

SRINIVAS UNIVERSITY
INSTITUTE OF ENGINEERING & TECHNOLOGY



BIG DATA ANALYTICS

B. TECH 7th SEM (2024-25)

PREPARED BY,
Mahesh Kumar V B
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

MODULE - I

What is Data?

Data is defined as individual facts, such as numbers, words, measurements, observations or just descriptions of things.

For example, data might include individual prices, weights, addresses, ages, names, temperatures, dates, or distances.

There are two main types of data:

Quantitative data is provided in numerical form, like the weight, volume, or cost of an item.

Qualitative data is descriptive, but non-numerical, like the name, sex, or eye colour of a person.

Characteristics of Data

The following are six key characteristics of data which discussed below:

1. Accuracy
2. Validity
3. Reliability
4. Timeliness
5. Relevance
6. Completeness

Accuracy

Data should be sufficiently accurate for the intended use and should be captured only once, although it may have multiple uses. Data should be captured at the point of activity.

Validity

Data should be recorded and used in compliance with relevant requirements, including the correct application of any rules or definitions. This will ensure consistency between periods and with similar organizations, measuring what is intended to be measured.

Reliability

Data should reflect stable and consistent data collection processes across collection points and over time. Progress toward performance targets should reflect real changes rather than variations in data collection approaches or methods. Source data is clearly identified and readily available from manual, automated, or other systems and records.

Timeliness

Data should be captured as quickly as possible after the event or activity and must be available for the intended use within a reasonable time period. Data must be available quickly and frequently enough to support information needs and to influence service or management decisions.

Relevance

Data captured should be relevant to the purposes for which it is to be used. This will require a periodic review of requirements to reflect changing needs.

Completeness

Data requirements should be clearly specified based on the information needs of the organization and data collection processes matched to these requirements.

Types of Digital Data

Digital data is the electronic representation of information in a format or language that machines can read and understand.

In more technical terms, Digital data is a binary format of information that's converted into a machine-readable digital format.

The power of digital data is that any analog inputs, from very simple text documents to genome sequencing results, can be represented with the binary system.

Types of Digital Data:

1. Structured
2. Unstructured
3. Semi Structured Data

Structured Data:

- Structured data refers to any data that resides in a fixed field within a record or file.
- Having a particular Data Model.
- Meaningful data.
- Data arranged in row and column.
- Structured data has the advantage of being easily entered, stored, queried and analysed.

E.g.: Relational Data Base, Spread sheets.

Structured data is often managed using Structured Query Language (SQL)

Sources of Structured Data:

- SQL Databases
- Spreadsheets such as Excel
- OLTP Systems
- Online forms
- Sensors such as GPS or RFID tags
- Network and Web server logs
- Medical devices

Advantages of Structured Data:

1. **Easy to understand and use:** Structured data has a well-defined schema or data model, making it easy to understand and use. This allows for easy data retrieval, analysis, and reporting.
2. **Consistency:** The well-defined structure of structured data ensures consistency and accuracy in the data, making it easier to compare and analyze data across different sources.
3. **Efficient storage and retrieval:** Structured data is typically stored in relational databases, which are designed to efficiently store and retrieve large amounts of data. This makes it easy to access and process data quickly.
4. **Enhanced data security:** Structured data can be more easily secured than unstructured or semi-structured data, as access to the data can be controlled through database security protocols.
5. **Clear data lineage:** Structured data typically has a clear lineage or history, making it easy to track changes and ensure data quality.

Disadvantages of Structured Data:

1. **Inflexibility:** Structured data can be inflexible in terms of accommodating new types of data, as any changes to the schema or data model require significant changes to the database.
2. **Limited complexity:** Structured data is often limited in terms of the complexity of relationships between data entities. This can make it difficult to model complex real-world scenarios.
3. **Limited context:** Structured data often lacks the additional context and information that unstructured or semi-structured data can provide, making it more difficult to understand the meaning and significance of the data.
4. **Expensive:** Structured data requires the use of relational databases and related technologies, which can be expensive to implement and maintain.
5. **Data quality:** The structured nature of the data can sometimes lead to missing or incomplete data, or data that does not fit cleanly into the defined schema, leading to data quality issues.

Unstructured Data:

- Unstructured data can not readily classify and fit into a neat box
- Also called unclassified data.
- Which does not conform to any data model.
- Business rules are not applied.
- Indexing is not required.

E.g.: photos and graphic images, videos, streaming instrument data, webpages, Pdf files, PowerPoint presentations, emails, blog entries, wikis and word processing documents.

Sources of Unstructured Data:

1. Web pages
2. Images (JPEG, GIF, PNG, etc.)
3. Videos
4. Reports
5. Word documents and PowerPoint presentations
6. Surveys

Advantages of Unstructured Data:

1. It supports the data which lacks a proper format or sequence
2. The data is not constrained by a fixed schema
3. Very Flexible due to absence of schema.
4. Data is portable
5. It is highly scalable
6. It can deal easily with the heterogeneity of sources.
7. These types of data have a variety of business intelligence and analytics applications.

Disadvantages Of Unstructured data:

1. It is difficult to store and manage unstructured data due to lack of schema and structure
2. Indexing the data is difficult and error prone due to unclear structure and not having pre-defined attributes. Due to which search results are not very accurate.
3. Ensuring security to data is difficult task.

Semi structured Data:

- Self-describing data.
- Metadata (Data about data).
- Also called quiz data: data in between structured and semi structured.

- It is a type of structured data but not followed data model.
- Data which does not have rigid structure.

E.g.: E-mails, word processing software.

XML and other markup language are often used to manage semi structured data.

Sources of semi-structured Data:

1. E-mails
2. XML and other markup languages
3. Binary executables
4. TCP/IP packets
5. Zipped files
6. Integration of data from different sources
7. Web pages

Advantages of Semi-structured Data:

1. The data is not constrained by a fixed schema
2. Flexible i.e., Schema can be easily changed.
3. Data is portable
4. It is possible to view structured data as semi-structured data
5. It can deal easily with the heterogeneity of sources.
6. **Flexibility:** Semi-structured data provides more flexibility in terms of data storage and management, as it can accommodate data that does not fit into a strict, predefined schema. This makes it easier to incorporate new types of data into an existing database or data processing pipeline.
7. **Scalability:** Semi-structured data is particularly well-suited for managing large volumes of data, as it can be stored and processed using distributed computing systems, such as Hadoop or Spark, which can scale to handle massive amounts of data.
8. **Faster data processing:** Semi-structured data can be processed more quickly than traditional structured data, as it can be indexed and queried in a more flexible way. This makes it easier to retrieve specific subsets of data for analysis and reporting.
9. **Improved data integration:** Semi-structured data can be more easily integrated with other types of data, such as unstructured data, making it easier to combine and analyze data from multiple sources.
10. **Richer data analysis:** Semi-structured data often contains more contextual information than traditional structured data, such as metadata or tags. This can provide additional insights and context that can improve the accuracy and relevance of data analysis.

Disadvantages of Semi-structured data

1. Lack of fixed, rigid schema make it difficult in storage of the data

2. Interpreting the relationship between data is difficult as there is no separation of the schema and the data.
3. Queries are less efficient as compared to structured data.
4. **Complexity:** Semi-structured data can be more complex to manage and process than structured data, as it may contain a wide variety of formats, tags, and metadata. This can make it more difficult to develop and maintain data models and processing pipelines.
5. **Lack of standardization:** Semi-structured data often lacks the standardization and consistency of structured data, which can make it more difficult to ensure data quality and accuracy. This can also make it harder to compare and analyze data across different sources.
6. **Reduced performance:** Processing semi-structured data can be more resource intensive than processing structured data, as it often requires more complex parsing and indexing operations. This can lead to reduced performance and longer processing times.
7. **Limited tooling:** While there are many tools and technologies available for working with structured data, there are fewer options for working with semi- structured data. This can make it more challenging to find the right tools and technologies for a particular use case.
8. **Data security:** Semi-structured data can be more difficult to secure than structured data, as it may contain sensitive information in unstructured or less- visible parts of the data. This can make it more challenging to identify and protect sensitive information from unauthorized access.

Overall, while semi-structured data offers many advantages in terms of flexibility and scalability, it also presents some challenges and limitations that need to be carefully considered when designing and implementing data processing and analysis pipelines.

BIG DATA

Big Data is a collection of data that is huge in volume, yet growing exponentially with time. It is a data with so large size and complexity that none of traditional data management tools can store it or process it efficiently. Big data is also a data but with huge size.

What is an Example of Big Data?

Following are some of the Big Data examples-

New York Stock Exchange: The New York Stock Exchange is an example of Big Data that generates about one terabyte of new trade data per day.



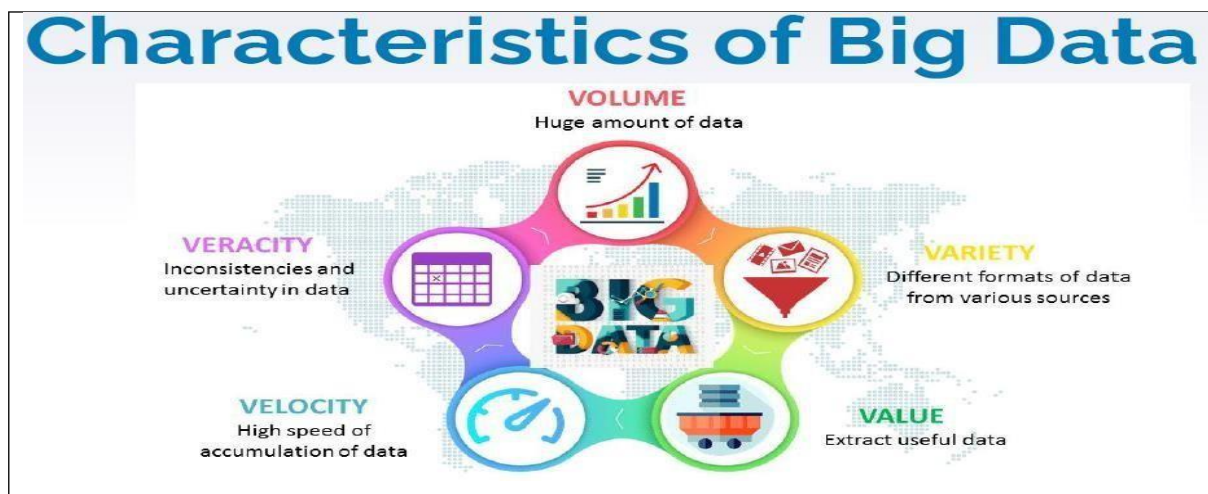
Social Media: The statistic shows that 500+terabytes of new data get ingested into the databases of social media site Facebook, every day. This data is mainly generated in terms of photo and video uploads, message exchanges, putting comments etc.



Jet engine: A single Jet engine can generate 10+terabytes of data in 30 minutes of flight time. With many thousand flights per day, generation of data reaches up to many Petabytes.



Big Data Characteristics



Volume: The name Big Data itself is related to an enormous size. Big Data is a vast 'volume' of data generated from many sources daily, such as business processes, machines, social media platforms, networks, human interactions, and many more.

Variety: Big Data can be structured, unstructured, and semi-structured that are being collected from different sources. Data will only be collected from databases and sheets in the past, but these days the data will come in array forms, that are PDFs, Emails, audios, SM posts, photos, videos, etc.

Veracity: Veracity means how much the data is reliable. It has many ways to filter or translate the data. Veracity is the process of being able to handle and manage data efficiently. Big Data is also essential in business development. E.g., Stockout or overstocking

Value: Value is an essential characteristic of big data. It is not the data that we process or store. It is valuable and reliable data that we store, process, and also analyze.

Velocity: Velocity plays an important role compared to others. Velocity creates the speed by which the data is created in real-time. It contains the linking of incoming data sets speeds, rate of change, and activity bursts. The primary aspect of Big Data is to provide demanding data rapidly.

