Vol. 7, No. 2 | July – December 2023

Towards a Trustworthy and Efficient ETL Pipeline for ATM Transaction Data

Muhammad Ahmad Ashfaq [1], Nimra Haq [2], Usman Arshad[3], Muhammad Farooq[4], and Shuja ur Rehman Baig

Abstract:

ATMs generate vast amounts of data daily, which needs to be analyzed and stored. Dealing with this data also termed big data, is a complex task, and here comes the role of ETL pipelines. ETL pipelines need extensive resources for operations, and their performance optimization is necessary as data must be dealt with in near or even real-time. If the pipeline deals with financial data such as ATM transactions, steps should be taken to ensure the data's security, privacy, confidentiality, and integrity. This can be achieved using Blockchain technology. It is a distributed ledger technology having an immutable nature. It has significant advantages in terms of providing security, but it has disadvantages as well, such as low throughput and transactional latency. If blockchain is used in an ETL pipeline, it will affect the overall performance. So, to prevent the decline in performance, steps should be taken to optimize it. In this paper, we are using parallelization and partitioning as techniques to optimize performance. The primary goal here is to achieve maximum security while maintaining performance.

Keywords: *ETL Pipeline, Big Data, Blockchain, Performance optimization, Kafka, Spark*.

1. Introduction

Today, technology has digitized the finance sector and changed how customers interact with financial institutions and get their services. This change can be seen in the wide use of Automated Teller Machines(ATMs). ATMs play a vital role in providing certain banking services at any time with convenience and ease. As users perform transactions using ATMs, a considerable amount of valuable financial data is generated. Businesses want to use this data for their benefit as it can provide insights into customer behaviors and preferences when analyzed.

This data can also be used for better-aimed marketing, analyzing overall market patterns, and getting more business insights. The banking institutions can use this data to take finer steps to improve customer experience and optimize their operations.

Big data can be explained in terms of volume, velocity, and variety. It refers to massive datasets that can be wide-ranging (structured, unstructured, or semi-structured) and have complex structures. Ten percent of data collected and generated by businesses is structured, ten percent is semi-structured, and the rest is unstructured. These datasets can pose significant difficulty in storing, analyzing, and visualizing them. Big data analytics is the research process into such datasets to identify patterns and hidden correlations. The data generated by ATM daily transactions is so massive that it can be termed Big Data.

[1]FCIT, Faculty of Computing and Information Technology, Shahrahe Quaid-e-Azam Allama Iqbal Campus (Old Campus), Lahore, Pakistan
Corresponding Athor: shuja@pucit.edu.pk

Managing, processing, and analyzing this vast data offers great complexity and difficulty. So, robust and scalable data pipelines are required for this purpose. Data pipelines are classified into many categories, one of them is ETL. ETL stands for Extract, Transform, and Load. The ETL pipelines function like circulatory systems, moving data from its source to the intended destination and enabling near real-time analysis and decision-making. The source and the destination can be separated physically, and transformations may take place in between. Data is retrieved from various heterogeneous sources such as databases, APIs, or other structured or unstructured sources during the extraction phase. The extracted data can be of different formats, e.g., text files, images, videos, emails, XML, JSON, CSV, etc. The transformation phase is quite diverse. For instance, basic transformation may include replacing NULL values with a zero or removing duplicate values. Transformation may have joins, which can be complex sometimes, aggregation of rows, splitting of columns, etc. In this phase, the data is transformed to make it usable at the destination. In this phase, the transformed data is loaded into the destination. The destination can be a traditional or non-traditional database, visualization tools, machine learning models, or deep learning models.

2. Literature Review

In the "Age of Data," industries and public bodies are producing vast amounts of new data at an unprecedented rate. Organizations invest heavily in utilizing this data to create value through big data analytics. The premise is that by analyzing large volumes of unstructured data from various sources, actionable insights can transform businesses and provide a competitive edge. These data-driven insights are crucial, especially for organizations in fast-paced environments where informed decisions are vital[15]. Collecting data from multiple resources, processing it for analytical purposes, and transporting it to the destination is challenging, and data pipelines are used to manage it efficiently. Data pipelines have become a necessity for all data-driven companies[1].

Raj et al. [2] created a pipeline for analyzing datasets containing trip records of Uber, yellow, and green taxis using big data technologies such as MapReduce, Hive, and Spark. The analysis enabled us to suggest whether yellow, green, or Uber is the right choice for a rider. This system could suggest the regions to focus on for drivers depending on competitor presence and historical pickups. Mehmood and Anees[3] focused on designing distributed real-time ETL architecture for unstructured big data. They proposed an architecture using Apache Kafka, MongoDB, and Apache Spark. The method they presented and employed for experimentation can be easily applied when distributed data needs to be combined with a fast incoming unstructured stream of data in real-time. Farki and Noughabi [4] suggested a real-time blood pressure prediction method. Apache Kafka and Apache Spark were utilized to handle the large influx of incoming signals from diverse sources, encompassing wearable technology and IoT sensors. Machine learning algorithms such as K-means and Random Forest Regression are implemented using Spark MLlib to improve the precision of this model.

Leveraging big data technologies like Apache Kafka and Apache Spark simplifies the management of data pipelines. Apache Kafka streamlines the processing of vast volumes of real-time data from diverse sources, offering fault tolerance, scalability, and efficient data handling. On the other hand, Apache Spark provides a scalable and practical approach to both machine learning model development and real-time data processing tasks.[4].

