MULTI-THREADING AND SHARED-MEMORY POOL TECHNIQUES FOR AUTHORIZATION OF CREDIT CARD SYSTEMS USING JAVA

SITI HAFIZAH AB. HAMID[1]
MOHD HAIRUL NIZAM M. NASIR[2]
WONG YEW MING[3]
HAZRINA HASSAN[4]

Faculty of Computer Science and Information Technology
University of Malaya, 50603 Kuala Lumpur, Malaysia
[1]sitihafizah@um.edu.my
[2]hairulnizam@um.edu.my
[3]yew.ming@yahoo.com
[4]hazrina.hassan@yahoo.com

Abstract. This research paper presents a framework and solution for improving the efficiency of the authorization processing of credit card transactions using multi-threading and shared-memory pool techniques. Through the use of both techniques, a prototype of a real-time multi-threaded authorization system has been developed with Java platforms to overcome the slow sequential authorization processing of a single-threaded model of current credit card authorization systems. Via multi-threading technique, it allows parallel execution of the validation functional units involved during the authorization process of credit card transaction through multiple threads. It also enables a separate thread to be executed in the background of the process to perform data synchronization maintained in the shared-memory pool with the main system database. Shared-memory pool has been used to provide a global point of access to the card information kept in the random access memory. During the authorization process, the respective worker thread performs a binary search to obtain the authentication data from the shared memory instead of the system database to hasten the authorization process of credit card transactions. Performance-testing has been carried out to measure the efficiency of a fixed number of credit card authorization processes running between the single-threaded and the multi-threaded authorization systems in a workstation using similar hardware capabilities. Specially-embedded tools are incorporated in the payment gateway applications to obtain the length of end-to-end execution.

Keywords: credit card, authorization system, multi-threading technique, shared-memory pool

(Received May 05, 2008 / Accepted July 16, 2008)

1 Introduction

Credit card authorization is a process whereby the card issuer decides whether to approve or decline requests to accept transactions performed by a cardholder based on a series of validation of card risk-management profiles to verify that the cardholder's account is open, the transaction amount is within the available credit limit and comes from the legitimate card, and many other related validation parameters [8]. The validation of a card's risk management profile can be classified in two categories, namely card restriction validation and online fraud validation [1].

Card restriction validation includes financial and non-financial verification related to the card whereas online fraud validation involves cryptographic operation through a host security module (HSM) to verify the se-

curity aspect of the authorization in order to determine the legitimacy of the card. HSM is an external device connected to the authorization host that keeps the card issuer's secret information in tamper-resistant hardware which is used to perform verification of the credit card transaction[19]. Owing to various validations during the authorization process for each transaction, it takes some time for a whole process to be completed. With old payment-processing methods of the conventional system, credit card transactions take longer during authorization processing [16].

This paper looks at the current issues surrounding credit card authorization processes. It concludes that a multi-threaded authorization system with shared-memory pool is needed to improve the response time of the process and to overcome the slow sequential authorization processing problem of a single-threaded model for current credit card authorization systems. The proposed multi-threaded authorization system was developed with JAVA, and the performance of the multi-threading implementation measured.

This paper is divided into seven sections. The first introduces the credit card authorization process in general and highlights some issues. The second section gives a brief overview of various methods of authorization, while the third section discusses issues relating to the current credit card authorization process. The fourth section presents a system analysis and architectural design of a real-time multi-threaded authorization system. The sixth section gives an evaluation in terms of performance between multi-threaded and single-threaded authentication engines. Section 7 concludes.

2 Overview

There are various methods proposed for improving the response time of the authorization process of credit card transaction. These include invention of host security modules (HSM), implementation of distributed authorization systems, utilization of cardholder-initiated transactions devices and deployment of digital network access system devices. Each method is elaborated in the following sub-sections.

2.1 Host Security Module (HSM)

HSM is the external device which is used to securely generate and store long-term secrets for use in cryptography and physically protect the access to and use of those secrets over time. These secrets include the private keys used in symmetric key protection and public key cryptography. HSM is implemented because hardware implementation is the only way to achieve speeds beyond the reach of general-purpose microprocessors [4].

