Journal of Computer Science 4 (2): 151-160, 2008
ISSN 1549-3636

Improving Response Time of Authorization Process of Credit Card System Using Multi-Threading and Shared-Memory Pool Techniques

Siti Hafizah Ab. Hamid, Mohd Hairul Nizam M. Nasir,
Wong Yew Ming and Hazrina Hassan
Department of Software Engineering,
Faculty of Computer Science and Information Technology,
University of Malaya, 50603 Kuala Lumpur, Malaysia

Abstract: Current credit card authorization was developed on single-threaded model whereby authentication process takes longer time to respond due to sequential process of card's risk management profile and its limitation of handling huge number of simultaneous transactions at single point of time. As a result, the performance of the authorization system was affected during peak hours. This study presented an architectural framework and prototype of credit card authorization system using multi-threading and shared memory pool techniques in order to improve the response time during authorization process. Through multi-threading technique, each worker thread will be assigned with several child threads to perform online fraud validation concurrently, depending on numbers of cryptographic elements presented in transaction message. Meanwhile, the worker threads itself performed card restriction validation based on the information stored in card's shared memory pool. This approach was implemented to reduce the idle time while waiting for slow cryptographic operation in each I/O operations that is performed through external device and to accelerate the authorization process through memory operation instead of accessing similar information from database. The implementation of these techniques, were then measured in terms of response time. The results showed that the performance of multi-threaded authentication engine was almost double of single-threaded authentication engine and the number of credit card authorization that can be processed using shared memory was ten percent higher than the number of credit card authorization that can be processed using database at single point of time.

Key words: Credit card, authorization system, multi-threading, shared-memory pool

INTRODUCTION

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

Card restriction validation includes the financial[12] and non-financial verification related to the card whereas online fraud validation involves cryptographic operation through host security module (HSM) to verify the security 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. Due to various validations during authorization process for each credit card transaction, it will take some time to make a whole process to be completed. Moreover, slow and expensive input and output operation during the card restriction and online fraud validation through database and HSM also causes some delays. Besides, validation in term of the security aspect of the card itself also takes some time to process due to the complexity of the algorithms involved. As a result, the performance is affected whenever the number of authorization process is increased. In this case, it will cause some of the

Corresponding Author: Mohd Hairul Nizam M. Nasir, Faculty of Computer Science and Information Technology, University of Malaya, Lembah Pantai, 50603, Kuala Lumpur, Malaysia
Tel: +603-79676435 Fax: +603-21784965

simultaneous authorizations accepted at a single point of time are not being able to respond within the allowed timeframe. These transactions failure are classified as timed-out in the context of electronic financial services. With old payment processing methods of the conventional system, credit card transaction takes longer time during authorization processing

This study will look into the current issues on credit card authorization process. It concludes that multi-threaded authorization system with shared-memory pool is needed in order to improve the response time of the credit card authorization process and to overcome the slow sequential authorization processing problem of a single-threaded model for current credit card authorization system. The prototype of multi-threaded authorization system was developed using .NET framework, then, the performance of the multi-threading implementation is measured.

There are various methods proposed in order to improve the response time of authorization process of credit card transaction. These include invention of Host Security Module (HSM), implementation of distributed authorization system, utilization of cardholder initiated transactions device and deployment of digital network access system device. The explanation of each approach is elaborated in this section.

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 keys protection and public key cryptography[2]. HSM is implemented because of the hardware implementation is the only way that can achieve speeds beyond the reach of general-purpose microprocessors. Therefore, HSM is used as cryptographic accelerator to hasten the intense of mathematical operation especially in public key encryption and provide better performance than normal software based cryptographic system[3]. The functionalities of HSM includes verification of an on-line Personal Identification Number (PIN) by comparing with an encrypted PIN block, validation of credit card transactions by checking card security codes and performing host processing component of an Europay MasterCard Visa (EMV) based transaction. Besides, HSM also supports cryptographic operation in smart card application during personalization and performs PIN block translation that involves encryption and decryption process. The only problem with HSM apparently is there is no global standard in the low level communication data exchange protocol due to the re-engineering cost and market dominancy. 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 type for the cryptography processing.