With the growing volume of data, the ETL jobs of many enterprises may take hours or days to complete. This latency may cause incorrect decision-making. So, there is a need to optimize the ETL pipeline data flow as the demand for shorter time processing time for ETL processes is increasing. Various case studies have provided evidence of the efficacy

of these approaches in practical settings. For instance, a research investigation conducted by Ranjan J (2009)[10] discovered that implementing data warehousing and business intelligence tools, alongside optimized data extraction processes, yielded notable enhancements in the performance of ETL tasks within a prominent financial services organization. According to [11], parallel processing is a powerful technique that enhances the performance of ETL processes by executing multiple tasks simultaneously. It increases overall throughput, allows scalability with additional resources, reduces latency, and improves fault tolerance. Task parallelism divides large ETL jobs into smaller tasks executed through thread-based, process-based, or cluster-based parallelism. Pipeline parallelism divides jobs into stages, executed via multi-threaded, multi-process, or multi-node pipelines. Cloud-based parallel processing utilizes cloud services for distributed execution. Data partitioning divides large datasets for parallel processing. Conversely, caching stores frequently accessed data in temporary storage, reducing retrieval time source system load and improving scalability and data consistency. In-memory, disk-based, and distributed caching strategies can be used. The choice of parallel processing and caching strategies depends on data, processing time, and resource requirements. These techniques collectively optimize ETL performance and efficiency.

Blockchain is the technology behind the birth of Bitcoin and cryptocurrency. According to Bhutta et al.[5], Blockchain's key characteristics include decentralization, transparency, autonomy, security, immutability, traceability, democratization, and fault tolerance. Blockchain is a transformational technology that can provide a basis to develop distributed and secure applications for industries like finance, health care, government, manufacturing, distribution, etc. One use case described by Teogenes & Gomes[6] is using blockchain in e-voting systems. The current voting methods, electronic or not, cause an unsatisfactory level of voter confidence. Blockchain would leverage security, transparency, and immutability to increase voter confidence and strengthen democracy.

Ali Syed et al.[7], talks about the use of blockchain in the vehicle industry. BMW has implemented blockchain technology to handle its asset and logistics operations; since 2016, Toyota has invested in blockchain-based supply chain management. Furthermore, BMW, Ford, Renault, and General Motors are part of the Mobility Open Blockchain Initiative (MOBI), including IBM, Bosch, and Blockchain at Berkeley, among 30 other companies. MOBI's primary objective is to encourage the adoption of blockchain technology and establish industry-wide collaboration.

According to Monrat et al.[8], Blockchain can be used in health care to trace medicines and patient data. One of the major concerns for the healthcare industry is managing patient data integrity. Blockchain can solve data integrity problems because of its immutable and secure nature. Haderet al.[9], presented a framework that integrates blockchain and big data to enhance supply chain traceability and facilitate information sharing within the textile industry.

In conclusion, the literature review highlights the significance of data pipelines in managing and processing data efficiently in today's data-driven landscape. It explores the implementation of big data technologies, such as Apache Kafka and Apache Spark, for optimized data management. Moreover, it emphasizes the potential of blockchain technology in various industries, including e-voting, the automotive sector, healthcare, and supply chain management. The reviewed studies demonstrate the real-world efficacy of these approaches and lay the foundation for further research and innovation in data pipeline management and blockchain integration.

3. Technological Frameworks

3.1 Apache Kafka

In distributed stream processing applications, ensuring strong correctness guarantees in the face of unexpected failures and out-of-order data is crucial. This provides reliable and authoritative results without depending on complementary batch results. While existing systems tackle issues like consistency and completeness, finding the optimal balance between correctness, performance, and cost remains a practical challenge for users. Apache Kafka addresses this challenge through its core design for stream processing, leveraging its persistent log architecture for storage and communication between processors. By doing so, it achieves the desired correctness guarantees. Kafka Streams, a scalable stream processing client library within Apache Kafka, utilizes read-process-write cycles to capture state updates and outputs as log appends, providing a robust and reliable solution [13].

In the current era of big data, the primary challenge lies in collecting the vast amounts of data generated [15]. Apache Kafka, a free and open-source distributed streaming platform/messaging system, stands out for its capability to manage large volumes of incoming data streams. It is widely utilized for data extraction from diverse and heterogeneous sources, owing to its ability to ingest expanding data volumes from unstructured or semi-structured data sources. Renowned organizations like Twitter, Walmart, and others extensively use Kafka. Apache Kafka's key features, such as high throughput, scalability, fault tolerance, and reliability, make it an excellent and preferred choice for handling ATM transaction data.

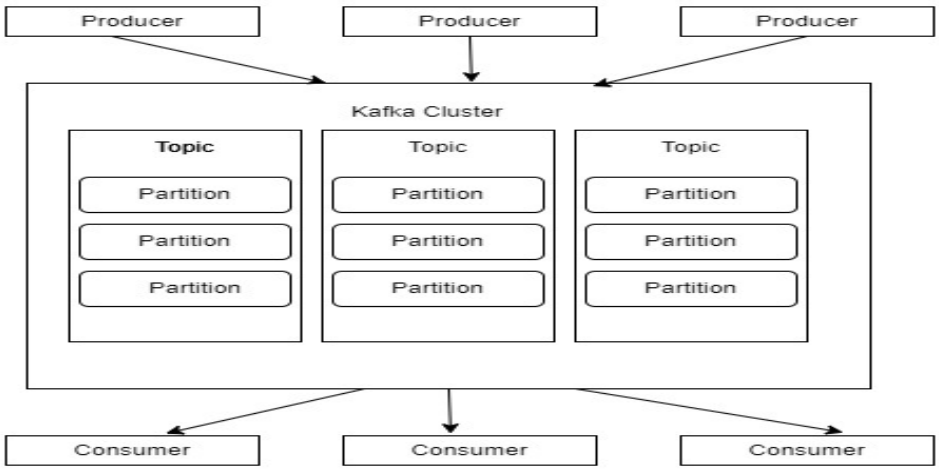


Fig. 1: A typical ANN model

Kafka consists of clusters of multiple brokers that store data assigned to different Kafka topics. A topic can have multiple partitions and be replicated across multiple brokers.

