operations. In practice, there are many cache misses in zero-knowledge proofs, therefore a read-only static lookup table performs badly.

Therefore, a dynamic lookup table is adopted in the zero-knowledge proofs as shown in the algorithm flow chart in Fig. 9. Specifically, a lookup table is built in the shared memory of the GPU since the shared memory is large and fast when performs writing. During initialization, the data that may be frequently accessed in the early stage are placed in the lookup table implemented as an array. When the GPU performs calculation, it first queries the data from the lookup table. If the query

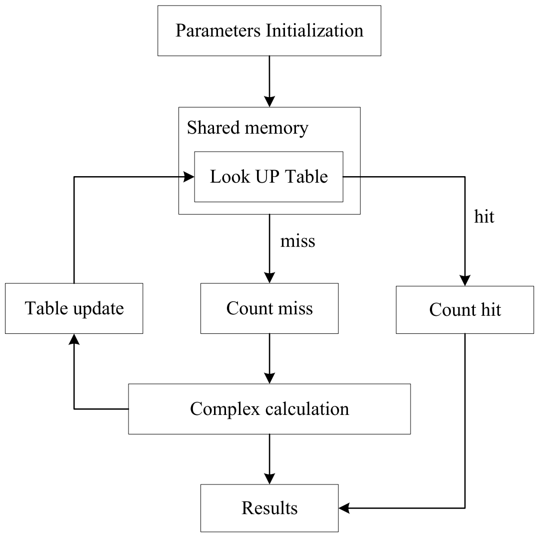

Fig. 9. LUT flow chart.

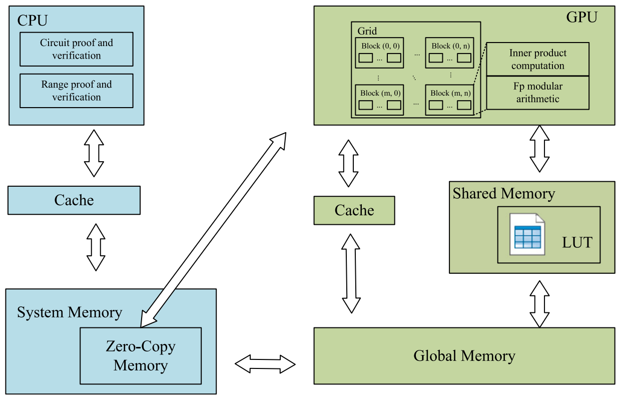

Fig. 10. Memory optimization based on zero-copy memory and LUT (lookup table).

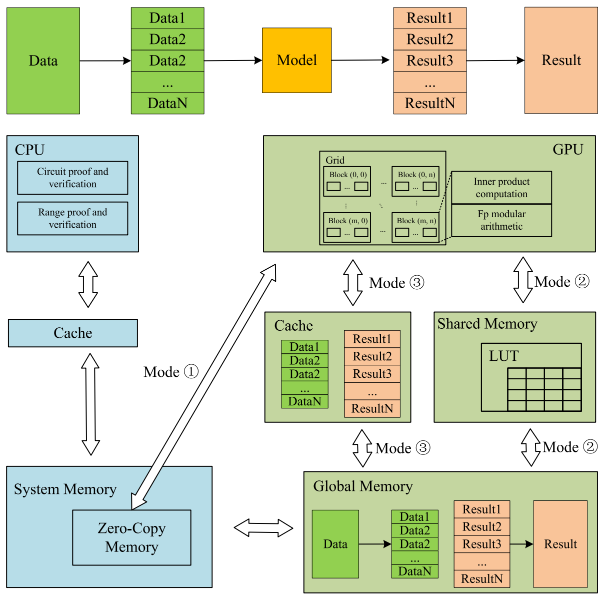

Fig. 11. An integrated approach for data fetching. There are three data fetching modes. Mode 1: zero-copy memory, Mode 2: LUT, and Mode 3: data segmentation.

Table 1
Specifications of CPU and GPU.

CPU: Intel® Xeon® Gold 6230		GPU: NVIDIA® RTX 2080 Ti	
Num of cores	20	GPU Architecture	Turing
Num of threads	40	RTX-OPS/T	76
Max Frequency/GHz	3.9	Boost Clock/MHz	1545
Basic Freq/GHz	2.1	Mem Bitwidth/bit	352
Cache size/MB	27.5	Shared Mem size/KB	64
Max Mem size/TB	1	Global Mem size/GB	11
Mem speed/MHz	2933	Mem speed/Gbps	14

is missed, the data will be fetched from the host and the lookup table will be updated if needed. At the beginning, updates and replacements will be more frequent. When the data is stable, the number of updates will decrease. At this time, a binary tree will replace the initial array for fast query and update. Fig. 10 shows the data fetching procedure based on the combined lookup table and zero-copy memory access approach as we described. By the combined approach, the data exchange cost between the host and the device is significantly reduced.

