

ANNAMACHARYA **INSTITUTE OF TECHNOLOGY AND SCIENCES** **(AUTONOMOUS)**

Approved by AICTE, New Delhi & Permanent Affiliation to JNTUA, Anantapur.

Three B. Tech Programmes (CSE , ECE & CE) are accredited by NBA, New Delhi, Accredited by NAAC with 'A' Grade , Bangalore.

A-grade awarded by AP Knowledge Mission. Recognized under sections 2(f) & 12(B) of UGC Act 1956.

Venkatapuram Village, Renigunta Mandal, Tirupati, Andhra Pradesh-517520.

Department of Artificial Intelligence



Academic Year 2023-24 **III. B.Tech I Semester** **BIG DATA TECHNOLOGIES** **(20APC3019)**

Prepared By
Mr. E D PAVANKUMAR M. Tech (Ph.D.)
Assistant Professor
Department of CSE, AITS

BIG DATA TECHNOLOGIES

UNIT-I

Introduction Big Data :

'Big Data' is also a **data** but with a **huge size**.

'Big Data' is a term used to describe collection of data that is huge in size and yet growing exponentially with time.

Data which are very large in size is called Big Data.

Normally we work on data of size MB(Word Doc, Excel) or maximum GB(Movies, Codes) but data in Pet bytes i.e. 10^{15} byte size is called Big Data.

It is stated that almost 90% of today's data has been generated in the past 3 years.

In short, such a data is so large and complex that none of the traditional data management tools are able to store it or process it efficiently.

Examples Of 'Big Data'

Following are some the examples of 'Big Data'-

The **New York Stock Exchange** generates about *one terabyte* of new trade data per day.

Social Media Impact

Statistic shows that *500+terabytes* of new data gets ingested into the databases of social media site **Facebook**, Google, LinkedIn, every day. This data is mainly generated in terms of photo and video uploads, message exchanges, putting comments etc.

Single **Jet engine** can generate *10+terabytes* of data in *30 minutes* of a flight time. With

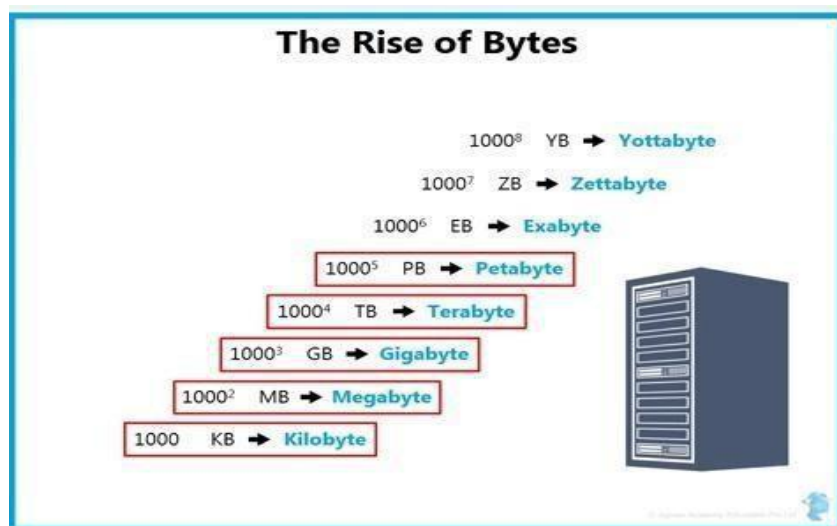
BIG DATA TECHNOLOGIES

many thousand flights per day, generation of data reaches up to many *Petabytes*.

E-commerce site: Sites like Amazon, Flipkart, Alibaba generates huge amount of logs from which users buying trends can be traced.

Weather Station: All the weather station and satellite gives very huge data which are stored and manipulated to forecast weather.

Telecom company: Telecom giants like Airtel, Vodafone study the user trends and accordingly publish their plans and for this they store the data of its million users.



What is Big Data?

Definition

Big data is a buzzword, or catch-phrase, used to describe a massive volume of both structured and unstructured data that is so large that it's difficult to process using traditional database and software techniques.

- Massive Volume
- Unstructured data
- Not possible to process using traditional techniques

Categories Of 'Big Data'

BIG DATA TECHNOLOGIES

Big data' could be found in three forms:

1. **Structured**
2. **Unstructured**
3. **Semi-**

structuredStructured

Any data that can be stored, accessed and processed in the form of fixed format is termed as a 'structured' data.

Do you know? 10^{21} bytes equals to 1 zettabyte or one billion terabytes forms a zettabyte.

Data stored in a relational database management system is one example of a 'structured' data.

Examples Of Structured Data

An 'Employee' table in a database is an example of Structured Data

Employee_ID	Employee_Name	Gender	Department	Salary_In_lacs
2365	Rajesh Kulkarni	Male	Finance	650000
3398	Pratibha Joshi	Female	Admin	650000
7465	Shushil Roy	Male	Admin	500000
7500	Shubhojit Das	Male	Finance	500000
7699	Priya Sane	Female	Finance	550000

Unstructured

Any data with unknown form or the structure is classified as unstructured data.

BIG DATA TECHNOLOGIES

Typical example of unstructured data is, a heterogeneous data source containing a combination of simple text files, images, videos etc.

Examples Of Un-structured Data

Output returned by 'Google Search' .

Semi-structured

Semi-structured data can contain both the forms of data.

Example of semi-structured data is a data represented in XML file.

Examples Of Semi-structured Data

Personal data stored in a XML file-

```
<rec><name>Prashant Rao</name><sex>Male</sex><age>35</age></rec>
```

```
<rec><name>Seema R.</name><sex>Female</sex><age>41</age></rec>
```

```
<rec><name>Satish Mane</name><sex>Male</sex><age>29</age></rec>
```

```
<rec><name>Subrato Roy</name><sex>Male</sex><age>26</age></rec>
```

```
<rec><name>Jeremiah J.</name><sex>Male</sex><age>35</age></rec>
```

BIG DATA TECHNOLOGIES

Unstructured Data

- No definite structure can be assigned to this data
- Cannot tabulate the data
- Cannot put it in rows and columns
- Cannot fit into any fixed schema



Example:

- Text files
- PDF document
- Web server logs
- Your WhatsApp messages:
 - ✓ Text
 - ✓ Photos
 - ✓ Voice



© Open University 2014. All rights reserved.

Semi-structured Data

- Data which is in between Structured and Unstructured
- Unstructured data embedded within some structures or tags or schema



Example:

XML file where unstructured data or text are embedded within tags to enforce some structure

© Open University 2014. All rights reserved.

BIG DATA TECHNOLOGIES

Characteristics of big data:

Volume

Variety

Velocity

y

Volume' is one characteristic which needs to be considered while dealing with 'Big Data'.

Size of data plays very crucial role in determining value out of data.

Variety

Variety refers to heterogeneous sources and the nature of data, both structured and unstructured.

During earlier days, spreadsheets and databases were the only sources of data considered by most of the applications.

Now days, data in the form of emails, photos, videos, monitoring devices, PDFs, audio, etc. is also being considered in the analysis applications.

Velocity

The term '**velocity**' refers to the speed of generation of data.

Big Data Velocity deals with the speed at which data flows in from sources like business processes, application logs, networks and social media sites, sensors, Mobile devices, etc. The flow of data is massive and continuous.

Big Data Analytics

Big data analytics describes the process of uncovering trends, patterns, and correlations in large amounts of raw data to help make data-informed decisions. These processes use familiar statistical analysis techniques—like clustering and regression—and apply them to more extensive datasets with the help of newer tools.

How big data analytics works

Big data analytics refers to collecting, processing, cleaning, and analyzing large datasets to help organizations operationalize their big data.

BIG DATA TECHNOLOGIES

1. Collect Data

Data collection looks different for every organization. With today's technology, organizations can gather both structured and unstructured data from a variety of sources — from cloud storage to mobile applications to in-store IoT sensors and beyond. can access it easily. Raw or unstructured data that is too diverse or complex for a warehouse may be assigned metadata and stored in a data lake.

2. Process Data

Once data is collected and stored, it must be organized properly to get accurate results on analytical queries, especially when it's large and unstructured. Available data is growing exponentially, making data processing a challenge for organizations. One processing option is **batch processing**, which looks at large data blocks over time. Batch processing is useful when there is a longer turnaround time between collecting and analyzing data. **Stream processing** looks at small batches of data at once, shortening the delay time between collection and analysis for quicker decision-making. Stream processing is more complex and often more expensive.

3. Clean Data

Data big or small requires scrubbing to improve data quality and get stronger results; all data must be formatted correctly, and any duplicative or irrelevant . Dirty data can obscure and mislead, creating flawed insights.

4. Analyze Data

Getting big data into a usable state takes time. Once it's ready, advanced analytics processes can turn big data into big insights. Some of these big data analysis methods include:

- **Data mining** sorts through large datasets to identify patterns and relationships by identifying anomalies and creating data clusters.
- **Predictive analytics** uses an organization's historical data to make predictions about the future, identifying upcoming risks and opportunities.
- **Deep learning** imitates human learning patterns by using artificial intelligence and machine learning to layer algorithms and find patterns in the most complex and abstract data.

1. Descriptive Analytics

This summarizes past data into a form that people can easily read. This helps in creating reports, like a company's revenue, profit, sales, and so on. Also, it helps in the tabulation of social media metrics.

Use Case: The Dow Chemical Company analyzed its past data to increase facility utilization across its office and lab space. Using. Dow was able to identify underutilized space. This space consolidation helped the company save nearly US \$4 million annually.

BIG DATA TECHNOLOGIES

2. Diagnostic Analytics

This is done to understand what caused a problem in the first place. Techniques like drill-down, , and data recovery are all examples. Organizations use diagnostic analytics because they provide. Use Case: An e-commerce company's report shows that their sales have gone down, although customers are adding products to their carts. This can be due to various reasons like the form didn't load correctly, the shipping fee is too high, or there are not enough payment options available. This is where you can use diagnostic analytics to find the reason.

3. Predictive Analytics

This type of analytics looks into the historical and present data to make predictions of the future. Predictive analytics uses [data mining](#), AI, and [machine learning](#) to analyze current data and make predictions about the future. It works on predicting customer trends, market trends, and so on.

Use Case: PayPal determines what kind of precautions they have to take to protect their clients against fraudulent transactions. Using [predictive analytics](#), the company uses all the historical payment data and user behavior data and builds an algorithm that predicts fraudulent activities.

4. Prescriptive Analytics

This type of analytics prescribes the solution to a particular problem. Perspective analytics works with both descriptive and predictive analytics. Most of the time, it relies on [AI and machine learning](#).

Exploring the use of big data in business context

In simple terms, Big Data is a combination of all the processes and tools related to utilizing and managing large data sets. The Big Data concept was born out of the need to understand trends, preferences, and patterns in the huge database generated when people interact with different systems and each other., business organizations can use analytics, and figure out the most valuable customers. It can also help businesses create new experiences, services, and products.

Why is big data important for businesses?

Before and tools were developed, many organizations could use only a small fraction of their data in operational and analytics applications. The rest often got pushed to the side as so-called dark data, which is processed and stored but not put to further use. Effective big data management processes enable businesses to better utilize their data assets.

BIG DATA TECHNOLOGIES

Being able to do so expands the kinds of data analytics that companies can run and the business value they can get. Big data creates increased opportunities for [machine learning](#), [predictive analytics](#), data mining, streaming analytics, text mining and other [data science](#) and advanced analytics disciplines. Using those disciplines, big data analytics applications help businesses better understand customers, identify operational issues, detect fraudulent transactions and manage supply chains, among other uses.

If done well, the end results include more effective marketing and advertising campaigns, improved business processes, increased revenue, reduced costs and stronger strategic planning — all of which can lead to better financial results and competitive advantages over business rivals. In addition, big data contributes to breakthroughs in medical diagnoses and treatments, scientific research and smart city initiatives, law enforcement and other government programs.

What are the business benefits of big data?

In an article on big data's business benefits, Donald Farmer, principal of analytics consultancy TreeHive Strategy, described big data as “the lifeblood of modern business.” He cited six potential benefits for organizations: better insight into customers, operational improvements, increased market intelligence, more agile supply chain operations, data-driven product innovation and more sophisticated recommendation engines that are better tuned to the interests and preferences of individual customers.

6 big data benefits for businesses

1. Better customer insight
2. Improved operations
3. More insightful market intelligence
4. Agile supply chain management
5. Data-driven innovation
6. Smarter recommendations and targeting



ART: VISUAL GENERATION/ADOBE STOCK
©2021 TECHTARGET. ALL RIGHTS RESERVED

At a higher level, big data benefits companies by generating actionable insights that enable them to implement data-driven strategies and decision-making. It can also point organizations toward new business opportunities, potential cost savings and emerging market trends. In addition, [real-time analytics](#) applications fueled by big data can be used to provide up-to-date information and alerts about problems to operations managers, call center agents, sales representatives and other frontline workers.

BIG DATA TECHNOLOGIES

1. Implement Risk Management

2. Understand Customers Better

Businesses must understand their customers, and big data will make this possible. From website visits to social media interactions, big data will reveal a lot about your customers. It will also allow the business to create customer profiles and buyer personas. When you understand them better, you can improve your products and services. It allows you to deliver the highest level of satisfaction. This can also help in [building customer loyalty](#).

3. Know Your Competition

[Big data](#) also provides the opportunity to get to know your competitors. For instance, it can provide information about their [pricing models](#) and how it is perceived by the customers. You will also learn about the customers' perceptions of your competitors. Plus, it will help you determine how they are performing online, such as their social media engagement.

4. Personalize Your Marketing

Marketing can make or break your business. For its success, one of the most important is personalization. This is another area where you can use big data. Because it lets you understand your customers, you can create marketing campaigns that will target a specific niche or segment. It can offer insights that will allow marketers to create high-converting materials.

5. Identify Trends

Through big data, it is also possible for businesses to identify trends, which can be useful in product research and development. From customer behaviors to buying patterns, big data will provide the management with the information it needs to analyze how trends will change over time, and in turn, will give the business the time to prepare for these changes. will play key role in doing collecting sample data, survey research and analyzing the information to make decisions.

Use of Big data Social Networking

The extensive use of these big data tactics is clearly demonstrated by the influx of posts, comments, likes, dislikes, followings, and followers from social media sources, such as the top 3 leaders - Facebook, YouTube, and Instagram. Facebook is not going away, as evidenced by Statista's estimate that it had 2.38 billion active monthly users in the first quarter of 2019.

Operating these massive amounts of information created every single second is crucial. Successful firms pay attention to what their consumers say because both positive and bad comments can affect their ability to attract new customers and maintain their good name.

Big data is essential to marketing analytics' ability to forecast future customer behavior without exaggeration. Many businesses invest in big data solution technologies to track customers' experiences in social media in real-time.

Advantages of Using Big Data in social media:

Let's take a quick look at the top 7 advantages of big data analytics for social media marketing.

BIG DATA TECHNOLOGIES

Channels of communication:

AI strategies enable the processing of data from various channels, particularly when synchronization and a widely used log-in technology are used. Many business websites encourage users to sign up using Google or Facebook accounts, allowing marketers to access data from social media activity, browser history, desktop and mobile applications, cloud storage, and other sources to learn more about their customers.

Real-time communication:

The key to a successful market study is user behavior on social media, such as advertising clicked, pages visited and followed, comments left, links saved, and friends added. No other source can provide a more accurate and current picture of market demand. The most important thing is to take advantage of the circumstance earlier than competitors because it changes so quickly.

Intended audience:

Similar to other company endeavors, social media marketing aims to boost sales, but it serves no use in feeding vegan meat. Knowing your intended audience is crucial, therefore. The breadth of ML solutions allows for extracting useful insights from various social network activities, including millions of photographs, music preferences, locations, and many other activities.

Future forecasts:

Using big data strategy and predictive analytics in the media allows for better decision-making based on historical data. Data-driven businesses frequently achieve great success because computers can predict future customer preferences. Even if they evolve over time, habits and interests generally stay connected. Following a purchase on social media, there is a strong likelihood that the consumer will select related goods.

Security concerns:

Private information is extremely important to customers due to the rise of social media and the public presentation of personal information, weird as it may sound. Although there is still much need for improvement in this area, most businesses give security concerns a top priority. Data vendors, marketers, and business owners must provide data security against leaks to unauthorized third parties. Different forms of protection are suggested by big data solutions, such as voice and facial recognition, authorization, check-in notifications, etc.

Campaign analysis:

The seesaw dynamics of ROI indicators may be properly tracked thanks to big data analytics. Marketers can learn more about a social media campaign's success. Predictive analytics tools excel when it comes to predicting the goods and services that customers will demand. Measuring user interactions and responses to online advertisements across various social media platforms can reveal much about consumer behavior and purchasing habits. Overall, the success or failure of a campaign can be predicted based on past customer behavior gathered from social media, historical website data, email subscriptions, and other forms of digital contact.

Affordable costs:

Because so many elements must be considered, pricing selections can occasionally be difficult. Typically, it begins with product costs, problems with competition, market demand, positive revenue, levels of currency and inflation, and finishes with a global economic scenario. In order to fully understand how much your loyal customers are willing to spend on your products, a solid Big Data strategy on social media should not only involve lavish payments to your Instagram influencers. It should also involve regular communication with these customers, perhaps through A/B testing or

BIG DATA TECHNOLOGIES

online surveys. All of this can assist marketers in making more precise and flexible price adjustments to meet client expectations.

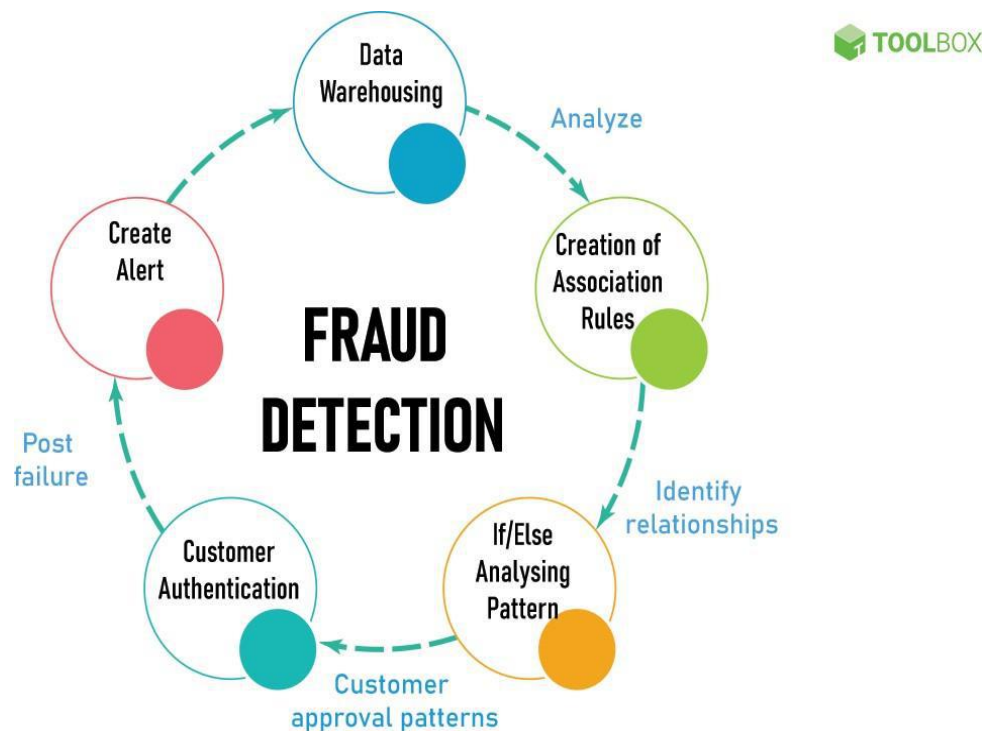
Innovation potential:

Through media monitoring, businesses can thoroughly grasp their goods and target market using data science Tools for social media analytics that can be set up to find market-wide capability gaps. For instance, user input expressing a need for lighter, more relaxed running shoes helped to propel the minimalist innovation in the market for running shoes. The most prosperous businesses in recent years have been those that can mine consumer feedback from social media platforms and use it to reinvent their businesses.

Use of big data Preventing Fraudulent Activities

FraudDetectionandPreventionDefinition

[Banking](#) and [healthcare](#) fraud account for tens of billions of dollars in losses annually, which results in compromised financial institutions, personal impact for bank clients, and higher premiums for patients. Fraud detection and prevention refers to the strategies undertaken to detect and prevent attempts to obtain money or property through deception.



What is Fraud Detection and Prevention?

Fraudulent activities can encompass a wide range of cases, including money laundering, cybersecurity threats, tax evasion, fraudulent insurance claims, forged bank checks, identity theft, and terrorist financing, and is prevalent throughout the financial institutions, government, healthcare, and insurance sectors.

BIG DATA TECHNOLOGIES

To combat this growing list of opportunities for fraudulent transactions, organizations are implementing modern fraud detection and prevention technologies and risk management strategies, which combine big data sources with real-time monitoring, and apply adaptive and [predictive analytics](#) techniques, such as Machine Learning, to create a risk of fraud score.

Detecting fraud with data analytics, fraud detection software and tools, and a fraud detection and prevention program enables organizations to predict conventional fraud tactics, cross-reference data through automation, manually and continually monitor transactions and crimes in real time, and decipher new and sophisticated schemes.

Fraud detection and prevention software is available in both proprietary and open source versions. Common features in fraud analytics software include: a dashboard, data import and export, data visualization, customer relationship management integration, calendar management, budgeting, scheduling, multi-user capabilities, password and access management, Application Programming Interfaces (API), two-factor authentication, billing, and customer database management.

FraudDetectionandPreventionTechniques

Fraud data analytics methodologies can be categorized as either statistical data analysis techniques or artificial intelligence (AI).

Statistical data analysis techniques include:

- Statistical parameter calculation, such as averages, quantiles, and performance metrics
- Regression analysis - estimates relationships between independent variables and a dependent variable
- Probability distributions and models
- Data matching - used to compare two sets of collected data, remove duplicate records, and identify links between sets
- Time-series analysis

AI techniques include:

- Data mining - data mining for fraud detection and prevention classifies and segments data groups in which millions of transactions can be performed to find patterns and detect fraud
- Neural networks - suspicious patterns are learned and used to detect further repeats
- Machine Learning - fraud analytics Machine Learning automatically identifies characteristics found in fraud
- Pattern recognition - detects patterns or clusters of suspicious behavior

The four most crucial steps in the fraud prevention and detection process include:

- Capture and unify all manner of data types from every channel and incorporate them into the analytical process.
- Continually monitor all transactions and employ behavioral analytics to facilitate real-time decisions.
- Incorporate analytics culture into every facet of the enterprise through data visualization.
- Employ layered security techniques.

BIG DATA TECHNOLOGIES

Use of big data in Retail industry

Broadly, the use of Big Data technology components has many benefits that can facilitate the profit-making potential of businesses and accelerate the efficiency of work management. Here is a brief list of Big Data benefits to help you understand the concept well.

Big Data in Retail Sector

Even though you might have understood that Big Data is very helpful in our everyday lives, you will now be reading about Big Data in the retail sector.

When it comes to the use of Big Data in the retail sector, a lot of Big Data applications can be included in this discussion. Let us discover the applications of Big Data in the Retail sector 2020.

1. Customer Preferences

One of the most prominent uses of Big Data in the retail sector is that such a vast amount of data can very well provide accurate results for customer preferences.

Thanks to technological advancements, Big [data analysis techniques](#) can help [machine learning algorithms](#) to dig into data and conjecture customer preferences based on an individual's past purchase records and buying history.

In the retail sector, understanding the psyche of a customer and analyzing the kind of products s/he desires is extremely helpful to promote the target products and help companies make profits in considerable time.

With Big Data, this can be achieved in the contemporary marketing scenario where technology has opened many windows for retail professionals to understand what their customers actually want.

BIG DATA TECHNOLOGIES

Even though [data mining](#) scientists suggest that a handful of datasets cannot accurately suggest what kind of customer preferences are the most desired, there is some other revelation that they have to make.

On the other hand, as they suggest, Big [data science tools](#) can accurately indicate customer preferences by employing trained algorithms with as many samples as possible.

