International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-8 Issue-3, September 2019

Identifying Requirements for Big Data Analytics and Mapping to Hadoop Tools

Urmil Bharti, Deepali Bajaj, Anita Goel, S. C. Gupta

Abstract— Big data is being generating in a wide variety of formats at an exponential rate. Big data analytics deals with processing and analyzing voluminous data to provide useful insight for guided decision making. The traditional data storage and management tools are not well-equipped to handle big data and its application. Apache Hadoop is a popular open-source platform that supports storage and processing of extremely large datasets. For the purposes of big data analytics, Hadoop ecosystem provides a variety of tools. However, there is a need to select a tool that is best suited for a specific requirement of big data analytics. The tools have their own advantages and drawbacks over each other. Some of them have overlapping business use cases however they differ in critical functional areas. So, there is a need to consider the trade-offs between usability and suitability while selecting a tool from Hadoop ecosystem. This paper identifies the requirements for Big Data Analytics (BDA) and maps tools of the Hadoop framework that are best suited for them. For this, we have categorized Hadoop tools according to their functionality and usage. Different Hadoop tools are discussed from the users' perspective along with their pros and cons, if any. Also, for each identified category, comparison of Hadoop tools based on important parameters is presented. The tools have been thoroughly studied and analyzed based on their suitability for the different requirements of big data analytics. A mapping of big data analytics requirements to the Hadoop tools has been established for use by the data analysts and predictive modelers.

Keywords: Hadoop Ecosystem, BDA Life Cycle, Data Ingestion Tools, Data Processing Frameworks, Data Storage and Access Tools

I. INTRODUCTION

Big Data is voluminous data that is generated at an exponential rate in a variety of formats and has become difficult to tackle using traditional data management tools. The traditional data storage and management systems cannot finish data processing in time and does not meet the requirements of high availability, scalability and reliability for any big data application [1].

Manuscript published on 30 September 2019
* Correspondence Author
Ms. Urmil Bharti*, Assistant Professor in Department of Computer Science, Shaheed Rajguru College of Applied Sciences for women (University of Delhi).
Ms. Deepali Bajaj, Assistant Professor in Department of Computer Science, Shaheed Rajguru College of Applied Sciences for women (University of Delhi).
Dr. Anita Goel, Associate Professor in Department of Computer Science, Dyal Singh College, University of Delhi, India.
Dr SC Gupta, Faculty at Dept of Computer Science and Engineering, IIT Delhi.

Big data processing poses many challenges like capturing, storing, analysis, searching, visualization, querying and updating of data [2]. Following are the activities involved in any big data processing life cycle:

- Data Ingestion into the system
- Storing the data in persistent form
- Computing Data and Analyzing results
- Results Visualization

Data ingestion is the operation of taking raw data and making it available to the system. This step is very complex and depends on the format and speed of the incoming data. The ingestion process moves the data either to data storage or to processing components. Batch analytics is performed on data stored on disk and real time analytics is performed on data directly flowing to the processing components.

Data ingestion is a challenging task because of inherent attributes of big data i.e. velocity, volume, variety and veracity. Data processing is the most diversified element in the big data processing life cycle since the analytic requirements can differ significantly depending upon the type of insights needed. Visualizing the data is the last step of life cycle that is used to spot trends, patterns and correlations in data that might go undetected.

Several commercial as well as open source platform are available to perform different types of analytics on big data like IBM BigInsights, Oracle Big Data Appliance, SAP Hana tool, Pivotal Big data suite, Lumify, Hadoop RapidMiner etc. Apache Hadoop is a leading open-source framework that is used for storing, processing and analyzing extremely large volumes of data on commodity hardware machines. It allows processing of extremely large set of data in a distributed manner on cluster of nodes. Hadoop is the cost-sensible and scalable alternative to commercially available big data management packages. It has become an essential part of almost any big data solution that are commercially available and de-facto industry standard for BI (business intelligence) applications [3]. Hadoop is currently used by Google, Facebook, LinkedIn, Yahoo!, Twitter and many more.

Hadoop ecosystem includes various components of Apache Hadoop software library. It comprises of various accessories and tools provided by the Apache Software Foundation [4]. Depending on the type and mode of analysis of big data, Hadoop ecosystem offers a wide range of tools like Sqoop, Flume, Kafka, Hive, Pig, Mahout, Spark, HBase, Storm, S4 and Samza [5]. The tools perform specific functions and have advantages and limitations in different scenarios. Many tools have similar use cases but diverge in important areas. To perform data analytics, understanding of Hadoop tools is important.

Deciding the best suited tool is a challenge for researchers and professionals who understand big data but are novice to Hadoop environment for big data analysis.

In this paper, we identify major BDA requirement for any big data application based on big data life cycle. Further, we broadly divide these requirements into four major categories - Data Ingestion, Data Storage, Data Access and Data Processing as shown in Fig 1. The BDA requirement is then mapped with the most appropriate tool in Hadoop stack. Each tool is described from users' perspective for various BDA requirements.

The proposed requirement to tool mapping is further supported by delineating real world use case scenarios where these tools are currently used.

The goal of this paper is to facilitate novice Hadoop users in selection of best available tool from Hadoop ecosystem. It is intended for researchers, data scientist, data analysts and predictive modelers who need information on the distinct tools available in Hadoop stack for different BDA tasks.

