2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications

Acceleration of Anomaly Detection in Blockchain Using In-GPU Cache

Shin Morishima and Hiroki Matsutani
Dept. of ICS, Keio University
3-14-1 Hiyoshi, Kohoku, Yokohama, Japan
Email: {morisima,matutani}@arc.ics.keio.ac.jp

Abstract

Blockchain is a distributed ledger system composed of a P2P network and is used for a wide range of applications, such as international remittance, inter-individual transactions, and asset conservation. In Blockchain systems, tamper resistance is enhanced by the property of transaction that cannot be changed or deleted by everyone including the creator of the transaction. However, this property also becomes a problem that unintended transaction created by miss operation or secret key theft cannot be corrected later. Due to this problem, once an illegal transaction such as theft occurs, the damage will expand. To suppress the damage, we need countermeasures, such as detecting illegal transaction at high speed and correcting the transaction before approval. However, abnormality detection in the Blockchain at high speed is computationally heavy, because we need to repeat the detection process using various feature quantities and the feature extractions become overhead. In this paper, to accelerate abnormality detection, we propose to cache transaction information necessary for extracting feature in GPU device memory and perform both feature extraction and abnormality detection in the GPU. We employ abnormality detection using K-means algorithm based on the conditional features. When the number of users is one million and the number of transactions is 100 millions, our proposed method achieves 37.1 times faster than CPU processing method and 16.1 times faster than GPU processing method that does not perform feature extraction on the GPU.

Keywords

Blockchain, Anomaly detection, and GPU

1. Introduction

Blockchain is a distributed ledger system composed of P2P network proposed by Bitcoin [1]. In Blockchain, sender and receiver can trade directly without trusted third party such as a bank, unlike existing online money transfer system. The most widely-used applications of Blockchain are crypto currencies. They are used for international transactions because of low cost and fast transactions by the direct transactions. Blockchain has various features, such as fault tolerance, tamper resistance, and anonymity. Applications of the the crypto currencies are conservation of assets utilizing fault tolerance and personal transactions that can protect personal information by anonymity. In addition, Blockchain is used for not only crypto currencies but also asset transactions other than currency [2][3], distributed application [4][5], document storage system, and registration of rand.

The data structure of Blockchain is a chain of hash values of blocks each of which contains a set of transactions. In this structure, an update of a transaction causes changes in all the blocks after the block containing the transaction. With this structure, update or delete of a Blockchain transaction is extremely difficult by everyone including the creator of the transaction; thus tamper resistance is high. However, this feature becomes a problem in which Blockchain system cannot modify fraudulent transactions made by miss operations or stolen secret keys. Because of this problem, once an illegal transaction, such as theft, occurs, the damage will expand. To suppress the damage, we need countermeasures, such as detecting illegal transaction at high speed and correcting the transaction before approval.

Pham et al. proposed an abnormality detection method for Blockchain transactions by using K-means clustering, Mahalanobis distance, and local outlier factor [6]. Using this method, actual theft transactions in past transactions of Bitcoin network were detected. However, abnormal transactions that can be detected by specific feature quantities and algorithms are limited. In reality, we need to repeat abnormality detections by changing feature quantities and algorithms, resulting in heavy computations and long computation time. Figure 1 shows the execution time when abnormality detection is executed using K-means algorithm with four feature quantities, which is a similar approach to [6]. When the number of users is one million, the execution time is about 25 seconds, which is longer than 15 seconds of the average block creation time of Ethereum [4] which is one of typical Blockchain systems. The number of users of one million is almost equal to the number of holders of one Bitcoin (equivalent to the market price

978-1-7281-1141-4/18/$31.00 ©2018 IEEE
DOI 10.1109/BDCloud.2018.00047

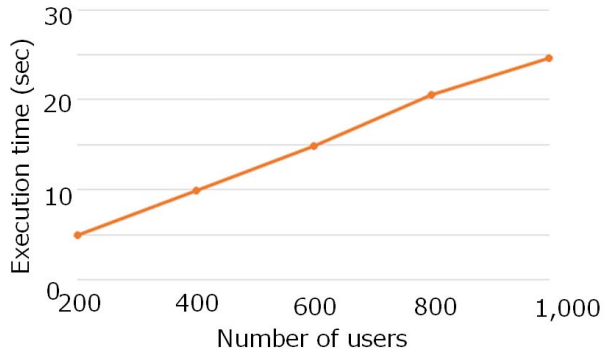

Figure 1. Execution time when abnormality detection is executed using K-means method with four feature quantities (similar approach to [6])

of 6,400 dollars) in the Bitcoin network [7] (in actual operation, more users may be targeted). This result shows that the time for a single abnormality detection exceeds the block creation time, which means that multiple abnormality detections cannot be executed within the creation time. In this paper, to accelerate abnormality detections, we propose to cache transaction information necessary for extracting features in GPU device memory and perform both the feature extraction and abnormality detection in the GPUs. Specifically, we propose to cache a graph structure representing transaction status of users as an array-based data structured suitable for GPU processing. By extracting a feature quantity from a graph as an information source, both the feature extraction and abnormality detection can be executed in the GPUs.

