2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)

2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom) | 978-1-6654-3574-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/ISPA-BDCloud-SocialCom-SustainCom52081.2021.00098

CPU-GPU Collaborative Acceleration of Bulletproofs - A Zero-Knowledge Proof Algorithm

Ying Huang*†, Xiaoying Zheng*†, Yongxin Zhu*†, Xiangcong Kong*†, Xinru Jing*†
*Shanghai Advanced Research Institute, Chinese Academy of Sciences, Shanghai 201210, China
†University of Chinese Academy of Sciences, Beijing 100049, China

Abstract—Zero-knowledge proofs help to protect the privacy and security of blockchains by keeping the transaction information private. Bulletproofs are the state-of-the-art zero-knowledge technologies that perform confidential transactions without a trusted setup. However, Bulletproofs are still computationally inefficient to be applied in blockchain applications, and it is of great significance to parallelize Bulletproofs on GPUs. In this paper, we present a CPU-GPU collaborative framework to accelerate the inner-product arguments of Bulletproofs. To our best knowledge, it is the first time that Bulletproofs are implemented in a CPU-GPU hybrid system. The experiments show that our implementation achieves an average speedup ratio of 3.7x. The results also demonstrate that the CPU-GPU collaborative acceleration of Bulletproofs has properties of small size, high efficiency, and high scalability.

Index Terms—zero-knowledge proof, blockchain, parallel acceleration, CPU-GPU co-acceleration, inner-product arguments.

I. 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]. By recording transactions in a shared ledger, it allows everybody to see what happens on the blockchain. Thus an unfortunate side effect is that all the transaction data are public, which is at odds with the normal expectation of privacy [4]. Insufficient financial privacy [5] can have serious security and privacy implications for both commercial and personal transactions [6].

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 [7].

For instance, assume that the two parties to the transaction are A and B, and A has two accounts at the beginning

The corresponding authors are X. Zheng (zhengxy@sari.ac.cn) and Y. Zhu (zhuyongxin@sari.ac.cn).

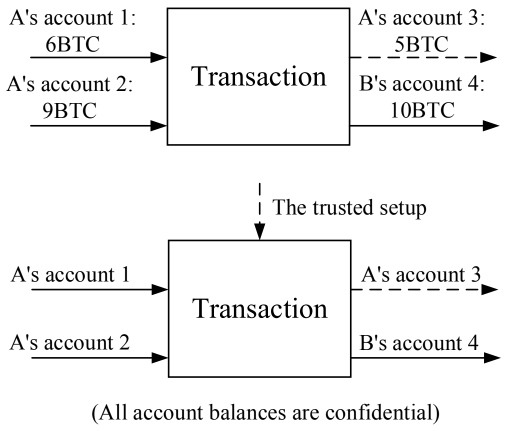

Fig. 1: The transaction without (top) or with (bottom) the confidential mechanism.

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 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.

At present, ZK-SNARKs (Zero-Knowledge Succinct Non-interactive Argument of Knowledge) [8], ZK-STARKs (Zero-Knowledge Scalable Transparent Arguments of Knowledge) and Bulletproofs [9] are the state-of-the-art zero-knowledge proof algorithms. ZK-SNARKs require a trusted setup ceremony to ensure that the two parties to the transaction will not collude. ZK-SNARKs are computationally efficient but the requirement of a trusted third party limits the applications in blockchains. ZK-STARKs improve upon ZK-SNARKs by removing the need for a trusted setup at the cost of computational inefficiency. Bulletproofs further improve ZK-STARKS by transforming interval proofs into vector inner product computation, which greatly reduces the complexity. However, verifying a Bulletproof is still more time-consuming than verifying a ZK-SNARK proof.

Fortunately, the inner product computation can be highly parallel computed in GPU. While inner product computation is readily parallelized, it is not trivial to apply CPU-GPU systems to boost the performance of inner product computation.

We propose to accelerate the verifying a Bulletproofs by fully unlocking the power of CPU-GPU systems to parallelize inner product computation. And to our best knowledge, it is the first time that the Bulletproofs are implemented in a CPU-GPU system. We summarize our contribution as follows.

