Migration of Batch Processing Systems in Financial Sectors to Near Real-Time Processing

Vigneshwaran Kennady*, Priya Mayilsamy**

*Senior Manager, Software Engineering, Capital one, Plano, Texas, USA
**Software Development Manager, Amazon Web Services, Dallas, Texas, USA

DOI: 10.29322/IJSRP.12.07.2022.p12755
http://dx.doi.org/10.29322/IJSRP.12.07.2022.p12755

Paper Received Date: 29th June 2022
Paper Acceptance Date: 12th July 2022
Paper Publication Date: 20th July 2022

Abstract- Technology has evolved and has become part of people's lives today. Information systems (IS) are now embraced in all spheres of management. Essentially, this is because of its efficiency and reliability in different fields. Knowledge of IS has enabled the control of advanced sectors (1). IS helps distinguish raw and factual data, which are helpful to any firm. A company must always follow a specific protocol while handling these transactions, whether placing orders for a customer or processing many invoices. The two most popular methods are batch and real-time processing. Batch processing is the procedure to process a large volume of data all at once whereas real time is the procedure to process data instantaneously record by record usually in a matter of seconds or milliseconds. They both solve different needs in financial sector and the industry chose one vs other depending on the criticality and complexity of the need, users of the outcome (internal vs external) and overall customer satisfaction index. In addition to above two methods, near real time processing is the process of being able to almost instantaneously analyze data that is streaming from one device to another. The financial sector is looking for solution to migrate batch processing system to near real time systems using streaming solutions like Apache Kafka and AWS Kinesis and re-use real time systems wherever possible for better customer experience and lower operational cost.

Index Terms- Batch processing, financial sector, Apache Kafka, AWS Kinesis, Near real time processing, Event streaming.

I. INTRODUCTION

This article talks about the benefit of using near real time processing in financial sector. Dealing with data has become a key element in today's business circles. Handlers must learn batch processing to ease the handling of sensitive data [4]. Batch processing systems are part of technology incorporated in today's business sphere. In most cases, to gather, transfer, preserve, retrieve, modify, and display data, the IS components must cooperate. The IS components interact together to produce data, and the information is complementary to the people and organization in filtering, processing, and distribution of data [9]. The hardware as a component refers to the physical equipment and its associate input and output devices that communicate for a common goal. Generally, the hardware is supported by the software, which refers to the internal programs in a computer.

Information system helps the organization in management functions. Through the information system, managers receive information regarding the performance of the employees in their roles. When there is a weakness, the management seeks ways of addressing the issue for improved versions. Moreover, the IS helps the organization's top managers strategically make decisions to improve the organization's goals and objectives [3]. In decision-making, managers enter the data regarding the problem at hand. On the other hand, the information systems offer a solution to the problem by seeking information from various computer databases. Furthermore, the IS assists in the performance of the executive roles in the organization. IS allows the top managers to closely monitor the organization's performance, specifically the workers' performances, assessing the organization's external environment since it is the main factor determining the organization's direction [3]. The external environment in business settings includes the market of the products and services, competition from other business firms, and a source of crucial resources. Therefore, the IS enables monitoring trends in the external environment, helping the organization make informed decisions.

When processing data through batch systems a sizable static dataset, the result is delivered after the calculations are complete [2]. Generally, it requires a longer time compared to real-time data processing due to the grouping. Since business circles entail handling data from various sources and for different purposes, batch processing must suit the high data analysis. Systems that handle huge data quantities use batch processing in their operations. The Batch system's ability to have no user interface makes the application more realistic and widely embraced due to its easy accessibility. Batch processing systems are used in the financial industry in analysis, controlling, compiling organization, and report generation. Generally, its reliability makes it a necessity in business operations today.

In the real-time data processing the input and output of data is available immediately upon processing. The real time framework takes less time and requires a continuous data flow as output occurs immediately after input [5]. An example for real-time processing is fast and interactive queries on big data warehouses, in which user wants the result of his queries in seconds rather than in hours. Principally, the goal of real-time processing is to provide solutions that can process big data very fast and interactively.
In the near-real time data processing, data are produced by producers, who create a stream of them, and consumed by consumers, who respond to them. This type of architecture enables applications to act on events as they happen. We can produce and react to a huge number of events in real time using scalable event driven architecture. Since neither the producers nor the events are aware that the consumers are listening to them, this architecture is also loosely coupled.
In this article, we talk about a financial sector use case which is currently running through batch process and how the system can be migrated to near-real time system using Apache Kafka.

II. FINANCIAL SECTOR USE CASES

Speed in performing E-Banking services is a determining factor of customer satisfaction according to Parasuraman, Zeithaml, and Berry (1985). Efficiency in terms of quick speedy service is also confirmed by Wirtz and Bateson (1995) and Khadem and Mousavi (2013). Liao and Cheung (2002) find reliability as one of the most important features that customers seek in evaluating their E-Banking service quality. A similar result was also obtained in an empirical study done by Kettinger and Lee (2005)[12].

In financial sector, providing customer service when it is needed is essential for improving client satisfaction, which will ultimately boost sales and the bank's image. Financial industries typically use both batch and real-time systems to manage customer support, depending on what is anticipated from the customer experience. While services that engage directly with consumers are frequently pushed toward real-time processing, those that are opaque to customers are frequently pushed toward batch processing solutions.

A. Product upgrade

Let’s consider a credit card company which offer product upgrade as a feature. The product upgrade is available to the customers in two ways 1. Whenever they reach out to the company through any servicing channel like Mobile app, Online, automated phone and request for a product upgrade which is often called as reactive 2. The credit card identifies the potential customer who are eligible for product upgrade and upgrade their credit card whoever has given a consent upfront to do which is called as proactive. In this scenario, usually reactive use cases are handled through the real time processing whereas proactive use cases are handled through batch processing system.