HSM is therefore used as a cryptographic accelerator to hasten the intensity of mathematical operations, especially in public key encryption, and provide better performance than normal software-based cryptographic systems [7]. The functionalities of HSM include verification of an on-line Personal Identification Number (PIN) by comparison with an encrypted PIN block, validation of credit card transactions by the checking of card security codes and performance of a host-processing component of a Europay MasterCard Visa (EMV) based transaction. HSM also supports cryptographic operations in smart card application during personalization and performs PIN block translation that involves encryption and decryption processes. The only problem with HSM apparently is that there is no global standard in the low-level communication data exchange protocol owing to the re-engineering cost and marketdominance. Hence, there are only common principles shared among HSM software developers and the current available credit card authorization systems have been tied up to specific HSM types for cryptography processing.

In recent years, the introduction of an HSM that supports Ethernet devices is gaining popularity because of its higher speed of data transmission during cryptographic processing [14]. In short, HSM provides the industry with a leading performance which significantly reduces credit card transaction processing time and lowers the cost per transaction [18].

2.2 Distributed Authorization System

A patented method of distributed authorization system has been proposed in the last decade to accelerate the authorization process. This distribution authorization system utilizes a host computer communicating with a network of remote electronic terminals from the host computer. It includes storing negative file data in the electronic terminal containing information used to identify accounts for which requested transactions are to be denied, and storing authorization file data in the electronic terminal containing information used to determine whether to authorize a requested transaction. Upon entry of a transaction request, the data are checked against the terminal negative file data and immediately denied if the card account is contained in the terminal's negative file. If the transaction is not denied, authorization logic is performed in the electronic terminal, resulting in terminal output denying the request, authorizing the

request, or establishing an electronic connection from the terminal to the host computer to obtain authorization from the host computer. In the establishment of this connection, account data are transmitted from the host back to the remote electronic terminal, resulting in terminal output either denying the request or authorizing the request. Also, during such connection, the terminal's authorization file is updated with account data, transmitted from the host computer to the electronic terminal. The completed transaction is stored in a terminal transaction queue file residing in the terminal for subsequent transmission to the host computer, and for use with a transaction request is subsequently entered at the terminal for the same account [10].

The increasing number of terminals and credit cards, however, will increase the network traffic and it is costly to maintain this information at the network level. Moreover, the card issuer has less control over the authorization profile. This would result in some information not being updated instantly into the network and could cause a bad credit account. There is also a higher potential risk of fraud that would cause financial loss in the event of a lost card.

2.3 Cardholder-Initiated Transaction Device

This approach allows end-user cardholders by means of their own card devices to authenticate POS terminal devices in a way substantially different from the existing EMV protocol. The EMV protocol is often used for authenticating user transmissions to Point-of-Sales (POS) terminal devices. In contrast, the invention performs authentication of the parties to a prospective transaction at the same time that it also transfers the message data necessary to carry out the authorization of the transaction through the POS terminal device. If both of the authentications are successful, the exchanged authentication data and transactions data sent between devices is used to complete the transaction. By this technique, the authentication of the card and terminal greatly reduce the time required to perform the transaction [15].

In this approach, three sets of messages, namely purchase request message, invoice message and acknowledgement message, each comprising a series of data packets, are transmitted to effect a financial transaction. This approach lets the card device initiate randomized challenges included in the purchase request message to the terminal. Then the terminal returns an authentication reply included in the invoice message. Next, cardholder apparatus validates the terminal authentication reply and sends an authenticated response to the financial transaction terminal, where it is yet again validated through real-time online authorization. The response of the authorization is sent through an acknowledgment message to complete the transaction. This approach claims to reduce to 25 percent the usual time taken to complete an electronic transaction which averages 15 to 30 seconds.

