SRINIVASAN COLLEGE OF ARTS & SCIENCE

(Affiliated Bharathidasan University, Tiruchirappalli)

PERAMBALUR-621212

Department of Computer Science

& Information Technology

COURSE MATERIAL

Subject	**:**	**Big Data Analytics**
Subject Code	**:**	**P16CSE5A**
Class	**:**	**II - M.Sc (CS)**
Semester	**:**	**IV**

ELECTIVE COURSE V

BIG DATA ANALYTICS

Objective:

To impart knowledge in Fundamentals, Big Data Analytics, Technologies and databases, Hadoop and Map Reduce Fundamentals

Unit I

Introduction to big data: Data, Characteristics of data and Types of digital data: Unstructured, Semi-structured and Structured, Sources of data, Working with unstructured data, Evolution and Definition of big data, Characteristics and Need of big data, Challenges of big data, Data environment versus big data environment

Unit II

Big data analytics: Overview of business intelligence, Data science and Analytics, Meaning and Characteristics of big data analytics, Need of big data analytics, Classification of analytics, Challenges to big data analytics, Importance of big data analytics, Basic terminologies in big data environment

Unit III

Big data technologies and Databases: Introduction to NoSQL, Uses, Features and Types, Need, Advantages, Disadvantages and Application of NoSQL, Overview of NewSQL, Comparing SQL, NoSQL and NewSQL, Introduction to MongoDB and its needs, Characteristics of MongoDB, Introduction of apache cassandra and its needs, Characteristics of Cassandra

Unit IV

Hadoop foundation for analytics: History, Needs, Features, Key advantage and Versions of Hadoop, Essential of Hadoop ecosystems, RDBMS versus Hadoop, Key aspects and Components of Hadoop, Hadoop architectures

Unit V

HadoopMapReduce and YARN framework: Introduction to MapReduce, Processing data with Hadoop using MapReduce, Introduction to YARN, Components, Need and Challenges of YARN, Dissecting YARN, MapReduce application, Data serialization and Working with common serialization formats, Big data serialization formats

Text Book

Seema Acharya and Subhashini Chellappan, "Big Data and Analytics", Wiley India Pvt. Ltd., 2016

Reference Books

1. 1."Big Data" by Judith Hurwitz, Alan Nugent, Dr. Fern Halper and Marcia Kaufman, Wiley Publications, 2014.
2. 2."Big Data Imperatives : Enterprise Big Data Warehouse, BI Implementations and Analytics" by Soumendra Mohanty, Madhu Jagadeesh and Harsha Srivatsa, Apress Media, Springer Science + Business Media New York, 2013
3. "Mining of Massive Datasets", Anand Rajaraman, Jure Leskovec, Jeffery D. Ullman, Springer, July 2013.
4. "Hadoop: The definitive Guide", Tom White, O'Reilly Media, 2010.

BIGDATA ANALYTICS

UNIT-I

Big data:

What is Big Data?

Big Data is a collection of large datasets that cannot be processed using traditional computing techniques. For example, the volume of data Facebook or Youtube need require it to collect and manage on a daily basis, can fall under the category of Big Data. However, Big Data is not only about scale and volume, it also involves one or more of the following aspects − Velocity, Variety, Volume, and Complexity.

Characteristics of Big Data

3 'V's of Big Data –

Variety,
Velocity,
Volume.
\

1) Variety

Variety of Big Data refers to structured, unstructured, and semistructured data that is gathered from multiple sources. While in the past, data could only be collected from spreadsheets and databases, today data comes in an array of forms such as emails, PDFs, photos, videos, audios, SM posts, and so much more.

2) Velocity

Velocity essentially refers to the speed at which data is being created in real-time. In a broader prospect, it comprises the rate of change, linking of incoming data sets at varying speeds, and activity bursts.

3) Volume

We already know that Big Data indicates huge 'volumes' of data that is being generated on a daily basis from various sources like social media platforms, business processes, machines, networks, human interactions, etc. Such a large amount of data are stored in data warehouses.

Types of Big Data

Structured

By structured data, we mean data that can be processed, stored, and retrieved in a fixed format. It refers to highly organized information that can be readily and seamlessly stored and accessed from a database by simple search engine algorithms. **For instance, the employee table in a company database will be structured as the employee details, their job positions, their salaries, etc.,** will be present in an organized manner.

Unstructured

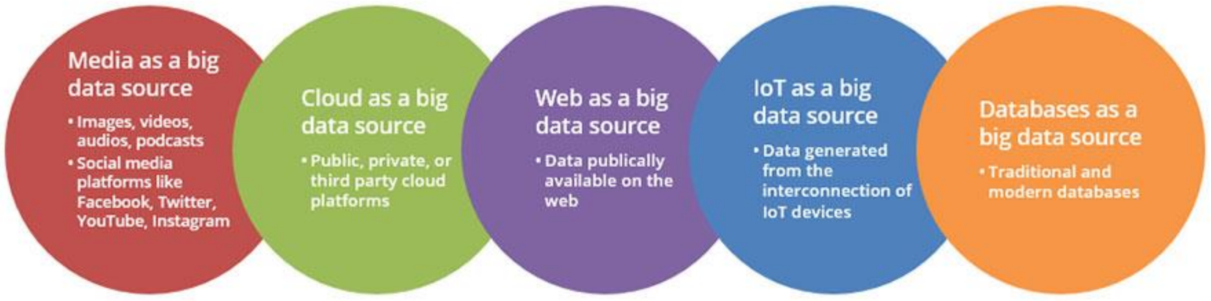
Unstructured data refers to the data that lacks any specific form or structure whatsoever. This makes it very difficult and time-consuming to process and analyze unstructured data. Email is an example of unstructured data.

Semi-structured

Semi-structured data pertains to the data containing both the formats mentioned above, that is, structured and unstructured data. To be precise, it refers to the data that although has not been classified under a particular repository (database), yet contains vital information or tags that segregate individual elements within the data.

Sources of big data

Voluminous amounts of big data make it crucial for businesses to differentiate, for the purpose of effectiveness, the disparate big data sources available


Media as a big data source
• Images, videos, audios, podcasts
• Social media platforms like Facebook, Twitter, YouTube, Instagram
Cloud as a big data source
• Public, private, or third party cloud platforms
Web as a big data source
• Data publically available on the web
IoT as a big data source
• Data generated from the interconnection of IoT devices
Databases as a big data source
• Traditional and modern databases

Media as a big data source
Media is the most popular source of big data, as it provides valuable insights on consumer preferences and changing trends. Since it is self-broadcasted and crosses all physical and demographical barriers, it is the fastest way for businesses to get an in-depth overview of their target audience, draw patterns and conclusions, and enhance their decision-making. Media includes social media and interactive platforms, like Google, Facebook, Twitter, YouTube, Instagram, as well as generic media like images, videos, audios, and podcasts that provide quantitative and qualitative insights on every aspect of user interaction.

Cloud as a big data source
Today, companies have moved ahead of traditional data sources by shifting their data on the cloud. Cloud storage accommodates structured and unstructured data and provides business with real-time information and on-demand insights. The main attribute of cloud computing is its flexibility and scalability. As big data

can be stored and sourced on public or private clouds, via networks and servers, cloud makes for an efficient and economical data source.

The web as a big data source

The public web constitutes big data that is widespread and easily accessible. Data on the Web or 'Internet' is commonly available to individuals and companies alike. Moreover, web services such as Wikipedia provide free and quick informational insights to everyone. The enormity of the Web ensures for its diverse usability and is especially beneficial to start-ups and SME's, as they don't have to wait to develop their own big data infrastructure and repositories before they can leverage big data.

IoT as a big data source

Machine-generated content or data created from IoT constitute a valuable source of big data. This data is usually generated from the sensors that are connected to electronic devices. The sourcing capacity depends on the ability of the sensors to provide real-time accurate information. IoT is now gaining momentum and includes big data generated, not only from computers and smartphones, but also possibly from every device that can emit data. With IoT, data can now be sourced from medical devices, vehicular processes, video games, meters, cameras, household appliances, and the like.

Databases as a big data source

Businesses today prefer to use an amalgamation of traditional and modern databases to acquire relevant big data. This integration paves the way for a hybrid data model and requires low investment and IT infrastructural costs. Furthermore, these databases are deployed for several business intelligence purposes as well. These databases can then provide for the extraction of insights that are used to drive business profits. Popular databases include a variety of data sources, such as MS Access, DB2, Oracle, SQL, and Amazon Simple, among others.

Working with unstructured data

The process of extracting and analyzing data amongst extensive big data sources is a complex process and can be frustrating and time-consuming. These complications can be resolved if organizations encompass all the necessary considerations of big data, take into account relevant data sources, and deploy them in a manner which is well tuned to their organizational goals.

Before the modern day ubiquity of online and mobile applications, databases processed straightforward, structured data. Data models were relatively simple and described a set of relationships between different data types in the database.

Unstructured data, in contrast, refers to data that doesn't fit neatly into the traditional row and column structure of relational databases. Examples of unstructured data include: emails, videos, audio files, web pages, and social media messages. In today's world of Big Data, most of the data that is created is unstructured with some estimates of it being more than 95% of all data generated.

As a result, enterprises are looking to this new generation of databases, known as NoSQL, to address unstructured data. MongoDB stands as a leader in this movement with over 10 million downloads and hundreds of thousands of deployments. As a document database with flexible schema, MongoDB was built specifically to handle unstructured data. MongoDB's flexible data model allows for development without a predefined schema which resonates particularly when most of the data in your system is unstructured.

The Evolution of Big Data

To truly understand the implications of Big Data analytics, one has to reach back into the annals of computing history, specifically business intelligence (BI) and scientific computing. The ideology behind Big Data can most likely be tracked back to the days before the age of computers, when unstructured data were the norm (paper records) and analytics was in its infancy. Perhaps the first Big Data challenge came in the form of the 1880 U.S. census, when the information concerning approximately 50 million people had to be gathered, classified, and reported on.

With the 1880 census, just counting people was not enough information for the U.S. government to work with—particular elements, such as age, sex, occupation, education level, and even the "number of insane people in household," had to be accounted for. That information had intrinsic value to the process, but only if it could be tallied, tabulated, analyzed, and presented. New methods of relating the data to other data collected came into being, such as associating occupations with geographic areas, birth rates with education levels, and countries of origin with skill sets.

The 1880 census truly yielded a mountain of data to deal with, yet only severely limited technology was available to do any of the analytics. The problem of Big Data could not be solved for the 1880 census, so it took over seven years to manually tabulate and report on the data.

With the 1890 census, things began to change, ...

Challenges of Big Data

It must be pretty clear by now that while talking about big data one can't ignore the fact that there are some obvious challenges associated with it. So moving forward in this blog, let's address some of those challenges.

- **Quick Data Growth**

Data growing at such a quick rate is making it a challenge to find insights from it. There is more and more data generated every second from which the data that is actually relevant and useful has to be picked up for further analysis.

- **Storage**

Such large amount of data is difficult to store and manage by organizations without appropriate tools and technologies.

- **Syncing Across Data Sources**

This implies that when organisations import data from different sources the data from one source might not be up to date as compared to the data from another source.

- **Security**

Huge amount of data in organisations can easily become a target for advanced persistent threats, so here lies another challenge for organisations to keep their data secure by proper authentication, data encryption, etc.

- **Unreliable Data**

We can't deny the fact that big data can't be 100 percent accurate. It might contain redundant or incomplete data, along with contradictions.

- **Miscellaneous Challenges**

These are some other challenges that come forward while dealing with big data, like **the integration** of data, **skill** and **talent availability, solution expenses** and **processing a large amount of data** in time and with accuracy so that the data is available for data consumers whenever they need it.

Data Environment versus Big Data Environment

Below are the lists of points, describe the comparisons between Small Data and Big Data.

BasisOf Comparison	**Small Data**	**Big Data**
Definition	Data that is 'small' enough for human comprehension.In a volume and format that makes it accessible, informative and actionable	Data sets that are so large or complex that traditional data processing applications cannot deal with them
Data Source	● Data from traditional enterprise systems like ○ Enterprise resource planning ○ Customer relationship management(CRM) ● Financial Data like general ledger data ● Payment transaction data from website	● Purchase data from point-of-sale ● Clickstream data from websites ● GPS stream data – Mobility data sent to a server ● Social media – Facebook, Twitter
Volume	Most cases in a range of tens or hundreds of GB.Some case few TBs (1 TB=1000 GB)	More than a few Terabytes (TB)
Velocity	● Controlled and steady data flow	● Data can arrive at very fast speeds.

