Business Case for the Cloud

for

Master of Science

Information and Communication Technology

David Beltran

University of Denver University College

June 4, 2023

Faculty: Michael Schwartz, M.S.

Director: Cathie Wilson, M.S.

Dean: Michael J. McGuire, MLS

Table of Contents

Abstract

This document includes a description of the high-level design of the credit card processing system.

Credit Card Processing Distributed System

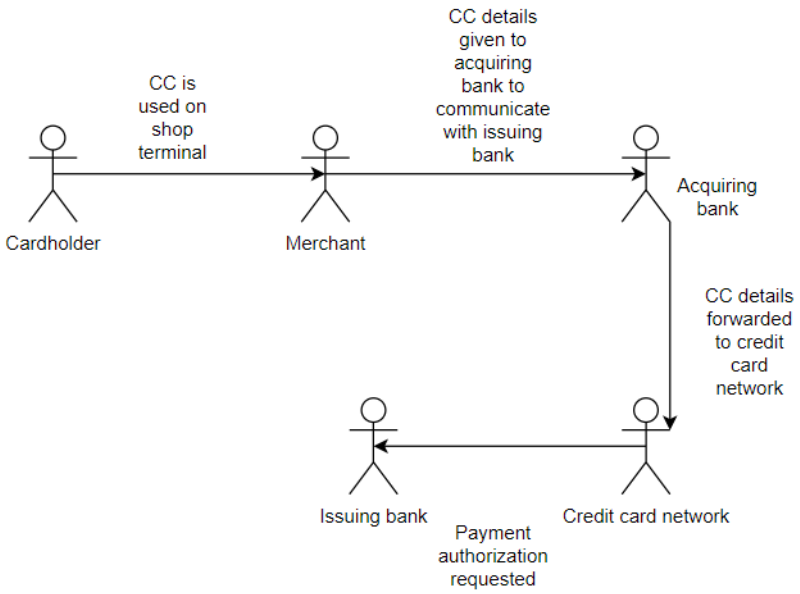
Credit cards have made making payments at a store simpler for the customer. The process for that payment to be handled so that money from the customer's bank account ends up in the merchant's bank account is more complex than swiping a card though. There are several actors that take part in processing credit card payments. To make the experience for the customer and the merchant quick and reliable, a credit card processing application needs to be deployed on a distributed system.

A credit card processing system lends itself to being designed as a distributed system. A large part this is due to is how many components are required to communicate with each other so that a transaction is completed. When a credit card is used as the form of payment, the transaction runs through three stages: authorization, authentication, and clearing/settlement (Kiernan 2022). Each of these stages contains their own communicating components, meaning there are several nodes each completing its own process. To perform a successful transaction these nodes in this system depend on each other even though they are distributed in a physically separated Wide-area network (WAN).

Stages of Credit Card Processing

The first stage of authorization demonstrates use cases where all participating nodes are communicating with each other. When a shopper provides their credit card to the merchant through the merchant's terminal, the credit card details are sent to the acquiring bank. To request authorization from the issuing bank, the acquiring bank first sends the credit card details to the credit card network which serves as the channel between the acquiring and issuing banks. The below use case diagram shows how communication works for authorization:

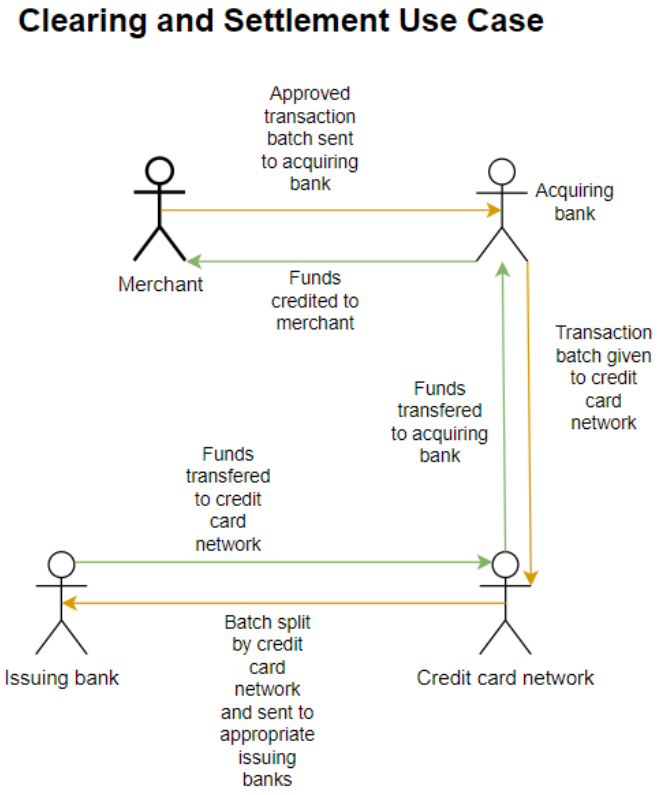
Authorization Use Case



The next stage is authentication where the issuing bank is required to verify the credit card. A similar diagram as the authorization use case diagram would exist for authentication, except communication goes in the reverse order. Communication begins where the issuing bank verifies the credit card information and sends an approval code back to the merchant. This code needs to travel through the proper network of nodes though by first being sent to the credit card network then to the acquiring bank and finally to the merchant. The merchant then provides a sales receipt to the customer indicating the payment was approved.