Big data velocity deals with the speed at the data flows from sources like application logs, business processes, networks, and social media sites, sensors, mobile devices, etc.

Why Big Data?

Big Data initiatives were rated as “extremely important” to 93% of companies. Leveraging a Big Data analytics solution helps organizations to unlock the strategic values and take full advantage of their assets.

It helps organizations like,

- To understand Where, When and Why their customers buy
- Protect the company’s client base with improved loyalty programs
- Seizing cross-selling and upselling opportunities
- Provide targeted promotional information
- Optimize Workforce planning and operations
- Improve inefficiencies in the company’s supply chain
- Predict market trends
- Predict future needs
- Make companies more innovative and competitive
- It helps companies to discover new sources of revenue
- Companies are using Big Data to know what their customers want, who are their best customers, why people choose different products. The more a company knows about its customers, the more competitive it becomes.
- We can use it with Machine Learning for creating market strategies based on predictions about customers. Leveraging big data makes companies customer-centric.

- Companies can use Historical and real-time data to assess evolving consumers' preferences. This consequently enables businesses to improve and update their marketing strategies which make companies more responsive to customer needs.

Importance of big data



Big Data importance doesn't revolve around the amount of data a company has. Its importance lies in the fact that how the company utilizes the gathered data.

Every company uses its collected data in its own way. More effectively the company uses its data, more rapidly it grows.

The companies in the present market need to collect it and analyze it because:

1. **Cost Savings:** Big Data tools like Apache Hadoop, Spark, etc. bring cost-saving benefits to businesses when they have to store large amounts of data. These tools help organizations in identifying more effective ways of doing business.
2. **Time-Saving:** Real-time in-memory analytics helps companies to collect data from various sources. Tools like Hadoop help them to analyze data immediately thus helping in making quick decisions based on the learnings.
3. **Understand the market conditions:** Big Data analysis helps businesses to get a better understanding of market situations. For example, analysis of customer purchasing behaviour helps companies to identify the products sold most and thus produces those products accordingly. This helps companies to get ahead of their competitors.
4. **Social Media Listening:** Companies can perform sentiment analysis using Big Data tools. These enable them to get feedback about their company, that is, who is saying what about the company. Companies can use big data tools to improve their online presence.

5. **Boost Customer Acquisition and Retention:** Customers are a vital asset on which any business depends on. No single business can achieve its success without building a robust customer base. But even with a solid customer base, the companies can't ignore the competition in the market.
6. If we don't know what our customers want then it will degrade companies' success. It will result in the loss of client which creates an adverse effect on business growth. Big data analytics helps businesses to identify customer related trends and patterns. Customer behaviour analysis leads to a profitable business.
7. **Solve Advertisers Problem and Offer Marketing Insights:** Big data analytics shapes all business operations. It enables companies to fulfil customer expectations. Big data analytics helps in changing the company's product line. It ensures powerful marketing campaigns.
8. **The driver of Innovations and Product Development:** Big data makes companies capable to innovate and redevelop their products.

Challenges of Big Data

When implementing a big data solution, here are some of the common challenges your business might run into, along with solutions.

- **Managing massive amounts of data:** It's in the name—big data is big. Most companies are increasing the amount of data they collect daily. Eventually, the storage capacity a traditional data center can provide will be inadequate, which worries many business leaders. Forty-three percent of IT decision-makers in the technology sector worry about this data influx overwhelming their infrastructure.
 - To handle this challenge, companies are migrating their IT infrastructure to the cloud. Cloud storage solutions can scale dynamically as more storage is needed. Big data software is designed to store large volumes of data that can be accessed and queried quickly.
- **Integrating data from multiple sources:** The data itself presents another challenge to businesses. There is a lot, but it is also diverse because it can come from a variety of different sources. A business could have analytics data from multiple websites, sharing data from social media, user information from CRM software, email data, and more. None of this data is structured the same but may have to be integrated and reconciled to gather necessary insights and create reports.
 - To deal with this challenge, businesses use data integration software, ETL software, and business intelligence software to map disparate data sources into a common structure and combine them so they can generate accurate reports.
- **Ensuring data quality:** Analytics and machine learning processes that depend on big data to run also depend on clean, accurate data to generate valid insights and predictions. If the data is corrupted or incomplete, the results may not be what you expect. But as the sources, types, and quantity of data increase, it can be hard to determine if the data has the quality you need for accurate insights.

- Fortunately, there are solutions for this. Data governance applications will help organize, manage, and secure the data you use in your big data projects while also validating data sources against what you expect them to be and cleaning up corrupted and incomplete data sets. Data quality software can also be used specifically for the task of validating and cleaning your data before it is processed.

- **Keeping data secure:** Many companies handle data that is sensitive, such as,
 - Company data that competitors could use to take a bigger market share of the industry
 - Financial data that could give hackers access to accounts
 - Personal user information of customers that could be used for identity theft

If a business handles sensitive data, it will become a target of hackers. To protect this data from attack, businesses often hire cybersecurity professionals who keep up to date on security best practices and techniques to secure their systems. Whether you hire a consultant or keep it in-house, you need to ensure that data is encrypted, so the data is useless without an encryption key. Add identity and access authorization control to all resources so only the intended users can access it. Implement endpoint protection software so malware can't infect the system and real-time monitoring to stop threats immediately if they are detected.

- **Selecting the right big data tools:** Fortunately, when a business decides to start working with data, there is no shortage of tools to help them do it. At the same time, the wealth of options is also a challenge. Big data software comes in many varieties, and their capabilities often overlap. How do you make sure you are choosing the right big data tools?
 - Often, the best option is to hire a consultant who can determine which tools will fit best with what your business wants to do with big data. A big data professional can look at your current and future needs and choose an enterprise data streaming or ETL solution that will collect data from all your data sources and aggregate it. They can configure your cloud services and scale dynamically based on workloads. Once your system is set up with big data tools that fit your needs, the system will run seamlessly with very little maintenance.
- **Scaling systems and costs efficiently:** If you start building a big data solution without a well-thought-out plan, you can spend a lot of money storing and processing data that is either useless or not exactly what your business needs. Big data is big, but it doesn't mean you have to process all of your data.
 - When your business begins a data project, start with goals in mind and strategies for how you will use the data you have available to reach those goals. The team involved in implementing a solution needs to plan the type of data they need and the schemas they will use before they start building the system so the project doesn't go in the wrong direction. They also need to create policies for purging old data from the system once it is no longer useful.

- **Lack of skilled data professionals:** One of the big data problems that many companies run into is that their current staff have never worked with big data before, and this is not the type of skill set you build overnight. Working with untrained personnel can result in dead ends, disruptions of workflow, and errors in processing.
 - There are a few ways to solve this problem. One is to hire a big data specialist and have that specialist manage and train your data team until they are up to speed. The specialist can either be hired on as a full -time employee or as a consultant who trains your team and moves on, depending on your budget.
 - Another option, if you have time to prepare ahead, is to offer training to your current team members so they will have the skills once your big data project is in motion.
 - A third option is to choose one of the self-service analytics or business intelligence solutions that are designed to be used by professionals who don't have a data science background.
- **Organizational resistance:** Another way people can be a challenge to a data project is when they resist change. The bigger an organization is, the more resistant it is to change. Leaders may not see the value in big data, analytics, or machine learning. Or they may simply not want to spend the time and money on a new project.
 - This can be a hard challenge to tackle, but it can be done. You can start with a smaller project and a small team and let the results of that project prove the value of big data to other leaders and gradually become a data-driven business. Another option is placing big data experts in leadership roles so they can guide your business towards transformation.

A COMPREHENSIVE HISTORY OF APACHE HADOOP

Hadoop is an open-source framework designed for distributed storage and processing of large data sets. It allows organizations to handle vast amounts of data across clusters of computers using simple programming models. Hadoop's core components are,

1. Hadoop Distributed File System (HDFS)
2. MapReduce programming model.

Core Components

1. Hadoop Distributed File System (HDFS)

Functionality: HDFS is designed to store large files across multiple machines. It splits files into blocks and distributes them across the cluster, ensuring fault tolerance and high availability.

Architecture: HDFS follows a master-slave architecture, where the Name Node manages the file system metadata, and Data Nodes store the actual data blocks.

2. MapReduce

Programming Model: MapReduce simplifies the processing of large data sets by dividing tasks into two main functions:

- a. **Map function** - which processes input data and generates key-value pairs
- b. **Reduce function** - which aggregates and processes these pairs to produce the final output.

Parallel Processing: MapReduce allows parallel processing of data across the cluster, making it efficient for handling large-scale computations.

Hadoop Ecosystem

Beyond HDFS and MapReduce, the Hadoop ecosystem includes various tools and projects that enhance its functionality. Some notable components are:

Apache Hive: A data warehouse infrastructure built on top of Hadoop, allowing users to query and analyze large data sets using a SQL-like language.

Apache Pig: A high-level platform for creating MapReduce programs using a scripting language called Pig Latin.

Apache HBase: A distributed, scalable, NoSQL database that runs on top of HDFS, providing real-time read/write access to large data sets.

Apache Spark: A fast, in-memory data processing engine that supports batch processing, interactive queries, and stream processing.

Apache Flume: A service for efficiently collecting, aggregating, and moving large amounts of log data into HDFS.

Apache Sqoop: A tool for transferring data between Hadoop and relational databases.

Industry Use Cases

Hadoop's versatility and scalability have led to its adoption across various industries. Some common use cases include:

Log and Event Data Analysis: Organizations use Hadoop to store and analyze log and event data from servers, applications, and devices. This helps in monitoring system performance, detecting anomalies, and gaining insights into user behavior.

Recommendation Engines: Companies like Netflix and Amazon use Hadoop to analyze user preferences and behaviors, providing personalized recommendations to enhance user experience.

Data Warehousing: Hadoop serves as a cost-effective data warehouse solution, enabling organizations to store and process large volumes of structured and unstructured data for business intelligence and reporting.

Social Media Analytics: Hadoop is used to analyze social media data, helping businesses understand customer sentiment, track brand mentions, and identify trends.

History of Apache Hadoop

Origins and Development: Creation by Doug Cutting

Background: Doug Cutting, a renowned software developer, initially created Apache Lucene, a widely used text search library. While working on Lucene, Cutting and Mike Cafarella decided to tackle the ambitious goal of building an open-source web search engine, leading to the creation of Apache Nutch.

Naming: The name "Hadoop" is distinctive and memorable. Doug Cutting's son named his stuffed yellow elephant "Hadoop," and Cutting adopted this name for the project because it was unique, easy to spell and pronounce, and had no prior associations.

Early Ambitions and Challenges

Goal: Developing a web search engine from scratch posed significant challenges. Not only was the software complex, but running it without a dedicated operations team was daunting. Cutting and Cafarella estimated that supporting a one-billion-page index would require approximately \$500,000 in hardware, with monthly operational costs around \$30,000.

Nutch Development: Nutch started in 2002 and quickly produced a functional crawler and search system. However, the architecture of Nutch faced scalability issues, preventing it from handling the vast number of pages on the web efficiently.

Influence of Google's Innovations

Google File System (GFS): Breakthrough: In 2003, Google published a paper on the Google File System (GFS), detailing a scalable storage solution designed to manage large amounts of data across numerous machines. This publication offered a roadmap for addressing the storage needs of large-scale web crawling and indexing.

Adoption: Inspired by the GFS paper, Nutch developers set out to create an open-source implementation. This led to the development of the Nutch Distributed Filesystem (NDFS), designed to manage large files generated during web crawling and indexing.

Introduction of MapReduce

Google's Contribution: In 2004, Google released a paper introducing MapReduce, a programming model for processing large data sets with a parallel, distributed algorithm on a cluster.

Integration into Nutch: By early 2005, Nutch had integrated MapReduce, allowing its algorithms to run more efficiently on large data sets using NDFS. This integration marked a significant step towards achieving scalable web search capabilities.

Evolution into Hadoop

Independent Project: By February 2006, the components of NDFS and MapReduce had matured to the point where they were separated from Nutch, forming an independent subproject of Lucene named Hadoop.