The rest of this paper is organized as follows. Section 2 surveys related work. Section 3 illustrates our proposed In-GPU cache based abnormality detection method for Blockchain network. Section 4 shows evaluation results and Section 5 concludes this paper.

2. Related Work

2.1. Data Structure of Blockchain

Figure 2 shows an overview of data structure of Blockchain. Upper half of the figure shows the overall Blockchain structure consisting of a chain of blocks. A block is a set of transactions, and detail of a transaction is shown in lower half of the figure. A block contains block ID, ID of previous block, and set of transactions. A block ID is a hash value of its entire block excluding the block ID. Because a block contains the previous block ID, the block is connected to the previous block by the hash value. The connection continues back to the genesis block, and the structure looks like a chain; so it is called Blockchain.

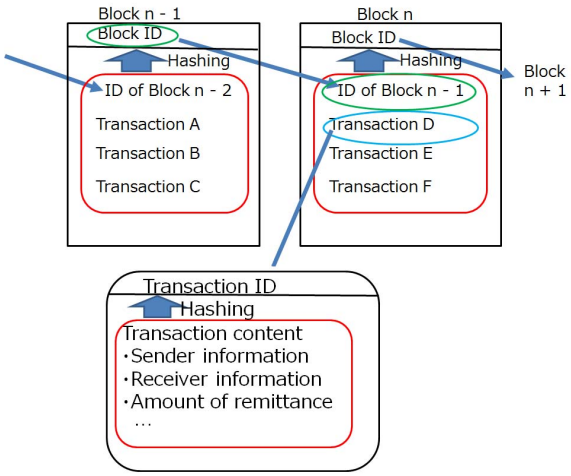

Figure 2. Overview of data structure of Blockchain

In other words, each block ID affects all the subsequent blocks, and an update of a transaction causes changes of all blocks after the block containing the transaction. This feature enhances tamper resistance, because if an attacker tampers a transaction, the attacker must tamper all blocks after the block containing the transaction. Lower half of the figure shows data structure of a transaction. The structure of transaction is similar to the structure of block. That is, it contains transaction ID and transaction detail. A transaction ID is a hash value of its entire transaction excluding the transaction ID.

The write query of Blockchain is always used to generate a new block appended as the latest block of the chain. This means that past blocks and transactions in Blockchain are not updated nor deleted [1]. Blockchain system employs a consensus algorithm, and a block is generated only when the condition based on the algorithm is satisfied, in order to exclude malicious network participants that generate selfish blocks. Regarding the consensus algorithm, several algorithms are proposed, such as PoW (Proof of Work), PoS (Proof of Stake), and PBFT (Practical Byzantine Fault Tolerance) [8].

For approval of a transaction in Blockchain system, the conditions based on the consensus algorithm must be satisfied, and once approved, the transaction cannot be changed even by a creator of the transaction nor system administrators. In addition, illegal transactions (e.g., those by theft secret keys) are not matters of the protocol; thus they cannot be canceled at the time of approval by the consensus algorithm. Please note that there is a time period for

1. Specifically, there is a possibility that a block is disabled due to a chain split or reconstruction. However, as a block becomes older, probability of invalidation becomes lower. In Bitcoin system, the probability that the 6th block before the latest block is invalidated is considered to be almost zero.

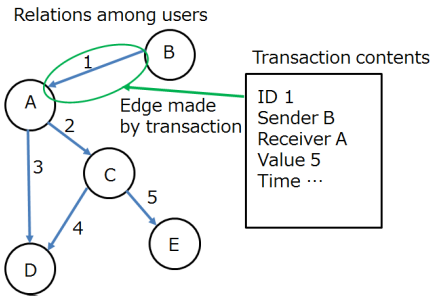

Figure 3. Overview of user graph

a transaction to satisfy the consensus condition (approval of the transaction) after creation of the transaction. Such illegal transactions can be corrected if they are detected during this period. The motivation of this paper is, thus, to detect abnormalities at high speed and high accuracy in order to suppress damage of the illegal transactions.