5.3. Data segmentation

When the bitwidth of input is larger than the bitwidth of the GPU cache, it will cause a short-term block and increase the transmission latency. Different GPU devices have different bitwidth. For instance, Nvidia 2080 Ti's bitwidth is 352 bits and V100's bitwidth is 4096 bits, respectively. A data segmentation method is applied to process the high-bit-width data and GPU bitwidth is used as a threshold to determine whether a data block is large-size or not for slicing. In Fig. 11, the process of data segmentation is presented. More specifically, in data segmentation mode, data are partitioned into multiple chunks before entering the pipeline; and after processing in GPU, these segments are assembled again.

5.4. An integrated approach for data fetching

Combining the three optimized data fetching modes in addition to the ordinary host-device transmission method, the zero-knowledge proofs algorithm has an integrated data fetching approach as described in Fig. 11.

- Mode 1: Data that are small-sized and accessed only once are stored in zero-copy memory. A GPU thread will directly access these data if needed.
- Mode 2: Data that are accessed repeatedly are stored in a preprocessed lookup table in the shared memory. A GPU thread will first search the lookup table. If there is a cache miss, the GPU thread reads the memory and updates the lookup table if necessary.
- Mode 3: For large-sized data, data are partitioned before they are fetched into the GPU memory.

6. Implementation

In this section, we describe the implementation details of the overall framework. Table 1 presents the specifications of CPU and GPU used in the experiments. The system version is Ubuntu 18.04, and the toolchain version is CUDA 10.1, respectively.

6.1. Single ZKP-acc model

Algorithm 1 Inner-product arguments acceleration

```
Input: $\mathbf{g}, \mathbf{h} \in \mathbb{G}^n, u, P \in \mathbb{G}, \mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n$;
  Prover's input: $(\mathbf{g}, \mathbf{h}, u, P, \mathbf{a}, \mathbf{b})$;
  Verifier's input: $(\mathbf{g}, \mathbf{h}, u, P)$;
Output: Verifier accepts or rejects;
  if $n = 1$ then
    Prover → Verifier: $a, b \in \mathbb{Z}_p$;
    Verifier: computes $c = a \cdot b$;
    Verifier checks:
    if $P = g^a h^b u^c$ then
      return accepts;
    else
      return rejects;
    end if
  else if $n > 1$ then
    Prover computes:
        $n' = \frac{n}{2}$
        Host → Device: $\mathbf{a}, \mathbf{b}, n'$, Device computes: $c_L, c_R$;
        Host → Device: $\mathbf{g}, \mathbf{h}, u$, Device computes: $L, R$;
        Device → Host: $L, R$;
    Prover → Verifier: $L, R$;
    Verifier → Prover: $x \overset{\$}{\leftarrow} \mathbb{Z}_p^*$;
    Prover computes:
        Host → Device: $P$, Device computes:
        $\mathbf{a}', \mathbf{b}', \mathbf{g}', \mathbf{h}', P'$
    Verifier computes:
        Host → Device: $P$, Device computes: $\mathbf{g}', \mathbf{h}', P'$;
    Prover updates:
        Device → Host: $\mathbf{g}', \mathbf{h}', P', \mathbf{a}', \mathbf{b}'$;
        $(\mathbf{g}', \mathbf{h}', P', \mathbf{a}', \mathbf{b}') \rightarrow (\mathbf{g}, \mathbf{h}, P, \mathbf{a}, \mathbf{b})$
    Verifier updates:
        Device → Host: $\mathbf{g}', \mathbf{h}', P'$;
        $(\mathbf{g}', \mathbf{h}', P') \rightarrow (\mathbf{g}, \mathbf{h}, P)$
    Recursively run algorithm 1.
  end if
```

Algorithm 1 is a specific collaborative acceleration scheme in the process of the inner-product argument. The communication volume of the argument was reduced by $\frac{1}{3}$. The prover and verifier calculate according to the formula (1)–(4). When it involves parallel inner product calculation and elliptic curve calculation, the data is sent to the GPU for parallel calculation, and the result is temporarily stored until it is used.

6.2. Data categorization

As aforementioned in Section 4, data are categorized into 3 types before processing as follows.

- *z_data*: data that are small-sized and accessed only once are called *z_data*. These data are fetched by Mode 1.
- *l_data*: data that are accessed repeatedly are called *l_data*. These data are fetched by Mode 2.
- *s_data*: data with a bitwidth larger than 352 bit[2] are called *s_data*. These data are fetched by Mode 3.

Data classification is assisted by the tool Perf [45]. Perf is a system performance optimization tool in the Linux kernel, which can instrument CPU performance counters, tracepoints, kprobes, and uprobes (dynamic tracing). Therefore, it can be used to view and analyze the operation of the program. For instance, *perf stat* is used to view the number of process switches, cache utilization, and *perf tracepoint* is used to sample tick time points to get hot spots in the kernel code.