Although the payment was approved through verification, the transaction process is not complete and still needs to perform the final stage of clearing and settlement. This is the stage where the cardholder's bank account is deducted from the amount for what the payment was approved in the previous two stages. To complete this task the merchant sends a batch of authorized and approved transactions to the acquiring bank. The acquiring bank then forwards

this batch to the credit card network. The credit card network then splits the batch and sends each transaction to their respective issuing banks. The issuing bank then transfers the appropriate funds back to the credit card network so that it can obtain its processing fee. The funds are then sent to the acquiring bank which then credits the merchant with the funds minus processing fees. The use case diagram below demonstrates this communication between nodes of this distributed system:



Distributed System Design Requirements

When looking back at the communication that is occurring during the authorization use case, several components are working together to approve a credit card transaction. Each of the actors in the processing system has their own responsibilities to ensure the correct information is being collected and delivered to the next correct actor through the

correct channel.

Each of these actors, which include the merchant, acquiring bank, credit card network, and issuing bank, operate their piece of the process on their independent machines. This becomes the highest level of a distributed system because all the nodes are working separately but towards the same task. To fully define the processing system as distributed, we must look at what each actor is responsible for and how they accomplish their responsibilities.

The first step of authorization takes place between the cardholder and merchant actors. The cardholder swipes their credit card to deliver the credit card information to the merchant. This is done through a point-of-sale (POS) terminal that needs to be provided by the merchant. This terminal/node holds the software that is designed to read the credit card strip correctly so that the identifying information is collected.

After the information is collected by the merchant's POS terminal, that information is sent to the acquiring bank. There are two important pieces needed for this information to be delivered from the merchant to the acquiring bank. The first piece is the internet. The internet serves as the WAN where communication between nodes and components can take place. The next important piece is the software that performs the communication between actors and nodes. This does not necessarily mean that every node needs to have the exact same software as those nodes within the acquiring and issuing banks because they may not be the same bank. Whichever software they chose though will need to have the ability to communicate with other software utilized for processing credit card transactions.

A similarity these actor's node software will share is how it is executed. For the

processing system to perform as expected without losing communication the software would need to be run through a cloud service. A cloud service provides the hardware needed for scalability. With multiple servers dedicated to processing transactions, it is a high certainty the transaction will take place even if a server goes down.

The service model that will provide enough cloud service is Platform as a Service (PaaS). There may be a Software as a Service (SaaS) model that provides the specific requirements for credit card processing. If so, then SaaS will be enough for the acquiring and issuing banks. The credit card network might want to procure a PaaS model though so that their developers have the flexibility to develop specific communication designs that fit their needs. The credit card network is the actor that performs the most important communication between the bank that pays and the bank that gets paid. Their software would require middleware that performs the task of transporting information using proper port connections.

Distributed System Architecture

The remainder of the steps taken during the authorization use case requires the actors and their nodes be designed within a distributed system architecture. An architecture that is suited for the task of processing credit card transactions would be an object-based architecture working on a structured peer-to-peer system (van Steen and Tanenbaum 2017). The actors and their nodes each interact with one another in a specific order working towards one task like peers. With credit card information consisting of a credit card number, expiration date, billing address, CVV number, and payment amount, it makes sense to encapsulate this information within an object. Once this object is created by the merchant's POS terminal, the credit card object can be delivered to the next actor's node for processing. This object is delivered all the

way to its destination, and what is returned from the issuing bank to the merchant is a code which can be delivered as a simple object as well. Even when delivering an approved transaction batch like in the clearing and settlement use case, a collection of approved transaction objects would be communicated within the object-based architecture.

Through the PaaS object-based structured peer-to-peer architecture system, the acquiring bank and credit card network are responsible for their client side machines to deliver the credit card information object from one to the other. As described in the authorization use case, the credit card network requests authorization from the issuing bank. The issuing bank and its nodes will be responsible for validating the credit card information object and delivering the validation code back through the processing system. This system of distributed nodes performing their individual tasks and communicating with one another encapsulates the credit card processing system. Although the tasks are divided amongst several actors and their nodes, they are all working together to perform their credit card transaction requirements.