2.2. Blockchain Search Using In-GPU Cache

In [9], we proposed an acceleration method of Blockchain transaction search using in-GPU cache. In Blockchain systems, most users use Blockchain indirectly via services provided by full nodes, such as exchanges and wallet applications. Therefore, most Blockchain search queries are concentrated to a part of full nodes that provide the services, and thus the search queries at the full nodes become a bottleneck of Blockchain system. We proposed an array-based tree structure taking advantage of GPU processing and the Blockchain-specific characteristics where past transactions are not updated nor deleted. The Blockchain transaction search was accelerated by 3.4 times faster than existing method, which is a general purpose GPU search based on key-value store structure [10][11]. By using the Blockchain transaction search [9] in combination with the Blockchain abnormality detection method proposed in this paper, an entire Blockchain system can be accelerated, because abnormality detection is executed at full nodes that provide the services.

2.3. Abnormality Detection on Blockchain

A method for detecting abnormal transactions in the Blockchain is proposed in [6]. The method uses the user graph that represents user-centric transactions flow. Figure 3 shows an overview of the user graph, where a user is a vertex and a transaction is an edge. As shown in the figure, when there is a transaction from B to A, an edge from B to A is created. The graph structure is expanded by new transactions represented as edges. This user graph is used to extract feature quantities for each Blockchain user. For example, the number of edges that enter a certain vertex is the number of deposit transactions, and the number of edges that leave represents the number of withdrawal transactions. They used six feature quantities, such as the number of deposit transactions, the number of withdrawal transactions, average deposit amount, and average withdrawal amount, for the abnormality detection. Also, they used three anomaly detection algorithms, i.e., K-means clustering, Mahalanobis distance, and Local Outlier Factor. By using the user graph based method, two actual theft cases on June and October in 2011 at Bitcoin network were detected. Please note that many abnormal transactions other than these transactions were detected by the method. However, it is very difficult to judge if they are really stolen cases, because information about most stolen cases would not be disclosed.

The detected theft cases are varied depending on the anomaly detection algorithm used. This means that we need to repeat abnormality detections by changing the algorithms. In addition to the six feature quantities used for the experiment, if we used another feature quantity such as balance, different abnormalities would be detected. The number of all the past transactions in Blockchain is huge and the computation cost for anomaly detection algorithms increases as the number of input samples increases. Thus, in reality, feature quantities are extracted from a range of transactions selected by a given condition for practical abnormality detection. For example, a range of transactions which were created within one year or recent transactions whose cumulated size is less than a certain limit can be used for the feature quantity extraction. Due to various choices of the conditions, computation cost for the transaction extraction becomes large. In this paper, both the feature extraction and abnormality detection are accelerated by GPUs. As a representative algorithm, K-means clustering is implemented and evaluated for the abnormality detection of Blockchain. Please note that our proposal method can be applied to the above algorithms other than K-means clustering.

3. Abnormality Detection Using In-GPU Cache

3.1. Overview of Proposed System

All the transaction information of Blockchain cannot be stored in the GPU device memory which is much smaller than host memory. For example, NVIDIA GeForce GTX 980 Ti has 6GB device memory, while size of Bitcoin transaction data is about 160GB. There are two general approaches for the abnormality detection with a limited GPU device memory. One approach is to repeat the processes in which a part of transactions are sent to GPU and abnormality detection on that part is performed using the GPU. Another approach is that feature quantities are extracted by CPU and then abnormality detection is done

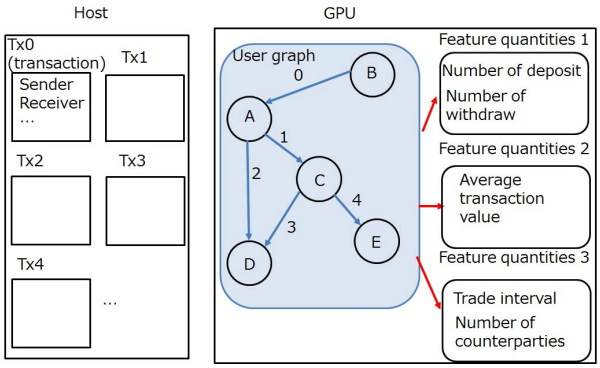

Figure 4. Overview of abnormality detection system using In-GPU cache

by GPU. For the first approach we need to consider the overhead for CPU-GPU data transfers, and for the second approach the issue is high computation cost for feature extractions by CPU. On the other hand, in this paper, we propose a method to perform both the feature extraction and abnormality detection on the GPU. In the proposed method, to extract feature quantities by GPU, a modified user graph that has information of transactions necessary for the extraction is used. The user graph can be cached in the GPU device memory, because the size of user graph can be reduced by eliminating some information unnecessary for feature quantity extraction, such as signature of the sender. By caching the user graph in the GPU device memory, both the feature extraction and abnormality detection can be performed by the GPU.