[2] The GPU memory bitwidth used in the experiment is 352 bit.

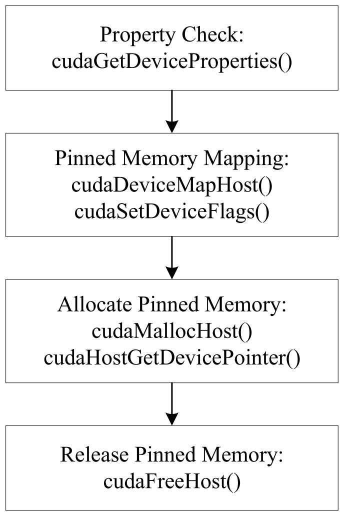

Fig. 12. Zero-copy memory implementation based on cuda library.

6.3. Zero-copy memory

The zero-copy memory is implemented with the help of the cuda library *functions*. We describe the procedure of calling APIs and their functions as showed in Fig. 12.

First, *cudaGetDeviceProperties()* is used to determine whether the GPU version supports direct mapping of host memory to GPU. Secondly, *cudaSetDeviceFlags()* is called by *cudaDeviceMapHost()* to start pinned memory mapping. This is because the pinned memory is shared by the host and the device, so it is necessary to maintain synchronization to prevent modifying the data in the zero-copy memory by more than one party at the same time. Thirdly, *cudaMallocHost()* is called to allocate the pinned memory to map the host memory, and obtain a pointer to the mapped device address space by calling *cudaHostGetDevicePointer()*. Finally, *cudaFreeHost()* is used to release the pinned memory.

6.4. Lookup table

When implementing the lookup tables, it is critical to determine the table size. In practice, we gradually increase the table size and watch the performance metrics including hit ratio, time required for initialization, and time cost in model calculation. The shared memory used in experiment is 48 KB.

Fig. 13 shows that as the size of the look-up table increases, the hit ratio also increases and gradually approaches a constant value. This is because the larger the size, the more data can be stored in advance. Similarly, the larger the size of the lookup table, the longer the time required to initialize the lookup table, but the shorter the time required for consequent model calculation. However, when the lookup table size is greater than a threshold, both the initialization time and the consequent model calculation time rapidly increase. It is because there is not enough space to maintain shared memory's own functions when the lookup table is too large. By considering all the performance metrics, a lookup table size of 24 KB is set in the experiment.

We apply the Fibonacci [46] search algorithm for searching and updating the lookup table due to its overall good performance and fast searching speed. Fig. 14 shows that hit ratio increases as the lookup table keeps updating. The table size is set to be 24 KB.

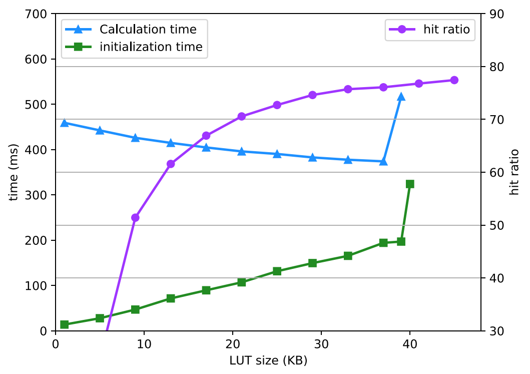

Fig. 13. Hit ratio, time for initialization, and time for model calculation with different LUT sizes.

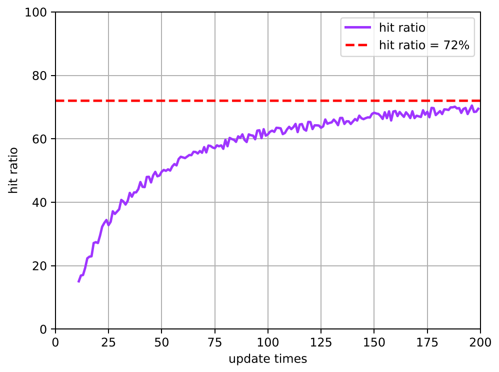

Fig. 14. Hit ratio increase as the table updates. The table size is 24 KB.

