IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 28 August 2024, accepted 10 October 2024, date of publication 14 October 2024, date of current version 1 November 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3479736

RESEARCH ARTICLE

Data Ingestions as a Service (DIaaS): A Unified Interface for Heterogeneous Data Ingestion, Transformation, and Metadata Management for Data Lake

SREEPATHY H. V.[1], DINESH RAO B.[1], MOHAN KUMAR JAYSUBRAMANIAN[2], AND DEEPAK RAO B.[1]

[1]Manipal School of Information Sciences, Manipal Academy of Higher Education, Manipal 576104, India
[2]Frameworks and Tools, MulticoreWare, Coimbatore 641049, India

Corresponding author: Dinesh Rao B. (dinesh.rao@manipal.edu)

ABSTRACT Data ingestion tools are critical component of Data Lake. Existing data ingestion tools face challenges of handling large variety, formats, sources of data. There exists void for unified data ingestion interface to handle the above research problems. This study proposes an innovative and integrated framework for data ingestion in a data lake, addressing the challenges posed by heterogeneous data sources, formats, and metadata management. The framework comprises three novel modules: First Unified Data Integration Connectors (UDIC), which provide seamless connectivity and data retrieval capabilities from diverse sources including databases, data warehouses, file systems, cloud storage, and APIs; Second, Adaptive Data Variety Transformation (ADVT), a module that intelligently handles the transformation and processing of structured, semi-structured, and unstructured data types, ensuring efficient ingestion into the data lake; and third, Intelligent Metadata Management (IMM), a module that captures, stores, and manages metadata associated with the ingested data, offering advanced search, discovery, and enrichment functionalities. Comparative study corroborates features offered by the service with existing data ingestion tools to evaluate the novelty and significance of the study. Performance validation shows varying ingestion latencies across different data types: approximately 148.1 microseconds per record for structured data, 234.2 microseconds per record for semi-structured data, 65.6 microseconds per kilobyte (KB) for video data, and 42.7 microseconds per KB for image data. These results underscore the importance of considering data structure and size in optimizing ingestion processes. Overall, this research aims to revolutionize data ingestion in data lake environments by providing a unified solution for handling diverse data sources, formats, and metadata management.

INDEX TERMS Data ingestion, data lake, data sources, data formats, unified interface, data ingestion service, big data.

I. INTRODUCTION AND BACKGROUND

Data Lake is a big data architecture for storing structured, semi-structured and un-structured data at any scale and lower cost. The data lake is becoming a common way to coordinate and develop the next generation of systems to tackle new Big data management problems. As a result, organization are adopting data lake as a storage solution. Data lake serves as a central repository to store varieties of data from different sources in its raw format and it is characterized by Metadata management, Data governance, User accessibility and Scalability. Organization can use these data to find potential insights and business value for their growth and to outperform their competitors. The ultimate aim of data lakes is to drive business results and gain value from data aggregated in lake. Data lake was implemented to resolve big

The associate editor coordinating the review of this manuscript and approving it for publication was Loris Belcastro.

data problems, specifically those arising from the variety of data. A data lake is an incredibly large storage management and analysis facility that manages all data formats. As these data marts and data warehouse are limited to answer only subset of business questions, Dixon introduced the idea of data lake as a solution to the later technologies. Dixon considers the lake to be a humongous storage facility for raw data from diverse origins, enables users to discover, collect and analyze information. Figure 1. provides an overview of Data Lake architecture.

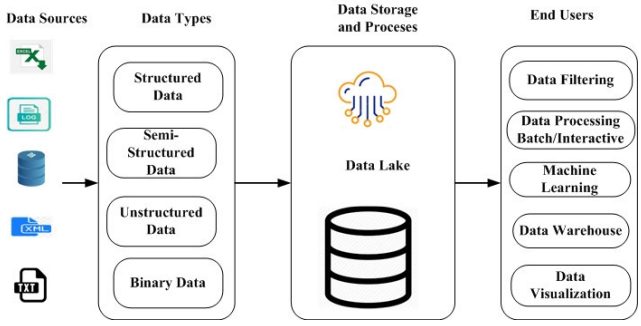

FIGURE 1. Overview of Data Lake Architecture.

Data Lake is a facility provisioned by the organization to serve as a single repository for all of its data. An organization's data is distributed and generated across various departments, business divisions, and sources. Data lakes play a critical role in modern data management by providing a centralized repository for storing vast amounts of raw data in its native format. This approach allows organizations to consolidate structured, semi-structured, and unstructured data from diverse sources into a single, scalable platform. Data lakes enable organizations to meet various analytical needs, from business intelligence and machine learning to real-time analytics, thereby enhancing decision-making processes and fostering innovation through advanced data analysis.

The ingestion of data is a critical step in bringing together an organization's disparate data sources into a Data Lake. Data ingestion is a primary and crucial step in Data Lake design process. Data intake into Data Lake has become exceedingly challenging as the volume, variety, and veracity of data collected from disparate sources has grown tremendously in recent years. Since a Data Lake stores information from numerous systems, including but not limited to databases, data warehouses, file stores, Internet of Things (IoT) gadgets, sensors, websites, embedded devices, healthcare equipment, and software, etc. Managing a plethora of data sources is a further issue for data ingestion platform. Because there is so much data, and because of its high quality and reliability, metadata is utilized to managing and cataloguing the data stored in the raw/landing zone of the Data Lake architecture. The data object's trend or pattern cannot be inferred from the data catalogue. Using data transformations at a later stage to organize data depending on its domain or context is not reliable or efficient. For creating data transformation and integration pipelines, there are a variety of big data ingestion technologies on the market.

However, the ingestion of data into data lakes presents significant challenges. The growing volume, variety, and veracity of data from disparate sources—such as databases, data warehouses, IoT devices, sensors, healthcare equipment, and more—make data ingestion a complex process. Effective data ingestion systems must manage diverse data formats, balance real-time and batch processing, and scale to accommodate increasing data volumes. Furthermore, robust metadata management is essential for ensuring governance, quality control, and traceability within the data lake. Current data ingestion technologies often handle only specific data varieties and processing types, leading to a fragmented approach that fails to address the comprehensive needs of modern data lake environments. There is a requirement for a data ingestion service to manage various variety and data processing types since each of those data ingestion technologies only handles certain variety and data processing kinds and data management at the ingestion phase. Major contribution of the proposed work are as follows:

- Design a unified framework as a service to ingest heterogeneous sources, variety and formats of data into Data Lake.
- Design an intelligent metadata module to extract metadata, patterns and trends and also to organize data for easy discovery.
- Evaluate the study by comparing features of the system with existing data ingestion tools and technologies.
- Validate the performance of the proposed data ingestion service by measuring the performance metrics gauge the service's speed and efficiency in processing data from diverse sources, varieties and protocols.

Paper is organized as follows. Section II lists and reviews related data ingestion services. Research Methodology and System architecture design is highlighted in Section III. Design Implementation is dissected in Section IV. Results and discussion of comparative study, performance validation are corroborated in Section V. Section VI discuss the conclusion of the work.

II. LITERATURE REVIEW

Data is collected from various data sources and ingested into a big data lake through a data ingestion system. The framework establishes secure connections to various sources, logs any changes, and duplicates those changes in the data lake. The data ingestion methodology makes the Data Lake a single station of enterprise data by maintaining consistency with data changes at the source systems. Two elements make up a typical ingestion framework: a data collector and a data integrator. The data integrator component handles integrating the data into the data lake, whereas the data collector is in charge of gathering or extracting the data