Figure 4 shows an overview of abnormality detection system using the In-GPU cache. As shown in the left side of the figure, all the transaction information is stored in the host main memory. The user graph is created based on the information of the sender and receiver, the necessary information for feature extraction is added to the graph, and the graph is cached in the GPU device memory. Detail about the graph creation method is described in Section 3.2. As shown in the right side of the figure, various feature quantities are extracted from the graph cached in the GPU device memory, and then abnormality detection is performed by the GPU. When a different algorithm for abnormality detection is performed using the same set of features, the extracted features can be reused to reduce the overhead of feature extraction. If an abnormality detection is performed using different features but free space to extract new features is not available in the GPU device memory, then old feature quantities are deleted from the In-GPU cache. Thus, both the feature extraction and abnormality detection are performed by the same GPU in the system. The proposed system can accelerate the abnormality detection of Blockchain compared to naive approaches that impose CPU-GPU data transfer overhead or heavy computations for feature extraction by CPU.

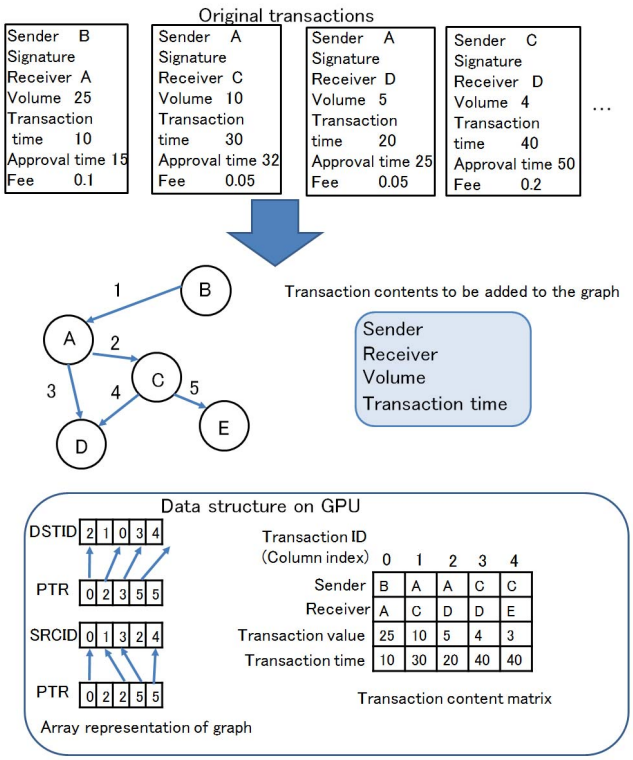

Figure 5. Data structure of user graph in In-GPU cache of proposed system

3.2. Data Structure of User Graph in GPU

Figure 5 shows the data structure of a user graph cached in the GPU device memory. The upper part of the figure shows transaction information in the host. At first, a user graph is created based on the sender and the receiver information from the original transactions. The graph created from the information is shown in the left side of the middle figure. In this graph, a vertex represents a Blockchain address, and an edge represents a transaction from the sender to the receiver. If a transaction includes multiple senders and receivers, edges are created for all the combinations of sender and receiver pairs.

The data structure for caching the graph in the GPU is shown in the lower left side of the figure. The graph is represented using a compressed row storage structure called CSR (Compressed Sparse Row) to be efficiently accessed by the GPU. CSR usually consists of two arrays: an array representing the destinations of edges (DST array) and that pointing sources of the edges (PTR array). The destination of an edge represents a receiver of the transaction and the source represents a sender. In the proposed

system, because a receiver can be found by referring to the transaction content, the transaction ID is stored in the DST array instead of the destination of the edge; so we call the array DSTID in this paper. Using these arrays, a withdrawal information from a sender can be found by searching from the PTR array, because transaction information is stored in these arrays in the sender's point of view. If the PTR array points outside of the DSTID array, it means there is no withdrawal from that vertex. On the other hand, an overall graph search is needed to obtain a deposit information to a receiver. In the proposed system, two additional arrays, where directions of edges in the PTR and DSTID arrays are reversed, are introduced to search a deposit information efficiently. Specifically, SRCID array and its PTR array are added. That is, the SRCID array represents senders of transactions and its PTR array points receivers of the transactions. Using these arrays, a deposit information to a receiver can be found by searching from the PTR array. In this structure, thus, deposit and withdrawal information can be searched bidirectionally by using the DSTID array for the withdrawal and the SRCID array for the deposit information.

