A Scalable Big Data Processing Architecture for Enterprises

[1]Aishrith P Rao, [2]Raghavendra JC, [3]Dr. Sowmyarani CN, [4]Dr.Padmashree T

[1, 2]Student, [3]Associate Professor, [4]Assistant Professor
[1]Department of Computer Science and Engineering,
[1]RV College of Engineering, Bengaluru, India

Abstract : The explosive pace of innovation, corporate apps, media platforms, and Internet of things in recent years has resulted in a massive increase of corporate data. The cloud service provider offered expandable storage capacity to handle company data expansion and gave evaluators, customers, and corporate clients with faster access to data. Classifying, organising, and storing most of this data, as well as processing it to offer business intelligence, is a problem. The diversity, speed, size, and usage of data pose a challenge to interpret big data adequately. Sophisticated business requirements are difficult to implement, produce intelligence, and enable smart info based choices in a timely manner. Consumers often examine enterprise-wide data from multiple database schemas since a massive data lake incorporates inflows of data from a variety of business segments. Based on required data models, consumers can perform complicated computation, execute queries, and make large database connections to create needed metrics. Finding insights from information is generally an exhaustive and high labor task. To achieve an ideal balance between price, efficiency, and availability, a big data architecture in the business demands considerable data modelling methodologies. This study tackles these issues by offering a scalable and efficient measure for organising and storing information in Big Data Lake. It explains some of the fundamental ideas and methodologies for creating scalable data architectures in a distributed setting. It also explains how it overcomes typical obstacles and displays the results.

IndexTerms **– Big data. Data lakes, Data Warehousing, Spark, Databricks**

I. INTRODUCTION

Big data processing and migration is a very important module of enterprises that comprise their data infrastructure. Multiple methods have been used in the past for tackling these issues but as the data sizes have increased over the past, the issue of managing the data have become even more complex and difficult. With larger data comes the need of more resources to process this data as well as store this data. Nowadays, more companies are focusing towards cloud services like Azure, AWS for their data solutions which reduces their computational costs and make the whole data infrastructure more streamlined.

Decision support and data analytics have traditionally used data warehouses. Data warehouse technology has evolved since its introduction in the late 1980s, and MPP architectures have resulted in systems that can manage larger data sets. While warehouses were ideal for structured data, today's businesses must manage with unstructured data, semi-structured data, and data having a high degree of diversity, velocity, and volume. Most of these use instances are not well suited for data warehouses, but they definitely may not be the most cost effective.

As businesses began to gather vast volumes of data from a variety of sources, architects began to imagine a single system that could contain data for a variety of analytic solutions and processes. Companies began establishing data lakes – reservoirs for raw data in a range of categories – around a decade ago. While data lakes are good for storing data, they lack some key features: they don't support transactions, they don't enforce quality of the data, and their loss of consistency / isolation makes mixing appends and reads, batch and streaming tasks nearly impossible. Many of the promises of data lakes have not materialised as a result of these factors, resulting in the loss of most of the advantages of data warehouses in many circumstances.

The demand for a versatile, high-performance system isn't going away any time soon. Companies need solutions for SQL analytics, real-time monitoring, business analytics, and artificial intelligence, among other data applications. The majority of recent AI advancements are now in better systems for processing unstructured data (text, photos, video, and audio), yet these are the kind of data which a data warehouse isn't really designed to handle. Using several systems, such as a data lake, numerous data warehouses, as well as other specialised systems like streaming, time-series, graph, as well as picture databases, is a typical technique. Having a variety of platforms adds complexity and, more crucially, time, because data professionals must frequently move or transfer data across systems.

In this paper, the migration and processing of Big Data using cloud services is discussed which can be altered according to the company data architecture. This helps to de couple the data and make it available for different tools provided by that cloud service. The processing is achieved using Spark, SparkSQL and Python and the environment can be chosen as Databricks where the spark jobs were deployed to the cluster. Ingestion can be handled using Kafka while data warehouses like data lakes can be used for storage.