	● Data accumulation is slow	● Enormous data can accumulate within very short periods of time
Variety	Structured data in tabular format with fixed schema and semi-structured data in JSON or XML format	High variety data sets which include Tabular data,Text files, Images, Video, Audio,XML,JSON,Logs,Sensor data etc.
Veracity (Quality of data)	Contains less noise as data collected in a controlled manner.	Usually, the quality of data not guaranteed. Rigorous data validation is required before processing.
Value	Business Intelligence, Analysis, and Reporting	Complex data mining for prediction, recommendation, pattern finding, etc.
Time Variance	Historical data equally valid as data represent solid business interactions	In some cases, data gets older soon(Eg fraud detection).
Data Location	Databases within an enterprise, Local servers, etc.	Mostly in distributed storages on Cloud or in external file systems.
Infrastructure	Predictable resource allocation.Mostly vertically scalable hardware	More agile infrastructure with a horizontally scalable architecture. Load on the system varies a lot.

UNIT-II

Bigdata Anlytics:

Overview of Business Intelligence:

Business Intelligence (BI) applications are decision support tools that enable real-time, interactive access to and analysis of mission-critical corporate information. BI applications bridge the gaps between information silos in an organization. Sophisticated analytical capabilities have access to such corporate information resources as data warehouses, transaction processing applications, and enterprise applications like Enterprise Resource Planning (ERP). BI enables users to access and leverage vast amounts of data, providing valuable insight into potential opportunities and areas for business process refinement.

BI applications can be classified as follows:

- Personalized Dashboards for Process Monitoring and Highlighting Exceptions
- Decision Support with Drill-Down and "What-If" Analysis
- Data-Mining to Understand and Discover Patterns and Behaviors
- Automated Agents to Drive Rule-Based Business Strategy via Integrated Processes

Investments made in an EPM (Enterprise Process Management) implementation can be very expensive, so it is imperative that every asset is leveraged. Youngsoft's team of experienced professionals provide a wide range of services across the suite of EPM products, consistently delivering solutions that reduce costs, increase profits, and improve overall efficiency.

Benefits:

- Cost Reduction during the Implementation Process
- Retained Knowledge after Completion of the Implementation
- End User Step by Step Training during the Implementation Process
- Providing Tier-One Production Support during Post-Implementation Process
- Shadowing the Implementation

What is Data Science?

Data Science is the combination of statistics, mathematics, programming, problem-solving, capturing data in ingenious ways, the ability to look at things differently, and the activity of cleansing, preparing, and aligning the data.

In simple terms, it is the umbrella of techniques used when trying to extract insights and information from data.

Applications of Data Science

- **Internet Search**

  Search engines make use of data science algorithms to deliver the best results for search queries in a fraction of seconds.

- **Digital Advertisements**

  The entire digital marketing spectrum uses the data science algorithms - from display banners to digital billboards. This is the mean reason for digital ads getting higher CTR than traditional advertisements.

- **Recommender Systems**

  The recommender systems not only make it easy to find relevant products from billions of products available but also adds a lot to user-experience. A lot of companies use this system to promote their products and suggestions in accordance with the user's demands and relevance of information. The recommendations are based on the user's previous search results.

Need of Bigdata Analytics

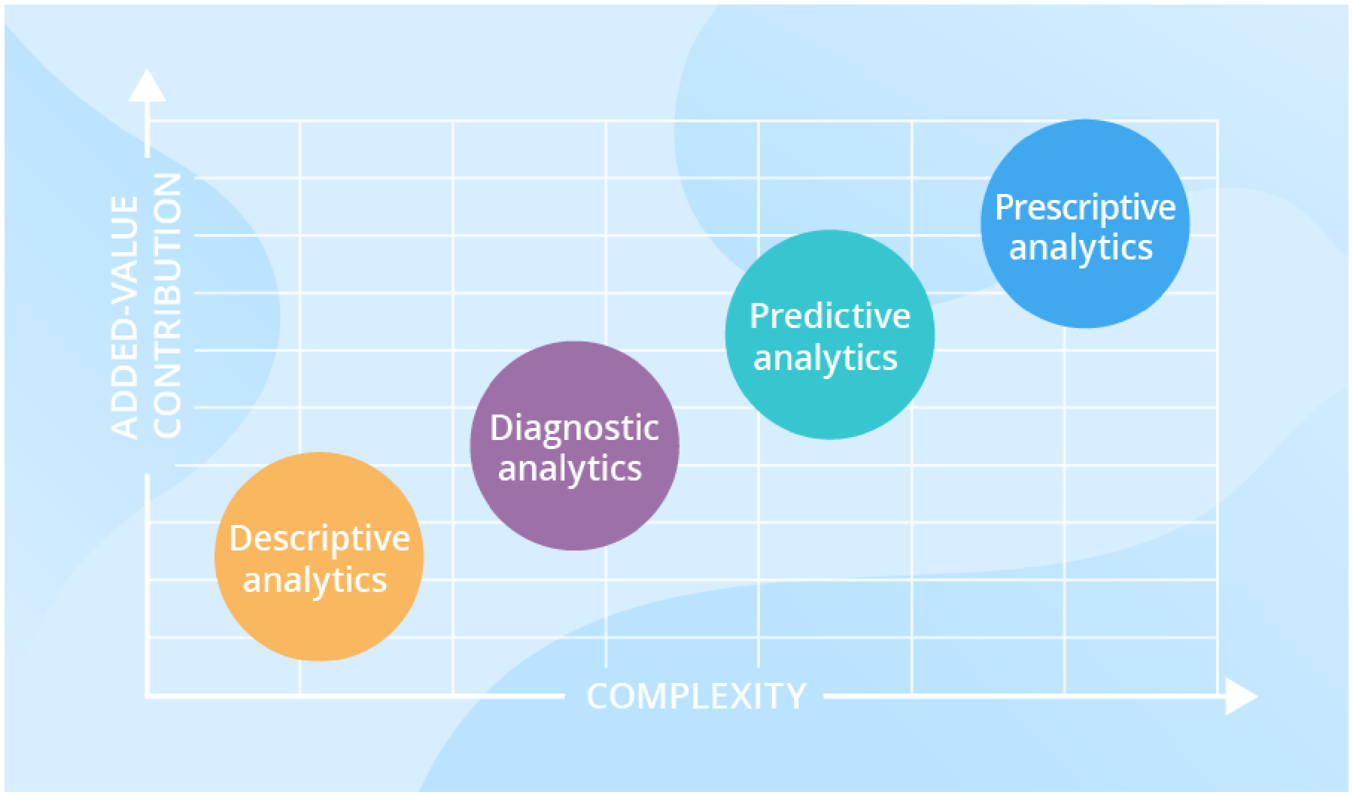
Why is big data analytics important?

Big data analytics helps organizations harness their data and use it to identify new opportunities. That, in turn, leads to smarter business moves, more efficient operations, higher profits and happier customers. In his report *Big Data in Big Companies*, IIA Director of Research Tom Davenport interviewed more than 50 businesses to understand how they used big data. He found they got value in the following ways:

1. **Cost reduction.** Big data technologies such as Hadoop and cloud-based analytics bring significant cost advantages when it comes to storing large amounts of data – plus they can identify more efficient ways of doing business.
2. **Faster, better decision making.** With the speed of Hadoop and in-memory analytics, combined with the ability to analyze new sources of data, businesses are able to analyze information immediately – and make decisions based on what they've learned.
3. **New products and services.** With the ability to gauge customer needs and satisfaction through analytics comes the power to give customers what they want. Davenport points out that with big data analytics, more companies are creating new products to meet customers' needs.

Types of data analytics

There are 4 different types of analytics.



Descriptive analytics

Descriptive analytics answers the question of *what happened*. Let us bring an example from ScienceSoft's practice: having analyzed monthly revenue and income per product group, and the total quantity of metal parts produced per month, a manufacturer was able to answer a series of 'what happened' questions and decide on focus product categories.

Descriptive analytics juggles raw data from multiple data sources to give valuable insights into the past. However, these findings simply signal that something is wrong or right, without explaining why. For this reason, our data consultants don't recommend highly data-driven companies to settle for descriptive analytics only, they'd rather combine it with other types of data analytics.

Diagnostic analytics

At this stage, historical data can be measured against other data to answer the question of *why something happened*. For example, you can check ScienceSoft's BI demo to see how a retailer can drill the sales and gross profit down to categories to find out why they missed their net profit target. Another flashback to our data analytics projects: in the healthcare industry, customer segmentation coupled with several filters applied (like diagnoses and prescribed medications) allowed identifying the influence of medications.

Diagnostic analytics gives in-depth insights into a particular problem. At the same time, a company should have detailed information at their disposal, otherwise, data collection may turn out to be individual for every issue and time-consuming.

Predictive analytics

Predictive analytics tells *what is likely to happen.* It uses the findings of descriptive and diagnostic analytics to detect clusters and exceptions, and to predict future trends, which makes it a valuable tool for forecasting. Check ScienceSoft's case study to get details on how advanced data analytics allowed a leading FMCG company to predict what they could expect after changing brand positioning.

Predictive analytics belongs to advanced analytics types and brings many advantages like sophisticated analysis based on machine or deep learning and proactive approach that predictions enable. However, our data consultants state it clearly: forecasting is just an estimate, the accuracy of which highly depends on data quality and stability of the situation, so it requires careful treatment and continuous optimization.

Prescriptive analytics

The purpose of prescriptive analytics is to literally prescribe *what action to take* to eliminate a future problem or take full advantage of a promising trend. An example of prescriptive analytics from our project portfolio: a multinational company was able to identify opportunities for repeat purchases based on customer analytics and sales history.

Prescriptive analytics uses advanced tools and technologies, like machine learning, business rules and algorithms, which makes it sophisticated to implement and manage. Besides, this state-of-the-art type of data analytics requires not only historical internal data but also external information due to the nature of algorithms it's based on. That is why, before deciding to adopt prescriptive analytics, ScienceSoft strongly recommends weighing the required efforts against an expected added value.

Big Data Analytics Challenges

Need For Synchronization Across Disparate Data Sources

As data sets are becoming bigger and more diverse, there is a big challenge to incorporate them into an analytical platform. If this is overlooked, it will create gaps and lead to wrong messages and insights.

2. Acute Shortage Of Professionals Who Understand Big Data Analysis

The analysis of data is important to make this voluminous amount of data being produced in every minute, useful. With the exponential rise of data, a huge demand for big data scientists and Big Data analysts has been created in the market. It is important for business organizations to hire a data scientist having skills that are varied as the job of a data scientist is multidisciplinary. Another major challenge faced by businesses is the shortage of professionals who understand Big Data analysis. There is a sharp shortage of data scientists in comparison to the massive amount of data being produced.

3. Getting Meaningful Insights Through The Use Of Big Data Analytics

It is imperative for business organizations to gain important insights from Big Data analytics, and also it is important that only the relevant department has access to this information. A big challenge faced by the companies in the Big Data analytics is mending this wide gap in an effective manner.

4. Getting Voluminous Data Into The Big Data Platform

It is hardly surprising that data is growing with every passing day. This simply indicates that business organizations need to handle a large amount of data on daily basis. The amount and variety of data available these days can overwhelm any data engineer and that is why it is considered vital to make data accessibility easy and convenient for brand owners and managers.

5. Uncertainty Of Data Management Landscape

With the rise of Big Data, new technologies and companies are being developed every day. However, a big challenge faced by the companies in the Big Data analytics is to find out which technology will be best suited to them without the introduction of new problems and potential risks.

6. Data Storage And Quality

Business organizations are growing at a rapid pace. With the tremendous growth of the companies and large business organizations, increases the amount of data produced. The storage of this massive amount of data is becoming a real challenge for everyone. Popular data storage options like data lakes/ warehouses are commonly used to gather and store large quantities of unstructured and structured data in its native format. The real problem arises when a data lakes/ warehouse try to combine unstructured and inconsistent data from diverse sources, it encounters errors. Missing data, inconsistent data, logic conflicts, and duplicates data all result in data quality challenges.

7. Security And Privacy Of Data

Once business enterprises discover how to use Big Data, it brings them a wide range of possibilities and opportunities. However, it also involves the potential risks associated with big data when it comes to the privacy and the security of the data. The Big Data tools used for analysis and storage utilizes the data disparate sources. This eventually leads to a high risk of exposure of the data, making it vulnerable. Thus, the rise of voluminous amount of data increases privacy and security concerns.

The Importance of Big Data Analytics

Driven by specialized analytics systems and software, as well as high-powered computing systems, big data analytics offers various business benefits, including:

- New revenue opportunities
- More effective marketing
- Better customer service
- Improved operational efficiency
- Competitive advantages over rivals

Big data analytics applications enable big data analysts, data scientists, predictive modelers, statisticians and other analytics professionals to analyze growing volumes of structured transaction data, plus other forms of data that are often left untapped by conventional BI and analytics programs. This encompasses a mix of semi-structured and unstructured data -- for example, internet clickstream data, web server logs, social media content, text from customer emails and survey responses, mobile phone records, and machine data captured by sensors connected to the internet of things (IoT).

Basic Terminologies in big data environment

we will discuss the terminology related to Big Data ecosystem. This will give you a complete understanding of Big Data and its terms.