Regarding transaction information, only the items related to feature extraction are stored in the In-GPU cache. A cached transaction information is shown in the right side of the middle figure. In this example, the sender, transaction amount, and transaction time are cached. Besides this, we can cache other information, such as block number, balance, and so on. As the information stored in the In-GPU cache is increased, although the data size increases, various features can be extracted. We can decide which information to be extracted, by considering the capacity of the GPU device memory. This content is created for each edge in the user graph, not for each transaction. That is, when there are multiple senders and receivers in one transaction, all the combinations of sender and receiver pairs are created as edges. The structure is represented as a matrix, where column index is used as transaction ID. Each column represents transaction contents on an edge (i.e., a transaction ID) of the user graph, and each row represents a specific value of the transaction contents such as transaction amount. Let i be the column index of an interested transaction ID in the matrix. By using i as the transaction ID in the DSTID and SRCID arrays, the transaction contents on each edge can be obtained from the user graph. The proposed method to extract feature quantities used for abnormality detection using this structure will be described in the next subsection.

3.3. Feature Extraction in User Graph

Figure 6 shows a flow chart for extracting feature quantities from the In-GPU cache with the structure described in Section 3.2. First of all, it is judged whether the feature quantities to be extracted are related to deposit or withdrawal. The graph structure of the SRCID array is used in the case of a deposit and that of the DSTID array

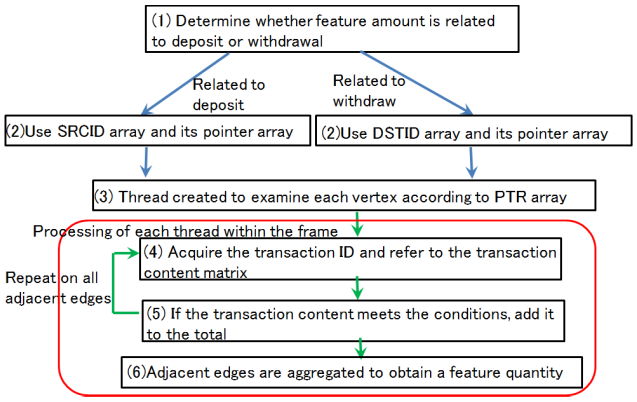

Figure 6. Processing flow of extracting feature quantities from In-GPU cache

is used in the case of a withdrawal. Next, GPU threads are launched according to the number of vertices (i.e., the number of users), and each thread searches a single element of the PTR array in parallel. Each thread refers to the SRCID or DSTID array from the corresponding PTR array and acquires a transaction ID representing a column of the transaction content matrix. Then, each thread refers to the row of the transaction content matrix according to the condition, acquires the element from the row related to the feature quantity if the condition is met, and aggregates the feature quantities on the basis of this element. By repeating these steps (4) and (5) in Figure 6 on all the edges adjacent to the vertex, the feature quantities based on all transaction information related to a specific user can be extracted. These steps represent the processing for extracting one feature quantity. By repeating these steps for each feature quantity, multiple features necessary for abnormality detection can be extracted. When multiple features are necessary, the conditional judgement is performed only in the extraction steps for the first feature; in the extraction steps for the other features, the conditional judgement results for the first feature can be reused.

Figure 7 shows as an example of feature extraction described in Figure 6. In this example, an average of all the deposit amounts which are greater than or equal to 15 is extracted. Each number in the figure is corresponding to each step in the flow chart shown in Figure 6. First, since the target is payment, the SRCID array is used as user graph. Next, a thread is launched for each element of the PTR array, and each element of the DSTID array is referred to from the PTR array. Since each content of the DSTID array is the transaction ID that represents the column number of the transaction content matrix, the deposit amount can be extracted from the column on the matrix. If the deposit amount is 15 or more, the total deposit amount and the number of deposit transactions are summed up. These steps are repeated until all the edges