II. LITERATURE REVIEW

Several surveys, studies, and techniques for big data processing have been advocated in the recent decade. Because there is so much data on big data platforms and no one solution that can address big data concerns, researchers are working to develop a more suitable approach to provide business intelligence.

The results of the survey related to big data modelling along with data analytics are reported in [1] using three different visualizations: operations and maintenance database systems, decision analysis database systems, with Big Data technologies. It

also shows a feature-by-feature comparison of large data modelling tools. Hashem and Ranc [2] offer a dynamic approach for dealing with the problem of huge data management. For speedier massive data analysis, non-relational models are offered. Sai and Jyothi [3] explain how to represent organised and unstructured data using big data modelling methodologies. Social media data, cloud platforms, and ontology modelling can all benefit from the proposed modelling.

The authors of [4] examine relational and non-relational models and recommend which database systems should be used for certain use cases. The authors examined two database management systems, including their descriptions, advantages, structures, and applications. They came to the conclusion that non-relational databases offer properties that are better suited to solving the challenge of huge data. NoSQL databases have become a popular alternative for storing, processing, and analysing enormous amounts of data [5].

The authors of [6] investigated ways to quantify or assess data quality. The following are the most essential dimensions to debate, according to them: Availability, a sufficient volume of data, completeness, consistency, interpretability, security, timeliness, objectivity, and so on.

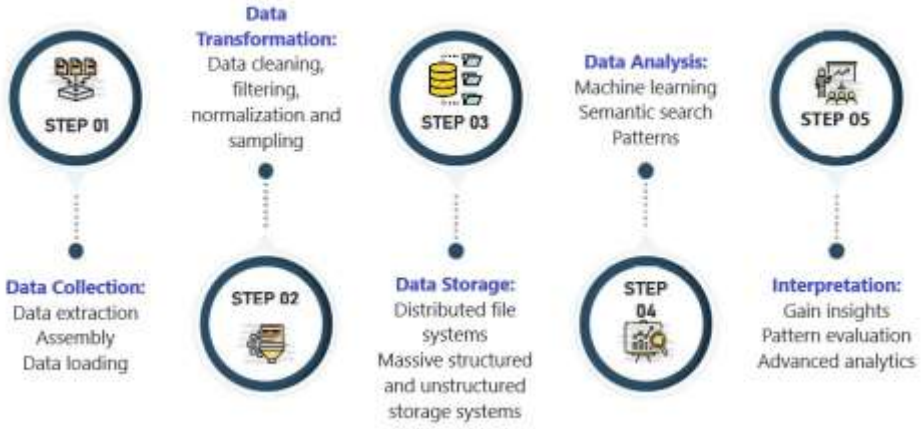
Because of the amount and rapidity of Big Data, it's critical to distinguish between accurate and corrupted data before doing any type of data analysis. Furthermore, because data is generated at a rapid rate from a variety of sources, a more complex type of data integration is required to synchronise the semantics of the data[7]. According to [8] enhancing the quality of incoming Big Data is less important since the amount of incorrect data is deemed negligible in influencing the overall result once the data has been processed and analysed. As a result, it appears that the amount and impact of erroneous data in a big dataset determines which of those two completely opposite schools of thought is most significant. As a result, it's even more critical to understand which dimensions are best for Big Data.

The authors of [9] provide a summary of big data analytics, including its use, benefits, and drawbacks. It also compares several systems for tackling large data storage and explores various tools and approaches for processing large data. Di Tria, Lefons, [10] present a big data design technique to shorten the time it takes to connect different types of data and incorporate new consumer needs on the go. The suggested approach relies on such a virtual data warehouse architecture which eliminates any need for importing. To demonstrate the validity of this strategy in comparison to a traditional one, examples of its use are shown.