Over time, Hadoop has become the nucleus of the Big Data ecosystem, where many new technologies have emerged and have got integrated with Hadoop. So it’s important that, first, we understand and appreciate the nucleus of modern Big Data architecture.

The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers, using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage.

Components of the Hadoop Ecosystem

Let’s begin by looking at some of the components of the Hadoop ecosystem:

Hadoop Distributed File System (HDFS™):

This is a distributed file system that provides high-throughput access to application data. Data in a Hadoop cluster is broken down into smaller pieces (called blocks) and distributed throughout the cluster. In this method, the map and reduce functions can be executed on smaller subsets of your larger data sets, and this provides the scalability needed for Big Data processing.

M apReduce:

MapReduce is a programming model specifically implemented for processing large data sets on Hadoop cluster. This is the core component of the Hadoop framework, and it is the only execution engine available for Hadoop 1.0.

The MapReduce framework consists of two parts:

1. A function called ‘Map’, which allows different points in the distributed cluster to distribute their work.

2. A function called ‘Reduce’, which is designed to reduce the final form of the clusters’ results into one output.

The main advantage of the MapReduce framework is its fault tolerance, where periodic reports from each node in the cluster are expected as soon as the work is completed.

The MapReduce framework is inspired by the ‘Map’ and ‘Reduce’ functions used in functional programming. The computational processing occurs on data stored in a file system or within a database, which takes a set of input key values and produces a set of output key values.

Each day, numerous MapReduce programs and MapReduce jobs are executed on Google’s clusters. Programs are automatically parallelized and executed on a large cluster of commodity machines.

Map Reduce is used in distributed grep, distributed sort, Web link-graph reversal, Web access log stats, document clustering, Machine Learning and statistical machine translation.

Pig:

Pig is a data flow language that allows users to write complex MapReduce operations in simple scripting language. Then Pig then transforms those scripts into a MapReduce job.

Hive:

Apache Hive data warehouse software facilitates querying and managing large datasets residing in distributed storage. Hive provides a mechanism for querying the data using a SQL-like language called HiveQL. At the same time, this language also allows traditional map/reduce programmers to plug in their custom mappers and reducers when it is inconvenient or inefficient to express this logic in HiveQL.

Sqoop:

Enterprises that use Hadoop often find it necessary to transfer some of their data from traditional relational database management systems (RDBMSs) to the Hadoop ecosystem.

Sqoop, an integral part of Hadoop, can perform this transfer in an automated fashion. Moreover, the data imported into Hadoop can be transformed with MapReduce before exporting them back to the RDBMS. Sqoop can also generate Java classes for programmatically interacting with imported data. Sqoop uses a connector-based architecture that allows it to use plugins to connect with external databases.

Flume:

Flume is a service for streaming logs into Hadoop. Apache Flume is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of streaming data into the Hadoop Distributed File System (HDFS).

Storm:

Storm is a distributed, real-time computation system for processing large volumes of high-velocity data. Storm is extremely fast and can process over a million records per second per node on a cluster of modest size. Enterprises harness this speed and combine it with other data-access applications in Hadoop to prevent undesirable events or to optimize positive outcomes.

Kafka:

Apache Kafka supports a wide range of use cases such as a general-purpose messaging system for scenarios where high throughput, reliable delivery, and horizontal scalability are important. Apache Storm and Apache HBase both work very well in combination with Kafka.

Oozie:

Oozie is a workflow scheduler system to manage Apache Hadoop jobs. The Oozie Workflow jobs are Directed Acyclical Graphs (DAGs) of actions, whereas the Oozie Coordinator jobs are recurrent Oozie Workflow jobs triggered by time (frequency) and data availability.

Oozie is integrated with the rest of the Hadoop stack and supports several types of Hadoop jobs out of the box (such as Java map-reduce, Streaming map-reduce, Pig, Hive, Sqoop and Distcp) as well as system-specific jobs (such as Java programs and shell scripts). Oozie is a scalable, reliable and extensible system.

Spark:

Apache Spark is a fast, in-memory data processing engine for distributed computing clusters like Hadoop. It runs on top of existing Hadoop clusters and accesses the Hadoop data store (HDFS).

Spark can be integrated with Hadoop's 2 YARN architecture, but cannot be used with Hadoop 1.0.

Apache Solr:

Apache Solr is a fast, open-source Java search server. Solr enables you to easily create search engines that search websites, databases, and files for Big Data

Apache Yarn:

Apache Hadoop YARN (Yet Another Resource Negotiator) is a cluster management technology. YARN is one of the key features in the second-generation Hadoop 2 version of the Apache Software Foundation's open-source distributed processing framework. Originally described by Apache as a redesigned resource manager, YARN is now characterized as a large-scale, distributed operating system for big data applications.

Tez:

Tez is an execution engine for Hadoop that allows jobs to meet the demands for fast response times and extreme throughput at petabyte scale. Tez represents computations as a dataflow graphs and can be used with Hadoop 2 YARN.

Apache Drill:

Apache Drill is an open-source, low-latency query engine for Hadoop that delivers secure, interactive SQL analytics at petabyte scale. With the ability to discover schemas on the go, Drill is a pioneer in delivering self-service data exploration capabilities on data stored in multiple formats in files or NoSQL databases. By adhering to ANSI SQL standards, Drill does not require a learning curve and integrates seamlessly with visualization tools.

Apache Phoenix:

Apache Phoenix takes your SQL query, compiles it into a series of HBase scans, and co-ordinates the running of those scans to produce a regular JDBC result set. Apache Phoenix enables OLTP and operational analytics in Hadoop for low-latency applications by combining the best of both worlds. Apache Phoenix is fully integrated with other Hadoop products such as Spark, Hive, Pig, Flume, and Map Reduce.

Cloud Computing:

Cloud Computing is a type of computing that relies on sharing computing resources rather than having local servers or personal devices to handle applications. Cloud Computing is comparable to grid computing, a type of computing where the unused processing cycles of all computers in a network are harnessed to solve problems that are too processor-intensive for any single machine.

In Cloud Computing, the word cloud (also phrased as "the cloud") is used as a metaphor for the Internet, hence the phrase cloud computing means "a type of Internet-based computing" in which different services such as servers, storage and applications are delivered to an organization's computers and devices via the Internet.

NoSQL:

The NoSQL database, also called Not Only SQL, is an approach to data management and database design that's useful for very large sets of distributed data. This database system is non-relational, distributed, open-source and horizontally scalable. NoSQL seeks to solve the scalability and big-data performance issues that relational databases weren't designed to address.

Apache Cassandra:

Apache Cassandra is an open-source distributed database system designed for storing and managing large amounts of data across commodity servers. Cassandra can serve as both a real-time operational data store for online transactional applications and a read-intensive database for large-scale business intelligence (BI) systems.

SimpleDB:

Amazon Simple Database Service (SimpleDB), also known as a key value data store, is a highly available and flexible non-relational database that allows developers to request and store data, with minimal database management and administrative responsibility.

This service offers simplified access to a data store and query functions that let users instantly add data and effortlessly recover or edit that data.

SimpleDB is best used by customers who have a relatively simple data requirement, like data storage. For example, a business might use cookies to track visitors that visit its company website. Some applications might read the cookies to get the visitor's identifier and look up the feeds they're interested in. Amazon SimpleDB gives users the option to store tens of attributes for a million customers, but not thousands of attributes for a single customer.

Google BigTable:

Google's BigTable is a distributed, column-oriented data store created by Google Inc. to handle very large amounts of structured data associated with the company's Internet search and Web services operations.

BigTable was designed to support applications requiring massive scalability; from its first iteration, the technology was intended to be used with petabytes of data. The database was designed to be deployed on clustered systems and uses a simple data model that Google has described as "a sparse, distributed, persistent multidimensional sorted map." Data is assembled in order by row key, and indexing of the map is arranged according to row, column keys, and timestamps. Here, compression algorithms help achieve high capacity.

MongoDB:

MongoDB is a cross-platform, document-oriented database. Classified as a NoSQL database, MongoDB shuns the traditional table-based relational database structure in favor of JSON-like documents with dynamic schemas (MongoDB calls the format BSON), making the integration of data in certain types of applications easier and faster.

MongoDB is developed by MongoDB Inc. and is published as free and open-source software under a combination of the GNU Affero General Public License and the Apache License. As of July 2015, MongoDB is the fourth most popular type of database management system, and the most popular for document stores.

HBase:

Apache HBase (Hadoop DataBase) is an open-source NoSQL database that runs on the top of the database and provides real-time read/write access to those large data sets.

HBase scales linearly to handle huge data sets with billions of rows and millions of columns, and it easily combines data sources that use a wide variety of different structures and schema. HBase is natively integrated with Hadoop and works seamlessly alongside other data access engines through YARN.

Neo4j:

Neo4j is a graph database management system developed by Neo Technology, Inc. Neo4j is described by its developers as an ACID-compliant transactional database with native graph storage and processing. According to db-engines.com, Neo4j is the most popular graph database.

Couch DB:

CouchDB is a database that completely embraces the web. It stores your data with JSON documents. It accesses your documents and queries your indexes with your web browser, via HTTP. It indexes, combines, and transforms your documents with JavaScript.

CouchDB works well with modern web and mobile apps. You can even serve web apps directly out of CouchDB. You can distribute your data, or your apps, efficiently using CouchDB’s incremental replication. CouchDB supports master-master setups with automatic conflict detection.

UNIT-III

Bigdata Terminologies and Database:

Introduction to NOSQL:

What is NoSQL Databases?

NoSQL is commonly referred to as not only SQL, non SQL, or non relational databases. These non SQL databases are built for extreme high throughput using key value pairs versus relational databases with relative dependence. Using loose dependences and quick indexes NoSQL databases are perfect for Streaming Analytics and IoT applications because data can quickly be stored and referenced.

SQL	**NOSQL**
Relational dependences	Loose dependences
Updates to tables time consuming	Updates to tables on-demand
Performance dependent on queries & indexes	Performance depend on hardware & network
Rigid scaling	Elastic scaling
Costly	Cost Efficient

Reasons to Use a NoSQL Database

- Storing large volumes of data without structure. A **NoSQL** database doesn't limit storable data types. ...
- **Using** cloud computing and storage. Cloud-based storage is a great solution, but it requires data to be easily spread across multiple servers for scaling. ...
- Rapid development.

Features of NoSQL

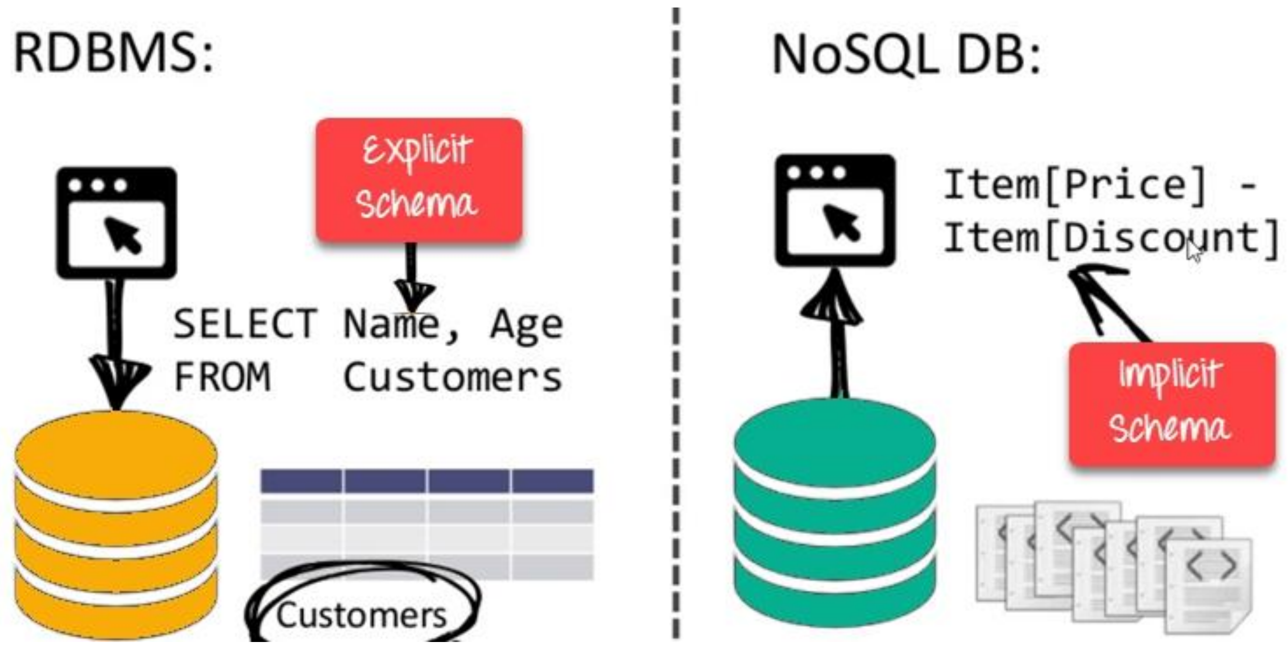
Non-relational

- NoSQL databases never follow the relational model
- Never provide tables with flat fixed-column records
- Work with self-contained aggregates or BLOBs
- Doesn't require object-relational mapping and data normalization
- No complex features like query languages, query planners,

  referential integrity joins, ACID

Schema-free

- NoSQL databases are either schema-free or have relaxed schemas
- Do not require any sort of definition of the schema of the data
- Offers heterogeneous structures of data in the same domain



