LAB MANUAL
PART A
(PART A: TO BE REFERRED BY STUDENTS)

Experiment No-01

A.1 Aim:

To Study of Hadoop system

A-2 Prerequisite

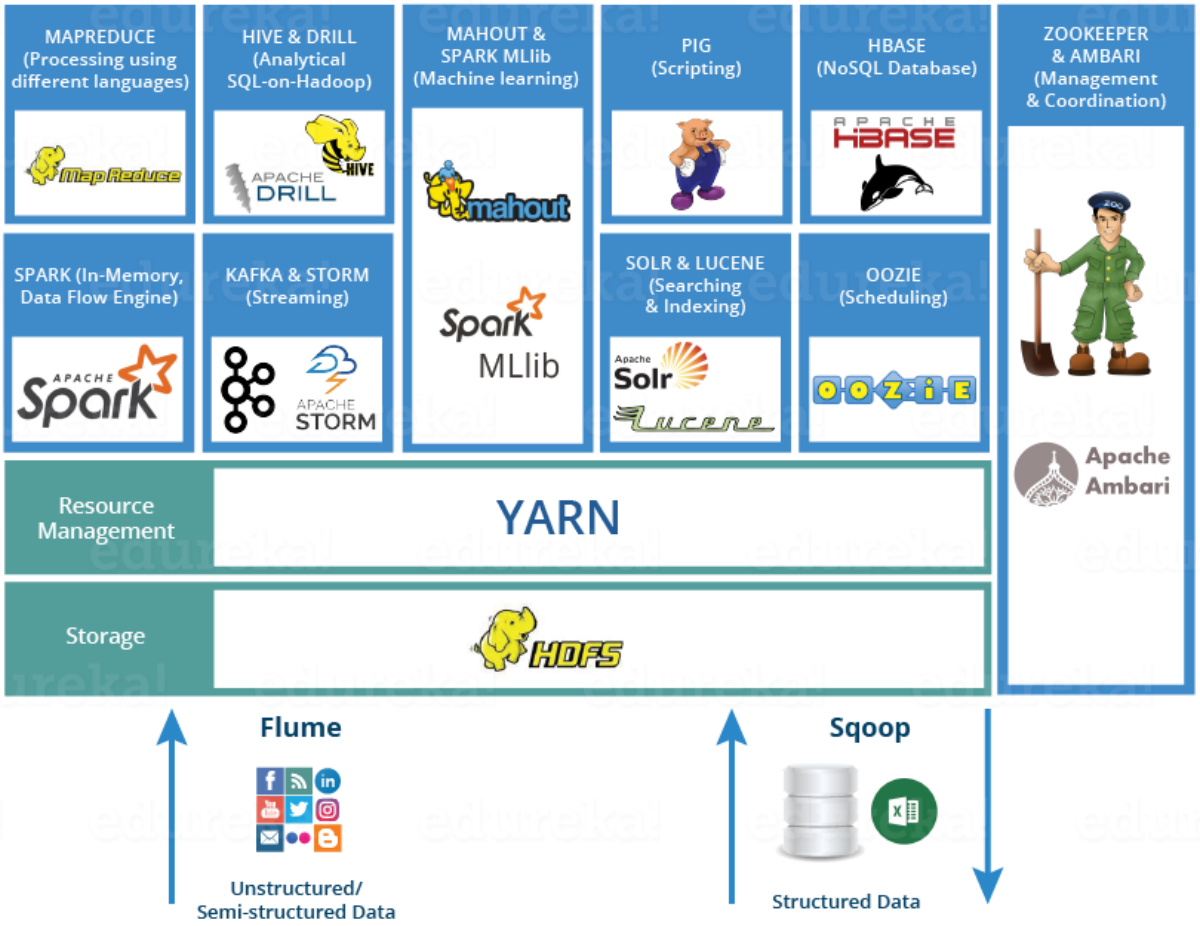
Knowledge of databases and Computer systems.