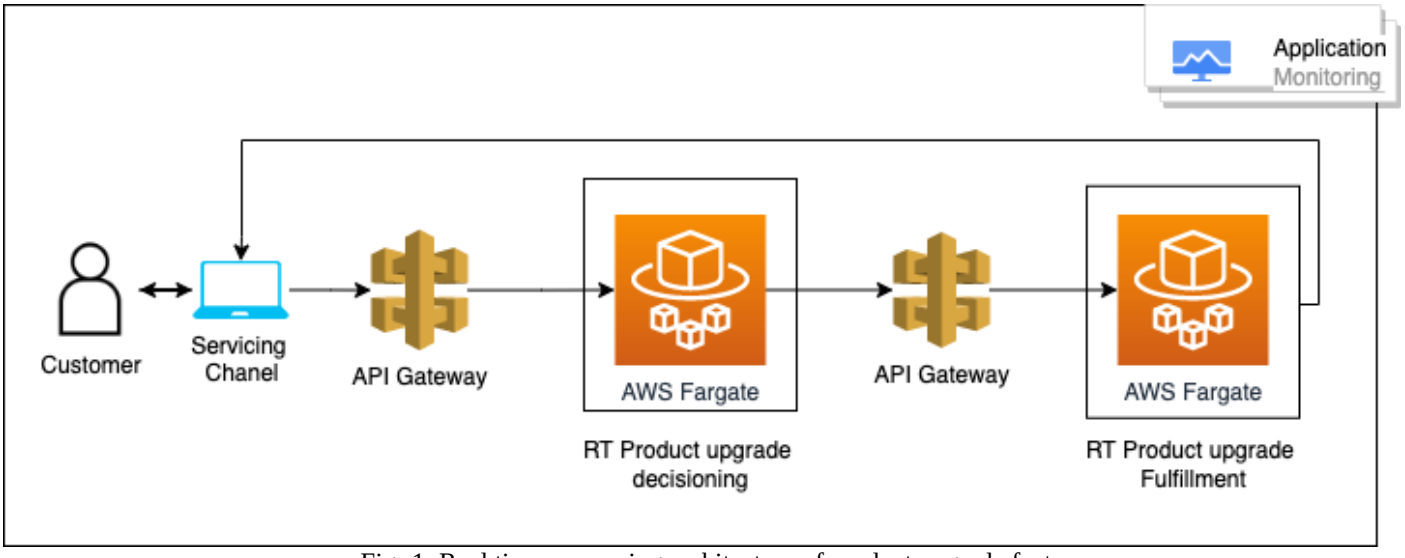

Fig -1: Real time processing architecture of product upgrade feature

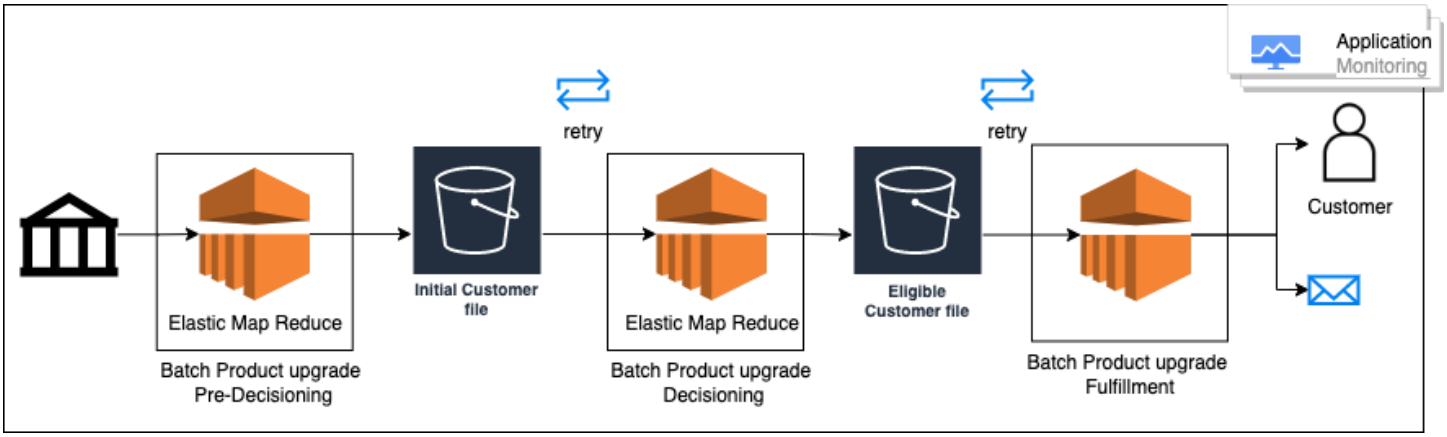

Fig -2: Batch processing of architecture of product upgrade feature

In this use case, credit card company spend duplicate operational cost as it need to maintain both batch and real time system but from the customer standpoint, both processes result in the product upgrade on their account. In addition, any business changes on the product upgrade often need to happen in both systems which duplicate the time to market cost. As batch system usually uses the batch files, deciding a customer for product upgrade usually happens with older data whereas real time system uses most accurate data which enables accurate decisioning and better customer experience.

B. Reward management

Let's consider another use case in the credit card company which offer rewards to the customer as points based on the credit card usage. Rewards are awarded to the customer once the transaction on credit card is approved by the bank. Once the transaction is approved, the transaction details are stored in analytical database. A daily batch job trigger every day and queries all the eligible transaction from the analytical database and feeds to a batch reward management decisioning engine. The decisioning system calculates the eligible reward for the account and add back to the customer account through a batch fulfillment process. Compared to critical needs of customers, reward management is often considered as non-critical and awarded to the customer through batch process daily instead of awarding it in real time upon the approval of the credit card transaction. This causes bad customer experience as the customer need to wait for at least a day or two days before they see the reward their account.