- Accelerate the verifying a Bulletproof by fully parallelizing inner product computation in CPU-GPU systems and scales down the execution time by 3-4x.
- Applying data grouping and pipeline methods to process high-level input data to improve acceleration efficiency.
- Implement Bulletproofs on different CPU-GPU platforms and demonstrate the high efficiency and scalability of Bulletproofs.

The remainder of this paper is structured as follows: Section II provides basic concepts and general information about inner-product argument and elliptic curve point arithmetic in Bulletproofs while the designed CPU-GPU collaborative algorithm is described in Section III. The experiments and results based on the algorithm are presented in Section IV. Finally, in Section V, the conclusion and future work are presented.

II. Background

The current acceleration schemes in the field of zero-knowledge proofs are all accelerating the zk-snark algorithm and the zk-stark algorithm [10], and most of them are based on FPGA to achieve acceleration [11] or by changing the hardware structure, such as adding pipeline accelerators to achieve hardware acceleration [12]. As for GPU acceleration, the blockchain project Filecoin uses GPU acceleration to shorten the PoST proof time in zk-snark, and Long et al. [13] 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. [14] proposed a novel privacy-preserving method using bipartite matching and attracted much follow-up work. Dai et al. [15] 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 [16]. Bulletproofs build on the communication efficient inner-product argument proposed by Bootle et al. [17]. The argument proves that the prover knows the openings of two binding Pedersen vector commitments that satisfy a given inner product relation [9].

A. Inner-product argument

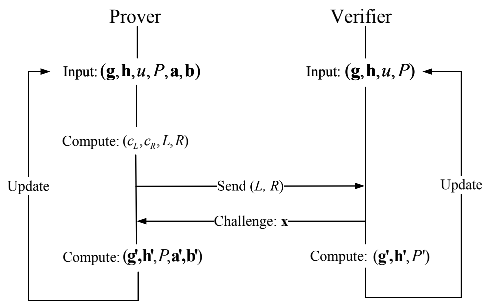
Prover
Verifier
Input: $(\mathbf{g},\mathbf{h},u,P,\mathbf{a},\mathbf{b})$
Input: $(\mathbf{g},\mathbf{h},u,P)$
Compute: (c_L,c_R,L,R)
Update
Send (L, R)
Update
Challenge: $\mathbf{x}$
Compute: $(\mathbf{g}',\mathbf{h}',P,\mathbf{a}',\mathbf{b}')$
Compute: $(\mathbf{g}',\mathbf{h}',P')$

Fig. 2: The procedure of inner-product argument in Bulletproofs.

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.

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

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.

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

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 equations (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.

B. 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 Fig. 3 and Fig. 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., $P + Q = -R$.

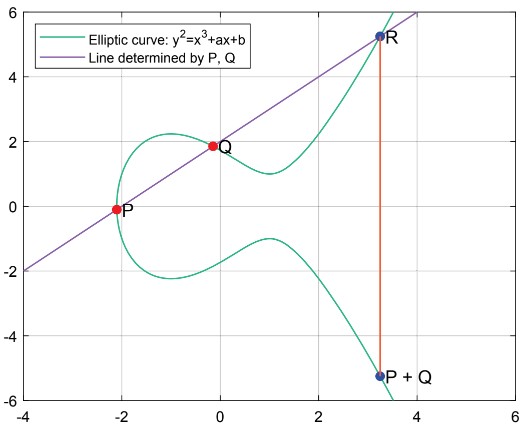

Fig. 3: Elliptic curve point addition ($P \neq Q$).

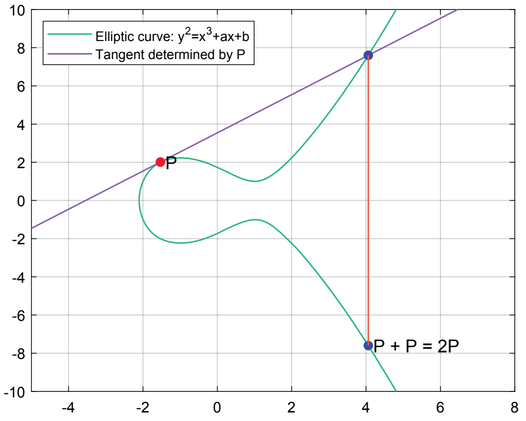

Fig. 4: Elliptic curve point addition ($P = Q$).