Support from Yahoo!: Around this time, Doug Cutting joined Yahoo!, which provided the necessary resources and a dedicated team to further develop Hadoop. Yahoo!'s involvement was crucial in transforming Hadoop into a robust, web-scale system.

Hadoop at Yahoo!

WebMap and Dreadnaught: Yahoo!'s search engine architecture included the WebMap, which required significant computational power to process its vast graph of web links and URLs. The existing system, Dreadnaught, needed to scale from 20 to 600 nodes, but further scaling required a complete redesign.

Switch to Hadoop: In early 2006, Yahoo! decided to adopt Hadoop, recognizing its potential for scalability and efficiency. This decision led to the deployment of a 200-node research cluster and eventually a 10,000-core production cluster, which generated Yahoo!'s search index by February 2008.

Significant Milestones

Apache Top-Level Project: In January 2008, Hadoop was recognized as an Apache top-level project, marking its success and the growth of its community.

Widespread Use: Companies like Last.fm, Facebook, and the New York Times began using Hadoop. A notable example is the New York Times, which utilized Amazon EC2 and Hadoop to process 4 terabytes of scanned archives in less than 24 hours.

Achievements and Performance Benchmarks

Record-Breaking Performance

Sorting Records: Hadoop made headlines in April 2008 by setting a new record for sorting 1 terabyte of data in 209 seconds. This achievement was later surpassed by Google's MapReduce and subsequently by Yahoo! in 2009, which sorted 1 terabyte in 62 seconds.

Gray Sort Benchmark: In the 2014 Gray Sort benchmark, a team from Databricks used a 207-node Spark cluster to sort 100 terabytes in 1,406 seconds, demonstrating the rapid advancements in big data processing technologies.

Current Status and Industry Impact

Mainstream Adoption

Enterprise Use: Hadoop has become a mainstream tool in many enterprises, recognized for its role in managing and analysing large datasets. Its ability to store vast amounts of data and perform complex analytics has made it a cornerstone of big data infrastructure.

Commercial Support: Hadoop's ecosystem is supported by major vendors like EMC, IBM, Microsoft, and Oracle, along with specialized Hadoop companies such as Cloudera, Hortonworks, and MapR. These companies offer various services, including commercial support, consulting, and training, to help organizations implement and optimize their Hadoop deployments.

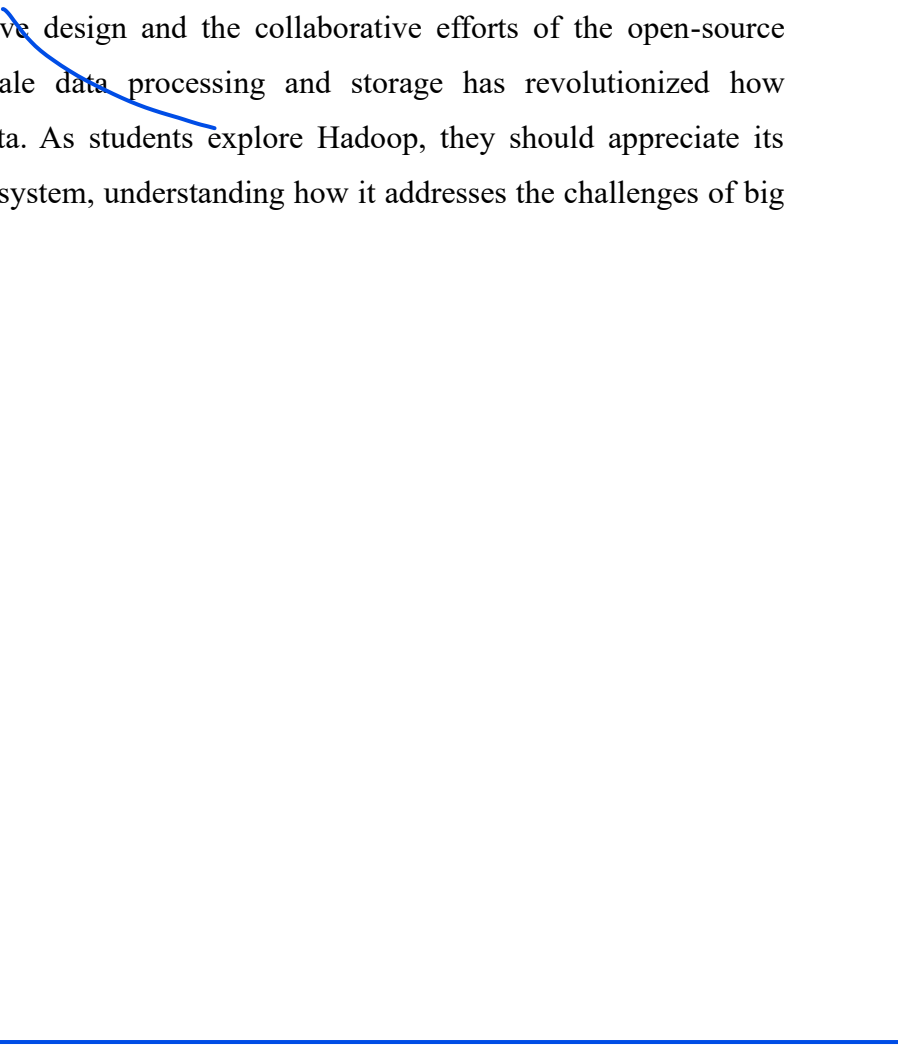
Industry Recognition

Widespread Integration: Hadoop's role as a general-purpose storage and analysis platform has been widely recognized. Many commercial products and services now incorporate Hadoop, reflecting its importance in the big data landscape.

Educational Importance: Understanding Hadoop and its ecosystem is essential for students and professionals aiming to work in fields involving big data. Its architecture, scalability, and ability to handle diverse data types make it a valuable tool for data scientists and engineers.

Conclusion

Hadoop's journey from an ambitious open-source project to a critical component of modern data infrastructure is a testament to its innovative design and the collaborative efforts of the open-source community. Its ability to handle large-scale data processing and storage has revolutionized how organizations manage and analyze their data. As students explore Hadoop, they should appreciate its historical context, core components, and ecosystem, understanding how it addresses the challenges of big data and its impact on various industries.



THE VALUE AND BENEFITS OF IBM INFOSPHERE BIGINSIGHTS RUNNING ON IBM SYSTEM Z

Current State of Business Analytics

1. Enterprise Data Warehouses:

- **Reliance on Data:** Most business analytics depend on data from enterprise data warehouses sourced from transaction and operational systems.
- **Characteristics:** This data is valuable, trusted, well-understood, and has known provenance.
- **Example:** A retail company uses its enterprise data warehouse to analyze sales trends and optimize marketing strategies.

Big Data Paradigm

1. Combining Diverse Data Sources:

- **Purpose:** Integrating enterprise data with social media, web logs, emails, documents, multimedia, text messages, and sensor information.
- **Benefit:** Provides a more comprehensive and enriched view of the world.
- **Example:** Combining customer transaction data with social media interactions to enhance customer profiles and tailor marketing efforts.

Technological Advancements

1. Hadoop and Map/Reduce:

- **Purpose:** Technologies like Hadoop use the map/reduce paradigm for parallel processing of large volumes of varied structured data across numerous nodes.
- **Function:** Breaks down large data sets into smaller analytic tasks, which are then combined for complete insights.
- **Example:** Processing large volumes of website clickstream data to understand user behaviour and improve website design.

Overview of Apache Hadoop in Business Analytics

1. Market Appeal:

- **Interest:** Analysts see the interest in Apache Hadoop as unstoppable due to its no-cost open-source software and low-cost commodity hardware.
- **Challenges:** Issues with scalability and reliability when applied to mission-critical data.
- **Example:** A start-up uses Hadoop to analyse customer data cost-effectively but faces challenges in scaling as the business grows.

Challenges in Big Data Initiatives

1. ETL Challenges:

- **Process:** Extracting, transforming, and loading (ETL) large volumes of data from traditional sources like IBM DB2, IBM IMS, and VSAM into Hadoop clusters.
- **Considerations:** Efficiency and cost-effectiveness.
- **Example:** A financial institution needing to move large volumes of transaction data into Hadoop for fraud detection analysis.

IBM InfoSphere BigInsights Solution

1. Enhancing Hadoop for Enterprise Use:

- **Purpose:** Transforms basic Hadoop into an enterprise-ready, business-critical analytics solution.
- **Integration:** Works with IBM System z, leveraging the platform's security and reliability for analyzing critical data and efficiently moving data between IBM z/OS and Hadoop clusters.

Advantages of IBM InfoSphere BigInsights on System z

1. Security and Efficiency:

- **Secure Storage and Analysis:** Ensures secure storage and analysis of sensitive information on System z or during transfer to Hadoop clusters.
- **Business Risk Reduction:** Minimizes potential data breaches and lowers costs associated with data management and movement.
- **Deeper Data Analysis:** Facilitates deeper data analysis, offering insights that drive business efficiencies, new revenue opportunities, and cost reductions.
- **Example:** A healthcare provider uses IBM InfoSphere BigInsights to securely analyze patient data, leading to improved patient outcomes and reduced operational costs.

Significance of IBM System z

1. Widespread Use:

- **Importance:** Critical for storing a significant amount of the world's business information.
- **Usage:** Utilized by 96 of the top 100 global banks and 23 of the top 25 US retailers.
- **Example:** A global bank relies on IBM System z for secure and reliable processing of financial transactions and customer data management.

Solution Overview

1. IBM InfoSphere BigInsights:

- **Combination:** Combines open-source Hadoop software with enterprise-grade capabilities.
- **Deployment:** Can be deployed both on and off the mainframe, depending on customer requirements.
- **Example:** A logistics company deploys IBM InfoSphere BigInsights to analyze supply chain data both on-premises and in the cloud, optimizing delivery routes and reducing costs.

Deployment Options

1. On System z Linux Partitions:

- **Security Perimeter:** Hadoop applications operate within the System z security perimeter.
- **Secure Data Access:** Mainframe technologies like IBM HiperSockets™ securely access and move production data to and from Hadoop.
- **Management Advantages:** Offers configuration flexibility, virtualized storage, and avoids the need for separate cluster nodes and network infrastructure.
- **Governance:** Extends System z governance to hybrid Hadoop implementations.
- **Example:** A government agency uses System z Linux partitions to securely analyze sensitive citizen data without the need for additional infrastructure.

2. On External Cluster Nodes:

- **Connection:** Connected via 10 GbE, suitable for less critical data.
- **Results Integration:** Results can be uploaded to a DB2 for z/OS-based data warehouse for augmenting customer records.
- **Example:** An e-commerce company analyzes website logs on external Hadoop clusters and integrates insights into their mainframe data warehouse to improve customer experience.

Visual Representation

1. System z Mainframe:

- **Components:** Includes DB2, VSAM, IMS, and logs.
- **Example:** A banking system uses these components to manage and analyze transaction data.