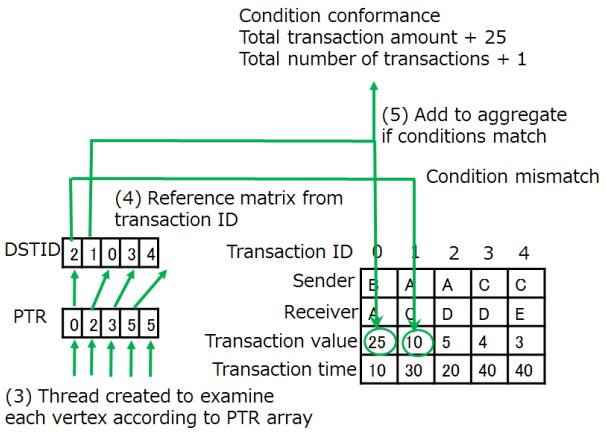

Figure 7. Example of feature extraction

adjacent to the vertex in charge are completed. Finally, the average deposit amount is obtained by dividing the total deposit amount by the deposit number. Although only one thread is shown in detail in the figure, each thread executes these steps in parallel at all the vertices. When the processing of all the threads is completed, an array can be created that represents an average deposit amount for each user. The number of elements in the array is equal to the number of vertices which means the number of users.

3.4. Abnormality Detection Using Extracted Feature Quantities

By executing various abnormality detection algorithms for the feature quantities extracted in Section 3.3, anomaly users in Blockchain can be detected. In this paper, as an example of the algorithms, an abnormality detection using K-means clustering is implemented and evaluated. Originally, K-means is not an algorithm for abnormality detection, and it is typically used for classifying a large number of data into clusters. When K-means algorithm is used as abnormality detection, feature quantity of a user that is far from the center of gravity of the cluster is regarded as abnormal after the K-means clustering. The abnormality detection using K-means clustering when the number of clusters is k is done by the following steps.

1) An initial cluster is randomly assigned to the feature quantity vector $x_i (i = 1, ..., n)$ of each vertex.
2) Calculate the cluster center of gravity $V_j (j = 1, ..., k)$.
3) Calculate the distance between each x_i and each V_j. Then x_i belongs to the cluster j of the nearest V_j.
4) Steps 2 and 3 are repeated until the process is converged. The convergence is detected when the number of iterations or total amount of changes exceeds a given threshold. Then the clustering is completed.
5) Calculate the distance between x_i and the center of gravity V_j of its cluster. x_i is detected as abnormal if the distance exceeds a given threshold.

When using the feature extraction method described in Section 3.3, the feature vectors are created as an array structure, which is suitable for GPU processing. Then GPU threads are launched according to the number of elements (i.e., number of vertices) of the array. Since each thread processes an element in parallel, the procedure can be accelerated by the GPU. Specifically, in the step 2, the center of gravity is obtained by calculating the average of the feature quantities of the respective clusters by the reduction operation. In the distance computations in the steps 3 and 5, the distance for each vertex is computed by each thread in parallel. These steps consist of numerical operations using the array structure. Even if we employ anomaly detection algorithms other than K-means clustering, a similar parallel processing can be applied by changing the operations.

4. Evaluations

4.1. Evaluation Environment

All evaluations are conducted with the same machine. The processor is Intel Xeon E5-2637v3 running at 3.5GHz and memory capacity is 256GB. A single NVIDIA GeForce GTX 980 Ti GPU is used for the GPU processing. Table 1 lists the specification of the GPU. We use CUDA version 7.5. To evaluate the scalability of the proposed system, we used synthetic transaction data that generated transactions at random by changing the average number of transactions per user from 20 to 100. We evaluated the number of users as 500 thousands and one million. As an anomaly detection algorithms, anomalies were detected by K-means clustering. Four feature quantities are examined: payment number, withdrawal number, average deposit amount, and average withdrawal amount. The following three methods are evaluated in terms of the execution time.

- GPU only: The proposed method using In-GPU cache with feature extraction and abnormality detection by GPU.
- CPU only: A method extracting feature quantities and detecting abnormalities by CPU.

Table 1. GPU specification used in the experiments

	GeForce GTX 980 Ti
Number of cores	2,816
Core clock	1,038MHz
Memory clock	7,010MHz
Memory datapath width	256bit
Memory capacity	6GB

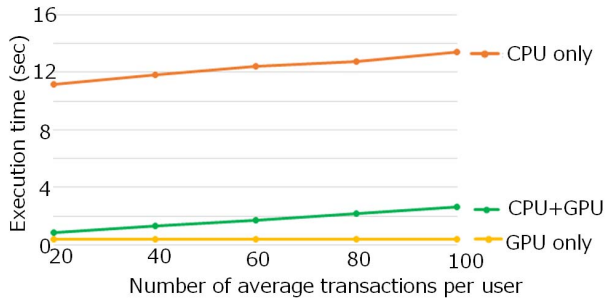

Figure 8. Execution time in the case of extracting feature quantities without condition and performing abnormality detection for 500 thousands users