A.3 OutCome

Students will be capable to define key issues in big data management and its associated applications in intelligent business and scientific computing

A.4 Theory:

- **HADOOP ECOSYSTEM:**
  - Hadoop Ecosystem is neither a programming language nor a service, it is a platform or framework which solves big data problems. It is considered as a suite that encompasses several services (ingesting, storing, analyzing and maintaining) inside it.
  - Below are the Hadoop components, that together form a Hadoop ecosystem, I will be covering each of them in this blog:
    - **HDFS** ->*Hadoop Distributed File System*
    - **YARN**-> *Yet Another Resource Negotiator*
    - **MapReduce**->*Data processing using programming*
    - **Spark**-> In-memory Data Processing
    - **PIG, HIVE**->*Data Processing Services using Query (SQL-like)*
    - **HBase**->*NoSQL Database*
    - **Mahout, Spark MLlib** ->*Machine Learning*
    - **Apache Drill** ->*SQL on Hadoop*
    - **Zookeeper**->*Managing Cluster*
    - **Oozie**->*Job Scheduling*
    - **Flume, Sqoop** ->*Data Ingesting Services*
    - **Solr& Lucene** ->*Searching & Indexing*
    - **Ambari**->*Provision, Monitor and Maintain cluster*


MAPREDUCE
(Processing using different languages)
HIVE & DRILL
(Analytical SQL-on-Hadoop)
MAHOUT & SPARK MLlib
(Machine learning)
PIG
(Scripting)
HBASE
(NoSQL Database)
ZOOKEEPER & AMBARI
(Management & Coordination)
SPARK (In-Memory, Data Flow Engine)
KAFKA & STORM
(Streaming)
SOLR & LUCENE
(Searching & Indexing)
OOZIE
(Scheduling)
Resource Management
YARN
Storage
Flume
Unstructured/
Semi-structured Data
Sqoop
Structured Data

PART B

(PART B: TO BE COMPLETED BY STUDENTS)

(Students must submit the soft copy as per the following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case there is no Blackboard access available)

Roll. No. 50	Name: AMEY MAHENDRA THAKUR
Class: COMPS-BE-B	Batch: B3
Date of Experiment: 14-07-2021	Date of Submission: 14-07-2021
Grade:	

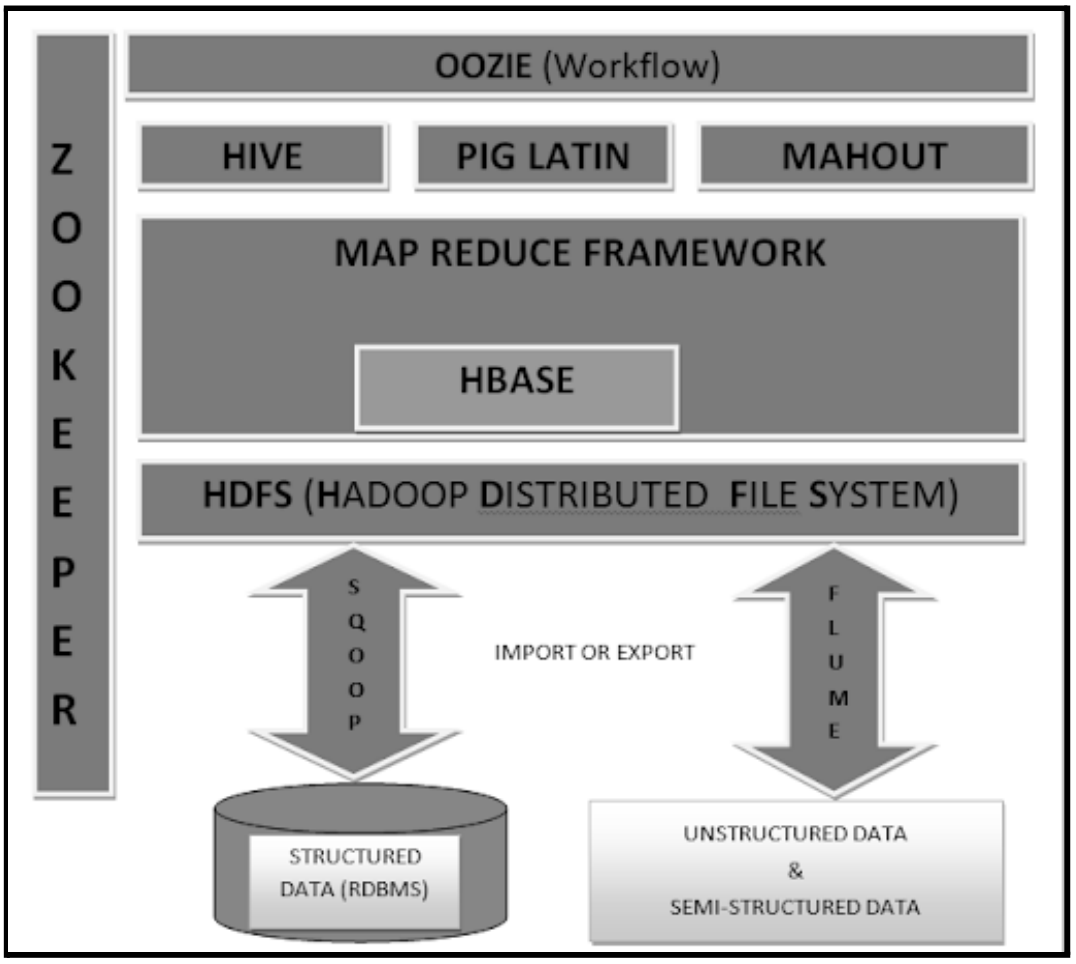
NOTE: Case Study - Hadoop in Banking *(Section B.5)*