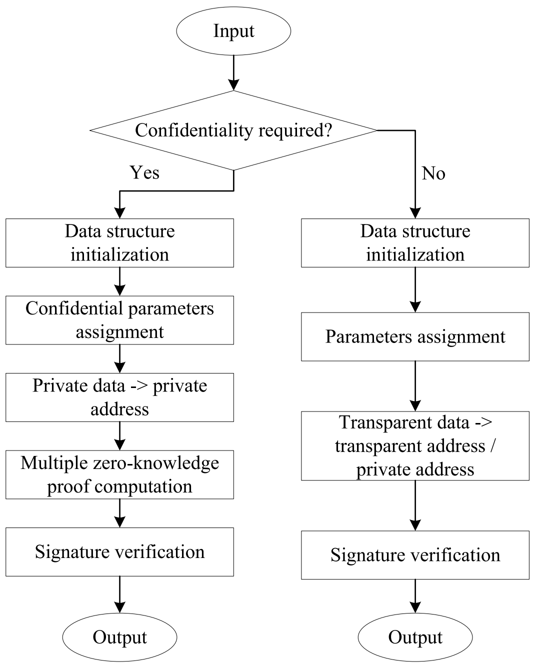
Input
Confidentiality required?
Yes
No
Data structure initialization
Confidential parameters assignment
Private data -> private address
Multiple zero-knowledge proof computation
Signature verification
Output
Data structure initialization
Parameters assignment
Transparent data -> transparent address / private address
Signature verification
Output

Fig. 15. The overall flow chart of parallel transactions.

6.5. Data segmentation

Algorithm 2 describes the data segmentation procedure for data with a bitwidth larger than 352 bit. The data is split into multiple slices with a size of 320 bits. The size of the slice should be a multiple

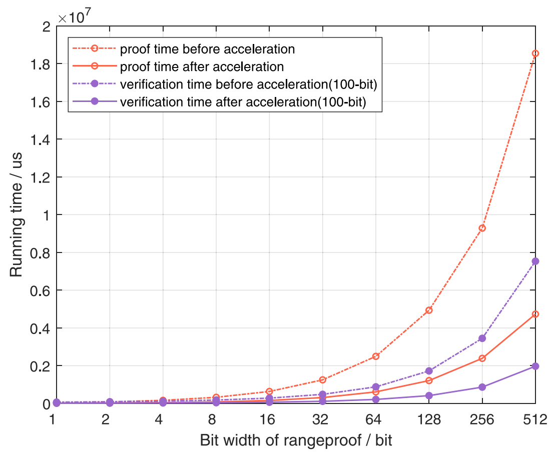

Fig. 16. Accelerated result comparison on RTX 2080 Ti.

of 16, and a slice of 320 bits performs best on Nvidia 2080 Ti in the experiments. The Prover operation is applied to each slice and the results are assembled to restall the final prover P.

Algorithm 2 Data Segmentation Implementation

```
if data.size > 352 bits then
    Partition data into $n$ slices of 320 bits
    Apply Prover operation on each slice
    Assemble the resulted slices to get $P$
end if
```

6.6. Implement parallel transactions

In Fig. 15, we describe the flow chart of parallel transactions. There are two kinds of transactions, transparent and confidential. Compared with transparent transactions, confidential transactions require zero-knowledge proofs to verify legitimacy while transparent transactions do not. Confidential transactions also require one more pair of keys to verify data addresses in addition to a pair of keys to verify identity. In the meanwhile, transparent transactions only need a pair of keys to verify identity. Finally, the data structures supporting confidential transactions are more complex than that of transparent transactions.

7. Performance evaluation

7.1. Accelerate a single Bulletproof

The experiments are conducted on a Docker image of version 20.10.3 on Intel® Xeon® Gold 6230 CPU @2.10 GHz, with the GPU devices of NVIDIA® Cuda release 10.1 on NVIDIA® GeForce® RTX 2080 Ti/NVIDIA® or Tesla® V100 to evaluate the performance. The Pedersen data and various hash functions are used as the input data.

Fig. 16 reports the running time of proof and verification in Bulletproofs' rangeproof procedure on the RTX 2080 Ti platform. We gradually increase the input size from (1-bit, 1-commit) to (512-bit, 256-commit). Fig. 16 shows that as the input bit-width increases, the speedup ratio of Algorithm 1 increases with both the proof and verification procedures. The speedup ratio of the verification procedure is slightly larger than that of the proof procedure. It is because the verification contains more elliptic curve point arithmetic operations that can be parallelized.

In order to demonstrate the universal applicability of Algorithm 1 and explore the acceleration ability of different GPU platforms, the experiments are also tested on the V100 platform. The input data are

Table 2
Comparison of proof time on different GPUs.

	Time on 2080 Ti/μs		Time on V100/μs	
	Without acc	With acc	Without acc	With acc
Pedersen-3	9947	3761	10 036	2695
Pedersen-6	17 917	4840	18 032	4829
Pedersen-12	34 469	9264	34 617	9286
Pedersen-24	67 600	18 017	67 497	18 062
Pedersen-48	133 705	35 475	132 979	35 568
Pedersen-96	261 177	69 462	260 316	69 627
Pedersen-192	508 701	136 028	509 822	136 318
Pedersen-384	999 127	266 884	1 000 085	267 433
Pedersen-768	1 986 601	526 438	1 971 723	527 467
Pedersen-1536	3 940 234	1 038 547	3 892 699	1 041 097
Pedersen-3072	7 894 753	2 062 248	7 727 506	2 066 544
Avg-speedup	3.657		3.736	

Table 3
Running time and speed-up ratios under different optimization method combinations.