In recent years, the introduction of HSM that supports Ethernet device is gaining its popularity because of its higher speed of the data transmission during cryptographic processing. The simulation result performed by students from one of the university in Brazil proves that IP performance version provides a better performance than other protocol solutions[4]. In short, HSM provides industry leading performance in which significantly reducing credit card transaction processing time and lowering the cost per transaction.

A patented method of distributed authorization system has been proposed in last decade to accelerate the authorization process. This system utilizes a host computer communicating with a network of remote electronic terminals from 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 order to determine authorization of a requested transaction.

A distributed authorization system and process for authorizing transactions utilizes a host computer communicating with a network of electronic terminals remote 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 is 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 establishing this connection, account data is 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[9].

However, the increasing number of terminals and credit cards, it will increase the network traffic and it is costly to maintain this information at the network level. Moreover, card issuer has less control over the authorization profile. This would result some information is not updated instantly into the network and may cause bad credit account. Besides that, there is higher potential risk of fraudulent case that would cost financial loss in the event of lost card.

Then, card holder initiated transaction device have been introduced. 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. By 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 would be used to complete the transaction. Through this technique, the authentication of the card and terminal would greatly reduce the time required to perform the transaction[14].

In this approach, three sets of messages namely purchase request message, invoice message and acknowledgement message in which each comprising a series of data packets would be transmitted to effectuate a financial transaction. This approach let the card device initiates randomized challenge 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 would be sent through acknowledgement message to complete the transaction. This approach claimed to reduce four times of the usual speed to complete an electronic transaction which is averaging 15-30 sec.

ISSUES ON CURRENT CREDIT CARD AUTHORIZATION PROCESS

The common emphases of authorization process of credit card transaction are performance and security. The performance aspect concerns on the time to authorize and complete a sales transaction whereas the security aspect concerns on the fraud prevention and confidentiality of financial information With increasing number of account and transaction volume, these two aspects remained to be major dilemma of the credit card authorization process. Current researches in recent years focus on the security area of the authorization process of credit card. This is because the number of fraudulent cases is growing dramatically and it becomes serious problem faced by credit card issuers. In 2004, credit card transactions had a total loss of 800 million dollars of fraud in United States while in United Kingdom, the loss due to the credit card fraud amounts to 425 million pounds[5].

Various fraud detection techniques have been proposed to combat the fraudulent case such as using smart cards and also implementing fraud detection system using data mining techniques including neural networks, logistic regression and decision tree. However, increasing security aspect will bring downside to performance when it is implemented using more advanced technology[6]. This is the trade off of authorization process when it is implemented using advanced technique like smart card due to the higher transmission bytes to server and longer processing time to perform verification. According to one of the local news published in Motor Traders, Managing Director of ProJET Malaysia, Matthew Selbie has mentioned that chip-based transaction will take a second or two longer than usual magnetic stripe transaction to complete the verification after deployment of the new devices to accept chip-based transaction in the petrol stations[13]. Besides that, implementation of advanced risk analysis techniques using the computer intellectual will also contribute to the processing time which may result in performance degradation.

Apart from that, the size of the database to manage the authentication data is also increasing enormously with the usage of more advanced technology such as smart card. Achievable performance levels off relatively quickly when the datasets is increasing. As a result, the verification performance decreases monotonically and appears to saturate when database size increases[7].

According to Bank Negara Malaysia's (BNM) Annual Report, the number of credit cards in circulation in Malaysia reached a total of 6.6 million as at the end of year 2004 with total transactions amounting to RM34.9 billion. Also in recent years, there has 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 but this phenomena spreads among the graduates. Understanding how consumer’s mental budgeting regarding usage

influences purchase decisions is important for marketers of financial services[8]. The credit card authorization systems that most banks are using are more than 15 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 of embedded business rules or workflow engines, resulting in, among other things, inefficient risk management operations. As a consequence, some of the transaction is not be able to get chance to be processed using conventional architecture design during high simultaneous transaction flow.