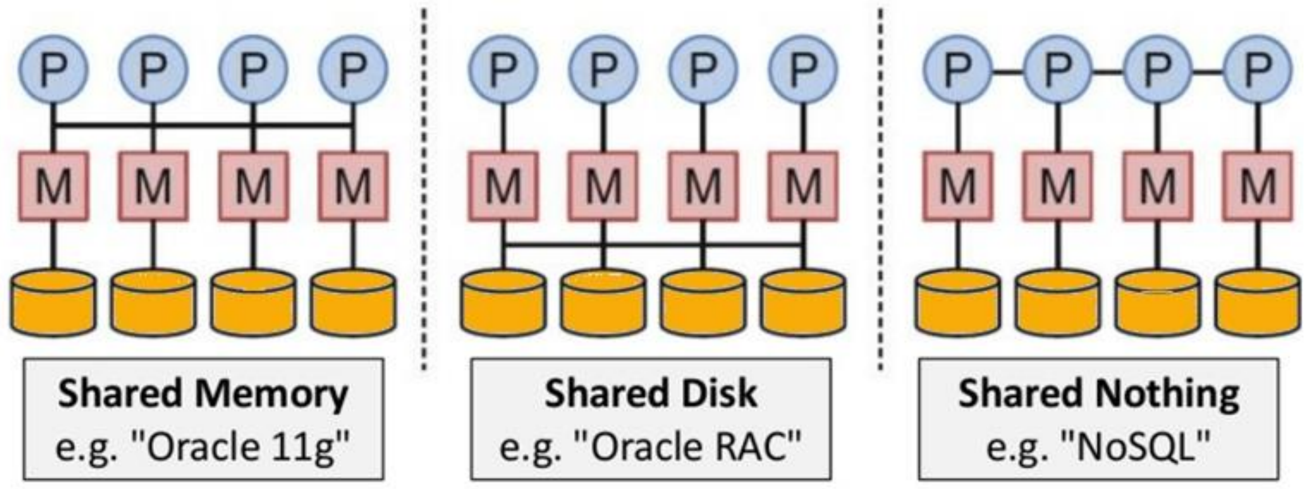
NoSQL is Schema-Free

Simple API

- Offers easy to use interfaces for storage and querying data provided
- APIs allow low-level data manipulation & selection methods
- Text-based protocols mostly used with HTTP REST with JSON
- Mostly used no standard based query language
- Web-enabled databases running as internet-facing services

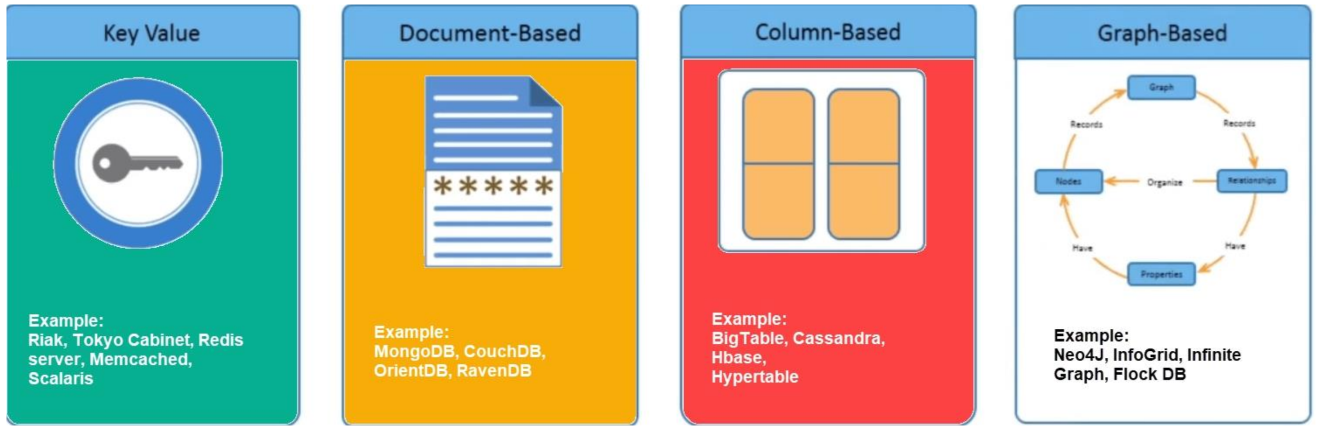
Distributed

- Multiple NoSQL databases can be executed in a distributed fashion
- Offers auto-scaling and fail-over capabilities
- Often ACID concept can be sacrificed for scalability and throughput
- Mostly no synchronous replication between distributed nodes Asynchronous Multi-Master Replication, peer-to-peer, HDFS Replication
- Only providing eventual consistency
- Shared Nothing Architecture. This enables less coordination and higher distribution.



NoSQL is Shared Nothing.

Types of NoSQL Databases


Key Value
Example:
Riak, Tokyo Cabinet, Redis server, Memcached, Scalaris
Document-Based
*****
Example:
MongoDB, CouchDB, OrientDB, RavenDB
Column-Based
Example:
BigTable, Cassandra, Hbase, Hypertable
Graph-Based
Graph
Records
Records
Nodes
Organize
Relationships
Have
Have
Properties
Example:
Neo4J, InfoGrid, Infinite Graph, Flock DB

There are mainly four categories of NoSQL databases. Each of these categories has its unique attributes and limitations. No specific database is better to solve all problems. You should select a database based on your product needs.

Let see all of them:

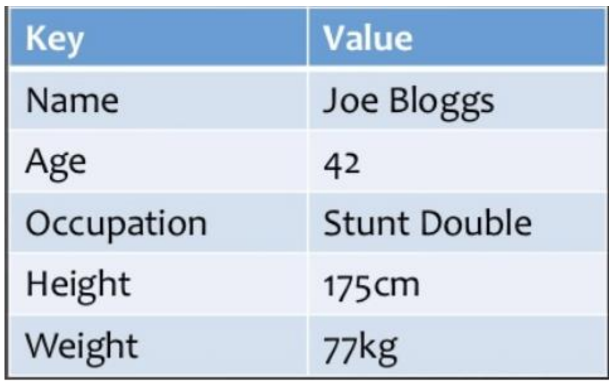
- Key-value Pair Based
- Column-oriented Graph
- Graphs based
- Document-oriented

Key Value Pair Based

Data is stored in key/value pairs. It is designed in such a way to handle lots of data and heavy load.

Key-value pair storage databases store data as a hash table where each key is unique, and the value can be a JSON, BLOB(Binary Large Objects), string, etc.

For example, a key-value pair may contain a key like "Website" associated with a value like "Guru99".



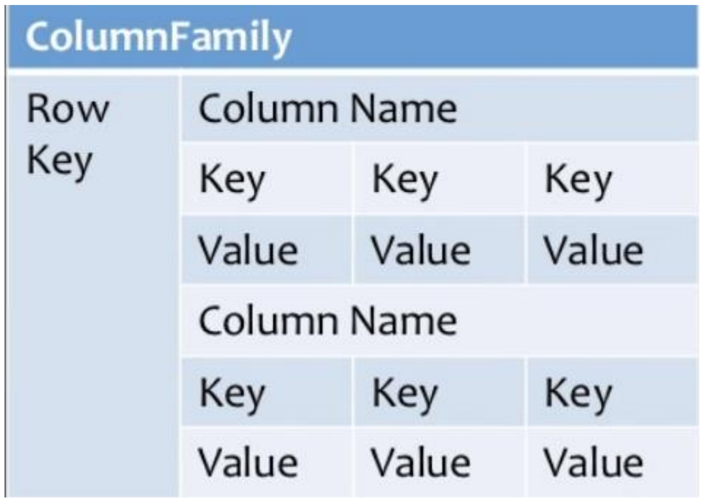
Key	Value
Name	Joe Bloggs
Age	42
Occupation	Stunt Double
Height	175cm
Weight	77kg

It is one of the most basic types of NoSQL databases. This kind of NoSQL database is used as a collection, dictionaries, associative arrays, etc. Key value stores help the developer to store schema-less data. They work best for shopping cart contents.

Redis, Dynamo, Riak are some examples of key-value store DataBases. They are all based on Amazon's Dynamo paper.

Column-based

Column-oriented databases work on columns and are based on BigTable paper by Google. Every column is treated separately. Values of single column databases are stored contiguously.



Column based NoSQL database

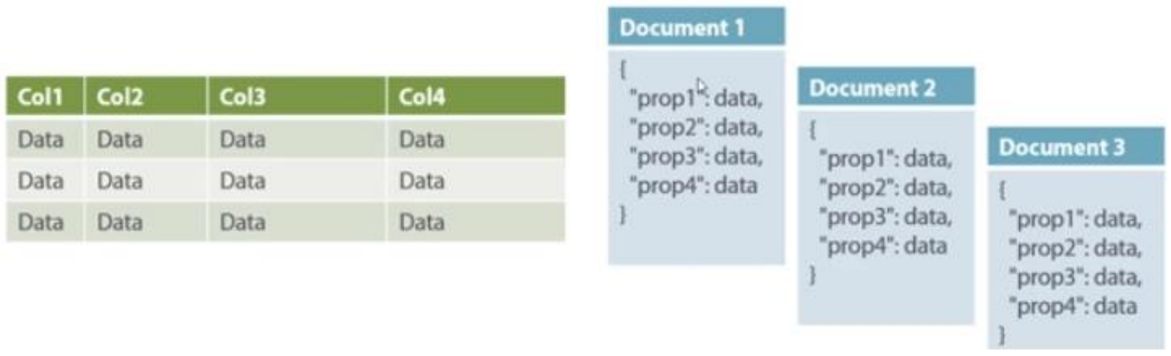
They deliver high performance on aggregation queries like SUM, COUNT, AVG, MIN etc. as the data is readily available in a column.

Column-based NoSQL databases are widely used to manage data warehouses, business intelligence, CRM, Library card catalogs,

HBase, Cassandra, HBase, Hypertable are examples of column based database.

Document-Oriented:

Document-Oriented NoSQL DB stores and retrieves data as a key value pair but the value part is stored as a document. The document is stored in JSON or XML formats. The value is understood by the DB and can be queried.


Col1
Col2
Col3
Col4
Data
Document 1
{ "prop1": data, "prop2": data, "prop3": data, "prop4": data }
Document 2
{ "prop1": data, "prop2": data, "prop3": data, "prop4": data }
Document 3
{ "prop1": data, "prop2": data, "prop3": data, "prop4": data }

Relational Vs. Document

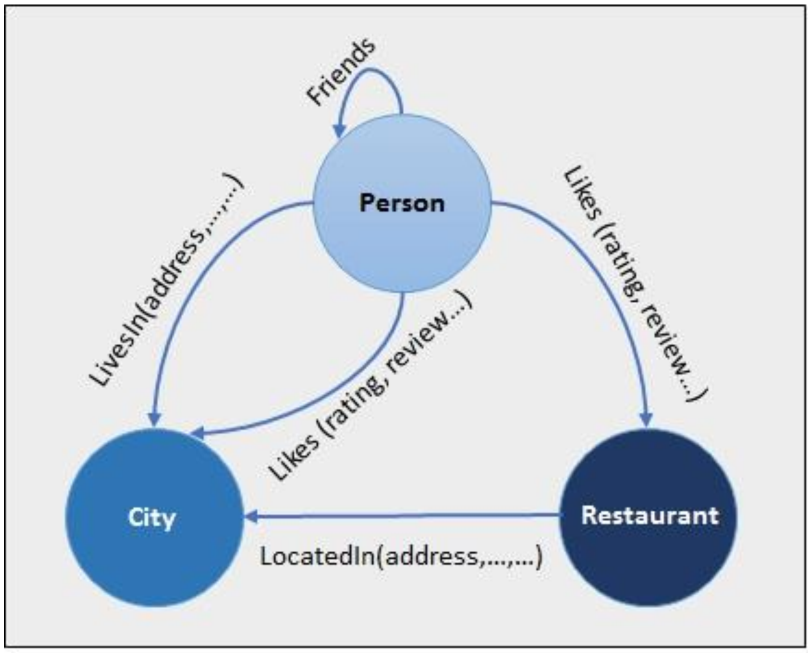
In this diagram on your left you can see we have rows and columns, and in the right, we have a document database which has a similar structure to JSON. Now for the relational database, you have to know what columns you have and so on. However, for a document database, you have data store like JSON object. You do not require to define which make it flexible.