	Proof time	Verify time	Average speedup ratio
Bulletproofs	4 201 727 μs	1 314 946 μs	/
Original model	1 143 327 μs	362 444 μs	3.664
om+zcm	1 063 558 μs	339 238 μs	3.933 (3.664 ∗ 1.073)
om+zcm+lut	916 072 μs	288 959 μs	4.578 (3.933 ∗ 1.164)
om+zcm+lut+ds	885 946 μs	281 088 μs	4.727 (4.578 ∗ 1.033)

Pedersen Hash varying from 8-bit (Pedersen-3) to 8192-bit (Pedersen-3072).

Table 2 presents the running time of the proof procedure in the circuit module of Bulletproofs with and without acceleration on different GPU platforms. It shows that there is no significant difference in the speedup ratios between different GPU devices. The more powerful Tesla V100 does not necessarily achieve a much higher speedup compared with the RTX 2080. We conclude that Algorithm 1 has the ability to perform well on a wide spectrum of GPU devices with sufficient cores.

In addition, Fig. 17 presents the speedup ratio of the proof procedure in the circuit module of Bulletproofs on different GPU platforms. It shows that the speed-up ratio of the proof process may be slightly larger as the number of verification bits increases on 2080 Ti while the acceleration ratio on the V100 is more stable. Except for a small jump at the beginning, the acceleration ratio of the 2080 Ti is mainly reflected in a small drop when the input data bitwidth is between 256 and 512. This is also consistent with our judgment that the bitwidth of 2080Ti is 352 bits. This is not the case with the V100 because its bitwidth of 4096 bits is sufficient large. Therefore, we propose to partition the data into multiple slices that align with the GPU bitwidth.

7.2. Accelerate multiple transactions

There are two independent procedures in the zero-knowledge proof algorithm, i.e., proof and verification. In experiments, we collected the running time for proof and verification separately. We also tried different combinations of the optimization methods proposed in Section 3. The Pedersen mixed data with a varying bitwidth between 192 bits and 3072 bits are fed into the model. Table 2 shows the running time and speedup ratios of proof and verification under different technique combinations. The notations om, zcm, lut, and ds are shorthands for models with basic GPU parallelization, with zero-copy memory, with look-up table, and with data segmentation, respectively. Fig. 18 plots the running time of each technique combinations. Both Table 3 and Fig. 18 show that the optimization methods we proposed achieve good results in general, and the running speed has been increased by nearly 30%.

Fig. 18 and Table 3 also show that the lookup table significantly reduces the running time compared the two other methods of zero-copy memory and data segmentation. The intuition behind the not-so-good performance of zero-copy memory is that there exist many small-sized data that are repeatedly accessed. Regarding the performance of data segmentation, the transmission and reassemble of data slices also consume time and therefore the speedup is not significant.

Finally, we compare the performance with the bare zero-knowledge proofs algorithm without any acceleration throughout the confidential transaction model. In the previous single zero-knowledge proof experiments, we mainly used Pedersen data as input. In order to increase the richness of the experiment, and also show that our algorithms and model are data ubiquitous, we use a hash function as input. Table 4 shows the running time and speedup ratios with different cryptographic hash functions. With any kind of hash functions, the overall security system achieves a speedup ratio close to 1.5x. Thus the proposed system can achieve a faster running speed while achieving partial information security.

Table 4
Speedup ratio compared with no acceleration.

	SHA1	SHA256	SHA512
Without acceleration	3 883 672 μs	4 281 238 μs	10 153 094 μs
With acceleration	2 629 433 μs	2 956 656 μs	6 944 660 μs
Speedup ratio	1.477	1.448	1.462

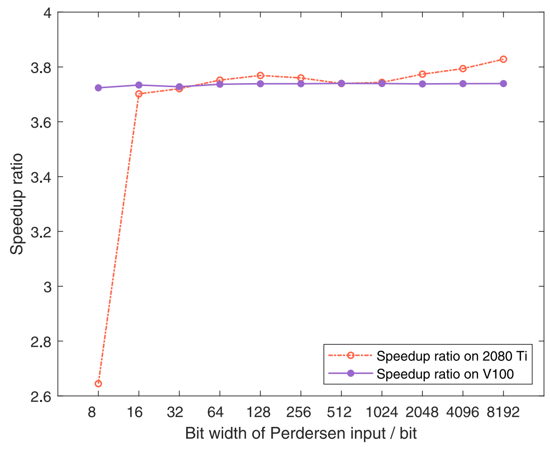

Fig. 17. Speedup ratio of proof procedure in circuit module on different GPUs.