III. BIG DATA ARCHITECTURE

3.1 Stages associated with Big Data Processing

There are many stages associated with big data processing which have to be handled while setting up the architecture which have been listed below in Fig.1 along with their functional necessities. The specifics of these stages can vary according to requirements.



Data collection is the process of gathering and analysing data on particular variables in a systematic way from a range of sources, such as real-time data from sensors or user behaviour, in order to answer to relevant queries and handle results. The data that has to be worked with comes in from a variety of data sources like customer activity, log data, network related data, IoT data etc. Traditional big data platforms, like Hadoop, fall short when it comes to processing this networked data. However, Apache Kafka, an open - sourced event - driven system developed by the Apache Software Foundation, is employed because it includes a massively scalable storage layer that can handle a high number of real-time data sources.

Data Transformation: Data cleaning and data transformation are the two major steps in data preprocessing. Data cleaning is the process of finding and recovering (or deleting) missing or inaccurate documents from a database, table, or collection of records. The primary aim is to identify and replace missing, inaccurate, untrustworthy, or trivial data in

data sets so that they may be replaced, updated, or deleted [11]. The process of changing a set of data values from one data system's data format to another data system's data format is known as data transformation [12]. The process of selecting, preparing, extracting, and altering data in order to permanently transfer it from one computer storage system to another is known as data migration. ETL operations, or Extract-Transform-Load operations, are a unique sort of data movement activity. The most optimal solution is to use the Databricks platform of different cloud services, which is based on Apache Spark, to extract data from legacy storage devices and move or copy it to Data Lakehouse. With the use of cluster computing, Apache Spark, another big data technology, is used to process the data. Spark executes operations much quicker than other big data platforms like Apache Hadoop. Also, Spark resolves the difficulty of processing huge and complicated interconnected data. It has a high level of fault tolerance. Spark Streaming, a basic of Apache Spark, is utilised to stream and analyse the data because enterprises are looking at real time data sources. The fundamental benefit of Spark over YARN (MapReduce on Hadoop) is that it allows for real-time processing. Only Batch processing is supported by YARN.

Data Storage: The process of uploading/storing data into a data storehouse, such as a Hadoop cluster using HDFS to store data across many nodes, or any NoSQL database, such as MongoDB, or other scalable databases, such as Cassandra, is referred to as data storage. Depending on the company's requirements, this approach varies significantly. In certain data warehouses, processed data is reformed on a regular basis, and accumulated data may be used to replace current data. Other data warehouses may upload additional data in a historical format on a regular basis—for example, every hour. The incoming streaming data can be temporarily stored in Data Factory from where it is ingested into blob storage in batches. Transformations and Actions are performed on this data from where depending on the data type it is then stored in SQL Data Warehouse or in a Data Lakehouse. A Data Lakehouse is really a new shared architecture that integrates the finest features of data lakes as well as warehouses into one system. It provides a ton of features like Transaction support, schema compliance to maintain data integrity, cloud services support, decoupling of storage from compute, ability to store a variety of data types and end to end streaming.

Data analysis is the process of analysing, cleansing, converting, and modelling data in order to extract useful information, make conclusions, and assist in decision-making. Data analysis encompasses a variety of approaches under a variety of titles in different commercial, academic, and social science areas [13].

3.2 Requirements of the architecture

Some of the other non-functional requirements of this architecture should be

Efficiency: The data processing and migration should run efficiently on any data size that might be ingested.
Validity: The data after processing should provide valid and unbiased results as per the requirements and for each of type use case – analytics / storage.
Availability: The user should be able to perform data processing from any location and at any time. In other words, the system must always be in proper functioning condition.
Uniformity: The software's user experience should be the same no matter what the underlying hardware configuration of the end-user's local system running the software.
Scalable: The data processing framework should be scalable to larger amounts of data with increasing number of partitions.