Data producers write data on different Kafka topics. The number of partitions and replication factors for a Kafka topic can be defined at the time of Kafka topic creation. A partition consists of messages in a sequence, and new messages are added at the end of the partition. Replication of topics across multiple brokers prevents data loss in case of a broker

failure. Consumers subscribe to a specific Kafka topic and fetch messages from it. A consumer may belong to a consumer group. A consumer can be a database such as HBase Cassandra or a real-time consumer such as Spark or Storm.

Numerous companies across various industries have adopted Apache Kafka as the fundamental infrastructure for data pipelines, streaming analytics, data integration, and critical applications. Data in Kafka is organized into topics, each of which can have multiple partitions. These partitions are maintained as immutable sequences of records, functioning like logs. Producers can continuously add data to partitions, while consumers can continuously read from them [16].

3.2 Apache Spark

Big data analytics, crucial in storing, processing, and analyzing massive datasets, has become indispensable [15]. With the emergence of distributed computing frameworks like Spark, efficient solutions to explore vast amounts of data are now available. Spark's popularity has surged due to its accessible application programming interface (API) and exceptional performance, surpassing the MapReduce framework. The default system parameters in Spark make it effortless for system administrators to deploy their applications and measure specific cluster performance using factory-set parameters [12].

Apache Spark, a widely adopted open-source framework, is renowned for its ability to handle extensive data processing tasks. It offers a programming interface that facilitates cluster programming with implicit data parallelism and ensures fault tolerance.

The process of training machine learning models faces challenges that cause slowdowns, such as the dataset size and the optimization parameters needed to create the best-fitting model. To address these issues, researchers have sought a more suitable approach. One potential solution is employing the Apache Spark tool, a high-speed cluster computing framework and open-source distributed programming tool for clusters. Additionally, Spark performs operations in memory, further enhancing its efficiency [17].

Spark provides Java, Scala, Python, and R APIs and an optimized engine that executes general execution graphs. Spark excels in iterative computations, making it an ideal choice for creating large-scale machine-learning applications.

In the Apache Spark architecture, when the Driver Program executes, it calls the actual application program and establishes a SparkContext containing all the fundamental functions. Alongside the SparkContext, the Spark Driver comprises other essential components such as the DAG Scheduler, Task Scheduler, Backend Scheduler, and Block Manager. These components combine to convert user-written code into jobs executed on the cluster.

The Cluster Manager is responsible for managing the execution of various jobs within the cluster. The Spark Driver works hand in hand with the Cluster Manager to oversee the execution of different jobs. The Cluster Manager allocates resources for the job, divides them into smaller tasks, and distributes them to worker nodes. The Spark Driver takes charge of controlling this execution process.

Multiple worker nodes can be employed to process an RDD created in SparkContext, and the results can also be cached for optimization. The Spark Context receives task information from the Cluster Manager and enqueues it on worker nodes. The executor manages the execution of these tasks. The lifespan of executors aligns with that of the Spark Application, and if desired, increasing the number of workers can enhance the system's performance, allowing for the division of jobs

into more manageable parts.

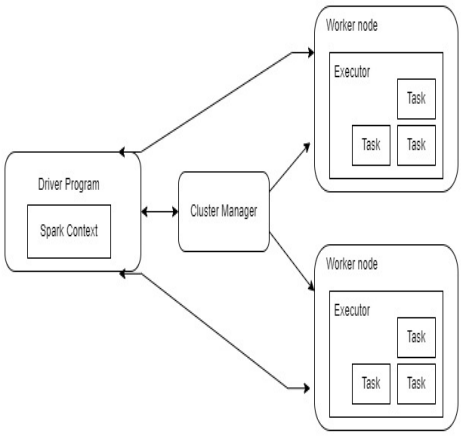


Fig. 2: Apache Spark Internal Architecture

There are several platforms and engines available for transforming the data, for example, Hadoop MapReduce, Apache Flink, and Apache Storm, but the spark is preferred for our ETL pipeline considering the following reasons:

Scalability: Spark is designed to handle extensive scale data processing, making it one of the best choices for data transformation. Distributing computation across a cluster of machines enables parallel processing and effective utilization of available resources.

Speed: Spark is renowned for its exceptional processing speed. It achieves this by conducting computations in memory, minimizing disk input/output (I/O), and considerably expediting data transformation operations. Moreover, Spark's capability to store intermediate data in memory through caching further amplifies its performance, resulting in accelerated data transformation tasks.

Fault Tolerance: Spark incorporates inherent fault tolerance mechanisms, providing a robust system for handling failures. It can automatically recover from errors, guaranteeing uninterrupted progress in the data transformation process.

Flexibility: Spark offers a wide range of programming interfaces, including Java, Scala, Python, and R. Moreover, It also boasts a comprehensive ecosystem comprising numerous libraries and extensions.

3.3 Random Walk Model

Blockchain technologies have become prominent in recent years, with many experts citing the technology's potential applications regarding different aspects of any industry, market, agency, or governmental organization. In the brief history of blockchain, many achievements have been made regarding how blockchain can be utilized and the impacts it might have on several industries.[18].

Blockchain is recognized for its decentralized, autonomous, and immutable characteristics, providing various features such as verification, fault tolerance, anonymity, auditability, and transparency [14]. Blockchain is a distributed and decentralized ledger that records transactions across a network of computers, ensuring immutability. It provides essential features such as authentication, integrity, traceability, privacy, confidentiality, and fault tolerance.