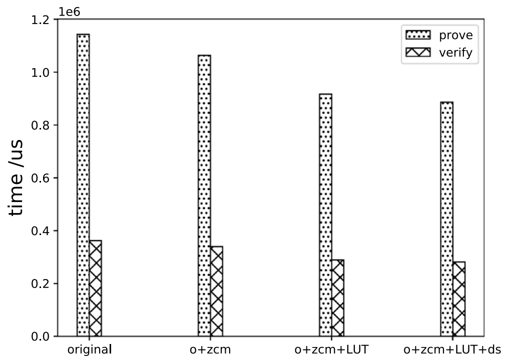

Fig. 18. Running time of the optimized model.

8. Conclusion

In response to the challenge that transaction privacy is desired in various blockchain applications, we have first proposed a CPU–GPU cooperative acceleration design for the Bulletproofs algorithm. The running time of Bulletproofs was scaled down by 3–4x by accelerating inner product computation and finite field modular arithmetic in inner product arguments. Then, we built a transaction processing system based on parallelized zero-knowledge proofs at GPU-enabled edge, and applied an integrated mechanism consisting of memory optimization and data segmentation. All the proposed techniques can be extended to most of zero-knowledge proof variants, such as zk-SNARK and zk-STARK. The Bulletproofs algorithm has been improved by a speedup ratio of 4.7x compared with the original version, and the overall system has achieves a 1.5x speedup. The work has broad application prospects in blockchain-related areas, as it demonstrates great potential to significantly increase the speed of data processing through heterogeneous acceleration at smart edge while guaranteeing data traceability and privacy.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The research in this paper was supported by National SKA Program of China (grant No. 2020SKA0120202), National Natural Science Foundation of China (grant No. U2032125), Independent Deployment Project of Shanghai Advanced Research Institute (grant No. E0560W1ZZ0, E052891ZZ1), Shanghai Committee of Science and Technology (grant No. 19511131202).

References

[1] H. Qiu, M. Qiu, G. Memmi, Z. Ming, M. Liu, A dynamic scalable blockchain based communication architecture for IoT, in: International Conference on Smart Blockchain, Springer, 2018, pp. 159–166, http://dx.doi.org/10.1007/978-3-030-05764-0_17.

[2] D. Yaga, P. Mell, N. Roby, K. Scarfone, Blockchain technology overview, 2019, http://dx.doi.org/10.48550/arXiv.1906.11078, arXiv preprint arXiv:1906.11078.

[3] S. Morishima, H. Matsutani, Acceleration of anomaly detection in blockchain using in-gpu cache, in: 2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom), IEEE, 2018, pp. 244–251, http://dx.doi.org/10.1109/BDCloud.2018.00047.

[4] K. Gai, M. Qiu, X. Sun, H. Zhao, Security and privacy issues: A survey on FinTech, in: International Conference on Smart Computing and Communication, Springer, 2016, pp. 236–247, http://dx.doi.org/10.1007/978-3-319-52015-5_24.

[5] E. Androulaki, G.O. Karame, M. Roeschlin, T. Scherer, S. Capkun, Evaluating user privacy in bitcoin, in: International Conference on Financial Cryptography and Data Security, Springer, 2013, pp. 34–51, http://dx.doi.org/10.1007/978-3-642-39884-1_4.

[6] O. Goldreich, Y. Oren, Definitions and properties of zero-knowledge proof systems, J. Cryptol. 7 (1) (1994) 1–32, http://dx.doi.org/10.1007/BF00195207.

[7] E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, M. Virza, SNARKs for C: Verifying program executions succinctly and in zero knowledge, in: Annual Cryptology Conference, Springer, 2013, pp. 90–108, http://dx.doi.org/10.1007/978-3-642-40084-1_6.

[8] E.B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, M. Virza, Zerocash: Decentralized anonymous payments from bitcoin, in: 2014 IEEE Symposium on Security and Privacy, IEEE, 2014, pp. 459–474, http://dx.doi.org/10.1109/SP.2014.36.

[9] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, G. Maxwell, Bulletproofs: Short proofs for confidential transactions and more, in: 2018 IEEE Symposium on Security and Privacy, SP, IEEE, 2018, pp. 315–334, http://dx.doi.org/10.1109/SP.2018.00020.

[10] J.M. Cecilia, J.-C. Cano, J. Morales-García, A. Llanes, B. Imbernón, Evaluation of clustering algorithms on GPU-based edge computing platforms, Sensors 20 (21) (2020) 6335, http://dx.doi.org/10.3390/s20216335.

[11] A. Kelkar, C. Dick, A GPU hyperconverged platform for 5G vRAN and multi - access edge computing, in: 2021 IEEE Canadian Conference on Electrical and Computer Engineering, CCECE, 2021, pp. 1–6, http://dx.doi.org/10.1109/CCECE53047.2021.9569133.

[12] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, J.P. Jue, All one needs to know about fog computing and related edge computing paradigms: A complete survey, J. Syst. Archit. 98 (2019) 289–330, http://dx.doi.org/10.1016/j.sysarc.2019.02.009.