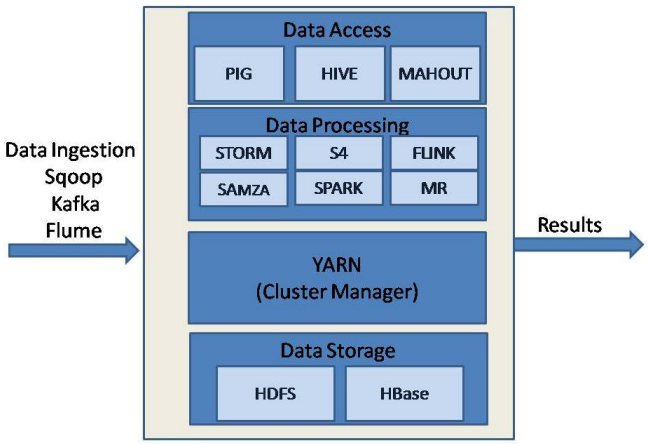

Fig 1: Hadoop Ecosystem

II. DATA INGESTION INTO HADOOP

Data ingestion is the mechanism of importing data either for processing or storing in a database [5]. Getting data into the Hadoop is very important task in any BDA application. Data to be ingested may come in different formats like structured, unstructured and semi-structured. Structured data is organized into fixed fields in a record or in a file. Data contained in relational databases or spreadsheets are examples of structured data. Data with no fixed format or form is known as unstructured data. Data sources containing a combination of text, images, videos etc. are typical examples of unstructured data. Data containing both forms of data i.e. structured and unstructured is called Semi-structured [6]. An XML file is an example of semi-structured data. Expert professionals of big data industry estimate that structured data contributes only 20%, rest 80% is unstructured data.

Data can be ingested in batches or streamed in real time. When it is ingested in batch mode, data-items are imported at periodic intervals as discrete chunks of time. Otherwise when data is ingested in real-time, each data-item is ingested as it is coming from the source.

For data ingestion, Apache provides multiple open source tools that can easily connect Hadoop to hundreds of different data sources like sensor and machine data, application logs, social media, geo-location data and point of sale data.

Following are the important data ingestion requirements for any BDA applications and their mapping to the most suitable tool available in Hadoop ecosystem.

BDA Requirement 1: To ingest structured data from any relational database

Apache Sqoop is an efficient tool used to import data into Hadoop storage. Sqoop (SQL-to-Hadoop) can be used for importing data from the Relational Database Management System like Teradata, Oracle, MySQL Server, PostgreSQL or any other JDBC compatible database into the HDFS, Hive or HBase. It can export data from HDFS to RDBMS as well. It can also ingest data from NoSQL databases i.e. MongoDB or Cassandra and hence allows direct data transfer to HDFS or Hive. It provides bulk import meaning it can import complete database into HDFS easily. Sqoop can also be forced to filter selectively by loading the required columns from source data in place of importing the entire input. Sqoop also supports parallel data transfer for optimal system resource utilization and gaining fast performance. This saves considerable amount of time [7].

Apollo Group Education Company makes use of Apache Sqoop to import data from external databases to HDFS and export results of Hadoop jobs back to RDBMS. Online marketer coupon.com applies Sqoop for transferring data between its IBM Netezza datawarehouse and Hadoop environment.

BDA Requirement 2: To ingest unstructured streaming data from multiple sources

Apache Flume is a distributed and highly reliable service used to collect and aggregate large amounts of streaming data from multiple servers into Hadoop environment [8]. Data from sensor and machine data, geo-location data, application logs and social media can be ingested to Hadoop using Flume. Flume allows users to ingest high throughput continuous stream of data to HDFS for persistent storage. Once streaming data is ingested in Hadoop, it can be used in further analysis using Apache HIVE interactive queries.

Goibibo uses Apache Flume to transfer their logs from the production system into HDFS. Firefox Mozilla uses Flume for BuildBot project along with ElasticSearch. Capillary technologies use Flume for collecting logs from 25 machines in production [9].

BDA Requirement 3: To ingest publish-subscribe message data, events streaming data, tracking and logging data

Apache Kafka is a popular publish-subscribe messaging system that processes and stores data as it passes through the system. Like other publish-subscribe messaging brokers, it enables different systems to broadcast and receive events without knowing source and destination. Kafka started as a project at LinkedIn to solve the problem of low-latency ingestion of large amounts of LinkedIn's event data in Hadoop.

Kafka has a few key advantages over other message brokers. It is a general purpose messaging system that is used to connect multiple systems [10]. While all message broker systems act as storage systems for messages that are in transit, Kafka writes data to disk and replicates it. So, probability of data loss is much lower while data is in transit, thus Kafka is an attractive option for

Retrieval Number C5524098319/2019©BEIESP
DOI: 10.35940/ijrte.C5524.098319
Journal Website: www.ijrte.org

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

applications that demand both speed and retention. Kafka can be used to build a user activity tracking pipeline as a set of real-time publish-subscribe feeds. In comparison to other messaging systems currently available, Kafka provides improved replication, throughput and fault tolerance that makes it a superior tool for message processing applications which handles data at a very large scale.

Uber, the Rider App, uses Kafka extensively. This app generates the cab-search request which in turn is logged to the kafka-pipelines. Modules calculating surge pricing consume the events through these pipelines. Sysomos (a Toronto based social media data Analytics Company) uses Kafka in the production system as well. Sysomos replaced their legacy sequential scheme of data backfilling with a producer-consumer based parallel solution implemented with Kafka. Square, credit card processing app, use Kafka as a channel to move all events data through their various datacenters which includes metrics, logs, custom events etc. [11].

In order to interact programmatically with Hadoop's file sytem, Hadoop provides multiple JAVA classes useful in manipulation of a file. These classes are available in Package named org.apache.hadoop.fs. Reading and writing unstructured batch data to/from Hadoop Distributed File System (HDFS) can be done by using these File System APIs through command-line-interface. This is one of the simplest ways to interact with HDFS. Table 1 compares above described data ingestion tools of Hadoop.