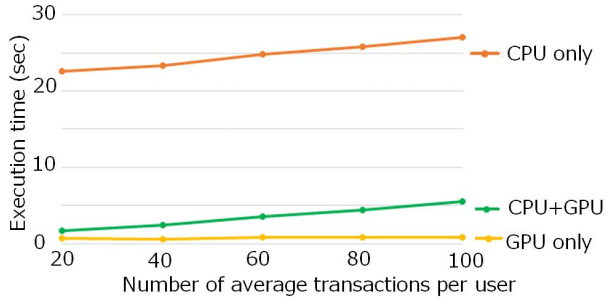

Figure 9. Execution time in the case of extracting feature quantities without condition and performing abnormality detection for one million users

- CPU + GPU: A method extracting feature quantities by CPU, transferring the feature quantities to GPU, and performing abnormality detection by the GPU.

For fairness of evaluation, we also created a user graph capable of extracting feature quantities on the host memory to extract feature quantities even when extracting features using the CPU. In addition, to evaluate the influence of adding a condition at the time of feature quantity extraction, we evaluated two cases: cases with and without a condition on the transaction amount for the extraction.

4.2. Execution Time of Abnormality Detection Using Feature Quantities without Condition

Figures 8 and 9 show the execution time in the case of extracting feature quantities without conditions and performing abnormality detection. The numbers of users are 500 thousands and one million in Figures 8 and 9, respectively. Based on [6], the number of clusters in the K-means method is seven and the number of repetitions of cluster division is 50 in both the cases. As shown in Figure 8, both CPU + GPU and GPU only are faster than CPU only. However, as the average number of transactions increases, the execution time of CPU + GPU increases, and the speed-up rate of the proposed method increases. Since the features extraction process is performed for each transaction, the time for the feature extractions is proportional to the average transaction number. On the other hand, since the number of extracted features do not change unless the number of users changes, the execution time of abnormality detection does not change.

Comparing Figure 8 and Figure 9, the number of users increases and the total transaction volume also increases, so the execution time increases both in feature extraction and in abnormality detection. The tendencies, such as the speed-up rate, are almost the same in Figures 8 and 9. When the number of users is 500 thousands and the average number of transactions is 100, the proposed method is 31.5 times faster than the CPU only and 6.3 times faster than CPU + GPU. When the number of users is one million and the average number of transactions is 100, the proposed method is 32.6 times faster than CPU only and 6.6 times faster than CPU + GPU.

4.3. Execution Time of Abnormality Detection Using Feature Quantities with Condition

Figures 10 and 11 show the execution time when abnormality detection is performed by extracting the features with the condition that the transaction volume is equal to or more than the threshold value. The numbers of users are 500 thousands and one million in Figures 10 and 11, respectively. As shown in Figure 10, the two GPU based methods (GPU only and CPU + GPU) are faster than the CPU only case, as well as the unconditional case (Figure 8). Also, as for the CPU + GPU case, the execution time increases as the average transaction number increases.

However, by setting conditions, the amount of computation for feature extraction increases, so the increase in execution time is larger than the case without condition. The same tendency can be seen in Figure 11 where the number of users is one million. As a result, when the number of users is 500 thousands and the average number of transactions is 100, the proposed method is 41.6 times faster than the CPU only and 17.1 times faster than the CPU + GPU. When the number of users is one million and the average number of transactions is 100, the proposed method is 37.1 times faster than CPU only and 16.1 times faster than CPU + GPU. Compared to the case without conditions, the speed-up rate is significantly improved, especially when compared with CPU + GPU.

In this evaluation, we assumed one-to-one transactions between one sender and one receiver, so the number of transactions is equal to the degree of the user graph. In the case of one-to-many or many-to-many transactions, edges are created for all the combinations of sender and

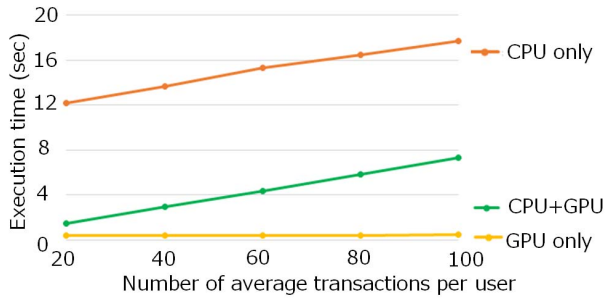

Figure 10. Execution time in the case of extracting feature quantities with condition and performing abnormality detection for 500 thousands users