from a data source. According to the big data technology stack, the implementation and design of the data collector and integrator components can be variable [1]. Lakes serve as a single source of storing heterogeneous sources of data for organization. This in-turn leads to challenges in terms data ingestion, discovery and metadata management. There is a need for an efficient system that can handle data from multiple sources, look after data discovery and effectively manage metadata of the system [2]. Challenges in design of data ingestion varies from case-by-case, so it needs proper attention, budget and planning [3]. Study aims to review existing tools that can handle different data sources, variety and processing paradigms (Batch and Stream Processing). Following are few existing tools that can handle variety of data formats, including relational, non-relational databases. Sqoop is a popular Hadoop ecosystem tool for ingesting data from relational databases like Oracle, MySQL etc. into Hadoop distributed files system (hdfs). It uses java connectors to connect to databases and utilizes MapReduce programming paradigm for handling data processes [4]. As it supports only structured data fails to handle unstructured and semi-structured data. Flume is an Apache project that gathers, combines, and sends huge amounts of log data from many different sources to a data centre in a way that is spread, available, and reliable. Flume source gets events from any source outside of it, like a Web server or a database. Events sent by outside sources are in a style that the target sources can understand. Flume is an ingestion tools addressed to handle semi and un-structured data from heterogeneous data sources [3], [4], [5]. Apache Nifi is a tools for handling structured and NoSQL data. It presents an interface consisting of processors for running different jobs. Nifi processors allow to execute SQL queries to process data and also have support for bulk data ingestion. It offers processors for specific databases like Apache HBase, Cassandra, CouchDB and MongoDB to support for NoSQL databases related operations and ingestions. On the whole Nifi is a feature rich tool for handling several data formats and varieties of data [3], [4], [6], [7]. The study reviews tools to handle data from different types of sources, such as databases, file systems, APIs, Internet of Things (IoT) devices, social media sites, and log files. Sqoop is Hadoop ecosystem tool to process only from relational databases sources. In contrast Nifi, can handle wide variety of data sources including relational, non-relational databases, local file systems, and messaging systems etc. Lastly, Flume is designed to process especially log data from different sources like servers, applications, services etc. Flume can be configured to adapt and connect to wide ranges of sources/systems. Together these three tolls can be utilized to address data collection from several different sources, which may need transformation of data from one formats to others [3], [4], [5], [6], [7]. There exist few tools to handle different data processing paradigms like batch and streaming processing. Batch processing is a paradigm for processing large volumes of data at specific time instances, whereas later is to process data in real time. Apache Kafka is a service for processing data streams using ''publish-subscribe'' model. Producers generate data to specific topics, consumers need to subscribe to those topics for data consumption. Distributed architecture of Kafka makes its highly scalable and also helps to process high throughput data streams. Kafka architecture allows to have different brokers in a cluster, which helps to achieve high available and fault tolerant. It has Application Programming Interfaces(API's) and connectors to several data sources makes it an ideal for real time stream processing. Kafka also integrates well with other stream processing tools like Apache Flink and Spark [3], [4], [8]. Apache Nifi is an alternative tool for both batch and real time processing. As its offers several processors to support batch and real time data ingestion processes. Apache Flume and Sqoop are the other options that mainly focus on batch processing. Flume can be configured to collect application and error/access logs from different sources with ease. Whereas Sqoop is to move relational data from data warehouses and databases in to Hadoop ecosystem only. Flume and Sqoop are good choice to handle batch data processing, as they lack features/capability offered by former tools as mentioned above [3]. Data Torrent, Wavefront and Gobblin are few more alternative ingestion tools to address different data variety, sources, formats and velocity. Data Torrent is geared towards high-speed real-time streaming data ingestion, Gobblin focuses on batch and streaming ingestion from structured sources, and Wavefront specializes in monitoring and analytics data ingestion. The choice of tool depends on the specific requirements of the use case and the nature of the data being ingested [3]. The following Table 1. summarizes all the tools presented using various criteria to show their functionality and features

TABLE 1. Comparison of data ingestion services [1].

Criteria	Sqoop	Flume	NIFI	Kafka
Primary written in	Java	Java	Java	Java
License	Open Source	Open Source	Open Source	Open Source
Basic nature	Works well with any RDBMS that has JDBC like Oracle	Works well for streaming data sources which continuously generating	Works well for data flow creation between different systems	Works well for messaging streaming data
Type of Data	Batch Data	Stream Data	Batch and Stream	Stream
Type of loading	Not-event driven	Event driven	Both (Event and not-event)	Event driven
Architecture	Connector based	Agent based	Flow based	Process topology
User Interface	Shell command line	Shell command line	Web user interface	Shell command line

A Data Lake loads unprocessed data during the ingestion stage. A Data Lake is scarcely usable without any additional knowledge or insights, i.e., metadata, as the structure and semantics of the data are unknown; this could potentially

turn the data lake into a "data swamp." Consequently, it is essential to extract as much metadata from the data as feasible [9]. There is a lack of data ingestion tools or technologies that can extract metadata, organize data based on context or domain, and also collect descriptive statistics of data to gain insights, patterns or trends in data, just at the time of import, with the exception of Nifi which performs a small number of data transformation pipelines. The review of data ingestion tools and technologies reveals that there is a wide range of options available, each specializing in handling specific data sources, variety, velocity, and protocols. While tools like NiFi, Flume, Sqoop, and Kafka excel in certain areas, they may not provide a comprehensive solution that covers all aspects of data ingestion for a Data Lake. While tools like NiFi, Flume, Sqoop, and Kafka excel in specific areas, they do not offer a comprehensive solution that covers all aspects of data ingestion for a data lake. Each of these tools has inherent limitations that expose a significant research gap.

- Sqoop, primarily designed for structured data within the Hadoop ecosystem, struggles with non-relational and unstructured data. Additionally, its reliance on MapReduce introduces latency, making it inefficient for real-time scenarios.
- Flume, optimized for real-time ingestion of log data, lacks robust support for batch processing and structured data. Its need for specific agents for different data sources complicates configuration, further limiting its versatility.
- Apache NiFi, although versatile in handling a wide range of data sources and formats, can suffer from performance bottlenecks due to its reliance on processors, particularly in high-throughput environments.
- Kafka, while excellent for real-time stream processing, lacks native support for batch processing, limiting its applicability in hybrid data environments.
- Other tools, like DataTorrent and Gobblin, while offering enhanced scalability and integration capabilities, still fall short in managing the diverse data sources and processing paradigms required by modern data lakes.
- Additionally, none of these tools effectively addresses the extraction and management of metadata during ingestion, which is crucial for data governance and operational efficiency.

These limitations underscore the need for a unified data ingestion framework that can handle the full spectrum of data sources and processing paradigms while also integrating efficient metadata management. Such a solution would provide a cohesive and comprehensive approach to data ingestion, eliminating the need for organizations to rely on multiple tools and technologies. It would take care of the issues posed by the diversity and sources of the data, ensuring the smooth intake of both structured and unstructured data. Additionally, it would enable enterprises to quickly ingest data because it would offer batch and real-time streaming processing. By filling this void with a unified data ingestion service, organizations can enhance the efficiency and effectiveness of their Data Lake operations, enabling more streamlined and comprehensive data ingestion processes.

This research identifies a significant gap in the current landscape of data ingestion tools. There is a pressing need for a unified data ingestion framework that offers a comprehensive interface capable of managing a wide range of data sources, formats, and processing paradigms. Existing tools lack a cohesive solution that integrates diverse data types and processing approaches while efficiently handling metadata extraction during ingestion. To address this void, our research aims to develop a framework that not only unifies data ingestion across various sources and formats but also incorporates efficient metadata management, providing a robust solution for modern data lake environments.

III. DESIGN OF DATA INGESTION AS A SERVICE

An ingestion service for Data Lake is the focus of the proposed work. Based on the study's intent, it can be classified as Applied Research type because it aims to provide a unified software service operating on top of Data Lake to ingest different kinds of data from different kinds of heterogeneous sources. Since this study uses a comparative approach to assessing a data ingestion service against existing big data ingestion tools or technologies, it falls within the area of Analytical research type based on its method. The proposed work utilizes a case study type of qualitative research approach to examine and highlight problems with current big data technology. Data ingestion as a service for Data Lake is theoretically verified using a positivist research paradigm.The study proposes a modularized software architecture approach to fulfil the requirements of a unified data ingestion system. Figure 2. provides a architectural block diagram of Data Ingestion service.

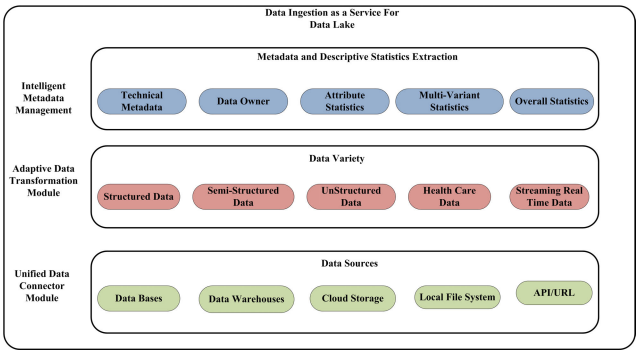

FIGURE 2. Data Ingestion as a Service for Data Lake - Architectural Block Diagram.