Table 1: Comparison of Data Ingestion Tools

Parameter	Apache Sqoop	Apache Flume	Apache Kafka
Real Time Analytics Support	Not supported	Supported	Supported
Integration with Hadoop	Specially designed for Hadoop Data	Specially designed for Hadoop Data	Not designed specifically for Hadoop, Supports multiple applications
Type of Data Transfer	Not event driven	Event driven	Event driven
Data Processing Capabilities	No Data Processing	Simple transformations using inceptors	No Data Processing
Message/ Data Size Limit	No restriction on data size	No restriction on message size	Designed for few KBs of message
Loading / Unloading of Data	Configuration tools available	Configuration tools available	Custom coding required
Data Replication	No	No	Yes
Data Loss	May occur while merging datasets	Possible	Very rare

III. DATA STORAGE IN HADOOP

Data storage is a key requirement of big data processing and analysis. BDA applications have placed extensive demand for storage due to increase in volume of data from gigabyte to zettabyte [6]. Thus there is a demand for a more organized storage. Due to huge amount of semi-structured and unstructured data, new storage technologies like NoSQL database and distributed file system are being used over the traditional storage technology. The new technologies of data storage and management satisfy core requirements of distributed storage, like, fault tolerance, scalability, high availability and reliability of data.

BDA Requirement 4: To store structured, semi-structured and unstructured data for batch analytics (sequential access)

Hadoop Distributed File System (HDFS) is the fundamental storage system for Apache Hadoop that provides extremely low cost per byte storage. It can store data in structured, unstructured, and semi-structured format. HDFS provides high-performance data access across Hadoop clusters deployed on commodity hardware machines [4].

HDFS is suitable for applications that have write-once-read-many access model. HDFS is efficient for sequential data access but doesn't provide the random read/write capability. A file once created and written in HDFS cannot be changed except for appends and truncates. A web crawler or a MapReduce application is best suited for HDFS.

HDFS emphasizes on high throughput of data access but with increased latency. HDFS is most suited for batch analytics that is capable of processing huge volumes of data in groups of transactions. A key drawback of HDFS is its inability to perform real-time analysis and interactive queries.

BDA Requirement 5: To store structured/semi structured data for real time analytics (random read/write access)

Apache HBase is a non-relational, NoSQL column-oriented database that runs on top of Hadoop and uses HDFS for underlying storage of data [12]. HBase is perfect choice for storing big tables and performing real-time analytics on it.

The key benefit of column-oriented database is that data is highly compressed. Also, columnar operations like SUM, COUNT, MIN, MAX and AVG are performed very fast. Another key benefit of column-oriented DBMSs is self-indexing hence it uses reduced disk space than any traditional RDBMS containing the same data. HBase provides very low latency access to small subset of data from very large datasets. It permits access to single row quickly from a table having billion of rows. Other important features of HBase are horizontal scalability, strong consistency and high availability [13]. Facebook's messaging platform that was built in 2010 uses HBase because of its high write throughput and low latency random reads features. HBase is used for real-time analytics like counting and storing "likes" for page/image/post [36]. Naxbio uses HBase to process massive amount of Human Genome Data. Following table compares above described data storage tools of Hadoop:

Table 2: Comparison of Data Storage Tools

Parameter	HDFS	HBase
Application Type	Java Based Distributed File system	Java Based NoSQL Database
Type of analysis	Batch	Real Time
Use Cases	Write Once Read Many	Random Read/ Write
Data Access Latency	High	Low
Data Storage Organization	Flat files	Key-value Pair
Other Dependency	No	Zookeeper

Dynamic Changes	Not allowed	Allowed
In-memory processing support	No	Yes
Processing of Sparse Datasets	Cannot handle	Can handle

IV. DATA ACCESSING INTO HADOOP

Data access defines what users and applications can do with data once it is stored in Hadoop. Analyst can process a huge volume of data in parallel as Hadoop not only supports distributed storage but distributed and parallel processing as well.

Apache Hadoop provides a variety data analytics tools for varied big data applications.

BDA Requirement 6: To access structured/ unstructured data using scripts for scalable analytics

Apache Pig is a scripting platform used to analyze large data sets [14]. It is a procedural data flow language used to process datasets like weblogs, streaming online data etc. by representing them as data flows. Pig works with many data sources like structured or unstructured and stores the results back in HDFS. Pig was designed to make MapReduce applications easier. Before Pig, MapReduce applications were written in Java. MapReduce programming model was not a convenient way for data analyst. So Apache Pig was built on top of Hadoop as an abstraction of MapReduce [38]. Development cycle of MapReduce applications used to be very long and programmers need to be proficient in Java [4]. Pig was designed to be extensible i.e. loading, storing, grouping, filtering and joining can be implemented using user defined functions (UDFs).

Pig has its own scripting language, PigLatin, which can be used for operations like Extract, Transform and Load (ETL), ad-hoc data analysis and iterative data processing. PigLatin helps the programmer to write very short scripts. This increases programmers' productivity and reduces development time. It has been observed that in one test, 200 lines of Java code can be replaced by 10 lines of PigLatin script. In terms of time, if it takes 4 hours to write in Java, then Pig scripts can be developed in only 15 minutes, approximately. Thus Pig opens Hadoop for non-Java programmers. There are few data processing scenarios where Pig may not work well. A Pig query will not perform well if it accesses only a small fraction of data from a large data set since it is designed to scan either whole dataset, or at least a large portion of it.

Apache pig can be used where data processing is needed by search engines. Yahoo! makes use of Pig in 40% of its jobs like news feeds and searching. It can also be used for time-sensitive data where data should be extracted, loaded and analyzed rapidly. Twitter use Pig to extract customer activities and analyze this data to uncover hidden patterns in customer behaviors and suggest appropriate recommendations immediately e.g. trending tweets.

BDA Requirement 7: To access structured data using SQL-based language for scalable analytics

Apache Hive is a data warehouse framework for data summarization, ad-hoc querying and analysis of large volumes of data stored in HDFS [15]. The query language used by Hive is known as Hive-QL and it provides SQL like interface. Hive facilitates analysts with strong SQL skills to execute queries on large volume of data stored in HDFS.