There are three main types of blockchains:

Permissionless or Public blockchains: These allow anyone to join the network and participate in managing the blockchain.

Permissioned or Private blockchains: Only invited individuals from a single organization can join the network and take part in managing the blockchain.

Consortium blockchains: Invited members from various organizations can join and participate in the consortium blockchain's management.

Here are some key aspects that contribute to blockchain's security and immutability:

1. **Decentralization:** To collectively maintain and validate the database, blockchain operates on a decentralized network of nodes. These nodes working in peer-to-peer architecture form a blockchain network. This decentralized system removes the need for a central authority, making it resistant to single points of failure and reducing the risk of unauthorized tampering or data manipulation.

2. **Distributed Ledger:** The blockchain functions as a distributed ledger, chronologically recording all transactions or data entries. Within the network, each node retains a copy of the complete blockchain, and the addition of transactions to the ledger follows a consensus mechanism like proof-of-work or proof-of-stake. This decentralized approach guarantees the existence of multiple copies of the ledger, rendering it challenging for attackers to alter the data without consensus from the majority of nodes.

3. **Cryptographic Hash Functions:** Blockchain employs cryptographic hash functions to safeguard the data's integrity. Every block within the blockchain contains a unique hash value generated based on its data content. If any alteration is made to the data within a block, it will lead to a distinct hash value, making any tampering easily detectable. This crucial characteristic guarantees that once a block is added to the blockchain, it becomes practically impossible to modify or erase the data without being noticed.

4. **Immutable Records**: Once data is added to the blockchain, it becomes virtually immutable. The decentralized and distributed nature of the blockchain, coupled with the cryptographic hash functions, ensures that historical transactions or data entries resist modification. This immutability provides high trust and transparency, as it becomes difficult to dispute or alter past records.

5. **Consensus Mechanism:** Blockchain networks rely on consensus mechanisms to agree on the validity of transactions or data entries. Consensus algorithms ensure that all nodes in the network reach an agreement on the order and validity of transactions, preventing fraudulent or conflicting entries. This consensus process strengthens the security of the blockchain by requiring a majority of nodes to validate and agree on the data being added.

6. **Encryption:** Blockchain can integrate encryption techniques to safeguard sensitive data. Encryption guarantees that the data stored on the blockchain remains confidential and can only be accessed by authorized parties possessing the correct decryption keys. Through data encryption, blockchain adds an extra layer of security, particularly for sensitive information like personal or financial data.

By combining these elements, blockchain technology provides a secure and immutable database resistant to tampering, fraud, and unauthorized access. Its decentralized nature, cryptographic principles, and consensus mechanisms create a trustless environment where participants can confidently interact and rely on the integrity and security of the stored data.

4. Proposed Solution

4.1 Data Collection and Ingestion

Apache Spark is used for data collection and ingestion. The whole process is described in this section.

4.1.1 Overview of ATM transaction data

Financial transactions conducted at ATMs provide valuable information about customer

behavior, banking patterns, and cash flow. Here is an overview of the typical information captured during ATM transactions:

Date and Time: The timestamp indicates when the transaction took place. It includes the date, time, and time zone.

Transaction Type: Specifies the nature of the transaction, such as cash withdrawal, cash deposit, funds transfer, or other services.

ATM Location: Records the physical location of the ATM, identified by an address or geographic coordinates (latitude and longitude).

Card Information: Encrypted details related to the card used for the transaction, including the card number and expiration date. Note that sensitive cardholder data like the cardholder's name, PIN, or CVV (Card Verification Value) is typically not stored in the transaction data.

Transaction Amount: Indicates the monetary value involved in the transaction.

Account Information: Identifies the bank account associated with the transaction, usually by an account number or an encrypted identifier.

Transaction Result: Specifies the transaction's outcome, whether it was successful, declined, canceled, or encountered an error.

ATM Terminal ID: A unique identifier assigned to each physical ATM terminal, distinguishing it from other ATMs within a network.

Currency: Denotes the currency in which the transaction was conducted, such as USD (United States Dollar), EUR (Euro), GBP (British Pound), etc.

Additional Messages: Records any additional message utilized during the transaction, like language selection, receipt printing, or screen customization.

The dataset used in this paper can be accessed at https://www.kaggle.com/datasets/sparnord/danish-atm-transactions.

4.1.2 Integration of Kafka for ATM Transaction Data Ingestion

We are using Kafka to collect ATM transaction data as it can handle high-throughput data streams, provide fault tolerance, and enable real-time processing. Collecting ATM transaction data from different ATMs involves setting up a Kafka infrastructure to receive, store, and process the data. Here's an explanation of the process:

ATMs are the source here. An ATM does not have its own dedicated Kafka producer. Instead, a middleware layer containing several Kafka producers is usually responsible for collecting the transaction data from multiple ATMs. This system acts as a producer and forwards the data to appropriate Kafka topics.

A Kafka topic is explicitly created for storing ATM transaction data. The middleware produces the collected ATM data for this Kafka topic. A Kafka topic is a channel where ATM transaction data is organized and published. Think of it as a virtual container or a labeled stream of data.