B.1 What is Hadoop? Explain its ecosystem in detail.

(Paste your Search material completed during the 2 hours of practice in the lab here)

Ans:

Hadoop is a framework which deals with Big Data but unlike any other framework it's not a simple framework, it has its own family for processing different things which are tied up in one umbrella called as Hadoop Ecosystem.



HDFS (Hadoop Distributed File System)

➔ HDFS is a main component of Hadoop and a technique to store the data in a distributed manner to compute fast. HDFS saves data in a block of 64MB(default) or 128 MB in size which is the logical splitting of data in a Datanode (physical storage of data) in a Hadoop cluster(formation of several Datanode which is a collection of commodity hardware connected through a single network). All information about data splits in data nodes known as metadata is captured in Namenode which is again a part of HDFS.

MapReduce Framework

➔ It is another main component of Hadoop and a method of programming in distributed data stored in HDFS. We can write Map reduce programs by using any language like JAVA, C++ PIPEs, PYTHON, RUBY etc. By name only Map Reduce gives its functionality Map will do mapping of logic into data (distributed in HDFS) and once the computation is over reducer will collect the result of Map to generate the final output result of MapReduce. MapReduce Program can be applied to any type of data whether Structured or Unstructured stored in HDFS. Example - word count using MapReduce.

HBASE

➔ Hadoop Database or HBASE is a non-relational (NoSQL) database that runs on top of HDFS. HBASE was created for large tables which have billions of rows and millions of columns with fault tolerance capability and horizontal scalability and based on Google Big Table. Hadoop can perform only batch processing, and data will be accessed only in a sequential manner for random access of huge data HBASE is used.

Hive

➔ Many programmers and analysts are more comfortable with Structured Query Language than Java or any other programming language for which Hive is created by Facebook and later donated to Apache foundation. Hive mainly deals with structured data which is stored in HDFS with a Query Language similar to SQL and known as HQL (Hive Query Language). Hive also run Map reduce program in a backend to process data in HDFS but here programmer has not to worry about that backend MapReduce job it will look similar to SQL and the result will be displayed on the console.

Pig

➔ Similar to HIVE, PIG also deals with structured data using the PIG LATIN language. PIG was originally developed at Yahoo to answer a similar need to HIVE. It is an alternative provided to programmers who love scripting and don't want to use Java/Python or SQL to process data. A Pig Latin program is made up of a series of operations, or transformations, that are applied to the input data which runs the MapReduce program in the backend to produce output.

Mahout

- ➔ Mahout is an open-source machine learning library from Apache written in java. The algorithms it implements fall under the broad umbrella of machine learning or collective intelligence. This can mean many things, but at the moment for Mahout it means primarily recommender engines (collaborative filtering), clustering, and classification. Mahout aims to be the machine learning tool of choice when the collection of data to be processed is very large, perhaps far too large for a single machine. In its current incarnation, these scalable machine learning implementations in Mahout are written in Java, and some portions are built upon Apache's Hadoop distributed computation project.