Hive is mainly designed for online analytical processing (OLAP). Hive is best used for batch jobs over large sets of immutable data collected over a period such as web logs to calculate trends [38]. It is not suited for real-time queries, online transaction processing (OLTP) and row level updates. hi5, a social networking site, use Hive for analytics, machine learning and social graph analysis. Chitika, online advertising company, use Hive for data mining and analysis of their 435 million monthly global users [16]. Top companies using Hive are Tata Consultancy Services, HortonworksInc, Facebook, Netflix and Qubole In. Table 3 compares above described data access tools of Hadoop.

Table 3: Comparison of Data Access Tools

Parameter	Pig	Hive
Data format	Structured, Unstructured, Semi-structured	Structured
UDF support	Easy	Difficult
Language used	Pig Latin – relational data flow language	Hive Query language(HQL) – similar to SQL
Use case	ETL applications	Ad-hoc queries, data summarization, reports
End users	Programmers, software developers and researchers	Big data analysts
Language type	Procedural dataflow language	Declarative SQL language
Avro File format	Supported	Not supported
Metadata Support	Not supported	Stores metadata in Darby, MySQL, Oracle
Data Updates	Supported	Not supported
Complex queries having many join and filter	Recommended	Non recommended

BDA Requirement 8: To access unstructured data for predictive analysis in scalable machine learning applications

Apache Mahout is a software library of scalable machine learning algorithms, implemented on the top of Apache Hadoop, once data is stored on HDFS [17]. Mahout offers data science tools to find meaningful patterns in those extremely large datasets automatically. Few important use case scenarios of Mahout include collaborative filtering (CF), classification, clustering and mining of frequent item sets.

Organizations such as Adobe, LinkedIn, Foursquare, Facebook, Twitter and Yahoo! use Apache Mahout internally. *Foursquare,* a social networking service helps its users in finding food, places and entertainment available in a location by using Mahout's recommender engine. Twitter also uses Mahout for user interest modeling and Yahoo! uses it for pattern mining. Amazon mines user behavior and makes product recommendations using collaborative filtering.

V. DATA PROCESSING IN HADOOP

Processing frameworks are the most essential components of any big data system. The goal of processing framework is to operate over data to provide better understanding, view into patterns and gain insights. Processing frameworks compute over data either by reading from non-volatile storage or streaming data. The frameworks are grouped by the state of data they are designed to handle - data in batches, data in a continuous stream and both kinds of data.

The datasets in batch processing are typically bounded (finite collection of data), persistent (data stored on a permanent storage) and extremely large. In fact, batch operations are mostly the only choice for processing extremely huge sets of persistent data so this type of processing is frequently used with historical data. Batch processing is applicable where calculations are required to perform on a complete set of data records. For example, for calculating averages and totals, datasets must be taken completely rather than using its subset. Handling of such large volume of data takes prolonged computation time. Thus, batch processing is not appropriate in situations where processing time is significant [39]. Stream processing systems process data as it gets into the system [18]. It needs a different processing framework than batch model. Stream processors define operations which are applied to each data item as it goes into the system, rather than defining them to apply to an entire dataset as in batch processing. In stream processing, datasets are "unbounded" which indicates the amount of data that has entered into the system so far. In comparison to total dataset, working dataset is more relevant as it is limited to a set of items at an instance of time. Stream processing is a continuous process and it does not finish until stopped explicitly. Results of stream processing are readily accessible and will be repeatedly updated as new stream of data enter the system [40]. These systems can process nearly unbounded amount of data. Processing can be either true stream processing that processes one data item or micro-batch processing where few items are processed at a time. In micro-batch processing minimum state is preserved between different records. Applications which demand near real-time processing are best handled by the streaming models [19]. Machine data captured by sensors connected to internet of things (IoT), server/ application log error, and other time-based metric are some examples where near-real time processing is required. Stream processing helps an analyst to react to changing trends or sudden spikes which may occur over time [20]. Apache Hadoop caters to various data processing requirements like the batch processing using MapReduce, stream data processing using Apache Storm, Apache Samza and Apache S4. Hadoop also includes hybrid processing engines like Apache Spark and Apache Flink that perform both batch processing and stream processing of data.

BDA Requirement 9: Low cost batch processing of very large data sets with no processing time constraints

MapReduce paradigm of Hadoop is the core of distributed programming model for big data applications [21]. MapReduce is the original framework based on Java for writing applications that process large amounts of structured and unstructured data stored in HDFS. MapReduce framework processes very large volumes of batch data in a split-apply-combine manner. In this strategy, analyst break up a big problem into manageable pieces, mapper operate on each piece independently and then reducer put all the pieces back together. In map-reduce, the mapper corresponds to split and apply, and reducer corresponds to combine [22].

Once a MapReduce application is written, scaling it to run on thousands of machines within a cluster is simply a configuration change. This incredible scalability potential attracts many Java programmers to use MapReduce model to process vast amount of data in parallel. This framework heavily uses permanent storage i.e. reading and writing multiple times for a task, which makes it fairly slow. Since disk space is typically one of the most abundant server resources and MapReduce can typically run on less expensive hardware as it does not exploit in-memory processing. Steep learning curve of MapReduce is its major drawback since complex business logic needs to be written in the form of mapper and reducer functions [38].

There are many challenging issues such as log and data analytics, fraud detection, recommendation engines and user behavior analysis where MapReduce is used as an optimal solution. Millions of user access sites daily such as Twitter, Facebook and LinkedIn to connect with their friends, colleagues and community [23]. These sites uses MapReduce model for solving problems such as finding common friends/followers on Facebook/Twitter or common connections in LinkedIn between any two users. Many interesting features such as visitors profile tracking on LinkedIn and readers count on Facebook or Twitter's post can be efficiently computed using the MapReduce model.