A. DISSECTING THE COMPONENTS OF DATA INGESTION AS A SERVICE (DIAAS)

According to a review of the literature, there is a knowledge/application void when it comes to unified data ingestion solutions that can manage heterogeneous sources, variety, and data protocol. In order to address the research, a software

as a service utility model is proposed. A Unified Approach for Heterogeneous Data Integration, Transformation, and Metadata Management is proposed in the study as a novel architectural framework for designing an integrative data lake ingestion framework. Data Ingestion as a Service(DIaaS) for Data Lake The data ingestion service plays a crucial role in collecting and integrating data from diverse sources into a data lake. To address the variety of data and effectively manage metadata, the data ingestion service is designed with the following components. Figure 3. Provides a system architecture overview

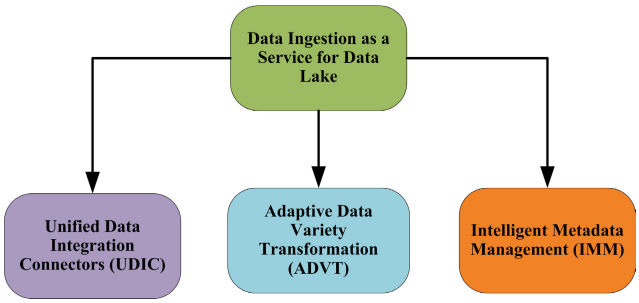

FIGURE 3. System architecture of data ingestion service.

Unified Data Integration connectors are essential component of service helps in connecting to wide range of data sources. Each sources require different connectors to establish connection with the heterogeneous sources. The Adaptive Data Variety Module is pivotal in managing the diversity and speed of data within the data ingestion service. This module is designed to handle a wide range of data types and formats, including structured, semi-structured, and unstructured data. It provides data variety parsers to parse and transform different varieties of data. It includes specialized parsers for formats such as CSV, JSON, XML, Avro, and Parquet, medical images, text and videos. Also, component is tailored for real-time and streaming data from sources like IoT sensors, social media platforms, log files, and geo-spatial data. To address the velocity aspect of data, the module incorporates streaming components capable of processing real-time data flows efficiently. This adaptive module not only transforms and validates data from diverse sources but also ensures that it is processed in real time, accommodating both the variety and velocity of modern data streams. It provides the necessary extractors and processing capabilities to effectively manage and prepare data for ingestion into the Data Lake, supporting a comprehensive and scalable data ingestion framework. Finally, Intelligent metadata management module house metadata extractors to extract technical, business, operational, administrative, semantics metadata at the time of data ingestion. Figure 4. represents the architectural block diagram provides an overview of batch ingestion of various sources and varieties of data.

Figure 5. is the Flow chart diagram representing complete workflow of data ingestion service for handling batch

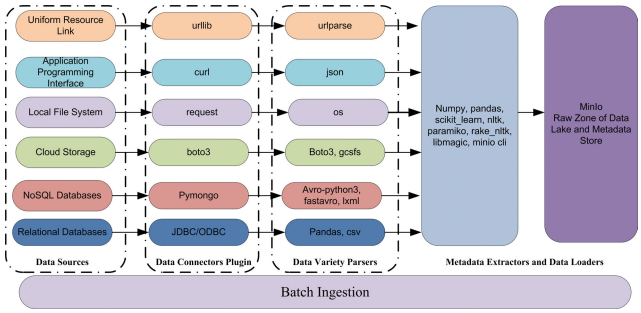

FIGURE 4. Architectural block diagram of batch ingestion of service.

and streaming ingestion of data into data lake. Following subsections elaborate details of each components of the service.

1) UNIFIED DATA INTEGRATION CONNECTORS (UDIC)

The connectors component serves as an interface between various kinds of data sources and the data ingestion service. It has connectors for APIs, file systems, cloud storage, databases, and data warehouses. Each connector is in charge of creating connections, establishing authentication with the relevant data source, and obtaining data. Each data source's individual data access protocols and file formats should be supported by these connections, allowing for seamless system integration.Proposed service offers a wide variety of plugins/connectors that allow users to establish connections to the wide assortment of data sources that collectively constitute the huge bulk of all data in big data systems.

a.Relational Databases: Supports cloud managed/hosted or on-premises relational databases by providing different JDBC/ODBC connectors for MySQL, PostgreSQL, SQL etc. to input table or tables of particular database by providing required database connection strings and credentials.

b. NoSQL Databases: Interface connector to document databases or key-value databases like MongoDB to ingest collections from particular database into Data Lake. Interface offers python libraries to connect to different nosql databases.

c. Data Warehouses: Data warehouses have been a predecessor to data lakes and continue to play a crucial role in data management and analytics. They are designed to store structured and organized data from various sources to support reporting, analysis, and decision-making processes within an organization. While data lakes and cloud-based data warehouses have emerged as newer technologies, data warehouses still hold significance in many industries. Module offers interfaces to cloud data warehouses like Amazon Red shift and GCP BigQuery etc.

d. Local File System: Connector allows to upload varieties of data stored in local file system in to Data Lake.

e. Web URL: Data from websites and URL in different formats can be ingested using this plugin.

f. Cloud Storages: Interface for collecting data from storages offered by Google, AWS and other cloud providers.

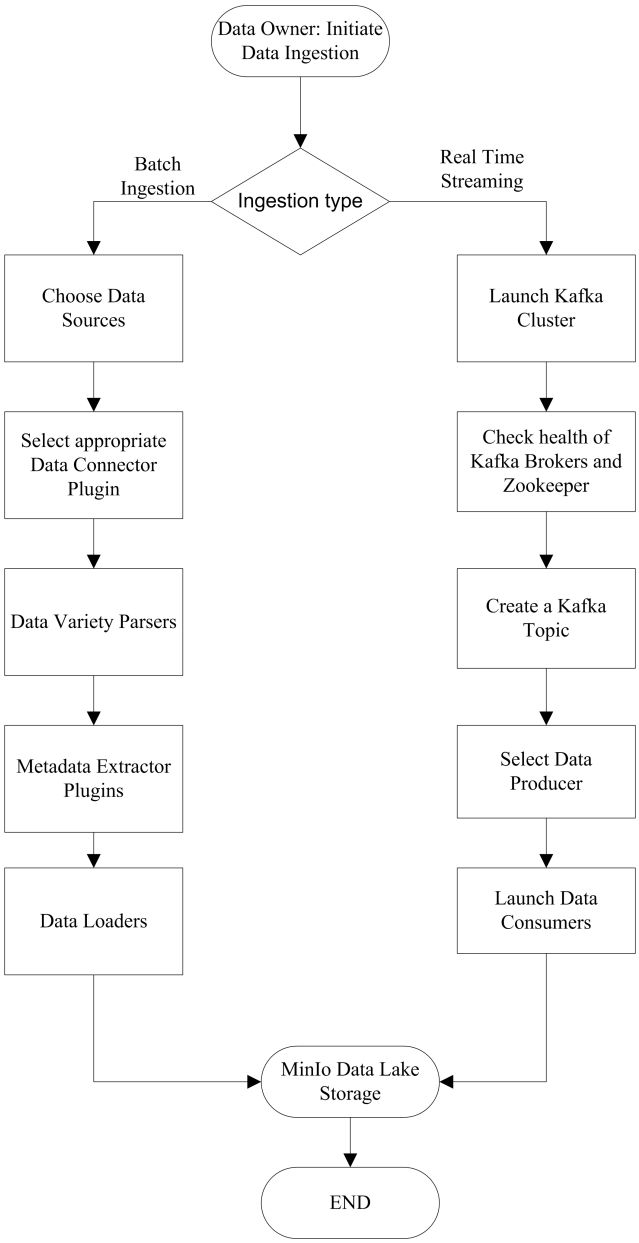

FIGURE 5. Flow diagram of data ingestion service workflow.

g. REST API Client: API provides a standard way to communicate and interact for different software components. It provides interoperability across technologies. API's scalability and performance leads to efficient sharing of data which is most needed as part of any data ingestion tool.