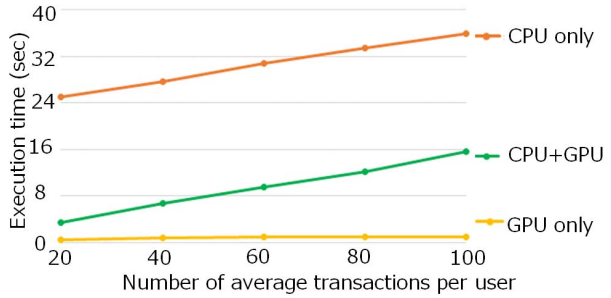

Figure 11. Execution time in the case of extracting feature quantities with condition and performing abnormality detection for one million users

receiver pairs. In this case, the computation cost of the feature quantity extraction is proportional to the number of edges of the user graph, not the number of transactions. The average number of transactions in an actual Bitcoin network is about 16, which is smaller than this evaluation. However, the average transaction size is about 500 bytes. In this size, the minimum number of participants (senders and receivers) in the average-sized transaction is five (i.e., four senders and one receiver [2]). Therefore, the average degree of the user graph is estimated as at least 64 (= 16 $\times$ 4).

5. Conclusions

Since Blockchain systems cannot modify approved transactions, it is necessary to detect abnormal transactions at high speed and high accuracy in order to suppress the damage of illegal transactions by countermeasures, such as correcting transactions before approval. However, to detect the abnormalities with high accuracy, we need to repeat the detections using various features, so extraction of the feature quantities becomes a bottleneck for high speed abnormality detection. Therefore, in this paper, to accelerate abnormality detection, we propose to cache transaction information necessary for extracting feature quantities in GPU device memory and performing both the feature extraction and abnormality detection by the GPU. In the experiments, we conducted abnormality detection using K-means clustering with considering a condition of transaction amount. When the number of users is one million and the number of transactions is 100 million, the proposed method is 37.1 times faster than the CPU processing and 16.1 times faster than the conventional GPU method. Even in the case with no condition on the feature extraction, the proposed method is 31.5 times faster than CPU processing and 6.3 times faster than the conventional GPU method.

2. Information size of a sender is larger than that of a receiver, since a signature is included in the sender information. If we assume more receivers, the number of participants will increase.

Acknowledgements This work was supported by JSPS KAKENHI Grant Number JP16J05641 and JST CREST Grant Number JPMJCR1785, Japan.

References

[1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," https://www.bitcoin.com/bitcoin.pdf.

[2] "Counterparty," https://counterparty.io/.

[3] A. Nordrum, "Wall Street Occupies the Blockchain - Financial Firms Plan to Move Trillions in Assets to Blockchains in 2018," *IEEE Spectrum*, pp. 40–45, Sep. 2017.

[4] "Ethereum Project," https://www.ethereum.org/.

[5] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor, "Making Smart Contracts Smarter," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, Oct. 2016, pp. 254–269.

[6] T. Pham and S. Lee, "Anomaly Detection in Bitcoin Network Using Unsupervised Learning Methods," *Compution Research Repository*, vol. abs/1611.03941, pp. 1–5, Nov. 2016.

[7] "Bitcon Rich List," https://bitinfocharts.com/top-100-richest-bitcoin-addresses.html.

[8] D. T. T. Anh, W. Ji, C. Gang, L. Rui, O. B. Chin, and T. Kian-Lee, "BLOCKBENCH: A Framework for Analyzing Private Blockchains," in *Proceedings of the International Conference on Management of Data*, May 2017, pp. 1085–1100.

[9] S. Morishima and H. Matsutani, "Accelerating Blockchain Search of Full Nodes Using GPUs," in *Proceedings of the 26th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP'18)*, Mar. 2018.

[10] T. H. Hetherington, T. G. Rogers, L. Hsu, M. O'Connor, and T. M. Aamodt, "Characterizing and Evaluating a Key-value Store Application on Heterogegenenous CPU-GPU Systems," in *Proceedings of the International Symposium on Performance Analysis of System and Software*, Apr. 2012, pp. 88–98.

[11] T. H. Hetherington, M. O'Connor, and T. M. Aamodt, "MemcachedGPU: Scaling-up Scale-out Key-value Stores," in *Proceedings of the ACM Symposium on Cloud Computing*, Aug. 2015, pp. 43–57.