BDA Requirement 10: Stream processing for near real-time analytics with extremely low latency and exactly-once /at-least-once processing semantics (Guaranteed processing)

Apache Storm is a stream processing framework which meets the requirement of extremely low latency and considered as one of the best options for near real-time processing workloads. Storm initiated the notion of stream as a distributed abstraction of data in motion with characteristics of fault tolerance and reliability that was difficult to realize using traditional methods used for real time analysis. Strom can process very huge volume of data and produces output very quickly as compared to other existing stream processing solutions. Apache Storm and Apache Kafka naturally blend with each other, and their dominant cooperation facilitates real time streaming analytics for fast-moving big data. Storm provides at-least-once guarantee of processing, meaning each message will be processed at least once, though some duplicates may occur in some failure situations [24]. Storm does not guarantee in-order processing of messages. Storm is often considered a good selection when processing time directly affects user experience, for example when providing feedback directly to the visitor's page on a website. Storm also provides abstraction layer called Trident which provides exactly once processing. Storm without this abstraction layer is called as Core Storm. Trident implements a micro batching model in contrast to item-by-item processing i.e. pure streaming system in Core Strom.

Trident's ensures that items are processed exactly once, and hence making it effective in those systems that does not have intelligence to manage duplicate messages. When there is a requirement to preserve state between different data items for example counting users' click in a predefined time interval then Trident is the only choice within Storm. It is a great tool for applications that necessitate real time analysis, machine learning and continuous monitoring on operations [25]. Yahoo! JAPAN, a dominant web portal, uses Apache Storm. In this portal Storm applications are used to process various streaming data such as social media data or logs. Storm is used to recognize trending topics, monitor systems and crawl on websites. Spotify, a digital music service, serves streaming music to nearly ten million subscribers and forty million active users. Apache Storm is used here to provide real time facilities like music recommendation, ads targeting and monitoring [26]. Verisign, a universal leader in Internet security and domain names, uses Storm to analyze network telemetry data of globally distributed infrastructure to detect and mitigate cyber attacks.

BDA Requirement 11: Stream processing for real-time analytics with low latency and at-least-once processing semantics

Apache Samza is a stream processing framework that is very tightly coupled with Apache Kafka messaging system. Samza is specially designed for Kafka and it takes full advantage of Kafka's unique architecture, fault tolerance, buffering, and state storage. Samza's strong association to Kafka permits loosely coupled stream processing. Any number of subscribers can be augmented at any step without preexisting coordination. This can be very helpful for scenarios where multiple teams of an organization require accessing related data. Topics of data which is entering into the system can be subscribed by different teams. Topics created and used by other teams can also be subscribed.

Samza provides fault-tolerant processing of streams i.e. Samza makes sure that messages would not lost, even a job crashes, or a machine goes dead, or in case of a network fault, or something else goes erroneous [27]. Prerequisites of input system implemented with Samza are

- The input stream may be sharded into one or more partition.
- Messages in each partition are sequenced in a fixed order and will always be consumed in the same sequence.
- A Samza job can consume messages from any starting offset in a partition.

Samza guarantees at-least-once processing semantics. Hence it ensures that analytical job doesn't miss any message, even if job's container restarted in case of any failure. In this scenario the same message can be processed more than once but duplication can be reduced by check pointing more frequently, at a slight performance cost. Samza only supports JVM languages like Scala, Clojure and Jython etc. presently. Apache Samza is a good option for streaming workloads where either, Hadoop and Kafka are already available or can be implemented. Samza presents low latency performance. It might not be a good fit if existing system isn't compatible with available deployment requirements.

Samza was originally developed at LinkedIn. It is currently used to process tracking data, service log data, and for data ingestion pipelines for real time services. Fortscale uses Samza to processes logs of security events that comes as a part of data ingestion pipelines and develops on-line machine learning models. TripAdvisor, a travel and restaurant website that provides hotel/restaurant reviews and bookings, makes use of Samza to process billions of daily events for business analytics, machine learning and website improvement [28].

BDA Requirement 12: Process multiple event streams and extract information in real time with low latency

Apache S4 is Simple Scalable Streaming System. It is a distributed stream processing engine based on MapReduce model [29]. It is a general-purpose, near real-time, event-driven and high-performance modular platform. S4 is partially fault tolerant platform which allows programmers to easily develop applications that requires continuous processing of unbounded streams of data. S4 hides processing complexity from its users. This engine has been designed to solve real world problems in the context of search applications that use machine learning and data mining algorithms [30]. Some important use cases of S4 are personalized search, twitter trends, online parameter optimization, spam filtering, and predict market prices for automatic trading and network intrusion detection. In comparison to Storm that provides basic tools to build the framework for desired real time analytics, S4 offers a well defined ready to use framework. Table 4 compares steam data processing tools of Hadoop.

Table 4: Comparison of Steam Data Processing Tools

Parameter	Apache Storm	Apache Samza	Apache S4
Framework	Microbatching, Stream Processing	Pluggable Apache Kafka, YARN	Actor Programming Model
Processing semantics	At least once	At least once	*
Stream Primitives	Tuple	Sequence of Messages	Events
Latency	Sub-second	Low millisecond	Few millisecond
Fault tolerance	Guaranteed delivery as tuples are reprocessed	Each job is recovered and restarted independently	New node starts from snapshot
Stream source	Spout	Message queuing system	Network
Computation Unit	Bolts	Broker	Processing elements

*Lack of reliable data delivery

BDA Requirement 13: Heavy batch processing workloads that value throughput over latency with some stream-oriented tasks

Apache Spark is the advanced batch processing framework with capabilities of stream data processing. It is a high speed in-memory data processing framework that aims to execute streaming, machine learning or interactive workloads efficiently. Such use cases require fast iterative access to datasets. Fast data processing capabilities and developer's convenience have made Apache Spark a strong contender for big data computations [31].