IV. METHODOLOGY

Real time data is incoming/streaming and enterprises can use Apache Kafke to ingest the data. The data is then temporarily stored in Data Lake using the Data Lake copy functionality. The data is then sent in batches to Databricks for data processing where the batches of data are partitioned into RDDs. In data processing, the required configuration such as storage method and persistence are read from the yaml file. Using the configurations, Spark transformations and actions are performed on the RDDs to process the data into the required format suitable for analytics or storage. These transformations have to be done on huge sets of data and are hence passed as Spark Jobs on the Databricks clusters where developers can monitor the performance using Monitors/ dashboards developed for this purpose. Data is then stored in the SQL Storehouse or Lakehouse depending on the structure of the data and requirements.

4.1 Stream ingestion

Fig 2 depicts a block diagram representing data processed within Stream Ingestion.

Big data solutions that function in real time operate on data which is in movement. This information is usually most valuable when it is first received. One may need to scale down resources if the inbound data stream gets too large to handle at that time. An HDInsight cluster, on the other hand, may scale up to match the streaming solution by adding nodes as needed.

One or more data sources generate events that must be digested quickly without losing any meaningful information in a streaming application. Stream buffering, also known as event queuing, is used to manage incoming events by a service like Apache Kafka or Event Hubs. You can then examine the data using just a real-time analytics platform inside the stream processing layer, such as Apache Storm or Apache Spark Streaming, after you've collected the events. The data can then be saved in protracted storage solutions like Data Lake Storage and shown in real time on a business analytics dashboard like Power BI, Tableau, or a customized web portal.