2. Linux for System z:

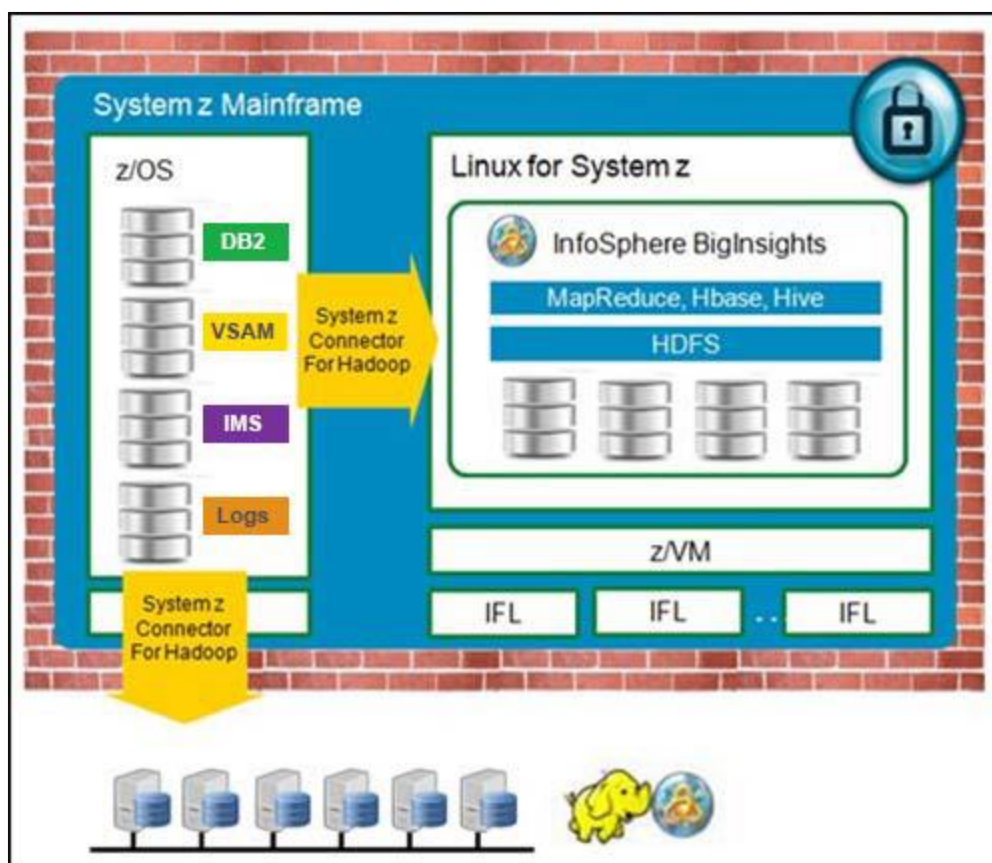
- **Components:** Hosts InfoSphere BigInsights with Hadoop components like MapReduce, HBase, Hive, and HDFS.
- **Example:** A research organization runs complex data analyses on Linux for System z to process and store scientific data.

3. System z Connector for Hadoop:

- **Function:** Facilitates data movement between the mainframe and Hadoop.

- **Example:** An insurance company uses the connector to transfer claims data to Hadoop for advanced analytics.
- 4. **z/VM with IFL:** (Integrated Facility for Linux)
 - **Support:** Provides virtualization support.
 - **Example:** A telecommunications company uses z/VM with IFL to virtualize and manage multiple workloads on a single mainframe.
- 5. **External System Nodes:**
 - **Connection:** Connected to the mainframe via 10 GbE, potentially running Hadoop.
 - **Example:** A media company uses external nodes for processing large volumes of multimedia data and integrates the results with their mainframe database.

This architecture provides flexibility in deployment while leveraging the strengths of IBM System z mainframes and Hadoop for big data analytics.



IBM InfoSphere BigInsights Software Capabilities

Overview

IBM InfoSphere BigInsights enhances Hadoop with enterprise-grade capabilities, making it suitable for various big data and analytics workloads. Its implementation in the System z environment offers unique advantages and considerations.

Hadoop Primer

Hadoop, inspired by Google's GFS and MapReduce, is a distributed file system and data processing engine designed to handle high data volumes. Developed by Douglas Cutting and later released as an Apache project, Hadoop has a growing ecosystem with numerous subprojects. Key components and principles of Hadoop include:

- **Hadoop Distributed File System (HDFS):** Ensures massive scalability across numerous nodes and disks.
- **Design for Failure:** Tolerates failures with block-replication, synchronization, and failover techniques.
- **Parallelism and Local Execution:** Executes tasks in parallel across a distributed cluster, processing data near its storage location, which is more efficient than moving large data blocks across networks.

Hadoop on the Mainframe

Implementing Hadoop on a virtualized System z mainframe involves unique challenges:

- **Cluster Nodes:** In a commodity environment, a cluster node is a physical server with many cores, memory, and HDDs. On the mainframe, a node is a virtual machine running Linux under IBM z/VM. Efficient resource allocation is managed by the z/VM administrator, and fewer, more powerful nodes are cost-effective due to BigInsights' licensing model.
- **Data Replication:** While HDFS assumes frequent server and disk failures, the stable Linux cluster on System z may have different data replication needs.

About IBM InfoSphere BigInsights

IBM's enterprise-grade Hadoop offering, InfoSphere BigInsights, is based on Apache Hadoop with additional capabilities:

- **Compatibility:** Ensures 100% compatibility with open-source Hadoop.
- **Enhanced Capabilities:** Provides tools for installation, management, and additional utilities without locking users into proprietary technology.

Standard Open-Source Utilities in InfoSphere BigInsights

InfoSphere BigInsights includes several standard open-source tools:

- **PIG:** A high-level platform for creating MapReduce programs used with Hadoop.
- **Hive/HCatalog:** Data warehousing tools that provide data summarization, query, and analysis.
- **Oozie:** A workflow scheduler system to manage Hadoop jobs.

- **HBase:** A distributed, scalable, big data store.
- **Zookeeper:** A centralized service for maintaining configuration information, naming, and providing distributed synchronization.
- **Flume:** A service for efficiently collecting, aggregating, and moving large amounts of log data.
- **Avro:** A framework for data serialization.
- **Chukwa:** A data collection system for monitoring large distributed systems.

By integrating these tools, InfoSphere BigInsights offers a comprehensive and flexible solution for big data analytics within the secure and scalable environment of IBM System z.

Key Software Capabilities of IBM InfoSphere BigInsights

IBM InfoSphere BigInsights offers advanced capabilities that distinguish it from other Hadoop distributions. Here are some of these key capabilities:

Big SQL

Big SQL provides a robust, ANSI-compliant SQL implementation with advantages over other SQL-on-Hadoop solutions:

- **SQL Language Compatibility:** Adheres to standard SQL syntax and features.
- **Support for Native Data Sources:** Can query data directly in its native format.
- **Performance:** Optimized for high performance in querying and processing.
- **Federation:** Allows querying and combining data from multiple sources.
- **Security:** Maintains high security standards.

Big SQL is designed to run natively on Hadoop data sets in HDFS and uses existing Hadoop standards like the Hive metastore, avoiding proprietary metadata or specific database requirements. For mainframe users, federation capabilities enable querying data across Hadoop clusters and DB2 on z/OS databases with appropriate permissions.

Big R

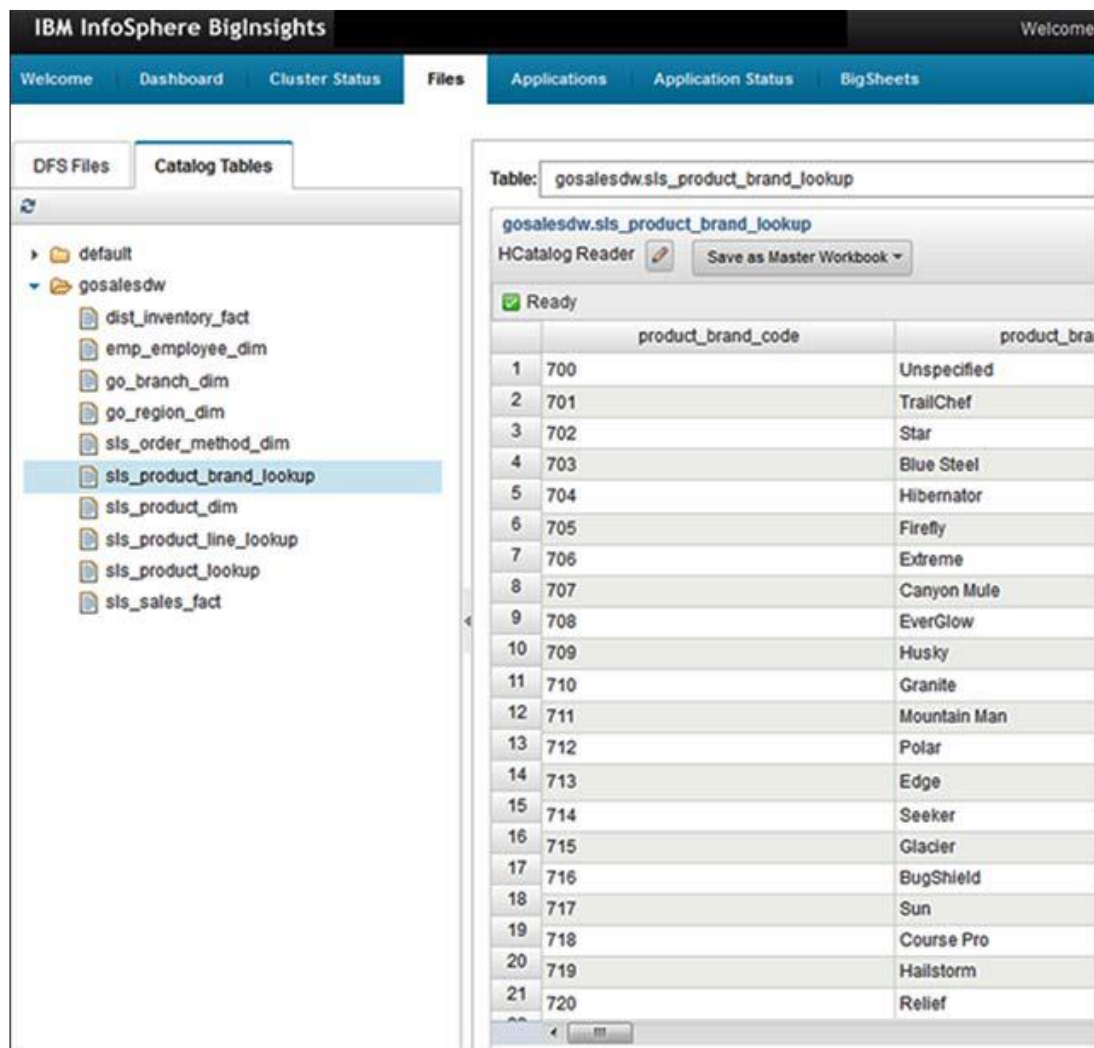
Big R integrates with the R programming language, providing a familiar environment for data scientists and developers:

- **End-to-End Integration:** Seamlessly integrates R with BigInsights.
- **Parallel Execution:** Enables parallelized execution of R code across the Hadoop cluster, eliminating the need for custom MapReduce code.
- **CRAN Support:** Allows downloading and running R packages from the Comprehensive R Archive Network in parallel on the cluster, enhancing performance and reducing development effort.

Big Sheets

Big Sheets offers a spreadsheet-style data manipulation and visualization tool for business users:

- **User-Friendly Interface:** Allows users to analyze data in Hadoop without needing to know Hadoop scripting or MapReduce programming.
- **Data Import:** Supports importing data from multiple formats, including data stored in Hive.
- **Visualization:** Provides tools for data visualization and analysis.



The screenshot shows the IBM InfoSphere BigInsights Big Sheets interface. On the left, a file tree under 'gosalesdw' lists various tables, with 'sis_product_brand_lookup' selected. The main area displays a table titled 'gosalesdw.sis_product_brand_lookup' with columns 'product_brand_code' and 'product_brand'. The table contains 21 rows of data, numbered 1 through 21. The interface includes a top navigation bar with tabs like 'Welcome', 'Dashboard', 'Cluster Status', 'Files', 'Applications', 'Application Status', and 'BigSheets'. Below the navigation bar, there are tabs for 'DFS Files' and 'Catalog Tables'. The table data is as follows:

	product_brand_code	product_brand
1	700	Unspecified
2	701	TrailChef
3	702	Star
4	703	Blue Steel
5	704	Hibernator
6	705	Firefly
7	706	Extreme
8	707	Canyon Mule
9	708	EverGlow
10	709	Husky
11	710	Granite
12	711	Mountain Man
13	712	Polar
14	713	Edge
15	714	Seeker
16	715	Glacier
17	716	BugShield
18	717	Sun
19	718	Course Pro
20	719	Hailstorm
21	720	Relief

Application Accelerators

IBM InfoSphere BigInsights extends the capabilities of open-source Hadoop with accelerators to speed up the development of high-quality applications for common big data use cases:

- **Text Analytics Accelerators:** Facilitate the development of applications that analyze text in multiple languages.
- **Machine Data Accelerators:** Provide tools for processing log files, such as web logs, mail logs, and specialized file formats.
- **Social Data Accelerators:** Enable importing and analyzing social data at scale from sources like tweets, boards, and blogs.