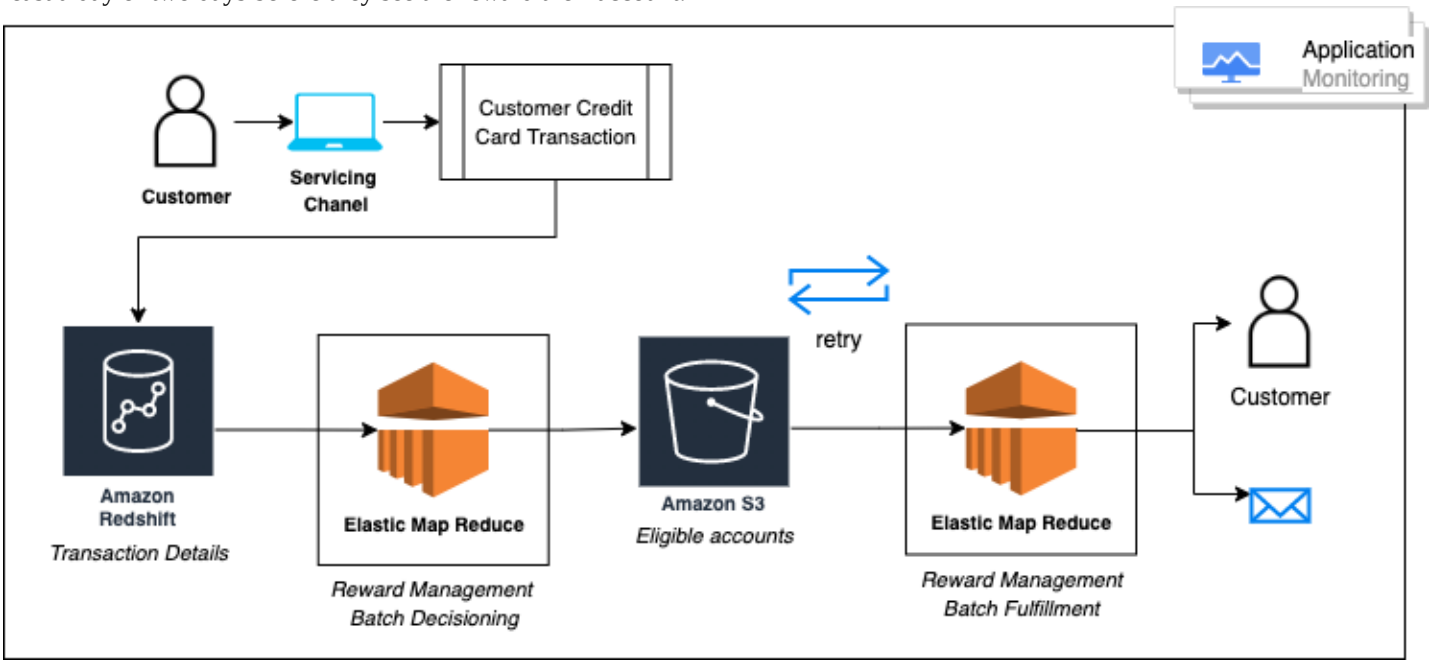

Fig -3: Batch processing architecture of reward management feature

One of the ways to handle the mentioned issues with batch processing and use the power of real time processing system for better customer experience and to reduce the operational cost for the company i

s to migrate to near-real time processing systems.

III. BATCH PROCESSING

In software world, Batch systems usually consists of Extract, transform and load (ETL) tools are software applications responsible for data extraction from multiple sources, cleansing, customization, and insertion into a data warehouse. ETL tools integrate data, which enables the building of the enterprise data warehouse.[11]

A. Batch processing architecture

The batch system functions primarily from three different architectural layouts. Essentially, these components serve as the input, processing, and output. All three are put together to create a seamless batch process together. The dataset is usually persistent, large, and bounded [2]. Essentially, this is because it serves in both the input and the output processes. As information from a queue is obtained, the processor computes the data according to the seller's requirements. Therefore, the output is not raw data but added data ready for use. The output after processing may be printed out or stored in a database awaiting the next step if required. Additionally, the Batch system is function-oriented and not object-oriented. The reason is that data in it is processed serially and does not need to be obtained in state. Such is seen when printing invoices, financial statements, and issuing renewals for invoices.
Batch processing works effectively for calculations where having access to all records is necessary. For example, datasets must be seen rather than a collection of individual records for calculating totals and averages. In essence, for the length of the computations, these operations demand that the state be maintained [2]. A bank statement is intended to demonstrate in detail what transpired with a person's account over the previous month, including their spending patterns and any expenses made. Most bank statements begin by compiling all deposits, giving customers a clear picture of what was deposited into their accounts throughout the previous month [7]. Clients can then see a summary of the withdrawal activities. The account balance at the start of the period is included in the summary, followed by the outstanding balance at the end of the time after all deposits and withdrawals have been tallied.
The following are some of the basic features of a batch system

A.1 Apache Hadoop

The first big data platform to experience considerable growth in the open-source community was Hadoop. Hadoop updates the algorithms and element stack to provide accessibility for large-scale batch operations [2]. They successfully tackle the most common business concerns like fraud, financial crimes, and data breaches. Banks can detect and reduce fraud by analyzing point of sale, authorization, and transactions, among other aspects. The time and resources needed to execute these tasks are significantly reduced thanks to big data, which assists in identifying unexpected trends and informing banks of them.
Hadoop has numerous strengths. Some of its strengths come from the MapReduce model. For example, easy programming model, near-linear speedup and scalability, and fault tolerance are three major features. Besides these, Hadoop itself provides some extra features like different schedulers, more sophisticated and complex job definitions using YARN, high available master machines, pluggable I/O facilities, and etc. Hadoop provides the basic platform for big data processing. For more usability, other solutions can be mounted over Hadoop [7]. Major examples are HBase for storing structured data on very large tables, Pig and Hive for data warehousing, and Mahout for machine learning. Although Hadoop, i.e., standard MapReduce, has numerous strengths, but it has several shortcomings too. MapReduce is not able to execute recursive or iterative jobs inherently [12]. Total batch behavior is another problem. All of the input must be ready before the job starts and this prevents MapReduce from online and stream processing use cases. The overhead of framework for starting a job, like copying codes and scheduling, is another problem that prevents it from executing interactive jobs and near real-time queries. MapReduce cannot run continuous computations and queries, too.[13]

A.2 Hadoop Distributed File System (HDFS)

The distributed file system component known as HDFS controls retention and synchronization among network nodes. Essentially, HDFS is the storage section of the Hadoop applications. HDFS keeps data accessible despite any inevitably occurring host malfunctions [2]. Additionally, it is used as the data source to preserve intermediate processing outcomes and the results of the final calculations. In banking, MapReduce processes the data, HDFS archives the data, and Yarn divides the responsibilities

B. Advantages of Batch Processing Systems

Using batch processing has a few benefits, including decreased cost being the primary factor. Batch processing has become more affordable due to the reduced frequency of the procedures. For instance, most banks do offer bank statements for free to their clients. Businesses view batch processing as a fixed expense. The total cost for each step would decrease the more batches processed in a given period. The well-proven batch processing approach of Apache Hadoop and its MapReduce computing engine is better suited for handling

massive datasets where speed is not a significant consideration [2]. The final benefit of batch patch processing is that it allows the company to edit the data before it is distributed. Transaction changes can be made without creating additional processes, if necessary.