2) ADAPTIVE DATA VARIETY TRANSFORMATION (ADVT)

Data Variety Modules handle the transformation and processing of different data types and formats within the data ingestion service. These modules cater to the specific requirements of various data types, such as structured, semi-structured, and unstructured data. Examples of data variety modules include parsers for CSV, JSON, XML, Avro, Parquet, or specialized modules for handling streaming or real time data generated by IoT sensor devices, social media platforms, log files and geo-spatial data. Finally, to transform data of OAUTH/Access Tokens secured REST API's a special module is includeed to adress the same. This later mentioned module is used to import specific data types like Diacom Images, Health Care Management Systems Data like Open EMR etc. These modules ensure that data from diverse sources is properly transformed, validated, and prepared for ingestion into the Data Lake.Module configuration involves setting up the appropriate libraries and modules for data processing tasks such as CSV parsing, JSON manipulation, or XML parsing or handling security authentications and authorizations. Following are the supported data types:

a. Structured Data: Element of a service mainly deals with bringing in structured data in the form of tables, CSV, TSV, and Excel files from relational databases, data warehouses, local file systems, websites or URLs, and cloud storages using the service's respective connectors.

b. Semi-Structured Data: The component of the service is primarily concerned with ingesting of semi-structured data in the form of key-value pairs, XML, Avro, parquet from document databases, NoSQL Databases, local file system, websites, URL's and Cloud Storages using its specific service connectors.

c. Un-Structured Data: Module focusses fundamentally on Unstructured data kind of texts, videos, images from local file system, websites, URL's and Cloud Storages using its specific service connectors.

d. Health Care Data: Module of the service to connect to open-source health care management systems like Open EMR and Orthanc an PACS Diacom Server to extract health care data like patient details, history, diagnosis, prescriptions, imaging or radiology modality data via API Access Connector by registering Data ingestion service as a REST API Client by following OAUTH security mechanism procedures.

e. Streaming Data Ingestion: The need for a streaming interface module or component as part of a unified data ingestion software arises from the increasing demand for real-time data processing and analysis. Streaming data is a constant flow of data that is created and handled in real time or very close to it. Traditional batch processing methods don't work well with streaming data because they were made for processing static datasets. By adding a streaming interface module to the unified data ingestion software, companies can deal with the problems that come with streaming data. It allows for real-time processing, low-latency analytics, continuous data availability, scalability, and event-driven capabilities. This lets businesses take advantage of the value of streaming data and make choices quickly and wisely based on the most up-to-date information. The process of ingesting streaming data with a Kafka cluster and saving it in a MinIo or S3 bucket is meant to be a smooth and efficient way to get real-time data into a data lake and keep it there. With this method, the user only needs to make a Kafka topic of interest

to start putting streaming or real-time data into the MinIo data lake.

3) INTELLIGENT METADATA MANAGEMENT (IMM)

The Metadata Management Module focuses on managing metadata associated with the ingested data. It includes functionalities for capturing and storing metadata, such as data schemas, lineage, descriptive statistics, and cataloging details. The module should provide capabilities for efficient metadata search, discovery, and retrieval, allowing users to understand the structure, context, and provenance of the ingested data. In addition, this module includes a feature to enrich metadata by adding descriptive statistics, helping to describe trends, patterns, and insights about the dataset for end-users.

A significant part of the service is extracting and handling metadata about the ingested data. This module is very important for keeping track of data lineage, owner information, technology specs, business context, and operational information. Data lineage information shows the data's origins and transformations, while owner details link responsible individuals or organizations, fostering accountability and collaboration.

The module also creates and stores pre-computed views of data statistics, providing insights without requiring intensive computations. It uses a metadata repository for efficient storage and retrieval of metadata, which is stored as binary large objects (blobs). This repository centralizes all extracted information and pre-computed views, improving discoverability.

The Intelligent Metadata Management Module leverages MinIO object storage for scalability, storing metadata attributes in JSON format, and integrates ElasticSearch for data cataloging, enhancing both performance and scalability. However, implementing metadata management in real-world scenarios can present challenges, particularly in scaling the solution across large datasets and ensuring seamless integration with legacy systems. For example, traditional data governance frameworks often rely on structured relational databases, while proposed metadata management benefits from object storage systems like MinIO. These differences may require an organization to adopt new data governance practices or develop middleware solutions to bridge gaps between legacy and modern systems.

Additionally, maintaining data governance across evolving infrastructures presents ongoing challenges. Metadata must remain compliant with organizational policies and regulatory requirements, meaning integration with existing governance frameworks is crucial. The module implements role based access control policies to implement authentication and authorization for only data lake administrator, data owner and domain expert to access the metadata catalogs. This implements governance policy, ensuring alignment with data security, privacy regulations, and auditability standards. By integrating these metadata management capabilities and implementing a structured data organization hierarchy, the data ingestion service enhances discoverability, reduces search time, and promotes efficient data exploration. This allows data analysts to locate relevant data within the cleaned and processed zones of the Data Lake, enabling valuable insights and informed decisions. Figure 6. depicts the overview of Metadata Management components of the Data Ingestion as a Service.

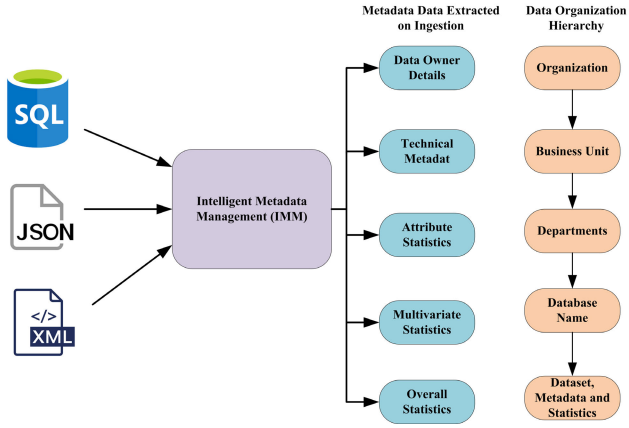

FIGURE 6. Intelligent Metadata Management components of the DIaaS.

The combination of these data ingestion components can handle data varieties, sources and protocol confronted in a data lake environment. This design helps effortless ingestion, transformation, and management of data from various sources, with thorough analysis and insights from their Data Lake.

IV. SERVICE IMPLEMENTATION USE-CASE SCENARIOS

Enterprise applications across various sectors—such as healthcare, finance, supply chain, smart city management, and energy harvesting—are increasingly reliant on the ability to handle diverse data types, including structured, semi-structured, and unstructured data. These applications must manage data from a wide range of sources, often in real-time, to deliver critical insights and maintain operational efficiency. The complexity of processing and integrating this data is further compounded by the need for scalable solutions that can handle both batch and real-time data processing. Additionally, effective metadata management is essential to ensure data governance, traceability, and compliance. Following are few example of enterprise-level use case, highlighting the need for a unified data ingestion service:

- Healthcare Data Integration: Integrates patient records, medical imaging, and IoT data, ensuring compliance and real-time monitoring of patient vitals.
- Financial Services Fraud Detection: Processes transactional and customer data in real-time to detect and prevent fraud, requiring secure and scalable data handling.
- Supply Chain Management: Collects and processes data from suppliers, logistics, and retailers, enabling

real-time tracking and efficient management of complex supply networks.
- Smart City Management: Ingests data from traffic sensors, public services, and social media to optimize urban operations and improve citizen services in real-time.
- Energy Harvesting Sector: Protects energy security and privacy while managing diverse data from IoT devices and energy nodes, requiring efficient real-time data processing [10].

The primary challenge of this work was designing a unified interface capable of handling diverse data sources, varying data velocities, and heterogeneous data formats. Each data source and format required specific connectors and plugins, adding complexity to the task. Additionally, optimizing the service for resource efficiency, scalability, and availability presented significant challenges. To address these challenges, there is a pressing need for a unified data ingestion service that can efficiently manage data variety, integrate heterogeneous data sources, support both real-time and batch processing, and ensure robust metadata extraction and management.

Following are the novel contribution of proposed services to handle data in a unified interface from OAuth secured API's from health care and other domains, and also to facilitate ingestion of real time streaming data generated by IoT devices, sensors, Embedded devices, social media feeds etc.

A. HEALTH CARE DATA INGESTION: DICOM API

API provides a standard way to communicate and interact for different software components. It provides interoperability across technologies. API's scalability and performance leads to efficient sharing of data which is most needed as part of any data ingestion tool. DICOM and PACS server open up APIs for importing data related to patient records. An overview of collecting data from PACS using the REST API Client of the proposed service is shown in Figure 7.

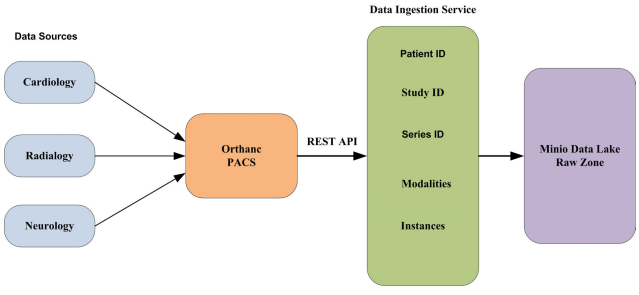

FIGURE 7. Overview of collecting data from PACS using the REST API Client.

Data ingestion from PACS into data lake raw zone requires identifiers like patient, series, study, modality etc. Relevant API's based on required data is chosen, further authentication and access token are used for confidential and integral data flow. After successful authentication and with the above specified parameters, specific data is collected and stored in formats such as PNG, JPEG, DCM etc. As the service uses MinIo as a raw zone for data lake which is highly scalable and helps in storing large volume of data. Data is stored in the following hierarchy as, patient ID, study ID, series ID, instances, and modalities, so that it can be accessed and analysed easily.