After the data is produced into the Kafka topic, the Kafka cluster provides the infrastructure to handle the data flow. A Kafka cluster consists of multiple Kafka brokers that form a distributed system. Brokers are individual server instances that include the distributed messaging system. Each broker is responsible for handling a portion of the data, including storing and replicating the data across the cluster. It facilitates the streaming of data in real-time. As new data arrives, it is immediately made available to consumers subscribed to the corresponding Kafka topic, enabling real-time processing, analytics, and integration with downstream systems. A

Kafka consumer refers to an application or service subscribing to a Kafka topic and consuming the transaction message published to that topic for further processing. The following figure explains the process:

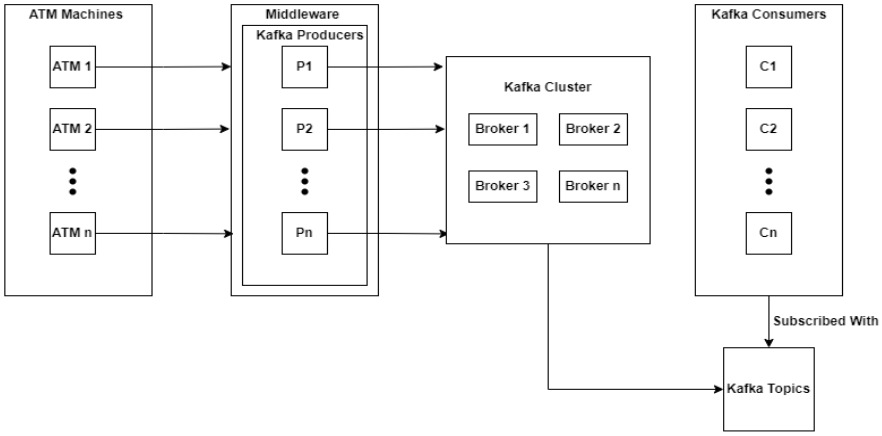


Fig. 3: Integration of Kafka for ATM transaction data ingestion

4.2 Data Transformation and Analysis

Various transformations can be applied depending on the dataset and specific use case. These transformations include Filtering, Mapping, Aggregation, Data Cleansing, and Machine Learning Transformations.

In our particular use case, the initial step involves mapping the dataset onto a specific schema, eliminating non-essential attributes. Subsequently, Data Cleansing is executed to address missing values and duplicates. Lastly, the Data Frame is aggregated based on the transaction month. The desired output comprises grouped rows of data that have been cleansed, mapped, and organized according to the month of the transactions.

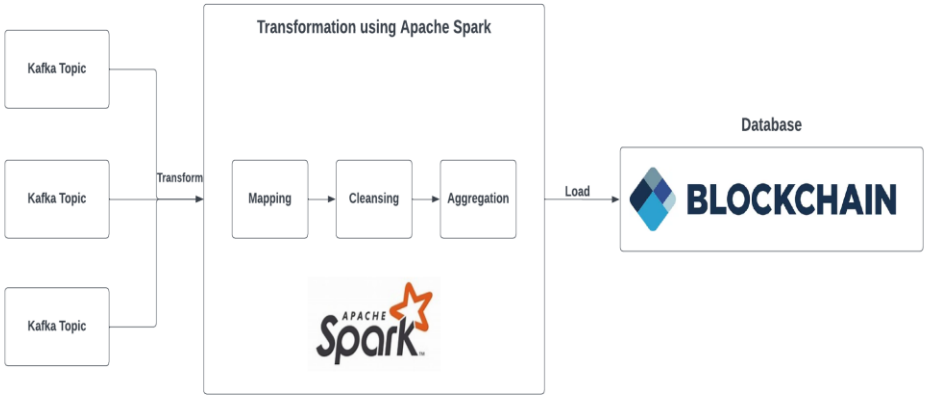


Fig. 4: Data Transforming using Apache Spark

The depicted diagram illustrates the workflow of the ETL (Extract, Transform, Load) process. Initially, the ATM Transaction Data is extracted and forwarded to Kafka topics. Following extraction, the data is transformed using Apache Spark. This transformation phase encompasses Mapping, Cleansing, and Aggregating the data frame based explicitly on the month attribute. After the transformation step, the next stage is loading, during which the transformed data is loaded into a database, depending on the ETL configuration in use.

4.3 Data Storage and Security

Implementing blockchain for ATM transaction data storage can provide security, transparency, and immutability to the transaction records. Multiple blockchain platforms can be used depending on the requirement. The most popular platforms include Ethereum, Hyperledger Fabric, and Corda. The choice of platform depends on the scalability, consensus mechanism, smart contract, and community support.

Using a platform like Ethereum, you can create smart contracts to define the rules and logic for processing transactions. Smart contracts are self-executing contracts with predefined conditions, enabling automated and trustworthy processing. Define the necessary functions and events to handle the ATM transaction data. Define the data structure through which information should be stored on the blockchain.

```
struct Transaction {
    uint256 dateAndTime;
    string transactionType;
    string atmLocation;
    CardInformation cardInfo;
    uint256 transactionAmount;
    AccountInformation accountInfo;
    string transactionResult;
    string atmTerminalID;
    string currency;
    string additionalMessages;
```

Write a smart contract that defines the functions and events to handle ATM transactions. Smart contracts are crucial in handling ATM transaction data in a blockchain-based system. Smart contracts are the system's backbone, ensuring ATM transaction data's accuracy, transparency, security, and trustworthiness in a blockchain-based environment. They provide a decentralized and automated approach to processing and storing transaction data, eliminating the need for intermediaries and enhancing the efficiency and reliability of the overall ATM transaction process.

Input: Kafka topic (string)

Processing:

```
contract KafkaDataLoader {
mapping(string => bool) private processedRecords;
event RecordLoaded(string recordId);
function loadFromKafka(string memory kafkaTopic) public {
KafkaConsumer consumer = createConsumer();
consumer.subscribe(kafkaTopic);
while (true) {
Message message = consumer.consume();
string memory recordId = extractRecordId(message);
string memory recordData = extractRecordData(message);
if (!processedRecords[recordId]) {
bool isValid = validateRecordData(recordData);
if (isValid) {
```

```
        storeOnBlockchain(recordId,
          recordData);

        processedRecords[recordId] =
              true;

        emit RecordLoaded(recordId);

    }   }   }   }   }
```

Output: Records loaded on the Ethereum blockchain, with the event RecordLoaded containing the recordId (string).