C. The Downsides of Batch Processing Systems

Batch processing has a lot of drawbacks in addition to these advantages. One example is the loss of transactions within a larger batch that needs to be handled. In essence, it could be challenging to identify and correct an error if the cluster contains too much data. Think about a business that operates its accounts receivable once every month. When a lot is processed at once, there can be mistakes with some of the bills that are sent out. However, this issue can be resolved by splitting the batch processing into two or even three distinct processes. Overall, batch processing of bank statements is only most effective when a large volume of data needs to be sent out at once. Grouped data and generated batches are not easy to monitor during the process. [2], this system is generally slow since it depends on permanent storage, reading, and recording for each activity. The inability to watch it means it will take considerable time to debug. The reason is that identifying the mistake traces back to a long chain before it can be identified. Having time to debug is a risk as the information safety through the process is uncertain. Errors in the process come costly. An instance is when a process malfunctions and there is a delay in sorting the issue. The delay means the entire process will have to be held to an unknown period, complicating the process, and delaying operations in the organization.

IV. REAL TIME PROCESSING

Unlike batch systems, real-time processing has a more cost-effective output [5]. The main reason why financial sectors are opting for migration is due to risk management. In real-time processing, the risk is noted, and management strategies are implemented to counter the threat almost immediately. However, more time may be required to sort the batches back to single data in Batch processing to identify the risk and set up management strategies. The process may take some time and delay output as data inflow must be postponed. Convenience being the critical marker of today's e-commerce platforms, real-time personalization comes in handy. Creating top-tier personalized customer service is what is embraced in today's world.

Most of the above-stated challenges are almost non-existent in real-time processing. The minimum downsides of real-time processes are why most organizations, especially financial sectors, are trying to migrate to real-time processing. Compared to human data input, computerized bank data extraction software simplifies the underwriting process.

For the banking use case mentioned above, we are considering real time systems which consists group of microservices orchestrated through API layer.

Micro services have risen in popularity as a novel approach to designing distributed applications. It is made up of several loosely connected software components that are designed to be independent, automatically deployable, and cohesive. This architecture claims to meet continuing software development requirements such as resilience, continuous delivery, enhanced maintainability, and scalability. Each micro service has a smaller codebase, which makes it easier for developers to grasp and hence more productive. Micro service design also allows for the organization of work across many development teams, as well as improved testability and deployment autonomy. In contrast to a monolithic design, where one failed component knocks down the entire stack, a micro service architecture makes it simple to isolate faults and detect concerns.

V. NEAR REAL TIME PROCESSING

Near real-time processing is when speed is important, but processing time in minutes is acceptable in lieu of seconds. The term "near real-time" or "nearly real-time", in telecommunications and computing, refers to the time delay introduced, by automated data processing or network transmission, between the occurrence of an event and the use of the processed data, such as for display or feedback and control purposes. For example, a near-real-time display depicts an event or situation as it existed at the current time minus the processing time, as nearly the time of the live event. The distinction between the terms "near real time" and "real time" is somewhat nebulous and must be defined for the situation at hand. The term implies that there are no significant delays. In many cases, processing described as "real-time" would be more accurately described as "near-real-time". The common pattern for near real time processing is through event driven architecture.

A. What's an event?

Any notable occurrence or state change for system software or hardware is referred to as an event. A message or notification provided by the system to inform another component of the system that an event has occurred is not the same thing as an event.

B. Types of event driven architecture models

B.1 Pub/sub model

This messaging architecture, which maintains track of subscribers, is based on subscriptions to an event stream. With this strategy, subscribers who need to be informed are notified once an event occurs or is published.

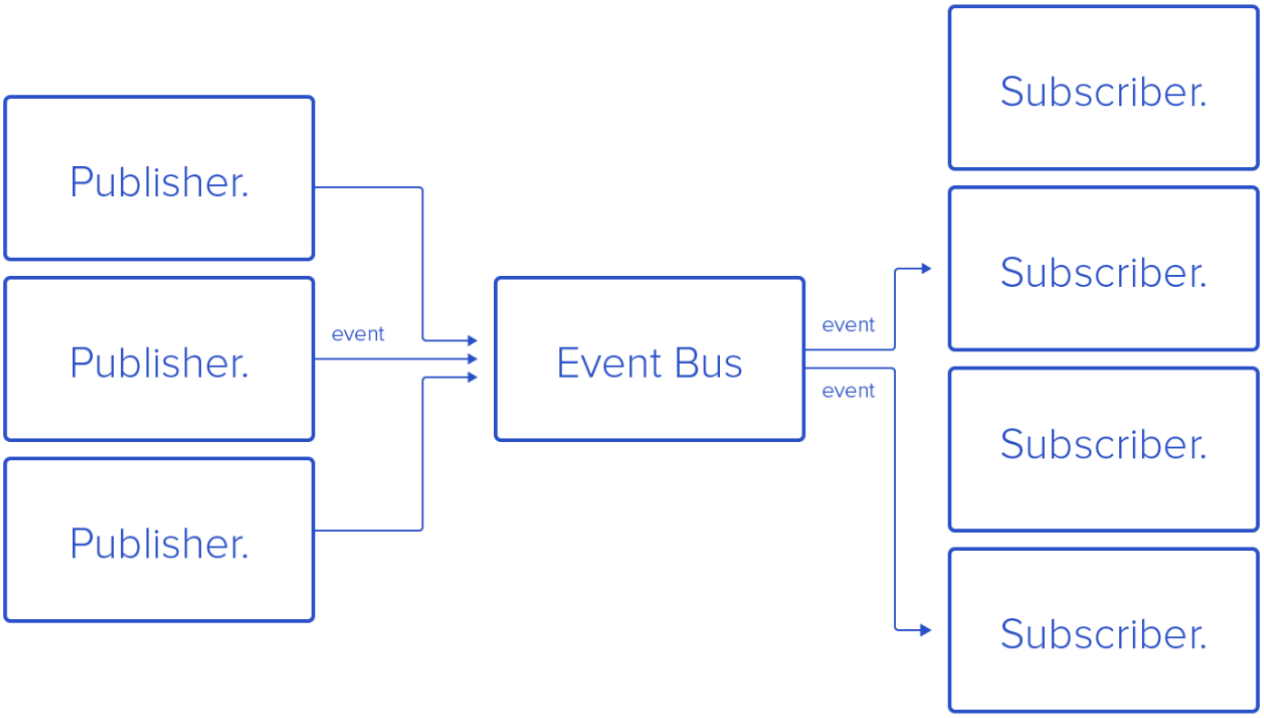

Fig -4: Pub/Sub Model

B.2 Event- streaming model

Event-processing programs aggregate information from distributed systems in real time, applying rules that reveal key patterns, relationships, or trends. An "event stream" is a sequence of business events ordered by time. With event stream processing you connect to all data sources and normalize, enrich, and filter the data. You can then begin to correlate events and over time you see patterns emerge that describe events you care about. You add contextual data to ensure proper interpretation of events and then you apply real-time business logic and rules or machine learning to trigger action.[15]