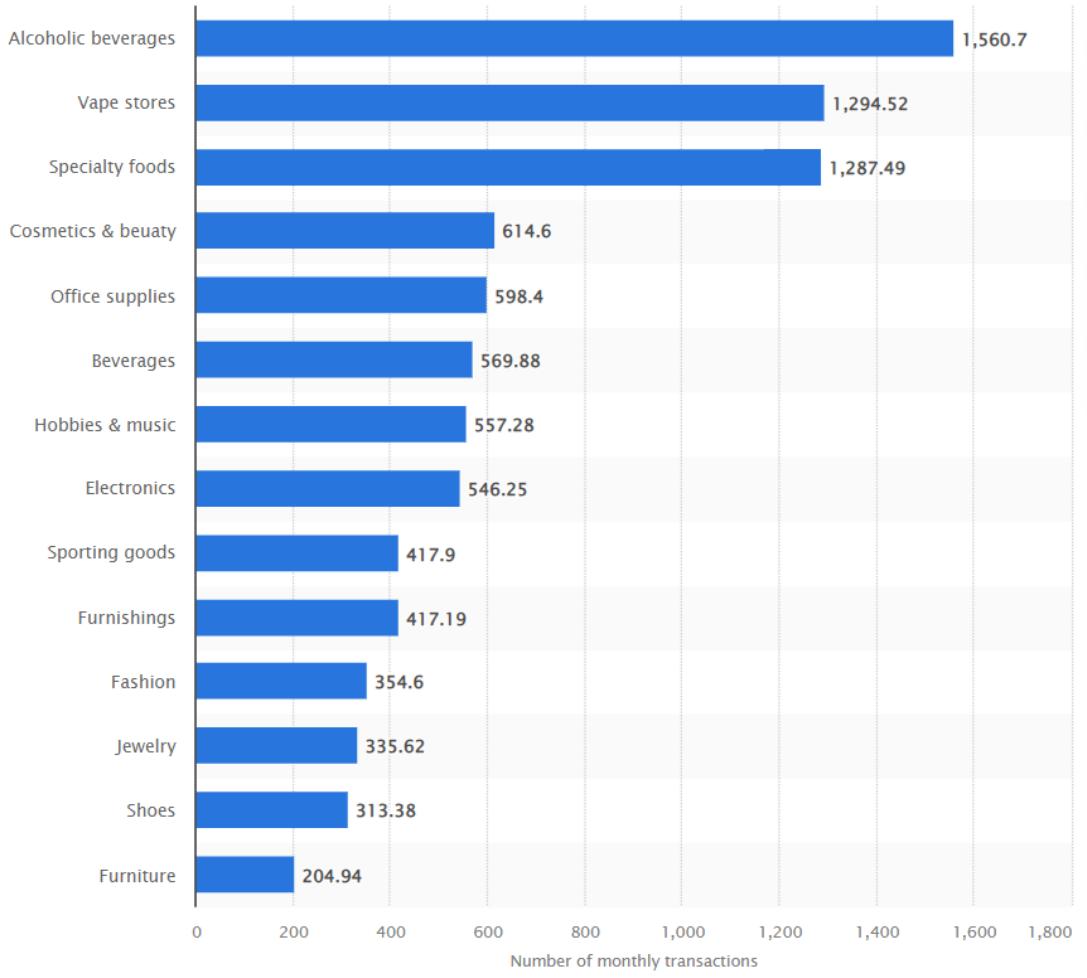
Financial Cost

Calculating the financial cost of implementing this system depends on each actor and how much hardware and software they invest in. The actor that will need to invest the most is the merchant because the most crucial tasks are performed by the merchant portion of the entire distributed system. The merchant portion is where the transactions are created and originate when communicating messages between all the other actors.

Establishing Usage

Considering the merchant portion of the credit card processing system, the costs

associated with running the distributed system for this actor require establishing its expected usage. When applying the software and technology involved with running the merchant's piece of the credit card processing system, the application is designed to fit the needs of stores selling any merchandise or service. This is considered when establishing what the expected transaction workload the application is required to handle. The average number of payment transactions that occur in stores worldwide is 647 per month per store in 2018 (Sabanoglu 2022). This average comes from several different store types included in the chart below:



This chart demonstrates that stores selling alcoholic beverages have the highest average

of 1560 transactions per month. This number and including the average of most store types being 300-500 transactions per month, suggest that the merchant's portion of the application should be designed to handle at least 1000 transactions per month, which is around .06 transactions per second per store.

AWS Cloud Service

The other category needing to be established is selecting a cloud provider. Looking at the operations of the top credit card processing companies helps make this decision. The company selected as the best credit card processor of 2023 is Square (Fabregas and Bottorff 2023). This top credit card processing company selected AWS as their cloud provider, making AWS the cloud provider to help calculate the costs of this credit card processing application (Narayanan 2019).

Moving the application to AWS's cloud service aids the merchant in managing and maintaining their processing system to a high degree. This is especially the case when selecting Amazon's EC2 (Elastic Compute Cloud) AWS package where software and hardware are managed by the cloud provider. This package enables the customer to select which level they believe their application needs to start when referring to number of processors, network bandwidth, and storage. An extra appeal of EC2 is AWS provides auto scaling. Not only does the merchant not need to purchase and manage their own servers, but AWS also scales the use of resources automatically depending on application and user access usage, free of charge (Amazon 2018). EC2's free auto scaling provides a service that takes away the requirements of monitoring how much the application must be scaled when traffic flow in and out of the application is lower or higher than expected. In the case that the applications scales at a high enough rate where usage exceeds the original selected resources (CPU, bandwidth, and

storage), the merchant will be moved up to the next level of available resources with a higher cost.