2. Optimization of Product Portfolios

While customer preferences are one aspect of the retail sector, product reports are another such aspect that holds a very eminent position in this field. In light of this, Big Data can also be applied in the optimization of product portfolios.

As customer feedback reports are accumulated through the source of Big Data, sales departments of various companies can optimize their product images and overall portfolios.

This helps them to understand what additions they can put in their products and what features are unwanted by customers.

Not only is this helpful for both customers and companies but it also makes the two-way flow of commodities much more efficient and enhanced.

The collection of customer feedback reports also helps to evaluate a company's performance in the long run and offers remotely available reviews for the progress of a brand.

"Aldo is a shoe and accessory company based in Canada that uses big data to address this crazy time of year. The company operates on a service-oriented big data architecture, integrating multiple data sources involved in payment, billing, and fraud detection. This integration project enables Aldo to deliver a seamless ecommerce experience—even on Black Friday."

BIG DATA TECHNOLOGIES

[Big Data Uses in Retail Industry](#)

3. Supply Chain Management

Another very important application of Big Data analytics can be observed in the field of the supply chain. In the retail sector, the supply of products is an extremely crucial role as it brings the products from one point to the other.

In such a case, many loopholes can occur in between the supply chain while products are being sent from one end to the other. Perhaps this is where Big Data enters into the scene and plays its role.

In every [supply chain management](#) system, Big Data can forecast any inefficiencies and can also detect anomalies, in case there are any.

With the ability to detect anomalies, Big Data can break down obstructions in supply chain management and can readily serve as a self-healing chain that lets the retail sector rectify such errors.

For instance, a chocolate manufacturing company supplies a fixed number (for instance, 100) of products to a grocery store in your locality.

However, due to some glitch, the process only supplies 25 pieces in the month of March. Yet, Big Data can run through such data and detect any such anomalies that will instantly let the company know about any loophole.

4. Detecting Quality Deficits

Besides detecting anomalies in the supply chain management, Big Data can also help the retail sector to identify quality defects in the products manufactured by a company.

BIG DATA TECHNOLOGIES

Retail chains use Big Data analytics to dig into data for quality deficits and evaluate a product's performance in the market. That said, Big Data in the retail sector can be of great use to data scientists who employ several procedures and mechanisms powered by AI. (Learn more about [how big data analytics uses AI](#))

In order to boost quality improvement in retail businesses, data scientists look for the best procedures to conduct retail operations and find the perfect match for their quality standards in order to compete with other such dealers and provide the best they can.

"Once we have derived intelligence from this data, we will be able to apply universal best practices across all plants to better align processes and improve quality performance."

[Big Data in Retail Sector](#)

5. Predictive Analysis of Sales

Even though you might have been awestruck by the applications of Big Data Analytics in the retail sector, there is one more use that can fully amaze you.

With the help of [predictive analytics tools](#), Big data scientists can conduct predictive analysis of sales that forecasts product demands in various demographic areas and among various target groups.

One of the biggest things that Big Data provides us with is vast amounts of data records obtained from the past and present. Thus, these records can be used to identify what seasons or what time periods witness more sales of a particular product.

In turn, this identification helps companies in the retail sector to prepare accordingly and deploy their sales teams during such times in the year. Apart from being an application, this use of Big Data also serves as the biggest benefit of this technology in the retail sector.

BIG DATA TECHNOLOGIES

UNIT-II

Handling Big Data Distributed and Parallel Computing

Distributed computing refers to a system where processing and data storage is distributed across multiple devices or systems, rather than being handled by a single central device. In a distributed system, each device or system has its own processing capabilities and may also store and manage its own data. These devices or systems work together to perform tasks and share resources, with no single device serving as the central hub.

One example of a distributed computing system is a cloud computing system, where resources such as computing power, storage, and networking are delivered over the Internet and accessed on demand. In this type of system, users can access and use shared resources through a web browser or other client software.

Components

There are several key components of a Distributed Computing System

- **Devices or Systems:** The devices or systems in a distributed system have their own processing capabilities and may also store and manage their own data.
- **Network:** The network connects the devices or systems in the distributed system, allowing them to communicate and exchange data.
- **Resource Management:** Distributed systems often have some type of resource management system in place to allocate and manage shared resources such as computing power, storage, and networking.

Characteristics

There are several characteristics that define a Distributed Computing System

- **Multiple Devices or Systems:** Processing and data storage is distributed across multiple devices or systems.
- **Peer-to-Peer Architecture:** Devices or systems in a distributed system can act as both clients and servers, as they can both request and provide services to other devices or systems in the network.
- **Shared Resources:** Resources such as computing power, storage, and networking are shared among the devices or systems in the network.
- **Horizontal Scaling:** Scaling a distributed computing system typically involves adding more devices or systems to the network to increase processing and storage capacity. This can be done through hardware upgrades or by adding additional devices or systems to the network..

Advantages and Disadvantages Advantages of the Distributed Computing System are:

- **Scalability:** Distributed systems are generally more scalable than centralized systems, as they can easily add new devices or systems to the network to increase processing and storage capacity.
- **Reliability:** Distributed systems are often more reliable than centralized systems, as they can continue to operate even if one device or system fails.
- **Flexibility:** Distributed systems are generally more flexible than centralized systems, as they can be configured and reconfigured more easily to meet changing computing needs.

BIG DATA TECHNOLOGIES

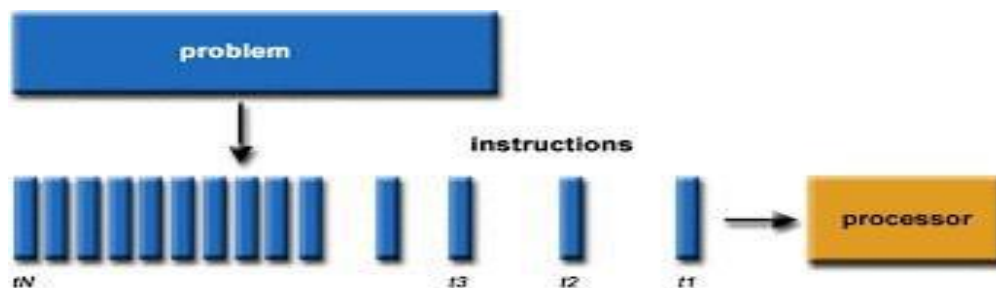
There are a few limitations to Distributed Computing System

- **Complexity:** Distributed systems can be more complex than centralized systems, as they involve multiple devices or systems that need to be coordinated and managed.
- **Security:** It can be more challenging to secure a distributed system, as security measures must be implemented on each device or system to ensure the security of the entire system.
- **Performance:** Distributed systems may not offer the same level of performance as centralized systems, as processing and data storage is distributed across multiple devices or systems.

Applications Distributed Computing Systems have a number of applications, including:

- **Cloud Computing:** Cloud Computing systems are a type of distributed computing system that are used to deliver resources such as computing power, storage, and networking over the Internet.
- **Peer-to-Peer Networks:** Peer-to-Peer Networks are a type of distributed computing system that is used to share resources such as files and computing power among users.
- **Distributed Architectures:** Many modern computing systems, such as microservices architectures, use distributed architectures to distribute processing and data storage across multiple devices or systems.

What is Parallel Computing?



Parallel computing refers to the process of executing several processors an application or computation simultaneously. Generally, it is a kind of computing architecture where the large problems break into independent, smaller, usually similar parts that can be processed in one go. It is done by multiple CPUs communicating via shared memory, which combines results upon completion. It helps in performing large computations as it divides the large problem between more than one processor.

Parallel computing also helps in faster application processing and task resolution by increasing the available computation power of systems. The parallel computing principles are used by most supercomputers employ to operate. The operational scenarios that need massive processing power or computation, generally, parallel processing is commonly used there.

Typically, this infrastructure is housed where various processors are installed in a server rack; the application server distributes the computational requests into small chunks then the requests are processed simultaneously on each server. The earliest computer software is written for serial computation as they are able to execute a single instruction at one time, but parallel computing is different where it executes several processors an application or computation in one time.

BIG DATA TECHNOLOGIES

There are many reasons to use parallel computing, such as save time and money, provide concurrency, solve larger problems, etc. Furthermore, parallel computing reduces complexity. In the real-life example of parallel computing, there are two queues to get a ticket of anything; if two cashiers are giving tickets to 2 persons simultaneously, it helps to save time as well as reduce complex.

Types of parallel computing

From the open-source and proprietary parallel computing vendors, there are generally three types of parallel computing available, which are discussed below:

1. **Bit-level parallelism:** The form of parallel computing in which every task is dependent on processor word size. In terms of performing a task on large-sized data, it reduces the number of instructions the processor must execute. There is a need to split the operation into series of instructions. For example, there is an 8-bit processor, and you want to do an operation on 16-bit numbers. First, it must operate the 8 lower-order bits and then the 8 higher-order bits. Therefore, two instructions are needed to execute the operation. The operation can be performed with one instruction by a 16-bit processor.
2. **Instruction-level parallelism:** In a single CPU clock cycle, the processor decides in instruction-level parallelism how many instructions are implemented at the same time. For each clock cycle phase, a processor in instruction-level parallelism can have the ability to address that is less than one instruction. The software approach in instruction-level parallelism functions on static parallelism, where the computer decides which instructions to execute simultaneously.
3. **Task Parallelism:** Task parallelism is the form of parallelism in which the tasks are decomposed into subtasks. Then, each subtask is allocated for execution. And, the execution of subtasks is performed concurrently by processors.

Cloud Computing

Rather than keeping files on a proprietary hard drive or local storage device, cloud-based storage makes it possible to save them to a remote database. As long as an electronic device has access to the web, it has access to the data and the software programs to run it.

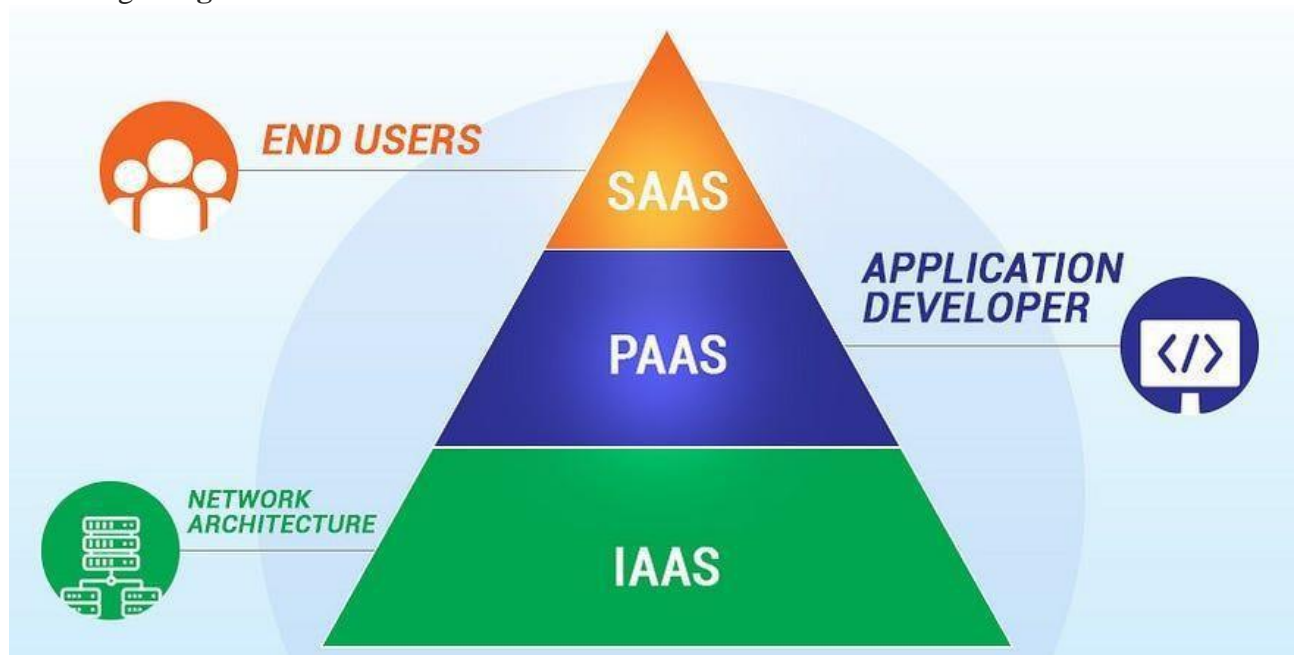
Cloud computing is named as such because the information being accessed is found remotely in the cloud or a virtual space. Companies that provide cloud services enable users to store files and applications on remote servers and then access all the data via the Internet. This means the user is not required to be in a specific place to gain access to it, allowing the user to work remotely.

BIG DATA TECHNOLOGIES



Cloud computing takes all the heavy lifting involved in crunching and processing data away from the device you carry around or sit and work at. It also moves all of that work to huge computer clusters far away in cyberspace. The Internet becomes the cloud, and voilà — your data, work, and applications are available from any device with which you can connect to the Internet, anywhere in the world.

The array of available cloud computing services is vast, but most fall into one of the following **categories**:



BIG DATA TECHNOLOGIES

Software-as-a-service [SaaS]

Software as a Service Software as a service represents the largest cloud market and most commonly used business option in cloud services. SaaS delivers applications to users over the internet. Applications that are delivered through SaaS are maintained by third-party vendors and interfaces are accessed by the client through the browser. Since most SaaS applications run directly from a browser, it eliminates the need for the client to download or install any software. In SaaS vendors manage applications, runtime, data, middleware, OS, virtualization, servers, storage, and networking which makes it easy for enterprises to streamline their maintenance and support.

Platform-as-a-service [PaaS]

Platform as a Service Platform as a Service model provides hardware and software tools over the internet which are used by developers to build customized applications. PaaS makes the development, testing, and deployment of applications quick, simple, and cost-effective. This model allows businesses to design and create applications that are integrated into PaaS software components while the enterprise operations or third-party providers manage OS, virtualization, servers, storage, networking, and the PaaS software itself. These applications are scalable and highly available since they have cloud characteristics.

Infrastructure-as-a-service [IaaS]

Infrastructure as a Service Infrastructure as a Service cloud computing model provides a self-servicing platform for accessing, monitoring, and managing remote data center infrastructures such as compute, storage and networking services to organizations through virtualization technology. IaaS users are responsible for managing applications, data, runtime, middleware, and OS while providers still manage virtualization, servers, hard drives, storage, and networking. IaaS provides the same capabilities as data centers without having to maintain them physically.

In-Memory Computing Technology

BIG DATA TECHNOLOGIES

In-memory Computing Definition

In-memory computing (or in-memory computation) is a technique based on RAM data storage and indexing, which proposed by the MIT research group, and its main purpose is to accelerate the convolution calculation. We know that convolution calculations can be expanded into weighted accumulation calculations. From another perspective, it is actually a weighted average of multiple numbers. Therefore, the circuit realizes the weighted average of the charge domain. The weight (1-bit) is stored in SRAM, and the input data (7-bit digital signal) becomes an analog signal through the DAC. According to the corresponding weight in the SRAM, the output is multiplied by 1 or -1 in the analog domain, which averaged in the analog domain, and finally read out by the ADC as a digital signal. Specifically, since the weight of the multiplication is 1-bit (1 or -1), it can be controlled by using a switch and a differential line simply. If the weight is 1, the capacitor on the side of the differential line is charged to the required output value. Otherwise, let the other side of the differential line be charged to this value. As for average, connect several differential lines together in the charge domain.

Of course, there is more than one circuit for in-memory calculation, and the calculation accuracy is not limited to 1-bit. However, we can see the above examples that the core idea of in-memory calculations is generally to convert calculations into weighted calculations. Store the weights in the memory unit, then modifications on the core circuit of the memory (such as the readout circuit) are made. So that the process of reading is like a process in which the input data and weights are multiplied in the analog domain, that is, convolution. Because convolution is a core part of AI and other calculations, in-memory computing can be widely used in such applications. In-memory computing uses analog circuits for calculations, which is the difference compared with traditional digital logic calculations.

In more traditional architectures, there are some multiply-accumulate circuits (MAC) for tensor math, especially the matrix multiplication. These architectures attempt to arrange the MAC in a way that moves weights and activations to the appropriate location. Activations are calculated from the previous neural network layer. Multiplication usually involves activations and weights, both must be moved to the place where multiplies them. In-memory computing makes use of it. Therefore, if the weights are stored in memory, the memory can access through activations to obtain multiplication and accumulation. The only difference from the actual memory is that the in-memory computing concatenates all word lines at once, instead of decoding the input to get one word line only.

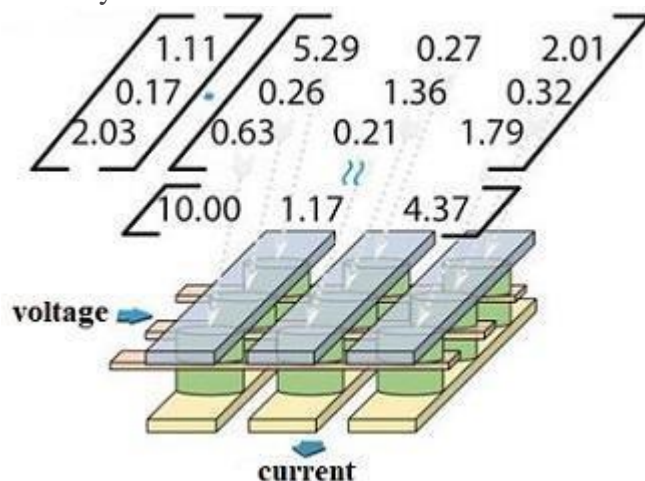


Figure 4. In-memory Computing Diagram

3.2 Four Realization Methods

The attempt is to enter the analog domain and treat the storage unit as an analog unit instead of a digital unit to reduce consumption. We have already got a way to use simulation on the front end of the inference engine. That is in-memory computing. Therefore, we take digital data, using a DAC to

BIG DATA TECHNOLOGIES

convert it to an analog value, and then driving a memory with this analog content to obtain an analog bit-line output, finally using an ADC to convert the result back to a digital format. However, the in-memory computing is still in the exploratory stage, and there are many specific implementation methods to study, currently there are three types: RRAM, Flash, SRAM, and DRAM.

- **Based on RRAM**

RRAM is the most common method of doing this, because it is easy to use by applying Ohm's law to a series of resistors, but it still has the problem of relying on RRAM. The relationship between programming and resistance is non-linear, which requires more work to be done to make viable calculation circuits in RRAM memory for market. So it is just an idea, and the specific plan is still under study.

- **Based on Flash**

NOR Flash memory has a more traditional word-line/bit-line structure. It is both resistive and capacitive. Generally, the memory cell is a transistor that is turned on or off. However, if it is partially conductive, it can be used as a resistor. The resistance depends on the amount of charge on the floating gate of the memory cell (capacitor). When running all the time, the cell will conduct to its maximum capacity. During this process, it does not conduct at all, however, it can be partially programmed. There is a problem is that you cannot precisely control the number of electrons. Moreover, the response to any number will vary with the process and temperature and other variables.

Two companies are studying this method. Microchip owns their memBrain array, thanks to their acquisition of SST, and Mythic is a start-up company dedicated to an inference engine that uses in-memory computing with flash memory. Both companies said that they are using extensive calibration techniques to deal with this change.

So, they need a large number of high-quality ADCs and DACs to keep the signal-to-noise ratio (SNR) within a scope of accurate reasoning, which is the focus of designing work. Mythic claims that they provide a novel ADC, so that Microchip can share it to reduce the number required. Although ADC does consume energy, it also greatly reduces overall system consumption.

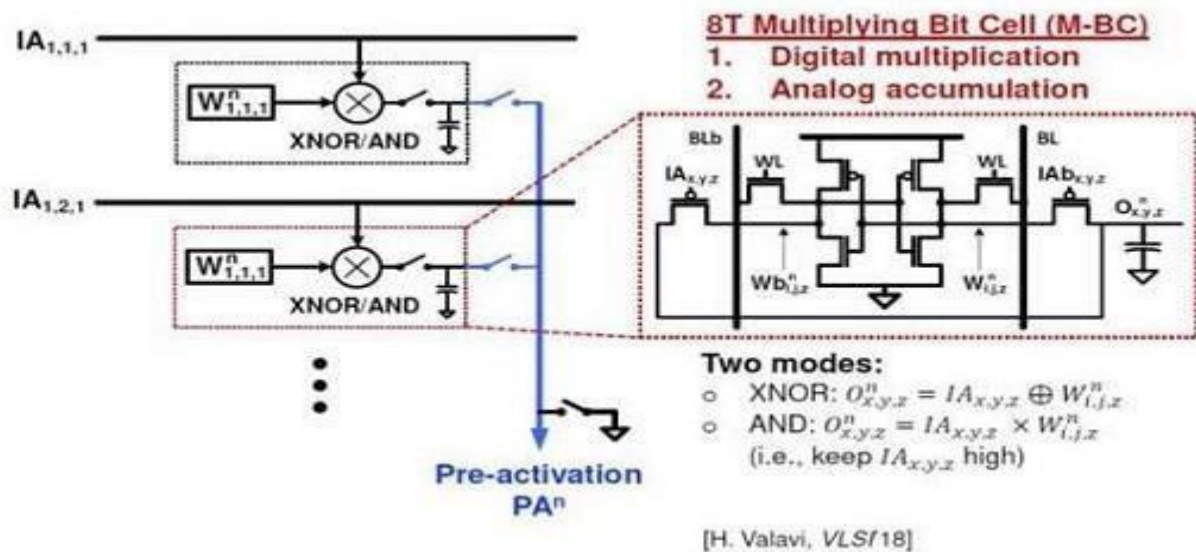
- **Based on SRAM**

This idea came from a lecture at Hot Chips at Princeton University. By definition, SRAM is a bistable unit. Therefore, it cannot be in an intermediate state, how should this be handled? And the DACs and ADCs that need to be corrected more over than the array in terms of area and power consumption.

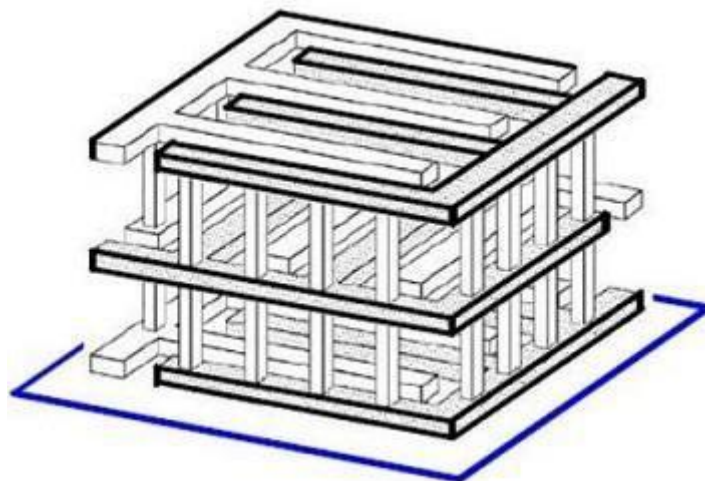
The point of this problem boils down to the question of how to simulate. They explained that this method uses more than one-bit line for calculation. Since the unit is still a digital value, it takes several bit lines to perform a calculation. The bit line can be split, and different groups perform different multiplications. The following figure illustrates it.