This represents a smart contract called "KafkaDataLoader" that facilitates loading data from a Kafka topic into the blockchain. The contract defines a mapping to keep track of processed records and emits an event when a record is successfully loaded. The "loadFromKafka" function creates a Kafka consumer, subscribes to the specified topic, and continuously consumes messages. It extracts the record ID and data from each message, validates the data, and stores it on the blockchain if it is deemed valid. The contract ensures that each record is processed only once by checking the mapping. Overall, this smart contract enables the integration of Kafka data with the blockchain, providing transparency, immutability, and audibility to the loaded data.

In the next step, we will submit transactions to the blockchain. Ethereum SDK, such as Web3.js, connects with the deployed contract. The "storeOnBlockchain" function is responsible for submitting the transactions to the Ethereum blockchain. Monitor the Ethereum network to ensure the transactions are successfully added to the blockchain. Listen for the NewTransaction event emitted by the smart contract to track new transactions.

4.4 Overall Architecture of Pipeline

Using Kafka to collect ATM transaction data enables high-throughput data streams, fault tolerance, and real-time processing. ATM data is collected by a middleware layer with Kafka producers, forwarded to dedicated Kafka topics, and consumed by applications for processing and analytics. ETL process applies filtering, mapping, aggregation, and data cleansing. Use cases involve mapping to a schema, data cleansing for missing values and duplicates, and aggregation.

Based on transaction months. Blockchain implementation for ATM transaction data offers security, transparency, and immutability. Smart contracts play a vital role, defining functions and events to process and store transaction data, ensuring accuracy, trustworthiness, and decentralization.

The following diagram describes the architecture of the pipeline proposed in this paper:

Fig. 5: Overall Architecture of Pipeline

5. Results and Discussion

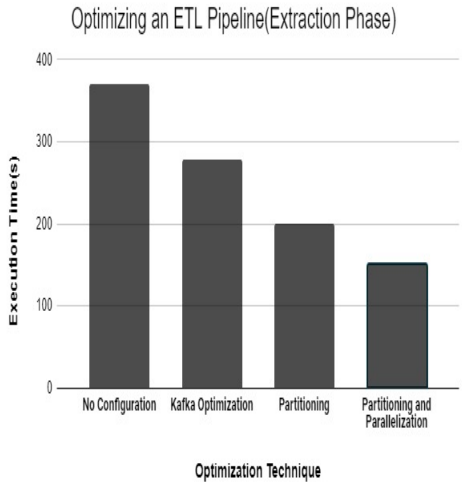
5.1 Analysis of ETL Pipeline Performance

The following section presents the performance evaluation of the ETL Pipeline. The experiment aimed to optimize the performance of the ETL Pipeline while considering the security factor.

Firstly, the goal was to improve the performance of the extraction phase. This phase includes sending the 24M transaction records data to Kafka topics. With the default configuration of Kafka, topics with one partition, and No Optimization method, the execution time is almost 6 minutes. Now, the goal is to optimize the methods involved so that the execution time can be reduced.

The first optimization method undertaken was to change the default configuration for Kafka. Here are the changes made to the Kafka configuration:

- Acks was set to 1, meaning the producer will only wait for the acknowledgment from the leader broker.
- Batch size was set to 32kb so that more data to a broker can be sent at a time.



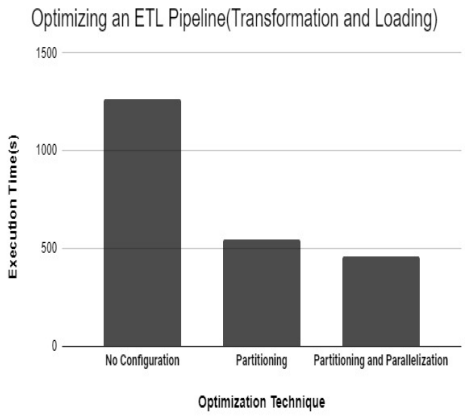
- Linger. ms, the delay before sending a new data packet was set to 5ms so that producers have enough time to store data in the batch before sending.
- Topics are created with three partitions rather than one by default.

After all these configuration changes, execution time was recorded to be almost 4.6 minutes. Then, the most critical optimization step suggested in [2] is the process of Partitioning and Parallelization. To achieve the following, the data source was partitioned into multiple sources, and multiple producers sent the partitioned data to Kafka topics. But only Partitioning wasn't enough for a good result as the execution time was decreased to almost 3.5 minutes. The last method to optimize this phase was Parallelization, which is achieved using multi- processing. Processes that send data to Kafka-topics were run in parallel, and the whole data was sent to Kafka-topics in around 2.5 minutes, which appears suitable for sending 24 million rows of data.

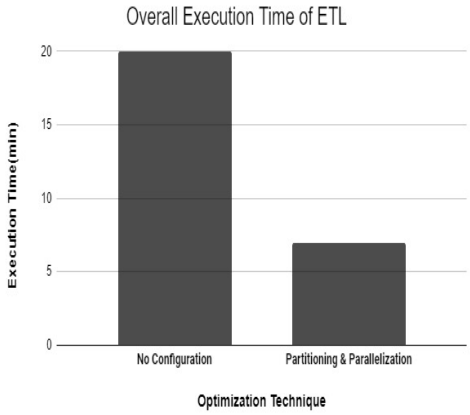
The next phase is the transformation and loading phase. This phase includes reading from Kafka topics, mapping, cleaning, aggregating, and loading the transformed data into a database. The non-optimized transformation phase took around 20 minutes, which is unacceptable.