2.4 Digital Network Access System Device

The approach generally describes a system for data transmission across what are commonly known as telephone lines, and more particularly, to a system for authorization of financial card transactions. Retail establishments are usually equipped with a terminal containing a modem which is connected to a local telephone line. A portion of the credit card is normally passed through a slot in the terminal at which time identification information is collected from the card. The terminal then automatically dials a previously programmed phone number to begin the authorization process. The number called by the authorization terminal will be an answering modem and this answering modem may be connected to a packet multiplexer that may be connected through another line to a host computer. When the answering modem takes the call from the authorization terminal modem, the identification information is transmitted across the line. The host computer processes this information and transmits back to the authorization terminal whether or not the credit card has been approved for this transaction.

Even if the telephone number which the authorization terminal modem dials is a local number, the retail establishment is still charged a nominal rate for each local call. In this regard, the modems were provided with ground start interfaces which allowed the call coming from the retailer to be answered by the answering modem before the answering modem actually rang [5].

This approach takes advantage of relatively new trunk interfaces known as feature groups. The retailer's authorization terminal modem initiates a call through a local exchange office and from the exchange office the call is directed to an access tandem (AT) switch to gain access to long-distance service. From the access tandem, the information is transferred across a feature group trunk to a Network Access System (NAS) device which demultiplexes and demodulates the signal. NAS works in conjunction with a plurality of asynchronous RS-232 interfaces and one or more micronodes which act as a packet switch for formatting and error-checking. NAS and the micronodes system are entirely digital and do not require analog-to-digital conversion capabilities.

From the micronodes, the transaction data are trans-

ferred through a value-added network protocol such as TCP/IP and are ultimately received by a host computer. The usage of the digital device results in a faster processing time that translates into less usage of telephone lines and therefore less cost per call.

3 Issues Relating to Current Credit Card Authorization Process

The common emphases of the authorization process of credit card transactions are performance and security. The performance aspect concerns the time taken to authorize and complete a sales transaction whereas the security aspect is concerned with fraud prevention and confidentiality of financial information [1]. With increasing numbers of account and transaction volumes, these two aspects remain a major dilemma for the credit card authorization process.

3.1 Performance versus Security

Current research focuses on the security aspect of the authorization process of credit cards. This is because the number of fraudulent cases is growing dramatically and it has become a serious problem faced by credit card issuers. In 2004, credit card transactions had a total loss through fraud of 800 million dollars in the United States while in the United Kingdom, the loss amounted to 425 million pounds [17].

In [9], various fraud detection techniques have been proposed to combat fraud such as the use of smart cards and also implementing fraud detection systems using data mining techniques., Increasing security will, however, bring a downside to performance when it is implemented using more advanced technology. The tradeoff for the authorization process when security is implemented with advanced techniques like the smart card means higher transmission bytes to the server and longer processing time to perform verification. According to an article in Motor Traders, the Managing Director of ProJET Malaysia, Matthew Selbie, has said that chip-based transaction will take a second or two longer than the usual magnetic stripe transaction to complete verification after deployment of the new devices to accept chip-based transactions in the petrol stations [20]. Besides that, implementation of advanced risk analysis techniques using the computer intellect will also contribute to the processing time, which may result in performance degradation.

The size of the database for managing the authentication data is also increasing enormously with the usage of more advanced technology such as the smart card. Achievable performance levels off relatively quickly when the dataset is increasing. As a result, the verification performance decreases monotonically and appears to saturate when database size increases [3].

3.2 Performance versus Volume

According to Bank Negara Malaysia's (BNM) Annual Report [2], the number of credit cards in circulation in Malaysia reached a total of 6.6 million at the end of 2004 with total transactions amounting to RM34.9 billion. In recent years, there has also been a dramatic growth in credit card usage among college students. It can be seen that the credit card usage is not only restricted to elite groups, as this phenomenon is spreading among graduates [11].

The credit card authorization systems that most banks are using are more than fifteen years old, hard-coded, rigid and time-consuming to change. Furthermore, many of these systems are at capacity and struggling to keep up with the large increase in card payment volume. Many systems lack embedded business rules or workflow engines, resulting in, among other things, inefficient risk management operations [13]. As a consequence, some of the transactions have no chance of being processed with conventional architecture design during high simultaneous transaction flows.