With 8 inputs at a time, so the input vector is sliced and several consecutive multiplications are carried out to obtain the final results. The bit line charge is deposited on the capacitor. When ready to read, the charge is read out and sent to the ADC for conversion back to the digital domain. Their basic unit structure is as follows:

BIG DATA TECHNOLOGIES



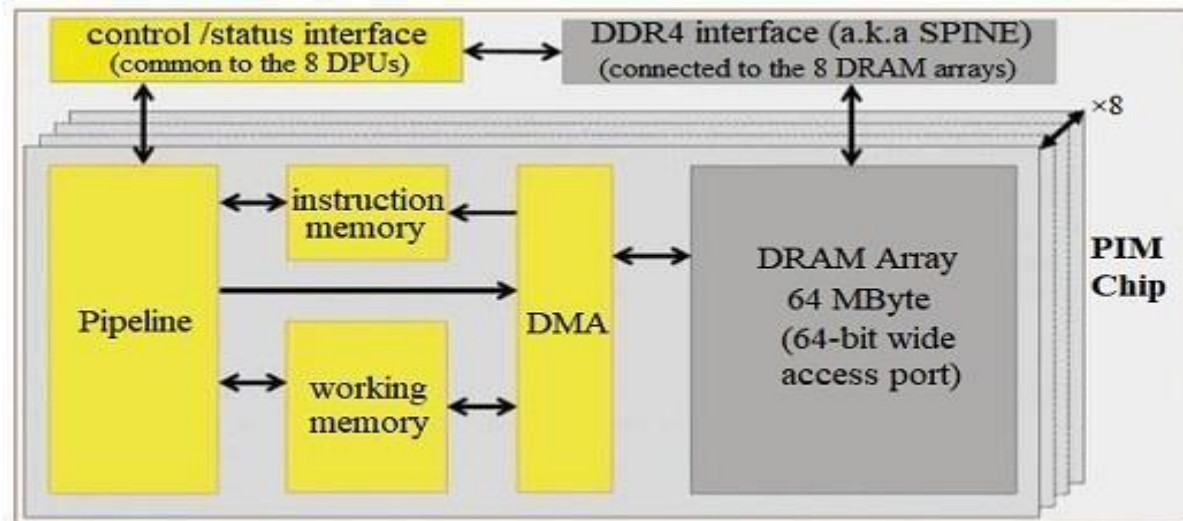
These capacitors may affect chip size issues, but they said that the metal above the cell can be used. Of course, one cell is now 80% larger than the standard 6T SRAM cell (even without capacitors), but they say that their overall circuit is still much smaller than a required circuit based on standard digital implementation. In addition, since their basic array operations are still in digital form, they are less sensitive to noise and changes, which means their ADCs can be simpler and consume less power.



- **Based on DRAM**

This idea refers to not using a lot of power to obtain DRAM content, and in some way incorporate calculations into the CPU or other computing structures and directly run it on the DRAM die, which is what UPMEM does. A simple processor is built on the DRAM die, also the architecture will not compete with Xeon chips, they call this set "processing in memory" or PIM.

BIG DATA TECHNOLOGIES



Instead of bringing data to calculations, they bring calculations to data. The runtime is performed by the CPU in DRAM chip. That is, there is no need to move the data to any location outside of the DRAM chip, just send the calculating result back to the host system. Also, since ML calculations usually involve a lot of reduction, less data required for calculations. Although this does require some minor changes to the DRAM, they did not change the manufacturing process. Under this case, a standard DRAM module will provide multiple opportunities for distributed computing. At the same time, it becomes complicated to use this function to write a program.

Introduction of Hadoop

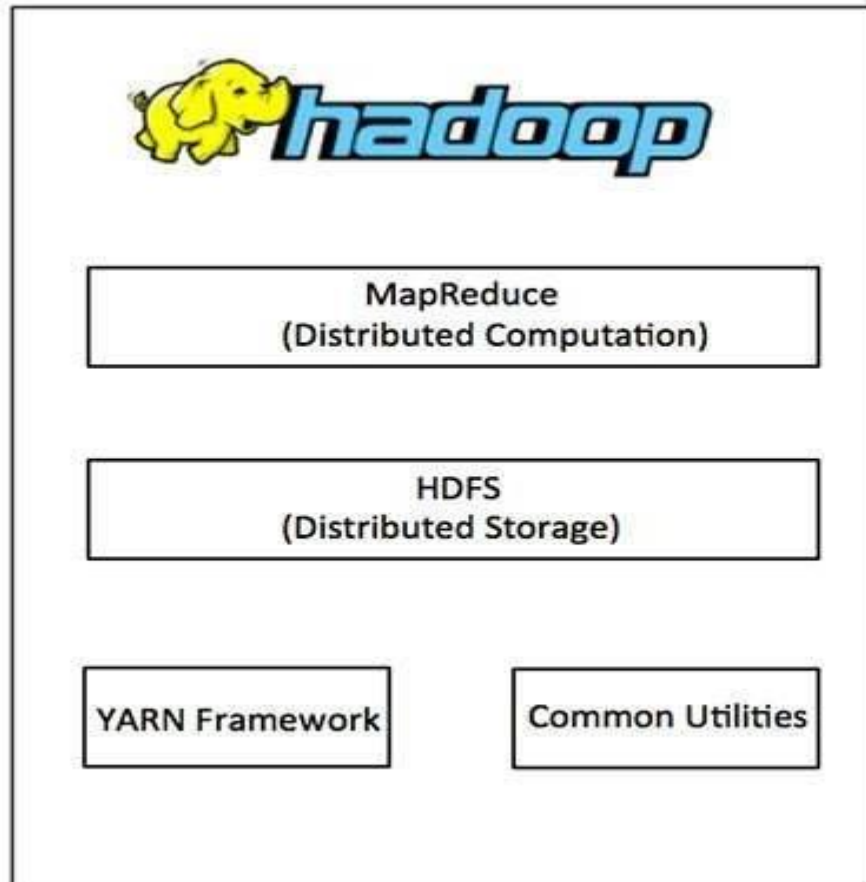
Hadoop is an Apache open-source framework written in java that allows distributed processing of large datasets across clusters of computers using simple programming models. The Hadoop framework application works in an environment that provides distributed *storage* and *computation* across clusters of 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).

BIG DATA TECHNOLOGIES



MapReduce

MapReduce is a parallel programming model for writing distributed applications devised at Google for efficient processing of large amounts of data (multi-terabyte data-sets), on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner. The MapReduce program runs on Hadoop which is an Apache open-source framework.

Hadoop Distributed File System

The Hadoop Distributed File System (HDFS) is based on the Google File System (GFS) and provides a distributed file system that is designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. It is highly fault-tolerant and is designed to be deployed on low-cost hardware. It provides high throughput access to application data and is suitable for applications having large datasets.

Apart from the above-mentioned two core components, Hadoop framework also includes the following two modules –

- **Hadoop Common** – These are Java libraries and utilities required by other Hadoop modules.
- **Hadoop YARN** – This is a framework for job scheduling and cluster resource management.

How Does Hadoop Work?

It is quite expensive to build bigger servers with heavy configurations that handle large scale processing, but as an alternative, you can tie together many commodity computers with single-CPU, as a single functional distributed system and practically, the clustered machines can read the dataset in parallel and provide a much higher throughput. Moreover, it is cheaper than one high-end server. So this is the first motivational factor behind using Hadoop that it runs across clustered and low-cost machines.

BIG DATA TECHNOLOGIES

Hadoop runs code across a cluster of computers. This process includes the following core tasks that Hadoop performs –

- Data is initially divided into directories and files. Files are divided into uniform sized blocks of 128M and 64M (preferably 128M).
- These files are then distributed across various cluster nodes for further processing.
- HDFS, being on top of the local file system, supervises the processing.
- Blocks are replicated for handling hardware failure.
- Checking that the code was executed successfully.
- Performing the sort that takes place between the map and reduce stages.
- Sending the sorted data to a certain computer.
- Writing the debugging logs for each job.

Advantages of Hadoop

- Hadoop framework allows the user to quickly write and test distributed systems. It is efficient, and it automatically distributes the data and work across the machines and in turn, utilizes the underlying parallelism of the CPU cores.
- Hadoop does not rely on hardware to provide fault-tolerance and high availability (FTHA), rather Hadoop library itself has been designed to detect and handle failures at the application layer.
- Servers can be added or removed from the cluster dynamically and Hadoop continues to operate without interruption.
- Another big advantage of Hadoop is that apart from being open source, it is compatible on all the platforms since it is Java based.

Understanding Hadoop Ecosystem and Distributed file systems

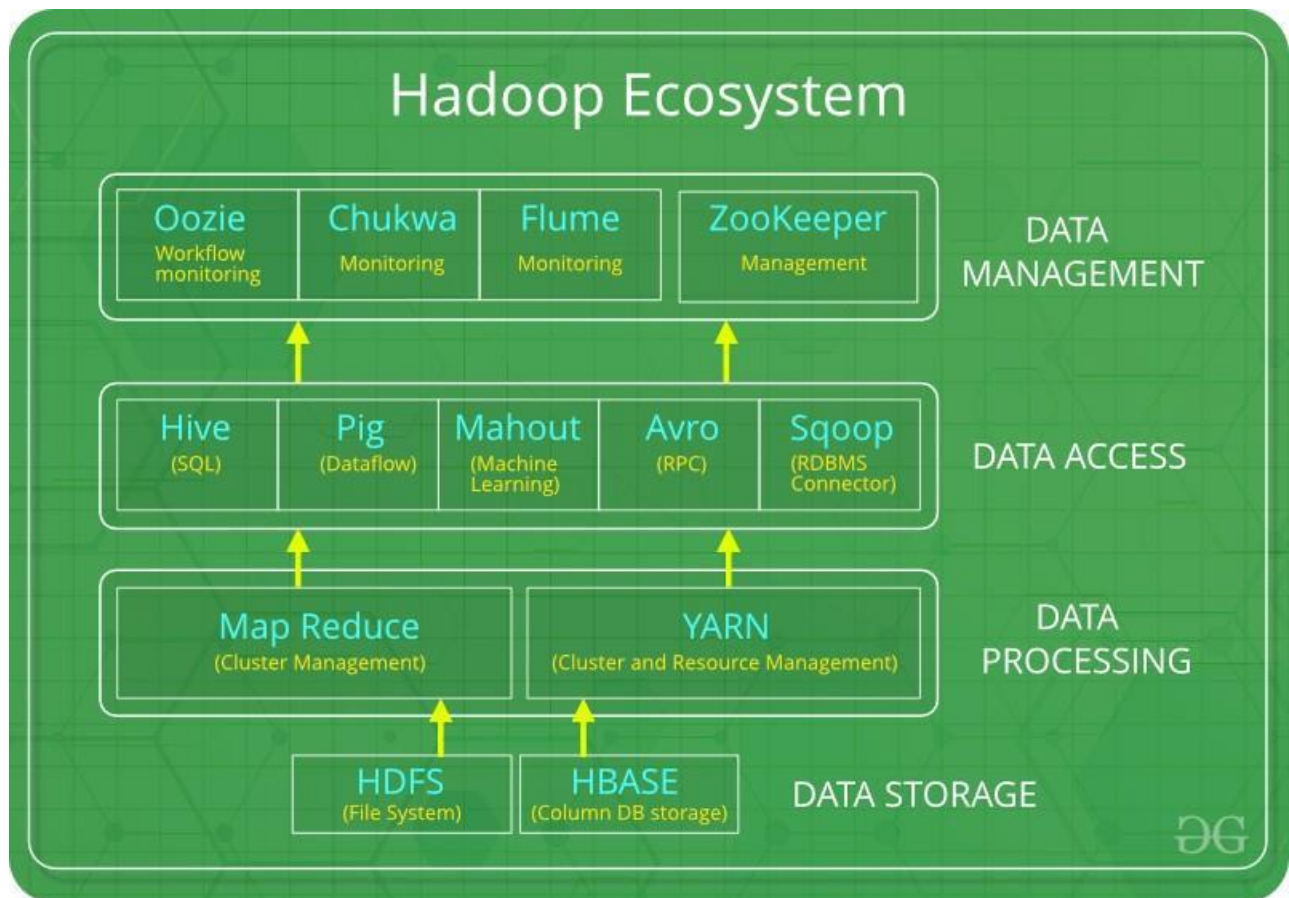
Apache Hadoop is an open source framework intended to make interaction with **big data** easier. However, for those who are not acquainted with this technology, one question arises that what is big data? Big data is a term given to the data sets which can't be processed in an efficient manner with the 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 need 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

BIG DATA TECHNOLOGIES



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

BIG DATA TECHNOLOGIES

- 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 on 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.

BIG DATA TECHNOLOGIES

- 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.

Introduction of HBase

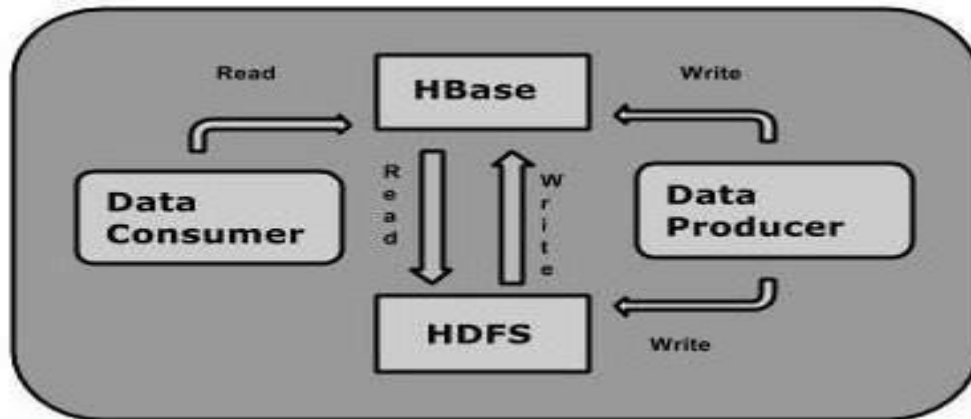
HBase is a distributed column-oriented database built on top of the Hadoop file system. It is an open-source project and is horizontally scalable.

HBase is a data model that is similar to Google's big table designed to provide quick random access to huge amounts of structured data. It leverages the fault tolerance provided by the Hadoop File System (HDFS).

It is a part of the Hadoop ecosystem that provides random real-time read/write access to data in the Hadoop File System.

One can store the data in HDFS either directly or through HBase. Data consumer reads/accesses the data in HDFS randomly using HBase. HBase sits on top of the Hadoop File System and provides read and write access.

BIG DATA TECHNOLOGIES



HBase and HDFS

HDFS	HBase
HDFS is a distributed file system suitable for storing large files.	HBase is a database built on top of the HDFS.
HDFS does not support fast individual record lookups.	HBase provides fast lookups for larger tables.
It provides high latency batch processing; no concept of batch processing.	It provides low latency access to single rows from billions of records (Random access).
It provides only sequential access of data.	HBase internally uses Hash tables and provides random access, and it stores the data in indexed HDFS files for faster lookups.

Storage Mechanism in HBase

HBase is a **column-oriented database** and the tables in it are sorted by row. The table schema defines only column families, which are the key value pairs. A table have multiple column families and each column family can have any number of columns. Subsequent column values are stored contiguously on the disk. Each cell value of the table has a timestamp. In short, in an HBase:

- Table is a collection of rows.
- Row is a collection of column families.
- Column family is a collection of columns.
- Column is a collection of key value pairs.

Given below is an example schema of table in HBase.

BIG DATA TECHNOLOGIES

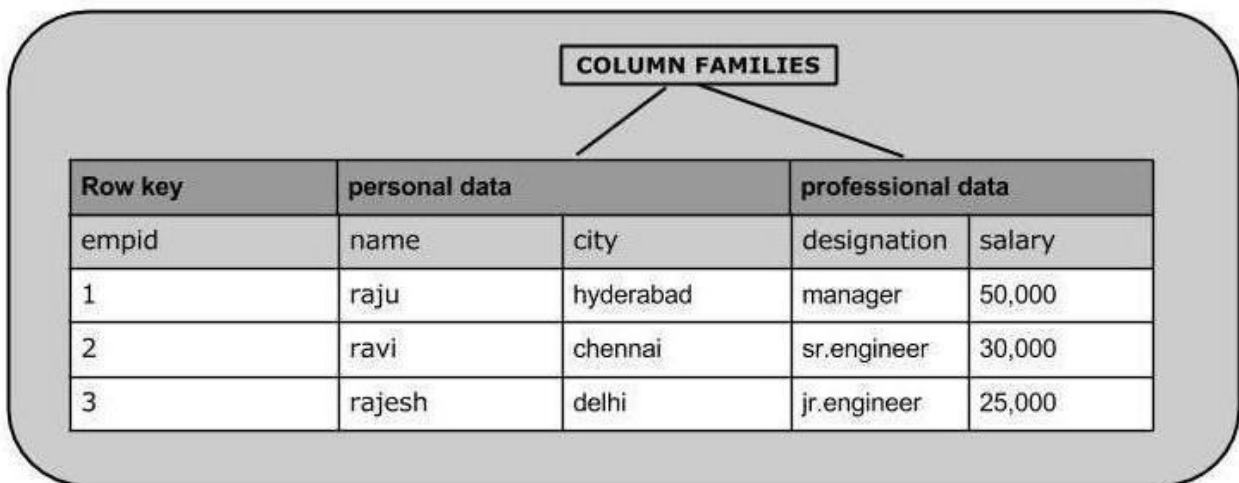
Rowid	Column Family			Column Family			Column Family			Column Family		
	col1	col2	col3	col1	col2	col3	col1	col2	col3	col1	col2	col3
1												
2												
3												
4												

Column Oriented and Row Oriented

Column-oriented databases are those that store data tables as sections of columns of data, rather than as rows of data. Shortly, they will have column families.

Row-Oriented Database	Column-Oriented Database
It is suitable for Online Transaction Process (OLTP).	It is suitable for Online Analytical Processing (OLAP).
Such databases are designed for small number of rows and columns.	Column-oriented databases are designed for huge tables.

The following image shows column families in a column-oriented database:



HBase	RDBMS
HBase is schema-less, it doesn't have the concept of fixed columns schema; defines only column families.	An RDBMS is governed by its schema, which describes the whole structure of tables.
It is built for wide tables. HBase is horizontally scalable.	It is thin and built for small tables. Hard to scale.
No transactions are there in HBase.	RDBMS is transactional.
It has de-normalized data.	It will have normalized data.

BIG DATA TECHNOLOGIES

It is good for semi-structured as well as structured data.

It is good for structured data.

Features of HBase

- HBase is linearly scalable.
- It has automatic failure support.
- It provides consistent read and writes.
- It integrates with Hadoop, both as a source and a destination.
- It has easy java API for client.
- It provides data replication across clusters.

Where to Use HBase

- Apache HBase is used to have random, real-time read/write access to Big Data.
- It hosts very large tables on top of clusters of commodity hardware.
- Apache HBase is a non-relational database modeled after Google's Bigtable. Bigtable acts up on Google File System, likewise Apache HBase works on top of Hadoop and HDFS.

Applications of HBase

- It is used whenever there is a need to write heavy applications.
- HBase is used whenever we need to provide fast random access to available data.
- Companies such as Facebook, Twitter, Yahoo, and Adobe use HBase internally.

UNIT-III

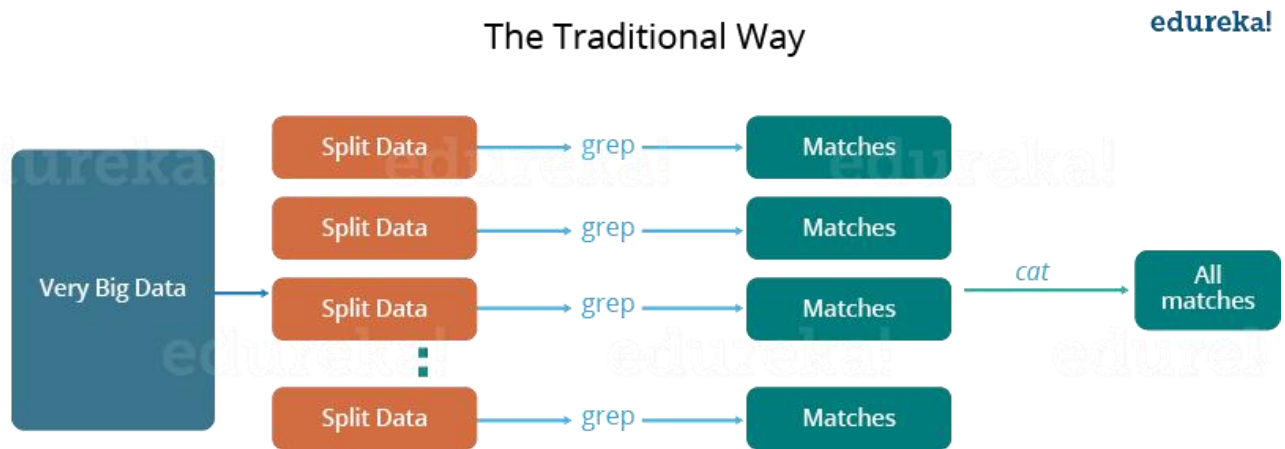
Map Reduce Fundamentals and H base Map Reduce Frame works

I am going to introduce you to MapReduce, which is one of the core building blocks of processing in the *Hadoop framework*. Before moving ahead, I would suggest you get familiar with HDFS concepts which I have covered in my previous *HDFS tutorial* blog. This will help you to understand the MapReduce concepts quickly and easily.

Google released a paper on MapReduce technology in December 2004. This became the genesis of the Hadoop Processing Model. So, MapReduce is a programming model that allows us to perform parallel and distributed processing on huge datasets.

Traditional Way

BIG DATA TECHNOLOGIES



Traditional Way - MapReduce Tutorial

Let us understand, when the MapReduce framework was not there, how parallel and distributed processing used to happen in a traditional way. So, let us take an example where I have a weather log containing the daily average temperature of the years from 2000 to 2015. Here, I want to calculate the day having the highest temperature each year.

So, just like in the traditional way, I will split the data into smaller parts or blocks and store them in different machines. Then, I will find the highest temperature in each part stored in the corresponding machine. At last, I will combine the results received from each of the machines to have the final output. Let us look at the challenges associated with this traditional approach:

1. **Critical path problem:** It is the amount of time taken to finish the job without delaying the next milestone or actual completion date. So, if, any of the machines delay the job, the whole work gets delayed.
2. **Reliability problem:** What if, any of the machines which are working with a part of data fails? The management of this failover becomes a challenge.
3. **Equal split issue:** How will I divide the data into smaller chunks so that each machine gets even part of data to work with. In other words, how to equally divide the data such that no individual machine is overloaded or underutilized.

BIG DATA TECHNOLOGIES

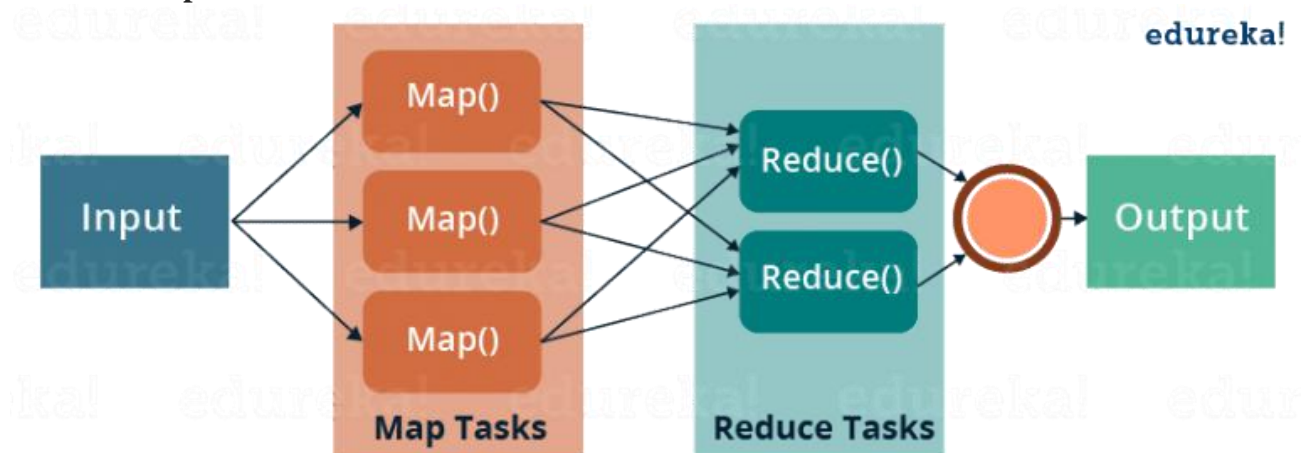
4. **Single split may fail:** If any of the machines fail to provide the output, I will not be able to calculate the result. So, there should be a mechanism to ensure this fault tolerance capability of the system.
5. **Aggregation of the result:** There should be a mechanism to aggregate the result generated by each of the machines to produce the final output.

These are the issues which I will have to take care individually while performing parallel processing of huge datasets when using traditional approaches.

To overcome these issues, we have the MapReduce framework which allows us to perform such parallel computations without bothering about the issues like reliability, fault tolerance etc.

Therefore, MapReduce gives you the flexibility to write code logic without caring about the design issues of the system.

What is MapReduce?