For better performance, the main data frame containing the whole data was partitioned into multiple sub-data frames. Now, transformation methods like filtering based on month cleaning were applied on sub-data frames instead of the main data frame, resulting in a sufficient decrease in execution time. Each sub-data frame was executed in parallel to optimize the complete processes.

This results in a final execution time of almost 7 minutes.



Now, to compare how much optimization we have achieved, the overall execution time of the whole ETL Process of non-optimized and optimized ETL Pipeline is shown in the figure:



5.2 Data Security Evaluation Using Blockchain

Data security is critical when utilizing blockchain technology to store ATM transaction data. Here's an evaluation of the security measures mentioned in the provided information:

1. **Access Control:** Implementing roles and permissions helps ensure that only authorized individuals can perform specific actions. By defining roles for ATM operators, bank administrators, and auditors, access to sensitive operations can be restricted. Using modifiers and require statements in the smart contract enforces these permissions effectively.

2. **Encryption:** Encrypting sensitive data fields before storing them on the blockchain adds an extra layer of protection. Utilizing encryption algorithms like AES or RSA helps ensure the confidentiality of card information, account details, and transaction amounts. Encrypting the data before storing it on the blockchain makes it more difficult for unauthorized parties to access sensitive information.

3. **Event Logging:** Emitting events for critical operations and storing event logs off-chain in a secure and centralized logging system aids in auditing and analysis. By keeping track of successful transactions, account balance updates, and access control changes, it becomes easier to monitor and review activities for any potential security breaches.

4. **Secure Data Handling:** Avoiding the storage of sensitive information such as card PINs or CVV numbers on the blockchain is crucial. These details should be handled securely outside the blockchain, with the blockchain storing only the necessary transaction metadata. Adhering to secure coding practices helps mitigate common vulnerabilities and ensures the integrity of the stored data.

5. **Code Auditing and Testing:** Conducting thorough code reviews and security audits is essential to identify and address potential vulnerabilities. By testing the smart contract extensively,

including simulated attack scenarios, weaknesses can be discovered and rectified. Regular audits and testing help maintain a robust and secure smart contract for handling ATM transaction data.

6. **Contract Upgradability:** Implementing a mechanism for contract upgradability is essential for addressing security patches or adding enhancements. However, caution must be exercised to ensure that contract upgrades do not compromise the integrity of stored transaction data or introduce new security risks. Best practices should be followed to maintain the security of the data during the upgrade process.

7. **External Dependency Security:** Using verified and audited libraries for cryptographic operations and other external dependencies is crucial. Relying on trusted code reduces the risk of introducing vulnerabilities or compromising data security. Care should be taken to thoroughly evaluate and vet any external dependencies in the blockchain solution.

8. **Regular Updates and Patching:** Staying informed about security updates and patches for the blockchain platform and smart contract frameworks is necessary. Promptly applying these updates ensures that known vulnerabilities are addressed, and the latest security measures are in place. Regular updates and patching are vital for maintaining a secure environment for ATM transaction data.

9. **Third-Party Audits:** Engaging independent security auditors helps ensure a comprehensive evaluation of the smart contract handling ATM transaction data. Third-party auditors can identify potential security weaknesses, validate the effectiveness of implemented security measures, and provide recommendations for improvement. These audits offer an objective perspective on the security of the blockchain solution.

By implementing these security measures and regularly evaluating the data security aspects, ATM transaction data stored on the blockchain can benefit from increased integrity and confidentiality.

5.3 Data Security Evaluation Using Blockchain

The following table highlights the differences between blockchain and traditional databases:

	Block Chain	**Traditional Databases**
Immutable and Tamper-Resistant	Blockchain databases store data in linked blocks with cryptographic hashes, ensuring highly secure and tamper-resistant storage.	Traditional databases may rely on centralized servers or administrators, which can be vulnerable to unauthorized modifications.
Decentralization	Blockchain databases' decentralized nature, distributing copies across a node network, mitigates single points of failure and centralized attack targets.	On the other hand, traditional databases often have a central server, which can be a potential weakness.

Consensus Mechanisms	Consensus mechanisms like proof-of-work or proof-of-stake in blockchain databases validate and require agreement among most network participants before adding transactions to the chain.	In traditional databases, transactions are often validated and controlled by a central authority, which may be more susceptible to corruption or hacking attempts.
Encryption and Security Measures	Due to their distributed ledger nature, blockchain databases employ robust encryption for securing transactions and identities.	Traditional databases can implement encryption techniques to protect data.
Privacy and Confidentiality	While blockchain offers transparency and immutability, it introduces privacy concerns as certain data might be visible to all participants based on the design.	Traditional databases, on the other hand, can implement access controls and encryption to restrict access to sensitive information.
Performance and Scalability	Blockchain databases may need help with scalability and performance due to slower transactions and limited scalability tied to consensus mechanisms and distributed structure.	Traditional databases, especially those designed for high-performance environments, can often handle larger volumes of data and provide faster response times.

6. Conclusion

ETL processes are known to be complex and resource-consuming. With the advancement of financial technology, the security and integrity of customer data have emerged as a big concern. So, there is a definite need to focus on the security aspects of a pipeline. However, the enhancement of pipeline security comes with a compromise on pipeline performance. To achieve a better level of security, we need to increase pipeline performance to maintain an optimum balance between security and performance.