According to Tim Kelly, director of TSYS, transaction delays in the COBOL-based programs running on mainframe affect the business hugely when the transaction flow is high [12]. To cater for this scenario, some of the banks have begun to upgrade the existing card processor application to a new enhanced processing platform. For instance, one of the largest banks in Germany, VÖB-ZVD Bank, has appointed Atos Origin to implement its new authorization solution named Worldline Pay. With the implementation of the new solution, VÖB-ZVD Bank hopes to achieve high performance authorization platforms that enable the bank to meet the demands of the market and the clients, and can reliably handle the future number of transactions [6].

Many banks are using home-grown authorization systems that are more than fifteen years and in need of functional and technical upgrades. The card authorization systems that most banks have in place are rigid, at capacity in terms of account and transaction volume and difficult to change in the face of changing regulations and market conditions [13]. Currently, there are a few big market players providing authorization system solutions to the credit card companies. Most of these authorization systems are parameter-driven in order to give

flexibility to the authorization process and meet the demand of the market [13]. There is still, however, room for improvement, as indicated in the latest industry survey report on the payment solution to cater for payment transaction volume.

Based on the existing research, the current credit card authorization systems do not utilize the multi - threading technique as part of their architecture design. Most of the systems are using Oracle as their database management system and none is using the shared-memory pool for authorization purposes. Apart from that, advanced language such as Java is not the most commonly-used in the current architecture of credit card authorization systems.

Performance is therefore still an issue that requires improvement, given the increasing number of transactions and implementation of greater security features. Moreover, there are many home-grown credit card authorization systems still using old technologies to perform authorization that could not support high transaction flow. Multi-threading should therefore be deployed as one of the techniques to improve the response time of the credit card authorization process, since modern operating systems with advanced multi-core processors have supportive multi-threading implementation.

4 System Analysis and Architecture

The functionalities of a proposed authorization credit card system can be categorized in two main broad components, namely front engine and back office components. The front engine component is the authentication engine of the credit card authorization system. This component consists of four modules, namely listener module, worker thread module, authorization module and shared-memory module. The listener module contains functionalities that include activate listener service, activate worker thread-pool, activate child thread-pool, activate shared-memory pool and accept socket connection. The worker thread module contains functionalities that include handle socket connection, parse authorization message, display authorization message, update authorization message, build authorization message, save authorization message, update card balance, save card changes and close socket connection. The authorization module contains functionalities related to card restriction validation and online fraud validation. Card restriction validation consists of check card existence, check card status, check card activation status, check card expiration date, check card usage and check card balance, whereas online fraud validation consists of check card security code, check card identification number, check personal identification number, check chip application cryptogram. The functionalities of online fraud validation are performed through child threads. Shared-memory module contains functionalities that include activate synchronization service, search modified card information and update card information.

On the other hand, the back office component stores the authentication data used in authorization of credit card systems. This component consists of user management and card management. The functionalities related to the user management include display user information, save user information and validate user information, whereas card management consists of display card information, display card activity, display card history, save card information, update card information, search card information and save card changes.

5 Architectural Design

The architectural design of multi-threaded authorization engines of credit card systems consists of front engine and back office. These two components will interact with the system database to store and retrieve application-related data. Apart from the system main components, there are a few sub-systems that have communication with the authorization of credit card system and include host security module (HSM) server, point-of-sale (POS) server, automated teller machine (ATM) server and electronic commerce (E-Commerce) server. The architectural design of multi-threaded authorization engines of credit card systems consists of front engine and back office. These two components will interact with the system database to store and retrieve application-related data. Apart from the system main components, there are a few sub-systems that have communication with the authorization of credit card system and include host security module (HSM) server, point-of-sale (POS) server, automated teller machine (ATM) server and electronic commerce (E-Commerce) server.

All these sub-systems will communicate with authorization of credit card through TCP/IP protocol. The message format that is used for communication between the authorization system and HSM server is specific proprietary command, whereas for the other sub- systems the message format that is used to communicate with the authorization system is ISO 8583. ISO 8583 is the standard interchange message specification defined by the International Organization for Standardization (ISO) for electronic transactions made by cardholders using payment cards.