What is MapReduce - MapReduce Tutorial

MapReduce is a programming framework that allows us to perform distributed and parallel processing on large data sets in a distributed environment.

BIG DATA TECHNOLOGIES

- MapReduce consists of two distinct tasks — Map and Reduce.
- As the name MapReduce suggests, reducer phase takes place after the mapper phase has been completed.
- So, the first is the map job, where a block of data is read and processed to produce key-value pairs as intermediate outputs.
- The output of a Mapper or map job (key-value pairs) is input to the Reducer.
- The reducer receives the key-value pair from multiple map jobs.
- Then, the reducer aggregates those intermediate data tuples (intermediate key-value pair) into a smaller set of tuples or key-value pairs which is the final output.

A Word Count Example of MapReduce

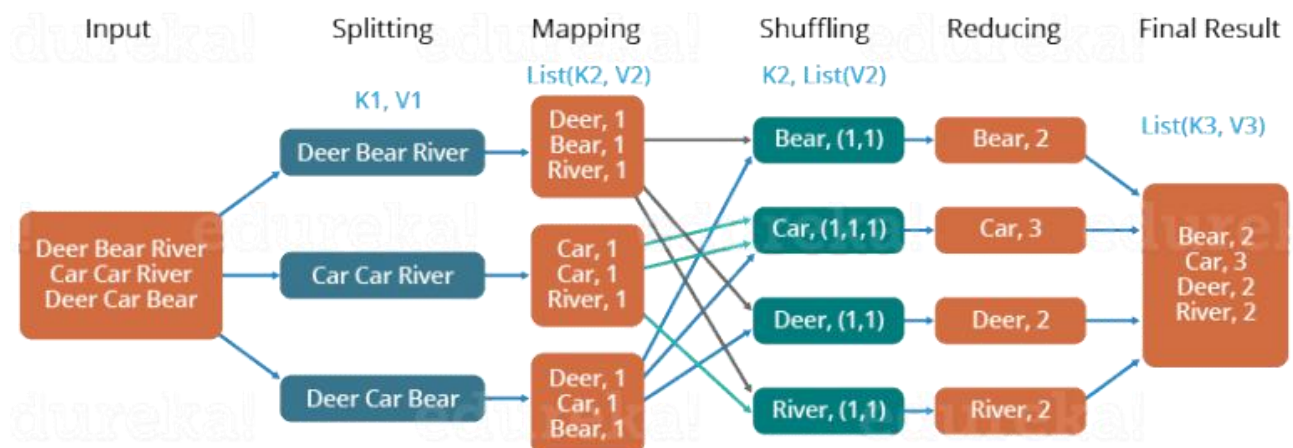
Let us understand, how a MapReduce works by taking an example where I have a text file called example.txt whose contents are as follows:

Dear, Bear, River, Car, Car, River, Deer, Car and Bear

Now, suppose, we have to perform a word count on the sample.txt using MapReduce. So, we will be finding unique words and the number of occurrences of those unique words.

The Overall MapReduce Word Count Process

edureka!



BIG DATA TECHNOLOGIES

MapReduce Example - MapReduce Tutorial

- First, we divide the input into three splits as shown in the figure. This will distribute the work among all the map nodes.
- Then, we tokenize the words in each of the mappers and give a hardcoded value (1) to each of the tokens or words. The rationale behind giving a hardcoded value equal to 1 is that every word, in itself, will occur once.
- Now, a list of key-value pair will be created where the key is nothing but the individual words and value is one. So, for the first line (Dear Bear River) we have 3 key-value pairs — Dear, 1; Bear, 1; River, 1. The mapping process remains the same on all the nodes.
- After the mapper phase, a partition process takes place where sorting and shuffling happen so that all the tuples with the same key are sent to the corresponding reducer.
- So, after the sorting and shuffling phase, each reducer will have a unique key and a list of values corresponding to that very key. For example, Bear, [1,1]; Car, [1,1,1]..., etc.
- Now, each Reducer counts the values which are present in that list of values. As shown in the figure, reducer gets a list of values which is [1,1] for the key Bear. Then, it counts the number of ones in the very list and gives the final output as — Bear, 2.
- Finally, all the output key/value pairs are then collected and written in the output file.

Advantages of MapReduce

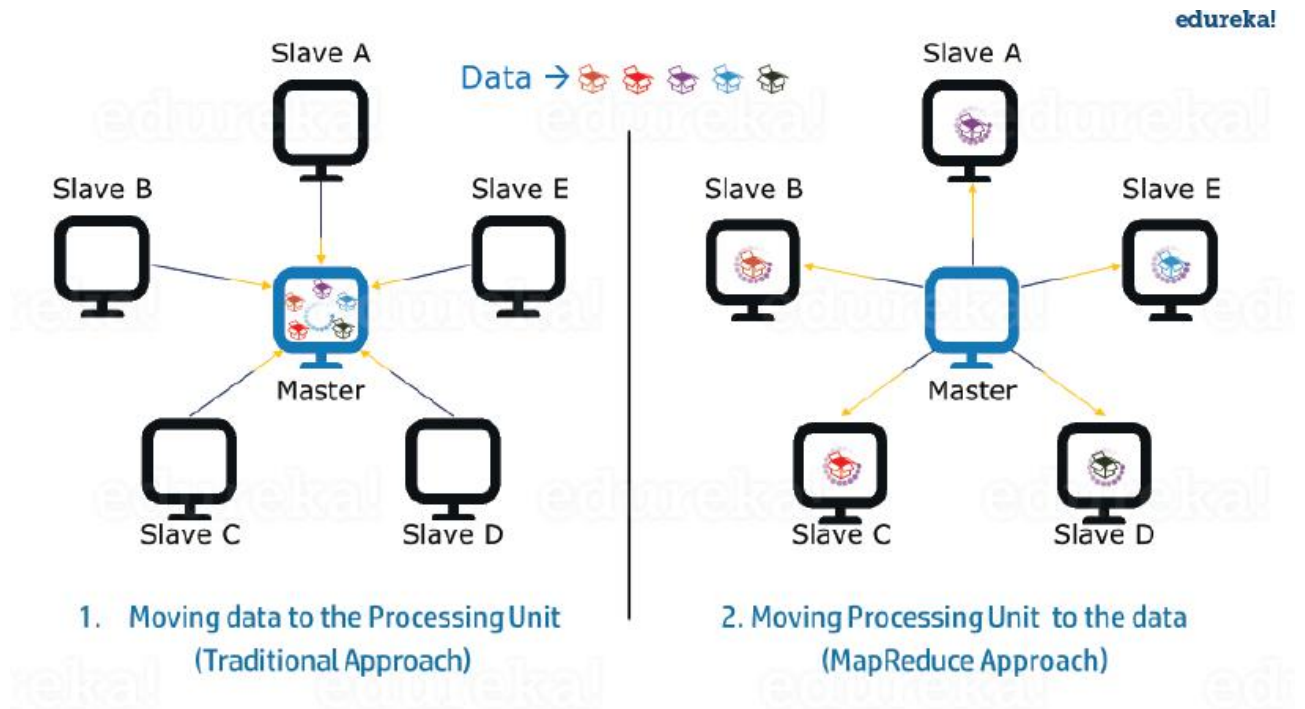
The two biggest advantages of MapReduce are:

1. Parallel Processing:

In MapReduce, we are dividing the job among multiple nodes and each node works with a part of the job simultaneously. So, MapReduce is based on Divide and Conquer paradigm which helps us to

BIG DATA TECHNOLOGIES

process the data using different machines. As the data is processed by multiple machines instead of a single machine in parallel, the time taken to process the data gets reduced by a tremendous amount as shown in the figure below (2).



Traditional Way Vs. MapReduce Way - MapReduce Tutorial

2. Data Locality:

Instead of moving data to the processing unit, we are moving the processing unit to the data in the MapReduce Framework. In the traditional system, we used to bring data to the processing unit and process it. But, as the data grew and became very huge, bringing this huge amount of data to the processing unit posed the following issues:

- Moving huge data to processing is costly and deteriorates the network performance.
- Processing takes time as the data is processed by a single unit which becomes the bottleneck.
- Master node can get over-burdened and may fail.

BIG DATA TECHNOLOGIES

Now, MapReduce allows us to overcome the above issues by bringing the processing unit to the data. So, as you can see in the above image that the data is distributed among multiple nodes where each node processes the part of the data residing on it. This allows us to have the following advantages:

- It is very cost effective to move the processing unit to the data.
- The processing time is reduced as all the nodes are working with their part of the data in parallel.
- Every node gets a part of the data to process and therefore, there is no chance of a node getting overburdened.

Techniques to optimize Map Reduce jobs

1. Reduce Data Shuffling

Data shuffling is the process of moving data between nodes in the cluster. This can be a time-consuming process, especially when dealing with large datasets. To reduce data shuffling, you should try to minimize the amount of data that needs to be shuffled.

One way to do this is by using partitioning. Partitioning is the process of dividing the data into smaller, more manageable chunks. By partitioning the data, you can ensure that each node in the cluster only processes a small portion of the data. This can help to reduce data shuffling and improve the performance of your MapReduce job.

2. Optimize Map and Reduce Functions

The map and reduce functions are the heart of your MapReduce job. To optimize your job, you should ensure that these functions are as efficient as possible.

One way to optimize the map function is by using combiners. A combiner is a mini-reduce function that runs on each node in the cluster after the map function has been run. The combiner function can help to reduce the amount of data that needs to be shuffled by combining the intermediate key-value pairs at each node.

To optimize the reduce function, you should ensure that it is as simple as possible. The reduce function should only perform the necessary calculations and should not perform any unnecessary operations.

3. Use Compressed Data

When dealing with large datasets, it's important to minimize the amount of disk space that is required. One way to do this is by using compressed data. Compressed data takes up less disk space, which can help to improve the performance of your MapReduce job.

BIG DATA TECHNOLOGIES

4. Use the Right File Format

The file format that you use can also have an impact on the performance of your MapReduce job. Some file formats, such as text files, can be inefficient when dealing with large datasets.

One way to optimize your job is by using a file format that is optimized for MapReduce. For example, the Sequence File format is a binary file format that is optimized for MapReduce. It can help to improve the performance of your job by reducing the amount of disk I/O that is required.

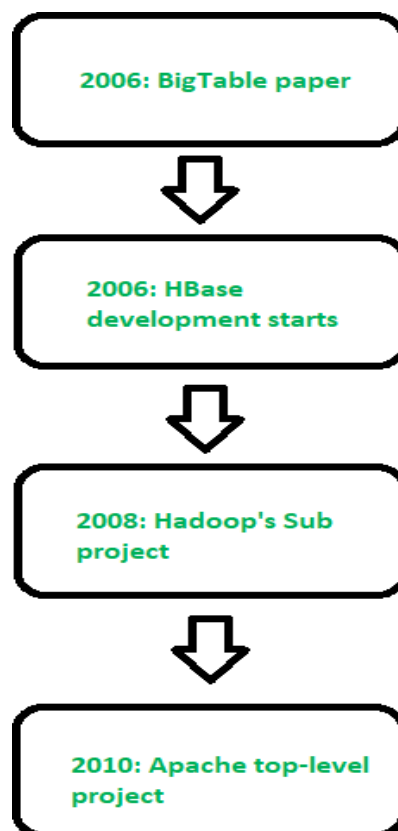
5. Use Commodity Hardware

When setting up your MapReduce cluster, it's important to use commodity hardware. Commodity hardware is inexpensive, widely available hardware that is designed for general-purpose computing.

Using commodity hardware can help to reduce the cost of your MapReduce cluster while still providing the processing power that you need. Additionally, commodity hardware is designed to be easily replaceable, which can help to reduce downtime in case of hardware failure.

Role of HBase in big data Processing

HBase is a data model that is similar to Google's big table. It is an open source, distributed database developed by *Apache* software foundation written in Java. HBase is an essential part of our Hadoop ecosystem. HBase runs on top of HDFS (Hadoop Distributed File System). It can store massive amounts of data from terabytes to petabytes. It is column oriented and horizontally scalable.



Developing Simple Map reduce Applications

BIG DATA TECHNOLOGIES

Entertainment: To discover the most popular movies, based on what you like and what you watched in this case Hadoop MapReduce help you out. It mainly focuses on their logs and clicks.

E-commerce: Numerous E-commerce suppliers, like Amazon, Walmart, and eBay, utilize the MapReduce programming model to distinguish most loved items dependent on clients' inclinations or purchasing behavior.

It incorporates making item proposal Mechanisms for E-commerce inventories, examining website records, buy history, user interaction logs, etc.

Data Warehouse: We can utilize MapReduce to analyze large data volumes in data warehouses while implementing specific business logic for data insights.

Fraud Detection: Hadoop and MapReduce are utilized in monetary enterprises, including organizations like banks, insurance providers, installment areas for misrepresentation recognition, pattern distinguishing proof, or business metrics through transaction analysis.

Points to consider while Designing Map Reduce

1. Input-Map-Reduce-Output
2. Input-Map-Output
3. Input-Multiple Maps-Reduce-Output
4. Input-Map-Combiner-Reduce-Output

Following are some real-world scenarios, to help you understand when to use which design pattern.

Input-Map-Reduce-Output



If we want to perform an aggregation operation, this pattern is used:

Scenario	Counting the total salary of employees based on gender.
Map (Key, Value)	Key : Gender Value : Their Salary
Reduce	Group by Key (Gender), then take the sum of the value (salary) for each group.

BIG DATA TECHNOLOGIES

EMPLOYEE NAME	GENDER	SALARY IN K
Krishna	Male	80
Namrata	Female	75
Ritesh	Male	100
Ram	Male	40
Vijay	Male	250
Jitu	Male	55
Ria	Female	90
Poonam	Female	300
Radha	Female	40

To count the total salary by gender, we need to make the key Gender and the value Salary. The output for the Map function is:

```
(Male, 80), (Female, 75), (Male, 100), (Male, 40), (Male, 250),  
(Male, 55), (Female, 90), (Female, 300), (Female, 40)
```

Intermediate splitting gives the input for the Reduce function:

```
(Male <80, 100, 40, 250, 55>), (Female<75, 90, 300, 40>)
```

And the Reduce function output is:

```
(Male, 525), (Female, 505)
```

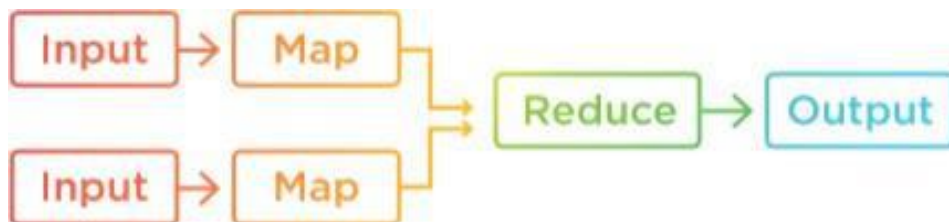
Input-Map-Output



The Reduce function is mostly used for aggregation and calculation. However, if we only want to change the format of the data, then the Input-Map-Output pattern is used:

BIG DATA TECHNOLOGIES

Scenario	The data source has inconsistent entries for Gender, such as "Female", "F", "f", or O. We want to make the column consistent, i.e., for male, "M" should be used, and for female, "F" should be used.
Map (Key, Value)	Key : Employee ID Value : Gender -> If Gender is Female/female/ F/ f/ O, then return F; else if Gender is Male/male/M/m/1, then return M.



In the Input-Multiple Maps-Reduce-Output design pattern, our input is taken from two files, each of which has a different schema. (Note that if two or more files have the same schema, then there is no need for two mappers. We can simply write the same logic in one mapper class and provide multiple input files.)

Scenario 1	<p>We have to find the total salary by gender. But we have 2 files with different schemas.</p> <p>Input File 1 Gender is given as a prefix to the name. <i>E.g. Ms. Shital Katkar</i> <i>Mr. Krishna Katkar</i></p> <p>Input File 2 There is a separate, dedicated column for gender. However, the format is inconsistent. <i>E.g. Female/Male, O/1, F/M</i></p>
Map (Key, Value)	<p>Map 1 (for input 1) We need to write program to split each prefix from its name and determine the gender according to the prefix. Then we prepare the key-value pair (Gender, Salary).</p> <p>Map 2 (for input 2) Here, the program will be straightforward. Resolve the mixed formats (as seen earlier) and make a key-value pair (Gender, Salary).</p>
Reduce	Group by Gender and take the total salary for each group.

BIG DATA TECHNOLOGIES

Scenario 2	<p>There are two large files, 'Books and 'Rating'.</p> <table border="1"> <thead> <tr> <th>Book ID</th> <th>Book Name</th> </tr> </thead> <tbody> <tr> <td>101</td> <td>Classical Mythology</td> </tr> <tr> <td>102</td> <td>The Kitchen God's Wife</td> </tr> <tr> <td>103</td> <td>Haveli</td> </tr> </tbody> </table> <table border="1"> <thead> <tr> <th>Book ID</th> <th>Rating</th> <th>User ID</th> </tr> </thead> <tbody> <tr> <td>101</td> <td>8</td> <td>X134</td> </tr> <tr> <td>102</td> <td>7</td> <td>D454</td> </tr> <tr> <td>101</td> <td>7</td> <td>C455</td> </tr> <tr> <td>103</td> <td>6</td> <td>B455</td> </tr> </tbody> </table> <p>We want to display each book and its average rating.</p>	Book ID	Book Name	101	Classical Mythology	102	The Kitchen God's Wife	103	Haveli	Book ID	Rating	User ID	101	8	X134	102	7	D454	101	7	C455	103	6	B455
	Book ID	Book Name																						
101	Classical Mythology																							
102	The Kitchen God's Wife																							
103	Haveli																							
Book ID	Rating	User ID																						
101	8	X134																						
102	7	D454																						
101	7	C455																						
103	6	B455																						
Map (Key, Value)	<p>In this case, we have to join two tables, Books and Rating, on the Book ID column, so we open the two files in two different Map functions.</p> <p>Map 1 (for the Books table) Since we want to join on Book ID, it should be sent as the key in both maps. Key-value pair: (Book ID, Book Name)</p> <p>Map 2 (for the Rating table) Key-value pair: (Book ID, Rating)</p>																							
Splitting Output	<p>(101< Classical Mythology, 8, 7, 9, 6,.....>)</p> <p>(102<The Kitchen God's Wife,7,8,5,6,.....>)</p> <p>(103<Haveli,6,7,5,6,.....>)</p>																							
Reduce	<p>Group By Book ID, take the average of Rating, and return (Book Name, Average Rating).</p> <p>(Classical Mythology, 7.5)</p> <p>(The Kitchen God's Wife, 6.5)</p> <p>(Haveli, 6)</p>																							

Input-Map-Combiner-Reduce-Output



Apache Spark is highly effective for big and small data processing tasks not because it best reinvents the wheel, but because it best amplifies the existing tools needed to perform effective analysis. Coupled with its highly scalable nature on commodity grade hardware, and incredible performance capabilities compared to other well known Big Data processing engines, Spark may finally let software finish eating the world.

A Combiner, also known as a semi-reducer, is an optional class that operates by accepting the inputs from the Map class and then passing the output key-value pairs to the Reducer class. The purpose of the Combiner function is to reduce the workload of Reducer.

In a MapReduce program, 20% of the work is done in the Map stage, which is also known as the data preparation stage. This stage does work in parallel.

80% of the work is done in the Reduce stage, which is known as the calculation stage. This work is not done in parallel, so it is slower than the Map phase. To reduce computation time, some work of the Reduce phase can be done in a Combiner phase.

BIG DATA TECHNOLOGIES

UNIT-IV

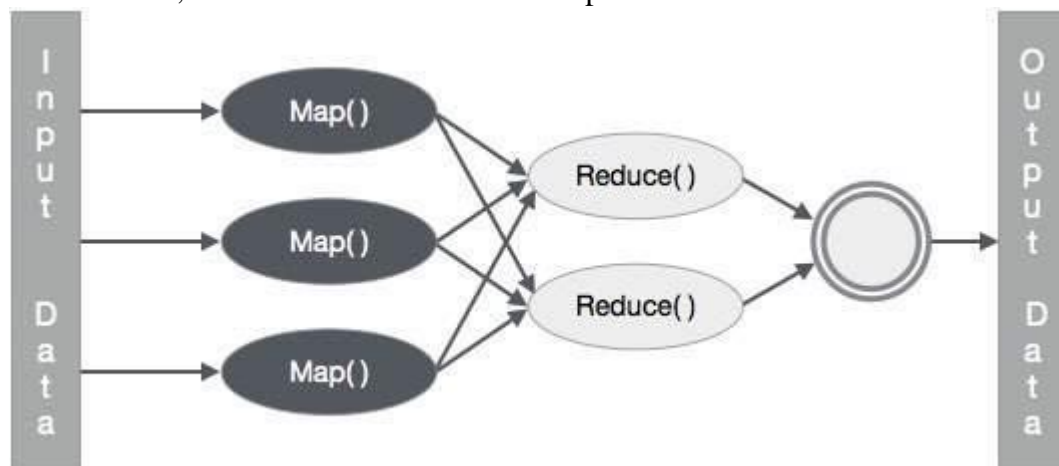
Customizing Map Reduce Execution and Implementing

MapReduce is a framework that is used for writing applications to process huge volumes of data on large clusters of commodity hardware in a reliable manner. This chapter takes you through the operation of MapReduce in Hadoop framework using Java.

MapReduce Algorithm

Generally MapReduce paradigm is based on sending map-reduce programs to computers where the actual data resides.

- During a MapReduce job, Hadoop sends Map and Reduce tasks to appropriate servers in the cluster.
- The framework manages all the details of data-passing like issuing tasks, verifying task completion, and copying data around the cluster between the nodes.
- Most of the computing takes place on the nodes with data on local disks that reduces the network traffic.
- After completing a given task, the cluster collects and reduces the data to form an appropriate result, and sends it back to the Hadoop server.



Inputs and Outputs (Java Perspective)

The MapReduce framework operates on key-value pairs, that is, the framework views the input to the job as a set of key-value pairs and produces a set of key-value pair as the output of the job, conceivably of different types.

The key and value classes have to be serializable by the framework and hence, it is required to implement the Writable interface. Additionally, the key classes have to implement the WritableComparable interface to facilitate sorting by the framework.

Both the input and output format of a MapReduce job are in the form of key-value pairs –

(Input) <k1, v1> -> map -> <k2, v2>-> reduce -> <k3, v3> (Output).

Input

Output

BIG DATA TECHNOLOGIES

Map <k1, v1> list (<k2, v2>)

Reduce <k2, list(v2)> list (<k3, v3>)

MapReduce Implementation

The following table shows the data regarding the electrical consumption of an organization. The table includes the monthly electrical consumption and the annual average for five consecutive years.

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Avg
1979	23	23	2	43	24	25	26	26	26	26	25	26	25
1980	26	27	28	28	28	30	31	31	31	30	30	30	29
1981	31	32	32	32	33	34	35	36	36	34	34	34	34
1984	39	38	39	39	39	41	42	43	40	39	38	38	40
1985	38	39	39	39	39	41	41	41	00	40	39	39	45

We need to write applications to process the input data in the given table to find the year of maximum usage, the year of minimum usage, and so on. This task is easy for programmers with finite amount of records, as they will simply write the logic to produce the required output, and pass the data to the written application.

Let us now raise the scale of the input data. Assume we have to analyze the electrical consumption of all the large-scale industries of a particular state. When we write applications to process such bulk data,

- They will take a lot of time to execute.
- There will be heavy network traffic when we move data from the source to the network server.

To solve these problems, we have the MapReduce framework.

Input Data

The above data is saved as **sample.txt** and given as input. The input file looks as shown below.

1979	23	23	2	43	24	25	26	26	26	26	25	26	25
1980	26	27	28	28	28	30	31	31	31	30	30	30	29
1981	31	32	32	32	33	34	35	36	36	34	34	34	34
1984	39	38	39	39	39	41	42	43	40	39	38	38	40
1985	38	39	39	39	39	41	41	41	00	40	39	39	45

Example Program

The following program for the sample data uses MapReduce framework.