The next step is deciding which level of the EC2 package to select. This will depend on the expectations of usage therefore looking at a single merchant store with an expected 1000 transactions per month, it is acceptable to consider the lower levels. Each credit card transaction is typically the size of around 1 KB (Nakedible 2011). With a merchant expecting 1000 transactions per month, the processing system at this single store communicates around 1000 KB of credit card data in a month. For one merchant store, 1000 KB will not demand very expensive amounts of cloud service.

Required CPU

With .06 transactions per second taking place, the number of CPU/processors needed is low. On average, a machine with one CPU will have two to four cores (Horowitz 2020). This allows for two to four instructions to be handled consecutively, meaning a merchant with an average machine server could run two to four transactions at the same time. Amazon's AWS servers have been recently upgraded with Graviton 3 processors. Each of these processors are packed with 64 cores, making one AWS CPU enough for a merchant with 1000 transactions per month (Mayersen 2022). Obtaining the required CPU from a cloud service eliminates the need for a dedicated CPU at the merchant location. The merchant would have access to the CPU virtually through the processing application. If considering the cost of CPU to AWS, Graviton 3 is a processor developed by Amazon. Their processors are currently not available to the open market other than virtual access. A processor that compares to the Graviton 3 and is available to the public for purchase is the Intel Xeon Scalable 'Ice Lake-SP' Processor which costs $9875

(Shilov 2021). The power and financial cost of a Graviton 3 processor or equivalent exceeds the needs of a single merchant's need for a credit card processing system, making cloud service even more reasonable.

The merchant would still be required to purchase the hardware needed in store to run the application's UI though. When comparing to the top ranked credit card processing system, Square, each register collecting credit card information is made up of several components. These include the monitor, cash drawer, receipt printer, kitchen printer (for restaurants or stores needing orders printed away from register), and a portable terminal. For most stores with merchants running 1000 transactions per month, one or two full kits of components will be needed. A Square kit, which is considered the top in the market, will cost the merchant $1439 plus tax (Square n.d.).

Required Network Bandwidth

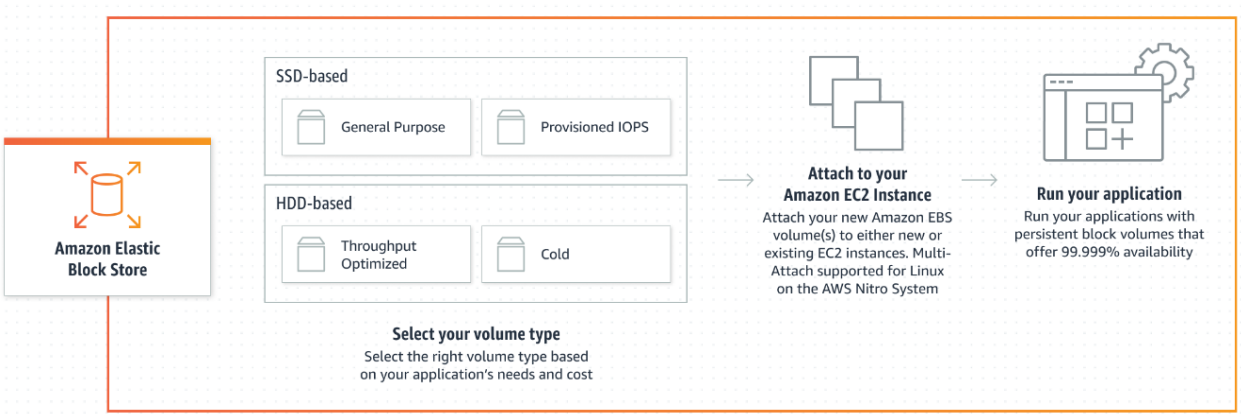
Obtaining enough network bandwidth for the credit card processing system at a merchant level should also be a lower cost. With transactions themselves being a small data size, one transaction uses 1.5 MBPS to download and 768 KBPS to upload (5GStore 2023). With 1000 transactions per month the merchant can expect to run around 34 transactions per 12-hour workday. The required bandwidth to handle .06 transactions per second is 250.5 KBPS for download and 128 KBPS for upload. These download and upload speeds are 16.7% of a single transaction. With lower transaction rates of .06 transactions per second, it could be argued that if network traffic is dedicated only to credit card processing, 2GBPS of network bandwidth is more than plenty. Choosing the internet provider will depend on the merchant's location, but prices should be similar in any location. For merchants operating in the Pacific time zone, COX is

a popular internet provider with a business plan that includes 2 GBPS service at $69/month (COX n.d.).

Required Storage

Storing data is necessary for the processing system because the merchant needs to hold onto all the transactions made in a day, and then send back as a batch record full of approved transactions to the acquiring bank. It is also helpful for the merchant to store the full history of successful transactions whether it's for marketing, budgeting, or analytical purposes. With each transaction costing as little as 1KB of data, the full history of transactions for a merchant running 1000 transactions per month can be stored in an 8 GB flash drive. Since batch records are created and stored at the POS terminal, the flash drive can be connected to the terminal. Considering that credit card transactions are sensitive data, it might be advisable to acquire a secure flash drive. Wired considers the Lexar Jumpdrive Fingerprint F35 the most secure flash drive (Hill 2023). As mentioned in its name, this flash drive is fitted with a fingerprint reader so that access to its data is controlled by the merchant. The smallest available size is 32GB which can be purchased on Amazon.com for $28.48 (Amazon n.d.).