4.2 Data Processing

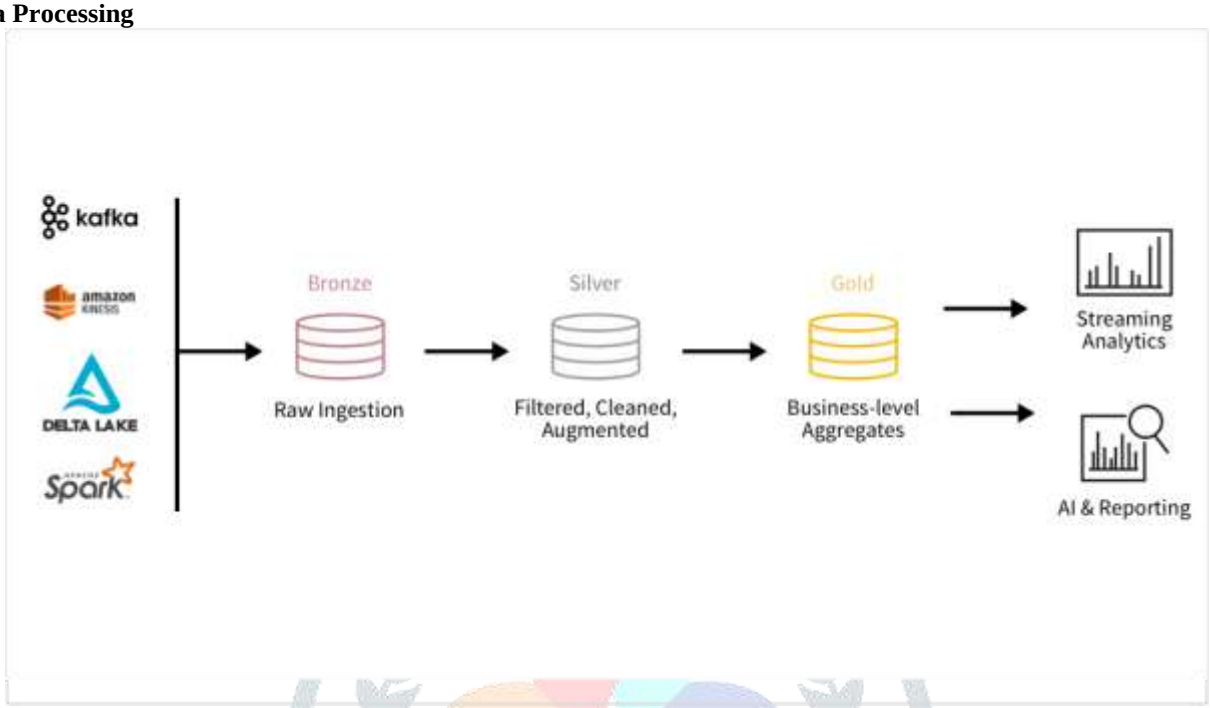


Fig 3 depicts the typical data processing stage

In Azure Factory, there are three crucial processes that include constructing an end-to-end pipeline that includes the Validation, Copy data, and Notebook operations.

Before you start the copy and analytics task, make sure your source dataset is suitable for downstream consumption. Copy data copies the source dataset to the sink storage, which in the Databricks notebook is mounted as DBFS. The dataset can then be ingested directly by Spark. The Databricks notebook that alters the dataset is triggered by the notebook. All of the transformation code is contained in this notebook. The dataset is also saved in a processed subfolder. The logic for the data transformation for the pipeline can be developed and applied to the raw data in order to filter it and produce the required transformations. With partial writes, failed tasks may damage and redundant data. Data integrity may be jeopardized if several data pipelines are reading and writing to your data lake at the same time. Delta Lake is an open source memory layer for any current data lake that keeps track of all data changes and provides dependability to Spark using timestamped Apache Parquet files and a transaction log. Diverse data pipelines may safely read and write data on the same table thanks to ACID transactions. Schema Enforcement guarantees that types of data are accurate and that necessary columns are present, whereas Schema Evolution enables these requirements to evolve as the data evolves. Using data processing architecture and clusters in place, users can now specify how they want their tasks to execute, such as batch jobs on a fixed schedule or Structured Streaming jobs that run constantly. These can be easily be integrated into the CI/CD pipeline and is also suitable for production and deployment.

4.3 Storage of data

The data is either kept in the SQL Storehouse, which makes it more accessible for business analytics reasons, or it is stored in the Data Lakehouse, depending on the requirement and the format of the structured data. The best of data warehouses and data linkages are combined in Data Lake House.

Many data pipelines in Lakehouse will be reading and writing data at the same time. Support for ACID transactions provides consistency when several parties, often using SQL, view or write data at the same time.

BI tools can be used immediately on the source data using Lakehouses. This decreases the price of having to integrate two copies of the data both in a data lake and a warehouse by reducing staleness and improving recency, reducing latency, and lowering the cost of having to incorporate two copies of the data.

The lakehouse could be used to store, refine, analyse, and access data kinds such as photos, video, audio, semi-structured data, and text, which are required for many emerging data applications.

V. EVALUATION METRICS AND PERFORMANCE

Monitoring is an important part of keeping Databricks workloads running in production. The first stage is to collect data and organise it in a workstation for analysis. Built in cloud analytics tools/ dashboards is the best strategy to combat log data. Although Databricks does not support transmitting log data to dashboards natively, a module for doing so is available on GitHub. This module allows one to log both Databricks service metrics and Apache Spark structure streaming query event data. A set of dashboards are then deployed as part of our production environment once this library has been successfully deployed to a Databricks cluster.A collection of time-series visualisations can be included in the log Analytics and dashboards. Each graph is a time-series visualisation of metric data for a Spark job, its phases, and the tasks that comprise each stage.

Job Latency: This graph depicts a job's execution delay, which is a rough representation of the job's overall performance. From start to finish, the job execution time is shown. The job start time and the job completion time are not the same. The percentiles of task execution referenced by cluster ID and application ID.

Task Latency: Task execution delay is depicted in this diagram. Latency is expressed as a percentile of job completion for each cluster, stage, and application.

Sum Task Execution per host: The total of task execution delay per host in a cluster is depicted in this diagram. When you look at task execution latency by host, you can see which hosts have much longer total task delay than the others. This might indicate that duties were assigned to hosts inefficiently or unequally.