[13] N. Eltayieb, R. Elhabob, A. Hassan, F. Li, A blockchain-based attribute-based signcryption scheme to secure data sharing in the cloud, J. Syst. Archit. 102 (2020) 101653, http://dx.doi.org/10.1016/j.sysarc.2019.101653.

[14] J. Yang, Z. Lu, J. Wu, Smart-toy-edge-computing-oriented data exchange based on blockchain, J. Syst. Archit. 87 (2018) 36–48, http://dx.doi.org/10.1016/j.sysarc.2018.05.001.

[15] J. Bonneau, A. Narayanan, A. Miller, J. Clark, J.A. Kroll, E.W. Felten, Mixcoin: Anonymity for bitcoin with accountable mixes, in: International Conference on Financial Cryptography and Data Security, Springer, 2014, pp. 486–504, http://dx.doi.org/10.1007/978-3-662-45472-5_31.

[16] S.-F. Sun, M.H. Au, J.K. Liu, T.H. Yuen, Ringct 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero, in: European Symposium on Research in Computer Security, Springer, 2017, pp. 456–474, http://dx.doi.org/10.1007/978-3-319-66399-9_25.

[17] H. Gao, Z. Ma, S. Luo, Z. Wang, BFR-MPC: a blockchain-based fair and robust multi-party computation scheme, IEEE Access 7 (2019) 110439–110450, http://dx.doi.org/10.1109/ACCESS.2019.2934147.

[18] T. Hardjono, N. Smith, Decentralized trusted computing base for blockchain infrastructure security, Front. Blockchain 2 (2019) 24, http://dx.doi.org/10.3389/fbloc.2019.00024.

[19] Z. Zhao, T.-H.H. Chan, How to vote privately using bitcoin, in: International Conference on Information and Communications Security, Springer, 2015, pp. 82–96, http://dx.doi.org/10.1007/978-3-319-29814-6_8.

[20] H.S. Galal, A.M. Youssef, Verifiable sealed-bid auction on the ethereum blockchain, in: International Conference on Financial Cryptography and Data Security, Springer, 2018, pp. 265–278, http://dx.doi.org/10.1007/978-3-662-58820-8_18.

[21] A. Delignat-Lavaud, C. Fournet, M. Kohlweiss, B. Parno, Cinderella: Turning shabby X. 509 certificates into elegant anonymous credentials with the magic of verifiable computation, in: 2016 IEEE Symposium on Security and Privacy, SP, IEEE, 2016, pp. 235–254, http://dx.doi.org/10.1109/SP.2016.22.

[22] T.P. Pedersen, Non-interactive and information-theoretic secure verifiable secret sharing, in: Annual International Cryptology Conference, Springer, 1991, pp. 129–140, http://dx.doi.org/10.1007/3-540-46766-1_9.

[23] G.G. Dagher, B. Bünz, J. Bonneau, J. Clark, D. Boneh, Provisions: Privacy-preserving proofs of solvency for bitcoin exchanges, in: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, 2015, pp. 720–731, http://dx.doi.org/10.1145/2810103.2813674.

[24] A. Kosba, A. Miller, E. Shi, Z. Wen, C. Papamanthou, Hawk: The blockchain model of cryptography and privacy-preserving smart contracts, in: 2016 IEEE Symposium on Security and Privacy, SP, IEEE, 2016, pp. 839–858, http://dx.doi.org/10.1109/SP.2016.55.

[25] Y. Zhang, D. Genkin, J. Katz, D. Papadopoulos, C. Papamanthou, vSQL: Verifying arbitrary SQL queries over dynamic outsourced databases, in: 2017 IEEE Symposium on Security and Privacy, SP, IEEE, 2017, pp. 863–880, http://dx.doi.org/10.1109/SP.2017.43.

[26] J. Zhang, Z. Fang, Y. Zhang, D. Song, Zero knowledge proofs for decision tree predictions and accuracy, in: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, 2020, pp. 2039–2053, http://dx.doi.org/10.1145/3372297.3417278.

[27] B.W. Zhao, Zhentang zhao: past and future of the shanghai synchrotron radiation facility, Natl. Sci. Rev. 8 (12) (2021) nwab185, http://dx.doi.org/10.1093/nsr/nwab185.

[28] J. Shi, C. Li, H. Wu, H. Gao, S. Jin, T. Huang, W. Zhang, Evaluating the parallel execution schemes of smart contract transactions in different blockchains: An empirical study, in: International Conference on Algorithms and Architectures for Parallel Processing, Springer, 2021, pp. 35–51, http://dx.doi.org/10.1007/978-3-030-95391-1_3.

[29] B. Peng, Y. Zhu, N. Jing, X. Zheng, Y. Zhou, Design of a hardware accelerator for zero-knowledge proof in blockchains, in: International Conference on Smart Computing and Communication, Springer, 2020, pp. 136–145, http://dx.doi.org/10.1007/978-3-030-74717-6_15.