Usage Scenarios for IBM InfoSphere BigInsights on IBM System z MainFrame

1. ETL Processing

Scenario: Extract, Transform, Load (ETL) processing is crucial for data integration from various sources into a data warehouse. While tools like IBM InfoSphere DataStage are optimized for structured data, Hadoop excels in handling large volumes of semi-structured or unstructured data. In Hadoop, the process is often termed ELT (Extract, Load, Transform), as transformations typically occur post-loading.

Benefits:

- **Scalability:** Suitable for large datasets benefiting from parallel processing.
- **Data Type Flexibility:** Efficiently handles semi-structured and unstructured data.

2. Analytic Models with Multiple Data Sources

Scenario: Businesses often require insights from diverse data sources. Combining customer transactional data from z/OS with external data sources (web logs, customer service interactions, social media, etc.) can provide a holistic view of customer behavior and preferences.

Benefits:

- **Ad Hoc Data Exploration:** Facilitates combining and analyzing data in a common sandbox environment.
- **Cost Efficiency:** Easier and more affordable to manage diverse data sources in Hadoop.

3. Ad Hoc Analysis of Mainframe Data

Scenario: Performing experimental analysis directly on operational mainframe data can be risky. Extracting this data to Hadoop allows safe experimentation without affecting production systems.

Benefits:

- **Risk Mitigation:** Prevents data corruption and maintains the quality of service.
- **Flexibility:** Allows creating various data views without impacting mainframe operations.

4. Mainframe Log File Analysis

Scenario: Analyzing extensive mainframe log files (e.g., IBM System Management Facility (SMF), DB2 logs, WebSphere MQ logs) can provide insights into system usage, performance, and security.

Benefits:

- **Detailed Insights:** Understand usage patterns and identify issues before they impact production.
- **Long-term Data Retention:** Enables storing raw log data for extended periods, facilitating trend analysis.

5. Extending DB2 Functions with Hadoop

Scenario: Incorporating external data into DB2 for z/OS can enhance data models. DB2 for z/OS can use user-defined functions to query Hadoop clusters, pulling summary information without needing to load all the data into z/OS.

Benefits:

- **Enhanced Analytics:** Enriches mainframe data models with external big data sources.
- **Efficient Data Processing:** Pre-process data in Hadoop before integrating with DB2.

6. Integration and Data Movement

Scenario: Integrating and moving data between the mainframe and Hadoop is critical. Tools like IBM InfoSphere System z Connector for Hadoop and functions such as JAQL_SUBMIT and HDFS_READ in DB2 for z/OS simplify this process.

Benefits:

- **Seamless Integration:** Facilitates efficient data transfer between mainframe and Hadoop.
- **Data Availability:** Ensures mainframe applications reflect the latest data from Hadoop.

7. Running Analytic Tools on Hadoop Data

Scenario: Running analytics tools like IBM SPSS, IBM Cognos, and third-party applications (e.g., SAS, MicroStrategy) on Hadoop clusters can derive insights from large datasets stored in Hadoop.

Benefits:

- **Advanced Analytics:** Utilize powerful analytical tools on big data.
- **Versatile Deployment:** These tools can run on or off the mainframe, depending on the application's requirements.

Conclusion

Deploying IBM InfoSphere BigInsights on IBM System z MainFrame opens up various usage scenarios that enhance data processing, integration, and analytics capabilities. This integration leverages the strengths of both mainframe and Hadoop environments, providing robust, scalable, and secure solutions for modern data challenges.

Analysing the Data with Unix Tools

Example 2-1. Format of a National Climatic Data Center record

```
0057
332130  # USAF weather station identifier
99999   # WBAN weather station identifier
19500101 # observation date
0300    # observation time
4
+51317  # latitude (degrees x 1000)
+028783 # longitude (degrees x 1000)
FM-12
+0171   # elevation (meters)
99999
V020
320     # wind direction (degrees)
1       # quality code
N
0072
1
00450   # sky ceiling height (meters)
1       # quality code
C
N
010000  # visibility distance (meters)
1       # quality code
N
9
-0128   # air temperature (degrees Celsius x 10)
1       # quality code
-0139   # dew point temperature (degrees Celsius x 10)
1       # quality code
10268   # atmospheric pressure (hectopascals x 10)
1       # quality code
```

The goal is to find the highest recorded global temperature for each year from a dataset. You're using a shell script combined with `awk` to process weather data. Here's a detailed breakdown:


1. **Input Data:** The dataset consists of compressed files (.gz) for each year containing weather records. Each line in the dataset includes temperature and quality code fields.
2. **Processing Tool:** The primary tool used is `awk`, which is a powerful text-processing language.
3. **Temperature Representation:** Temperature values are scaled by a factor of 10 (e.g., 317 represents 31.7°C).
4. **Quality Code:** Quality codes indicate if the temperature reading is valid. Values of 9999 denote missing data. The quality code must be in a specific set to be considered valid.

Script Explanation

Here's the detailed explanation of the script:

```
#!/usr/bin/env bash
for year in all/*
do
    echo -ne `basename $year .gz`"\t"
    gunzip -c $year | \
    awk '{ temp = substr($0, 88, 5) + 0;
    q = substr($0, 93, 1);
    if (temp !=9999 && q ~ /[01459]/ && temp > max) max = temp }
    END { print max }'
done
```

Sample Data in `1901.gz`:

 Copy code


```
00670119909999991950010104004...9999999N9+00300+9999999999...
00430119909999991950010110004...9999999N9-00500+9999999999...
00430119909999991950010116004...9999999N9+00800+9999999999...
```

This sample file contains three lines of weather data for the year 1901.

Script Breakdown and Execution

Step 1: Script Header

bash

 Copy code

```
#!/usr/bin/env bash
```

- **Explanation:** This line indicates the script should be run using the ``bash`` shell. It ensures the correct interpreter is used.

Step 2: Loop Through Files

bash

Copy code

```
for year in all/*  
do
```

- **Explanation:** This begins a loop that processes each file in the ``all/`` directory. Here, it will process ``1901.gz`` if it's present in the directory.

Step 3: Print Year (Without .gz Extension)

bash

Copy code

```
echo -ne `basename $year .gz`\t"
```

- **Explanation:** For ``1901.gz``, ``basename $year .gz`` removes the ``.gz`` extension, resulting in ``1901``. The ``echo -ne`` prints ``1901`` followed by a tab, but doesn't add a newline.

Output:

yaml

Copy code

```
1901
```

Step 4: Decompress and Process with ``awk``

bash

Copy code

```
gunzip -c $year | \  
awk '{ temp = substr($0, 88, 5) + 0;  
      q = substr($0, 93, 1);  
      if (temp !=9999 && q ~ /[01459]/ && temp > max) max = temp }  
END { print max }'
```


Here's what happens for each line in ``1901.gz``:

1. Decompress the File:

- Command: ``gunzip -c 1901.gz``
- Result: Outputs the decompressed data to the ``awk`` command.

2. Process Data with ``awk``:


Line 1:

 Copy code

```
0067011990999991950010104004...9999999N9+00300+9999999999...
```

- Extract Temperature: ``substr($0, 88, 5)`` gives ``+00300``, which converts to ``300``.
- Extract Quality Code: ``substr($0, 93, 1)`` gives ``+``.
- Valid Check: Temperature is valid and quality code is acceptable. ``max`` is updated to ``300``.


Line 2:

 Copy code

```
0043011990999991950010110004...9999999N9-00500+9999999999...
```

- Extract Temperature: ``substr($0, 88, 5)`` gives ``-00500``, which converts to ``-500``.
- Extract Quality Code: ``substr($0, 93, 1)`` gives ``+``.
- Valid Check: Temperature is valid and quality code is acceptable. ``max`` remains ``300`` (since ``300`` is greater than ``-500``).

Line 3:

 Copy code

```
0043011990999991950010116004...9999999N9+00800+9999999999...
```


- Extract Temperature: ``substr($0, 88, 5)`` gives ``+00800``, which converts to ``800``.
- Extract Quality Code: ``substr($0, 93, 1)`` gives ``+``.
- Valid Check: Temperature is valid and quality code is acceptable. ``max`` is updated to ``800``.

3. Print the Maximum Temperature:

- After processing all lines, ``awk`` prints the maximum temperature found, which is ``800``.

Output:

yaml

 Copy code


```
1901      800
```

Full Example of Script Execution

Assuming you have only ``1901.gz`` in the ``all/`` directory, the script would process it and provide the following output:

Complete Output:

yaml

 Copy code

```
1901      800
```

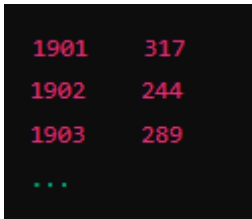
This means that for the year 1901, the highest recorded temperature from the sample dataset is 800 (which would be 80.0°C in the original scale if temperatures are scaled by a factor of 10).

5. AWK Script:

- ``temp = substr($0, 88, 5) + 0;`` extracts the temperature field (starting at position 88, length 5) and converts it to an integer.
- ``q = substr($0, 93, 1);`` extracts the quality code (starting at position 93, length 1).
- ``if (temp !=9999 && q ~ /[01459]/ && temp > max) max = temp`` checks if the temperature is valid (not 9999) and if the quality code is acceptable, then updates the maximum temperature.
- ``END { print max }`` prints the highest temperature found.

Example Output

The script outputs the maximum temperature for each year in the dataset. For example:



```
1901    317
1902    244
1903    289
...
```

Here, 317 for 1901 represents a maximum temperature of 31.7°C.

Performance and Parallelization

- **Single Machine:** The script runs on a single machine, which may take a considerable amount of time, especially for large datasets.
- **Parallel Processing:** To speed up, you can process different years in parallel. However, this introduces complexity:
 - **Unequal Workloads:** File sizes vary, so some processes finish earlier than others.
 - **Combining Results:** Requires combining results from each process. For fixed-size chunks, this involves additional steps to find the maximum temperature for each year.
- **Scaling Issues:** When data exceeds the capacity of a single machine, using frameworks like Hadoop helps manage distributed processing and addresses coordination and reliability challenges.

Analyzing the Data with Hadoop and Hadoop streaming in python

To take advantage of the parallel processing that Hadoop provides, we need to express our query as a MapReduce job. After some local, small-scale testing, we will be able to run it on a cluster of machines.

MapReduce is a programming model used for processing large data sets with a distributed algorithm on a cluster. It consists of two main phases:

1. **Map Phase:** Processes input data to produce a set of intermediate key-value pairs.
2. **Reduce Phase:** Aggregates intermediate results to produce the final output.

Input Data

For this example, the input data consists of weather records from the NCDC dataset, where each line includes temperature and other information.

Steps for MapReduce Job

1. Map Phase

Goal: Extract the year and temperature from each line of the dataset.

- **Input:** Raw NCDC data lines. Each line contains various fields including year and temperature.
- **Key-Value Pairs:** In Hadoop, each line of input data is represented as a key-value pair. The key is usually the line offset in the file (which is ignored in this case), and the value is the entire line of text.

Example Input Lines:

```
00670119909999991950051507004...9999999N9+00001+9999999999...
00430119909999991950051512004...9999999N9+00221+9999999999...
00430119909999991950051518004...9999999N9-00111+9999999999...
00430126509999991949032412004...0500001N9+01111+9999999999...
00430126509999991949032418004...0500001N9+00781+9999999999...
```

Map Function:

- **Extract Year and Temperature:** The map function processes each line to extract the year and temperature.
- **Filter:** The map function should filter out any invalid data (e.g., missing or erroneous temperatures).