Fig 3. performance comparison of Spark Jobs

Task metrics: This graphic depicts a set of execution metrics for a certain job. The size and timing of a data shuffle, the length of serialisation and deserialization processes, and other metrics are among them.

Cluster Throughput: This visualisation depicts the quantity of work completed per cluster and application using a high-level view of work items referenced by cluster and application. It displays the total number of jobs, tasks, and stages performed in one-minute increments for each cluster, application, and stage.

Throughput and Latency for Streaming: The metrics linked with a structured streaming query are the subject of this graphic. The graph depicts the number of rows supplied and processed per second. The application-specific streaming metrics are also displayed. These metrics are supplied whenever the OnQueryProgress event is fired while the structured streaming query is being performed, and the graphic depicts streaming latency as the time it takes to run a query batch in milliseconds.

Fig 4. Other performance metrics on such dashboards

VI. CONCLUSION

From the above sections it is evident to us that the data migration and data processing tasks are extremely complicated procedures and involve a huge amount of resources and computational power hence cloud resources are an hour ago to standard for any data architecture and solutions for companies. In this project we have mainly explored the data ingestion, data processing and data migration along with data storage using Azure cloud services to enhance scalability and to deal with the huge amounts of incoming streaming data. The performance related to spark jobs with different configuration has also been explored in this project as it is very essential in understanding the cost and efficiency. The data migration and data processing tasks are successful in achieving the required results and do not have any limitations per se but more experimentation can be done in optimising the performance even more by reducing the storage as well as the computational costs that are incurred during the course of the execution of the spark jobs on Databricks. The spark jobs developed for the given data requirements can be optimised to reduce the cloud computation cost and the data that that has been migrated can be further explored using the BI tools and better dashboards can be developed in order to make better business decisions while also maintaining security and privacy.

REFERENCES

[1]Ribeiro, A. Silva, and A. R. da Silva, Data Modeling and Data Analytics: A Survey from a Big Data Perspective, Journal of Software Engineering and Applications, 24-Dec-2015

[2]Hashem, Hadi and Daniel Ranc. An Integrative Modeling of BigData Processing. IJCSA 12 (2015): 1-15.

[3]Sai, B & Jyothi, Singaraju. (2015). A Study On Big Data Modeling Techniques. B. Sai Jyothi, S. Jyothi.

[4]Huda, M Misbachul & Rahma Latifa Hayun, Dian & Martun, Zhin. (2015). Data Modeling for Big Data. Jurnal Ultima InfoSys. 6. 1-11. 10.31937/si.v6i1.273. 2696

[5]Kaur, K., Rani, R. (2013). Modeling and querying data in NoSQL databases. Published in: Big Data, 2013 IEEE International Conference. INSPEC Accession Number 13999217.

[6]Bhadani, A., Jothimani, D. (2016), Big data: Challenges, opportunities and realities, In Singh, M.K., & Kumar, D.G. (Eds.), Effective Big Data Management and Opportunities for Implementation (pp. 1-24), Pennsylvania, USA, IGI Global

[7]H. Zhou, J. G. Lou, H. Zhang, H. Lin, H. Lin, and T. Qin, "An Empirical Study on Quality Issues of Production Big Data Platform," in 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering (ICSE), 2015, vol. 2, pp. 17–26.

[8]R. Han, L. Nie, M. M. Ghanem, and Y. Guo, "Elastic algorithms for guaranteeing quality monotonicity in big data mining," in 2013 IEEE International Conference on Big Data, 2013, pp. 45–50.

[9]Emmanuel and C. Stanier, "Defining Big Data," in Proceedings of the International Conference on Big Data and Advanced Wireless Technologies, New York, NY, USA, 2016

[10]Di Tria, F.; Lefons, E.; Tangorra, F. A Proposal of Methodology for Designing Big Data Warehouses. Preprints 2018, 2018060219 (doi: 10.20944/preprints201806.0219.v1).