BIG DATA TECHNOLOGIES

```
package hadoop;

import java.util.*;
import java.io.IOException;
import java.io.IOException;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;

public class ProcessUnits
{
    //Mapper class
    public static class E_EMapper extends MapReduceBase implements
    Mapper<LongWritable, /*Input key Type */
    Text,          /*Input value Type*/
    Text,          /*Output key Type*/
    IntWritable>    /*Output value Type*/
    {
        //Map function
        public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output,
Reporter reporter) throws IOException
        {
            String line = value.toString();
            String lasttoken = null;
            StringTokenizer s = new StringTokenizer(line,"t");
            String year = s.nextToken();

            while(s.hasMoreTokens()){
                lasttoken=s.nextToken();
            }

            int avgprice = Integer.parseInt(lasttoken);
            output.collect(new Text(year), new IntWritable(avgprice));
        }
    }

    //Reducer class

    public static class E_EReduce extends MapReduceBase implements
    Reducer< Text, IntWritable, Text, IntWritable >
    {
        //Reduce function
        public void reduce(Text key, Iterator <IntWritable> values, OutputCollector<>Text, IntWritable>
output, Reporter reporter) throws IOException
        {
            int maxavg=30;
            int val=Integer.MIN_VALUE;
            while (values.hasNext())
            {
```

BIG DATA TECHNOLOGIES

```
        if((val=values.next().get())>maxavg)
        {
            output.collect(key, new IntWritable(val));
        }
    }
}

//Main function

public static void main(String args[])throws Exception
{
    JobConf conf = new JobConf(Eleunits.class);

    conf.setJobName("max_electricityunits");

    conf.setOutputKeyClass(Text.class);
    conf.setOutputValueClass(IntWritable.class);

    conf.setMapperClass(E_EMapper.class);
    conf.setCombinerClass(E_EReduce.class);
    conf.setReducerClass(E_EReduce.class);

    conf.setInputFormat(TextInputFormat.class);
    conf.setOutputFormat(TextOutputFormat.class);

    FileInputFormat.setInputPaths(conf, new Path(args[0]));
    FileOutputFormat.setOutputPath(conf, new Path(args[1]));

    JobClient.runJob(conf);
}
}
```

Save the above program into **ProcessUnits.java**. The compilation and execution of the program is given below.

Compilation and Execution of ProcessUnits Program

Let us assume we are in the home directory of Hadoop user (e.g. /home/hadoop).

Follow the steps given below to compile and execute the above program.

Step 1 – Use the following command to create a directory to store the compiled java classes.

```
$ mkdir units
```

Step 2 – Download Hadoop-core-1.2.1.jar, which is used to compile and execute the MapReduce program. Download the jar from mvnrepository.com. Let us assume the download folder is /home/hadoop/.

Step 3 – The following commands are used to compile the **ProcessUnits.java** program and to create a jar for the program.

BIG DATA TECHNOLOGIES

```
$ javac -classpath hadoop-core-1.2.1.jar -d units ProcessUnits.java  
$ jar -cvf units.jar -C units/ .
```

Step 4 – The following command is used to create an input directory in HDFS.

```
$HADOOP_HOME/bin/hadoop fs -mkdir input_dir
```

Step 5 – The following command is used to copy the input file named **sample.txt** in the input directory of HDFS.

```
$HADOOP_HOME/bin/hadoop fs -put /home/hadoop/sample.txt input_dir
```

Step 6 – The following command is used to verify the files in the input directory

```
$HADOOP_HOME/bin/hadoop fs -ls input_dir/
```

Step 7 – The following command is used to run the Eleunit_max application by taking input files from the input directory.

```
$HADOOP_HOME/bin/hadoop jar units.jar hadoop.ProcessUnits input_dir output_dir
```

Wait for a while till the file gets executed. After execution, the output contains a number of input splits, Map tasks, Reducer tasks, etc.

```
INFO mapreduce.Job: Job job_1414748220717_0002  
completed successfully  
14/10/31 06:02:52  
INFO mapreduce.Job: Counters: 49
```

File System Counters

```
FILE: Number of bytes read=61  
FILE: Number of bytes written=279400  
FILE: Number of read operations=0  
FILE: Number of large read operations=0  
FILE: Number of write operations=0
```

```
HDFS: Number of bytes read=546  
HDFS: Number of bytes written=40  
HDFS: Number of read operations=9  
HDFS: Number of large read operations=0  
HDFS: Number of write operations=2 Job Counters
```

```
Launched map tasks=2  
Launched reduce tasks=1  
Data-local map tasks=2
```

```
Total time spent by all maps in occupied slots (ms)=146137  
Total time spent by all reduces in occupied slots (ms)=441  
Total time spent by all map tasks (ms)=14613  
Total time spent by all reduce tasks (ms)=44120
```

BIG DATA TECHNOLOGIES

Total vcore-seconds taken by all map tasks=146137
Total vcore-seconds taken by all reduce tasks=44120

Total megabyte-seconds taken by all map tasks=149644288
Total megabyte-seconds taken by all reduce tasks=45178880

Map-Reduce Framework

Map input records=5

Map output records=5
Map output bytes=45
Map output materialized bytes=67

Input split bytes=208
Combine input records=5
Combine output records=5

Reduce input groups=5
Reduce shuffle bytes=6
Reduce input records=5
Reduce output records=5

Spilled Records=10
Shuffled Maps =2
Failed Shuffles=0
Merged Map outputs=2

GC time elapsed (ms)=948
CPU time spent (ms)=5160

Physical memory (bytes) snapshot=47749120
Virtual memory (bytes) snapshot=2899349504

Total committed heap usage (bytes)=277684224

File Output Format Counters

Bytes Written=40

Step 8 – The following command is used to verify the resultant files in the output folder.

```
$HADOOP_HOME/bin/hadoop fs -ls output_dir/
```

Step 9 – The following command is used to see the output in **Part-00000** file. This file is generated by HDFS.

```
$HADOOP_HOME/bin/hadoop fs -cat output_dir/part-00000
```

Following is the output generated by the MapReduce program –

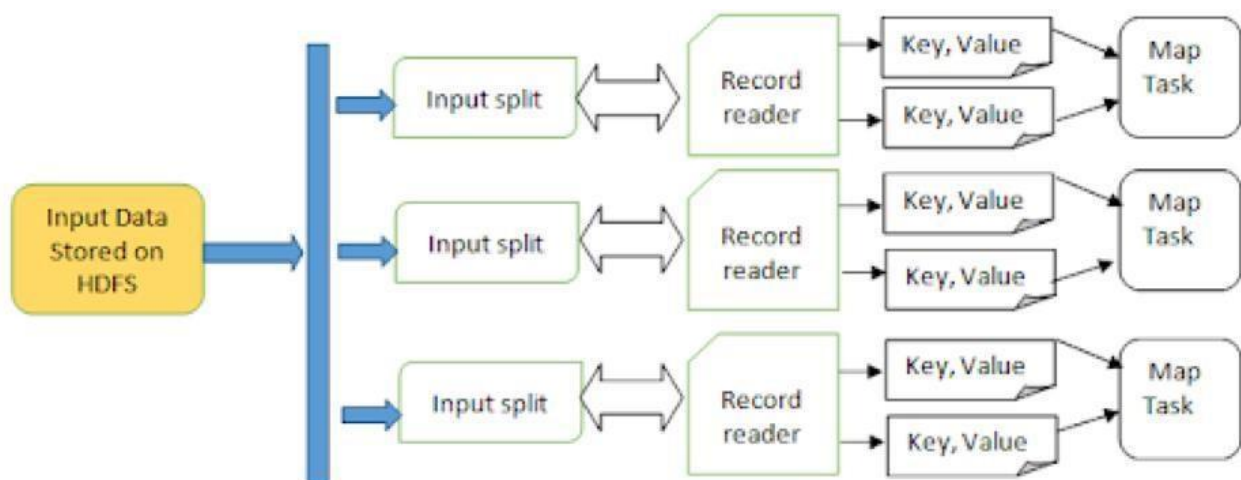
BIG DATA TECHNOLOGIES

1981	34
1984	40
1985	45

Step 10 – The following command is used to copy the output folder from HDFS to the local file system.

Map Reduce Execution with input format

In this tutorial, we are going to cover the other component of the MapReduce process i.e. Hadoop MapReduce Input Format. We will discuss What is Input Format in Hadoop, What functionalities are offered by MapReduce Input Format. We will also cover the types of Input Format in MapReduce, and how to get the data from the mapper using Input Format.



What is Hadoop MapReduce Input Format?

Hadoop Input Format illustrates the input-specification for the execution of the Map-Reduce job.

Input Format illustrates how to divide and read input files. In MapReduce job execution, Input Format is the initial step. It is also accountable for creating the input splits and dividing them into records.

Input files store the data for MapReduce job. Input files reside in HDFS. Though these files format is random, we can also use line-based log files and binary format. Therefore, In MapReduce, Input Format class is one of the fundamental classes which offers below functionality:

- Input Format selects the files or other objects for input.
- It also defines the Data splits. It defines both the size of individual Map tasks and its possible execution server.

BIG DATA TECHNOLOGIES

Procedures to get the data from mapper are get splits () and create Record Reader () which are as below:

1. public abstract class Input Format<K, V>
2. {
3. public abstract List< Input Split > get Splits(Job Context context)
4. throws IO Exception, InterruptedException;
5. public abstract Record Reader<K, V>
6. create Record Reader (Input Split split,
7. Task Attempt Context context) throws IO Exception,
8. InterruptedException;
9. }

Types of Input Format in MapReduce

There are different types of MapReduce Input Format in Hadoop which are used for different purposes. Let us discuss the Hadoop input Format types below:

File Input Format

FileInputFormat is the source class for all file-based Input Formats. FileInputFormat also specifies the input directory which has data files location. When we initiate a MapReduce job execution, FileInputFormat offers a path containing files to read. This InputFormat will read all files. Then it splits these files into one or more Input Splits.

TextInputFormat

It is the default input Format. This input Format considers each line of each input file as a separate record. It does no parsing. TextInputFormat is useful for unformatted data or line-based records like log files. Therefore,

- **Key** – It is the byte offset of the beginning of the line within the file (not whole file one split). Hence it will be unique if combined with the file name.
- **Value** – It is the subject of the line. It excludes line terminators.

BIG DATA TECHNOLOGIES

Key Value Text Input Format

It is similar to TextInputFormat. This input Format also considers each line of input as a separate record. While the difference is that TextInputFormat considers the entire line as the value, but the KeyValueTextInputFormat divides the line itself into key and value by a tab character ('/t'). Hence,

- **Key** – The whole thing up to the tab character.
- **Value** – Value is the remaining part of the line after tab character.

Sequence File Input Format

It is an Input Format that reads sequence files. Sequence files are binary files. These files also store sequences of binary key-value pairs. These are block-compressed and offer direct serialization and deserialization of several random data. Hence,

Key & Value both are user-defined.

Sequence File As Text Input Format

It is the variant of SequenceFileInputFormat. This format transforms the sequence file key values to Text objects. Hence, it performs conversion by calling 'to string ()' on the keys and values. Therefore, Sequence File As Text Input Format makes sequence files suitable input for streaming.

Sequence File As Binary Input Format

By utilizing SequenceFileInputFormat we can obtain the sequence file's keys and values as an opaque binary object.

Nline Input Format

It is the other form of Text Input Format where the keys are byte offset of the line. And values are contents of the line. Hence, each mapper receives a variable number of lines of input with Text Input Format and Key Value Text Input Format. The number relies on the size of the split. Moreover, it relies on the length of the lines. Therefore, if we want our mapper to receive a fixed number of lines of input, then we use Nline Input Format.

N- It is the number of lines of input that each mapper accepts.

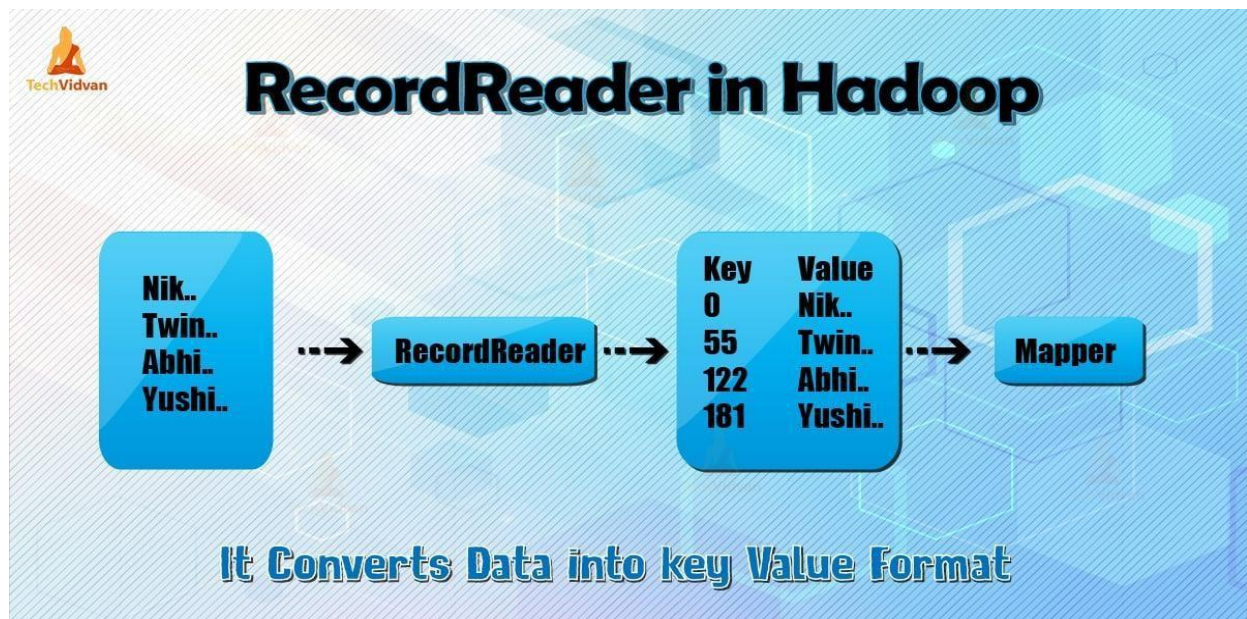
By default (N=1), each mapper accepts exactly one line of input.

Suppose N=2, then each split contains two lines. So, one mapper accepts the first two Key-Value pairs. Another mapper accepts the second two key-value pairs.

BIG DATA TECHNOLOGIES

Reading Data with Custom record reader

In our previous blog, we have studied **Hadoop Counters** in detail. Now in this tutorial, we are going to discuss the Record Reader in Hadoop. Here we will cover the introduction to Hadoop Record Reader, working of Record Reader. We will also discuss the types of Record Reader in MapReduce, the size of the single Record in Hadoop MapReduce in this MapReduce



A Record Reader converts the byte-oriented view of the input to a record-oriented view for the **Mapper** and **Reducer** tasks for processing.

To understand Hadoop Record Reader, we need to understand MapReduce Dataflow. Let us learn how the data flow:

MapReduce is a simple model of data processing. Inputs and outputs for the map and reduce functions are **key-value pairs**. Following is the general form of the map and reduce functions:

- **Map:** (K1, V1) → list (K2, V2)
- **Reduce:** (K2, list (V2)) → list (K3, V3)

Now before processing starts, it needs to know on which data to process. So, **Input Format** class helps to achieve this. This class selects the file from **HDFS** that is the input to the map function. It is also responsible for creating the input splits.

Also, divide them into records. It divides the data into the number of splits (typically 64/128mb) in HDFS. This is known as Input Split. Input Split is the logical representation of data. In a MapReduce job, execution number of map tasks is equal to the number of Input Splits.

By calling '**get Split ()**' the client calculates the splits for the job. Then it sent to the application master. It uses their storage locations to schedule map tasks that will process them on the cluster.

BIG DATA TECHNOLOGIES

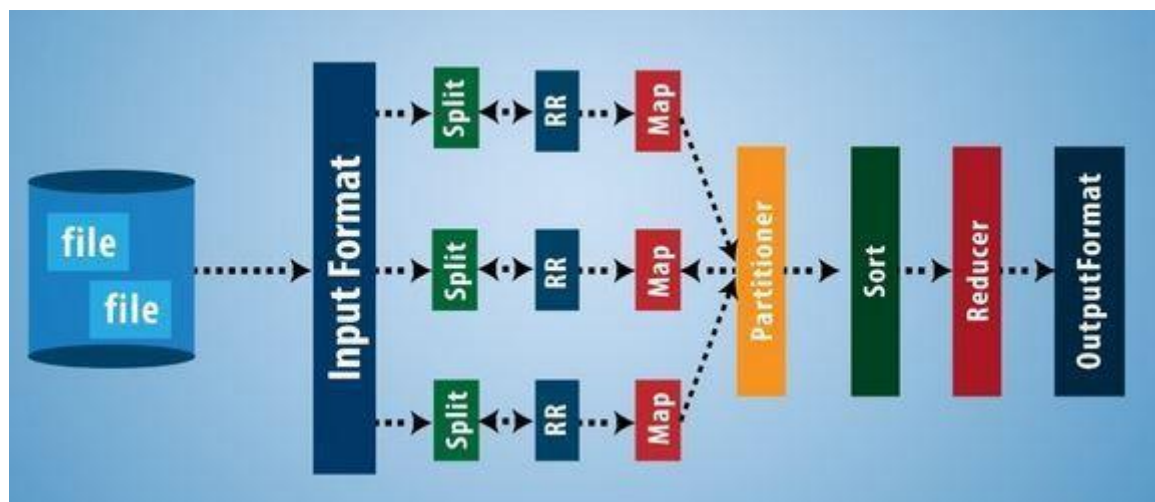
After that map task passes the split to the **create Record Reader()** method. From that, it obtains Record Reader for the split. Record Reader generates record (key-value pair). Then it passes to the map function.

Hadoop Record Reader in MapReduce job execution uses the data within the boundaries that are being created by the input split. And it then creates Key-value pairs for the mapper. The “start” is the byte position in the file.

At the Start, Hadoop Record Reader starts generating key/value pairs. The “end” is where Record Reader stops reading records. In Record Reader, the data is loaded from its source.

Then the data are converted into key-value pairs suitable for reading by the Mapper. It communicates with the input split till the file reading is not completed.

Output Data with Output formats



Output Format check the output specification for execution of the Map-Reduce job. It describes how Record Writer implementation is used to write output to output files.

Record Writer in Hadoop MapReduce

Reducer takes as input a set of an intermediate key-value pair produced by the mapper and runs a reducer function on them to generate output that is again zero or more key-value pairs. Record Writer writes these output key-value pairs from the Reducer phase to output files.

Hadoop Output Format

Hadoop Record Writer takes output data from Reducer and writes this data to output files. The way these output key-value pairs are written in output files by Record Writer is determined by the Output

BIG DATA TECHNOLOGIES

Format. The Output Format and input Format functions are alike. Output Format instances provided by Hadoop are used to write to files on the HDFS or local disk. Output Format describes the output-specification for a Map-Reduce job. On the basis of output specification;

1. MapReduce job checks that the output directory does not already exist.
2. Output Format provides the Record Writer implementation to be used to write the output files of the job. Output files are stored in a Filesystem.

File Output Format. Set Output Path() method is used to set the output directory. Every Reducer writes a separate file in a common output directory.

Types of Output Format in MapReduce

1. Text Output Format

Text Output Format is the default Hadoop reducer Output Format in MapReduce, which writes (key, value) pairs on individual lines of text files and its keys and values can be of any type since Text Output Format turns them to string by calling to String () on them. Each key-value pair is separated by a tab character, which can be changed using map Reduce. Output .text output format. separator property. Key Value Text Output Format is used for reading these output text files since it breaks lines into key-value pairs based on a configurable separator.

2. Sequence FileOutputFormat

Sequence FileOutputFormat is an Output Format which writes sequences files for its output and it is intermediate format use between MapReduce jobs, which rapidly serialize arbitrary data types to the file; and the corresponding SequenceFileInputFormat will deserialize the file into the same types and presents the data to the next mapper in the same manner as it was emitted by the previous reducer, since these are compact and readily compressible. Compression is controlled by the static methods on Sequence FileOutputFormat.

BIG DATA TECHNOLOGIES

3. Sequence File As Binary Output Format

Sequence File As Binary Output Format is another variant of Sequence File Input Format. It also writes keys and values to sequence file in binary format.

4. MapFileOutputFormat

MapFileOutputFormat is another form of FileOutputFormat. It also writes output as map files. The framework adds a key in a Map File in order. So we need to ensure that reducer emits keys in sorted order.

5. Multiple Outputs

Multiple Outputs allows writing data to files whose names are derived from the output keys and values, or in fact from an arbitrary string.

6. LazyOutputFormat

Sometimes FileOutputFormat will create output files, even if they are empty. LazyOutputFormat is a wrapper Output Format which ensures that the output file will be created only when the record is emitted for a given partition.

7. DBOutputFormat

DBOutputFormat in Hadoop is an Output Format for writing to relational databases and HBase. It sends the reduce output to a SQL table. It accepts key-value pairs, where the key has a type extending DB writable. Returned Record Writer writes only the key to the database with a batch SQL query.

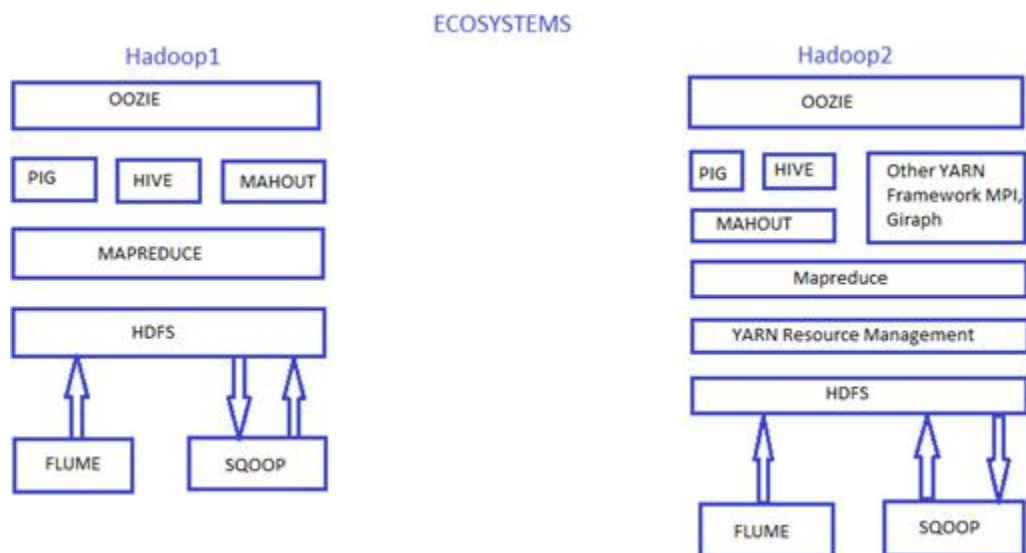
Map Reduce Execution in terms of Yarn

- Hadoop 1 has two components first one is HDFS (Hadoop Distributed File System), and the second is Map Reduce. Whereas Hadoop 2 it also has two components, HDFS and YARN/MRv2 (we usually call Yarn as MapReduce version 2).
- In Map Reduce, when Map-reduce stops working, then automatically, all his slave nodes will stop working. This is the one scenario where job execution can interrupt, called a single point

BIG DATA TECHNOLOGIES

of failure. YARN overcomes this issue because of its architecture; Yarn has the concept of an Active name node and a standby name node. When an active node stop working for some time, a passive node starts working as an active node and continues the execution.

- Map reduce has a single master and multiple slave architecture; if master-slave goes down, the entire slave will stop working. This is the single point of failure in HADOOP1, whereas HADOOP2, based on YARN architecture, has the concept of multiple masters and slaves; if one master goes down, another master will resume its process and continue the execution.
- As shown in the diagram below, the difference in both Ecosystems is HADOOP1 and HADOOP2. Component-wise, Yarn Resource Management interacts with Map-reduce and HDFS.



- So basically, Yarn is responsible for resource management, which means which job will be executed and which system gets decided by Yarn. In contrast, map-reduce is a programming framework responsible for executing a particular job, so basically, Map-reduce has two components, a mapper and a reducer, for the execution of a program.
- In Map, each data node runs individually, whereas, in Yarn, each node runs by a node manager.

BIG DATA TECHNOLOGIES

- Map reduce uses a Job tracker to create and assign a task to a task tracker. Due to data, the resource management is not impressive, resulting in some of the data nodes being idle and of no use, whereas YARN has a Resource Manager for each cluster, and each data node runs a Node Manager. For each job, one slave node will act as the Application Master, monitoring resources/tasks.