Blockchain is known for its secure nature. In this paper, we propose that the ATM transaction data should be stored in a blockchain in the load phase of the ETL pipeline. This ensures the security and integrity of crucial financial data as Blockchain provides immutability, transparency, traceability, decentralization, reliability, privacy, confidentiality, and fault tolerance. Blockchain offers encryption/decryption, cryptography, digital signature and timestamp, and hash trees to ensure security. Compromising on security is synonymous with compromising on customer trust. However, blockchain has the disadvantage of low throughput, making the pipeline slow. So, we tried to optimize the pipeline performance using parallelization and partitioning in the extraction phase. Before optimization, for 24 million transactions, it took 6 min to send data to Kafka topics. After optimization, it took 2.5 minutes. In this way, we enhanced the pipeline security without affecting performance.

REFERENCES

[1] A. Raj, J. Bosch, H. H. Olsson, and T. J. Wang, “Modelling Data Pipelines,” 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Portoroz, Slovenia, 2020, pp. 13-20, doi: 10.1109/SEAA51224.2020.00014.

[2] H. Sun, S. Hu, S. McIntosh, and Y. Cao, "Big data trip classification on the New York City taxi and Uber sensor network," Journal of internet Technology, vol. 19, no. 2, pp. 591–598, 2018.

[3] Mehmood, E., Anees, T. Distributed real-time ETL architecture for unstructured big data. Knowl Inf Syst 64, 3419–3445 (2022). https://doi.org/10.1007/s10115-022-01757-7

[4] A. Farki and E. A. Noughabi, "Real-Time Blood Pressure Prediction Using Apache Spark and Kafka Machine Learning," 2023 9th International Conference on Web Research (ICWR), Tehran, Iran, Islamic Republic of, 2023, pp. 161-166, doi: 10.1109/ICWR57742.2023.10138962.

[5] M. N. M. Bhutta et al., "A Survey on Blockchain Technology: Evolution, Architecture and Security," in IEEE Access, vol. 9, pp. 61048-61073, 2021, doi: 10.1109/ACCESS.2021.3072849.

[6] Moura, Teogenes & Gomes, Alexandre. (2017). Blockchain Voting and its Effects on Election Transparency and Voter Confidence. 574-575. 10.1145/3085228.3085263.

[7] T. Ali Syed, A. Alzahrani, S. Jan, M. S. Siddiqui, A. Nadeem and T. Alghamdi, "A Comparative Analysis of Blockchain Architecture and its Applications: Problems and Recommendations," in IEEE Access, vol. 7, pp. 176838-176869, 2019, doi: 10.1109/ACCESS.2019.2957660.

[8] A. A. Monrat, O. Schelén, and K. Andersson, "A Survey of Blockchain From the Perspectives of Applications, Challenges, and Opportunities," in IEEE Access, vol. 7, pp. 117134-117151, 2019, doi: 10.1109/ACCESS.2019.2936094.

[9] Hader, Manal & Tchoffa, David & El Mhamedi, Abderrahman & Ghodous, P. & Dolgui, Alexandre & Abouabdellah, Abdellah. (2022). Applying integrated Blockchain and big data technologies to improve supply chain traceability and information sharing in the textile sector. Journal of Industrial Information Integration. 28. 100345. 10.1016/j.jii.2022.100345.

[10] Jayanthi Ranjan, "Business Intelligence: Concepts, Components, Techniques and Benefits," Journal of Theoretical and Applied Information Technology, vol. 9, no. 1, pp. 60-70, 2009.

[11] Dhamotharan Seenivasan (2023). We are improving the Performance of the ETL Jobs. 71(3), pp.27–33. Doi https://doi.org/10.14445/22312803/ijctt-v71i3p105.

[12] Ahmed, N., Barczak, A.L.C., Susnjak, T. et al. A comprehensive performance analysis of Apache Hadoop and Apache Spark for large scale data sets using HiBench. J Big Data 7, 110 (2020). https://doi.org/10.1186/s40537-020-00388-5

[13] Wang, G., Chen, L., Dikshit, A., Gustafson, J., Chen, B., Sax, M.J., Roesler, J., Blee-Goldman, S., Cadonna, B., Mehta, A., Madan, V. and Rao, J. (2021). Consistency and Completeness. Proceedings of the 2021 International Conference on Management of Data. doi:https://doi.org/10.1145/3448016.3457556.

[14] Guo, H. and Yu, X. (2022). A Survey on Blockchain Technology and its security. Blockchain: Research and Applications, 3(2), p.100067. doi https://doi.org/10.1016/j.bcra.2022.100067.

[15] Mikalef, P., Boura, M., Lekakos, G. and Krogstie, J. (2019). Big data analytics and firm performance: Findings from a mixed-method approach. Journal of Business Research, [online] 98(2), pp.261–276. Doi https://doi.org/10.1016/j.jbusres.2019.01.044.

[16] Wang, G., Chen, L., Dikshit, A., Gustafson, J., Chen, B., Sax, M.J., Roesler, J., Blee-Goldman, S., Cadonna, B., Mehta, A., Madan, V. and Rao, J. (2021). Consistency and Completeness. Proceedings of the 2021 International Conference on Management of Data. doi:https://doi.org/10.1145/3448016.3457556.

[17] Haggag, M., Tantawy, M.M. and El-Soudani, M.M.S. (2020). Implementing a Deep Learning Model for Intrusion Detection on Apache Spark Platform. IEEE Access, 8, pp.163660–163672. doi:https://doi.org/10.1109/access.2020.3019931.

[18] Berdik, D., Otoum, S., Schmidt, N., Porter, D. and Jararweh, Y. (2021). A Survey on Blockchain for Information Systems Management and Security. Information Processing & Management, 58(1), p.102397. doi:https://doi.org/10.1016/j.ipm.2020.102397.