Oozie

- ➔ It is a workflow scheduler system to manage Hadoop jobs. It is a server-based Workflow Engine specialized in running workflow jobs with actions that run Hadoop MapReduce and Pig jobs. Oozie is implemented as a Java Web-Application that runs in a Java Servlet-Container. Hadoop deals with bigdata and when some programmer wants to run many jobs successively like the output of job A will be input to Job B and similarly, the output of job B is input to job C and the final output will be the output of job C. To automate this sequence we need a workflow and to execute same we need an engine for which OOZIE is used.

Zookeeper

- ➔ Writing distributed applications is difficult because partial failure may occur between nodes to overcome this Apache Zookeeper has been developed by maintaining an open-source server that enables highly reliable distributed coordination. ZooKeeper is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services. In case of any partial failure, clients can connect to any node and be assured that they will receive the correct, up-to-date information.

B.2 Observations and learning:

(Students are expected to comment on the output obtained with clear observations and learning for each task/ subpart assigned)

Hadoop runs few applications on distributed systems with thousands of nodes involving petabytes of information. It has a distributed file system, called Hadoop Distributed File System or HDFS, which enables fast data transfer among the nodes.

B.3 Conclusion:

(Students must write the conclusion as per the attainment of individual outcome listed above and learning/observation noted in section B.3)

We are able to define key issues in big data management and its associated applications in intelligent business and scientific computing.

B.4 Question of Curiosity

(To be answered by student based on the practical performed and learning/observations)

Q1: Discuss the limitations of Hadoop?

Ans:

Various limitations of Hadoop are given below along with their solution-

A. Issues with Small Files

- ➔ The main problem with Hadoop is that it is not suitable for small data. HDFS cannot support the random reading of small due to its high capacity design.
- ➔ Small files are smaller than the HDFS block size (default 128MB). If you are storing these huge numbers of small files, HDFS cannot handle these lots of small files.
- ➔ As HDFS was designed to work with a small number of large files for storing large data sets rather than a large number of small files. If there are lot many small files, then the NameNode will be overloaded since it stores the namespace of HDFS.

Solution:

- ➔ Simply merge the small files to create bigger files and then copy bigger to HDFS.
- ➔ Hadoop Archives (HAR files) deals with the problem of lots of small files. Hadoop Archives works by building a layered filesystem on the top of HDFS.
- ➔ With the help Hadoop archive command, HAR files are created; this runs a MapReduce job to pack the files being archived into a small number of HDFS files. Reading files through HAR is not more efficient than reading through HDFS.
- ➔ As each HAR file access requires two index files read as well the data file to read, this will make it slower.
- ➔ Sequence files also overcome the small file problem. In which we use the filename as the key and the file contents as the value.
- ➔ By writing a program for files (100 KB), we can put them into a single Sequence file and then we can process them in a streaming fashion operating on the Sequence file.
- ➔ MapReduce in Hadoop can break the Sequence file into chunks and operate on each chunk independently because the Sequence file is splittable.
- ➔ By storing files in Hbase we can overcome the small file problem. We are not storing millions of small files into HBase rather adding the binary content of the file to a cell.

B. Slow Processing Speed

- ➔ MapReduce processes a huge amount of data. In Hadoop, MapReduce works by breaking the processing into phases: Map and Reduce. So, MapReduce requires a lot of time to perform these tasks, thus increasing latency. Hence, reduces processing speed.

Solution:

- ➔ By in-memory processing of data, Apache Spark overcomes this issue. As in In-memory processing, no time is spent in moving the data/processes in and out of the disk, thus this makes it faster.
- ➔ Apache Spark is 100 times faster as compared to MapReduce because it processes everything in memory.

➔ Flink can also overcome this issue. Flink processes faster than Spark because of its streaming architecture.

C. Support for Batch Processing only

➔ Hadoop only supports batch processing, it is not suitable for streaming data. Hence, overall performance is slower. MapReduce framework doesn't leverage the memory of the Hadoop cluster to the maximum.

Solution

➔ Apache Spark solves this problem as it supports stream processing. But Spark stream processing is not as much efficient as Flink as it uses micro-batch processing. Apache Flink improves the overall performance as it provides single run-time for the streaming as well as batch processing.