MapReduce vs Yarn Comparison Table

Below is the comparison between MapReduce vs Yarn:

Basis for Comparison	Yarn	MapReduce
Meaning	Yarn Stands for Yet Another Resource Negotiator.	Map Reduce is self-defined.
Version	Introduce in Hadoop 2.0.	Introduce in Hadoop 1.0.
Responsibility	Now Yarn is responsible for the Resource management part.	Earlier, MapReduce was responsible for Resource Management as well as data processing.
Execution Model	The yarn execution model is more generic as compared to MapReduce.	Less Generic as compared to Yarn.
Application Execution	Yarn can also execute those applications that don't follow the MapReduce model.	MapReduce can execute its own model-based application.

BIG DATA TECHNOLOGIES

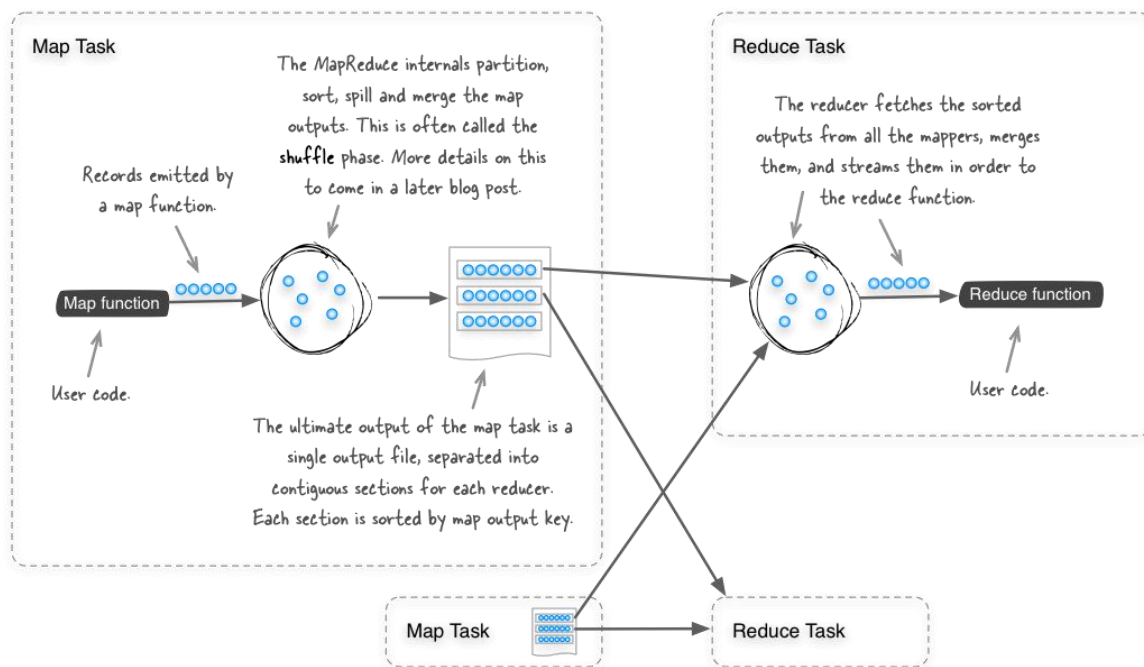
Architecture	Yarn is introduced in MR2 on top of the job and task tracks. In the place of job tracker and task tracker Application, the master comes into the picture.	In the earlier version of MR1, YARN was not there. In the place of YARN, a job tracker and task tracker was present, which help in the execution of application or jobs.
Flexibility	Yarn is more isolated and scalable.	Less scalable as compared to Yarn.
Daemons	The Yarn has a Name node, Data node, Secondary name node, Resource manager, and Node manager.	Map Reduce has a Name node, Data node, Secondary Name node, Job tracker, and Task tracker.
Limitation	There is no concept of a single point of failure in YARN because it has multiple Masters, so if one got failed, another master will pick it up and resume the execution.	Single point of failure, low resource utilization(Max of 4200 clusters by YAHOO), and less scalability when compared to Yarn.
Size	By default, the size of a data node in Yarn is 128MB.	By default, the size of a data node in MapReduce is 64MB.

BIG DATA TECHNOLOGIES

Map Reduce program for sorting text data

In my [last post](#) i wrote about sorting files in linux. decently large files (in the tens of gb's) can be sorted fairly quickly using that approach. but what if your files are already in hdfs, or ar hundreds of gb's in size or larger? in this case it makes sense to use mapreduce and leverage your cluster resources to sort your data in parallel.

mapreduce should be thought of as a ubiquitous sorting tool, since by design it sorts all the map output records (using the map output keys), so that all the records that reach a single reducer are sorted. the diagram below shows the internals of how the shuffle phase works in mapreduce.



given that MapReduce already performs sorting between the map and reduce phases, then sorting files can be accomplished with an identity function (one where the inputs to the map and reduce phases are emitted directly). this is in fact what the *sort* example that is bundled with Hadoop does. you can look at the [how the example code works](#) by examining the [org.apache.hadoop.examples.sort](#) class. to use this example code to sort text files in hadoop, you would use it as follows:

```
shell$ export hadoop_home=/usr/lib/hadoop
shell$ $hadoop_home/bin/hadoop jar $hadoop_home/hadoop-examples.jar sort \
  -informat org.apache.hadoop.mapred.keyvaluetextinputformat \
  -outformat org.apache.hadoop.mapred.textoutputformat \
  -outkey org.apache.hadoop.io.text \
  -outvalue org.apache.hadoop.io.text \
  /hdfs/path/to/input \
  /hdfs/path/to/output
```

BIG DATA TECHNOLOGIES

this works well, but it doesn't offer some of the features that i commonly rely upon in linux's sort, such as sorting on a specific column, and case-insensitive sorts.

Testing and Debugging map Reduce Application **Debugging Map reduce locally**

Test a map/reduce script on demand, you should split the script into entry point level sections. Use the sections to form unit tests. Each section should function as an entry point script that can be executed without external dependencies. If passing in modules, make sure that you use the require function and absolute paths.

To test map or reduce stages, you will need to create a mock context that seeds values and provides the dependent objects and parameters. Then, check the states, behavior, inputs, and outputs of map/reduce functions using assertion statements. Note that the `getInputData` stage does not take parameters, so it will not require mock context and can be tested more conventionally.

To test a summarize stage, if it contains logic operating on a final set of data that has no return, use assertions and logs to gather information.

To assist development of your unit test, download the Suite Script 2.x API files, specifically those representing the [map Context](#), reduce Context, and summary Context objects. These files can act as a schema for the properties and methods you want to test or mock. To access the files, do the following:

1. From NetSuite, select *Documents > Files > File Cabinet*.
2. Select Suite Script 2.0 API to download the zip folder.
3. Extract the MapReduce Context and MapReduce Summary .js files.

Performing unit testing for map reduce applications

In order to make sure that your code is correct, you need to Unit test your code first. And like you unit test your Java code using JUnit testing framework, the same can be done using MRUnit to test MapReduce Jobs.

MRUnit is built on top of JUnit framework. So we will use the JUnit classes to implement unit test code for MapReduce. If you are familiar with JUnits then you will find unit testing for MapReduce jobs also follows the same pattern.

I will now discuss the template that can be used for writing any unit test for MapReduce job.

To Unit test MapReduce jobs:

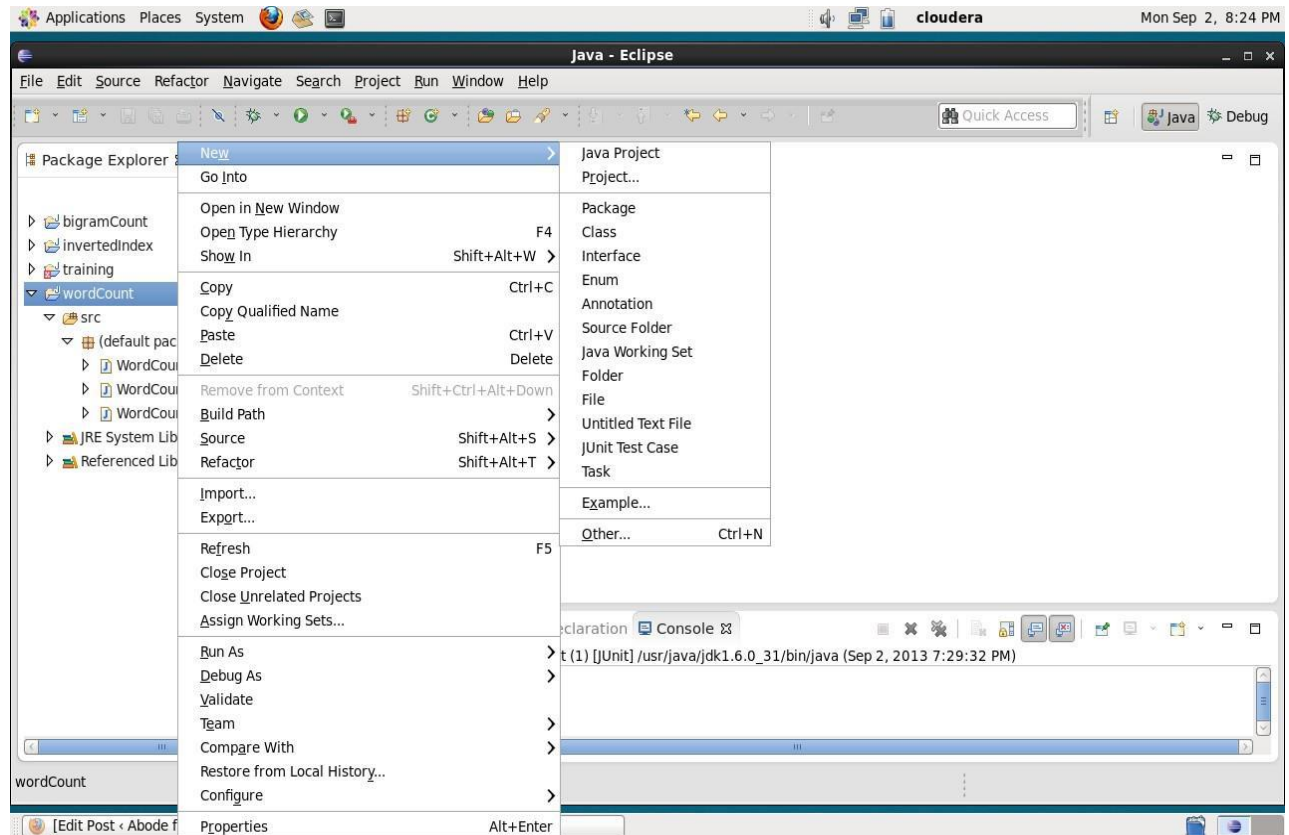
1. Create a new test class to the existing project
2. Add the mrunit jar file to build path
3. Declare the drivers
4. Write a method for initializations & environment setup
5. Write a method to test mapper

BIG DATA TECHNOLOGIES

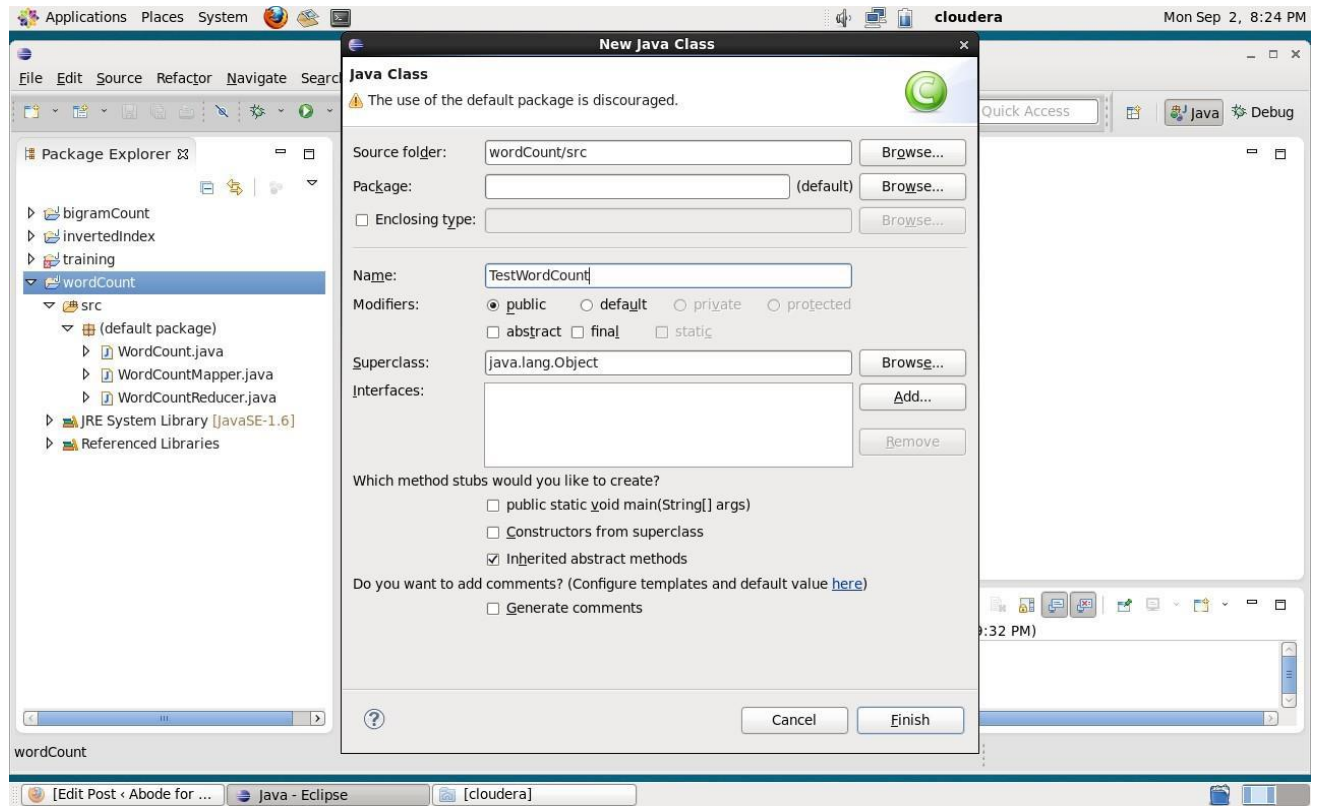
6. Write a method to test reducer
7. Write a method to test the whole MapReduce job
8. Run the test

Create a new test class to the existing project

I'll use the [WordCount example](#) to demonstrate unit testing. First create a new class with the name "TestWordCount" in the existing wordCount project.



BIG DATA TECHNOLOGIES



BIG DATA TECHNOLOGIES

UNIT-V

Introduction of HIVE

No one can better explain what Hive in Hadoop is than : data warehouse software facilitates reading, writing, and managing large datasets residing in distributed storage using SQL. The structure can be projected onto data already in storage."

In other words, Hive is an open-source system that processes structured data in Hadoop, residing on top of the latter for summarizing Big Data, as well as facilitating analysis and queries.

Now that we have investigated what is Hive in Hadoop, let's look at the features and characteristics.

Architecture of Hive

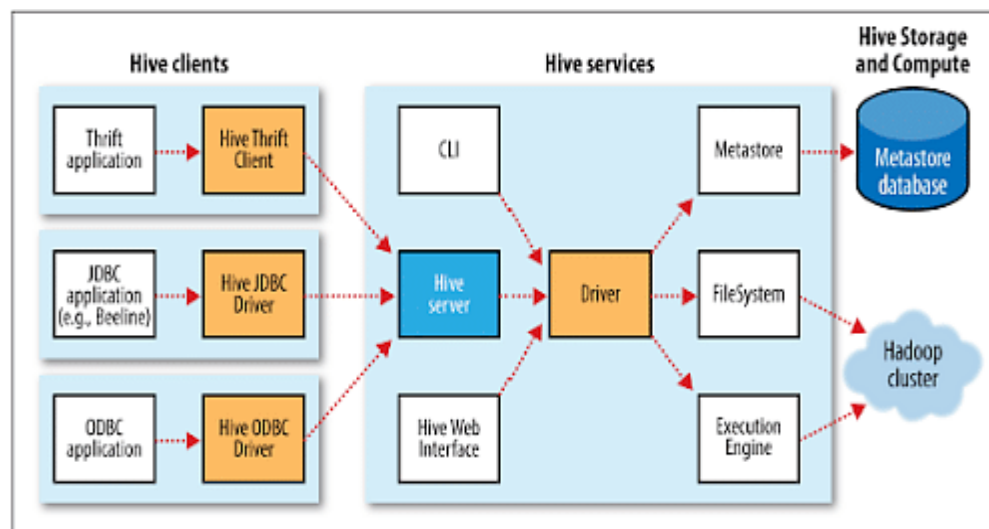


Figure 17-1. Hive architecture

[Source.](#)

Hive chiefly consists of three core parts:

- Hive Clients: Hive offers a variety of drivers designed for communication with different applications. For example, Hive provides Thrift clients for Thrift-based applications.

BIG DATA TECHNOLOGIES

These clients and drivers then communicate with the Hive server, which falls under Hive services.

- **Hive Services:** Hive services perform client interactions with Hive. For example, if a client wants to perform a query, it must talk with Hive services.
- **Hive Storage and Computing:** Hive services such as file system, job client, and meta store then communicates with Hive storage and stores things like metadata table information and query results.

What is Hive in Hadoop?

No one can better explain what Hive in Hadoop is than [the creators of Hive](#) themselves: "The Apache Hive™ data warehouse software facilitates reading, writing, and managing large datasets residing in distributed storage using SQL. The structure can be projected onto data already in storage."

In other words, Hive is an open-source system that processes structured data in Hadoop, residing on top of the latter for summarizing Big Data, as well as facilitating analysis and queries.

Now that we have investigated what is Hive in Hadoop, let's look at the features and characteristics.

BIG DATA TECHNOLOGIES

Aírchteúí of Hive

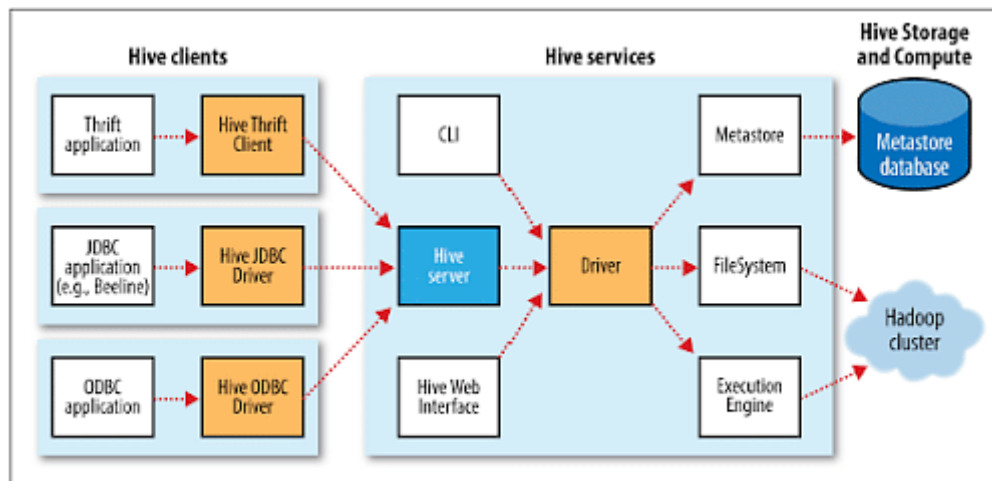


Figure 17-1. Hive architecture

[Souíce.](#)

Hive chiefly consists of three core parts:

- **Hive Clients:** Hive offers a variety of drivers designed for communication with different applications. For example, Hive provides Thrift clients for Thrift-based applications. These clients and drivers then communicate with the Hive server, which falls under Hive services.
- **Hive Services:** Hive services perform client interactions with Hive. For example, if a client wants to perform a query, it must talk with Hive services.
- **Hive Storage and Computing:** Hive services such as file system, job client, and meta store then communicates with Hive storage and stores things like metadata table information and query results.

Hive's Features

These are Hive's chief characteristics:

- Hive is designed for querying and managing only structured data stored in tables
- Hive is scalable, fast, and uses familiar concepts
- Schema gets stored in a database, while processed data goes into a Hadoop Distributed File System (HDFS)

BIG DATA TECHNOLOGIES

- Tables and databases get created first; then data gets loaded into the proper tables
- Hive supports four file formats: ORC, SEQUENCEFILE, RCFILE (Record Columnar File), and TEXTFILE
- Hive uses an SQL-inspired language, sparing the user from dealing with the complexity of MapReduce programming. It makes learning more accessible by utilizing familiar concepts found in relational databases, such as columns, tables, rows, and schema, etc.
- The most significant difference between the Hive Query Language (HQL) and SQL is that Hive executes queries on Hadoop's infrastructure instead of on a traditional database
- Since Hadoop's programming works on flat files, Hive uses directory structures to "partition" data, improving performance on specific queries
- Hive supports partition and buckets for fast and simple data retrieval
- Hive supports custom user-defined functions (UDF) for tasks like data cleansing and filtering. Hive UDFs can be defined according to programmers' requirements

Limitations of Hive

Of course, no resource is perfect, and Hive has some limitations. They are:

- Hive doesn't support OLTP. Hive supports Online Analytical Processing (OLAP), but not Online Transaction Processing (OLTP).
- It doesn't support subqueries.
- It has a high latency.
- Hive tables don't support delete or update operations.

How Data Flows in the Hive?

1. The [data analyst](#) executes a query with the User Interface (UI).

BIG DATA TECHNOLOGIES

2. The driver interacts with the query compiler to retrieve the plan, which consists of the query execution process and metadata information. The driver also parses the query to check syntax and requirements.
3. The compiler creates the job plan (metadata) to be executed and communicates with the metastore to retrieve a metadata request.
4. The metastore sends the metadata information back to the compiler
5. The compiler relays the proposed query execution plan to the driver.
6. The driver sends the execution plans to the execution engine.
7. The execution engine (EE) processes the query by acting as a bridge between the Hive and Hadoop. The job process executes in MapReduce. The execution engine sends the job to the JobTracker, found in the Name node, and assigns it to the TaskTracker, in the Data node. While this is happening, the execution engine executes metadata operations with the metastore.
8. The results are retrieved from the data nodes.
9. The results are sent to the execution engine, which, in turn, sends the results back to the driver and the front end (UI).

Since we have gone on at length about what Hive is, we should also touch on what Hive is not:

- Hive isn't a language for row-level updates and real-time queries
- Hive isn't a relational database
- Hive isn't a design for Online Transaction Processing

HIVE DDL & HIVE Data Manipulation

Create Database Statement

A database in Hive is a namespace or a collection of tables.

1. `hive> CREATE SCHEMA userdb;`
2. `hive> SHOW DATABASES;`

Drop database

1. `hive> DROP DATABASE IF EXISTS userdb;`

BIG DATA TECHNOLOGIES

Creating Hive Tables

Create a table called Sonoo with two columns, the first being an integer and the other a string.

1. `hive> CREATE TABLE Sonoo(foo INT, bar STRING);`

Create a table called HIVE_TABLE with two columns and a partition column called ds. The partition column is a virtual column. It is not part of the data itself but is derived from the partition that a particular dataset is loaded into. By default, tables are assumed to be of text input format and the delimiters are assumed to be ^A(ctrl-a).

1. `hive> CREATE TABLE HIVE_TABLE (foo INT, bar STRING) PARTITIONED BY (ds STRING);`

Browse the table

1. `hive> Show tables;`

Altering and Dropping Tables

1. `hive> ALTER TABLE Sonoo RENAME TO Kafka;`
2. `hive> ALTER TABLE Kafka ADD COLUMNS (col INT);`
3. `hive> ALTER TABLE HIVE_TABLE ADD COLUMNS (col1 INT COMMENT 'a comment');`
4. `hive> ALTER TABLE HIVE_TABLE REPLACE COLUMNS (col2 INT, weight STRING, baz INT COMMENT 'baz replaces new_col1');`

To understand the Hive DML commands, let's see the employee and employee department table first.