According to Tim Kelly, director of TSYS, the transaction delays in the COBOL-based programs running on mainframe has affected their business tremendously when the transaction flowing is high. To cater for this scenario, some of the banks have begun to upgrade the existing card processor application to 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 new solution, VÖB-ZVD Bank hopes to achieve a high performance authorization platform that can reliably handle all payment transactions. Managing Director of the VÖB-ZVD Bank, Gabriele Cremer-Wichelhaus has stated that new requirements and the constantly increasing number of transactions in card and internet-based payments require precocious system adaptations which would enable the bank to meet the demands of the market, the clients and to handle the future number of transactions.

ANALYSIS ON CURRENT CREDIT CARD AUTHORIZATION SYSTEM

Many banks are using home-grown authorization of credit card systems that are more than 15 years and in need of functional and technical upgrades. Other banks are using packaged applications that still need upgrades as well. In either case, 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[10]. Currently, there are a few big market players in providing authorization system solution 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[10]. However, there are still rooms for improvement as mentioned in the latest industry survey report on the payment solution to cater for payment transaction volume.

Analysis on the current authorization systems solution in the market has been conducted whereby most of the sources were obtained from the industry research report produced by Gartner recently and the information gathered as of June 2007. The findings of this analysis show that, the current credit card authorization system does 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 of current credit card authorization system is using the shared memory pool for authorization purpose. Apart from that, advanced programming languages such as .NET for example are not most commonly used in the current architecture of credit card authorization system.

In this case, performance is still remained an issue that required improvement with 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. Therefore, multi-threading should be deployed as one of the techniques to improve the response time of credit card authorization process since modern operating systems with advanced multi-core processors have a good support of multi-threading implementation.

MATERIALS AND METHODS

The prototype of credit card authorization system was developed using .NET programming language in order to measure the performance of authorization process when both techniques are applied. The functionalities of the prototype authorization credit card system is categorized into two main broad components namely front engine component and back office component.

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. 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. 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. 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, back office component stores the authentication data used in authorization of credit card system. 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.

Architectural design: As shown in Fig. 1, the architecture design of authorization of credit card system 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 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 authorization system and HSM server is specific proprietary command whereas for the other sub-systems, the message format that is used to communicate with authorization system is ISO 8583. ISO 8583 is the standard interchange message specifications defined by International Organization for Standardization (ISO) for electronic transactions made by cardholders using payment cards.

How does multi-threaded architecture work: Multi-threading technique is adopted into the architectural design of authorization engine of credit card system. 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 authorization process of credit card transaction. In this project, thread-pool model is used to handle the concurrent authorization requests from the payment gateway and shared memory pool is implemented in conjunction to multi-threading technique to hasten the authorization processing. Shared memory pool is implemented in this project to reduce the time searching card information from system database which involves expensive I/O operation as compared to obtain the similar information through shared memory pool stored in random access memory using binary search. 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. Besides that, all the card information is also loaded to shared memory pool before the authorization request could be serviced.

The worker threads in the pool are combined with a work queue. Each accepted client socket through listening thread from payment gateway will be put in the work queue. The work queue will signal 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 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, timer has been set to release the mutex after pre-defined period elapses.

The worker thread assigned for 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. Similar 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