Spark can be deployed either as a standalone cluster or can be attached to Hadoop as an alternative to the MapReduce framework. It means, Spark runs on Apache Hadoop, Apache Mesos, standalone, or in the cloud. It can access diverse data sources like HDFS, Hbase, Cassandra and Amazon S3.

Spark uses a fundamental data structure called Resilient Distributed Datasets, or RDDs, that are immutable structures that represent collections of data available in memory.

Each RDD can be linked to its parent RDD and finally to the data stored on the disk hence Spark provides fault tolerant capability without writing results back to disk after each and every operation [32]. Spark powers a stack of libraries including Spark Streaming for streaming data applications, Spark SQL for interactive queries, MLlib for machine learning and GraphX for graph processing. Spark streaming implements a concept called micro-batches which treats data streams as a chain of very small batches that can be handled using the elementary operations of batch processing. As the RAM is more expensive than disk space, Spark cost more to run applications in contrast to other batch processing engines that are disk-based. However, increased processing speed makes task to get completed much faster. Apache Spark uses micro-batches for all types of workloads. It is not satisfactory for those use cases where we have to process very large streams of live data and provide real time results [37].

In real time, Apache Spark is used in many notable business industries such as Oracle, Hortonworks, Cisco, Verizon, Visa, Microsoft, Databricks, Amazon, Uber, Pinterest, eBay and Yahoo! etc. These companies collect terabytes of event data from its users and engage them with real-time interactions such as video streaming and many other user interfaces. Thus, Spark facilitates in maintaining the constantly smooth and high-quality user experience.

BDA Requirement 14: Heavy stream processing workloads with some batch-oriented tasks

Apache Flink is a distributed stream data processing framework which is capable of handling batch processing also[33]. Flink accurately processes streaming data which is coming out-of-order. Flink guarantees exactly-once processing semantics using checkpointing mechanism for stateful computations hence eliminates duplication. Stateful means that an applications maintains an aggregation or summary of data that has been processed over the period of time. Light weight fault tolerance mechanism of Flink results in high throughput rates and at the same time it provides strong consistency guarantee as well.

Apache Flink uses streams for batch workload as well. Batch data is a finite collection of streamed data items with fixed boundaries. So, batch processing is treated as a subset of stream data processing. It can emulate batch processing, however at its core it is a native streaming processing engine. Flink salient features are exactly once fault management, high throughput, automated memory management, low latency, real entry-by-entry processing and advanced streaming capabilities (such as windowing features, etc) [34].

Flink also provides a web-based scheduling view to track tasks and monitor the system. This view also displays optimized plan for each submitted tasks. For big data analysis tasks, Flink offers SQL-style querying, graph processing, machine learning libraries, and in-memory computation. Overall, holistic performance of Flink is outstanding as compared to other data processing systems. It employs existing closed-loop iterative operators that make faster graph processing and machine learning workloads. Flink processes data at lightning speed.

King, the designer of candy crush saga game, makes real time analytics available to its data scientists via a Flink-enabled platform, which significantly reduces the time to gain insights from game data. Ericsson uses Flink to develop a real-time anomaly detector over huge infrastructures using machine learning algorithms. Bettercloud, which is a multi-SaaS management platform, employs Flink to find near real-time intelligence from SaaS application activity [35].

Following table compares above described steam data processing tools of Hadoop:

Table 5: Comparison of Hybrid Data Processing Tools

Parameter	Apache Spark	Apache Flink
Framework	Micro Batching	True complex stream processing
Memory Management	Configurable	Automatic
Stream Primitive	RDD	Rows after rows of real time data
Data Flow	Directed Acyclic Graph	Cyclic Dependency Graph
Data processing	Microbatches	Continuous flow
Performance	Not efficient	Excellent
Latency	High	Low
Processing Guarantee	Only once	Only once

Fig 2 shows consolidated diagram of various tool in Hadoop ecosystem and their mapping to various BDA requirements.