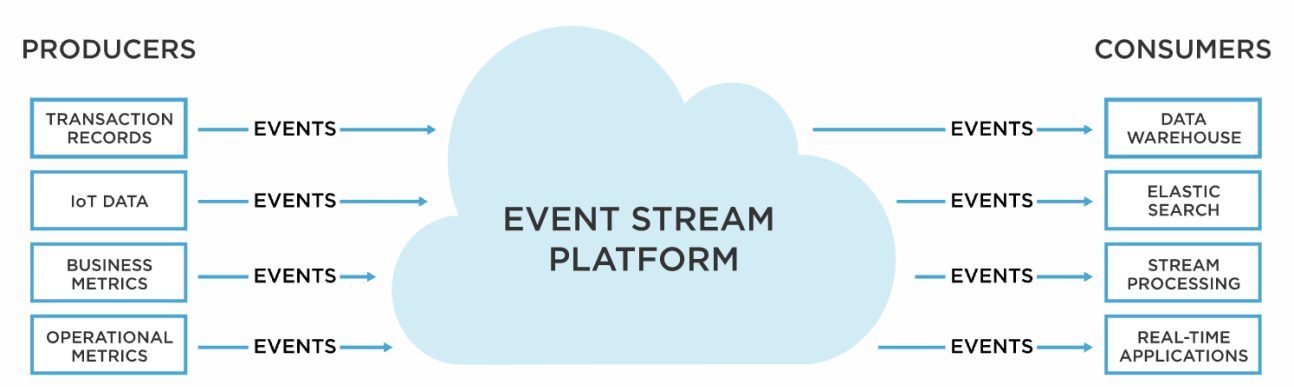

Fig -5: Event Streaming Model

C. Options for Near real time processing

The available software architecture to move the batch system to near real-time processing include Apache Kafka, AWS Kinesis, AWS SQS, spark streaming.

C.1 Apache Kafka

Kafka is a solution to the real-time problems of any software solution, that is, to deal with real-time volumes of information and route it to multiple consumers quickly. Kafka provides seamless integration between information of producers and consumers without blocking the producers of the information and without letting producers know who the final consumers are. Kafka [2] is distributed and scalable and offers high throughput. On the other hand, Kafka provides an API like a messaging system and allows applications to consume log events in real time. [14]

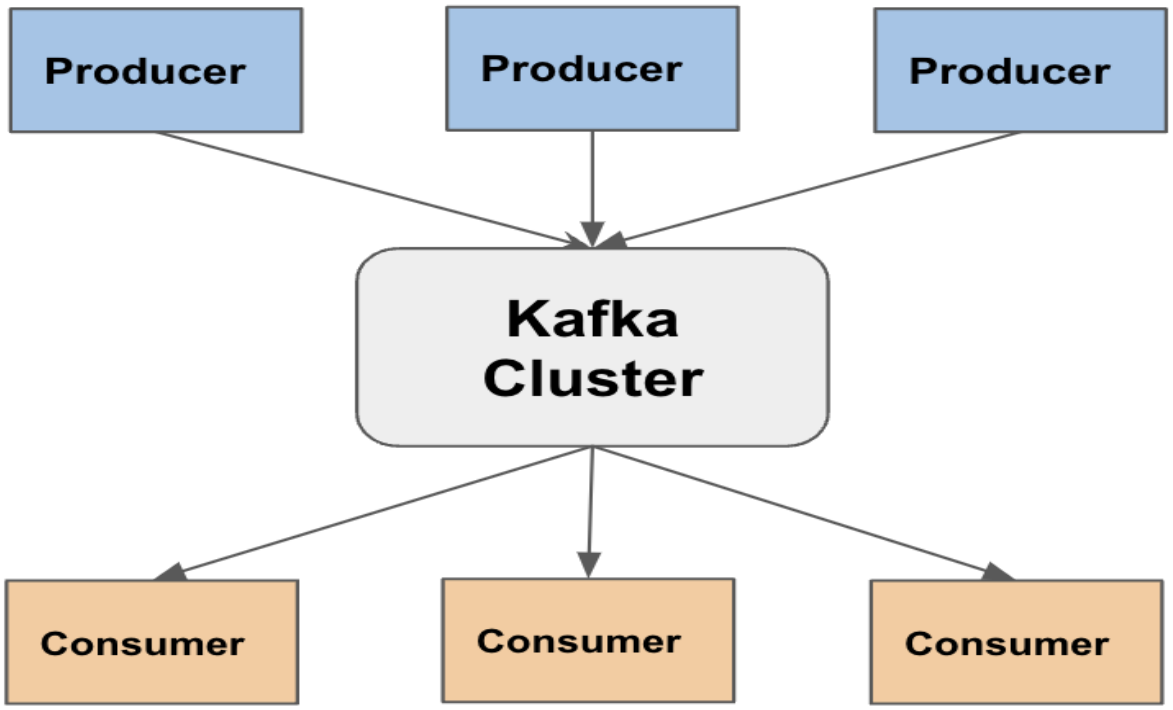

Fig -6: Kafka Framework

Kafka is a distributed, partitioned, replicated commit log service. Kafka [3] maintains feeds of messages in categories called topics. We'll call processes that publish messages to a Kafka topic are producers. And we'll call processes that subscribe to topics and process the feed of published messages are consumers. Kafka is run as a cluster comprised of one or more servers each of which is called a broker. At a high level, producers send messages over the network to the Kafka cluster which in turn serves them up to consumers like this in Fig.4.