5.1 How Does Multi-Threaded Architecture Work?

As illustrated in Figure1 below, thread-pool models have been used to handle concurrent authorization requests from the payment gateway and a shared- memory pool is implemented in conjunction with the multi-threading technique to hasten the authorization processing. A shared-memory pool is implemented in this project to reduce the time spent searching card information from the system database, which involves expensive I/O operation compared with obtaining similar information through a shared- memory pool stored in random access memory by use of a binary search.

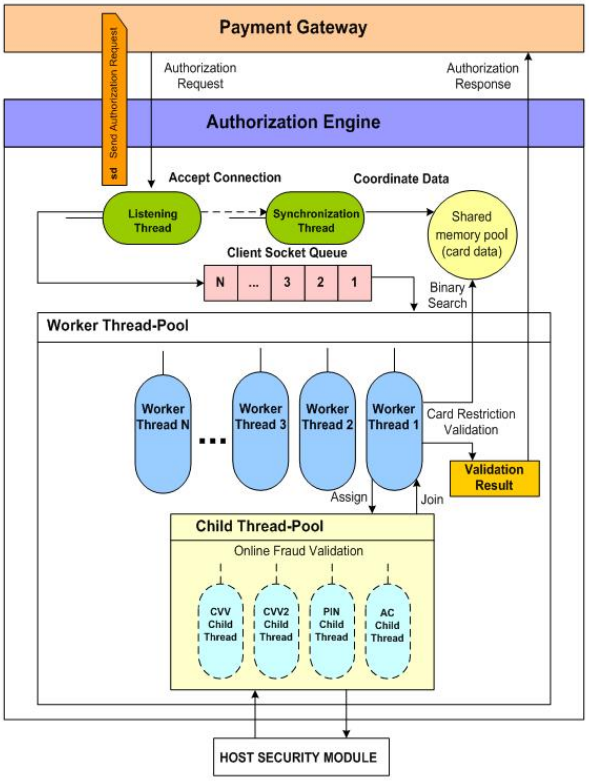

Figure 1: Multi-threaded Authorization Engine

There are two thread-pools implemented in the system, namely worker thread-pool and child thread-pool. When listener service is activated, all the worker threads and child threads are constructed and started in their related thread-pools through listening thread. Additionally, all the card information is loaded to the shared-memory pool before the authorization request can be serviced. The worker threads in the pool are combined with a work queue. The work queue signals waiting worker threads each time a new authorization job arrives to get the relative waiting threads to process the authorization request immediately. Each authorization job is mapped to a client connection. The assigned worker thread gets a socket from the queue and serves the request on that socket until connection is closed. Once an authorization job is accepted, the worker thread will acquire mutex lock not only to synchronize the access to the shared data area but also to accelerate the processing in thread-pool environment. In avoid starvation situation, the timer has been set to release the mutex after a pre-defined period elapses.

The worker thread assigned to each authorization process of credit card transaction will begin to read raw buffer message in ISO8583 format from the socket connection accepted and proceed with message parsing to obtain all the elements. Once the message is parsed, the worker thread will perform card restriction validation and online fraud validation based on the element present in the message. The worker thread begins to assign several child threads to perform cryptographic operations in online fraud validation and the number of child threads assigned for online fraud validation is in accordance with the number of cryptographic elements present in the credit card transaction itself. Similarly to worker threads, child threads in the pool are also combined with a child queue. Each assignment of child thread is put in the child queue and the child queue will signal available waiting child threads each time the cryptographic task is added. The assigned child thread will remove the cryptographic task and proceed with its validation through HSM. These cryptographic operations encompass card security code validation, card identification number validation, personal identification number validation and chip application cryptogram validation.

Once all the child threads have been assigned for these cryptographic operations, the worker thread itself will perform an operation pertaining to card restriction validation. This operation is done in parallel with the child threads handling the cryptographic processing. The card restriction validation includes card existence validation, card status validation, card activation status validation, card expiry date validation, card usage validation and card balance validation. All the operations related to card restriction validation are done through the shared-memory pool without accessing the system database.