The document type is mostly used for CMS systems, blogging platforms, real-time analytics & e-commerce applications. It should not use for complex transactions which require multiple operations or queries against varying aggregate structures.

Amazon SimpleDB, CouchDB, MongoDB, Riak, Lotus Notes, MongoDB, are popular Document originated DBMS systems.

Graph-Based

A graph type database stores entities as well the relations amongst those entities. The entity is stored as a node with the relationship as edges. An edge gives a relationship between nodes. Every node and edge has a unique identifier.


Friends
Person
LivesIn(address,...,...)
Likes (rating, review...)
Likes (rating, review...)
City
Restaurant
LocatedIn(address,...,...)

Compared to a relational database where tables are loosely connected, a Graph database is a multi-relational in nature. Traversing relationship is fast as they are already captured into the DB, and there is no need to calculate them.

Graph base database mostly used for social networks, logistics, spatial data.

Advantages of NoSQL

- Can be used as Primary or Analytic Data Source
- Big Data Capability

- No Single Point of Failure
- Easy Replication
- No Need for Separate Caching Layer
- It provides fast performance and horizontal scalability.
- Can handle structured, semi-structured, and unstructured data with equal effect
- Object-oriented programming which is easy to use and flexible
- NoSQL databases don't need a dedicated high-performance server
- Support Key Developer Languages and Platforms
- Simple to implement than using RDBMS
- It can serve as the primary data source for online applications.
- Handles big data which manages data velocity, variety, volume, and complexity
- Excels at distributed database and multi-data center operations
- Eliminates the need for a specific caching layer to store data
- Offers a flexible schema design which can easily be altered without downtime or service disruption

Disadvantages of NoSQL

- No standardization rules
- Limited query capabilities
- RDBMS databases and tools are comparatively mature
- It does not offer any traditional database capabilities, like consistency when multiple transactions are performed simultaneously.
- When the volume of data increases it is difficult to maintain unique values as keys become difficult
- Doesn't work as well with relational data
- The learning curve is stiff for new developers
- Open source options so not so popular for enterprises.

NoSQL Applications

Facebook messaging platform

Apache Cassandra was created by Facebook to power their Inbox. It did this for a number of years. Cassandra worked by doing the following:

- Cassandra indexed users' messages and the terms (words, and so on) in the messages and drove a search over all the content in those messages. The user ID was the primary key. Each term became a super column, and the message IDs were the column names.

Amazon DynamoDB

Amazon originally published the Dynamo paper, thereby launching the concept of NoSQL key-value stores. Since then, Amazon has created a separate database called **DynamoDB** as a service offered on the Amazon Web Services marketplace site.

Google Mail

Google’s Bigtable was created to provide wide-column storage for a range of Google’s applications, including Orkut, Google Earth, web indexing, Google Maps, Google Books, YouTube, blogger.com, Google Code and **Google Mail.**

LinkedIn

LinkedIn has used Hadoop to churn information about relationships overnight and to push the latest graph information to the Voldemort key-value NoSQL store for query the next day. In this way, LinkedIn maintained a rolling view of all data in the service.

BBC iPlayer online media catalog

The British Broadcasting Corporation has an online service to provide UK citizens with a free catchup service called the iPlayer for BBC television and radio shows.

The information for episodes, series, and brands is updated by a different team from that responsible for scheduling episodes for TV.

BBC Sport and Olympics platforms

In 2011, the BBC realized that its journalists were spending a lot of time deciding where to publish stories on the BBC Sport website. This cost a lot of time and money and stories weren’t consistently available to users in different areas of the sports website.

HealthCare.gov

Healthcare.gov has been called the most complex IT system implementation of all time. Building it required several systems, with the most visible one being the HealthCare.gov marketplace.

UK NHS Spine 2 Backbone

The UK National Health Service comprises hundreds of organizations, all under one national umbrella. For example, general practice surgeries and hospitals each have their own systems.

Secure information sharing

In many situations, you need to provide access to information while also maintaining its security. Here are several examples:

- A book publisher providing access to summaries so that you can verify the relevance of a book before purchase, but only view the full book after purchase

Citizen engagement

Governments use NoSQL databases to empower citizens with information about how their country is governed. A good example is Fairfax County in Virginia, which uses MarkLogic Server to provide geospatial information through an online browse and search interface to government agencies and residents.

The service covers a range of information — for example, **geographic points in the county and police-related events**.

Overview of NewSQL

While various variants of the NoSQL database continue to be used, there is another paradigm arising in parallel to NoSQL — **NewSQL**. NewSQL promises to combine benefits from RDBMS (strong consistency) with benefits from NoSQL (scalability); it mainly achieves this through new architecture patterns and efficient SQL storage engines.

Most current NewSQL databases are based on Google's Spanner database and the theories in academic papers such as Calvin: Fast Distributed Transactions for Partitioned Database Systems from Yale. Spanner is Google's scalable, multi-version, globally-distributed, and synchronously-replicated database. It was the first system to distribute data at global scale and support externally-consistent distributed transactions. The *Calvin* academic paper from Yale was on leveraging determinism to guarantee active replication and full ACID-compliance of distributed transactions without two-phase commit.

Comparing SQL, NoSQL, and NewSQL

Feature	**SQL**	**NoSQL**	**NewSQL**
Relational Property	Yes, follows relational modeling to a large extent.	No, it doesn't follow a relational model, it was designed to be entirely different from that.	Yes, since the relational model is equally essential for real-time analytics.
ACID	Yes, ACID properties are fundamental to their application	No, rather provides for CAP support	Yes, Acid properties are taken care of.

SQL	Support for SQL	No support for old SQL	Yes, proper support and even enhanced functionalities for Old SQL
OLTP	Inefficient for OLTP databases.	Supports such databases but it is not the best suited.	Fully functionally supports OLTP databases and is highly efficient
Scaling	Vertical scaling	Vertical scaling	Vertical + Horizontal scaling
Query Handling	Can handle simple queries with ease and fails when they get complex in nature	Better than SQL for processing complex queries	Highly efficient to process complex queries and smaller queries.
Distributed Databases	No	Yes	Yes

What Is MongoDB?

MongoDB is an open-source document database built on a horizontal scale-out architecture. Founded in 2007, MongoDB has a worldwide following in the developer community.

Instead of storing data in tables of rows or columns like SQL databases, each row in a MongoDB database is a document described in JSON, a formatting language.

Here’s a simple JSON document describing contact information:

```
{
 "name" :  "Carlos Smith",
 "title" : "Product Manager",
 "location" : "New York, NY",
 "twitter" : "@MongoDB",
 "facebook" : "@MongoDB"
}
```

Document databases are extremely flexible, allowing variations in the structure of documents and allowing storage of documents that are partially complete. One document can have others embedded in it.

Fields in a document play the role of columns in a SQL database, and like columns, they can be indexed to increase search performance.

Best of all for many developers, the programmer can change the structure of the database easily as needs change. Some say this turns data into code.

From its founding, MongoDB was built on a scale-out architecture, a structure that allows many small machines to work together to create systems that are fast and handle huge amounts of data.

MongoDB has always focused on providing developers an excellent user experience, which, in addition to all its other properties, has made MongoDB a favorite of developers worldwide for a huge variety of applications.

Why Use MongoDB?

MongoDB is a document database built on a scale-out architecture that has become popular with developers of all kinds who are building scalable applications using agile methodologies.

MongoDB was built for people who are building internet and business applications who need to evolve quickly and scale elegantly. If you are doing that, you should consider MongoDB.

Companies and development teams of all sizes use MongoDB because:

- The document data model is a powerful way to store and retrieve data that allows developers to move fast.
- MongoDB’s horizontal, scale-out architecture can support huge volumes of both data and traffic.
- MongoDB has a great user experience for developers who can install MongoDB and start writing code immediately.
- MongoDB can be used everywhere by anyone:
    - For free through the open source community edition
    - In the largest data centers through the enterprise edition
    - In any of the major public clouds through MongoDB Atlas
- MongoDB has developed a large and mature platform ecosystem, which means:
    - *MongoDB has a worldwide community of developers and consultants, so it is easy to get help.*

- MongoDB works on all types of computing platforms, both on-premise and in the cloud.
- MongoDB can be used from all major languages.
- MongoDB can be accessed from all major ETL and data management systems.
- MongoDB has enterprise-grade support.

Why use MongoDB? Simply to go further and faster when developing software applications that have to handle data of all sorts in a scalable way.

Thousands of companies like Bosch, Barclays, and Morgan Stanley run their businesses on MongoDB, and use it to handle their most demanding apps in areas like IoT, Gaming, Logistics, Banking, e-Commerce, and Content Management.

MongoDB is a great choice if you need to:

- Represent data with natural clusters and variability over time or in its structure
- Support rapid iterative development.
- Enable collaboration of a large number of teams
- Scale to high levels of read and write traffic.
- Scale your data repository to a massive size.
- Evolve the type of deployment as the business changes.
- Store, manage, and search data with text, geospatial, or time series dimensions.

MongoDB as a company has grown because the number of use cases with these characteristics keep growing.

Characteristics of MangoDB

High Performance

MongoDB provides high performance data persistence. In particular,

- Support for embedded data models reduces I/O activity on database system.
- Indexes support faster queries and can include keys from embedded documents and arrays.

Rich Query Language

MongoDB supports a rich query language to support read and write operations (CRUD) as well as:

- Data Aggregation
- Text Search and Geospatial Queries.

SEE ALSO

- SQL to MongoDB Mapping Chart
- SQL to Aggregation Mapping Chart

High Availability

MongoDB's replication facility, called replica set, provides:

- *automatic* failover
- data redundancy.

A replica set is a group of MongoDB servers that maintain the same data set, providing redundancy and increasing data availability.

Horizontal Scalability

MongoDB provides horizontal scalability as part of its *core* functionality:

- Sharding distributes data across a cluster of machines.

- Starting in 3.4, MongoDB supports creating zones of data based on the shard key. In a balanced cluster, MongoDB directs reads and writes covered by a zone only to those shards inside the zone. See the Zones manual page for more information.

Support for Multiple Storage Engines

MongoDB supports multiple storage engines:

- WiredTiger Storage Engine (including support for Encryption at Rest)
- In-Memory Storage Engine.

In addition, MongoDB provides pluggable storage engine API that allows third parties to develop storage engines for MongoDB.

What is Apache Cassandra?

Apache Cassandra is an open source, distributed and decentralized/distributed storage system (database), for managing very large amounts of structured data spread out across the world. It provides highly available service with no single point of failure.

Listed below are some of the notable points of Apache Cassandra −

- It is scalable, fault-tolerant, and consistent.
- It is a column-oriented database.
- Its distribution design is based on Amazon’s Dynamo and its data model on Google’s Bigtable.
- Created at Facebook, it differs sharply from relational database management systems.
- Cassandra implements a Dynamo-style replication model with no single point of failure, but adds a more powerful “column family” data model.
- Cassandra is being used by some of the biggest companies such as Facebook, Twitter, Cisco, Rackspace, ebay, Twitter, Netflix, and more.

Features of Cassandra

Cassandra has become so popular because of its outstanding technical features. Given below are some of the features of Cassandra:

- **Elastic scalability** − Cassandra is highly scalable; it allows to add more hardware to accommodate more customers and more data as per requirement.
- **Always on architecture** − Cassandra has no single point of failure and it is continuously available for business-critical applications that cannot afford a failure.
- **Fast linear-scale performance** − Cassandra is linearly scalable, i.e., it increases your throughput as you increase the number of nodes in the cluster. Therefore it maintains a quick response time.

- **Flexible data storage** − Cassandra accommodates all possible data formats including: structured, semi-structured, and unstructured. It can dynamically accommodate changes to your data structures according to your need.
- **Easy data distribution** − Cassandra provides the flexibility to distribute data where you need by replicating data across multiple data centers.
- **Transaction support** − Cassandra supports properties like Atomicity, Consistency, Isolation, and Durability (ACID).
- **Fast writes** − Cassandra was designed to run on cheap commodity hardware. It performs blazingly fast writes and can store hundreds of terabytes of data, without sacrificing the read efficiency.

History of Cassandra

- Cassandra was developed at Facebook for inbox search.
- It was open-sourced by Facebook in July 2008.
- Cassandra was accepted into Apache Incubator in March 2009.
- It was made an Apache top-level project since February 2010.

UNIT-IV

Hadoop foundation of analytics:

HADOOP

Hadoop is an Apache open source framework written in java that allows distributed processing of large datasets across clusters of computers using simple programming models. The Hadoop framework application works in an environment that provides distributed *storage* and *computation* across clusters of computers. Hadoop is designed to scale up from single server to thousands of machines, each offering local computation and storage.

Hadoop Architecture

At its core, Hadoop has two major layers namely −

- Processing/Computation layer (MapReduce), and
- Storage layer (Hadoop Distributed File System).