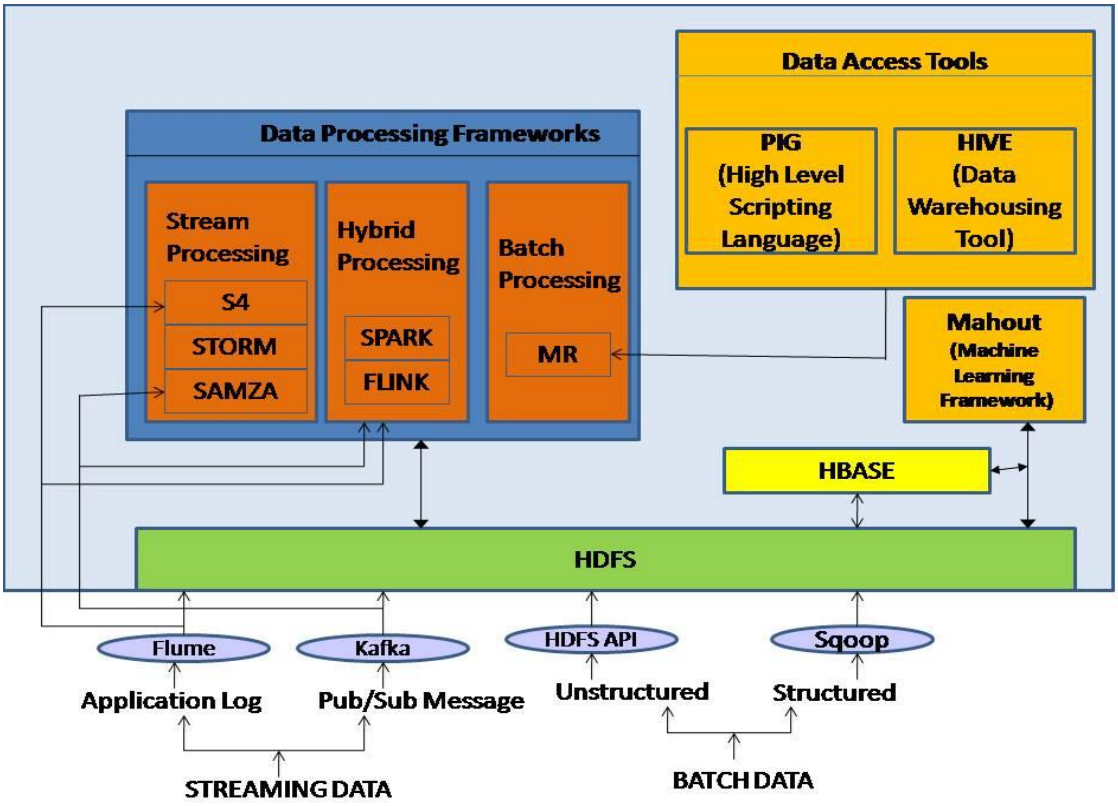

Fig 2: Tools of Hadoop Ecosystem mapped to BDA Requirements

VI. Conclusion

With an ever-increasing number of tools available, the task of selecting right tool for big data analysis is a challenge. The tools available in Hadoop ecosystem have their own advantages and limitations for the different Big Data Analytics requirements. Many Hadoop tools have similar core functionality but distinct features, for a single business requirement. In order to evaluate among different available tool options, there is a need to understand the tools well. In this paper, we have identified main BDA requirements based on big data lifecycle. These requirements have been further categorized into four main classes i.e. data ingestion, data storage, data access and data processing. BDA requirements of each class are mapped to the most relevant tool of Hadoop ecosystem. Each tool has been described in detail to justify its mapping to the stated requirement along with its current use in industry. Feature based comparison of tools for each identified class is presented in a tabular form for easy reference and evaluation. This paper aids the data scientists, researchers and novel industry professionals to select the best fitted tool from Hadoop ecosystem for various BDA requirements.

Future Scope

Hadoop is one of the popular big data technologies and has an extensive and boundless scope in near future. Today, BDA and machine learning (ML) have become rudimentary in business intelligence and being exponentially used as a fundamental part of business strategies for organizations. Hadoop will continue to be used and evolved in emerging requirements of pervasive areas like data science, IoT and machine learning. Hadoop-as-a-service in the cloud makes it possible to run large scale applications in highly elastic and scalable environment.

References

1. Zikopoulos, P., & Eaton, C. (2011). *Understanding big data: Analytics for enterprise class hadoop and streaming data*. McGraw-Hill Osborne Media.
2. https://www.digitalocean.com/community/tutorials/an-introduction-to-big-data-concepts-and-terminology, accessed on August 28, 2018
3. https://www.marketanalysis.com/?p=279 Hadoop Market Forecast, accessed on September 9, 2019
4. White, T. (2012). *Hadoop: The definitive guide*. " O'Reilly Media, Inc.".
5. Demchenko, Y., De Laat, C., & Membrey, P. (2014, May). Defining architecture components of the Big Data Ecosystem. In *Collaboration Technologies and Systems (CTS), 2014 International Conference on* (pp. 104-112). IEEE
6. https://www.smartdatacollective.com/quick-guide-structured-and-unstructured-data/, accessed on December 2, 2018
7. Chen, L., Ko, J., & Yeo, J. (2015). Analysis of the influence factors of data loading performance using Apache Sqoop. *KIPS Transactions on Software and Data Engineering*, *4*(2), 77-82
8. Hoffman, S. (2015). *Apache flume: Distributed log collection for hadoop*. Packt Publishing Ltd
9. https://www.dezyre.com/article/sqoop-vs-flume-battle-of-the-hadoop-etl-tools-/176
10. Dunning, T., & Friedman, E. (2016). *Streaming Architecture: New Designs Using Apache Kafka and MapR Streams*. " O'Reilly Media, Inc.".
11. https://cwiki.apache.org/confluence/display/KAFKA/Powered+By, accessed on January 17, 2018
12. Naresh, P., Shekhar, G. N., Kumar, M. K., & Rajyalakshmi, P. (2017). Implementation of Multi-node Clusters in Column Oriented Database using HDFS. *Empirical Research Press Ltd.*, 186.
13. Vora, M. N. (2011, December). Hadoop-HBase for large-scale data. In *Computer science and network technology (ICCSNT), 2011 international conference on* (Vol. 1, pp. 601-605). IEEE.