D. No Real-time Processing

➔ Apache Hadoop is a batch processing framework. It means it takes a huge amount of data in input, processes it and produces the result.

➔ Batch processing is very efficient for processing a high volume of data, but depends on the size of data being processed and the computational power of the system; output can be delayed significantly. Apache Hadoop is not suitable for Real-time processing.

Solution:

➔ Spark is suitable for stream processing. Steaming processing provide continuous input/output data. It process data within a small amount of time.

➔ Flink provides single run-time for both streamings as well as batch processing.

E. Iterative Processing

➔ Apache Hadoop is not much efficient for iterative processing. As Hadoop is not supported cyclic data flow (i.e. a chain of stages in which each output of the previous stage is the input to the next stage).

Solution:

➔ Spark overcomes this issue. As Apache Spark accesses data from RAM instead of the Disk. This dramatically improves the performance of an iterative algorithm that accesses the same dataset repeatedly.

➔ In Apache Spark, for iterative processing, each iteration has to be scheduled and executed separately.

F. Latency

➔ MapReduce in Hadoop is slower because it supports different formats, is structured and has a huge amount of data. In MapReduce, Map takes a set of data and converts it into another set of data, where an individual element is broken down into a key-value pair.

➔ Reduce takes the output from the map as and Reduce takes the output from the map as input and process further. MapReduce requires a lot of time to perform these tasks thereby increasing latency.

Solution:

➔ Apache Spark can reduce this issue. Although Spark is the batch system, it is relatively faster, because it caches much of the input data on memory by RDD. Apache Flink data streaming achieves low latency and high throughput.

G. No Ease of Use

➔ MapReduce developer in Hadoop needs to hand code for each and every operation which makes it very difficult to work. In Hadoop, MapReduce has no interactive mode, but adding hive and pig makes working with MapReduce a little easier.

Solution:

➔ Spark has overcome this issue, as Spark has an interactive mode. So, that developers and users alike can have intermediate feedback for queries and other activities.

➔ As spark has tons of high-level operators so it is easy to program Spark. One can also use Apache Flink as it also has high-level operators.

H. Security Issue

➔ Apache Hadoop is challenging in maintaining complex applications. Hadoop is missing encryption at the storage and network levels, which is a major point of concern. Apache Hadoop supports Kerberos authentication, which is hard to manage.

Solution:

➔ Apache Spark provides a security bonus. If you run Apache Spark in HDFS, it can use HDFS ACLs and file-level permissions.

I. Vulnerable by Nature

➔ Apache Hadoop is written in Java. Java is the most popular language, hence java is most heavily exploited by cybercriminals.

J. No Caching

➔ Apache Hadoop is not efficient for caching. MapReduce cannot cache the intermediate data in memory for further requirements and this diminishes the performance of Hadoop.

Solution:

➔ Spark and Flink overcome this issue. Spark and Flink cache data in memory for further iterations which enhance the overall performance.

K. Lengthy Code

➔ Apache Hadoop has 1, 20,000 lines of code. The number of lines produces the number of bugs. Hence it will take more time to execute the programs.

Solution:

➔ Spark and Flink are written in Scala and Java. But the implementation is in Scala, so the number of lines of code is lesser than Hadoop. Thus, it takes less time to execute the programs.

Q2: What are the drawbacks of the existing system and how Hadoop is giving solutions for it?

Ans:

A. Scalable

➔ Hadoop is a highly scalable storage platform because it can store and distribute very large data sets across hundreds of inexpensive servers that operate in parallel. Unlike traditional relational database systems (RDBMS) that can't scale to process large amounts of data.

B. Cost-effective

➔ Hadoop also offers a cost-effective storage solution for businesses exploding data sets. The problem with traditional relational database management systems is that it is extremely cost-prohibitive to scale to such a degree in order to process such massive volumes of data. In an effort to reduce costs, many companies in the past would have had to down-sample data and classify it based on certain assumptions as to which data was the most valuable. The raw data would be deleted, as it would be too cost-prohibitive to keep.