Example Map Function (in Python):

```
def map_function(line):
    year = line[15:19] # Extract year (4 characters starting from position 15)
    temp_str = line[87:92] # Extract temperature (5 characters starting from position 87)

    try:
        temperature = int(temp_str) # Convert temperature to integer
    except ValueError:
        temperature = None # Handle invalid temperature

    if temperature is not None and temperature != 9999: # Filter out invalid temperatures
        yield (year, temperature)
```

Example map output

```
(1950, 0)
(1950, 22)
(1950, -11)
(1949, 111)
(1949, 78)
```

Shuffle and Sort Phase

- **Process:** After the map phase, Hadoop sorts and groups the output by key (year).
- **Intermediate Data:** The output pairs are sorted and grouped so that each key (year) has a list of associated values (temperatures).

Example Intermediate Data:

```
(1949, [111, 78])
(1950, [0, 22, -11])
```

Reduce Phase

Goal: For each year, find the maximum temperature from the list of temperatures.

Reduce Function:

- **Input:** Key-value pairs where the key is the year and the value is a list of temperatures.
- **Output:** Each key (year) with the maximum temperature value.

Example Reduce Function (in Python):

```
def reduce_function(year, temperatures):
    max_temp = max(temperatures) # Find the maximum temperature
    yield (year, max_temp)
```

Example Reduce Output:

```
(1949, 111)
(1950, 22)
```

Complete MapReduce Workflow

1. **Input:** Raw weather data lines.

2. **Map Phase:** Extract year and temperature, and emit (year, temperature) pairs.
3. **Shuffle and Sort:** Sort and group the intermediate data by year.
4. **Reduce Phase:** Find the maximum temperature for each year.

Visualization

The MapReduce flow is typically visualized in a diagram showing the input, map, shuffle/sort, and reduce phases:

- **Input Data:** Raw text lines.
- **Map Phase:** Extract year and temperature.
- **Shuffle and Sort:** Group by year.
- **Reduce Phase:** Compute maximum temperature per year.

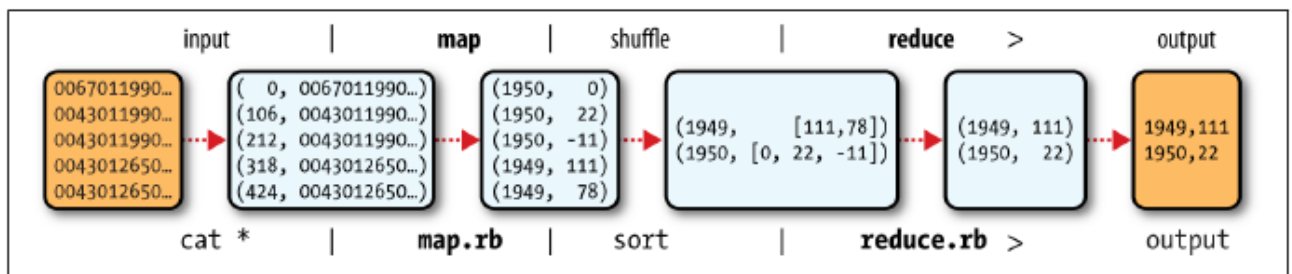


Figure 2-1. MapReduce logical data flow

Hadoop Streaming

Hadoop Streaming is a utility that comes with Hadoop, allowing users to create and run MapReduce jobs with any executable or script as the Mapper and Reducer. This is particularly useful for those who prefer writing in languages other than Java, such as Python, Ruby, or Perl.

Theory

Components of Hadoop Streaming

1. **Mapper:**
 - Processes input data and emits key-value pairs.
 - Runs in parallel across the cluster.
 - Example: Tokenizing text data and outputting word counts.
2. **Reducer:**
 - Takes the key-value pairs produced by the Mapper, groups them by key, and processes each group.

- Combines values based on the key to produce the final output.
- Example: Aggregating word counts by summing up counts for each word.

3. Hadoop Streaming Interface:

- Reads input data from HDFS or local filesystem.
- Passes the data to the Mapper.
- Collects output from the Mapper, sorts and groups it by key.
- Passes the grouped data to the Reducer.
- Writes the final output to HDFS.

Execution Flow

1. **Job Submission:** Submit a Hadoop Streaming job using the Hadoop command line.
2. **Data Processing:** Data is read from HDFS, passed through the Mapper and Reducer scripts.
3. **Output Storage:** The final output is written back to HDFS.

Example

Let's use a simple example of word count to illustrate Hadoop Streaming.

Sample Dataset

Suppose we have a text file (input.txt) with the following content:

```
hello world
hello hadoop
hadoop mapreduce
```


Mapper Script (`mapper.py`)

The Mapper script reads each line of input, splits it into words, and outputs each word with a count of `1`.

```
python

#!/usr/bin/env python
import sys

for line in sys.stdin:
    words = line.strip().split()
    for word in words:
        print(f"{word}\t1")
```

 Copy code

Reducer Script (reducer.py)

The Reducer script reads the key-value pairs output by the Mapper, aggregates counts for each word, and outputs the total count.

```
#!/usr/bin/env python
import sys

current_word = None
current_count = 0

for line in sys.stdin:
    word, count = line.strip().split('\t')
    count = int(count)

    if current_word == word:
        current_count += count
    else:
        if current_word:
            print(f"{current_word}\t{current_count}")
        current_word = word
        current_count = count

if current_word:
    print(f"{current_word}\t{current_count}")
```

Running Hadoop Streaming Job

Assuming you have Hadoop set up and the input data is in HDFS, you can run the Hadoop Streaming job with the following command:

```
hadoop jar /path/to/hadoop-streaming.jar \
  -input /path/to/input.txt \
  -output /path/to/output \
  -mapper /path/to/mapper.py \
  -reducer /path/to/reducer.py
```

Detailed Breakdown

1. Job Submission Command:

- **hadoop jar /path/to/hadoop-streaming.jar:** Specifies the Hadoop Streaming JAR file.

- **-input /path/to/input.txt:** Path to the input file in HDFS.
- **-output /path/to/output:** Path where the output will be written in HDFS.
- **-mapper /path/to/mapper.py:** Path to the Mapper script.
- **-reducer /path/to/reducer.py:** Path to the Reducer script.

2. Execution:

- Hadoop will distribute the Mapper and Reducer scripts across the cluster.
- Data is processed in parallel, with the Mapper script outputting intermediate key-value pairs and the Reducer script aggregating these pairs.

3. Output:

- The final word counts will be available in the specified output directory in HDFS.

Record of Climatological Observations
These data are quality controlled and may not be identical to the original observations.
Generated on 04/09/2015

Elev: 38 ft. Lat: 38.333° N Lon: 76.417° W
Station: PATUXENT RIVER, MD US GHCND:USC00186915
Observation Time Temperature: Unknown Observation Time Precipitation: Unknown

P r e l i m i n a r y	Y e a r	M o n t h	D a y	Temperature (F)		a t O b s e r v a t i o n	Precipitation(see **)					Evaporation		Soil Temperature (F)					
				24 hrs. ending at observation time			24 Hour Amounts ending at observation time				At Obs Time	24 Hour Wind Movement (mi)	Amount of Evap. (in)	4 in depth			8 in depth		
				Max.	Min.		Rain, melted snow, etc. (in)	F l a g	Snow, ice pellets, hail (in)	F l a g	Snow, ice pellets, hail, ice on ground (in)			Ground Cover (see *)	Max.	Min.	Ground Cover (see *)	Max.	Min.
	1989	4	1	52	41		0.00		0.0		0								
	1989	4	2	51	33		0.00		0.0		0								
	1989	4	3	71	47		0.21		0.0		0								
	1989	4	4	81	57		0.00		0.0		0								
	1989	4	5	71	50		0.84		0.0		0								
	1989	4	6	58	44		0.22		0.0		0								
	1989	4	7	50	36		1.29		T		0								
	1989	4	8	52	37		0.02		0.0		0								
	1989	4	9	60	43		0.00		0.0		0								
	1989	4	10	54	41		0.00		0.0		0								
	1989	4	11	49	35		0.00		0.0		0								
	1989	4	12	55	33		0.00		0.0		0								
	1989	4	13	58	36		0.00		0.0		0								
	1989	4	14	58	44		0.00		0.0		0								
	1989	4	15	53	50		1.08		0.0		0								
	1989	4	16	63	44		0.00		0.0		0								
	1989	4	17	70	44		0.00		0.0		0								
	1989	4	18	77	58		0.29		0.0		0								
	1989	4	19	60	44		0.03		0.0		0								
	1989	4	20	64	44		0.00		0.0		0								
	1989	4	21	65	41		0.00		0.0		0								
	1989	4	22	64	49		0.00		0.0		0								
	1989	4	23	59	40		0.00		0.0		0								
	1989	4	24	62	44		0.00		0.0		0								
	1989	4	25	73	47		0.00		0.0		0								
	1989	4	26	76	53		0.00		0.0		0								
	1989	4	27	72	51		0.00		0.0		0								
	1989	4	28	66	55		0.00		0.0		0								
	1989	4	29	64	55		0.77		0.0		0								
	1989	4	30	71	57		0.00		0.0		0								
Summary				62.6	45.1		4.75		0.0										

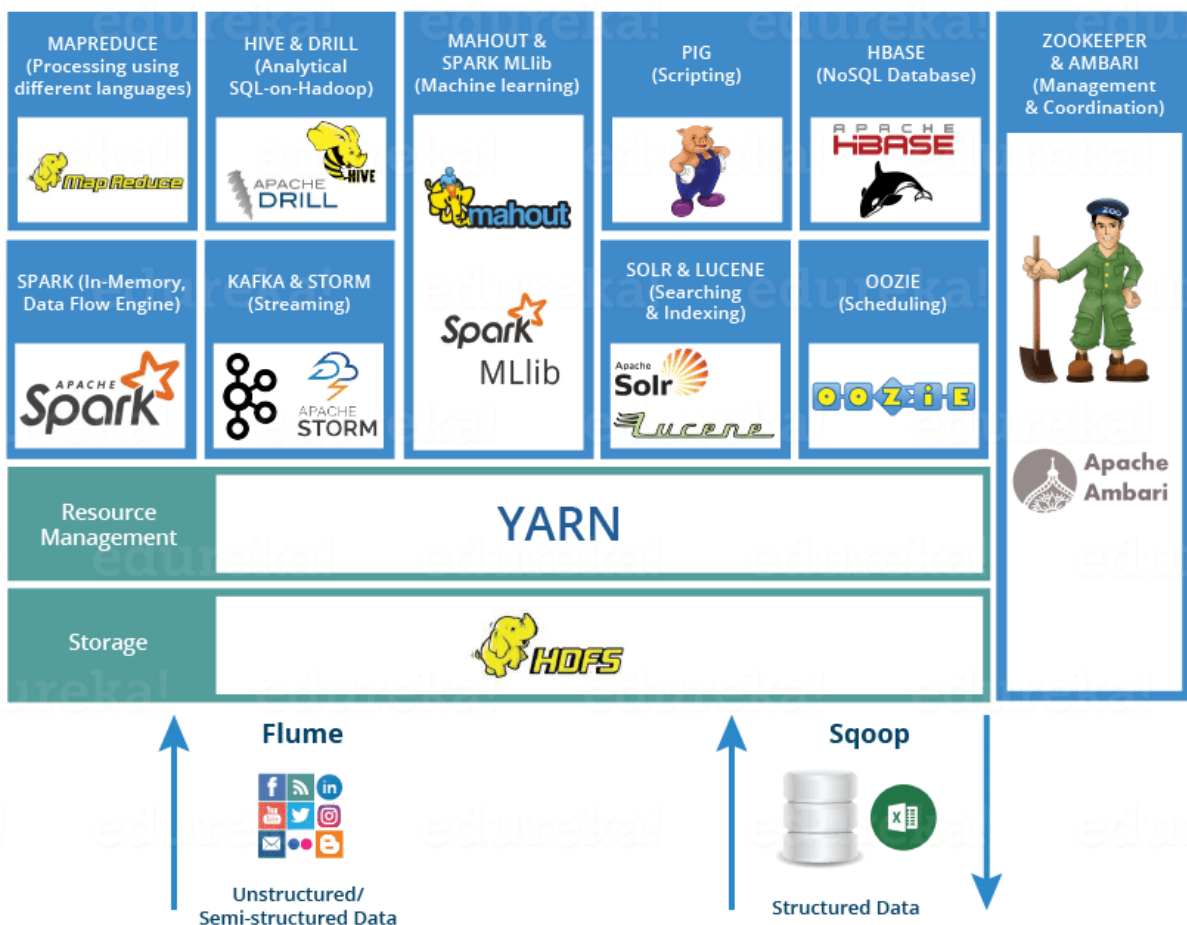
The "*" flags in Preliminary indicate the data have not completed processing and qualitycontrol and may not be identical to the original observation
Empty, or blank, cells indicate that a data observation was not reported.
*Ground Cover: 1=Grass; 2=Fallow; 3=Bare Ground; 4=Brome grass; 5=Sod; 6=Straw mulch; 7=Grass muck; 8=Bare muck; 0=Unknown
"s" This data value failed one of NCDC's quality control tests.
"T" values in the Precipitation category above indicate a TRACE value was recorded.
"A" values in the Precipitation Flag or the Snow Flag column indicate a multiday total, accumulated since last measurement, is being used.
Data value inconsistency may be present due to rounding calculations during the conversion process from SI metric units to standard imperial units.