Producers publish messages to Kafka topics, and consumers subscribe to these topics and consume the messages. A server in a Kafka cluster is called a broker. For each topic, the Kafka cluster maintains a partition for scaling, parallelism, and fault-tolerance. Each partition is an ordered, immutable sequence of messages that is continually appended to a commit log. The messages in the partitions are each assigned a sequential id number called the offset.

C.2 AWS Kinesis

Amazon Kinesis, better known as AWS Kinesis, is a new software that serves real-time data streaming. Large-scale data input and real-time analysis of streaming data are made possible by AWS Kinesis Streams [8]. Essentially, it allows for record ordering and the reading or replaying of records in the same order. The system can briefly handle large chunks of data, thus ensuring efficiency in the data computation process. AWS Kinesis is better has made modern-day consumer interaction easier as it generates a new reaction to new information. Moreover, it gives one the ability to create massive data. They have ensured a seamless transition from traditional to modern, batch, and real-time. Generally, AWS provides the most comprehensive and in-depth selection of machine learning solutions that can be used to improve compliance and verification, automate procedures, enhance consumer experiences, and detect fraud.

In serverless systems, publish-subscribe or pub-sub messaging is a type of asynchronous service-to-service communication. Any communication sent to a subject is immediately received by all the users in a pub-sub paradigm. Essentially, this enables software components like AWS Kinesis to connect the topic to receive and send messages. On the other hand, in event streams, decisions are made on data as soon as it is produced or broadcast. Generally, this can be made possible by Apache Kafka, eliminating end-to-end dependency and ETL duplication.

VI. Software Architecture to Move the Batch System to Near Real-Time Processing

Out of the options available to enable near real time processing, we used Apache Kafka and below we discuss the approach of financial sector need through near real time processing.

A. Near real processing architecture of product upgrade feature

As mentioned above, batch processing of product upgrade produces a file which consist of data of eligible customer for product upgrade. Instead of processing the files through multiple batch systems to do decisioning (inclusion, exclusions, identify offer) and fulfillment (product upgrade, notification), eligible customer file can be streamed through Kafka producer module. The real time application which processes the real time request from customers in Fig2 can be enabled to support REST API protocol as well as SDP consumer pattern. This enables us to re-use the real time processing application for batch need as well.

The one of the disadvantages to the above approach, is that existing batch system can do reconciliation of input to output at various stages of batch processing and retries in case needed whereas in real time processing, there is a need to build a module to reconciliation and retry in case of any failures. The reconciliation and retries are enabled in this pattern through dead letter queue which is part of recon/retry module which publish the data back to same Kafka topic in case of any failure due to system error.

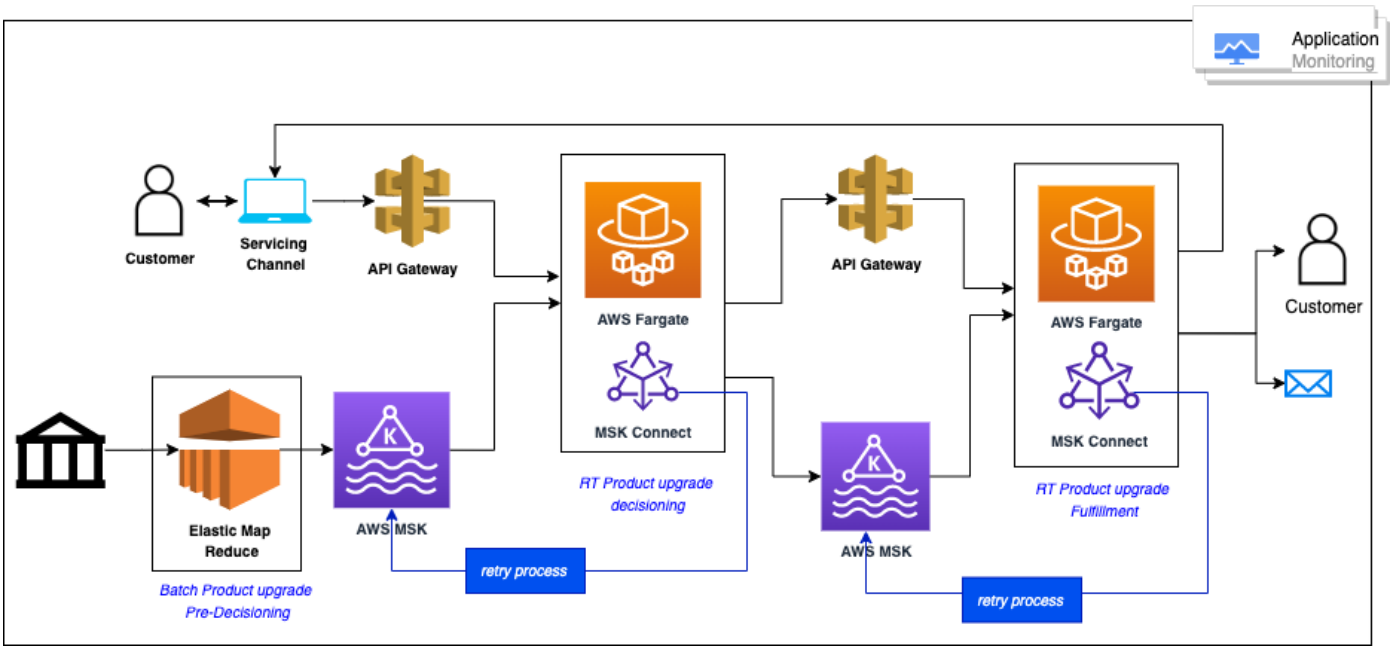

Fig -7: Near real time processing architecture of product upgrade feature

B. Near real processing architecture of reward management feature

In case of reward management architecture to near real time, as soon as the customer transaction is approved, the event can be published to a Kafka topic through a publisher module which can be consumed by a microservice which takes care of updating the reward to the customer account instantaneously. As this process happens asynchronously, monitoring can be built to support recon and retry in case of any failure. Once the reward is added successfully to the customer, the events published from the reward management can be fed to another Kafka topic which is consumed by a microservice to send a communication back to customer in near real time.