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

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

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.

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

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.

III. Algorithm Design

According to the algorithm level division implemented on GPU, parallelization can be divided into coarse-grained parallelization and fine-grained parallelization [18] [19]. Taking the parallelization in the elliptic curve as an example, coarse-grained parallelization refers to the use of serialization in the underlying algorithm, that is, a single thread calculates a finite field, but parallelism is used at the modular exponentiation level. The fine-grained parallelization refers to the split parallelization started by the underlying modular multiplication algorithm, and the finite field algorithm is calculated by multiple threads [20]. Although the coarse-grained parallel scheme combines the advantages of full serial and fine-grained parallel to a certain extent, due to the asymmetric characteristics of the elliptic curve algorithm, the design of coarse-grained parallel algorithms is much more difficult, and it is also easy to cause waste of computing resources, so the fine-grained parallelization scheme is selected here.

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 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 [21] [22]. 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.

Overall, 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.

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 $\rightarrow$ 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 $\rightarrow$ Device: $\mathbf{a}, \mathbf{b}, n'$, Device computes: $c_L, c_R$;
          Host $\rightarrow$ Device: $\mathbf{g}, \mathbf{h}, u$, Device computes: $L, R$;
          Device $\rightarrow$ Host: $L, R$;
      Prover $\rightarrow$ Verifier: $L, R$;
      Verifier $\rightarrow$ Prover: $x \xleftarrow{\$} \mathbb{Z}_p^*$;
      Prover computes:
          Host $\rightarrow$ Device: $P$, Device computes:
          $\mathbf{a}', \mathbf{b}', \mathbf{g}', \mathbf{h}', P'$
      Verifier computes:
          Host $\rightarrow$ Device: $P$, Device computes: $\mathbf{g}', \mathbf{h}', P'$;
      Prover updates:
          Device $\rightarrow$ 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 $\rightarrow$ Host: $\mathbf{g}', \mathbf{h}', P'$;
          $(\mathbf{g}', \mathbf{h}', P') \rightarrow (\mathbf{g}, \mathbf{h}, P)$
      Recursively run algorithm 1.
    end if
```

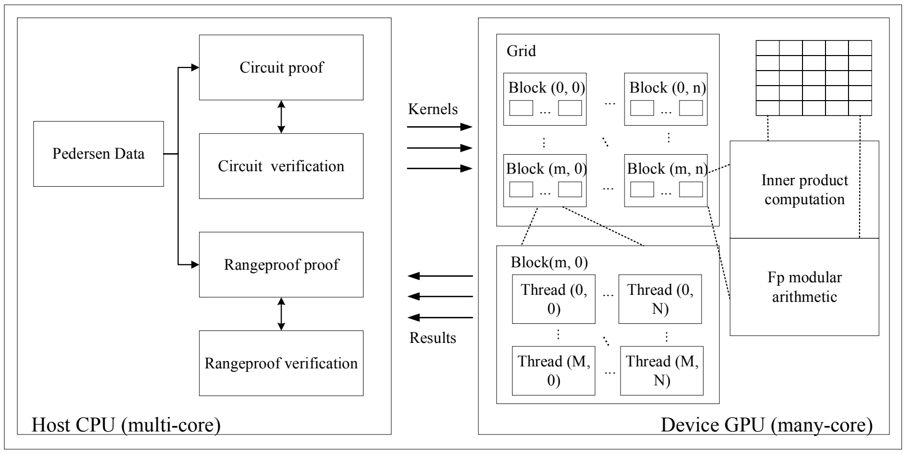

Fig. 5: The CPU-GPU collaborative operation framework.

IV. Experiments

The experiments are conducted on a Docker image of version 20.10.3 on Intel® Xeon® Gold 6230 CPU @ 2.10GHz, with the GPU devices of NVIDIA® Cuda release 10.1 on NVIDIA® GeForce® RTX 2080 Ti/NVIDIA® Tesla® V100 to evaluate the proposed algorithm with heterogeneous devices.