An alternative to storing long-term data in-house would be to utilize AWS's storage database. Examining Amazon's offered packages, most of their packages offer Elastic Block Store (EBS-only storage). This cloud service emulates storage like a flash drive and can be provided as either SSD-based or HDD-based (Amazon n.d.):



If selecting to store batch record history with AWS, the HDD-based volume type fits the needs of the merchant because of the longer lifespan when compared to SDD (Villinger 2022). The cost of storing data with AWS's EBS is $.096 per GB per month (Amazon n.d.).

Scaling Single Store

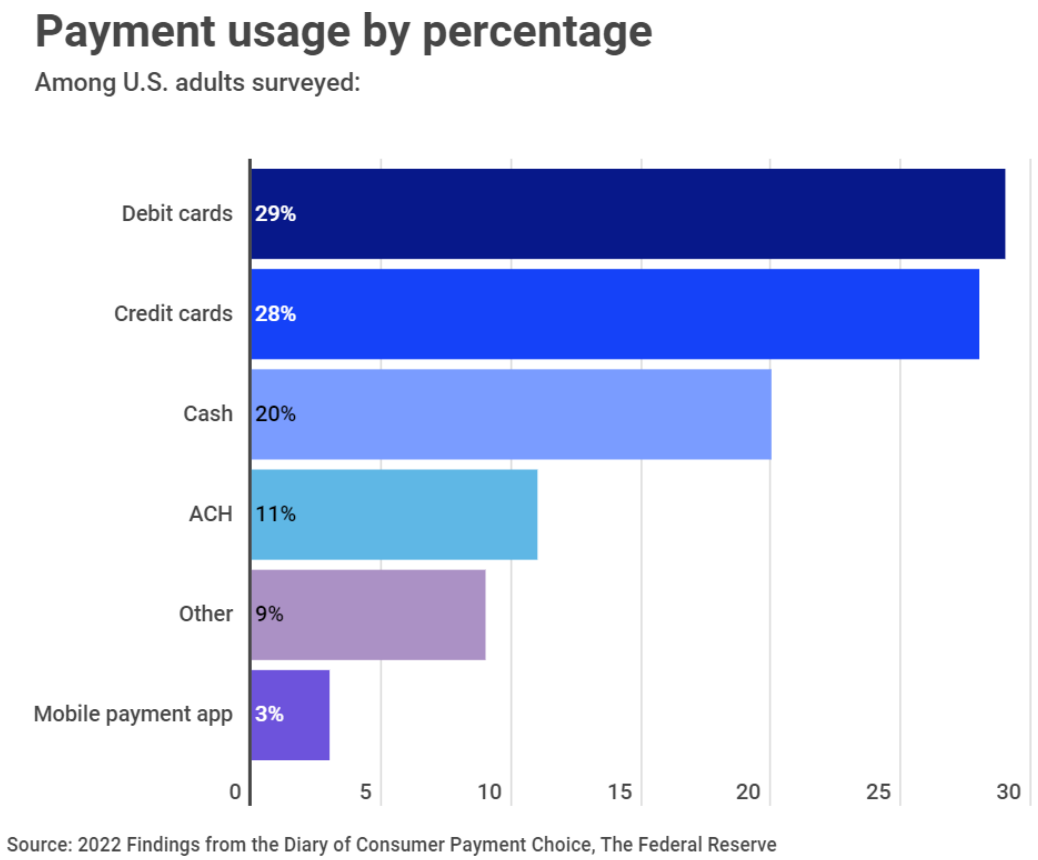
Knowing the CPU, network bandwidth, and storage needs of the credit card processing system in a merchant's store running 1000 transactions per month, the r7g.medium EC2 instance should suffice to begin with AWS. This instance is a memory optimized option that benefits the merchant in handling large workloads quickly. Memory optimization fits the credit card processing system because transactions are expected to work quickly to allow customers a satisfactory experience in making purchases. The r7g.medium instance offers one vCPU, 8 GB of memory, EBS (if selected), and up to 12.5 GBPS of network bandwidth (Amazon n.d.). The price for this service is $39.128 per month (Vantage.sh n.d.). The bottom instance tier under the memory optimized EC2 package provides more than enough power to run the merchant's portion of the credit card processing system when considering they are running the average number of transactions. The number of users communicating to the web server in a merchant's store with 34 transactions per day can be easily handled by an AWS CPU since one CPU core can

handle 250 concurrent requests at one time (Smith 2022). With Amazon's Graviton 3 processor packed with 64 cores, and .06 transactions occurring per second, AWS's cloud web server will be able to handle credit card processing requests with r7g.medium instance offering one vCPU. Regarding network traffic, the bandwidth availability of up to 12.5 GBPS is overkill for the systems required 250.5 KBPS bandwidth to handle each transaction.