C. Flexible

➔ Hadoop enables businesses to easily access new data sources and tap into different types of data (both structured and unstructured) to generate value from that data. This means businesses can use Hadoop to derive valuable business insights from data sources such as social media, email conversations.

D. Fast

➔ Hadoop's unique storage method is based on a distributed file system that basically 'maps' data wherever it is located on a cluster. The tools for data processing are often on the same servers where the data is located, resulting in much faster data processing. If you're dealing with large volumes of unstructured data, Hadoop is able to efficiently process terabytes of data in just minutes, and petabytes in hours.

E. Resilient to failure

➔ A key advantage of using Hadoop is its fault tolerance. When data is sent to an individual node, that data is also replicated to other nodes in the cluster, which means that in the event of failure, there is another copy available for use.

B.5 Case Study

Hadoop in Banking: The Game Changer

What is Hadoop?

Apache Hadoop is an open-source software framework written in Java for distributed storage and processing of very large datasets on multiple clusters. Developed by Doug Cutting and Mike Cafarella in 2005, the core of Apache Hadoop consists of a Hadoop Distributed File system for storage and MapReduce for processing data. The basic philosophy of Hadoop is to reduce dependence on expensive legacy system hardware to enable distributed parallel processing of very large amounts of data across inexpensive, standard, commodity servers to process and store data without any volume limitations. Hadoop makes the process of storing and managing data economical and reliable.

What are the key features of Hadoop?

- Reliable: Fail-Safe technology that prevents loss of data even in an event of hardware failure.
- Powerful: Unique storage method based on distributed file system resulting in the faster data processing.
- Scalable: stores and distributes datasets to operate in parallel, allowing businesses to run applications on thousands of nodes.
- Cost-effective: Runs on commodity machines & network
- Simple and flexible APIs: Enables a large ecosystem of solution providers such as log processing, recommendation systems, data warehousing, fraud detection, etc.

How is it revolutionizing the Finance Industry?

Big Data is no longer just a buzzword for the banking and financial industry, for it addresses various issues with the 5Vs; Volume, Velocity, Variety and Veracity. Wondering what the 5th V means? As Bernard Marr, a leading business and Data Expert explains, it is the most important V of big data, for it defines how the other attributes work; Value. Value refers to an organization's ability to turn data into something worth more. It is this Value addition that makes Big Data and Hadoop not just a new trend but a breakthrough for the finance industry.

Banks like BNY Mellon, Morgan Stanley, Bank of America, Credit Suisse, PNC, etc. are already working on strategies around Big Data in Banking, and other banks are rapidly catching up.

How is it used in banking and finance?

The reason for Hadoop's success in the banking and finance domain is its ability to address various issues faced by the financial industry at minimal cost and time. Despite the various benefits of Hadoop, applying it to a particular problem needs due diligence. Some of the scenarios in which it is used are:

Fraud detection:

Fraud, financial crimes and data breaches are some of the most costly challenges in the industry. Hadoop analytics help financial organizations detect, prevent and eliminate internal and external fraud, as well as reduce the associated costs. Analyzing points of sale, authorizations and transactions, and other data points help banks identify and mitigate fraud. For example, big data technology can alert the bank that a credit or debit card has been lost or stolen by picking up on unusual behaviour patterns. This then gives the bank time to put a temporary hold on the card while contacting its account owner.

Risk management:

Every financial firm needs to assess risk accurately, and big data solutions enable them to do so by effectively evaluating credit exposures. Banks analyze transactional data to determine risk and exposure based on simulated market behaviour, scoring customers and potential clients. Hadoop solutions allow for a complete and accurate view of risk and impact, enabling firms to make the best, most informed decisions.

Contact centre efficiency optimization:

Ensuring customers are satisfied is of utmost importance when it comes to finances, and big data can help resolve problems quickly by allowing banks to anticipate customer needs ahead of time. Analyzing data within the contact centre provides agents with timely and concise insight that satisfies customers quickly and efficiently, ensuring cost-effectiveness and even improving cross-sales success rates.

Customer segmentation for optimized offers:

Big data provides a way to understand customers' needs at a granular level so that banks and financial organizations can deliver targeted offers more effectively. In turn, these more personalized offers result in higher acceptance rates, increased customer satisfaction, higher profitability and greater retention. Detailed information about customers derived from social media and transactions can be utilized to reduce customer acquisition costs as well as turnover.