Fig. 6 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. 6 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.

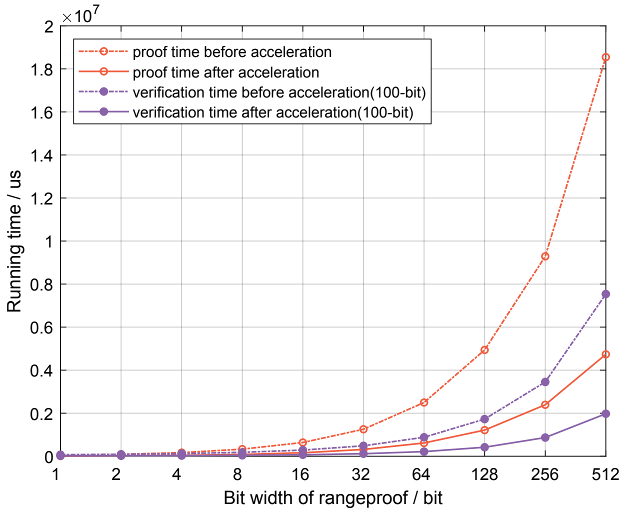

Fig. 6: Accelerated result comparison on RTX 2080 Ti.

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

Table I 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.

TABLE I: Comparison of proof time on different GPUs

	Time on 2080 Ti / us		Time on V100 / us	
	without acc	with acc	without acc	with acc
Perdersen-3	9947	3761	10036	2695
Perdersen-6	17917	4840	18032	4829
Perdersen-12	34469	9264	34617	9286
Perdersen-24	67600	18017	67497	18062
Perdersen-48	133705	35475	132979	35568
Perdersen-96	261177	69462	260316	69627
Perdersen-192	508701	136028	509822	136318
Perdersen-384	999127	266884	1000085	267433
Perdersen-768	1986601	526438	1971723	527467
Perdersen-1536	3940234	1038547	3892699	1041097
Perdersen-3072	7894753	2062248	7727506	2066544
Avg-speedup	3.657		3.736	

In addition, Fig 7 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.

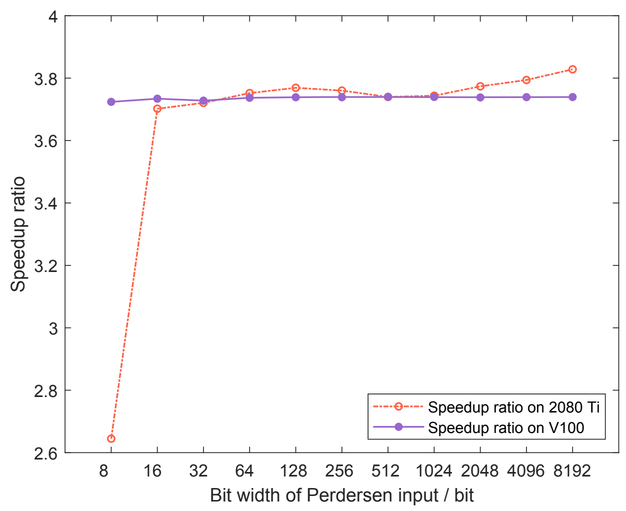

Fig. 7: Speedup ratio of proof procedure in circuit module on different GPUs.

V. Conclusion

This paper 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. In the meanwhile, by testing Algorithm 1 on different GPU devices, it demonstrated that the CPU-GPU cooperative acceleration design has the features of universality, high efficiency, and high scalability. It was noticed that the speedup ratio of the verification procedure depends on the proof and verification bit-width which has an impact on the GPU I/O efficiency. A verification bit-width that is a multiple of the GPU's I/O bit-width usually achieves a higher speedup ratio [23].

The future work includes data grouping and pipelining for high-bit data, and loop unrolling or software pipelining to reduce the data transmission consumption in the CPU-GPU systems.

VI. Acknowledgement

The research in this paper was supported by National Key Research and Development Project (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 Science and Technology Commission Project (grant No. 19511131202).

References

[1] H. Qiu, M. Qiu, G. Memmi, Z. Ming, and M. Liu, "A dynamic scalable blockchain based communication architecture for iot," in International Conference on Smart Blockchain. Springer, 2018, pp. 159–166.

[2] D. Yaga, P. Mell, N. Roby, and K. Scarfone, "Blockchain technology overview," arXiv preprint arXiv:1906.11078, 2019.

[3] S. Morishima and 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), 2018, pp. 244–251.

[4] K. Gai, Y. Wu, L. Zhu, Z. Zhang, and M. Qiu, "Differential privacy-based blockchain for industrial internet-of-things," IEEE Transactions on Industrial Informatics, vol. 16, no. 6, pp. 4156–4165, 2019.

[5] K. Gai, M. Qiu, X. Sun, and H. Zhao, "Security and privacy issues: A survey on fintech," in International Conference on Smart Computing and Communication. Springer, 2016, pp. 236–247.

[6] E. Androulaki, G. O. Karame, M. Roeschlin, T. Scherer, and S. Capkun, "Evaluating user privacy in bitcoin," in International Conference on Financial Cryptography and Data Security. Springer, 2013, pp. 34–51.