Employee			Employee Department	
EMP ID	Emp Name	Address	Emp ID	Department
1	Rose	US	1	IT
2	Fred	US	2	IT
3	Jess	In	3	Eng
4	Frey	Th	4	Admin

LOAD DATA

1. `hive> LOAD DATA LOCAL INPATH './usr/Desktop/kv1.txt' OVERWRITE INTO TABLE Employee;`

BIG DATA TECHNOLOGIES

SELECTS and FILTERS

1. hive> SELECT E.EMP_ID FROM Employee E WHERE E.Address='US'

Data Retrieval Queries

Hive Query language (HiveQL) provides SQL type environment in Hive to work with tables, databases, queries.

We can have a different type of Clauses associated with Hive to perform different type data manipulations and querying. For better connectivity with different nodes outside the environment. HIVE provide JDBC connectivity as well.

Hive queries provides the following features:

- Data modeling such as Creation of databases, tables, etc.
- ETL functionalities such as Extraction, Transformation, and Loading data into tables
- Joins to merge different data tables
- User specific custom scripts for ease of code
- Faster querying tool on top of Hadoop

In this article, you will learn-

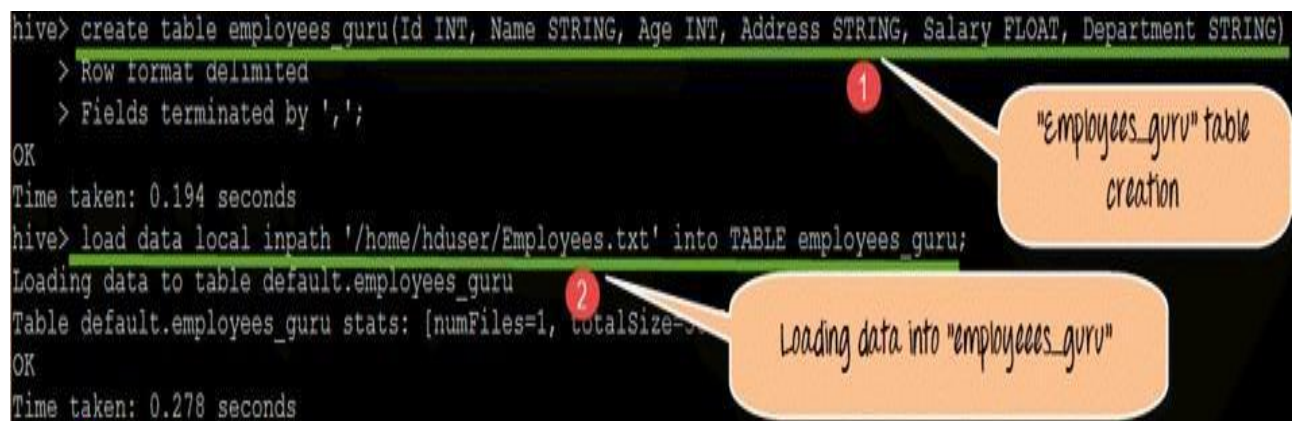
- [Order by query](#)
- [Group by query](#)
- [Sort by](#)
- [Cluster By](#)
- [Distribute By](#)

Creating Table in Hive

Before initiating with our main topic for this tutorial, first we will create a table to use it as references for the following tutorial.

Here in this tutorial, we are going to create table “employees_guru” with 6 columns.

```
hive> create table employees_guru(Id INT, Name STRING, Age INT, Address STRING, Salary FLOAT, Department STRING)
> Row format delimited
> Fields terminated by ',';
OK
Time taken: 0.194 seconds
hive> load data local inpath '/home/hduser/Employees.txt' into TABLE employees_guru;
Loading data to table default.employees_guru
Table default.employees_guru stats: [numFiles=1, totalSize=3]
OK
Time taken: 0.278 seconds
```



1 "employees_guru" table creation

2 Loading data into "employees_guru"

BIG DATA TECHNOLOGIES

From the above screen shot,

1. We are creating table “employees_guru” with 6 column values such as Id, Name, Age, Address, Salary, Department, which belongs to the employees present in organization “guru.”
2. Here in this step we are loading data into employees_guru table. The data that we are going to load will be placed under Employees.txt file

Order by query:

The ORDER BY syntax in HiveQL is similar to the syntax of ORDER BY in SQL language.

Order by is the clause we use with “SELECT” statement in Hive queries, which helps sort data. Order by clause use columns on Hive tables for sorting particular column values mentioned with Order by. For whatever the column name we are defining the order by clause the query will selects and display results by ascending or descending order the particular column values.

If the mentioned order by field is a string, then it will display the result in lexicographical order. At the back end, it has to be passed on to a single reducer.

```
hive> SELECT * FROM employees_guru ORDER BY Department;
Query ID = hduser_20151117170229_6e8af657-126b-470f-8b88
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined by Job: 1
In order to change the amount of memory per reducer, set hive.exec.reducers.bytesinmemory=<number> in the conf
set hive.exec.reducers.bytesinmemory=134217728
In order to limit the maximum number of reducers, set hive.exec.reducers.max=<number> in the conf
set hive.exec.reducers.max=1
In order to set a constant number of reducers:
set mapred.reduce.tasks=<number>
Starting Job = job_201511171701_0001, Tracking URL = http://localhost:8020/jobdetails.jsp?jobid=job_201511171701_0001
Kill Command = /usr/local/hadoop-1.2.1/libexec/bin/hadoop kill job_201511171701_0001
Hadoop job information for Stage-1: number of mappers: 1
2015-11-17 17:02:39,820 Stage-1 map = 0%, reduce = 0%
2015-11-17 17:02:44,008 Stage-1 map = 100%, reduce = 0%
2015-11-17 17:02:52,078 Stage-1 map = 100%, reduce = 33%
2015-11-17 17:02:53,085 Stage-1 map = 100%, reduce = 100%
MapReduce Total time: 2 seconds 40 msec
Ended Job = job_201511171701_0001
MapReduce Jobs finished: SUCCESS
Stage-Stage-1: Map: 1 Red: 1 Cumulative CPU: 2.04 sec
FS Write: 380 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 40 msec
OK
103 Animesh 26 Bangalore 25000.0 ADMIN
104 Anirudh 27 bangalore 27000.0 ADMIN
106 Ramesh 30 Goa 24000.0 FINANCE
101 Rajesh 27 Bangalore 20000.0 HR
102 Rajiv 28 Delhi 30000.0 HR
107 Sravanthi 32 Chennai 20000.0 IT
108 Sravan 31 Mumbai 60000.0 IT
109 Suresh 32 Kolkata 20000.0 IT
110 Ravi 28 bangalore 20000.0 IT
105 Santosh 33 bangalore 25000.0 PR
111 Syam 33 bangalore 25000.0 PR
Time taken: 25.224 seconds, Fetched: 11 row(s)
```

order by query on "employees_guru" table

order by query output

From the Above screen shot, we can observe the following

1. It is the query that performing on the “employees_guru” table with the ORDER BY clause with Department as defined ORDER BY column name.”Department” is String so it will display results based on lexicographical order.
2. This is actual output for the query. If we observe it properly, we can see that it get results displayed based on Department column such as ADMIN, Finance and so on in orderQuery to be perform.

BIG DATA TECHNOLOGIES

Query :

```
SELECT * FROM employees_guru ORDER BY Department;
```

Group by query:

Group by clause use columns on Hive tables for grouping particular column values mentioned with the group by. For whatever the column name we are defining a “groupby” clause the query will selects and display results by grouping the particular column values.

For example, in the below screen shot it’s going to display the total count of employees present in each department. Here we have “Department” as Group by value.

```
hive> SELECT Department, count(*) FROM employees_guru GROUP BY Department;
Query ID = nduser_20151105155307_1574ca2b-866e-437a-8d14-637e02b7e513
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified, assuming 1 per mapper (can change with -j <number>) size: 1
In order to change the average size of the reducers:
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapred.reduce.tasks=<number>
Starting Job = job_201511051442_0005, Tracking URL = http://localhost:5003
Kill Command = /usr/local/hadoop-1.2.1/libexec/./bin/hadoop job -kill job_201511051442_0005
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2015-11-05 15:53:19,229 Stage-1 map = 0%, reduce = 0%
2015-11-05 15:53:21,235 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 1.13 sec
2015-11-05 15:53:28,277 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 2.26 sec
2015-11-05 15:53:29,281 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 3.39 sec
MapReduce Total cumulative CPU time: 2 seconds 130 msec
Ended Job = job_201511051442_0005
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 3.39 sec HDFS Read: 7 B HDFS Write: 0 B
Total MapReduce CPU Time Spent: 2 seconds 130 msec
OK
ADMIN      2
FINANCE    1
HR          2
IT          4
PR          2
Time taken: 23.057 seconds, Fetched: 5 row(s)
```

Groupby query on "employees_guru"

Group by query output

From the above screenshot, we will observe the following

1. It is the query that is performed on the “employees_guru” table with the GROUP BY clause with Department as defined GROUP BY column name.
2. The output showing here is the department name, and the employees count in different departments. Here all the employees belong to the specific department is grouped by and displayed in the results. So the result is department name with the total number of employees present in each department.

Query:

```
SELECT Department, count(*) FROM employees_guru GROUP BY Department;
```


BIG DATA TECHNOLOGIES

Sort by:

Sort by clause performs on column names of Hive tables to sort the output. We can mention DESC for sorting the order in descending order and mention ASC for Ascending order of the sort.

In this sort by it will sort the rows before feeding to the reducer. Always sort by depends on column types.

For instance, if column types are numeric it will sort in numeric order if the columns types are string it will sort in lexicographical order.

```
hive> Select * from employees_guru SORT BY id DESC;
Query ID = hdus120151105164027_55bf2f64-6f5b-4764-b94e-e03f
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified, assuming 1 output data
In order to change the average number of reducers per node:
  set hive.exec.reducers.bytespernode=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapred.reduce.tasks=<number>
Starting Job = job_201511051442_0007, Tracking URL = http://1
Kill Command = /usr/local/hadoop-1.2.1/libexec/./bin/hadoop
Hadoop job information for Stage-1: Map: 1 Reducers: 1; num
2015-11-05 16:40:34,093 Stage-1-Map-1: 0%
2015-11-05 16:40:36,098 Stage-1-Map-1: 0%
2015-11-05 16:40:44,145 Stage-1-Map-1: 100%, C
MapReduce Total cumulative CPU time: 1.62 seconds 620 msec
Ended Job = job_201511051442_0007
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduc: 1 Cumulative CPU: 1.62 sec
Total MapReduce CPU Time Spent: 1 seconds 620 msec
OK
111      Syam      33      bangalore      25000.0 PR
110      Ravi      28      bangalore      20000.0 IT
109      Suresh   32      Kolkata 20000.0 IT
108      Sravan   31      Mumbai  60000.0 IT
107      Sravanthi 32      Chennai 20000.0 IT
106      Ramesh   30      Goa      24000.0 FINANCE
105      Santosh  33      bangalore      25000.0 PR
104      Anirudh  27      bangalore      27000.0 ADMIN
103      Animesh  26      Bangalore      25000.0 ADMIN
102      Rajiv    28      Delhi   30000.0 HR
101      Rajesh   27      Bangalore      20000.0 HR
Time taken: 18.088 seconds, Fetched: 11 row(s)
```

sort by query

sort by output on "employees_guru" table

From the above screen shot we can observe the following:

1. It is the query that performing on the table “employees_guru” with the SORT BY clause with “id” as define SORT BY column name. We used keyword DESC.
2. So the output displayed will be in descending order of “id”.

BIG DATA TECHNOLOGIES

Query:

```
SELECT * from employees_guru SORT BY Id DESC;
```

Cluster By:

Cluster By used as an alternative for both Distribute BY and Sort BY clauses in Hive-QL.

Cluster BY clause used on tables present in Hive. Hive uses the columns in Cluster by to distribute the rows among reducers. Cluster BY columns will go to the multiple reducers.

- It ensures sorting orders of values present in multiple reducers

For example, Cluster By clause mentioned on the Id column name of the table employees_guru table. The output when executing this query will give results to multiple reducers at the back end. But as front end it is an alternative clause for both Sort By and Distribute By.

This is actually back end process when we perform a query with sort by, group by, and cluster by in terms of Map reduce framework. So if we want to store results into multiple reducers, we go with Cluster By.

Using JOINS in HIVE

JOIN is a clause that is used for combining specific fields from two tables by using values common to each one. It is used to combine records from two or more tables in the database.

Syntax

join_table:

```
table_reference JOIN table_factor [join_condition]
| table_reference {LEFT|RIGHT|FULL} [OUTER] JOIN table_reference
join_condition
| table_reference LEFT SEMI JOIN table_reference join_condition
| table_reference CROSS JOIN table_reference [join_condition]
```

Example

We will use the following two tables in this chapter. Consider the following table named CUSTOMERS..

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Consider another table ORDERS as follows:

BIG DATA TECHNOLOGIES

OID	DATE	CUSTOMER_ID	AMOUNT
102	2009-10-08 00:00:00	3	3000
100	2009-10-08 00:00:00	3	1500
101	2009-11-20 00:00:00	2	1560
103	2008-05-20 00:00:00	4	2060

There are different types of joins given as follows:

- JOIN
- LEFT OUTER JOIN
- RIGHT OUTER JOIN
- FULL OUTER JOIN

JOIN

JOIN clause is used to combine and retrieve the records from multiple tables. JOIN is same as OUTER JOIN in SQL. A JOIN condition is to be raised using the primary keys and foreign keys of the tables.

The following query executes JOIN on the CUSTOMER and ORDER tables, and retrieves the records:

```
hive> SELECT c.ID, c.NAME, c.AGE, o.AMOUNT
FROM CUSTOMERS c JOIN ORDERS o
ON (c.ID = o.CUSTOMER_ID);
```

On successful execution of the query, you get to see the following response:

ID	NAME	AGE	AMOUNT
3	kaushik	23	3000
3	kaushik	23	1500
2	Khilan	25	1560
4	Chaitali	25	2060

LEFT OUTER JOIN

The HiveQL LEFT OUTER JOIN returns all the rows from the left table, even if there are no matches in the right table. This means, if the ON clause matches 0 (zero) records in the right table, the JOIN still returns a row in the result, but with NULL in each column from the right table.

A LEFT JOIN returns all the values from the left table, plus the matched values from the right table, or NULL in case of no matching JOIN predicate.

The following query demonstrates LEFT OUTER JOIN between CUSTOMER and ORDER tables:

```
hive> SELECT c.ID, c.NAME, o.AMOUNT, o.DATE
FROM CUSTOMERS c
LEFT OUTER JOIN ORDERS o
```

BIG DATA TECHNOLOGIES

```
ON (c.ID = o.CUSTOMER_ID);
```

On successful execution of the query, you get to see the following response:

```
+-----+-----+-----+-----+
| ID | NAME   | AMOUNT | DATE           |
+-----+-----+-----+-----+
| 1 | Ramesh | NULL   | NULL           |
| 2 | Khilan | 1560   | 2009-11-20 00:00:00 |
| 3 | kaushik | 3000   | 2009-10-08 00:00:00 |
| 3 | kaushik | 1500   | 2009-10-08 00:00:00 |
| 4 | Chaitali | 2060   | 2008-05-20 00:00:00 |
| 5 | Hardik | NULL   | NULL           |
| 6 | Komal  | NULL   | NULL           |
| 7 | Muffy  | NULL   | NULL           |
+-----+-----+-----+-----+
```

RIGHT OUTER JOIN

The HiveQL RIGHT OUTER JOIN returns all the rows from the right table, even if there are no matches in the left table. If the ON clause matches 0 (zero) records in the left table, the JOIN still returns a row in the result, but with NULL in each column from the left table.

A RIGHT JOIN returns all the values from the right table, plus the matched values from the left table, or NULL in case of no matching join predicate.

The following query demonstrates RIGHT OUTER JOIN between the CUSTOMER and ORDER tables.

```
hive> SELECT c.ID, c.NAME, o.AMOUNT, o.DATE FROM CUSTOMERS c
RIGHT OUTER JOIN ORDERS o ON (c.ID = o.CUSTOMER_ID);
```

On successful execution of the query, you get to see the following response:

```
+-----+-----+-----+-----+
| ID | NAME   | AMOUNT | DATE           |
+-----+-----+-----+-----+
| 3 | kaushik | 3000   | 2009-10-08 00:00:00 |
| 3 | kaushik | 1500   | 2009-10-08 00:00:00 |
| 2 | Khilan  | 1560   | 2009-11-20 00:00:00 |
| 4 | Chaitali | 2060   | 2008-05-20 00:00:00 |
+-----+-----+-----+-----+
```

FULL OUTER JOIN

The HiveQL FULL OUTER JOIN combines the records of both the left and the right outer tables that fulfil the JOIN condition. The joined table contains either all the records from both the tables, or fills in NULL values for missing matches on either side.

The following query demonstrates FULL OUTER JOIN between CUSTOMER and ORDER tables:

```
hive> SELECT c.ID, c.NAME, o.AMOUNT, o.DATE
FROM CUSTOMERS c
FULL OUTER JOIN ORDERS o
```

BIG DATA TECHNOLOGIES

```
ON (c.ID = o.CUSTOMER_ID);
```

On successful execution of the query, you get to see the following response:

```
+-----+-----+-----+-----+
| ID | NAME | AMOUNT | DATE |
+-----+-----+-----+-----+
| 1 | Ramesh | NULL | NULL |
| 2 | Khilan | 1560 | 2009-11-20 00:00:00 |
| 3 | kaushik | 3000 | 2009-10-08 00:00:00 |
| 3 | kaushik | 1500 | 2009-10-08 00:00:00 |
| 4 | Chaitali | 2060 | 2008-05-20 00:00:00 |
| 5 | Hardik | NULL | NULL |
| 6 | Komal | NULL | NULL |
| 7 | Muffy | NULL | NULL |
| 3 | kaushik | 3000 | 2009-10-08 00:00:00 |
| 3 | kaushik | 1500 | 2009-10-08 00:00:00 |
| 2 | Khilan | 1560 | 2009-11-20 00:00:00 |
| 4 | Chaitali | 2060 | 2008-05-20 00:00:00 |
```

Introduction NOSQL Data models Schema

NoSQL Database is a non-relational Data Management System, that does not require a fixed schema. It avoids joins, and is easy to scale. The major purpose of using a NoSQL database is for distributed data stores with humongous data storage needs. NoSQL is used for Big data and real-time web apps. For example, companies like Twitter, Facebook and Google collect terabytes of user data every single day.

NoSQL database stands for “Not Only SQL” or “Not SQL.” Though a better term would be “NoREL”, NoSQL caught on. Carl Strozzi introduced the NoSQL concept in 1998.

Traditional RDBMS uses SQL syntax to store and retrieve data for further insights. Instead, a NoSQL database system encompasses a wide range of database technologies that can store structured, semi-structured, unstructured and polymorphic data. Let’s understand about NoSQL with a diagram in this NoSQL database tutorial:

Features of NoSQL

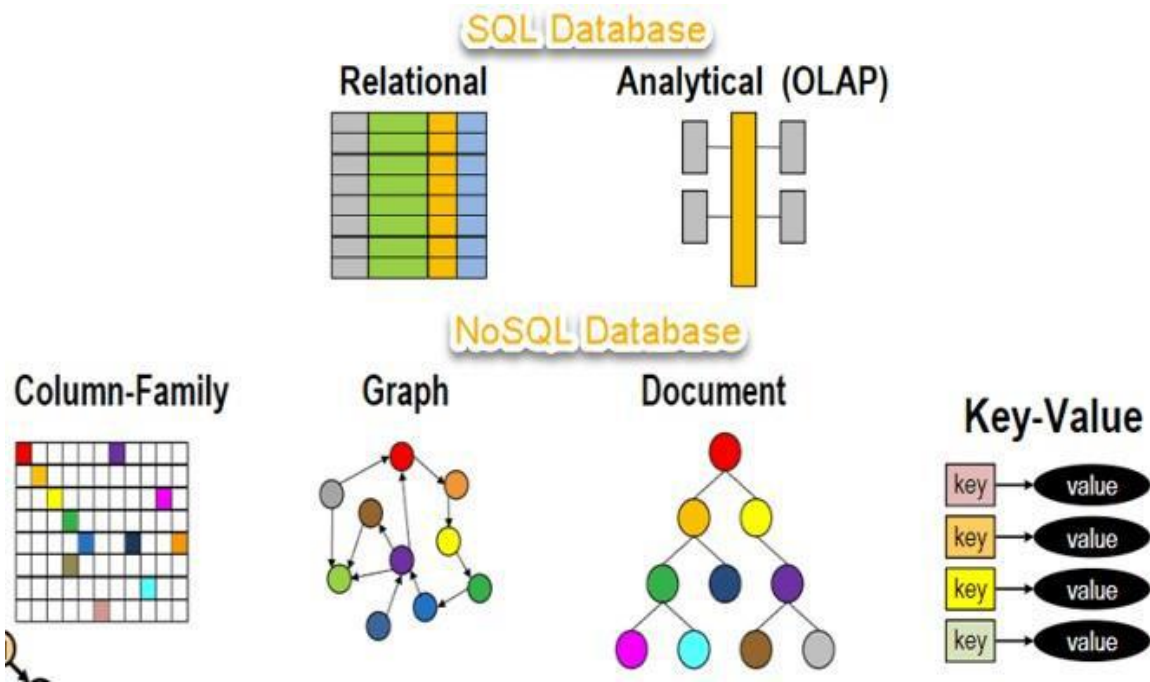
Non-relational

- 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

BIG DATA TECHNOLOGIES

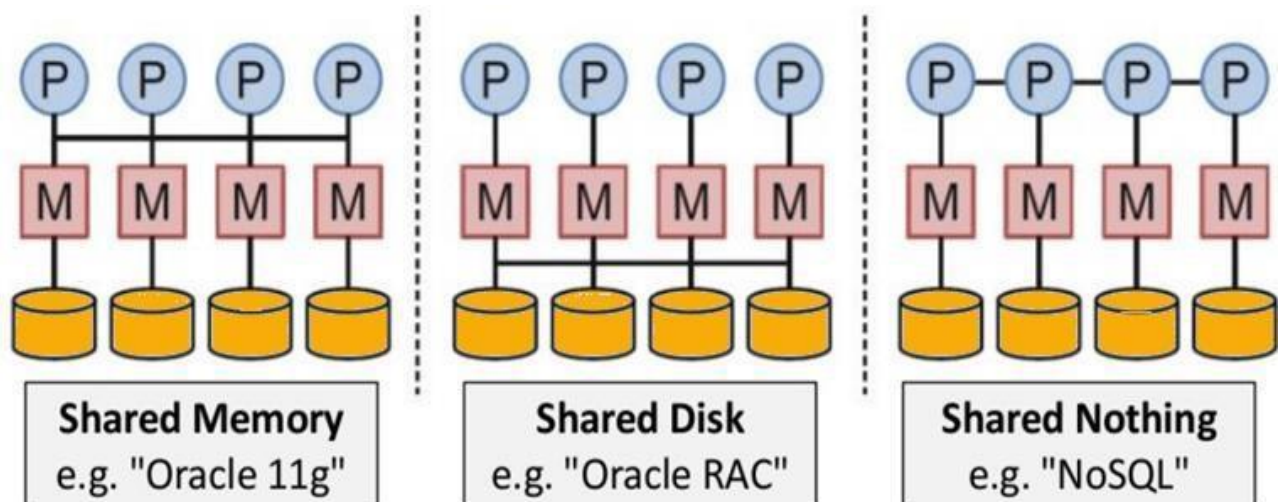


Simple API

- 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 NoSQL 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.