[30] A.A. Yazdeen, S.R. Zeebaree, M.M. Sadeeq, S.F. Kak, O.M. Ahmed, R.R. Zebari, FPGA implementations for data encryption and decryption via concurrent and parallel computation: A review, Qubahan Acad. J. 1 (2) (2021) 8–16, http://dx.doi.org/10.48161/qaj.v1n2a38.

[31] M.Z. Uddin, A wearable sensor-based activity prediction system to facilitate edge computing in smart healthcare system, J. Parallel Distrib. Comput. 123 (2019) 46–53, http://dx.doi.org/10.1016/j.jpdc.2018.08.010.

[32] Y. Zhang, S. Wang, X. Zhang, J. Dong, X. Mao, F. Long, C. Wang, D. Zhou, M. Gao, G. Sun, Pipezk: Accelerating zero-knowledge proof with a pipelined architecture, in: 48th IEEE/ACM International Symposium on Computer Architecture, ISCA, 2021, http://dx.doi.org/10.1109/ISCA52012.2021.00040.

[33] J. Ren, J. Luo, K. Wu, M. Zhang, H. Jeon, D. Li, Sentinel: Efficient tensor migration and allocation on heterogeneous memory systems for deep learning, in: 2021 IEEE International Symposium on High-Performance Computer Architecture, HPCA, IEEE, 2021, pp. 598–611, http://dx.doi.org/10.1109/HPCA51647.2021.00057.

[34] S. Morishima, Scalable anomaly detection method for blockchain transactions using GPU, in: 2019 20th International Conference on Parallel and Distributed Computing, Applications and Technologies, PDCAT, IEEE, 2019, pp. 160–165, http://dx.doi.org/10.1109/PDCAT46702.2019.00039.

[35] J. Zhang, L. Liu, Publicly verifiable watermarking for intellectual property protection in FPGA design, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 25 (4) (2017) 1520–1527, http://dx.doi.org/10.1109/TVLSI.2016.2619682.

[36] J. Zhang, T. Xie, Y. Zhang, D. Song, Transparent polynomial delegation and its applications to zero knowledge proof, in: 2020 IEEE Symposium on Security and Privacy, SP, IEEE, 2020, pp. 859–876, http://dx.doi.org/10.1109/SP40000.2020.00052.

[37] L. Xiang, B. Tang, C. Yang, Accelerating exact inner product retrieval by CPU-gpu systems, in: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2019, pp. 1277–1280, http://dx.doi.org/10.1145/3331184.3331376.

[38] M. Qiu, K. Gai, Z. Xiong, Privacy-preserving wireless communications using bipartite matching in social big data, Future Gener. Comput. Syst. 87 (2018) 772–781, http://dx.doi.org/10.1016/j.future.2017.08.004.

[39] W. Dai, M. Qiu, L. Qiu, L. Chen, A. Wu, Who moved my data? Privacy protection in smartphones, IEEE Commun. Mag. 55 (1) (2017) 20–25, http://dx.doi.org/10.1109/MCOM.2017.1600349CM.

[40] B. Bünz, M. Maller, P. Mishra, N. Vesely, Proofs for inner pairing products and applications, IACR Cryptol. ePrint Arch. 1177 (2019) 2019, http://dx.doi.org/10.1007/978-3-030-92078-4_3.

[41] J. Bootle, A. Cerulli, P. Chaidos, J. Groth, C. Petit, Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting, in: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2016, pp. 327–357, http://dx.doi.org/10.1007/978-3-662-49896-5_12.

[42] Y. Huang, X. Zheng, Y. Zhu, X. Kong, X. Jing, CPU-GPU collaborative acceleration of bulletproofs-a zero-knowledge proof algorithm, in: 2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), IEEE, 2021, pp. 674–680, http://dx.doi.org/10.1109/ISPA-BDCloud-SocialCom-SustainCom52081.2021.00098.

[43] G. Alavani, K. Varma, S. Sarkar, Predicting execution time of CUDA kernel using static analysis, in: 2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom), IEEE, 2018, pp. 948–955, http://dx.doi.org/10.1109/BDCloud.2018.00139.

[44] M. Qiu, Z. Chen, M. Liu, Low-power low-latency data allocation for hybrid scratch-pad memory, IEEE Embedded Syst. Lett. 6 (4) (2014) 69–72, http://dx.doi.org/10.1109/LES.2014.2344913.

[45] A.C. De Melo, The new linux'perf'tools, in: Slides from Linux Kongress, Vol. 18, 2010, pp. 1–42.

[46] S. Falcón, Á. Plaza, On the Fibonacci k-numbers, Chaos Solitons Fractals 32 (5) (2007) 1615–1624, http://dx.doi.org/10.1016/j.chaos.2006.09.022.