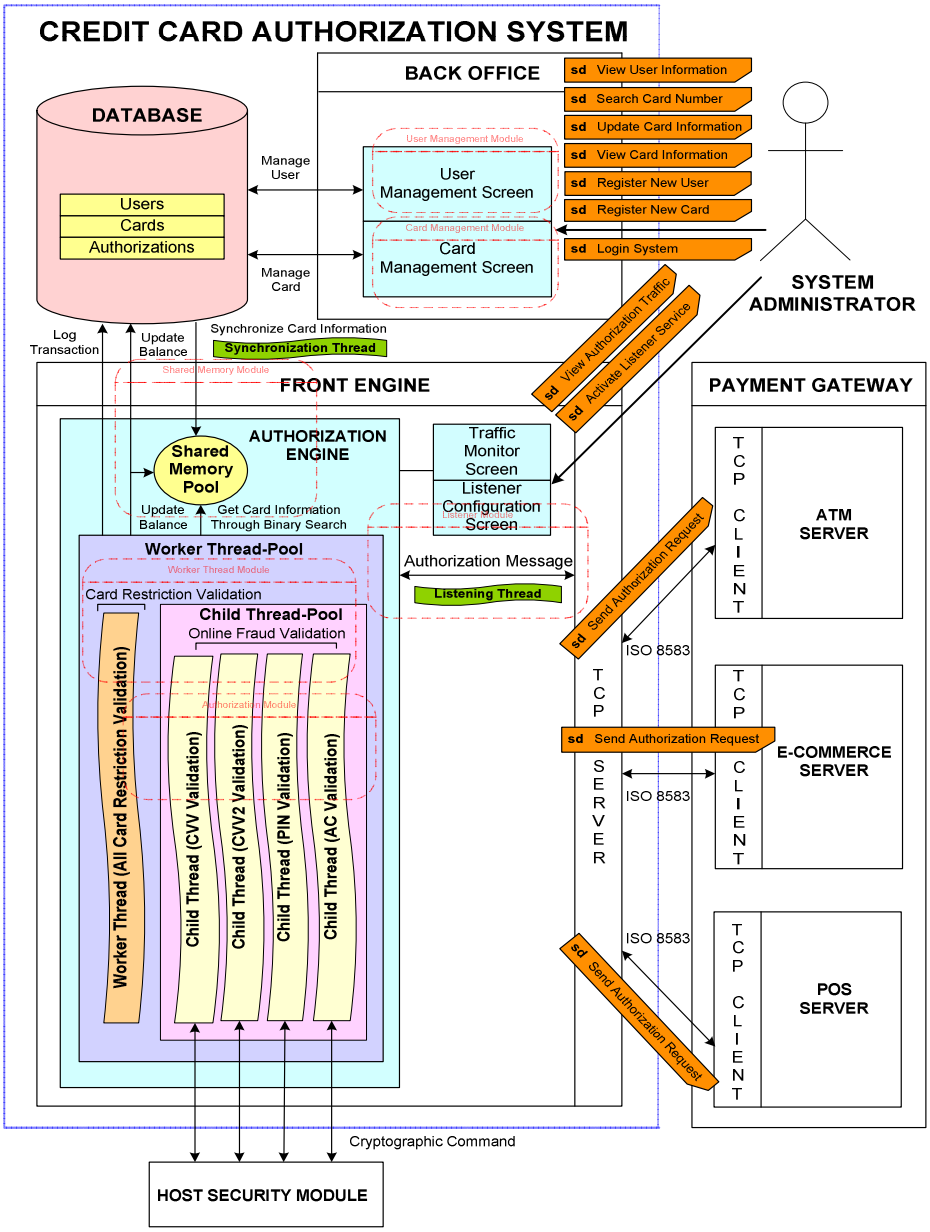

Fig. 1: Overall System Architecture

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 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 expiration date validation, card usage validation and card balance validation. All the operations related to card restriction validation are done through shared memory pool without accessing system database.

Once the worker thread finishes its card restriction's operation, it waits for completion signal from the child threads that perform online fraud validation. Upon receiving all the completion signals from the child threads, all the assigned child threads are put back to child thread-pool for next assignment while the worker thread will be working on providing final response code to the cardholder on whether to approve or decline the transaction based on the result of entire validation. If there were any decline during validation, the final response code will be based on the first occurrence of the decline. Else, the transaction will be approved and a unique authorization number is randomly generated as part of the authorization response message that will be used as reference. Next, the assigned worker thread will proceed with building 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 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 authorizations table for record purpose and performing balance updating for the particular card. The balance adjustment will be updated in both shared memory pool and system database. Next, the acquired mutex is released and the pending timer set earlier is cancelled before worker thread is put back to the worker thread-pool for its next assignment.