HADOOP ECOSYSTEM

Hadoop Ecosystem is neither a programming language nor a service, it is a platform or framework which solves big data problems. You can consider it as a suite which encompasses a number of services (ingesting, storing, analyzing and maintaining) inside it.

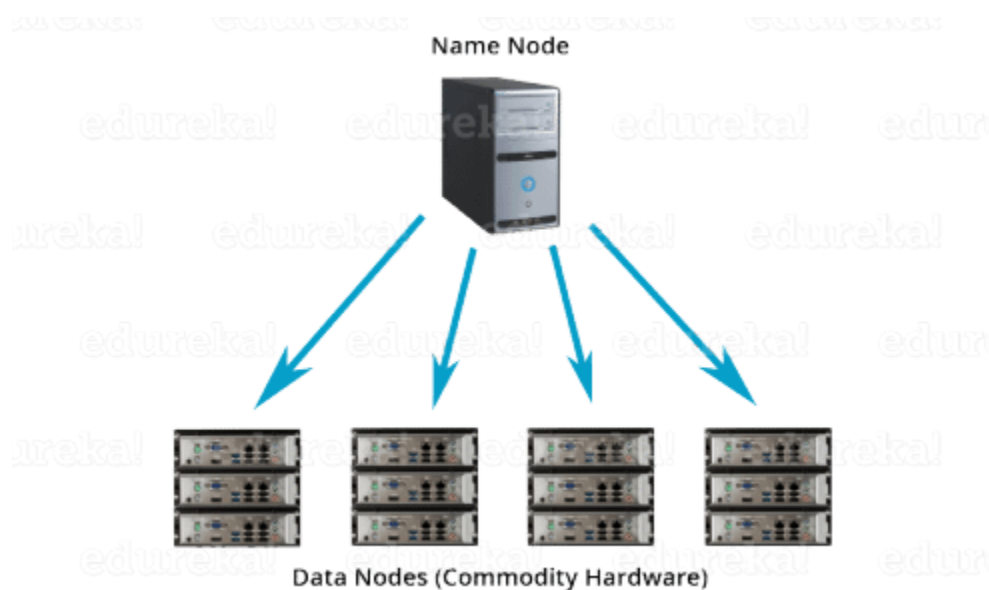
Below are the Hadoop components, that together form a Hadoop ecosystem.

- **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



HDFS

- Hadoop Distributed File System is the core component or you can say, the backbone of Hadoop Ecosystem.
- HDFS is the one, which makes it possible to store different types of large data sets (i.e. structured, unstructured and semi structured data).
- HDFS creates a level of abstraction over the resources, from where we can see the whole HDFS as a single unit.
- It helps us in storing our data across various nodes and maintaining the log file about the stored data (metadata).
- HDFS has two core components, i.e. **NameNode and DataNode**.
 1. The **NameNode** is the main node and it doesn't store the actual data. It contains metadata, just like a log file or you can say as a table of content. Therefore, it requires less storage and high computational resources.
 2. On the other hand, all your data is stored on the **DataNodes** and hence it requires more storage resources. These DataNodes are commodity hardware (like your laptops and desktops) in the distributed environment. That's the reason, why Hadoop solutions are very cost effective.
 3. You always communicate to the NameNode while writing the data. Then, it internally sends a request to the client to store and replicate data on various DataNodes.



YARN

Consider YARN as the brain of your Hadoop Ecosystem. It performs all your processing activities by allocating resources and scheduling tasks.

- It has two major components, i.e. **Resource Manager and Node Manager**.
 1. **Resource Manager** is again a main node in the processing department.
 2. It receives the processing requests, and then passes the parts of requests to corresponding Node Managers accordingly, where the actual processing takes place.
 3. **Node Managers** are installed on every Data Node. It is responsible for execution of task on every single Data Node.
- 1. **Schedulers:** Based on your application resource requirements, Schedulers perform scheduling algorithms and allocates the resources.
 2. **Applications Manager:** While Applications Manager accepts the job submission, negotiates to containers (i.e. the Data node environment where process executes) for executing the application specific Application Master and monitoring the progress. ApplicationMasters are the daemons which reside on DataNode and communicates to containers for execution of tasks on each DataNode.
 3. ResourceManager has two components: Schedulers and application manager

MAPREDUCE



It is the core component of processing in a Hadoop Ecosystem as it provides the logic of processing. In other words, MapReduce is a software framework which helps in writing applications that processes large data sets using distributed and parallel algorithms inside Hadoop environment.

- In a MapReduce program, **Map() and Reduce()** are two functions.
 1. The **Map function** performs actions like filtering, grouping and sorting.
 2. While **Reduce function** aggregates and summarizes the result produced by map function.
 3. The result generated by the Map function is a key value pair (K, V) which acts as the input for Reduce function.

Student	Department	Count	(Key, Value), Pair
Student 1	D1	1	(D1, 1)
Student 2	D1	1	(D1, 1)
Student 3	D1	1	(D1, 1)
Student 4	D2	1	(D2, 1)
Student 5	D2	1	(D2, 1)
Student 6	D3	1	(D3, 1)
Student 7	D3	1	(D3, 1)

Let us take the above example to have a better understanding of a MapReduce program.

We have a sample case of students and their respective departments. We want to calculate the number of students in each department. Initially, Map program will execute and calculate the students appearing in each department, producing the key value pair as mentioned above. This key value pair is the input to the Reduce function. The Reduce function will then aggregate each department and calculate the total number of students in each department and produce the given result.

Department	Total Student
D1	3
D2	2
D3	2

APACHE PIG



- PIG has two parts: **Pig Latin**, the language and **the pig runtime**, for the execution environment. You can better understand it as Java and JVM.
- It supports *pig latin* language, which has SQL like command structure.

10 line of pig latin = approx. 200 lines of Map-Reduce Java code

But don't be shocked when I say that at the back end of Pig job, a map-reduce job executes.

- The compiler internally converts pig latin to MapReduce. It produces a sequential set of MapReduce jobs, and that's an abstraction (which works like black box).
- PIG was initially developed by Yahoo.
- It gives you a platform for building data flow for ETL (Extract, Transform and Load), processing and analyzing huge data sets.

How Pig works?

In PIG, first the load command, loads the data. Then we perform various functions on it like grouping, filtering, joining, sorting, etc. At last, either you can dump the data on the screen or you can store the result back in HDFS.

APACHE HIVE



- Facebook created HIVE for people who are fluent with SQL. Thus, HIVE makes them feel at home while working in a Hadoop Ecosystem.
- Basically, HIVE is a data warehousing component which performs reading, writing and managing large data sets in a distributed environment using SQL-like interface.

$$\text{HIVE} + \text{SQL} = \text{HQL}$$

- The query language of Hive is called Hive Query Language(HQL), which is very similar like SQL.
- It has 2 basic components: **Hive Command Line and JDBC/ODBC driver.**
- The **Hive Command line** interface is used to execute HQL commands.
- While, Java Database Connectivity (**JDBC**) and Object Database Connectivity (**ODBC**) is used to establish connection from data storage.
- Secondly, Hive is highly scalable. As, it can serve both the purposes, i.e. large data set processing (i.e. Batch query processing) and real time processing (i.e. Interactive query processing).
- It supports all primitive data types of SQL.
- You can use predefined functions, or write tailored user defined functions (UDF) also to accomplish your specific needs.

APACHE MAHOUT



Now, let us talk about Mahout which is renowned for machine learning. Mahout provides an environment for creating machine learning applications which are scalable.

Machine learning algorithms allow us to build self-learning machines that evolve by itself without being explicitly programmed. Based on user behaviour, data patterns and past experiences it makes important future decisions. You can call it a descendant of Artificial Intelligence (AI).

What Mahout does?

It performs **collaborative filtering**, **clustering** and **classification**. Some people also consider **frequent item set mining** as Mahout's function. Let us understand them individually:

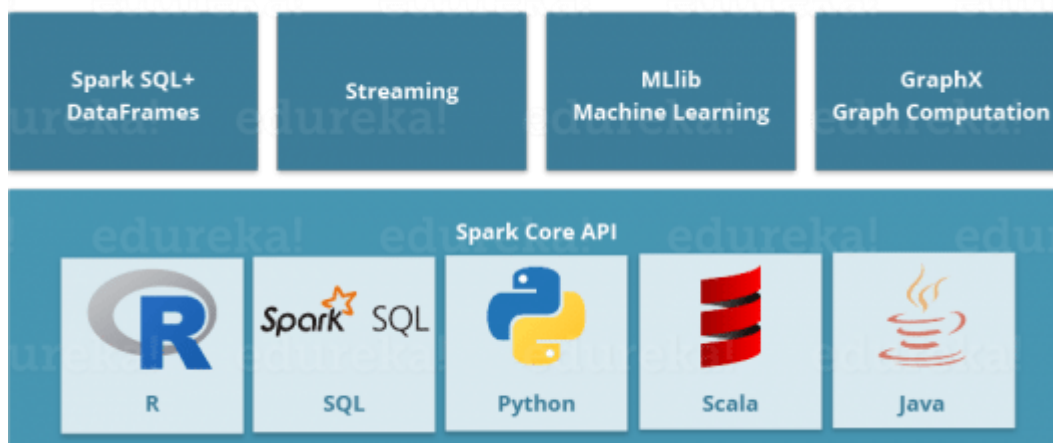
1. **Collaborative filtering:** Mahout mines user behaviors, their patterns and their characteristics and based on that it predicts and make recommendations to the users. The typical use case is E-commerce website.
2. **Clustering:** It organizes a similar group of data together like articles can contain blogs, news, research papers etc.
3. **Classification:** It means classifying and categorizing data into various sub-departments like articles can be categorized into blogs, news, essay, research papers and other categories.
4. **Frequent item set mining:** Here Mahout checks, which objects are likely to be appearing together and make suggestions, if they are missing. For example, cell phone and cover are brought together in general. So, if you search for a cell phone, it will also recommend you the cover and cases.

Mahout provides a command line to invoke various algorithms. It has a predefined set of library which already contains different inbuilt algorithms for different use cases.

APACHE SPARK



- Apache Spark is a framework for real time data analytics in a distributed computing environment.
- The Spark is written in Scala and was originally developed at the University of California, Berkeley.
- It executes in-memory computations to increase speed of data processing over Map-Reduce.
- It is 100x faster than Hadoop for large scale data processing by exploiting in-memory computations and other optimizations. Therefore, it requires high processing power than Map-Reduce.