14. Schumacher, A., Pireddu, L., Niemenmaa, M., Kallio, A., Korpelainen, E., Zanetti, G., & Heljanko, K. (2013). SeqPig: simple and scalable scripting for large sequencing data sets in Hadoop. *Bioinformatics*, *30*(1), 119-120.
15. Thusoo, A., Sarma, J. S., Jain, N., Shao, Z., Chakka, P., Anthony, S., ... & Murthy, R. (2009). Hive: a warehousing solution over a map-reduce framework. *Proceedings of the VLDB Endowment*, *2*(2), 1626-1629.
16. https://wiki.apache.org/hadoop/Hive/PoweredBy, accessed on March 31, 2018
17. Ingersoll, G. (2009). Introducing apache mahout. *IBM developerWorks Technical Library*.
18. Cherniack, M., Balakrishnan, H., Balazinska, M., Carney, D., Cetintemel, U., Xing, Y., & Zdonik, S. B. (2003, January). Scalable Distributed Stream Processing. In *CIDR* (Vol. 3, pp. 257-268).
19. Stonebraker, M., Çetintemel, U., & Zdonik, S. (2005). The 8 requirements of real-time stream processing. *ACM SIGMOD Record*, *34*(4), 42-47.
20. Zaharia, M., Das, T., Li, H., Shenker, S., & Stoica, I. (2012). Discretized Streams: An Efficient and Fault-Tolerant Model for Stream Processing on Large Clusters. *HotCloud*, *12*, 10-10.
21. Dean, J., & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. *Communications of the ACM*, *51*(1), 107-113.
22. Wickham, H. (2011). The split-apply-combine strategy for data analysis. *Journal of Statistical Software*, *40*(1), 1-29.
23. Sumbaly, R., Kreps, J., & Shah, S. (2013, June). The big data ecosystem at linkedin. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data* (pp. 1125-1134). ACM.
24. Van der Veen, J. S., van der Waaij, B., Lazovik, E., Wijbrandi, W., & Meijer, R. J. (2015, March). Dynamically scaling apache storm for the analysis of streaming data. In *Big Data Computing Service and Applications (BigDataService), 2015 IEEE First International Conference on* (pp. 154-161). IEEE.
25. Toshniwal, A., Taneja, S., Shukla, A., Ramasamy, K., Patel, J. M., Kulkarni, S., ... & Bhagat, N. (2014, June). Storm@ twitter. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data* (pp. 147-156). ACM.
26. http://storm.apache.org/Powered-By.html, accessed on July 2, 2018
27. https://samza.apache.org/learn/documentation/0.13/container/checkpointing.html, accessed on Setember 16, 2018
28. https://cwiki.apache.org/confluence/display/SAMZA/Powered+By, accessed on March 13, 2019
29. Neumeyer, L., Robbins, B., Nair, A., & Kesari, A. (2010, December). S4: Distributed stream computing platform. In *Data Mining Workshops (ICDMW), 2010 IEEE International Conference on* (pp. 170-177). IEEE.
30. Bifet, A. (2013). Mining big data in real time. *Informatica*, *37*(1).
31. García-Gil, D., Ramírez-Gallego, S., García, S., & Herrera, F. (2017). A comparison on scalability for batch big data processing on Apache Spark and Apache Flink. *Big Data Analytics*, *2*(1), 1.
32. Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., ... & Ghodsi, A. (2016). Apache Spark: A unified engine for big data processing. *Communications of the ACM*, *59*(11), 56-65.
33. Carbone, P., Katsifodimos, A., Ewen, S., Markl, V., Haridi, S., & Tzoumas, K. (2015). Apache flink: Stream and batch processing in a single engine. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, *36*(4).
34. Friedman, E., & Tzoumas, K. (2016). Introduction to Apache Flink: Stream Processing for Real Time and Beyond
35. https://flink.apache.org/poweredby.html, accessed on May 11, 2019
36. Borthakur, D., Gray, J., Sarma, J. S., Muthukkaruppan, K., Spiegelberg, N., Kuang, H., ... & Schmidt, R. (2011, June). Apache Hadoop goes realtime at Facebook. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data* (pp. 1071-1080). ACM.
37. Khan, I., Naqvi, S. K., Alam, M., & Rizvi, S. N. A. (2017). An efficient framework for real-time tweet classification. *International Journal of Information Technology*, *9*(2), 215-221.
38. Kumar, J. S., Raghavendra, B. K., & Raghavendra, S. (2019). Big data Processing Comparison using Pig and Hive.
39. Ajah, I. A., & Nweke, H. F. (2019). Big Data and Business Analytics: Trends, Platforms, Success Factors and Applications. Big Data and Cognitive Computing, 3(2), 32.
40. Kolajo, T., Daramola, O., & Adebiyi, A. (2019). Big data stream analysis: a systematic literature review. Journal of Big Data, 6(1), 47.

AUTHORS PROFILE

Ms. Urmil Bharti has over 13 years of teaching experience as Assistant Professor in Department of Computer Science, Shaheed Rajguru College of Applied Sciences for women (University of Delhi). Earlier she has more than 10 years of industry experience. Her last designation was Senior Quality Analyst. She is currently doing her research in the area of Cloud and Distributed Computing. Her key research area is open source serverless frameworks. She has authored several national and international research publications.

Ms. Deepali Bajaj has over 13 years of teaching experience as Assistant Professor in Department of Computer Science, Shaheed Rajguru College of Applied Sciences for women (University of Delhi). She is currently doing her research in the area of Cloud and Distributied Computing. Her key research areas are Platform as a service (PaaS) and Function as a service (FaaS) of serverless technology. She has authored several national and international research publications.

Dr. Anita Goel is an Associate Professor in Department of Computer Science, Dyal Singh College, University of Delhi, India. She has received her Ph.D. in Computer Science and Masters in Computer Applications from Jamia Millia Islamia and Department of Computer Science (University of Delhi), respectively. She has a work experience of more than 30 years. She is a visiting faculty to Delhi Technological University and NIIT University. From 2009-10, she was Fellow in Computer Science, at Institute of Life Long Learning (ILLL) in University of Delhi. She has served as member of program committee of International conferences like IEEE BigData Congress 2015 and ICWI 2015. She has guided several students for their doctoral studies and has travelled internationally to present research papers. She has authored books in Computer Science and has several national and international research publications.

Dr SC Gupta is B.Tech (EE) from IIT Delhi and has worked at Computer Group at Tata Institute of Fundamental Research and NCSDCT (now C-DAC Mumbai), Till recently, he worked as Deputy Director General, Scientist-G and Head of Training at National Informatics Centre, New Delhi and was responsible for keeping its 3000 scientists/ engineers uptodate in various technologies. He has extensive experience in design and development of large Complex Software Systems. Currently he is a Visiting Faculty at Dept of Computer Science and Engineering, IIT Delhi. His research interests includes Software Engineering, Data Bases and Cloud Computing. He has been teaching Cloud Computing at IIT Delhi, which includes emerging disruptive technologies like SDN and SDS. He has guided many M.Tech & PhD Research students in these technologies and has many publications in Software Engineering and Cloud Technology in National and International Conferences and Journals.