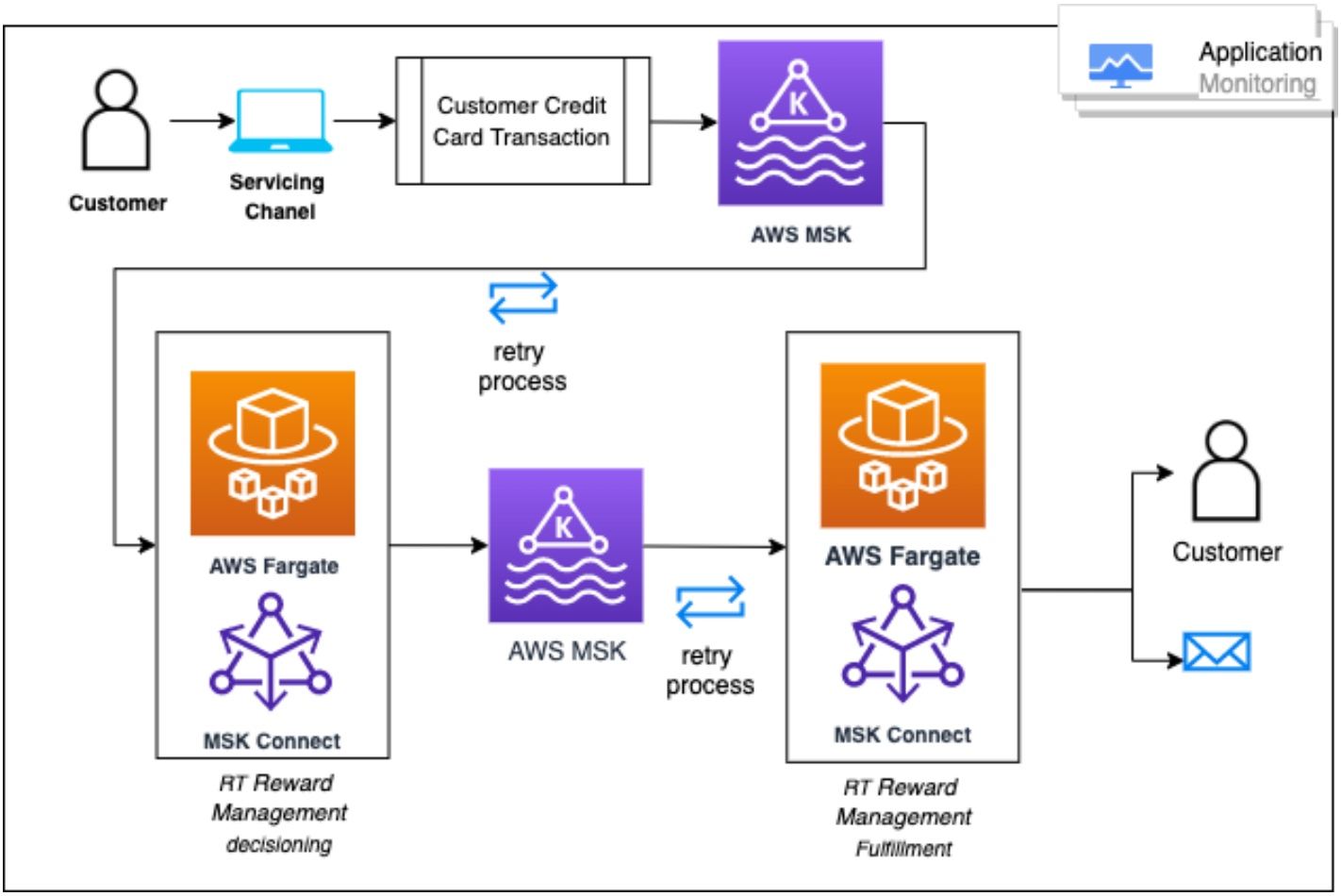

Fig -8: Near real time processing architecture of reward management feature

C. Comparison of cost and execution time

Below work is conducted to process 1M transaction using batch and near real time processing using above discussed architecture . Best case scenario includes no failure happened during the processing window. The processing time of batch process include the time taken in between batch process considering the batch job initiate every 12 hours. The processing time of near real time process consider the systems process at the rate of 100 TPS.
Worst case scenario include failure retries happen during the failure which increase the overall processing time of batch process as well as reduced the TPS of near real time process to 50TPS.