Hadoop | History or Evolution

Hadoop is an open source framework overseen by Apache Software Foundation which is written in Java for storing and processing of huge datasets with the cluster of commodity hardware. There are mainly two problems with the big data. First one is to store such a huge amount of data and the second one is to process that stored data. The traditional approach like RDBMS is not sufficient due to the heterogeneity of the data. So Hadoop comes as the solution to the problem of big data i.e. storing and processing the big data with some extra capabilities. There are mainly two components of Hadoop which are **Hadoop Distributed File System (HDFS)** and **Yet Another Resource Negotiator(YARN)**.

Hadoop History

Hadoop was started with **Doug Cutting and Mike Cafarella** in the year 2002 when they both started to work on Apache Nutch project. Apache Nutch project was the process of building a search engine system that can index 1 billion pages. After a lot of research on Nutch, they concluded that such a system will cost around half a million dollars in hardware, and along with a monthly running cost of $30, 000 approximately, which is very expensive. So, they realized that their project architecture will not be capable enough to the workaround with billions of pages on the web. So they were looking for a feasible solution which can reduce the implementation cost as well as the problem of storing and processing of large datasets.

In 2003, they came across a paper that described the architecture of Google's distributed file system, called **GFS (Google File System)** which was published by Google, for storing the large data sets. Now they realize that this paper can solve their problem of storing very large files which were being generated because of web crawling and indexing processes. But this paper was just the half solution to their problem.

In 2004, Google published one more paper on the technique **MapReduce**, which was the solution of processing those large datasets. Now this paper was another half solution for Doug Cutting and Mike

Cafarella for their Nutch project. These both techniques (GFS & MapReduce) were just on white paper at Google. Google didn't implement these two techniques. Doug Cutting knew from his work on Apache Lucene (It is a free and open-source information retrieval software library, originally written in Java by Doug Cutting in 1999) that open-source is a great way to spread the technology to more people. So, together with Mike Cafarella, he started implementing Google's techniques (GFS & MapReduce) as open-source in the Apache Nutch project.

In 2005, Cutting found that Nutch is limited to only 20-to-40 node clusters. He soon realized two problems:
(a) Nutch wouldn't achieve its potential until it ran reliably on the larger clusters
(b) And that was looking impossible with just two people (Doug Cutting & Mike Cafarella). The engineering task in Nutch project was much bigger than he realized. So he started to find a job with a company who is interested in investing in their efforts. And he found Yahoo!.Yahoo had a large team of engineers that was eager to work on this there project.

So **in 2006,** Doug Cutting joined Yahoo along with Nutch project. He wanted to provide the world with an open-source, reliable, scalable computing framework, with the help of Yahoo. So at Yahoo first, he separates the distributed computing parts from Nutch and **formed a new project Hadoop (He gave name Hadoop it was the name of a yellow toy elephant which was owned by the Doug Cutting's son. and it was easy to pronounce and was the unique word.**) Now he wanted to make Hadoop in such a way that it can work well on thousands of nodes. So with GFS and MapReduce, he started to work on Hadoop.

In 2007, Yahoo successfully tested Hadoop on a 1000 node cluster and start using it.

In January of 2008, Yahoo released Hadoop as an open source project to ASF(Apache Software Foundation). And in **July of 2008, Apache Software Foundation successfully tested a 4000 node cluster with Hadoop.**

In 2009, Hadoop was successfully tested to sort a PB (PetaByte) of data in less than 17 hours for handling billions of searches and indexing millions of web pages. And **Doug Cutting left the Yahoo and joined Cloudera to fulfill the challenge of spreading Hadoop to other industries.**

In **December of 2011,** Apache Software Foundation released **Apache Hadoop version 1.0.**

And later in Aug 2013, **Version 2.0.6 was available**.

And currently, we have **Apache Hadoop version 3.0** which released in **December 2017**.

Advantages of Hadoop:

1. Scalable

Hadoop is a highly scalable storage platform, because it can stores 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, Hadoop enables businesses to run applications on thousands of nodes involving many thousands of terabytes of data.

2. 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. While this approach may have worked in the short term, this meant that when business priorities changed, the complete raw data set was not available, as it was too expensive to store.

3. 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. Hadoop can be used for a wide variety of purposes, such as log processing, recommendation systems, data warehousing, market campaign analysis and fraud detection.

4. 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.

5. 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.

Disadvantages of Hadoop:

As the backbone of so many implementations, Hadoop is almost synomous with big data.

1. Security Concerns

Just managing a complex applications such as Hadoop can be challenging. A simple example can be seen in the Hadoop security model, which is disabled by default due to sheer complexity. If whoever managing the platform lacks of know how to enable it, your data could be at huge risk. Hadoop is also missing encryption at the storage and network levels, which is a major selling point for government agencies and others that prefer to keep their data under wraps.

2. Vulnerable By Nature

Speaking of security, the very makeup of Hadoop makes running it a risky proposition. The framework is written almost entirely in Java, one of the most widely used yet controversial programming languages in existence. Java has been heavily exploited by cybercriminals and as a result, implicated in numerous security breaches.

3. Not Fit for Small Data

While big data is not exclusively made for big businesses, not all big data platforms are suited for small data needs. Unfortunately, Hadoop happens to be one of them. Due to its high capacity design, the Hadoop Distributed File System, lacks the ability to efficiently support the random reading of small files. As a result, it is not recommended for organizations with small quantities of data.

4. Potential Stability Issues

Like all open source software, Hadoop has had its fair share of stability issues. To avoid these issues, organizations are strongly recommended to make sure they are running the latest stable version, or run it under a third-party vendor equipped to handle such problems.

5. General Limitations

The article introduces Apache Flume, MillWheel, and Google's own Cloud Dataflow as possible solutions. What each of these platforms have in common is the ability to improve the efficiency and reliability of data collection, aggregation, and integration. The main point the article stresses is that companies could be missing out on big benefits by using Hadoop alone.

Hadoop Versions:

Hadoop is a Software which on an open-source framework storing data using a distributed network rather than a centralized one thereby processing the data in a parallel transition. This enables Hadoop to act as one of the most reliable batch processing engine and layered storage and resource management system.
As the data beings stored and processed increases in its complexity so do Hadoop where the developers bring out various versions to address the issues (bug fixes) and simplify the complex data processes. The updates are automatically implemented as Hadoop development follows the trunk (base code) – branch (fix)model. Hadoop has two versions: a) Hadoop 1.x (Version 1) and b) Hadoop 2 (Version 2)
Below are the two Hadoop Versions:

- Hadoop 1.x (Version 1)
- Hadoop 2 (Version 2)

1. Hadoop 1.x
Below are the Components of Hadoop 1.x

1. The Hadoop Common Module is a jar file which acts as the base API on top of which all the other components work.

2. Version one being the first one to come in existence is rock solid and has got no new updates

3. It has a limitation on the scaling nodes with just a maximum of 4000 nodes for each cluster

4. The functionality is limited utilizing the slot concept, i.e., the slots are capable of running a map task or a reduce task.

5. The next component if the Hadoop Distributed File System commonly known as HDFS, which plays the role of a distributed storage system that is designed to cater to large data, with a block size of 64 MegaBytes (64MB) for supporting the architecture. It is further divided into two components:

- Name Node which is used to store metadata about the Data node, placed with the Master Node. They contain details like the details about the slave note, indexing and their respective locations along with timestamps for timelining.
- Data Nodes used for storage of data related to the applications in use placed in the Slave Nodes.

6. Hadoop 1 uses Map Reduce (MR) data processing model It is not capable of supporting other non-MR tools.

MR has two components:

- Job Tracker is used to assigning or reassigning task-related (in case scenario fails or shutdown) to MapReduce to an application called task tracker is located in the node clusters. It additionally maintains a log about the status of the task tracker.
- The Task Tracker is responsible for executing the functions which have been allocated by the job tracker and sensor cross the status report of those task to the job tracker.

7. The network of the cluster is formed by organizing the master node and slave nodes. Which of this cluster is further divided into tracks which contain a set of commodity computers or nodes.

8. Whenever a large storage operation for big data set is is received by the Hadoop system, the data is divided into decipherable and organized blocks that are distributed into different nodes.

2. Hadoop Version 2

Version 2 for Hadoop was released to provide improvements over the lags which the users faced with version 1. Let's throw some light over the improvements that the new version provides:

- HDFS Federation which has improved to provide for horizontal scalability for the name node. Moreover, the namenode was available for a single point of failure only, it is available on varied points. This is going to the Hadoop stat has been increased to include the stacks such as Hive, Pig, which make this tap well equipped enabling me to handle failures pertaining to NameNode.
- YARN stands for Yey Another Resource Network has been improved with the new ability to process data in the larger term that is petabyte and terabyte to make it available for the HDFS while using the applications which are not MapReduce based. These include applications like MPI and GIRAPH.
- **Version – 2.7.x Released on 31st May 2018:** The update focused to provide for two major functionalities that are providing for your application and providing for a global resource manager, thereby improving its overall utility and versatility, increasing scalability up to 10000 nodes for each cluster.
- **Version 2.8.x – Released in September 2018:** The updated provided improvements include the capacity scheduler which is designed to provide multi-tenancy support for processing data over Hadoop and it has been made to be accessible for window uses so that there is an increase in the rate of adoption for the software across the industry for dealing with problems related to big data.

Version 3

Below is the latest running Hadoop Updated Version

Version 3.1.x – released on 21 October 2019: This update enables Hadoop to be utilized as a platform to serve a big chunk of Data Analytics Functions and utilities to be performed over event processing alongside using real-time operations give a better result.

- It has now improved feature work on the container concept which enables had to perform generic which were earlier not possible with version 1.

- The latest **version 3.2.1 released on 22nd September 2019** addresses issues of non-functionality (in terms of support) of data nodes for multi-Tenancy, limitation to you only MapReduce processing and the biggest problem than needed for an alternate data storage which is needed for the real-time processing and graphical analysis.
- The ever-increasing Avalanche of data and Big Data Analytics pertaining to just business standing at an estimated 169 billion dollars (USD), the predicted growth to 274 billion dollars by 2022, the market seems to be growing ecstatically.
- This all the more calls for a system that is integrable in its functioning for the abandoned Utah which is growing day by day. Hadoop app great to store, process and access the great solution which works to store process and access this heterogeneous set of data which can be unstructured/ structure in an organized manner.
- With the feature of constant updates which act as tools to rectify the bugs that developers say while using Hadoop, and the improved versions increase the scope of application and improve the dimension and flexibility of using Hadoop, increases the chances of it is the next biggest to for all functions related to big data processing and Analytics.

Hadoop Ecosystem

Overview: Apache Hadoop is an open source framework intended to make interaction with **big data** easier, However, for those who are not acquainted with this technology, one question arises that what is big data ? Big data is a term given to the data sets which can't be processed in an efficient manner with the help of traditional methodology such as RDBMS. Hadoop has made its place in the industries and companies that need to work on large data sets which are sensitive and needs efficient handling. Hadoop is a framework that enables processing of large data sets which reside in the form of clusters. Being a framework, Hadoop is made up of several modules that are supported by a large ecosystem of technologies.

Introduction: *Hadoop Ecosystem* is a platform or a suite which provides various services to solve the big data problems. It includes Apache projects and various commercial tools and solutions. There are *four major elements of Hadoop* i.e. **HDFS, MapReduce, YARN, and Hadoop Common**. Most of the tools or solutions are used to supplement or support these major elements. All these tools work collectively to provide services such as absorption, analysis, storage and maintenance of data etc.

Following are the components that collectively form a Hadoop ecosystem:

- **HDFS:** Hadoop Distributed File System
- **YARN:** Yet Another Resource Negotiator
- **MapReduce:** Programming based Data Processing
- **Spark:** In-Memory data processing
- **PIG, HIVE:** Query based processing of data services
- **HBase:** NoSQL Database
- **Mahout, Spark MLLib:** Machine Learning algorithm libraries
- **Solar, Lucene:** Searching and Indexing
- **Zookeeper:** Managing cluster
- **Oozie:** Job Scheduling

Note: Apart from the above-mentioned components, there are many other components too that are part of the Hadoop ecosystem.

All these toolkits or components revolve around one term i.e. *Data*. That's the beauty of Hadoop that it revolves around data and hence making its synthesis easier.

HDFS:

- HDFS is the primary or major component of Hadoop ecosystem and is responsible for storing large data sets of structured or unstructured data across various nodes and thereby maintaining the metadata in the form of log files.
- HDFS consists of two core components i.e.
  1. Name node
  2. Data Node
- Name Node is the prime node which contains metadata (data about data) requiring comparatively fewer resources than the data nodes that stores the actual data. These data nodes are commodity hardware in the distributed environment. Undoubtedly, making Hadoop cost effective.

- HDFS maintains all the coordination between the clusters and hardware, thus working at the heart of the system.