Once the worker thread finishes its card restriction's operation, it waits for a completion signal from the child threads that perform online fraud validation. Upon receipt of all the completion signals from the child threads, all the assigned child threads are put back to the child thread-pool for the next assignment while the worker

thread will be working on providing a final response code to the cardholder on whether to approve or decline the transaction based on the result of the entire validation. If there is any rejection during validation, the final response code will be based on the first occurrence of the rejection. Otherwise, the transaction will be approved and a unique authorization number randomly generated aspart of the authorization response message that will be used as reference. Next, the assigned worker thread will proceed with building an authorization response message in ISO8583 format. Once the response message is built, the worker thread will write the message to the socket and this authorization response will be sent back to the payment gateway that originates the transaction.

After the authorization response is sent, the assigned worker thread will drop the socket connection and proceed with internal processing. This internal processing includes saving the authorization message into an authorization table for record purposes and performing balance updating for the particular card. The balance adjustment will be updated in both the shared-memory pool and the system database. Next, the acquired mutex is released and the pending timer set earlier is cancelled before the worker thread is put back to the worker thread-pool for its next assignment.

In this project, an additional synchronization thread is started in the background of the authorization engine to update any changed information of the card done through the back office component into the shared- memory pool. This is implemented to insure the data kept in shared memory are always synchronous with similar information stored in the system database.

5.2 How Does a Singleton Design Pattern Operate?

The singleton design pattern is applied to the card object which is acting as the shared-memory pool that holds all the card information for authorization purposes. Through singleton design pattern, a class is constructed with only one instance that can be accessed globally within the multi-threaded credit card authorization system. When the listener service is activated, the listening thread will load all the information on the cards into random access memory through a configurable array. After an authorization is received, a worker thread will obtain the only instance of the card object and perform a binary search through the related array of the card objects in order to retrieve the information of the card related to the transaction from the shared memory for authorization purposes. In this project, a separate synchronization thread is initialized in the background of the authorization engine to browse the system database for any modified card information required to be updated in the shared-memory pool. This is implemented to insure the data kept in the database are synchronized with the data in the shared-memory pool. Once modified card information is loaded to the shared-memory pool, the synchronization thread will update the system database to mark that the card has been processed.

5.3 Why Multi-threaded Architecture Is Applied

Through this technique, multiple threads can be run simultaneously within the single memory space of the process and all the threads share the same system resources during the authorization process of credit card transactions. In the single-threaded credit card authorization system, both card restriction validation and online fraud validation have to be done one after another. Thus, system resources are not fully optimized because the waiting time of slow I/O operation, especially during the validation of cryptographic elements, is wasted. This not only causes the authorization to take longer to process but also degrades the performance of the server, especially during the heavy traffic in peak hours. In that case, cardholders might encounter problems getting authorization because of the slow response time from the credit card authorization system.

Multiple tasks of the authorization process could be executed concurrently through multiple threads to accelerate the authorization process. If there were two or more cryptographic operations to be performed during the authorization process, the idle time of waiting I/O operation could be reduced to at least half of the total time required in processing those operations sequentially. Apart from time, a thread-pool model is applied to minimize system resources spent in creating and destroying this type of recyclable thread.

Response time could also be further reduced if all the card information were loaded into random access memory to let the authorization system obtain information from the shared-memory pool through a binary search instead of accessing similar data from the database for authorization processing. For all these methods, the response time of the credit card authorization process could be significantly improved.

5.4 Why the Singleton Design Pattern Is Used

The singleton design pattern is applied to insure all the worker threads can access the shared-memory pool for card information during authorization. Without the singleton design pattern, shared- memory pool implemen-

tation is not possible in an object-oriented environment. Through the shared-memory pool, the access time is faster and hence improves the response time of the credit card authorization process.

6 System Evaluations

Performance-testing has been used to evaluate the response time of the authorization process under different circumstances. The response time was measured using the embedded testing tools that were built in as part of both authorization systems and payment gateway to obtain the time taken before and after a transaction was sent and received. The measurement unit for response time was recorded in seconds.