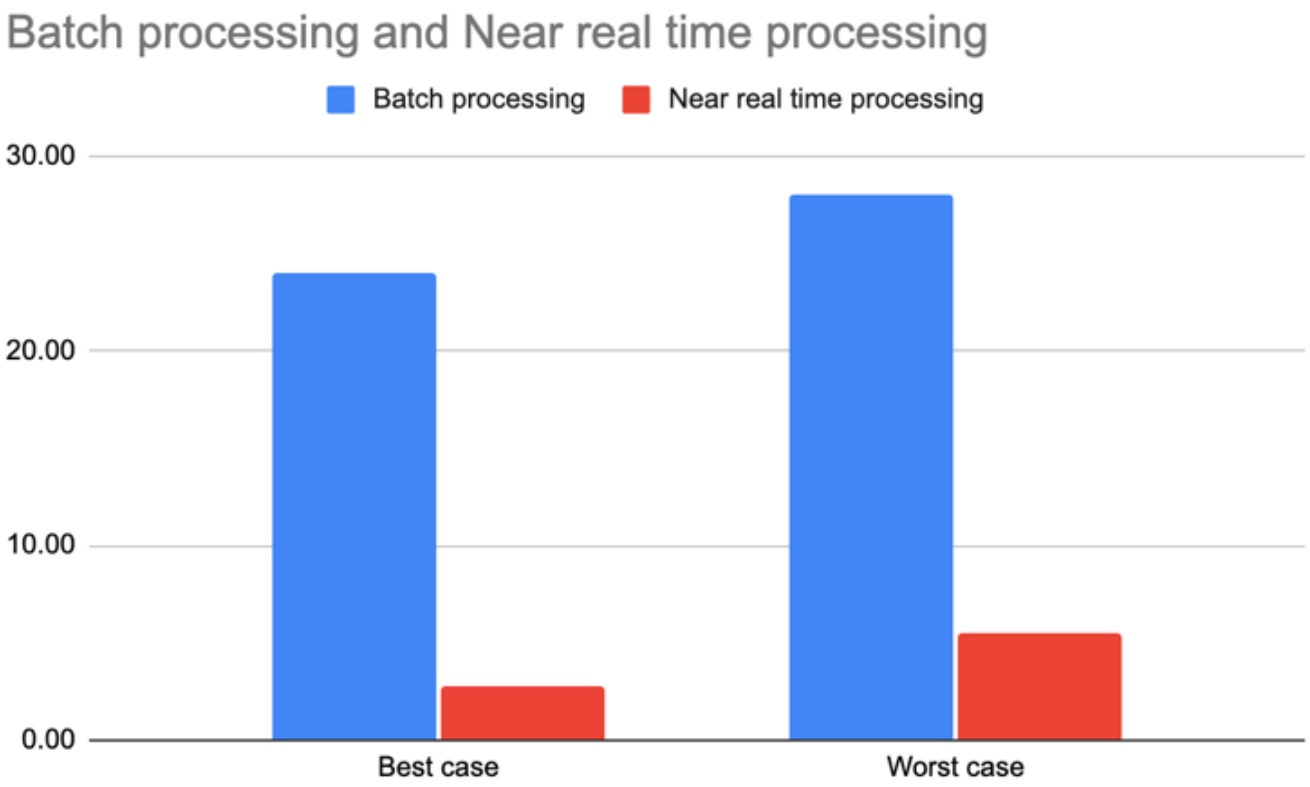

Fig -8: comparison of processing time between batch vs near real time processes.

VII. CONCLUSION

Essentially, because many banks and other financial institutions' operations, systems, and architecture are still batch-based for the non-time critical proactive applications and this in turn causes different customer experience as well as takes times more time which reduces customer satisfaction index overall. Along the same line, the financial institution is spending extra cost on maintaining both real time and batch. Considering the reactive nature of real time application which often need to respond in seconds to milliseconds as well as throw error in case of any issue. Hence just migrating batch applications to real time application tends to be over kill which could increase error rate based on the volume. Hence another option for financial institution is to switch their batch processing to near real-time processing. The benefit of this transition are, processes which needn't be real time can be asynchronously processed by re-using the efficiency of real time and be done in minutes as well as infrastructure, operational and maintenance cost can be reduced and consolidated. The transition can be gradual, beginning with processes that make the most sense for each bank separately, then broadening as advantages become apparent. The great news is that even with this strategy, banks will still see tangible benefits because near real-time processing enhances overall systems.

REFERENCES

[1] Dastmalchian, A., Bacon, N., McNeil, N., Steinke, C., Blyton, P., Satish Kumar, M., Bayraktar, S., Auer-Rizzi, W., Bodla, A. A., Cotton, R., Craig, T., Ertenu, B., Habibi, M., Huang, H. J., İmer, H. P., Isa, C. R., Ismail, A., Jiang, Y., Kabasakal, H., & Varnali, R. (2020). High-performance work systems and organizational performance across societal cultures. Journal of International Business Studies, 51(3), 353-388. https://doi.org/10.1057/s41267-019-00295-9

[2] Gurusamy, V., Kannan, S., & Nandhini, K. (2017). The real time big data processing framework advantages and limitations. International Journal of Computer Sciences and Engineering, 5(12), 305-312. https://doi.org/10.26438/ijcse/v5i12.305312

[3] Harun, H., Hashim, M. K., & Rahmat, A. (2018). Education information systems planning practices and performance of government agencies in Malaysia. Ideas: Jurnal Pendidikan, Sosial, dan Budaya, 4(1), 27-30. https://jurnal.ideaspublishing.co.id/index.php/ideas/article/view/61/17

[4] Jawhar, Q., Thakur, K., & Singh, K. J. (2020). Recent advances in handling big data for wireless sensor networks. IEEE Potentials, 39(6), 22-27. https://doi.org/10.1109/mpot.2019.2959086

[5] Kraetz, D., & Morawski, M. (2021). Architecture patterns—Batch and real-time capabilities. The Digital Journey of Banking and Insurance, Volume III, 89-104. https://doi.org/10.1007/978-3-030-78821-6_6

[6] Lessig, m. M. (2022, May 2). Five benefits of real-time processing. FIS. https://www.fisglobal.com/en/insights/what-we-think/2020/november/five-benefits-of-making-the-move-to-real-time-processing

[7] Marquit, M. (2022, January 15). What is a bank statement? The Balance. https://www.thebalance.com/what-is-a-bank-statement-5092371

[8] Modi, K. (2021, July 27). System Design — Choosing between AWS kinesis and AWS SQS. Medium. https://medium.com/nerd-for-tech/system-design-choosing-between-aws-kinesis-and-aws-sqs-2586c814be8d

[9] O'Connor, M., Conboy, K., & Dennehy, D. (2022). Time is of the essence: A systematic literature review of temporality in information systems development research. Information Technology and People. https://doi.org/10.1108/itp-11-2019-0597

[10] Pattanaik, M., & Umarkar, P. (2021, June 3). Is Apache Kafka the next big thing in banking? Oracle. https://oracle.com/financialservices/post/is-apache-kafka-the-next-big-thing-in-banking

[11] Nayem Rahman, Navneet Kumar & Dale Rutz (2016) Managing application compatibility during ETL tools and environment upgrades, Journal of Decision Systems, 25:2, 136-150, DOI: 10.1080/12460125.2016.1138392

[12] Hammoud, J., Bizri, R. M., & El Baba, I. (2018). The Impact of E-Banking Service Quality on Customer Satisfaction: Evidence From the Lebanese Banking Sector. SAGE https://doi.org/10.1177/2158244018790633

[13] Shahrivari, S. (2014). Beyond Batch Processing: Towards Real-Time and Streaming Big Data. *Computers*, *3*(4), 117–129. MDPI AG. Retrieved from http://dx.doi.org/10.3390/computers3040117

[14] Thein, K. M. M. (2014). Apache kafka: Next generation distributed messaging system. International Journal of Scientific Engineering and Technology Research, 3(47), 9478-9483.

[15] https://www.tibco.com/reference-center/what-is-event-streaming

AUTHORS

First Author – Vigneshwaran Kennady, Bachelor of Technology, Senior Manager, Software Engineering, Capital one, Plano, Texas, USA, kennadyvignesh@gmail.com,+1804-382-7703

Second Author – Priya Mayilsamy, Bachelor of Technology, Software Development Manager, Amazon Web Services, Dallas, Texas, USA,priyamayilsamy11@gmail.com, +1 804-241-9824