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

In 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 optimised because the waiting time of slow I/O operation especially during the validation of cryptographic element is wasted. This not only causes the authorization of credit card transaction will take longer time to process but also it degrades the performance of the server especially during the heavy traffic during peak hours. In that case, cardholder might encounter problem of getting authorization from the system because of the slow response time from the credit card authorization system.

In this project, multi-threaded credit card authorization system is used to accelerate the authorization process. Multiple tasks of authorization process could be executed concurrently through multiple threads. 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 total time required in processing those operations sequentially. Apart from time, thread-pool model is applied to minimise system resources spent in creating and destroying this type of recyclable threads.

Besides that, response time could be further reduced by loading all the cards information into random access memory to let authorization system to obtain information from the shared memory pool through binary search instead of accessing similar data from database for authorization processing. Through all these methods, the response time of the credit card authorization process could be significantly improved.

How does singleton design pattern operate: 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. The singleton design pattern is applied into the card object which is acting as shared memory pool that holds all the cards' information for authorization purpose. When the listener service is activated, listening thread will load all the information of 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 cards object and perform binary search through the related array of the cards object in order to retrieve the information of the card related to the transaction from the shared memory for authorization purpose. In this project, a separate synchronization thread is initialised in the background of the authorization engine to browse the system database for any modified card information required to be updated into the shared memory pool. This is implemented to ensure the data kept in the database is synchronised with the data in the shared memory pool. Once modified card information is loaded to shared memory pool, the synchronization thread will update the system database to mark the card has been processed.

Singleton design pattern is applied to ensure all the workers threads can access to shared memory pool for card information during authorization. Without singleton design pattern, shared memory pool implementation is not possible in object-oriented environment. Through shared memory pool, the access time is faster and hence, improving the response time of a credit card authorization process.

RESULTS AND DISCUSSION

Timing 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 transaction was sent and received. The measurement unit for response time was recorded in seconds.

In this research project, the response time was evaluated in two major aspects. These aspects are authorization system using multi-threaded authentication engine against the authorization system using single-threaded authentication engine and multi-threaded authentication engine accessing shared memory pool for authentication data against multi-threaded authentication engine accessing system database for authentication data. For both aspects, incremental testing approach has been chosen as the technique to obtain the result.

On the comparison between 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. For this evaluation, there is no simultaneous authorization performed. The next authorization will be sent upon receiving response from 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 based on the best response time taken in five attempts for each category. This is done to minimise 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. The results were then plotted in the table and figure as shown in Table 1 and Fig. 2.

Based on the test result as shown in Table 1 and Fig. 2, it has been confirmed that the performance of multi-threaded authentication engine is better than single-threaded authentication engine in NET platform. From the result, the performance of multi-threaded authentication engine is almost double of single-threaded authentication engine in both platforms.

In the second aspect, the testing was carried out to access the response time of a group of authorizations performed one after another using multi-threaded authentication engine accessing shared memory pool for authentication data and multi-threaded authentication engine accessing system database for authentication data. Similar to first aspect, the next authorization will be sent upon receiving response from previous transaction and there is no simultaneous authorization performed. The number of worker threads and child threads that were used in multi-threaded authorization of credit card system is also similar, which is three and nine respectively. The test result is recorded based on the best response time taken in five attempts for each category. This is done to minimise 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. The results were then plotted in the table and figure as shown in Table 2 and Fig. 3 below.

Based on the test result as shown in Table 2 and Fig. 3, the performance of multi-threaded authentication engine using shared memory for authentication data is better than multi-threaded authentication engine using database for authentication data in .NET platform. The difference is insignificant at the earlier stage, but it is getting more significant when the number of authorizations is increasing. From the test result, the number of credit card authorization that can be processed using shared memory is ten percent more than the number of credit card authorization that can be processed using database at single point of time.