In this project, the response time was evaluated from two major aspects. These perspectives are an authorization system using a multi- threaded authentication engine against an authorization system using single-threaded authentication engine, and a multi- threaded authentication engine accessing a shared-memory pool for authentication data against a multi-threaded authentication engine accessing a system database for authentication data. In both cases, an incremental testing approach has been chosen.

For the comparison between the multi-threaded authentication engine and single-thread authentication engine, incremental testing was performed to evaluate the response time of a group of authorizations performed sequentially, as shown in Table 1. For this evaluation, no simultaneous authorization is performed. The next authorization is sent upon receiving a response from the previous transaction. The number of worker threads and child threads that were used in multi-threaded authorization system is three and nine respectively. In this testing, the result is recorded according to the best response time taken in five attempts for each category. This is done to minimize the impact of the context switching between multiple threads running in the system over the result obtained and to ensure the accuracy of the testing performed.

On the basis of the test result, it is confirmed that the performance of the multi-threaded authentication engine is better than the single-threaded authentication engine in Java platform. The performance of the multi-threaded authentication engine is almost double that of the single-threaded authentication engine in Java platform.

In the second case, the result is plotted as shown in Table 2 below. The testing was carried out to access the response time of a group of authorizations performed one after another using a multi-threaded authentication engine accessing a shared-memory pool for authentication data and a multi-threaded authentication engine accessing a system database for authentication data. Similarly to the first case, the next authorization is sent upon receiving a response from the previous transaction and no simultaneous authorization is performed.

Table 1: Test Result of Multi-Threaded and Single-Threaded Authentication Engines

No	$Multi-Threaded$	$Single-Threaded$
10	5.5	9.5
20	10.5	19.0
30	15.9	28.7
40	21.0	38.0
50	26.7	47.8
60	32.7	56.9
70	37.2	66.9
80	41.9	76.4
90	47.5	86.8
100	53.2	95.7

Table 2: Test Result of Authentication Engine Using Shared Memory and Database

No	$SharedMemory$	$Database$
10	4.9	5.3
20	10.1	10.5
30	14.9	15.6
40	20.3	20.8
50	25.5	25.9
60	30.7	31.1
70	35.6	36.5
80	40.7	41.9
90	45.9	47.5
100	50.1	52.7

On the basis of the test result, the performance of the multi-threaded authentication engine using shared memory for authentication data is better than that of the multi-threaded authentication engine using a database for authentication data in Java platform. The difference is insignificant at the earlier stage, but it is more significant when the number of authorizations is increasing. From the test result, the number of credit card authorizations that can be processed using shared memory is 10 percent more than the number of credit card authorizations that can be processed using a database at a single point of time.

7 Conclusions

This research provides a solution to optimize the performance of credit card authorization systems through

multi-threading technique in JAVA platform. This technique enables authorization of credit card transactions to be processed in a shorter time. From a business perspective, a fast and reliable authorization process will generate more revenue to the organization whereas, from the customer's point of view, authorization process on time builds the confidence of the cardholder to use the credit card as a payment method. In short, this project provides a win-win situation for both organization and community since both parties will get the benefits of implementation from multi- threaded authorization of credit card systems.

The multi-threaded authorization of credit card systems implemented in this project also enables several tasks related to card risk management profile validation to be executed concurrently during the authorization process. This will not only provide better response time for the authorization process but also enables more credit card transactions to be processed in a shorter time.

The shared-memory pool is also used in conjunction with the multi-threading technique. Since multiple threads are running in a single process space, a shared-memory pool is implemented to keep all the card information that will be used for credit card authorization process in the random access memory area. This is implemented to allow the authorization process to access the shared-memory pool for card information, which is faster than accessing similar information from a system database because it involves a less expensive I/O operation. For this reason, a synchronization thread is introduced to maintain the information in the shared-memory pool so that any update in the system database will reflect the shared-memory pool. Through shared-memory implementation, the response time of the authorization process is further improved.

The multi-threaded architectural design presented in this project also supports dynamic tuning of the size of the thread-pool running at runtime. The number of fixed worker threads and child threads can be adjusted to insure the utilization of the multiple threads to their optimal level. This is implemented to insure that the capacity of the thread-pool matches the necessities of the application based on the estimated volume and velocity of the credit card transaction processed in the specified period.