B. HEALTH CARE DATA INGESTION: OPENEMR API

OpenEMR an open-source health care management system is used to store patient related data in health care. To extract data like patient details, history, diagnosis, prescriptions, imaging or radiology modality data via API Access connector by registering Data ingestion service as a REST API Client by following OAUTH security mechanism procedures. API security is indeed a crucial aspect when it comes to exchanging data between applications. One commonly used method for securing APIs is through OAuth and access tokens. OAuth is an open standard for authorization that enables secure authentication and authorization of applications to access protected resources on behalf of a user. Any service that targets to acquire data from API need to register as a client application with the given credentials. The credentials help in authorization and getting the access token which has short expiry time and the data is collected securely. The proposed system would get the Medical Records from OpenEMR API using above specified methodology. With the help of OAuth mechanism, the proposed work can securely ingest data from OpenEMR API by upholding integrity and privacy. The following Figure 8. depicts data transfer as mentioned above.

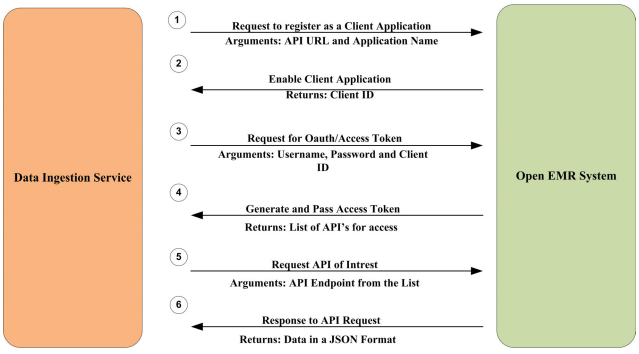

FIGURE 8. Data Transfer Flow between Data Ingestion Service and OpemEMR.

C. STREAMING DATA INGESTION

The need for a streaming interface module or component as part of a unified data ingestion software arises from the increasing demand for real-time data processing and analysis. Streaming data is a constant flow of data that is created and handled in real time or very close to it.

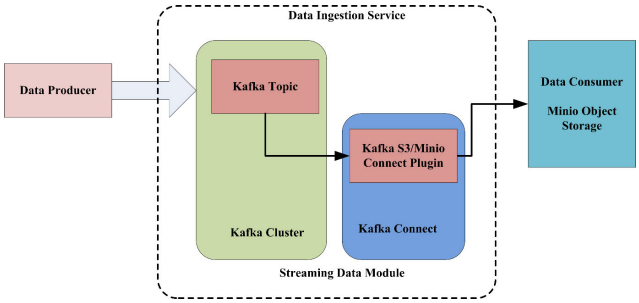

FIGURE 9. Streaming Data Ingestion using DIaaS.

Traditional batch processing methods don't work well with streaming data because they were made for processing static datasets. By adding a streaming interface module to the unified data ingestion software, companies can deal with the problems that come with streaming data. It allows for real-time processing, low-latency analytics, continuous data availability, scalability, and event-driven capabilities. This lets businesses take advantage of the value of streaming data and make choices quickly and wisely based on the most up-to-date information. The process of ingesting streaming data with a Kafka cluster and saving it in a MinIo or S3 bucket is meant to be a smooth and efficient way to get real-time data into a data lake and keep it there. With this method, the user only needs to make a Kafka topic of interest to start putting streaming or real-time data into the MinIo data lake. First, the Kafka cluster set up in the background by the data entry service works as a reliable and scalable message streaming platform that can handle large amounts of data in real time. It has a publish-subscribe approach where data producers send messages to Kafka topics and data consumers subscribe to those topics to get the data. To start the process of ingestion, the user makes a Kafka topic for the specific stream of interest. This topic serves as a way for the information to move. The user can identify different topics based on the sources of the data, the types of the data, or any other relevant criteria. Once the Kafka topic is set up, data producers like IoT sensor devices, social media sites, or log files start sending data messages to the topic. Depending on the application and needs, these messages can be in any format, such as JSON, Avro, or binary. The Kafka cluster makes sure that the incoming data is spread out evenly across its parts so that it can be processed in parallel and handle errors. It handles streaming data in a scalable way, meeting the needs of real-time data ingestion for high throughput and low delay.

On the other side, the Unified Data Ingestion Service has a feature that connects to the Kafka cluster and stores the data in the MinIo or S3 bucket. The Kafka Connect framework and the Kafka S3 Connect plugin are used by this package. The Kafka S3 Connect tool connects the Kafka cluster to either the MinIo or S3 storage. It reads the data messages from the Kafka topic and writes them to the bucket in the data lake that was set up for that purpose. The plugin handles any possible errors or retries during the ingestion process. This makes sure that the delivery is solid and that there are no problems. By using this method, a user can easily put streaming or real-time data into the MinIo data lake by making a Kafka topic and then adding the data to it. This streamlined method makes it possible to bring in data in a way that is efficient, scalable, and reliable. This makes it easy to analyze, process, and use the data within the data ecosystem. Figure 9. provides complete picture of ingesting streaming data in to Data Lake using streaming interface of DIaaS.

V. RESULTS AND DISCUSSION

A. EXPERIMENTAL SET-UP OF UNIFIED DATA INGESTION SERVICE FRAMEWORK

To validate the proposed Unified Data Ingestion System across various data sources and varieties, we have established a robust experimental framework. Firstly, a Data Lake environment has been set up using Minio Object Storage serves as a raw zone of lake. Subsequently, the proposed Data Ingestion Service framework is containerized and packed with all its packages, libraries, connectors, plugin as a docker container and deployed on a Dell Workstation with Intel Xeon Dual Core Processor, 8 GB DIMM DRAM 10Gb Ethernet Port and 500 GB SSD Hard Disk. This includes connectors for different databases, storage, cloud storage, etc., all meticulously provisioned. Additionally, API servers such as Orthanc and PACS, alongside a Kafka cluster, have been integrated to facilitate testing with streaming data. Following Table 2. list the technical specifications such as programming languages, tools, platforms, libraries, connectors, data formatters, metadata extractors and data loaders. To comprehensively evaluate the functionality and performance of the data ingestion service, we have curated sample data sources from diverse origins. These sources encompass various data types and formats, including Relational database tables, CSV files, JSON, XML, Avro, and JSON APIs, ensuring representation across structured, semi-structured, and unstructured data categories. Dataset collected consists of structured datasets with varying record counts, different item counts in semi-structured data and finally, different sizes of unstructured data. Dataset for the experiments with above settings are collected from domains such as Retail Data [11], Stock Market [12], [13], ERP, CRM [14], Weather [15], [16] and HealthCare data [17]. Validation of the study is performed by corroborating features of proposed system in comparison with existing tools and also measure system performance metrics for different data sources, verities and protocols.

B. COMPARATIVE STUDY

Study performs validation of the Unified Data Ingestion Service, it is crucial to corroborate its capabilities with existing data ingestion tools and technologies. A comparative study is conducted by identifying and analyzing various features of both services. Parameters such as sources,

TABLE 2. Data ingestion service: technical specifications.

Data Ingestion Service: Technical Specifications	
Category	**Details**
Programming languages	Python, HTML, CSS, JavaScript
Platforms	Streamlit Web Framework
Data Source Connectors	paramiko, requests, boto3, pymongo, mysql.connector, psycopg2-binary
Data Variety Parsers	csv, json, PyPDF2, pdfplumber, docx2txt, docx, python-docx, rdflib, pytesseract, moviepy
Metadata Extractors	minio, elasticsearch, libmagic, python-magic, rake_nltk, pandas, PIL, tika, spacy, nltk, scipy, sklearn, networkx, matplotlib
Security and System	os, subprocess
Visualization	Matplotlib, plotly-dash
Containerization	Docker, Docker Compose

varieties, frequency, and protocols supported play a vital role in this evaluation. Firstly, the sources from which data can be ingested should be compared, including databases, file systems, APIs, cloud storages, file browse, streaming platforms, and IoT devices. Assessing whether the Unified Data Ingestion Service supports a similar or broader range of sources compared to existing tools ensures compatibility and data coverage. Secondly, the types of data varieties supported by each service should be compared, including structured, semi-structured, unstructured, healthcare and multimedia data. Determining whether the Unified Data Ingestion Service can handle the same or wider variety of data types ensures versatility in data ingestion. Next, evaluating the frequency of data ingestion, such as batch or real-time, is essential to match the requirements of different use cases. Lastly, protocols supported by the services should be considered, such batch or stream processing or custom protocols. By carefully assessing these parameters, a comprehensive comparative study can be conducted to determine the strengths and capabilities of the Unified Data Ingestion Service in relation to existing tools and technologies. Results of the comparative study is tabulated as below in Table 3.