Scaling the application requires nothing from the merchant since AWS handles this for free. The expected growth and decline of transactions in a store fitting under the averages of small businesses worldwide should not exceed the need to move up more than one EC2 instance. The next level of r7g.large carries two vCPUs, 16 GB of memory, and remains at 12.5 GBPS of network bandwidth costing $78.183 per month. Including another processor would require a large spike in business for a single merchant. Even doubling the number of transactions to 2000 per month would come to .09 transactions per second. Doubling business would mean the merchant is able to continue purchasing the r7g.medium instance. A merchant will only need to concern themselves with scaling if their store franchises.

Scaling Franchise

Considering franchising numbers scales the merchant significantly in not just cloud service. If expecting the merchant becoming as successful as the top franchise in the world, McDonalds, the number of transactions significantly increases (Gupta 2023). This would mean the merchant has increased their transactions to 69 million per day which comes to 95,833 transactions per second in a twelve-hour workday (McCain 2023). We can cut down the number of transactions to 54,624 if we account that 57% of consumers use a credit/debit card as payment (Murray 2023):



With 54,624 transactions being processed per second, the processing system would need 218 processors to handle this much traffic. The merchant would then need to purchase four separate instances of the memory optimized EC2 package but would need to move up to the r7g.metal instance costing $2501.86 per month for each instance coming to a grand total of $10,007.44 per month. AWS's free auto-scaling would make a significant difference in budgeting to handle user traffic changes at this scale.

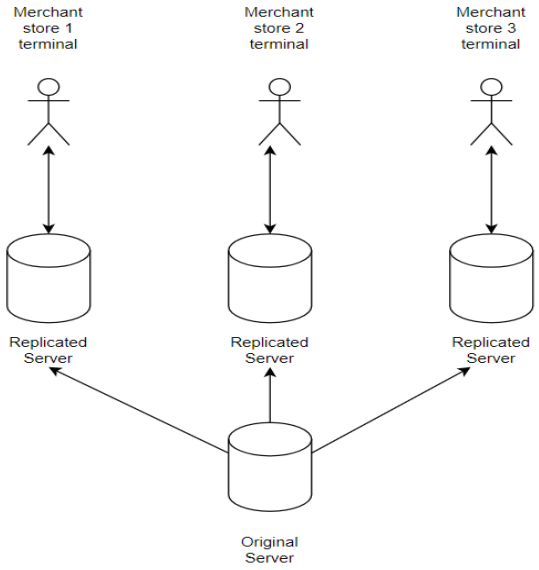
Scaling with AWS

It is very handy for the merchant to leave the scaling to AWS. Amazon's cloud service though requires monitoring how much the merchant's portion usage of the processing system is growing or shrinking. Applying only 1000 to 2000 transactions per month may not require

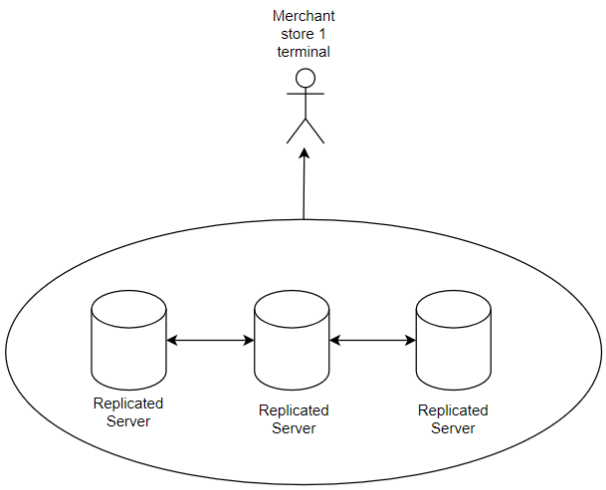
very much replication of data among several servers. Without adding more stores, the merchant's store would require growing significantly on its own. With more transactions though means AWS would need to scale for size since there is more data and processes needing to be handled by more users (Joshi 2019). If the merchant's portion expands to more stores or even franchises, replication of its data will be mandatory particularly due to the application requiring geographical scalability. With more stores scattered across different locations, AWS will need to replicate the merchant's server data to servers near each of the merchant's stores to maintain low latency performance.

Handling credit card processing will steer AWS to model the merchant's distributed system in a data-centric consistency as well. For the application to be reliable, communication throughout the distributed system will need its data replication to remain consistent by adhering to data stores waiting to read and/or write until the last write operation took place on that data. Each POS terminal within the merchant's portion will sacrifice slower transactions so that the order in which each transaction is made is kept in a first-in-first-out (FIFO) order. Modeling this system as client-centric would be catastrophic for this application since accessing the same customer's bank account at the same time is possible. A client-centric model would allow a transaction to be authorized if a customer does not have the funds because the client machine may not be aware of changes made by another client machine (Steen and Tanenbaum 2017).