The user is enabled to monitor authorization traffic through the screen and navigate to the back office component to view the transaction details by clicking the specific record on the screen. The web-based back office component is developed in this project so that users can access the card information from other locations as long as the internet connection is provided. Both multi-threaded credit card authorization systems implemented in this project can accept multiple connections from payment systems at single port numbers. This allows more simultaneous authorizations to be received through these multiple links for load balancing usage in future.

References

[1] Agent Systems,Inc. Credit Card Authorization and Settlement for Customer-Operated POS Equipment, Agent Transaction Manager, Texas, 2007.

[2] Bank Negara Malaysia. The Payment and Settlement Systems, Annual Report 2004, Kuala Lumpur, 2004.

[3] Bourlai, T., Kittler, J. and Messer, K. Database Size Effects on Performance on A Smart Card Face Verification System. Proceedings of the 7th International Conference on Automatic Face and Gesture Recognition FGR '06, April 10-12, pp. 61-66, 2006.

[4] Chodowiec, P., and Gaj, K. Very compact FPGA Implementation of The AES Algorithm. Lecture Notes in Computer Science. Vol. 2779, pp. 319-333, 2003.

[5] Kennedy, R.A. Financial Card Authorization System, Free Patent Online, CompuServe, Inc. United States. April 08, 1997. `https://www.freepatentsonline.com/5619559.html`

[6] Computer Business Review. VÖB-ZVD Bank and Atos Origin to Build New Authorization System, CBR, London, 2005.

[7] Eslami, Y., Sheikholeslami, A., Gulak, P.G., Masui, S., and Mukaida, K. An Area-Efficient Universal Cryptography Processor for Smart Cards. Very Large Scale Integration (VLSI) Systems, IEEE Transactions, v.14(1), p43-56, 2006.

[8] Europay, Inc. Authorisation Guide, Belgium, Europay Documentation Services, 2003.

[9] Hwang, D.D and Verbauwhede, I. Design of Portable Biometric Authenticators - Energy, Performance, and Security Tradeoffs. IEEE Transactions on Consumer Electronics, v. 50(4), p1222-1231, 2004.

[10] Jewell, T. L. Distributed Authorization System, Free Patent Online, Gascard, Inc. United States. January 02, 1990. https://www.freepatentsonline.com/4891503.html

[11] Lawrence, F.C., Christofferson, R. C., Nester S. E., Moser, E. B., Tucker J. A. and Lyons A. C. Credit Card Usage of College Students:Evidence from Louisiana State University. Research Information Sheet Number 107, LSU AgCenter Communications, Los Angeles, 2003.

[12] Microsoft Corporation. Financial Services Company Increases Uptime, Cuts Delays, Attracts New Customers, Microsoft Windows Server System Customer Solution Case Study, New York, 2007.

[13] Moyer, K. R. and Richard J.D.L, MarketScope for Multiregional Card Management Software, Gartner Industry Research, 2007.

[14] Panato, A., Barcelos, M. and Reis, R. An IP of an Advanced Encryption Standard for Altera Devices. Proceedings of the 15th Symposium on Integrated Circuits and Systems Design, pp. 197-202, 2002.

[15] Russell, D. Method System for Accelerating Financial Transactions, Free Patent Online, United States. September 15, 2005. https://www.freepatentsonline.com/y2005/0203856.html

[16] Saum, J. DataDirect Shadow Transforms Their Mainframe Into Re-Usable Web Services, Seattle Times, Washington, 2007.

[17] Shen, A., Tong, R. and Deng, Y. Application of Classification Models on Credit Card Fraud Detection. Proceedings of the 2007 International Conference on Service Systems and Service Management, June 09-11, pp. 1-4, 2007.

[18] Thales, Inc. Host Security Module 8000, Thales e-Security, 2006.

[19] Thales, Inc. Personalisation Preparation, Thales e-Security, 2007.

[20] Yap, C. All ProJET Stations Accept Chip-Based Cards, Motor Trader. January 06-12, 2005.