Customer churn analysis:

Everybody knows that it's cheaper to keep a customer than it is to go out and find a new one. Big data and Hadoop technologies can help financial firms keep retain more of their customers by analyzing behaviour and identifying patterns that lead to customer abandonment. When are customers most likely to leave for the competition, and why? What causes customer dissatisfaction? Where did the firm fail? This information for determining how to avoid customer abandonment is priceless. It's imperative for financial firms to learn the right steps to implement in order to meet customer needs and save their most profitable customers.

Sentiment analysis:

Hadoop and advanced analytics tools help analyze social media in order to monitor user sentiment of a firm, brand or product. If a bank is running a campaign, big data tools can monitor social media by name and report on it by hashtag, campaign name or platform. Analytics on the fine-grained details are insightful, and the bank could then make decisions more accurately based on these insights in terms of timing, targeting and demographics.

Customer experience analytics:

As consumer-facing enterprises, financial institutions need to take advantage of the customer data that resides in all of the silos across various lines of business. These include portfolio management, customer relationship management, loan systems, contact centre, etc. Big data can provide better insight and understanding, allowing firms to match offers to a customer or prospect's needs. This then helps the firm to optimize and improve profitable and long-term customer relationships.

Data Storage and Security:

Protection, easy storage and access to financial data are the optimal needs of banks and finance firms. While Hadoop Distributed File System (HDFS) provides scalable and reliable data storage designed to span large clusters of commodity servers, MapReduce processes each node in parallel, transferring only the package code for that node. This means information is stored in more than one cluster but with additional safety to provide a better and safer data storage option.

Does Hadoop have limitations too?

Although Hadoop has been embraced by several banking organizations and it forms the backbone of several applications running Big Data technology, there are also several reasons Hadoop may not always be the best solution. Some of them are:

Big Data understanding:

Hadoop is normally implemented when Big Data is to be implemented. But before using it, one must ask the right questions and ponder upon whether it is the right solution. Any organization that has a huge inflow of data from various sources and which is facing issues to store and effectively use the existing data can use Hadoop and Big Data solutions for their enterprise.

It is not a solution, but a tool:

Hadoop is not the complete solution. Although fraud detection and risk management leverage the strengths of Hadoop, Hadoop by itself does not solve these issues. Programmers need to write codes with an understanding of the problem so that they utilize Hadoop's strong points to solve the business problem. e.g. Big Data does not help in picking up unusual patterns. Big data merely allows large data to be processed concurrently.

Not a unique service:

Hadoop allows analysis, but there are many products that allow you to do data analysis. So, though Hadoop can be used for the purpose of analysis, implementing the framework only to address analytical issues will not be a smart idea. Hadoop is beneficial only if one finds more than one scenario where its USPs can be used properly.

Other vulnerabilities:

Like any other technology, Hadoop is not foolproof or foolproof for that matter. Data is at risk as encryption is missing in the Hadoop system at storage and network levels. Also, since Hadoop makes various duplicates to store data so that it can be retrieved in case of failure, it is vulnerable to data breaches like Java, the language in which the framework of Hadoop is written.

Here's an analogy: Consider a dinner knife. You can use it very well to butter your toast, to cut a piece of potato, to wedge open a shut-in, or use it as a screwdriver for wide-notched screws; but it is useless if you want to drink soup or if you want to make a phone call.

Hadoop is like a knife. The programmers use it to do things effectively where it is applicable. Hadoop does not do fraud detection or risk management, actually, it does not do any business logic by itself; it just manages the storage and retrieval of data in a distributed way.

Conclusion:

The bottom line is that all enterprises, especially financial firms, need to use big data and Hadoop technologies to their fullest potential now, particularly with the overwhelming amount of data and transactions amassed on a daily basis. In order to remain competitive and maintain current customers while attracting new ones, financial firms should start planning to utilize big data technologies today or risk losing more customers to competitors utilizing these tools. That doesn't necessarily mean in every way possible – it just means in the best way possible for each organization.

Big data and Hadoop technologies are powerful and help financial organizations stay ahead in the market. Set them in motion and watch them deliver results.