Table 1: Test result of multi-threaded and single-threaded authentication engine

Number of sequential authorizations	Multi-threaded authentication engine (sec)	Single-threaded authentication engine (sec)
10	4.7	8.1
20	9.4	16.5
30	14.2	24.8
40	18.8	33.0
50	23.4	41.1
60	28.2	49.4
70	32.7	57.7
80	37.6	65.9
90	42.1	74.7
100	46.9	82.3

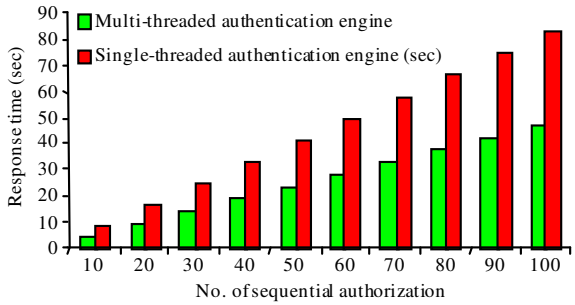

Fig. 2: Test result of multi-threaded and single-threaded authentication engine

Table 2: Test result of authentication engine using shared memory and database

Number of sequential authorizations	Authentication data via shared memory (sec)	Authentication data via database (sec)
10	4.1	4.4
20	8.1	8.9
30	12.1	13.4
40	16.4	17.7
50	20.3	22.0
60	24.5	26.7
70	28.7	31.4
80	32.8	36.2
90	36.9	41.0
100	40.5	45.9

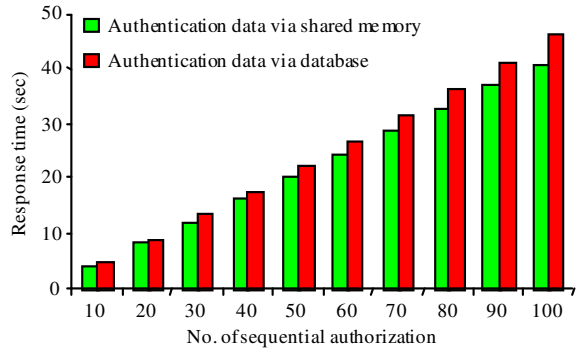

Fig. 3: Test result of multi-threaded and single-threaded authentication engine

CONCLUSION

This research provides a solution to optimize the performance of credit card authorization system through multi-threading technique in .NET platform. Through this technique, it enables authorization of credit card transaction to be processed in shorter time. From business point of view, fast and reliable authorization process will generate more revenue to the organization whereas from customer point of view, authorization process on time builds confidence of the cardholder to use the credit card as payment method. In short, this project provides win-win situation to both organization and community as both parties will get the benefits of implementation from multi-threaded authorization of credit card system.

Besides that, multi-threaded authorization of credit card system implemented in this project enables several tasks related to card's risk management profile validation being executed concurrently during authorization process. This will not only provide better response time to the authorization process but also it enables more credit card transaction can be processed in multi-threaded authorization system at shorter time.

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

Besides that, the multi-threaded architectural design presented in this project 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 ensure the utilization of the multiple threads to its optimal level. This is implemented to ensure 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 specified period.

Apart from that, 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. Web-based back office component is developed in this project so that user can access the card information from other location as long as the internet connection is provided. Another unique feature implemented in this project is any changes made to the card are recorded for audit purpose. Both multi-threaded credit card authorization systems implemented in this project can accept multiple connections from payment system at single port number. This is implemented to allow more simultaneous authorizations to be received through these multiple links for load balancing usage in future.

ACKNOWLEDGEMENT

First and foremost we would like to express our gratitude to almighty that gave us the possibility to complete the research work successfully. Secondly, we would like to forward our deepest thank to my colleagues, lecturers and technical staffs from the Department of Software Engineering for their endless assistance, technical advice and co-operation.