1) INFERENCES OF COMPARATIVE STUDY

Comparative study results reveal the component modules, features and services offered by this work are not implemented in any of the existing approaches. Service offers a unified interface to perform and handle different needs of data varieties, requirements and from different sources efficiently. From this evaluation results, this proposed work will try to fill the research gap in literature on existing data ingestion approaches.

C. PERFORMANCE VALIDATION OF THE STUDY

The validation of the Unified Data Ingestion Service entails measuring key performance metrics across varying data fields, types, sources, and protocols to ensure its efficiency, scalability, and resource utilization. Metrics such as Data Ingestion Time (Latency/Throughput) gauge the service's

TABLE 3. Comparative study results.

Validation Parameter	Unified Data Ingestion Service	Sqoop	Flume	Nifi	Kafka
Data Variety	Structured, Semi-Structured, Un-Structured	Structured	Semi-Structured	Structured and Semi-Structured	Semi-Structured
Data Format	CSV, JSON, Avro, XML, Parquet, TXT, PNG, JPEG, DIM, MP3, MP4	CSV	JSON, Avro, XML, Parquet	CSV, JSON, Avro, XML, Parquet, TXT	JSON, Avro, XML, Parquet
Data Protocol	(REST API, CURL, HTML, HTTP, HTTPS)	✓	×	×	×
Relational Databases Sources	MySQL, PostgreSQL	✓	✓	×	✓
NoSQL Databases	(Mongo and Casandra DB)	✓	×	✓	×
Unstructured Data	(Images, Text, Video)	✓	×	✓	×
Cloud Storage Interfaces	✓	×	×	×	×
File Browse	(CSV, JSON, XML, Avro)	✓	×	✓	×
Web URL Streaming Data	✓ ✓	× ×	× ✓	× ✓	× ✓
Access Data from OAuth2 Authenticated API Servers	✓	×	×	×	×
Ingest from PACS (Orthanc Server) – Health-Care Data	✓	×	×	×	×
Metadata Extraction	✓	×	×	×	×
Data Organization	✓	×	×	×	×
User Interface	Web Service	CLI	CLI	Web user interface	CLI
Service Throughput	Batch Data Throughput: 33.27 MB/s	NA	NA	NA	605 MB/s (10 Partitions, instance with 8 vCores, 64 GB RAM) [18]

speed and efficiency in processing data from diverse sources and protocols, while monitoring CPU and memory usage offers insights into resource utilization during ingestion processes. Network Input/output (I/O) measurements assess data transfer rates and network latency, while evaluating Block Device Bandwidth provides an understanding of storage device performance. By analysing these metrics comprehensively, potential optimization areas can be identified, ensuring the service's capability to handle different data scenarios effectively and validating its overall system performance

1) STRUCTURED DATA VALIDATION

To validate the database connectors and structured data variety module of the Data Ingestion Service, relational tables with varying record counts across domains like retail, ERP, and CRM are synthesized using MySQL and PostgreSQL databases. These databases are configured on local and remote servers to simulate different deployment scenarios. The Data Ingestion Service is then configured to connect to these databases, ingesting data into the Data Lake from the relational tables. Performance metrics data is tabulated below in Table 4.

TABLE 4. Structured dataset performance validation.

No. Dataset Records	Ingestion Time (Seconds)	CPU (%)	Memory Usage (7.676 GiB)	Memory Usage (%)	NET I/O (MB)	BLOCK I/O (MB)
2,300 [11]	0.1157	14.17	195.6 MiB	2.49	216	44.6
7,000 [12]	0.147346	16.95	195.6 MiB	2.49	216	44.6
35,500 [13]	6.686514	99.51	516.4 MiB	6.57	262	44.6
54,200 [14]	7.786514	100.15	870.9 MiB	11.08	312	44.6

Figure 10. provides a visual depiction of the intricate relationship among various performance metrics associated with structured data ingestion utilizing the Data Ingestion as a Service framework within the context of a Data Lake environment:

2) SEMI-STRUCTURED JSON DATA VALIDATION

Semi-Structured datasets in form of JSON/Avro/XML with items ranging from 2560 to 2,09,579 datasets across different domains are synthesized and ingested using Unified Data Ingestion Service to measure the performance metrics of the system like Data Ingestion Time, CPU Utilization, Memory Usage, Network I/O and Block Memory I/O. Results from the experiment is tabulated as follows in Table 5.

Figure 11. illustrates plot of graph among various performance metrics associated with semi-structured data ingestion utilizing the Data Ingestion as a Service framework within the context of a Data Lake environment

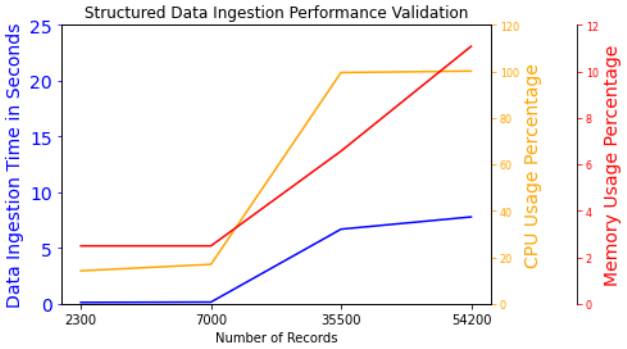

FIGURE 10. Structured data ingestion performance validation.

TABLE 5. Semi-structured data performance validation.

No. Data Items	Ingestion Time (Seconds)	CPU (%)	Memory Usage (7.676 GiB)	Memory Usage (%)	NET I/O (MB)	BLOCK I/O (MB)
2,560 [12]	0.041271	6.15	217.2 MiB	2.76	314	44.6
10,000 [11]	2.289944	101.8	325.1 MiB	4.14	316	44.6
15,000 [14]	6.86483	102	413.6 MiB	5.26	321	44.6
22,635 [15]	15.022683	108	1.5 GiB	20.60	822	44.6
2,09,579 [16]	36.62437	118.8	1.715 GiB	22.34	869	44.6

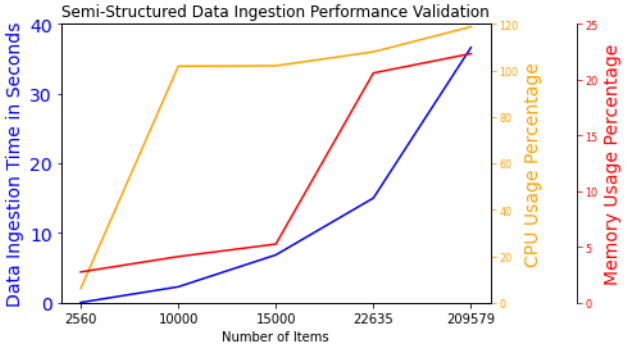

FIGURE 11. Semi-structured data ingestion performance validation plot.

3) UNSTRUCTURED VIDEO/TEXT DATA VALIDATION

Unstructured datasets in form of Video/Text with size from 532 KiloBytes to 100MB video/text datasets are synthesized and ingested using Unified Data Ingestion Service to measure the performance metrics of the system are tabulated in Table 6.

The graphic in Figure 12. illustrates the link between performance indicators of Unstructured video and text data ingestion utilizing Data Ingestion as a Service for Data Lake.

4) UNSTRUCTURED IMAGE DATA VALIDATION

Un-Structured datasets in form of images with size from 100 KB to 10MB from different domains like machine learning image datasets to healthcare datasets are synthesized

TABLE 6. Video/text data performance validation.

Data Size (KB)	Ingestion Time (Seconds)	CPU (%)	Memory Usage (7.676 GiB)	Memory Usage (%)	NET I/O (MB)	BLOCK I/O (MB)
532 [23]	0.106968	7.06	546.9 MiB	6.96	506	44.6
10000 [23]	1.045364	57.74	546.8 MiB	6.96	517	44.6
50000 [23]	4.602294	64.38	547.2 MiB	6.96	541	44.6
100000 [23]	4.779542	69.81	930.8 MiB	11.84	675	44.6

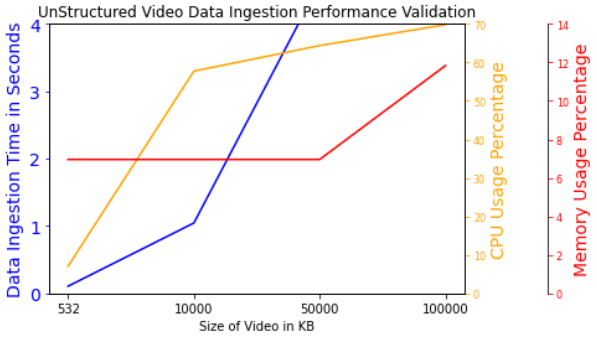

FIGURE 12. Unstructured Video/Text Data Performance Validation Plot.