YARN:

- Yet Another Resource Negotiator, as the name implies, YARN is the one who helps to manage the resources across the clusters. In short, it performs scheduling and resource allocation for the Hadoop System.
- Consists of three major components i.e.
    1. Resource Manager
    2. Nodes Manager
    3. Application Manager
- Resource manager has the privilege of allocating resources for the applications in a system whereas Node managers work on the allocation of resources such as CPU, memory, bandwidth per machine and later on acknowledges the resource manager. Application manager works as an interface between the resource manager and node manager and performs negotiations as per the requirement of the two.

MapReduce:

- By making the use of distributed and parallel algorithms, MapReduce makes it possible to carry over the processing's logic and helps to write applications which transform big data sets into a manageable one.
- MapReduce makes the use of two functions i.e. Map() and Reduce() whose task is:
    1. *Map()* performs sorting and filtering of data and thereby organizing them in the form of group. Map generates a key-value pair based result which is later on processed by the Reduce() method.
    2. *Reduce()*, as the name suggests does the summarization by aggregating the mapped data. In simple, Reduce() takes the output generated by Map() as input and combines those tuples into smaller set of tuples.

PIG:

- Pig was basically developed by Yahoo which works on a pig Latin language, which is Query based language similar to SQL.
- It is a platform for structuring the data flow, processing and analyzing huge data sets.
- Pig does the work of executing commands and in the background, all the activities of MapReduce are taken care of. After the processing, pig stores the result in HDFS.
- Pig Latin language is specially designed for this framework which runs on Pig Runtime. Just the way Java runs on the JVM.

- Pig helps to achieve ease of programming and optimization and hence is a major segment of the Hadoop Ecosystem.

HIVE:

- With the help of SQL methodology and interface, HIVE performs reading and writing of large data sets. However, its query language is called as HQL (Hive Query Language).
- It is highly scalable as it allows real-time processing and batch processing both. Also, all the SQL datatypes are supported by Hive thus, making the query processing easier.
- Similar to the Query Processing frameworks, HIVE too comes with two components: *JDBC Drivers* and *HIVE Command Line*.
- JDBC, along with ODBC drivers work on establishing the data storage permissions and connection whereas HIVE Command line helps in the processing of queries.

Mahout:

- Mahout, allows Machine Learnability to a system or application. Machine Learning, as the name suggests helps the system to develop itself based on some patterns, user/environmental interaction or om the basis of algorithms.
- It provides various libraries or functionalities such as collaborative filtering, clustering, and classification which are nothing but concepts of Machine learning. It allows invoking algorithms as per our need with the help of its own libraries.

Apache Spark:

- It's a platform that handles all the process consumptive tasks like batch processing, interactive or iterative real-time processing, graph conversions, and visualization, etc.
- It consumes in memory resources hence, thus being faster than the prior in terms of optimization.
- Spark is best suited for real-time data whereas Hadoop is best suited for structured data or batch processing, hence both are used in most of the companies interchangeably.

Apache HBase:

- It's a NoSQL database which supports all kinds of data and thus capable of handling anything of Hadoop Database. It provides capabilities of Google's BigTable, thus able to work on Big Data sets effectively.
- At times where we need to search or retrieve the occurrences of something small in a huge database, the request must be processed within a short quick span of time. At such times, HBase comes handy as it gives us a tolerant way of storing limited data.

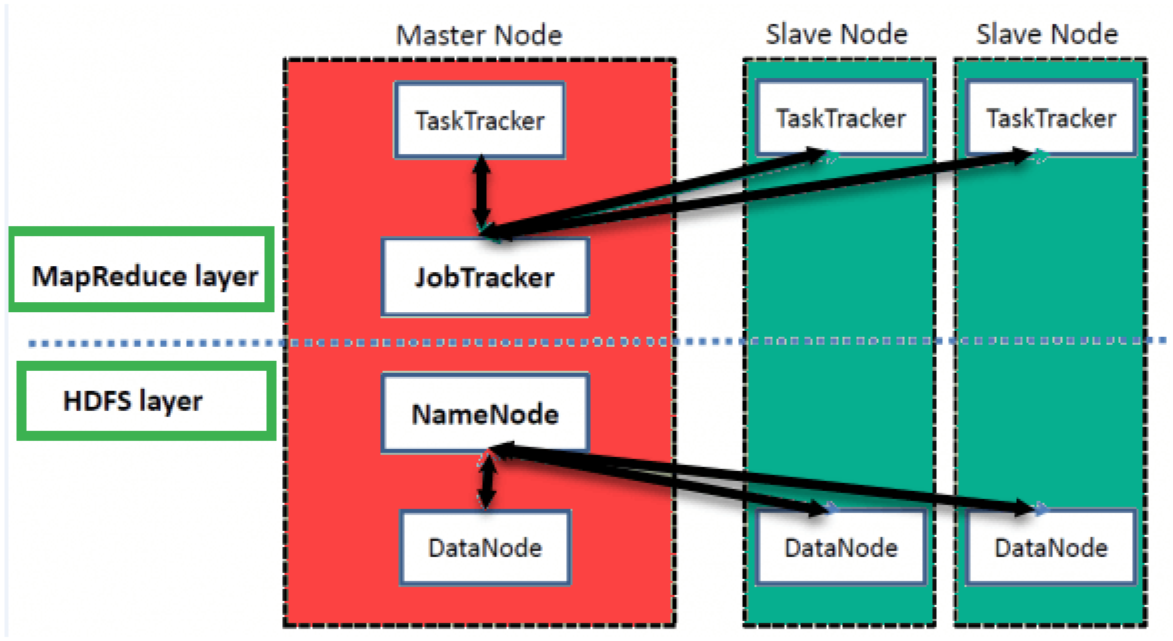
Other Components: Apart from all of these, there are some other components too that carry out a huge task in order to make Hadoop capable of processing large datasets. They are as follows:

- **Solr, Lucene:** These are the two services that perform the task of searching and indexing with the help of some java libraries, especially Lucene is based on Java which allows spell check mechanism, as well. However, Lucene is driven by Solr.
- **Zookeeper:** There was a huge issue of management of coordination and synchronization among the resources or the components of Hadoop which resulted in inconsistency, often. Zookeeper overcame all the problems by performing synchronization, inter-component based communication, grouping, and maintenance.
- **Oozie:** Oozie simply performs the task of a scheduler, thus scheduling jobs and binding them together as a single unit. There is two kinds of jobs .i.e Oozie workflow and Oozie coordinator jobs. Oozie workflow is the jobs that need to be executed in a sequentially ordered manner whereas Oozie Coordinator jobs are those that are triggered when some data or external stimulus is given to it.

Hadoop vs RDBMS

Hadoop software framework work is very well structured semi-structured and unstructured data. This also supports a variety of data formats in real-time such as XML, JSON, and text-based flat file formats. RDBMS works efficiently when there is an entity-relationship flow that is defined perfectly and therefore, the database schema or structure can grow and unmanaged otherwise. i.e., An RDBMS works well with structured data. Hadoop will be a good choice in environments when there are needs for big data processing on which the data being processed does not have dependable relationships.

Hadoop Architecture



High Level Hadoop Architecture

Hadoop has a Master-Slave Architecture for data storage and distributed data processing using MapReduce and HDFS methods.

NameNode:

NameNode represented every files and directory which is used in the namespace

DataNode:

DataNode helps you to manage the state of an HDFS node and allows you to interacts with the blocks

MasterNode:

The master node allows you to conduct parallel processing of data using Hadoop MapReduce.

Slave node:

The slave nodes are the additional machines in the Hadoop cluster which allows you to store data to conduct complex calculations. Moreover, all the slave node comes with Task Tracker and a DataNode. This allows you to synchronize the processes with the NameNode and Job Tracker respectively.

In Hadoop, master or slave system can be set up in the cloud or on-premise

Features Of 'Hadoop'

• Suitable for Big Data Analysis

As Big Data tends to be distributed and unstructured in nature, HADOOP clusters are best suited for analysis of Big Data. Since it is processing logic (not the actual data) that flows to the computing nodes, less network bandwidth is consumed. This concept is called as **data locality concept** which helps increase the efficiency of Hadoop based applications.

• Scalability

HADOOP clusters can easily be scaled to any extent by adding additional cluster nodes and thus allows for the growth of Big Data. Also, scaling does not require modifications to application logic.

• Fault Tolerance

HADOOP ecosystem has a provision to replicate the input data on to other cluster nodes. That way, in the event of a cluster node failure, data processing can still proceed by using data stored on another cluster node.

UNIT –V

HadoopMapReduce and YARN framework:

Introduction to MapReduce:

What is MapReduce?

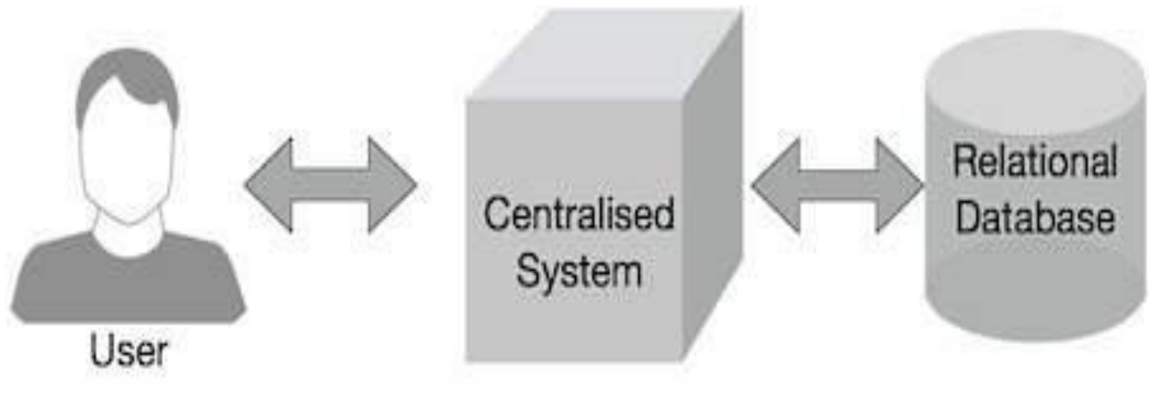
MapReduce is a programming model for writing applications that can process Big Data in parallel on multiple nodes. MapReduce provides analytical capabilities for analyzing huge volumes of complex data.

What is Big Data?

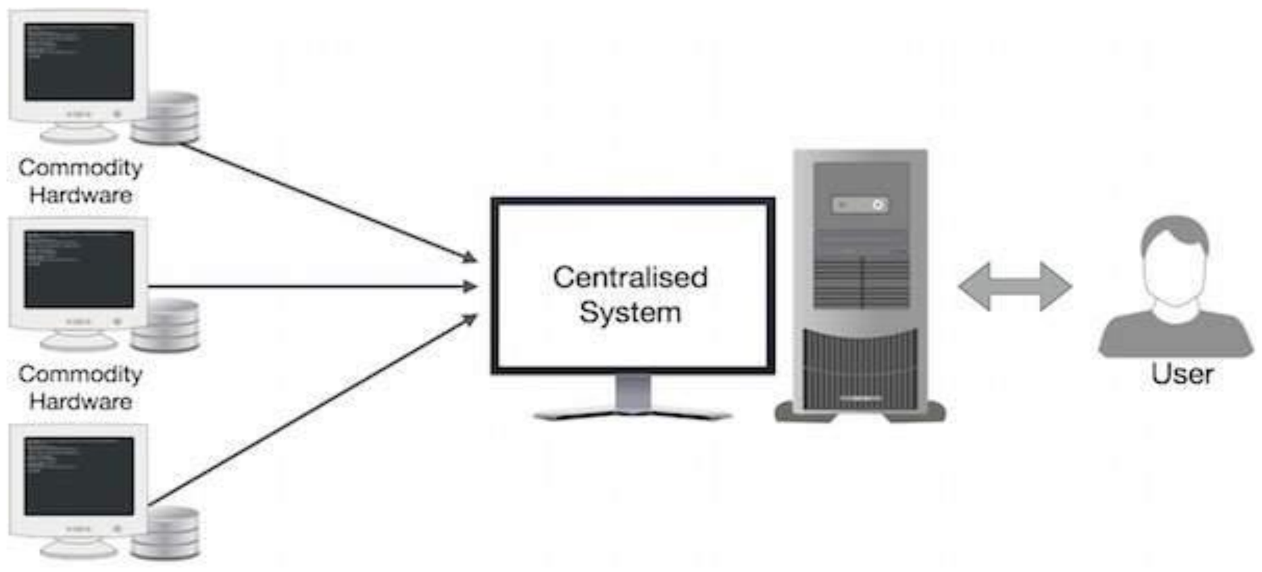
Big Data is a collection of large datasets that cannot be processed using traditional computing techniques. For example, the volume of data Facebook or Youtube need require it to collect and manage on a daily basis, can fall under the category of Big Data. However, Big Data is not only about scale and volume, it also involves one or more of the following aspects − Velocity, Variety, Volume, and Complexity.

Why MapReduce?