The diagram below demonstrates how each merchant store in a geographically scaled system communicates with their own local replicated server:



To maintain a data-centric model, once the terminal at store 1 reads a credit card and sends the data to its local replicated AWS server/s, that terminal must wait for a response until all servers in the distributed server system authorize the last transaction approval write. Once the last write has been confirmed then the authorized code can be sent to the store 1 terminal. Each replicated server within the geographically distributed system communicates with one another ensuring the final authorization is not sent until the last transaction write has been verified at each server:



Whether as one store or a franchise, AWS handling scaling for the merchant provides a cost-effective solution. Any changes to the cloud service the merchant may require because of an increase in transactions are made simply by selecting the correct EC2 instance. When the correct EC2 instance is selected, Amazon's Elastic Load Balancer (ELB) monitors user traffic. The ELB analyzes how much use a server can handle, and when user traffic increases, the load balancer moves traffic to another server not being used. AWS's load balancer is effective and provided for free.

Privacy and Security

The mock application that runs the merchant's portion of the distributed credit card processing system communicates between each server using HTTP. To further secure this communication ensuring the customer's and merchant's information is kept private, cryptography can be implemented into the HTTP communication design. Cryptography ensures two things: authentication and privacy. When a merchant sends either a single transaction or a batch of transactions to the acquiring bank, the message can be encrypted. To decrypt the message the acquiring bank would share a public key with the merchant in what is called a symmetric cryptosystem. If further security is desired, the cryptosystem can be combined with an asymmetric system where the merchant and the acquiring bank each have their own secret key along with a shared public key.

This strategy works with replicated servers as well whether they are from the merchant or acquiring bank portion. In a symmetric and asymmetric cryptosystem, server communication is kept secure through cryptography with a shared public key and individual secret keys. A merchant will know they are communicating with their acquiring bank because

the message they sent is encrypted, and if the receiver is the acquiring bank, only they will be able to decrypt that message with the public and secret keys keeping that information private.

Reliability

Whether the merchant decides to store their authorized transactions in-house or through a cloud service, replication of that data maintains the system will be more reliable. This will be the case as will regarding the software running each node of the distributed system. With multiple servers running replicated software or storing replicated data, a server crashing should not affect the reliability of the application. With the crash of one server, another server kicks in and maintains performance since the data and software have been replicated.

With the actors accessing the application through SaaS, monitoring the system will be mostly the responsibility of AWS. If there are crashes or failures in the system due to communication errors involving network connectivity, appropriate alerts will be sent to the affected actor. Server failure alerts will be sent and handled by the AWS team responsible for running the SaaS application.

References

5GStore. 2023. “How Much Data do I Need for My Business.” January 31, 2023. https://5gstore.com/blog/tag/credit-card-transactions/.

Amazon. 2018. “Introducing AWS Auto Scaling.” January 16, 2018. https://aws.amazon.com/about-aws/whats-new/2018/01/introducing-aws-auto-scaling/#:~:text=With%20AWS%20Auto%20Scaling%2C%20your,and%20Amazon%20CloudWatch%20monitoring%20fees.

Amazon. n.d. “Amazon EBS Pricing.” Accessed April 27, 2023. https://aws.amazon.com/ebs/pricing/#:~:text=With%20Amazon%20Elastic%20Block%20Store,until%20you%20release%20the%20storage.

Amazon. n.d. “Amazon EC2.” Accessed April 27, 2023. https://aws.amazon.com/ec2/instance-types/#:~:text=All%20instances%20have%20the%20following,EBS%2Doptimized.

Amazon. n.d. “Amazon Elastic Block Store (EBS).” Accessed April 27, 2023. https://aws.amazon.com/ebs/.

Amazon. n.d. “USB Flash Drive.” Accessed April 27, 2023. https://www.amazon.com/Lexar-LJDF35-32GBNL-JumpDrive-Fingerprint-Silver/dp/B07GSMSP34/ref=sr_1_2?crid=3S8RGW1T28A5A&keywords=Lexar+Jumpdrive+Fingerprint+F35+%2832+GB%29&qid=1682457121&sprefix=lexar+jumpdrive+fingerprint+f35+32+gb+%2Caps%2C180&sr=8-2.

COX. n.d. “Popular Business Services.” Accessed April 27, 2023. https://www.cox.com/business/solutions/small-

business.html?sc_id=cblfo_cbsem_cbint_cp-p-i-izzz-23tz_cbgoog_std_cbnet_intsmallbizgen_tier3_brand_exact&phonelabel=control4&cq_src=google_ads&cq_cmp=19560716829&cq_con=144773836506&cq_term=cox%20small%20business%20internet%20plans&cq_med=&cq_plac=&cq_net=g&cq_pos=&cq_plt=gp&&ca.cr=644914863799&ca.kw=cox%20small%20business%20internet%20plans&ca.mt=e&ca.mp=Google&cb.device=c&ca.target=kwd-658094196895&SRCHCAMPAIGN=COXBU_SEM_STD_NC_Tier3_Internet_Brand_Exact_Google&SRCHKW=cox+small+business+internet+plans&ds_kid=43700074839826523cq_src=GOOGLE&cq_cmp=71700000104883869&cq_con=Internet+-+Small+Business+-+General&cq_term=cox+small+business+internet+plans&cq_net=g&cq_med=COXBU_SEM_STD_NC_Tier3_Internet_Brand_Exact_Google&cq_plt=gp&gclid=CjwKCAjw9J2iBhBPEiwAErwpedSd-8l9-xQf7pLOdMyZYQSAHmSqWbgfClFn94X_mp2jVWCmT1vbqhoC_n4QAvD_BwE&gclsrc=aw.ds.