BIG DATA TECHNOLOGIES

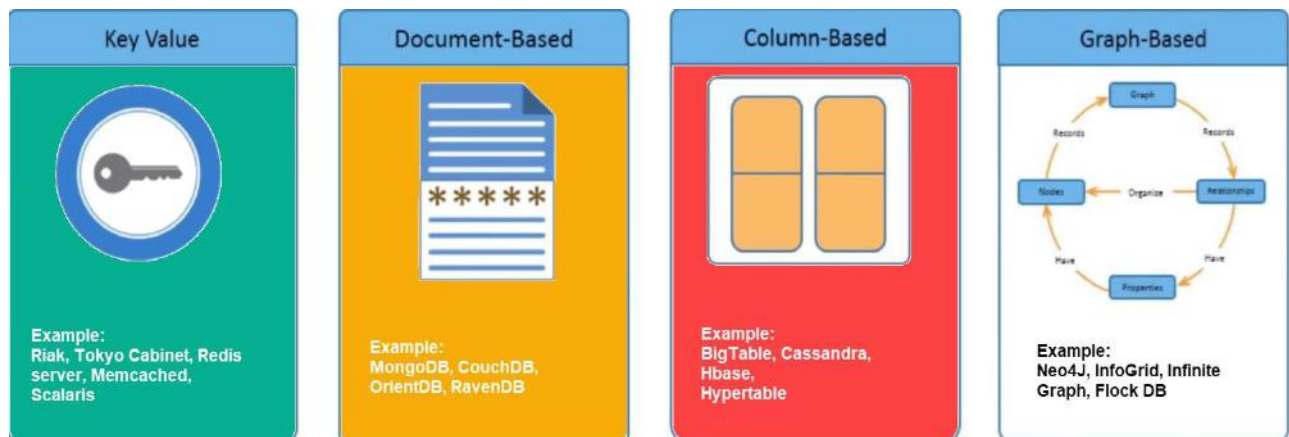
NoSQL is Shared Nothing.

Types of NoSQL Databases

NoSQL Databases are mainly categorized into four types: Key-value pair, Column-oriented, Graph-based and Document-oriented. Every category has its unique attributes and limitations. None of the above-specified database is better to solve all the problems. Users should select the database based on their product needs.

Types of NoSQL Databases:

- 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

BIG DATA TECHNOLOGIES

It is one of the most basic NoSQL database example. 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 NoSQL 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.

ColumnFamily			
Row Key	Column Name		
	Key	Key	Key
	Value	Value	Value
	Column Name		
	Key	Key	Key
	Value	Value	Value

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,

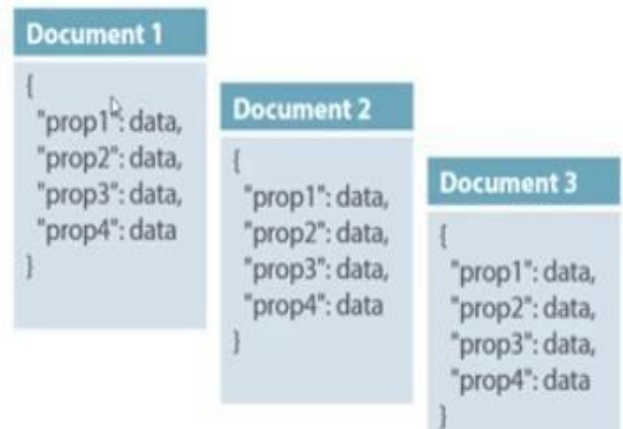
HBase, Cassandra, HBase, Hypertable are NoSQL query examples of column based database.

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.

BIG DATA TECHNOLOGIES

Col1	Col2	Col3	Col4
Data	Data	Data	Data
Data	Data	Data	Data
Data	Data	Data	Data



Relational Vs. Document

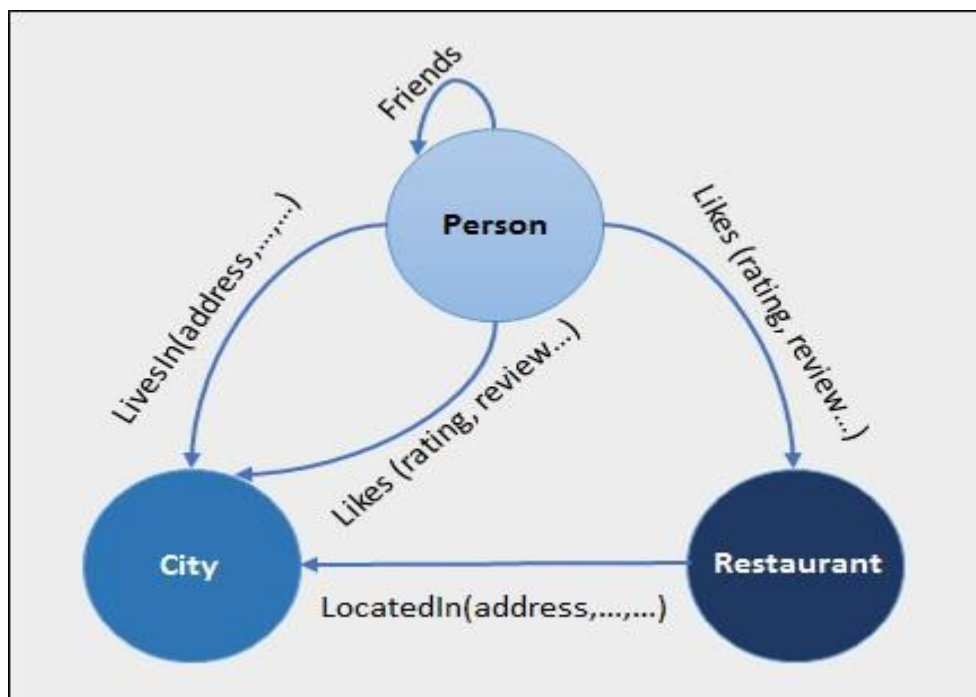
In this diagram on your left you can see we have rows and columns, and in the right, we have a document database which has a similar structure to JSON. Now for the relational database, you have to know what columns you have and so on. However, for a document database, you have data store like JSON object. 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.



BIG DATA TECHNOLOGIES

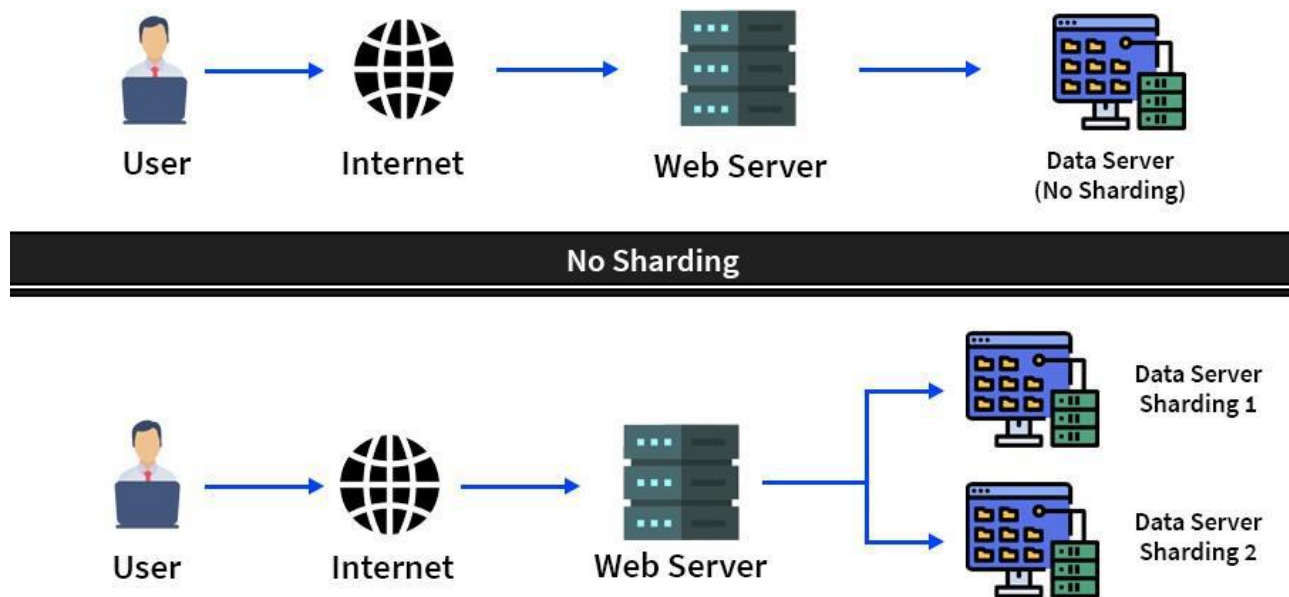
Compared to a relational database where tables are loosely connected, a Graph database is a multi-relational in nature. Traversing relationship is fast as they are already captured into the DB, and there is no need to calculate them.

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

Neo4J, Infinite Graph, Orient DB, Flock DB are some popular graph-based databases.

Distribution Models, Sharding

Take the example of *Pizza* (yes!!! your favorite food). You get the pizza in different slices and you share these slices with your friends. **Sharding** which is also known as **data partitioning** works on the same concept of sharing the Pizza slices. It is basically a database architecture pattern in which we split a large dataset into smaller chunks (logical shards) and we store/distribute these chunks in different machines/database nodes (physical shards). Each chunk/partition is known as a “**shard**” and each shard has the same database schema as the original database. We distribute the data in such a way that each row appears in exactly one shard. It’s a good mechanism to improve the **scalability** of an application.



Database shards are autonomous; they don't share any of the same data or computing resources. In some cases, though, it may make sense to replicate certain tables into each shard to serve as reference tables.

Advantages of Sharding

- **Solve Scalability Issue:** With a single database server architecture any application experience performance degradation when users start growing on that application. Reads and write queries become slower and the network bandwidth starts to saturate. At some point, you will be running out of disk space. Database sharding fixes all these issues by partitioning the data across multiple machines.
- **High Availability:** A problem with single server architecture is that if an outage happens then the entire application will be unavailable which is not good for a website with more number of users. This is not the case with a sharded database. If an outage

BIG DATA TECHNOLOGIES

happens in sharded architecture, then only some specific shards will be down. All the other shards will continue the operation and the entire application won't be unavailable for the users.

- **Speed Up Query Response Time:** When you submit a query in an application with a large monolithic database and have no sharded architecture, it takes more time to find the result. It has to search every row in the table and that slows down the response time for the query you have given. This doesn't happen in sharded architecture. In a sharded database a query has to go through fewer rows and you receive the response in less time.
- **More Write Bandwidth:** For many applications writing is a major bottleneck. With no master database serializing writes sharded architecture allows you to write in parallel and increase your write throughput.
- **Scaling Out:** Sharding a database facilitates *horizontal scaling*, known as *scaling out*. In horizontal scaling, you **add more machines** in the network and distribute the load on these machines for faster processing and response. This has many advantages. You can do more work simultaneously and you can handle high requests from the users, especially when writing data because there are parallel paths through your system. You can also load balance web servers that access shards over different network paths, which are processed by different CPUs, and use separate caches of RAM or disk IO paths to process work.

Disadvantages of Sharding

- **Adds Complexity in the System:** You need to be careful while implementing a proper sharded database architecture in an application. It's a complicated task and if it's not implemented properly then you may lose the data or get corrupted tables in your database. You also need to manage the data from multiple shard locations instead of managing and accessing it from a single entry point. This may affect the workflow of your team which can be potentially disruptive to some teams.
- **Rebalancing Data:** In a sharded database architecture, sometimes shards become unbalanced (when a shard outgrows other shards). Consider an example that you have two shards of a database. One shard store the name of the customers begins with letter A through M. Another shard store the name of the customer begins with the letters N through Z. If there are so many users with the letter L then shard one will have more data than shard two. This will affect the performance (slow down) of the application and it will stall out for a significant portion of your users. The A-M shard will become unbalance and it will be known as *database hotspot*. To overcome this problem and to rebalance the data you need to do re-sharding for even data distribution. Moving data from one shard to another shard is not a good idea because it requires a lot of downtimes.
- **Joining Data From Multiple Shards is Expensive:** In a single database, joins can be performed easily to implement any functionalities. But in sharded architecture, you need to pull the data from different shards and you need to perform joins across multiple networked servers. You can't submit a single query to get the data from various shards. You need to submit **multiple queries** for each one of the shards, pull out the data, and join the data across the network. This is going to be a very expensive and time-consuming process. It adds **latency** to your system.
- **No Native Support:** Sharding is not natively supported by every database engine. For example, PostgreSQL doesn't include automatic sharding features, so there you have to do manual sharding. You need to follow the "roll-your-own" approach. It will be difficult for you to find the tips or documentation for sharding and troubleshoot the problem during the implementation of sharding.

Sharding Architectures

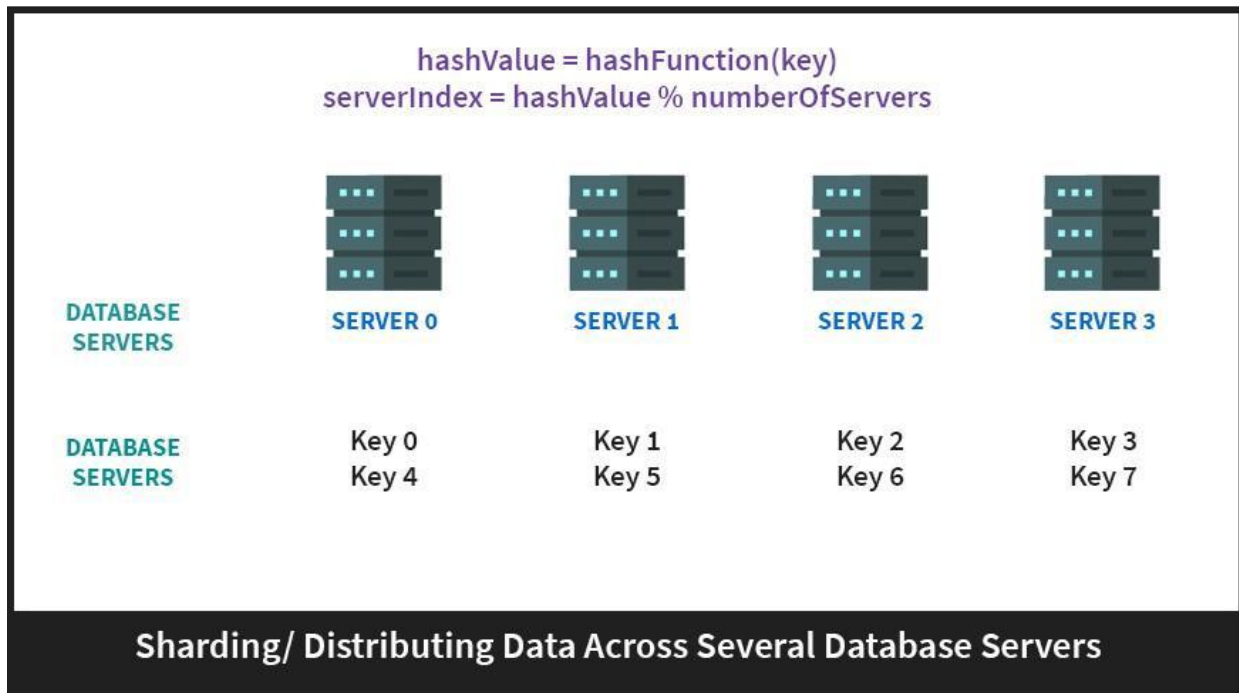
1. Key Based Sharding

This technique is also known as **hash-based** sharding. Here, we take the value of an entity such as customer ID, customer email, IP address of a client, zip code, etc and we use this value as an input

BIG DATA TECHNOLOGIES

of the **hash function**. This process generates a **hash value** which is used to determine which shard we need to use to store the data. We need to keep in mind that the values entered into the hash function should all come from the **same column** (shard key) just to ensure that data is placed in the correct order and in a consistent manner. Basically, shard keys act like a *primary key* or a unique identifier for individual rows.

Consider an example that you have 3 database servers and each request has an application id which is incremented by 1 every time a new application is registered. To determine which server data should be placed on, we perform a modulo operation on these applications id with the number 3. Then the remainder is used to identify the server to store our data.



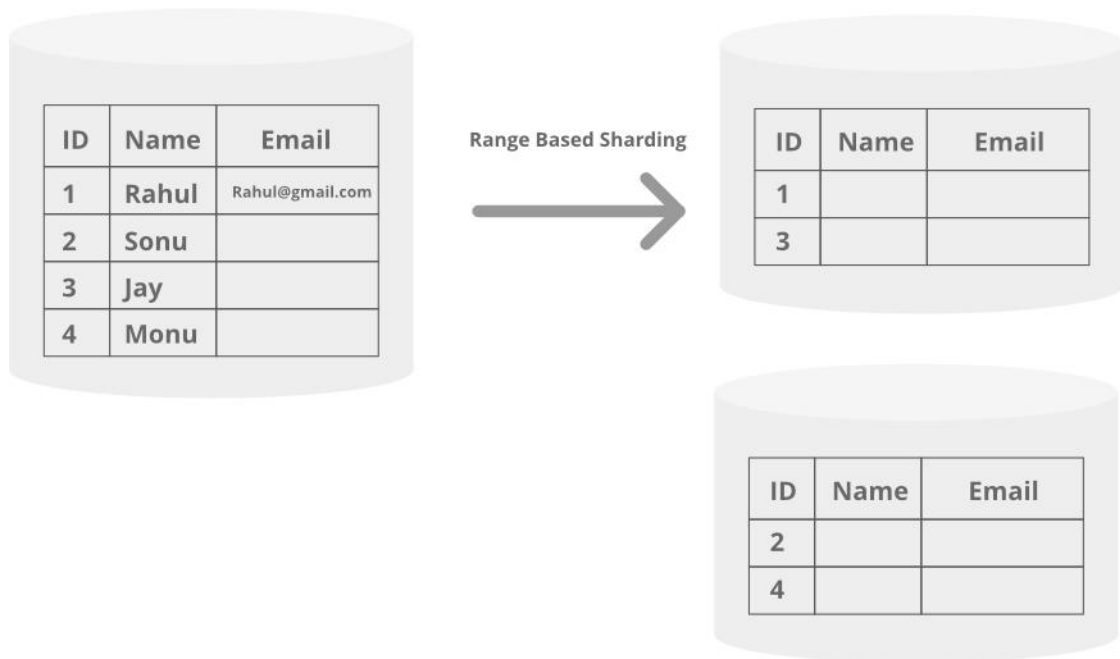
The downside of this method is **elastic load balancing** which means if you will try to add or remove the database servers dynamically it will be a difficult and expensive process. For example, in the above one if you will add 5 more servers then you need to add more corresponding hash values for the additional entries. Also, the majority of the existing keys need to be remapped to their new, correct hash value and then migrated to a new server. The hash function needs to be changed from modulo 3 to modulo 8. While the migration of data is in effect both the new and old hash functions won't be valid. During the migration, your application won't be able to service a large number of requests and you'll experience downtime for your application till the migration completes.

Note: *A shard shouldn't contain values that might change over time. It should be always static otherwise it will slow down the performance.*

2. Horizontal or Range Based Sharding

In this method, we split the data based on the **ranges** of a given value inherent in each entity. Let's say you have a database of your online customers' names and email information. You can split this information into two shards. In one shard you can keep the info of customers whose first name starts with A-P and in another shard, keep the information of the rest of the customers.

BIG DATA TECHNOLOGIES



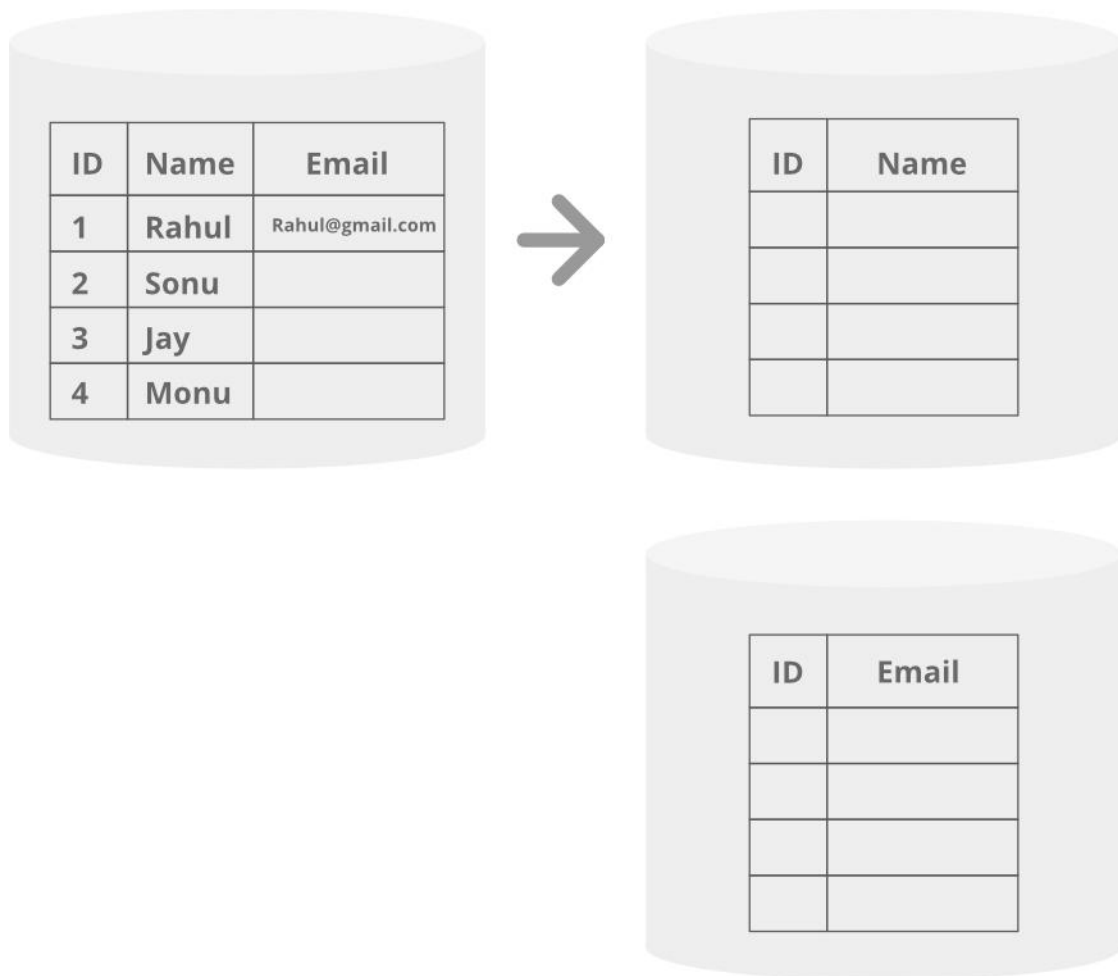
Range-based sharding is the simplest sharding method to implement. Every shard holds a different set of data but they all have the same schema as the original database. In this method, you just need to identify in which range your data falls, and then you can store the entry to the corresponding shard. This method is best suitable for storing non-static data (example: storing the contact info for students in a college.)

The drawback of this method is that the data may not be **evenly distributed** on shards. In the above example, you might have a lot of customers whose names fall into the category of A-P. In such cases, the first shard will have to take more load than the second one and it can become a system bottleneck.

3. Vertical Sharding

In this method, we split the entire column from the table and we put those columns into new distinct tables. Data is totally independent of one partition to the other ones. Also, each partition holds both distinct rows and columns. Take the example of Twitter features. We can split different features of an entity in different shards on different machines. On Twitter users might have a profile, number of followers, and some tweets posted by his/her own. We can place the user profiles on one shard, followers in the second shard, and tweets on a third shard.

BIG DATA TECHNOLOGIES



In this method, you can separate and handle the critical part (for example user profiles) non-critical part of your data (for example, blog posts) individually and build different replication and consistency models around it. This is one of the main advantages of this method.

The main drawback of this scheme is that to answer some queries you may have to combine the data from different shards which unnecessarily increases the development and operational complexity of the system. Also, if your application will grow later and you add some more features in it then you will have to further shard a feature-specific database across multiple servers.

4. Directory-Based Sharding

In this method, we create and maintain a **lookup service** or lookup table for the original database. Basically we use a **shard key** for lookup table and we do **mapping** for each entity that exists in the database. This way we keep track of which database shards hold which data.

BIG DATA TECHNOLOGIES

PRE - SHARDED TABLE

Shard Key

DELIVERY ZONE	FIRST NAME	LAST NAME
3	RAHUL	JAIN
1	AJAY	KUMAR
2	ANSHUL	GARG
4	SONU	SHARMA



Shard Key

DELIVERY ZONE	SHARD ID
1	S1
2	S2
3	S3
4	S4



S1

1	AJAY	KUMAR
---	------	-------

S2

2	ANSHUL	GARG
---	--------	------

S3

3	RAHUL	JAIN
---	-------	------

S4

4	SONU	SHARMA
---	------	--------