As you can see, Spark comes packed with high-level libraries, including support for R, SQL, Python, Scala, Java etc. These standard libraries increase the seamless integrations in complex workflow. Over this, it also allows various sets of services to integrate with it like MLlib, GraphX, SQL + Data Frames, Streaming services etc. to increase its capabilities.

. Apache Spark best fits for real time processing, whereas Hadoop was designed to store unstructured data and execute batch processing over it. When we combine, Apache Spark's ability, i.e. high processing speed, advance analytics and multiple integration support with Hadoop's low cost operation on commodity hardware, it gives the best results.

That is the reason why, Spark and Hadoop are used together by many companies for processing and analyzing their Big Data stored in HDFS.

APACHE HBASE



- HBase is an open source, non-relational distributed database. In other words, it is a NoSQL database.
- It supports all types of data and that is why, it's capable of handling anything and everything inside a Hadoop ecosystem.
- It is modelled after Google's BigTable, which is a distributed storage system designed to cope up with large data sets.
- The HBase was designed to run on top of HDFS and provides BigTable like capabilities.
- It gives us a fault tolerant way of storing sparse data, which is common in most Big Data use cases.
- The HBase is written in Java, whereas HBase applications can be written in REST, Avro and Thrift APIs.

For better understanding, let us take an example. You have billions of customer emails and you need to find out the number of customers who has used the word complaint in their emails. The request needs to be processed quickly (i.e. at real time). So, here we are handling a large data set while retrieving a small amount of data. For solving these kind of problems, HBase was designed.

APACHE DRILL



Apache Drill is used to drill into any kind of data. It's an open source application which works with distributed environment to analyze large data sets.

- It is a replica of Google Dremel.
- It supports different kinds NoSQL databases and file systems, which is a powerful feature of Drill. For example: Azure Blob Storage, Google Cloud Storage, HBase, MongoDB, MapR-DB HDFS, MapR-FS, Amazon S3, Swift, NAS and local files.

So, basically the main aim behind Apache Drill is to provide scalability so that we can process petabytes and exabytes of data efficiently (or you can say in minutes).

- The main power of Apache Drill lies in *combining a variety of data stores just by using a single query*.
- Apache Drill basically follows the ANSI SQL.
- It has a powerful scalability factor in supporting millions of users and serve their query requests over large scale data.

APACHE ZOOKEEPER



- Apache Zookeeper is the coordinator of any Hadoop job which includes a combination of various services in a Hadoop Ecosystem.
- Apache Zookeeper coordinates with various services in a distributed environment.

Before Zookeeper, it was very difficult and time consuming to coordinate between different services in Hadoop Ecosystem. The services earlier had many problems with interactions like common configuration while synchronizing data. Even if the services are configured, changes in the configurations of the services make it complex and difficult to handle. The grouping and naming was also a time-consuming factor.

Due to the above problems, Zookeeper was introduced. It saves a lot of time by performing **synchronization, configuration maintenance, grouping and naming**.

Although it's a simple service, it can be used to build powerful solutions.

APACHE OOZIE



Consider Apache Oozie as a clock and alarm service inside Hadoop Ecosystem. For Apache jobs, Oozie has been just like a scheduler. It schedules Hadoop jobs and binds them together as one logical work.

There are two kinds of Oozie jobs:

1. **Oozie workflow:** These are sequential set of actions to be executed. You can assume it as a relay race. Where each athlete waits for the last one to complete his part.
2. **Oozie Coordinator:** These are the Oozie jobs which are triggered when the data is made available to it. Think of this as the response-stimuli system in our body. In the same manner as we respond to an external stimulus, an Oozie coordinator responds to the availability of data and it rests otherwise.

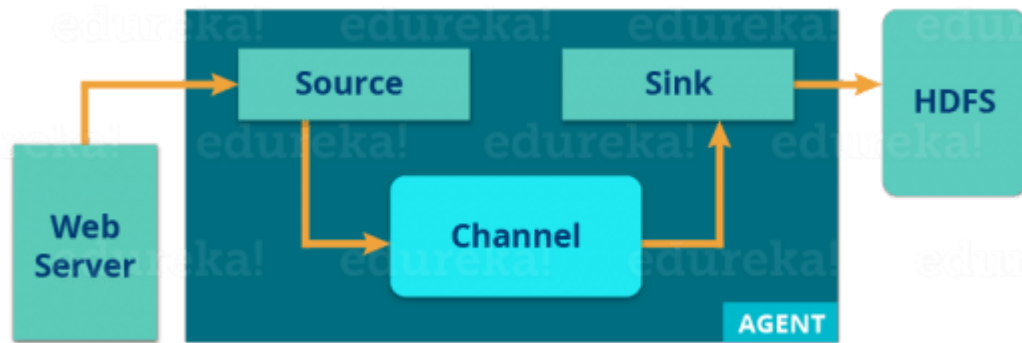
APACHE FLUME



Ingesting data is an important part of our Hadoop Ecosystem.

- The Flume is a service which helps in ingesting unstructured and semi-structured data into HDFS.
- It gives us a solution which is reliable and distributed and helps us in **collecting, aggregating** and **moving large amount of data sets**.
- It helps us to ingest online streaming data from various sources like network traffic, social media, email messages, log files etc. in HDFS.

Now, let us understand the architecture of Flume from the below diagram:



There is a **Flume agent** which ingests the streaming data from various data sources to HDFS. From the diagram, you can easily understand that the web server indicates the data source. Twitter is among one of the famous sources for streaming data.

The flume agent has 3 components: **source, sink and channel**.

1. **Source:** it accepts the data from the incoming streamline and stores the data in the channel.
2. **Channel:** it acts as the local storage or the primary storage. A Channel is a temporary storage between the source of data and persistent data in the HDFS.
3. **Sink:** Then, our last component i.e. Sink, collects the data from the channel and commits or writes the data in the HDFS permanently.

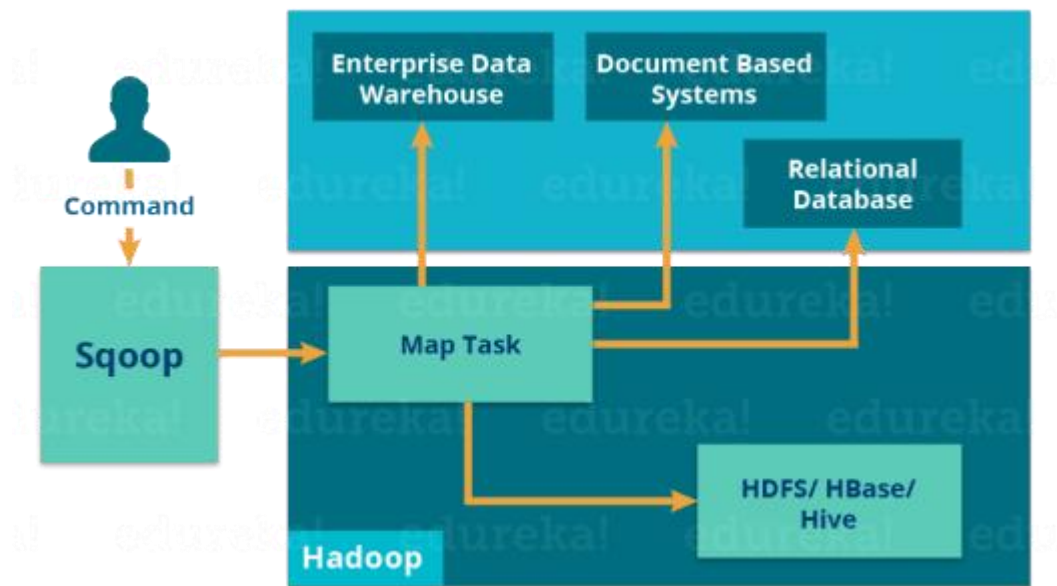
APACHE SQOOP



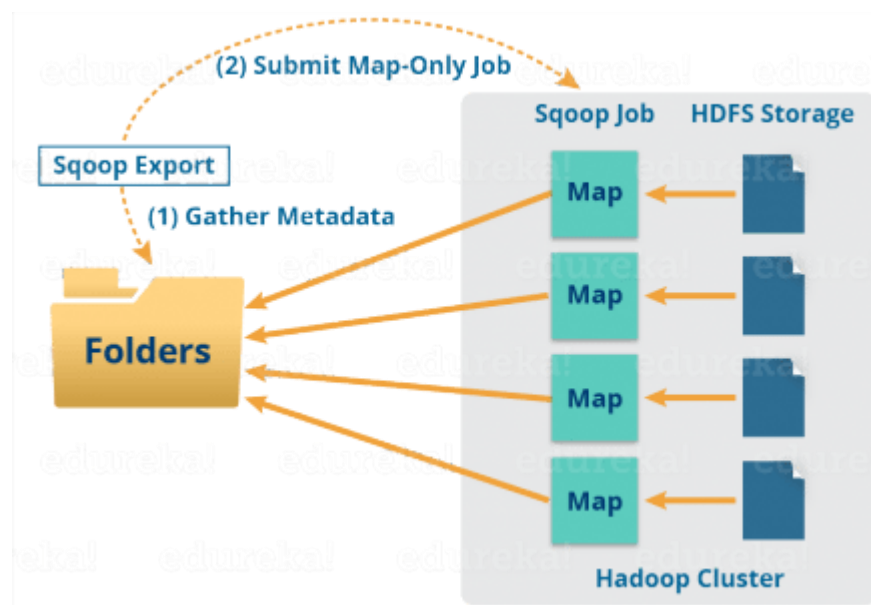
The major difference between Flume and Sqoop is that:

- Flume only ingests unstructured data or semi-structured data into HDFS.
- While Sqoop can import as well as export structured data from RDBMS or Enterprise data warehouses to HDFS or vice versa.

Let us understand how Sqoop works using the below diagram:



When we submit Sqoop command, our main task gets divided into sub tasks which is handled by individual Map Task internally. Map Task is the sub task, which imports part of data to the Hadoop Ecosystem. Collectively, all Map tasks imports the whole data.



Export also works in a similar manner.

When we submit our Job, it is mapped into Map Tasks which brings the chunk of data from HDFS. These chunks are exported to a structured data destination. Combining all these exported chunks of data, we receive the whole data at the destination, which in most of the cases is an RDBMS (MYSQL/Oracle/SQL Server).

APACHE SOLR & LUCENE



Apache Solr and Apache Lucene are the two services which are used for searching and indexing in Hadoop Ecosystem.

- Apache Lucene is based on Java, which also helps in spell checking.
- If Apache Lucene is the engine, Apache Solr is the car built around it. Solr is a complete application built around Lucene.
- It uses the Lucene Java search library as a core for search and full indexing.

APACHE AMBARI



Ambari is an Apache Software Foundation Project which aims at making Hadoop ecosystem more manageable.



It includes software for **provisioning, managing and monitoring** Apache Hadoop clusters.

The Ambari provides:

1. **Hadoop cluster provisioning:**
 - It gives us step by step process for installing Hadoop services across a number of hosts.
 - It also handles configuration of Hadoop services over a cluster.
2. **Hadoop cluster management:**
 - It provides a central management service for starting, stopping and re-configuring Hadoop services across the cluster.
3. **Hadoop cluster monitoring:**
 - For monitoring health and status, Ambari provides us a dashboard.
 - The **Amber Alert framework** is an alerting service which notifies the user, whenever the attention is needed. For example, if a node goes down or low disk space on a node, etc.

At last, I would like to draw your attention on three things importantly:

1. Hadoop Ecosystem owes its success to the whole developer community, many big companies like Facebook, Google, Yahoo, University of California (Berkeley) etc. have contributed their part to increase Hadoop's capabilities.
2. Inside a Hadoop Ecosystem, knowledge about one or two tools (Hadoop components) would not help in building a solution. You need to learn a set of Hadoop components, which works together to build a solution.
3. Based on the use cases, we can choose a set of services from Hadoop Ecosystem and create a tailored solution for an organization.