[7] O. Goldreich and Y. Oren, "Definitions and properties of zero-knowledge proof systems," Journal of Cryptology, vol. 7, no. 1, pp. 1–32, 1994.

[8] E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza, "Snarks for c: Verifying program executions succinctly and in zero knowledge," in Annual cryptology conference. Springer, 2013, pp. 90–108.

[9] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," in 2018 IEEE Symposium on Security and Privacy (SP). IEEE, 2018, pp. 315–334.

[10] J. Zhang and L. Liu, "Publicly verifiable watermarking for intellectual property protection in fpga design," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, no. 4, pp. 1520–1527, 2017.

[11] J. Zhang, T. Xie, Y. Zhang, and 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.

[12] Y. Zhang, S. Wang, X. Zhang, J. Dong, X. Mao, F. Long, C. Wang, D. Zhou, M. Gao, and G. Sun, "Pipezk: Accelerating zero-knowledge proof with a pipelined architecture," in 48th IEEE/ACM International Symposium on Computer Architecture (ISCA), June 2021.

[13] L. Xiang, B. Tang, and 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.

[14] M. Qiu, K. Gai, and Z. Xiong, "Privacy-preserving wireless communications using bipartite matching in social big data," Future Generation Computer Systems, vol. 87, pp. 772–781, 2018.

[15] W. Dai, M. Qiu, L. Qiu, L. Chen, and A. Wu, "Who moved my data? privacy protection in smartphones," IEEE Communications Magazine, vol. 55, no. 1, pp. 20–25, 2017.

[16] B. Bünz, M. Maller, P. Mishra, and N. Vesely, "Proofs for inner pairing products and applications," IACR Cryptol. ePrint Arch, vol. 1177, p. 2019, 2019.

[17] J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and 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.

[18] Y. Gao, S. Iqbal, P. Zhang, and M. Qiu, "Performance and power analysis of high-density multi-gpgpu architectures: A preliminary case study," in 2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems. IEEE, 2015, pp. 66–71.

[19] Y. Guo, Q. Zhuge, J. Hu, J. Yi, M. Qiu, and E. H.-M. Sha, "Data placement and duplication for embedded multicore systems with scratch pad memory," IEEE Transactions on Computer-Aided

Design of Integrated Circuits and Systems, vol. 32, no. 6, pp. 809–817, 2013.

[20] Y. Guo, Q. Zhuge, J. Hu, M. Qiu, and E. H.-M. Sha, “Optimal data allocation for scratch-pad memory on embedded multi-core systems,” in 2011 International Conference on Parallel Processing. IEEE, 2011, pp. 464–471.

[21] G. Alavani, K. Varma, and 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.

[22] M. Qiu, Z. Chen, and M. Liu, “Low-power low-latency data allocation for hybrid scratch-pad memory,” IEEE Embedded Systems Letters, vol. 6, no. 4, pp. 69–72, 2014.

[23] L. Zhang, M. Qiu, W.-C. Tseng, and E. H.-M. Sha, “Variable partitioning and scheduling for mpsoc with virtually shared scratch pad memory,” Journal of Signal Processing Systems, vol. 58, no. 2, pp. 247–265, 2010.