REFERENCES

1. Leung, W.K., and Lai K.K, 2001. Improving The Quality of The Credit Card Authorization Process-A Quantitative Approach. Source: International Journal of Service Industry Management. 12(4), 328-341.
doi:10.1108/EUM0000000005679
http://www.ingentaconnect.com/content/mcb/085/2001/00000012/00000004/art00002
2. Chodowiec, P., and Gaj, K. 2003. Very compact FPGA Implementation of The AES Algorithm. In: Lecture Notes in Computer Science. (eds Springer Berlin / Heidelberg), vol. 2779, pp: 319-333.
doi:10.1007/b13240
http://cpe02.gmu.edu/rcm/publications/CHES_2003_AES.pdf
3. Eslami, Y., Sheikholeslami, A., Gulak, P.G., Masui, S., and Mukaida, K., 2006. An Area-Efficient Universal Cryptography Processor For Smart Cards. Source: Very Large Scale Integration (VLSI) systems. IEEE Transaction On, 14 (1), 43-56.
doi: 10.1109/TVLSI.2005.863188
http://ieeexplore.ieee.org/Xplore/login.jsp?url=/iel5/92/33693/01603567.pdf?temp=x
4. Panato, A., Barcelos, M., and Reis, R., 2002. An IP of an Advanced Encryption Standard For Altera Devices. Source: The 15th Symposium on Integrated Circuits and Systems Design, pp: 197-202
doi: 10.1109/SBCCI.2002.1137658
http://ieeexplore.ieee.org/Xplore/login.jsp?url=/iel5/8229/25383/01137658.pdf?arnumber=1137658
5. Shen, A., Tong, R., and Deng, Y., 2007. Application of Classification Models on Credit Card Fraud Detection. Source: Service System and Service Management, 2007 International Conference, pp: 1-4.
doi: 10.1109/ICSSSM.2007.4280163
http://ieeexplore.ieee.org/Xplore/login.jsp?url=/iel5/4280076/4280077/04280163.pdf?tp=&isnumber=&arnumber=4280163
6. Hwang, D.D., and Verbauwhede, I., 2004. Design of Portable Biometric Authenticators Energy, Performance and Security Tradeoffs. Source: IEEE Transaction on Consumer Electronics, 50(4): 1222-1231.
doi: 10.1109/TCE.2004.1362523
http://ieeexplore.ieee.org/Xplore/login.jsp?url=/iel5/30/29849/01362523.pdf?arnumber=1362523
7. Bourlai, T., Kittler, J., and Messer, K., 2006. Database Size Effects on Performance on A Smart Card Face Verification System. Source: Proceedings of the 7th International Conference on Automatic Face and Gesture Recognition FGR '06, pp: 61-66.
doi: 10.1109/FGR.2006.36
http://portal.acm.org/citation.cfm?id=1126250.1126272&coll=GUIDE&dl=GUIDE
8. Yang, S., Markoczy, L., and Qi, M., 2007. Unrealistic Optimism in Consumer Credit Card Adoption. Source: Journal of Economic Phychology. 28(2), 170-185.
doi: 10.1016/j.joep.2006.05.006
http://cat.inist.fr/?aModele=afficheN&cpsidt=18631617
9. Jewell, T.L., 1990. Distributed Authorization System. Source: Free Patent Online.
http://www.freepatentsonline.com/4891503.ht ml
10. Moyer, K.R. and De Lotto, R.J., 2007. MarketScope for Multiregional Card Management Software. Source: Gartner Industry Research
http://www.gartner.com/DisplayDocument?id=508317&ref=g_sitelink
11. Russell, D., 2005. Method and System for Accelerating Financial Transactions. Source: Free Patent Online.
http://www.freepatentsonline.com/y2005/0203856.html
12. Visa, 2001. Inc. Visa Integrated Circuit Card Specification. Visa Public, Singapore.
www.scardsoft.com/documents/VISA/ICC_Card.pdf
13. Yap, C., 2005. All ProJET Stations Accept Chip-Based Cards. Motor Trader.
http://www.motortrader.com.my/NUS/articles/0/article_157/page_m.asp