Traditional Enterprise Systems normally have a centralized server to store and process data. The following illustration depicts a schematic view of a traditional enterprise system. Traditional model is certainly not suitable to process huge volumes of scalable data and cannot be accommodated by standard database servers. Moreover, the centralized system creates too much of a bottleneck while processing multiple files simultaneously.



Google solved this bottleneck issue using an algorithm called MapReduce. MapReduce divides a task into small parts and assigns them to many computers. Later, the results are collected at one place and integrated to form the result dataset.



An Introduction to YARN

Get a top-down introduction to YARN. YARN allows integration of frameworks, such as Spark and HAMA, into Hadoop to expand the popular Data tool beyond MapReduce.

YARN/Hadoop 2.x has a completely different architecture with compared to Hadoop 1.x.

In Hadoop 1.x JobTracker serves two major functions:

1. Resource management
2. Job scheduling/Job monitoring

Recall that in Hadoop 1.x, there was a single JobTracker per Hadoop cluster serving these functions in which scaling can overwhelm the JobTracker. Also, having a single JobTracker makes it a single point of failure; if the JobTracker goes down, the entire cluster goes down with all the current jobs.

YARN tries to separate above mentioned functionalities into two daemons:

1. Global Resource Manager
2. Per-application Application Master

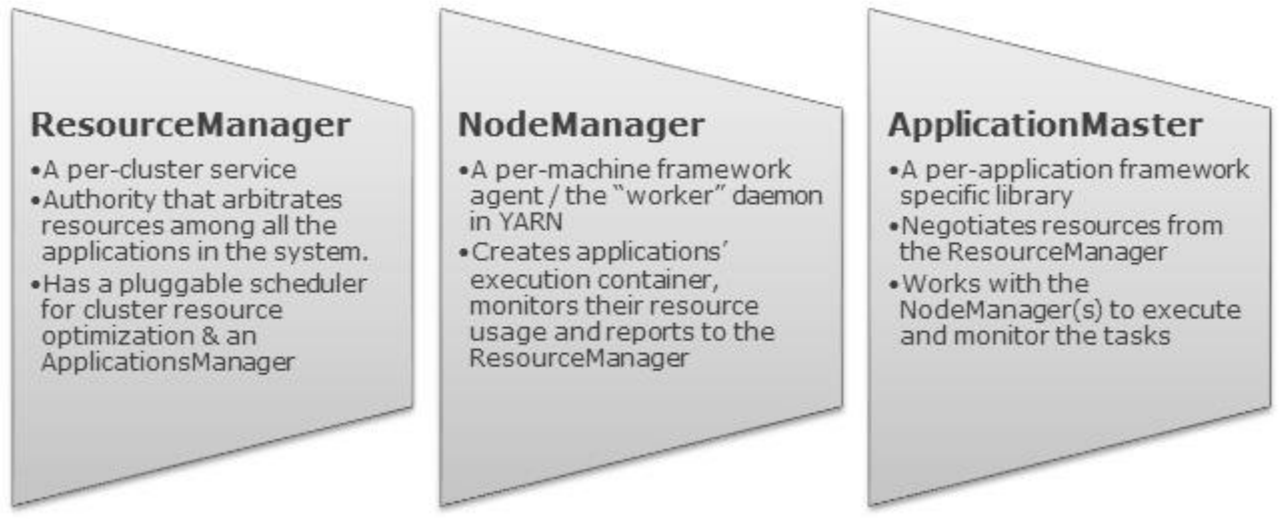
Before YARN, Hadoop was designed to support MapReduce jobs only. As time went by, people encountered Big Data computation problems that cannot be addressed by MapReduce and thus came up with different frameworks which work on top of HDFS to address their problems. Some of these were:

- Apache Spark
- Apache HAMA
- Apache Giraph.

YARN provides a way for these new frameworks to be integrated into Hadoop framework, sharing the same underlying HDFS. YARN enables Hadoop to handle jobs beyond MapReduce.

YARN components

YARN divides the responsibilities of JobTracker into separate components, each having a specified task to perform. In Hadoop-1, the JobTracker takes care of resource management, job scheduling, and job monitoring. YARN divides these responsibilities of JobTracker into ResourceManager and ApplicationMaster. Instead of TaskTracker, it uses NodeManager as the worker daemon for execution of map-reduce tasks. The ResourceManager and the NodeManager form the computation framework for YARN, and ApplicationMaster is an application-specific framework for application management.


ResourceManager
•A per-cluster service
•Authority that arbitrates resources among all the applications in the system.
•Has a pluggable scheduler for cluster resource optimization & an ApplicationsManager
NodeManager
•A per-machine framework agent / the "worker" daemon in YARN
•Creates applications' execution container, monitors their resource usage and reports to the ResourceManager
ApplicationMaster
•A per-application framework specific library
•Negotiates resources from the ResourceManager
•Works with the NodeManager(s) to execute and monitor the tasks

ResourceManager

A ResourceManager is a per cluster service that manages the scheduling of compute resources to applications. It optimizes cluster utilization in terms of memory, CPU cores, fairness, and SLAs. To allow different policy constraints, it has algorithms in terms of pluggable schedulers such as capacity and fair that allows resource allocation in a particular way.

ResourceManager has two main components:

- **Scheduler**: This is a pure pluggable component that is only responsible for allocating resources to applications submitted to the cluster, applying constraint of capacities and queues. Scheduler does not provide any guarantee for job completion or monitoring, it only allocates the cluster resources governed by the nature of job and resource requirement.
- **ApplicationsManager** (**AsM**): This is a service used to manage application masters across the cluster that is responsible for accepting the application submission, providing the resources for application master to start, monitoring the application progress, and restart, in case of application failure.

NodeManager

The NodeManager is a per node worker service that is responsible for the execution of containers based on the node capacity. Node capacity is calculated based on the installed memory and the number of CPU cores. The NodeManager service sends a heartbeat signal to the ResourceManager to update its health status. The NodeManager service is similar to the TaskTracker service in MapReduce v1. NodeManager also sends the status to ResourceManager, which could be the status of the node on which it is running or the status of tasks executing on it.

ApplicationMaster

An ApplicationMaster is a per application framework-specific library that manages each instance of an application that runs within YARN. YARN treats ApplicationMaster as a third-party library responsible for negotiating the resources from the ResourceManager scheduler and works with NodeManager to execute the tasks. The ResourceManager allocates containers to the ApplicationMaster and these containers are then used to run the application-specific processes. ApplicationMaster also tracks the status of the application and monitors the progress of the containers. When the execution of a container gets complete, the ApplicationMaster unregisters the containers with the ResourceManager and unregisters itself after the execution of the application is complete.

Container

A container is a logical bundle of resources in terms of memory, CPU, disk, and so on that is bound to a particular node. In the first version of YARN, a container is equivalent to a block of memory. The ResourceManager scheduler service dynamically allocates resources as containers. A container grants rights to an ApplicationMaster to use a specific amount of resources of a specific host. An ApplicationMaster is considered as the first container of an application and it manages the execution of the application logic on allocated containers.

Advantages of YARN

The architecture of YARN ensures that the Hadoop cluster can be enhanced in the following ways:

- **Multi-tenancy**

YARN lets you access various proprietary and open-source engines for deploying Hadoop as a standard for real-time, interactive, and batch processing tasks that are able to access the same dataset and parse it.

- **Cluster Utilization**

YARN lets you use the Hadoop cluster in a dynamic way, rather than in a static manner by which MapReduce applications were using it, and this is a better and optimized way of utilizing the cluster.

- **Scalability**

YARN gives the power of scalability to the Hadoop cluster. YARN ResourceManager (RM) service is the central controlling authority for resource management and it makes allocation decisions.

- **Compatibility**

YARN tool is highly compatible with the existing Hadoop MapReduce applications, and thus those projects that are working with MapReduce in Hadoop 1.0 can easily move on to Hadoop 2.0 with YARN without any difficulty, ensuring complete compatibility.

MapReduce Applications

Here are a few examples of big data problems that can be solved with the MapReduce framework:

1. Given a repository of text files, find the frequency of each word. This is called the **WordCount** problem.
2. Given a repository of text files, find the number of words of each word length.
3. Given two matrices in sparse matrix format, compute their product.
4. Factor a matrix given in sparse matrix format.
5. Given a symmetric graph whose nodes represent people and edges represent friendship, compile a list of common friends.
6. Given a symmetric graph whose nodes represent people and edges represent friendship, compute the average number of friends by age.
7. Given a repository of weather records, find the annual global minima and maxima by year.
8. Sort a large list. Note that in most implementations of the MapReduce framework, this problem is trivial, because the framework automatically sorts the output from the map() function.
9. Reverse a graph.
10. Find a **minimal spanning tree** (**MST**) of a...

Data serialization

Data serialization is the process of converting data objects present in complex data structures into a byte stream for storage, transfer and distribution purposes on physical devices

Text-based Data Serialization formats and their key features?

Without being exhaustive, here are some common ones:

- **XML (Extensible Markup Language)** - Nested textual format. Human-readable and editable. Schema based validation. Used in metadata applications, web services data transfer, web publishing.

- **CSV (Comma-Separated Values)** - Table structure with delimiters. Human-readable textual data. Opens as spreadsheet or plaintext. Used as plaintext Database.
- **JSON (JavaScript Object Notation)** - Short syntax textual format with limited data types. Human-readable. Derived from JavaScript data formats. No need of a separate parser (like XML) since they map to JavaScript objects. Can be fetched with an XMLHttpRequest call. No direct support for DATE data type. All data is dynamically processed. Popular format for web API parameter passing. Mobile apps use this extensively for user interaction and database services.
- **YAML (YAML Ain't Markup Language)** - Lightweight text format. Human-readable. Supports comments and thus easily editable. Superset of JSON. Supports complex data types. Maps easily to native data structures. Used in configuration settings, document headers, Apps with need for MySQL style self-references in relational data.

Binary Data Serialization formats and their key features?

Without being exhaustive, here are some common ones:

- **BSON (Binary JSON)** - Created and internally used by MongoDB. Binary format, not human-readable. Deals with attribute-value pairs like JSON. Includes datetime, bytearray and other data types not present in JSON. Used in web apps with rich media data types such as live video. Primary use is storage, not network communication.
- **MessagePack** - Designed for data to be transparently converted from/to JSON. Compressed binary format, not human-readable. Supports static typing. Supports RPC. Better JSON compatibility than BSON. Primary use is network communication, not storage. Used in apps with distributed file systems.
- **protobuf (Protocol Buffers)** - Created by Google. Binary message format that allows programmers to specify a schema for the data. Also includes a set of rules and tools to define and exchange these messages. Transparent data compression. Used in multi-platform applications due to easy interoperability between languages. Universal RPC framework. Used in performance-critical distributed applications.

What are Data Serialization Storage format?

Storage formats are a way to define how information is stored in the file. Most of the time, this information can be assumed from the extension of the dat`a. Both structured and unstructured data can be stored on HADOOP enabled systems. Common Hdfs file formats are –

- Plain text storage
- Sequence files
- RC files
- AVRO
- Parquet

Why Storage Formats?

- File format must be handy to serve complex data structures
- HDFS enabled applications to take time to find relevant data in a particular location and write back data to another location.
- Dataset is large
- Having schemas
- Having storage constraints

Why choose different File Formats?

Proper selection of file format leads to –

- Faster read time
- Faster write time
- Splittable files (for partial data read)
- Dchema evolution support (modifying dataset fields)
- Advance compression support
- Snappy compression leads to high speed and reasonable compression/decompression.
- File formats help to manage Diverse data.

Guide to Data Serialization in Hadoop

- Data serialization is a process to format structured data in such a way that it can be reconverted back to the original form.
- Serialization is done to translate data structures into a stream of data. This stream of data can be transmitted over the network or stored in DB regardless of the system architecture.
- Isn’t storing information in binary form or stream of bytes is a right approach.
- Serialization does the same but isn’t dependent on architecture.

Consider CSV files contains a comma (,) in between data, so while Deserialization wrong outputs may occur. Now, if metadata is stored in XML form, which is a self architected form of data storage, data can be easily deserialized in the future.

Why Data Serialization for Storage Formats?

- To process records faster (Time-bound).
- When Proper format of data need to be maintained and to be transmitted over data without schema support on another end.
- Now when in future, data without structure or format needs to be processed, complex Errors may occur.
- Serialization offers data validation over transmission.

Areas of Serialization for Storage Formats

To maintain the proper format of a data serialization system must have the following four properties –

- **Compact –** helps in the best use of network bandwidth
- **Fast –** reduces the performance overhead
- **Extensible –** can match new requirements
- **Inter-operable –** not language-specific

Serialization in Hadoop has two areas –

Inter process communication

When a client calls a function or subroutine from one pc to the pc in-network or server, that Procedure of calling is known as a remote procedure call.

Persistent storage

It is better than java 's inbuilt serialization as java serialization isn' t compact Serialization and Deserialization of data helps in maintaining and managing corporate decisions for effective use of resources and data available in Data warehouse or any other database -writable – language specific to java

****