TABLE 7. Image data performance validation.

Image Data Resolution (KB)	Ingestion Time (Seconds)	CPU (%)	Memory Usage (7.676 GiB)	Memory Usage (%)	NET I/O (MB)	BLOCK I/O (MB)
100 [24]	0.055744	12.95	570.6 MiB	7.26	501	44.6
500 [16]	0.084176	33.33	570.6 MiB	7.26	501	44.6
2500 [17]	0.131601	46.65	585.9 MiB	8	504	44.6
10000 [17]	0.287565	87.33	668.8 MiB	8.51	505	44.6

and ingested using Unified Data Ingestion Service. Results from the experiment is tabulated as follows in Table 7.

Figure 13. is a graph that exhibits the correlation between performance indicators of Unstructured image data ingestion utilizing Data Ingestion as a Service for Data Lake.

D. DATA ANALYSIS

Inferential/Statistical analysis is performed by Testing Hypothesis and Statistical relationships and significance of variables is estimated by Regression using Minitab statistical tool

Step1: Formulate Hypothesis:

- Hypothesis -1
  H1o: There is no significant relation between Number of Records/Items of a dataset and ingestion time.

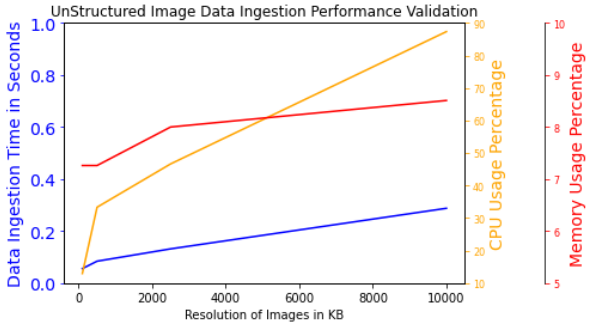

FIGURE 13. Unstructured image data performance validation Plot.

  H1a: There is a significant relation between between Number of Records/Items of a dataset and ingestion time.
- Hypothesis -2
  H2o: There is no significant relation between Number of Records/Items of a dataset and CPU Resources Usage H2a: There is a significant relation between between Number of Records/Items of a dataset and CPU Resources Usage.
- Hypothesis -3
  H3o: There is no significant relation between Number of Records/Items of a dataset and Memory Resources Usage.
  H3a: There is a significant relation between between Number of Records/Items of a dataset and Memory Resources Usage.
- Hypothesis −4
  H4o: There is no significant relation between Number of Records/Items of a dataset and Network Input/Output. H4a: There is a significant relation between between Number of Records/Items of a dataset Network Input/Output.

Step2: Select Significance Level A multiple regression test is conducted to find the statistical relationship between endogenous and exogenous variables. A significant level of 5

Step3: Select Distribution Normal Distribution is assumed.

Step4: Sample Description Data for different number of records and items of structured and semi-structed data from different domains/category are sampled to test hypothesis using statistical methods.

Step5: Regression Test Regression is the determination of statistical relation between two or more variables. In this study statistical relation between No. Records as endogenous variable and Ingestion Time, CPU Usage, Memory Usage and Network IO as exogenous variables.

As the study involves continuous, four independent variables, multiple linear regression performed to identify the statistical relationship. Tabulation below illustration the regression analysis. Multiple Linear Regression: No. Records vs Ingestion Time, CPU Usage, Memory Usage and Network IO.

TABLE 8. Sample size.

Link function	Logit
Rows used	9

TABLE 9. Regression co-efficient table.

Term	Coef	SE Coef	Z-Value	P-Value	VIF
Constant	44505	16566	2.69	0.055	
Ingestion Time	8598	1042	8.25	0.001	5.47
CPU Usage	-223	160	-1.39	0.237	2.08
Memory Usage	540	2675	0.20	0.850	15.99
Network IO	-160.4	71.8	-2.23	0.089	12.22

TABLE 10. Model performance metrics.

Metric	Value
Deviance	14703.9
R-Sq	97.51%
Deviance R-Sq (adj)	95.02%
AIC	0.00%
AICc	14703.9
BIC	97.51%
Area Under ROC Curve	97.51%

TABLE 11. Analysis of variance test.

Source	DF	Adj SS	Adj MS	F-Value	P-Value
Regression	4	33852526054	8463131513	39.14	0.002
Ingestion Time	1	14732733095	14732733095	68.14	0.001
CPU Usage	1	416623162	416623162	1.93	0.237
Memory Usage	1	8795329	8795329	0.04	0.850
Network IO	1	1079092747	1079092747	4.99	0.089
Error	4	864821004	216205251		
Total	8	34717347058			

Method: Sample size of dataset is tabulated in Table 8.

Regression Equation:

$$P(1) = \frac{\exp(Y')}{1 + \exp(Y')} \tag{1}$$

$$\begin{aligned} \text{No. Records} = {} & 44505 + 08598\,\text{Ingestion Time} \\ & - 223\,\text{CPU Usage} + 540\,\text{Memory Usage} \\ & - 160.4\,\text{Network IO} \end{aligned} \tag{2}$$

Coefficients: Table 9. records the regression co-efficient data

Model Summary: Model summary is tabulated in Table 12.

Analysis of Variance: Results from analysis of variance test are recorded in Table 11.

Residual Plots for Records: Figure 14. provides the plots for representing residual records

E. INFERENCE OF DATA ANALYSIS

Inference Drawn from Hypothesis-1: Null hypothesis is rejected and alternative Hypothesis is accepted, hence there is significant association between Number of Records/Items of a dataset to Ingestion Time.

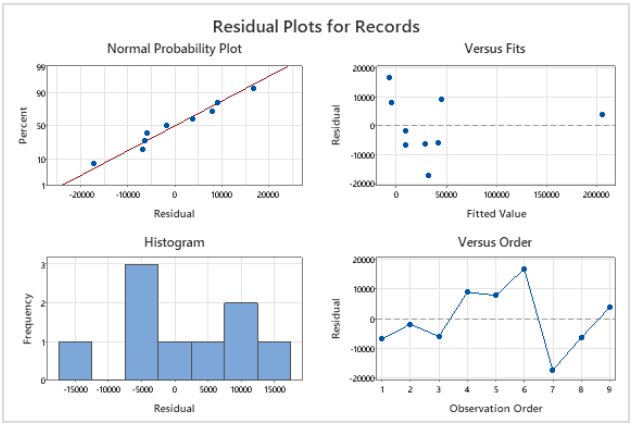

FIGURE 14. Residual plots for records.

TABLE 12. Summary of hypothesis testing.

Hypothesis	Null Hypothesis	Alternative Hypothesis	Independent Variable	Dependent Variable
H1	Rejected	Accepted	Number of Records/Items	Ingestion Time
H2	Accepted	Rejected	Number of Records/Items	CPU Resource Usage
H3	Accepted	Rejected	Number of Records/Items	Memory Resource Usage
H4	Rejected	Accepted	Number of Records/Items	Network Input/Output

Inference Drawn from Hypothesis-2: Null hypothesis is accepted, hence there is no significant association between Number of Records/Items of a dataset to CPU Resource usage.

Inference Drawn from Hypothesis-3: Null hypothesis is accepted, hence there is no significant association between Number of Records/Items of a dataset to Memory Resource usage.

Inference Drawn from Hypothesis-4: Null hypothesis is rejected and alternative Hypothesis is accepted, a significant relation between Number of Records/Items of a dataset to Network Input/Output.

Summary of Inferences: Summary of data analysis results are tabulated in Table 12.

VI. CONCLUSION

Study aimed at filling the existing research gap in data ingestion tools for data lake. This work proposed a design of Data Ingestion as a service for data lake. Novelty of the service is a unified interface to support different data sources, variety, protocols and paradigms, to address different data requirements of organization. It offers a scalable solution for ingesting and transforming data from various sources. Comparative study proves the superiority of proposed service to existing tools, in terms of handling data variety, sources, formats and protocols. Performance evaluation of the proposed system underscore a notable

association between the average latency of the data ingestion service and the size of the dataset, wherein network input/output (I/O) similarly demonstrates dependency on data size, while CPU and memory usage exhibit no significant correlation. Performance validation study of data ingestion latencies across various data types, significant differences were observed based on the structure and size of the data. For structured data, with records ranging from 2,300 to 54,200, the average ingestion latency per record was calculated to be approximately 148.1 microseconds. In contrast, semi-structured data, with records varying from 2,560 to 209,579, exhibited a higher average ingestion latency per record of approximately 234.2 microseconds. This indicates that the semi-structured data format incurs additional overhead during ingestion compared to structured data. Furthermore, the ingestion latency for video data was measured on a per kilobyte (KB) basis, resulting in an average latency of approximately 65.6 microseconds per KB. Image data ingestion demonstrated the lowest latency among the evaluated data types, with an average of approximately 42.7 microseconds per KB. These findings highlight the varying performance implications of different data formats during ingestion processes, underscoring the importance of considering data structure and size in the optimization of data ingestion systems. The usability and flexibility of the service have also been assessed, highlighting its user-friendly interface and ease of configuration. To further enhance the scalability, availability, and efficiency of the data ingestion service, future work will focus on addressing key challenges related to the optimal storage of large volumes of real-time data. Developing advanced storage optimization techniques is crucial for ensuring that the system can manage big data efficiently, which directly impacts overall system performance and resource utilization. Additionally, the study aims to tackle the challenge of adapting to changes in data sources by designing and implementing generic adapters or connectors. These would allow the system to dynamically accommodate various data sources as they evolve, providing a robust solution to the problem of source variability and ensuring seamless data ingestion across heterogeneous environments.