Fabregas, Krista and Bottorff, Krista. 2023. “10 Best Credit Card Processing Companies 2023.” April 3, 2023. https://www.forbes.com/advisor/business/software/best-credit-card-processing-companies/.

Gupta, Amit. 2023. “Top 10 Biggest Franchises In The World In 2023.” April 12, 2023. https://drfranchises.com/largest-franchises-in-the-world/.

Hill, Simon. 2023. “The Best USB Flash Drives. These WIRED-tested Memory Sticks are a Virtual Filing Cabinet in your Pocket.” January 13, 2023. https://www.wired.com/gallery/best-usb-flash-drives/.

Horowitz, Daniel. 2020. “CPU Cores: How Many Do I Need?” August 24, 2020. https://www.hp.com/us-en/shop/tech-takes/cpu-cores-how-many-do-i-need#:~:text=When%20buying%20a%20new%20computer,want%20at%20least%206%20cores.

Joshi, Vaidehi. 2019. “Scalability: Growing a System in Different Directions.” February 13, 2019. https://medium.com/baseds/scalability-growing-a-system-in-different-directions-ae16469c4cb3.

Kiernan, John S. 2022. “How Credit Card Transaction Processing Works.” August 12, 2022. https://wallethub.com/edu/cc/credit-card-transaction/25511.

Lemire, Daniel. 2022. “Memory-level Parallelism: Intel Ice Lake versus Amazon Graviton 3.” June 7, 2022. https://www.google.com/search?q=what+does+graviton3+compare+to&oq=w&aqs=chrome.2.69i60j69i59l3j69i60j69i61l2j69i60.8292j0j4&sourceid=chrome&ie=UTF-8.

Mayersen, Isaiah. 2022. “Amazon’s Graviton3 CPU has 7 Chiplets, 64 Cores, and Tri-Socket Motherboards.” May 28, 2022. https://www.techspot.com/news/94750-amazon-graviton3-cpu-has-7-chiplets-64-cores.html.

McCain, Abby. 2023. “22 McDonald’s Statistics [2023]: Restaurant Counts, Facts, and Trends.” March 21, 2023. https://www.zippia.com/advice/mcdonalds-statistics/#:~:text=McDonald's%20Statistics%20FAQ,world%20buy%20food%20from%20McDonald's.

Murray. 2023. “Payment Method Statistics.” February 28, 2023. https://www.bankrate.com/finance/credit-cards/payment-method-statistics/.

Nakedible. 2011. “How Much Data, On Average, Does a Credit Card Transaction Use?” September 1, 2011. https://stackoverflow.com/questions/7272273/how-much-data-on-average-does-a-credit-card-transaction-use#:~:text=Depends%20on%20what%20you%20mean,track%202%20is%2040%20characters.

Narayanan, Harihara Krishnan. 2019. “Adopting AWS VPC Endpoints at Square. Secure Communication Between Data Centers and the Cloud.” December 20, 2019. https://developer.squareup.com/blog/adopting-aws-vpc-endpoints-at-square/#:~:text=Square%20uses%20a%20Shared%20VPC,a%20centrally%20managed%20AWS%20account.

Sabanoglu, Tugba. 2022. “Average Monthly Retail Store Transactions Worldwide as of 2018, By Retail Segment.” February 3, 2022. https://www.statista.com/statistics/1012332/average-monthly-retail-store-transactions-by-category-worldwide/.

Shilov, Anton. 2021. “Specs and Prices of Intel’s Xeon Scalable ‘Ice Lake-SP’ Leak.” March 31, 2021. https://www.tomshardware.com/news/intel-ice-lake-sp-more-specifications-leak.

Smith, Marcus. 2022. “How Many Concurrent Requests can a Web Server Handle?” March 28, 2022. https://developerpitstop.com/how-many-concurrent-requests-can-a-web-server-handle/.

Square. n.d. “Square Stand Restaurant Station with Square Terminal.” Accessed April 27, 2023. https://squareup.com/shop/hardware/us/en/products/terminal-pos-kit-restaurant.

Steen, van Maarten and Andrew Tanenbaum. 2017. Distributed Systems. CreateSpace Independent Publishing Platform. https://www.distributed-systems.net/index.php/books/ds3/.

Transparent. n.d. “What Is Batch Processing?” Accessed April 27, 2023. https://www.trytransparent.com/question/batch-processing/.

van Steen, Maarten and Andrew Tanenbaum. 2017. Distributed Systems. CreateSpace Independent Publishing Platform. https://www.distributed-systems.net/index.php/books/ds3/.

Vantage.sh n.d. “r7g.medium.” Accessed April 27, 2023. https://instances.vantage.sh/aws/ec2/r7g.medium.

Villinger, Sandro. 2022. “SSD vs HDD: What’s the Difference?” June 3, 2022. https://www.avast.com/c-ssd-vs-hdd.