REFERENCES

[1] S. Gupta and V. Giri, "Data lake ingestion strategies," in *Practical Enterprise Data Lake Insights*. Berkeley, CA, USA: Apress, 2018, pp. 33–85.

[2] F. Nargesian, E. Zhu, R. J. Miller, K. Q. Pu, and P. C. Arocena, "Data lake management: Challenges and opportunities," *Proc. VLDB Endowment*, vol. 12, no. 12, pp. 1986–1989, Aug. 2019, doi: 10.14778/3352063.3352116.

[3] M. Irfan and J. P. George, "A systematic review of challenges, tools, and myths of big data ingestion," in *Data Science and Security* (Lecture Notes in Networks and Systems), 2022, doi: 10.1007/978-981-19-2211-4_43.

[4] J. Alwidian, S. A. Rahman, M. Gnaim, and F. Al-Taharwah, "Big data ingestion and preparation tools," *Modern Appl. Sci.*, vol. 14, no. 9, p. 12, Aug. 2020, doi: 10.5539/mas.v14n9p12.

[5] H. Mehmood, E. Gilman, M. Cortes, P. Kostakos, A. Byrne, K. Valta, S. Tekes, and J. Riekki, "Implementing big data lake for heterogeneous data sources," in *Proc. IEEE 35th Int. Conf. Data Eng. Workshops (ICDEW)*, Apr. 2019, pp. 37–44, doi: 10.1109/ICDEW.2019.00-37.

[6] M. Cherradi and A. E. Haddadi, "A scalable framework for data lakes ingestion," *Proc. Comput. Sci.*, vol. 215, pp. 809–814, Jan. 2022, doi: 10.1016/j.procs.2022.12.083.

[7] A. Matacuta and C. Popa, "Big data analytics: Analysis of features and performance of big data ingestion tools," *Inf. Economica*, vol. 22, pp. 25–34, Jun. 2018, doi: 10.12948/issn14531305/22.2.2018.03.

[8] G. Sharma, V. Tripathi, and A. Srivastava, "Recent trends in big data ingestion tools: A study," in *Research in Intelligent and Computing in Engineering*, 2021.

[9] R. Hai, C. Koutras, C. Quix, and M. Jarke, "Data lakes: A survey of functions and systems," 2021, *arXiv:2106.09592*.

[10] Q. Pan, J. Wu, A. K. Bashir, J. Li, W. Yang, and Y. D. Al-Otaibi, "Joint protection of energy security and information privacy for energy harvesting: An incentive federated learning approach," *IEEE Trans. Ind. Informat.*, vol. 18, no. 5, pp. 3473–3483, May 2022, doi: 10.1109/TII.2021.3105492.

[11] M. Zaharia. (2019). *Spark: The Definitive Guide*. Accessed: Mar. 4, 2022. [Online]. Available: https://github.com/databricks/Spark-The-Definitive-Guide/tree/master/data/retail-data

[12] A. Pal. (2023). *Indian Stock Market Dataset*. Accessed: Jun. 15, 2023. [Online]. Available: https://www.kaggle.com/datasets/adritpal08/indian-stock-market-dataset?select=stk.json

[13] M. Verma. (2018). *NSE Stocks Data*. Accessed: Apr. 3, 2022. [Online]. Available: https://www.kaggle.com/datasets/minatverma/nse-stocks-data

[14] N. Dincer. (2021). *Customer Relationship Management (CRM)*. Accessed: Jun. 13, 2023. [Online]. Available: https://www.kaggle.com/code/nihandincer/customer-relationship-management-crm/input

[15] (2020). *Openweathermap—Climatic Information*. Accessed: Sep. 8, 2020. [Online]. Available: https://openweathermap.org/

[16] J. O. Oickle. (2012). *Openweathermap.org*. Accessed: Jun. 15, 2023. [Online]. Available: https://openweathermap.org/

[17] FNL for CR. (2023). *Cancerimagingarchive*. Accessed: Jun. 15, 2023. [Online]. Available: https://www.cancerimagingarchive.net/access-data/

[18] A. Foundations. (2023). *Apache Kafka Performance*. Accessed: Jun. 13, 2023. [Online]. Available: https://developer.confluent.io/learn/kafka-performance/

[19] J. Shil. (2018). *Amazon Product Dataset*. Accessed: Jun. 15, 2023. [Online]. Available: https://www.kaggle.com/datasets/joyshil0599/a-comprehensive-dataset-of-100k-amazon-products

[20] M. S. R. Reddy. (2023). *Salary Data*. Accessed: Jun. 15, 2023. [Online]. Available: https://www.kaggle.com/datasets/mohithsairamreddy/salary-data

[21] B. Chambers and M. Zaharia. *Bike Trip Data*. Accessed: Jun. 14, 2023. [Online]. Available: https://github.com/databricks/Spark-The-Definitive-Guide/tree/ master/data/bike-data

[22] M. Zaharia. *Spark—Retail Dataset*. Accessed: Jun. 14, 2023. [Online]. Available: https://github.com/databricks/Spark-The-Definitive-Guide/tree/master/data/retail-data/all

[23] (2023). *Sample Video Files*. Accessed: Jun. 15, 2023. [Online]. Available: https://file-examples.com/index.php/sample-video-files/

[24] (2023). *Sample Images Download*. Accessed: Jun. 14, 2023. [Online]. Available: https://file-examples.com/index.php/sample-images-download/

SREEPATHY H. V. received the B.E. degree in electronics and communication engineering from Visvesvaraya Technological University, Belgaum, in 2009, and the M.S. degree in embedded systems from Manipal Academy of Higher Education, Manipal, Karnataka, India, in 2012. He has experience in setting up private cloud using the OpenStack cloud suite and has good hands-on experience with AWS Cloud Services, as well as the DevOps life cycle stages and tools. He has also a Technical Consultant with the International Telecommunication Union (ITU), sponsored GovStack project for API testing, sandbox integration, and the Cloud Infrastructure Team. He is currently an Assistant Professor with Manipal School of Information Sciences, Manipal Academy of Higher Education. He is an Amazon Web Services (AWS) accredited AWS, an AWS Academy Graduate-AWS Academy Cloud Foundations, and specializes in cloud architecting and cloud operations. His research interests include cloud computing, DevOps, data lakes, and big data architecture.

DINESH RAO B. received the M.Tech. degree from the Indian Institute of Technology Kanpur, Kanpur, in 1993, and the Ph.D. degree from Manipal Academy of Higher Education, Manipal, in 2014. He is currently a Professor with Manipal School of Information Sciences, Manipal Academy of Higher Education. He has around 26 years of teaching experience and has published around 11 research papers in national/international journals/conferences. His research interests include fractals and theoretical computer science.

MOHAN KUMAR JAYSUBRAMANIAN received the B.E. degree in electrical engineering from Bharathiar University, Tamil Nadu, in 2000, and the M.S. degree in VLSI-CAD and the Ph.D. degree from Manipal Academy of Higher Education, Manipal, Karnataka, India, in 2003 and 2019, respectively. He was an Associate Professor with Manipal School of Information Sciences, Manipal Academy of Higher Education, until December 2022. He was also with the Department of Electronics and Communication Engineering, Kumaraguru College of Technology, Coimbatore, India, until July 2024. He is currently a Consultant with Frameworks and Tools, MulticoreWare, Coimbatore, India. His research interest includes embedded systems.

DEEPAK RAO B. received the B.E. degree in mechanical engineering from Mangalore University, Karnataka, in 1995, and the M.S. degree in e-commerce and the Ph.D. degree from Manipal Academy of Higher Education, Manipal, Karnataka, India, in 2002 and 2021, respectively. He is currently an Associate Professor with Manipal School of Information Sciences, Manipal Academy of Higher Education. His research interests include big data and data analytics.